AP2PS 2011

The Third International Conference on Advances in P2P Systems

ISBN: 978-1-61208-173-1

November 20-25, 2011

Lisbon, Portugal

AP2PS 2011 Editors

Antonio Liotta, Eindhoven University of Technology, The Netherlands

Nikos Antonopoulos, University of Derby, UK

Giuseppe Di Fatta, The University of Reading, UK

Takahiro Hara, University of Osaka, Japan

Quang Hieu Vu, ETISALAT BT Innovation Center (EBTIC)/ Khalifa University, UAE

AP2PS 2011

Foreword

The Third International Conference on Advances in P2P Systems [AP2PS 2011], held between November 20 and 25, 2011 in Lisbon, Portugal, was a dedicated forum for academic researchers and industrial practitioners to present and discuss the latest architectures, protocols, applications and innovative ideas in the field of overlay and peer-to-peer networking.

Peer-to-peer systems have considerably evolved since their original conception, in the 90's. The idea of distributing files using the user's terminal as a relay has now been widely extended to embrace virtually any form of resource (e.g., computational and storage resources), data (e.g. files and real-time streams) and service (e.g., IP telephony, IP TV, collaboration).

Robustness, resilience and autonomic management are the key evolutionary step which makes them best fitted in dynamic, large-scale, decentralized environments. Peer-to-Peer systems provide network applications the opportunity to leap over the boundaries of the standardized and best effort protocols and the constraints of administrative domains.

We take here the opportunity to warmly thank all the members of the AP2PS 2011 Technical Program Committee, as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to AP2PS 2011. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the AP2PS 2011 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that AP2PS 2011 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of P2P systems.

We are convinced that the participants found the event useful and communications very open. We also hope the attendees enjoyed the historic charm of Lisbon, Portugal.

AP2PS 2011 Chairs:

Marco Aiello
Nick Antonopoulos
Giuseppe Di Fatta
Anders Fongen
Takahiro Hara
Yasushi Kambayashi
Antonio Liotta
Christoph Schuba
Roman Y. Shtykh
Quang Hieu Vu
Ouri Wolfson

AP2PS 2011

Committee

AP2PS General Chairs

Nick Antonopoulos, University of Derby, UK
Antonio Liotta, Eindhoven University of Technology, The Netherlands
Giuseppe Di Fatta, The University of Reading, UK

AP2PS Advisory Chairs

Marco Aiello, University of Groningen, The Netherlands
Takahiro Hara, University of Osaka, Japan
Ouri Wolfson, University of Illinois at Chicago, USA

AP2PS 2011 Industry Liaison Chair

Christoph Schuba, Oracle Corp., USA
Roman Y. Shtykh, Rakuten, Inc., Japan

AP2PS 2011 Research Chairs

Yasushi Kambayashi, Nippon Institute of Technology, Japan
Anders Fongen, Norwegian Defense Research Establishment, Norway
Quang Hieu Vu, ETISALAT BT Innovation Center (EBTIC)/ Khalifa University, UAE

AP2PS 2011 Technical Program Committee

Jemal H. Abawajy, Deakin University - Geelong, Australia
Marco Aiello, University of Groningen, The Netherlands
Nikos Antonopoulos, University of Surrey, UK
Farnoush Banaei-Kashani, University of Southern California, USA
Ataul Bari, University of Western Ontario, Canada
Andreas Berl, University of Passau, Germany
Dumitru Dan Burdescu, University of Craiova, Romania
Candido Caballero-Gil, University of La Laguna, Spain
Frances Brazier, Tup-Delft, The Netherlands
Ahmet Burak Can, Hacettepe University, Turkey
Juan-Carlos Cano, Universidad Politécnica de Valencia, Spain
Charalampos Chelmis, University of Southern California, USA
Chou Cheng-Fu, National Taiwan University, Taiwan
Giovanni Chiola, Università di Genova, Italia
Carmela Comito, University of Calabria - Rende, Italy
Noël Crespi, IT-ParisSud, France
Antonio Cuadra-Sanchez, Indra, Spain
Rosario De Chiara, Università degli Studi di Salerno - Fisciano, Italy
Giuseppe Di Fatta, The University of Reading, UK
Anne Doucet , University Pierre et Marie Curie (Paris VI), France
Nashwa Mamdouh El-Bendary, Arab Academy for Science, Technology, and Maritime Transport - Giza, Egypt

Thanasis G. Papaioannou, EPFL, Switzerland
Jens Myrup Pedersen , Aalborg University - Aalborg East, Denmark
Rubem Pereira, Liverpool John Moores University, UK
Jean-Marc Pierson, IRIT / Université Paul Sabatier - Toulouse, France
Thomas Risse, L3S Research Center, Germany
Tapani Ristaniemi, University of Jyväskylä, Finland
Claudia Lucia Roncancio, Grenoble INP/Ensimag, France
Tomas Sanchez Lopez, EADS Innovation Works, UK
Rossano Schifanella, University of Turin, Italy
Florence Sedes, IRIT, France
Patricia Serrano Alvarado, University of Nantes, France
Roman Y. Shtykh, Rakuten, Inc., Japan
Simone Silvestri, Sapienza University of Rome, Italy
Dora Souliou, National Technical University of Athens, Greece
Ahmad Tajuddin Bin Samsudin, Telekom Malaysia (Research & Development), Malaysia
Orazio Tomarchio, University of Catania, Italy
Bo (Rambo) Tan, University of Illinois at Urbana - Champaign, USA
Kurt Tutschku, Institute of Distributed and Multimedia Systems / University of Vienna, Austria
Miguel A. Vega-Rodríguez, University of Extremadura, Spain
Kevin Vella, University of Malta, Malta /University of Kent, UK
Wenjing Wang, Attila Technologies, USA
Ouri Wolfson, University of Illinois at Chicago, USA

Additional reviewers

Alexander Allan, University of Reading, UK
Piotr Szczurek, University of Illinois at Chicago, USA

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

SGR-Tree: a Skip Graph based R-Tree for multi-dimensional data indexing in Peer-to-Peer systems

Quang Hieu Vu
ETISALAT BT Innovation Center
Khalifa University of Science, Technology and Research, UAE
quang.vu@kustar.ac.ae

***Abstract*—In this paper, we propose SGR-Tree, an index structure for multi-dimensional data in Peer-to-Peer (P2P) systems. SGR-Tree is an R-Tree index structure constructed on top of a Skip Graph, a P2P overlay network. In SGR-Tree, each Skip Graph node corresponds to an R-Tree leaf node. However, different from R-Tree, SGR-Tree does not employ internal nodes for routing purpose. Instead, at each Skip Graph node, we virtually partition the whole system into non-overlapping regions, each of which is connected to the node via a neighbor node. For each region, the node keeps the minimum hyper-rectangle covering all hyper-rectangles, which are in charged by nodes falling in the region. In this way, when a node issues or receives a query, it simply sends or forwards the query to neighbor nodes whose minimum covering hyper-rectangle intersects with the search region. The main advantage of SGR-Tree is that it can avoid not only the bottleneck problem at the root node but also the high cost of maintaining internal R-Tree nodes, especially when the index structure is often changed. Nevertheless, we prove that SGR-Tree is still able to process multi-dimensional queries efficiently within a boundary of $logN$ steps, where N is the number of nodes in the system. We have done experiments to validate the practicability and efficiency of SGR-Tree.**

Keywords - *Multi-dimensional data indexing, P2P.*

I. INTRODUCTION

In the past decade, Peer-to-Peer (P2P) systems have received a lot of interests from both computer users and researchers. The main advantage of P2P systems is the capability of sharing resources so that large systems can be easily formed by low-cost computers instead of expensive servers. Since sharing data such as images, music files, and textual documents are often represented as multi-dimensional points in a multi-dimensional space, supporting multi-dimensional data indexing in P2P systems is extremely important.

Multi-dimensional data indexing has been well studied in centralized systems. A straightforward method to index multi-dimensional data is to employ Space Filling Curves [1] such as Hilbert curve or Z-curve (Z-order) to first convert multi-dimensional data to one-dimensional data and then to index the converted one-dimensional data in popular one-dimensional index structures. The problem of this method, however, is that it cannot index high-dimensional data efficiently. Alternatively, several index structures that directly index multi-dimensional data such as R-Tree [2], M-Tree [3], and SS-Tree [4] have been proposed. These index structures are generally based on a tree, where the data space is hierarchically divided into smaller subspaces when the space is overloaded and the tree grows up.

A popular approach to support multi-dimensional data indexing in P2P is to adapt centralized multi-dimensional index structures in P2P environment. Given a tree based index structure, the biggest challenge of this approach, however, is how to deal with the bottle-neck problem at the root of the tree structure since all queries need to be started at the root node. VBI-Tree [5] solves this challenge by keeping an upside-path routing table at every node in the tree structure and using upside-paths together with sideways-routing tables for routing purpose. In this way, since a query can start at any node in the tree structure, the bottle neck problem at the root node is eliminated. Nevertheless, the disadvantage of this solution is that it incurs a high cost for updating upside-paths of nodes when the index tree structure is changed.

To avoid the high cost of maintaining upside-paths, we propose SGR-Tree, an R-Tree [2] index structure built on top of a Skip Graph [6]. SGR-Tree uses a different way to build routing tables where no upside-paths are needed. In SGR-Tree, each Skip Graph node corresponds to a leaf node in the R-Tree. For routing purpose, each Skip Graph node virtually partitions the whole system into *non-overlapping* regions, each of which is connected to the node via a neighbor node. For each region, the node keeps the *minimum hyper-rectangle* covering all hyper-rectangles, which are in charged by nodes falling in that region. In this way, a query is still able to start at any node in the Skip Graph structure. When a node issues or receives a query, it needs to send or forward the query to all neighbor nodes whose minimum covering hyper-rectangle intersects with the search region. In addition to SGR-Tree, since load balancing is an important aspect of P2P systems, we also propose a mechanism for load balancing in SGR-Tree. Finally, we conduct experiments to evaluate the performance of SGR-Tree.

The rest of this paper is organized as follows. Section II introduces related work. Section III presents the architecture of SGR-Tree. Section IV describes how query is processed in SGR-Tree. Section V discusses the load balancing mech-

 ISBN: 978-1-61208-173-1

anism. Finally, Section VI shows experimental study and Section VII concludes the paper.

II. Related Work

In general, there are two main approaches to support multi-dimensional data indexing in P2P systems. In the first approach, multi-dimensional data is first converted to one-dimensional data using Space Filling Curves [1]. After that, the result one-dimensional data is indexed to P2P systems supporting one-dimensional data indexing. For example, authors of [7] and [8] share the same idea of using Hilbert curve for data conversion in the first step and Chord [9] for data indexing in the second step. In these methods, the system first encodes each dimensional value to a set of bit keys so that an n-dimensional data item is represented by n sets of bit keys. Hilbert curve is then used to convert these sets of bits keys to a single value for indexing to Chord. On the other hand, authors of [10] use Z-curse for data conversion and Skip Graph [6] for data indexing. The main disadvantage of methods belonging to the first approach is that they are not efficient to index high-dimensional data. Furthermore, these methods often have bad performance when data distribution is skewed since the cost of load balancing is very high.

To overcome the weakness of the first approach, the second approach tries to adapt centralized multi-dimensional data indexing structures in P2P environment. For example, CAN [11], the first P2P system supporting multi-dimensional data indexing, has a structure that is similar to kd-tree [12] and grid file [13]. Alternatively, Skip Index [14] utilizes kd-tree [12] to partition the data space into smaller parts and then maps these parts to Skip Graph [6] by encoding them into unique keys. On the other hand, P2PR-Tree [15] proposes a tree structure, which is adapted from R-Tree [2]. Additionally, VBI-Tree [5] and DP-Tree [16] are designed as frameworks that can deploy different types of index structures such as the R-Tree [2], M-Tree [3], SS-Tree [4] as well as their variants. By employing tree structures in distributed systems, the main challenge of methods belonging to the second approach is how to avoid the potential bottleneck occurred at the root node or nodes near the root since queries always start at the root node. The current solution used by existing index structures is to assign each peer node to represent a leaf node and to let the leaf node keep information about all internal nodes from itself to the root for routing purpose. Since SGR-Tree uses a different way to route queries where information of internal nodes is not needed to maintain, this is the main difference between SGR-Tree and existing index structures.

III. System Architecture

A. Overlay Network

In SGR-Tree, peers participating the system form a Skip Graph structure (i.e., each peer is a Skip Graph node) and each Skip Graph node corresponds to a leaf node in an R-Tree index [1]. While using Skip Graph as the overlay network, SGR-Tree is different from the original Skip Graph structure in three main features.

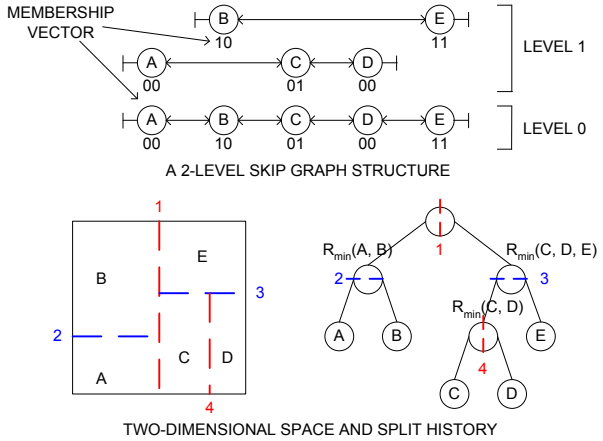


Figure 1. SGR-Tree architecture

- Each node in SGR-Tree is in charge of a *hyper-rectangle* in a multi-dimensional space instead of a range of values as in the original Skip Graph structure.
- To share the load of an existing node for a new coming node, the existing node splits its own hyper-rectangle on *one dimension* into two smaller hyper-rectangles so that the number of data covered by each hyper-rectangle is approximately equal. After splitting, the node on the left of the Skip Graph takes the lower hyper-rectangle while the node on the right takes the upper hyper-rectangle on the split dimension.
- When an existing node leaves the system, it passes its hyper-rectangle to the neighbor node, whose hyper-rectangle shares the border of its hyper-rectangle on one dimension.

An example of an SGR-Tree supporting two-dimensional data indexing with five nodes, A, B, C, D, and E in a two-level Skip Graph is shown in Figure 1 in which the top of the figure displays the Skip Graph overlay network while the bottom of the figure describes hyper-rectangles in charged by each node and the history of splitting the whole two-dimensional space into these hyper-rectangles.

B. Routing Table

SGR-Tree does not use information of internal nodes in the R-Tree structure for routing purpose as existing solutions do. Instead, each node in SGR-Tree creates its own routing table via its neighbor nodes. To create a routing table of a node x, x virtually partitions nodes in the system into *non-overlapping* regions, each of which is connected to x

[1] Since the terms "peer", "Skip Graph node", and "R-Tree leaf node" are interchangeable in our system, we shall simply refer to them as "node" when such reference does not cause any confusion. On the other hand, the term "internal node" in R-Tree shall be kept as it is.

 ISBN: 978-1-61208-173-1

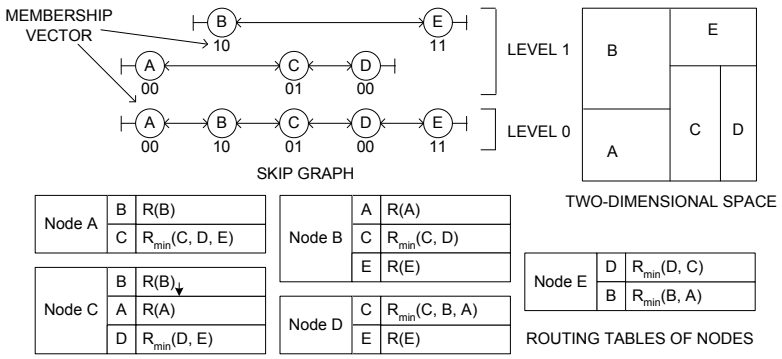


Figure 2. Routing tables of nodes in an SGR-Tree

through a neighbor node. For each region, the routing table of x contains information of the *minimum hyper-rectangle* covering all hyper-rectangles of nodes falling in that region. Given a neighbor y of x at level l in the Skip Graph structure, we define the non-overlapping region connected through y in the view of x as follows.

- If y is a neighbor of x at the highest level, the *non-overlapping region* associated with y consists of all nodes following y on the same side of x, including y. For example, as in Figure 2, since C is a neighbor of A at level 1, which is the highest level in the Skip Graph, the *non-overlapping region* associated with C in the routing table of A consists of C, D, and E (D and E are nodes following C on the right side of A).
- If y is not a neighbor of x at the highest level, the *non-overlapping region* associated with y includes all nodes falling between y and z, the neighbor of x at the nearest level higher than l on the same side of x as y, including y. For example, as in Figure 2, since C is a neighbor of B at level 0, which is not the highest level in the Skip Graph, the *non-overlapping region* associated with C in the routing table of B includes C and D (D is the node falling between C and E, the next neighbor node on the right side of B at level 1).

Figure 2 shows an SGR-Tree index structure with five nodes and details of routing tables at these nodes. In each routing table, the last two columns contain information of neighbor nodes and the *minimum hyper-rectangle* covering all hyper-rectangles of nodes in non-overlapping regions associated with the neighbor nodes. Note that $R(x)$ denotes the hyper-rectangle in charged by x and $R_{min}(x, y, z)$ denotes the minimum hyper-rectangle covering all hyper-rectangles in charged by x, y, and z.

C. Routing Table Construction

To create the routing table of a new node x, each neighbor y of x needs to send to x the *minimum hyper-rectangle* covering the hyper-rectangle in charged by y and all hyper-rectangles in charged by nodes following y on the opposite side with x (this minimum hyper-rectangle can be calculated from the routing table of y). Using received information from neighbor nodes, x builds its routing table as follows.

- If y is a neighbor of x at the highest level in the Skip Graph structure, the *minimum hyper-rectangle* associated with y in the routing table is the *minimum hyper-rectangle* x receives from y.
- If y is a neighbor of x at level l, which is not at the highest level in the Skip Graph structure, the *minimum hyper-rectangle* associated with y in the routing table is the *minimum hyper-rectangle* covering the remainder region of $\mathcal{R}(y) \setminus \mathcal{R}(z)$, where $\mathcal{R}(y)$ is the minimum hyper-rectangle x receives from y and $\mathcal{R}(z)$ is the minimum hyper-rectangle x receives from z, the neighbor of x at the nearest level higher than l on the same side of x as y.

For example, assume that a new node N joins to the right of the existing node B of the SGR-Tree in Figure 2. By joining the system, N takes over a part of the hyper-rectangle in charged by B. The overlay network as well as hyper-rectangles in charged by nodes in the SGR-Tree after the join of N are shown in the top of Figure 3. As the figure shows, the new node N has three neighbor nodes: A at level 1, B at level 0, and C and both levels 0 and 1. Thus, A, B, and C need to send to N the minimum hyper-rectangles covering all nodes following A, B and C in the opposite side of N. They are respectively $R(A)$, $R_{min}(A, B)$, and $R_{min}(C, D, E)$. Since A and C are neighbors of N at the highest level in the Skip Graph structure, $R(A)$ and $R_{min}(C, D, E)$ are also the minimum hyper-rectangles associated with A at C in the routing table of N. On the other hand, since B is not a neighbor at the highest level in the Skip Graph structure, the minimum hyper-rectangle associated with B in the routing table of N is the minimum hyper-rectangle covering $R_{min}(A, B) \setminus R(A) = R(B)$.

Note that since the join of a new node affects the routing tables of the new node's neighbors, neighbors of the new node also need to adjust their routing tables to reflect the existence of the new node and calculate the *minimum hyper-rectangle* associated with the new node from information

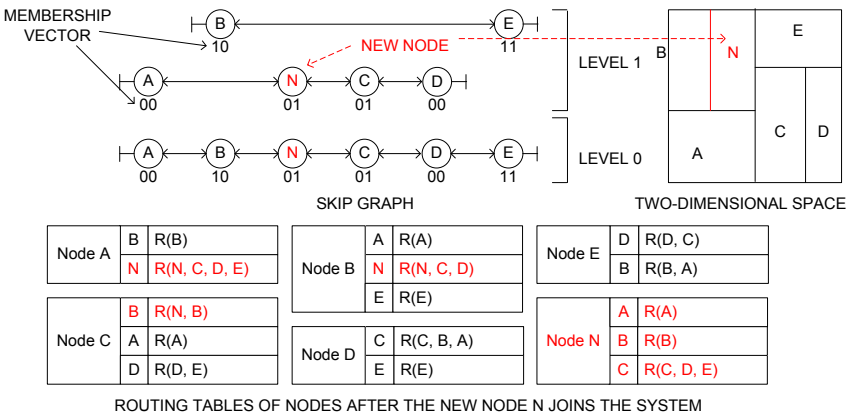


Figure 3. Routing table of the new node N and changes in the routing tables of A, B, and C (neighbors of N) in the SGR-Tree

provided by the new node. Nevertheless, the process of creating the routing table for the new node and updating routing tables of the new node's neighbors incur no additional message cost because the routing information can be piggy-backed with required messages used in the join process (messages used to set up neighbor links of the new node and update neighbor links of existing nodes).

IV. Query Processing

Since a multi-dimensional point query can be considered as a special case of a multi-dimensional range query where the searched region is a point, we only introduce the algorithm for processing a multi-dimensional range query in this section. Basically, when a node x issues a multi-dimensional range query q, x sends q to all neighbor nodes, whose *minimum hyper-rectangle* intersects with the searched region of q. Besides, to avoid sending duplicate query messages to the same node, when x sends q to a neighbor node y at level l, x also determines and sends to y a *lookup region* $\mathcal{L}(y, q)$, which limits neighbor nodes y can further forward q. $\mathcal{L}(y, q)$ is defined as follows.

- If y is a neighbor of x at the highest level in the Skip Graph structure, $\mathcal{L}(y, q)$ are all nodes following y in the opposite side of x.
- If y is not a neighbor of x at the highest level in the Skip Graph structure, $\mathcal{L}(y, q)$ are all nodes falling between y and z, the neighbor of x at the nearest level higher than l on the same side of x as y.

When y receives q from x, if the hyper-rectangle of which y is in charge intersects with the searched region of q, y executes q locally and returns the result to x. Additionally, if there is any neighbor node t of y, which falls into $\mathcal{L}(y, q)$ and has the *minimum hyper-rectangle* intersecting with the searched region of q, y first calculates $\mathcal{L}(t, q)$ based on the position of t and $\mathcal{L}(y, q)$ and then forwards q together with $\mathcal{L}(t, q)$ to t. Note that y does not need to forward q to other neighbor nodes not falling in $\mathcal{L}(y, q)$ because these neighbor nodes should receive the same query q from x or other neighbor nodes of x sooner or later.

For example, assume that node A issues a query q for the shaded region as in Figure 4. While having three neighbor nodes B, C, and D, A only sends q to D since the minimum hyper-rectangle of D, which is $R_{min}(D, E, F, G, H, I, J)$, intersects with the searched region of q (note that $\mathcal{L}(D, q) = \{E, F, G, H, I, J\}$). On the other hand, since the minimum hyper-rectangles of B and C, which are $R(B)$ and $R(C)$, do not intersect with the searched region of q, q is sent to neither B nor C. When D receives q from A, by checking its routing table, D continues to forward the query to E with $\mathcal{L}(E, q) = \{F\}$ and G with $\mathcal{L}(G, q) = \{H, I, J\}$ since E and G are neighbor nodes having minimum hyper-rectangles intersecting with the searched region of q in the opposite side of A. Similarly, q is then continuously forwarded to F with $\mathcal{L}(F, q) = \{\}$ from E; to H with $\mathcal{L}(H, q) = \{I, J\}$ from G; and to I with $\mathcal{L}(I, q) = \{J\}$ from H. Finally, the query is processed locally at F, G, H, and I since hyper-rectangles of which these nodes are in charge intersect with the searched region of q. Note that in Figure 4, routing entries with red and blue colors contain minimum hyper-rectangles that intersects with the searched region of q. However, only neighbor nodes in red routing entries fall in the limited lookup region defined by the query sender, and hence q is only forwarded to these nodes. As an example, when F receives q from E, even though it has two neighbor nodes G and H, whose minimum hyper-rectangle intersects with the searched region of q, it does not forward q to G and H because G and H are not in $\mathcal{L}(F, q)$.

According to the query processing algorithm, when a query q is sent from a node x to a neighbor node y, the lookup region for query processing at y, $\mathcal{L}(y, q)$, is limited by nodes either falling between y and the next neighbor node z at the nearest level higher than the level of y in the same direction with y or all nodes following y if y is the farthest neighbor node in one side of x. This way of limiting the lookup region is actually similar to that of the traditional

 ISBN: 978-1-61208-173-1

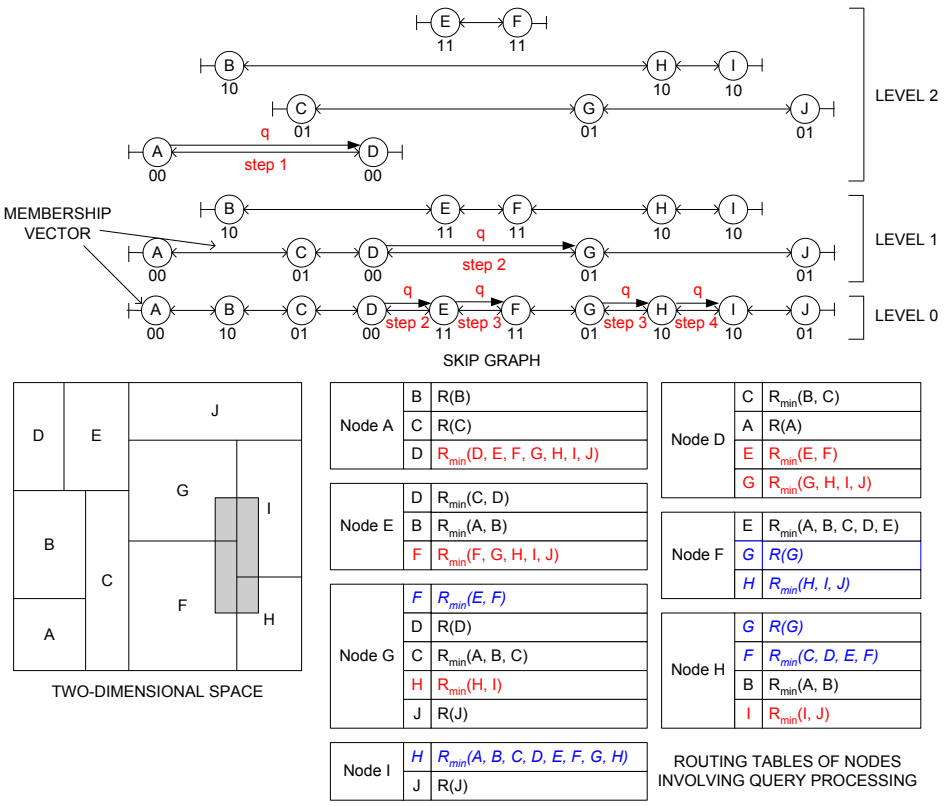


Figure 4. Query Processing in SGR-Tree

query processing in Skip Graph for single-dimensional range query. The only difference between our query processing algorithm and the traditional query processing is that our algorithm allows to send a query to multiple nodes instead of only one node. As a result, similar to traditional query processing algorithm in Skip Graph, the maximum number of steps required for processing a query in SGR-Tree is also bounded at $O(logN)$.

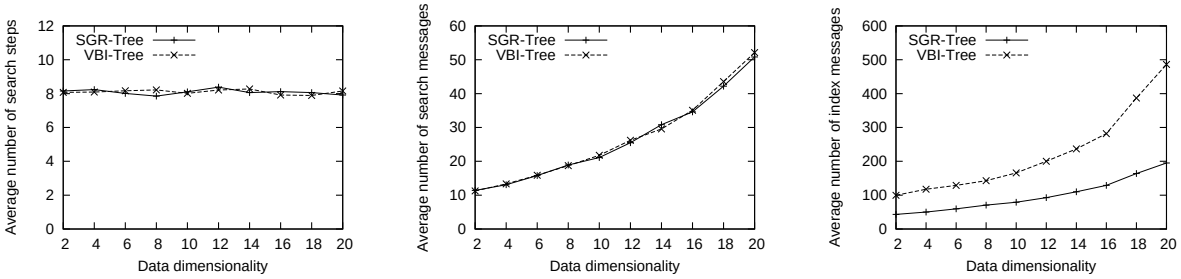
V. Load Balancing

To serve load balancing purpose, in addition to keeping minimum hyper-rectangles associated with neighbor nodes, we also maintain the number of data belonging to these minimum hyper-rectangles in the routing table. As a result, when a new node joins the system, it can select a heavily loaded node to join as an adjacent to share the heavy load. On the other hand, when a node is overloaded, it can also leverage information in the routing table to search for a lightly loaded node. The lightly loaded node then leaves the system and joins next to the heavily loaded node for load balancing. In other to keep the information up to date, whenever a load of a node is changed by a factor θ, the node sends an update load request to all of its neighbor nodes. When a node receives an update load request from its neighbor, it first updates the load of the minimum hyper-rectangle associated with the sender node. After that, if the new load of the whole group is also changed by a factor θ, the node continues to update its neighbor nodes but the sender node about the change in load.

VI. Experimental Study

To evaluate the performance of SGR-Tree, we implemented a simulator in Java to simulate an SGR-Tree of 10,000 nodes, where we inserted one by one 1,000,000 random multi-dimensional data objects. We tested the simulator with different data dimensionality from 2 to 20 and evaluated the system's performance according to three important criteria: the number of steps required to process a query (search steps), the number of messages required to process a query (search messages), and the number of messages required for building and updating routing tables (index messages). We used VBI-Tree [5] for comparison purpose. The experimental results are shown in Figure 5. The results show that SGR-Tree is comparable to VBI-Tree in terms of search steps and search messages. In particular, in both SGR-Tree and VBI-Tree the number of search steps is independent on data dimensionality while the number of search messages increases with the increasing of data dimensionality. On the other hand, in terms of index messages, SGR-Tree is much better than VBI-Tree. In most cases, SGR-Tree only incurs half of the cost compared to VBI-Tree. This confirms the efficiency of SGR-Tree.

 ISBN: 978-1-61208-173-1



(a) Average number of search steps (b) Average number of search messages (c) Average number of index messages

Figure 5. Performance of SGR-Tree and VBI-Tree

VII. CONCLUSION

In this paper, we have introduced SGR-Tree, a structure that is a combination of Skip Graph and R-Tree to support multi-dimensional data indexing in P2P systems. In SGR-Tree, participant peers form a Skip Graph overlay network in which each Skip Graph node corresponds to a leaf node in the R-Tree structure. For routing purpose, each node builds its own routing table by virtually dividing the whole system into non-overlapping regions, each of which is connected to the node through a neighbor node. For each region, the node maintains the minimum hyper-rectangle covering all hyper-rectangles in charged by nodes in the region. Based on this routing table structure, we have developed a query processing algorithm that can process any multi-dimensional query within $O(logN)$ steps and the query can start at any node in the SGR-Tree. Experiments have been done to evaluate the efficiency of SGR-Tree.

REFERENCES

[1] H. Sagan, *Space-Filling Curves*. Springer-Verlag, 1994.

[2] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1984, pp. 47–57.

[3] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: an efficient access method for similarity search in metric spaces," in *Proceedings of the 23rd International Conference on Very Large Databases (VLDB)*, 1997, pp. 426–435.

[4] D. A. White and R. Jain, "Similarity indexing with the ss-tree," in *Proceedings of the 12th IEEE International Conference on Data Engineering (ICDE)*, 1996, pp. 516–523.

[5] H. V. Jagadish, B. C. Ooi, Q. H. Vu, R. Zhang, and A. Zhou, "Vbi-tree: a peer-to-peer framework for supporting multi-dimensional indexing schemes," in *Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE)*, 2006.

[6] J. Aspnes and G. Shah, "Skip graphs," in *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003, pp. 384–393.

[7] J. Lee, H. Lee, S. Kang, S. Choe, and J. Song, "CISS: an efficient object clustering framework for DHT-based peer-to-peer applications."

[8] C. Schmidt and M. Parashar, "Flexible information discovery in decentralized distributed systems," in *Proceedings of the 12th International Symposium on High Performance Distributed Computing (HPDC)*, 2003.

[9] D. Karger, F. Kaashoek, I. Stoica, R. Morris, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications," in *Proceedings of the ACM SIGCOMM Conference*, 2001, pp. 149–160.

[10] Y. Shu, K.-L. Tan, and A. Zhou, "Adapting the content native space for load balanced," 2004, pp. 122–135.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of the ACM SIGCOMM Conference*, 2001, pp. 161–172.

[12] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, no. 9, pp. 509–517, 1975.

[13] K. Hinrichs and J. Nievergelt, "The grid file: a data structure designed to support proximity queries on spatial objects," in *Proceedings of the 9th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, 1983.

[14] C. Zhang, A. Krishnamurthy, and R. Y. Wang, "SkipIndex: towards a scalable Peer-to-Peer index service for high dimensional data," Princeton University, Tech. Rep. TR-703-04, 2004.

[15] A. Mondal, Y. Lifu, and M. Kitsuregawa, "P2PR-Tree: an R-Tree-based spatial index for Peer-to-Peer environments," in *Proceedings of the EDBT Workshop on P2P and Databases (P2PDB)*, 2004.

[16] M. Li, W.-C. Lee, and A. Sivasubramaniam, "DPTree: a balanced tree based indexing framework for peer-to-peer systems," in *Proceedings of the 14th International Conference on Network Protocols (ICNP)*, 2006.

 ISBN: 978-1-61208-173-1

Virtual Landmarking for Locality Aware Peer IDs

Alexander Allan, James Bradbury and Giuseppe Di Fatta
School of Systems Engineering
The University of Reading
Whiteknights, Reading, Berkshire, RG6 6AY, UK
A.J.M.Allan@student.reading.ac.uk, siu06jb@reading.ac.uk and G.DiFatta@reading.ac.uk

***Abstract*—In Peer-to-Peer (P2P) networks, it is often desirable to assign node IDs which preserve locality relationships in the underlying topology. Node locality can be embedded into node IDs by utilizing a one dimensional mapping by a Hilbert space filling curve on a vector of network distances from each node to a subset of reference landmark nodes within the network. However this approach is fundamentally limited because while robustness and accuracy might be expected to improve with the number of landmarks, the effectiveness of 1 dimensional Hilbert Curve mapping falls for the curse of dimensionality. This work proposes an approach to solve this issue using Landmark Multidimensional Scaling (LMDS) to reduce a large set of landmarks to a smaller set of virtual landmarks. This smaller set of landmarks has been postulated to represent the intrinsic dimensionality of the network space and therefore a space filling curve applied to these virtual landmarks is expected to produce a better mapping of the node ID space. The proposed approach, the Virtual Landmarks Hilbert Curve (VLHC), is particularly suitable for decentralised systems like P2P networks. In the experimental simulations the effectiveness of the methods is measured by means of the locality preservation derived from node IDs in terms of latency to nearest neighbours. A variety of realistic network topologies are simulated and this work provides strong evidence to suggest that VLHC performs better than either Hilbert Curves or LMDS use independently of each other.**

***Keywords*-Peer-to-Peer Networks; Hilbert Curve; Landmark Multidimensional Scaling; Virtual Landmarks; Network Coordinates**

I. INTRODUCTION

In Peer to Peer (P2P) networks it is useful in many circumstances for a node to be aware of its relative locality, the neighbourhood in which it resides. Latency optimizations that neighbourhood knowledge provides bring clear benefits to search and routing algorithms [1] [2] [3] [4]. In addition, locality information itself is key to many self organizing, cooperative and gossip based networks which represent an emerging paradigm in P2P technology [5] [6].

A node's calculation of its network locality is problematic as it is impractical for each node to apply a distance measure such as Round Trip Time (RTT) to all other nodes in the network.

A solution to this problem comes in the form of landmark clustering [3] used to create a scalar node ID space [7].

The assumption behind this technique is that RTT distances from any node to a predetermined set of landmark nodes (a subset of nodes which act as reference points) will be similar for other nodes in the immediate neighbourhood. The vector of distances to landmarks can be used in itself if the network protocols support a multidimensional index [8]. This work will be mainly focusing on the 1D scalar index that can be generated from this vector, noting that such an index can be more easily integrated into many existing peer-to-peer distributed hash table (DHT) systems. A scalar index provides an intuitive notion of locality for node naming schemes in all areas of P2P networks.

In order to create this scalar node ID from a distance vector, a dimensionality reduction algorithm is required. Previous work [9] has shown that in the context of landmark vector reduction, Hilbert Curves (HC), which are a type of space filling curve with good locality preserving properties [10] [11], outperform Principal Component Analysis (PCA) and Sammon Mapping [12] in terms of neighbourhood preservation. The Hilbert Curve can also be deployed in a distributed manner, requiring only a set of landmark vectors and a predefined HC granularity at each node to produce a homogeneous indexing for all nodes.

A problem identified with this method and with landmark indexing in general stems from the vulnerability and network traffic implication inherent in having a small number of landmark nodes upon which all other nodes depend to produce their node ID. Furthermore the less landmarks used, the greater the importance of landmark placement to the accuracy of the overall algorithm [13]. It is desirable therefore to increase the number of landmarks used as much as possible so as to negate the need for a heuristic landmark selection process and to decrease the vulnerability to single node failures.

However, Hilbert Curves are affected considerably by the curse of dimensionality [9], which causes accuracy to fall and computational load to increase with each additional dimension.

This work proposes a method to avoid the critical trade off in the number of landmark nodes. The notion of *virtual landmarks* is exploited to decouple the two issues. A large number L of actual landmark nodes are projected into an opportune small number K ($K < L$) of virtual landmarks which define a network coordinates space for a given topology. The HC is applied to the network coordinates space

 ISBN: 978-1-61208-173-1

which has lower dimensionality than the space defined by the actual landmarks. In particular, a decentralised version of the Landmark Multidimensional Scaling (LMDS) [14] algorithm is adopted to map the landmark vectors into virtual landmark vectors of lower dimensionality.

The proposed method, Virtual Landmarks Hilbert Curve (VLHC), is expected to be more accurate in preserving locality in the peer ID space than using either LMDS or Hilbert Curve alone.

In the experimental analysis realistic network topologies are used to compute and compare the accuracy of the methods in preserving nearest neighbour relations.

The remainder of the paper is organized in the following manner: Section II shows an overview of the various algorithms and techniques used, Section III describes the proposed method, Section IV describes the experimental methodology used for the comparison and benchmarking, Section V displays the experimental results and presents their analysis and, finally, Section VI provides some conclusive remarks and suggests future extensions to this work.

II. Overview of Techniques

A. Hilbert's Curve

The Hilbert Curve (HC) is a method of sequentially indexing points in space via a non intersecting line. The idea of space filling curves was first proposed theoretically by Giuseppe Peano in 1890 then geometrically by David Hilbert a year later. These curves allow points in an N dimensional space to be ordered on a 1D line in a manner which preserves locality relationships. Although a number of other space filling curves exist, such as Lebesque Curves, it has been shown that Hilbert Curves are among the best at preserving locality in terms of compact regions of space [15] [16].

B. Virtual Landmarking

Virtual landmarks in the context of network coordinates were proposed by Tang et al in 2003 [17]. The concept arose from analysis of the process in which locality information was embedded as sets of vectors to landmarks. This analysis showed that the intrinsic dimensionality of these embeddings in Internet-like networks was typically around 8 [17].

The concept of intrinsic dimensionality can be illustrated by imagining a mechanical arm with 5 joints. A data set might have the angle of each joint sampled at a certain time interval with the tip of the arm represented by a 5 dimensional coordinates. Since the mechanical arm exists in a 3D lab however, this position could be described by a 3 dimensional coordinate system with no reduction in accuracy. The embedding dimension of this data set would then be 5, with an intrinsic dimensionality of 3.

This work is based on the assumption that the intrinsic dimensionality of computer networks is relatively small. The approximate figure of 8 given by Tang et al. suggests that projecting a high dimensional coordinate space to a few dimensions will retain most of the locality information.

C. LMDS

Multidimensional Scaling (MDS) [18][19] is a family of dimensionality reduction methods. In classical MDS an $N \times N$ distance matrix is required, where N is the number of objects (nodes). The algorithm performs the eigendecomposition of the distance matrix and has a complexity of $O(N^3)$. In order to generate the landmark vectors each node would need to determine the communication latency to every other node in the topology. This approach clearly has practical limitations in a large-scale P2P network.

Landmark Multidimensional Scaling (LMDS) [20], [21] is a scalable MDS variant which does not require computationally expensive matrix calculations, nor the entire distance matrix. Landmark MDS instead performs a classical MDS on the subset of landmark nodes only, and computes embedding coordinates for the other nodes by using distance-based triangulation by means of the decomposed landmark matrix.

Landmark MDS was developed primarily as a technique to speed up the ISOMap procedure [20], and has been shown to be equivalent to the Nyström approximation of the eigenvectors and eigenvalues of a matrix [22], The method works by utilising properties of kernel matrices to calculate embedding coordinates based upon estimations of eigenvalues and eigenvectors.

Two distance matrices are calculated, landmark node to landmark node matrix A, and landmark node to non-landmark node matrix B. To distinguish Landmark MDS from the Nyström approximation, A and B must undergo double-centering, akin to the classical MDS procedure [22]. The Nyström approximation then calculates estimated embedding coordinates using values from the eigendecomposed A matrix and values from the B matrix only, therefore negating the need for the costly calculation of the $N \times N$ distance matrix as well as its eigendecomposition. LMDS has a complexity of $O(NLk + L^3)$, where L ($L \ll N$) is the number of landmarks and k ($k \leq L$) is the number of the largest positive eigenvalues used in the approximation.

D. Landmark Selection

The landmark selection problem has itself been the subject of much work as for small numbers of landmarks it can have a large impact on any triangulation based methods [23][13][24]. Various heuristics have been proposed which attempt to select landmarks spread throughout the network with a uniform distribution.

However, Tang et al (2004) [13] find that as the number of landmarks in a network surpass 20, most landmark selection techniques (with the exception of a computationally infeasible greedy approach) are no better than random selection because a uniform distribution inevitably emerges from any random selection method given sufficient points.

 ISBN: 978-1-61208-173-1

This work adopts a random selection of a large number of landmarks, thus avoiding the complexity of implementing a distributed non random selection heuristic.

III. Virtual Landmarks Hilbert Curve (VLHC)

This work proposes a novel approach for 1 dimensional index generation by combining the notion of virtual network landmarks with the two techniques HC and LMDS. In a two-stage process at each node, LMDS is applied to the local RTT vector v^L obtained from a large set of L landmarks. LMDS is used to project the high-dimensional RTT vector to a lower K-dimensional space, where $K < L$. The corresponding low-dimensional vector v^K is referred to as the *virtual landmark vector*.

In the second stage the local virtual landmark vector is converted into a scalar index by means of a Hilbert Curve.

A traditional approach would apply HC to the landmark vectors in L dimensions. In the proposed approach HC is applied to the virtual landmark vectors in K dimensions. The LMDS dimensionality reduction is adopted to overcome the restriction of the Hilbert Curve that performs badly in high dimensional spaces. This combination is also likely to be more effective than using the LMDS to project directly into the 1D index space, as Hilbert Curves have more favourable locality preserving properties.

A. Decentralised Algorithm

A distributed approach for conducting the LMDS at each node would be as follows.

In the initialisation step the landmark nodes ping each other in order to create a matrix of latencies (RTT) between them. The landmark nodes perform an all-to-all broadcast operation to propagate their local latency vector and to generate the $L \times L$ matrix. Each landmark node then uses this matrix to perform classic MDS locally and independently. All landmark nodes will compute the eigenvalue decomposition of the matrix. The landmark nodes generate an identical reference set of K[1] eigenvalues and vectors.

When a node joins the network, it pings the landmarks to create a local landmark vector. The node also requests and receives the reference set of eigenvalues and eigenvectors from any one of the landmarks. From the reference set and its local RTT vector, the node can calculate its own approximate position in the lower dimensional MDS projection using the Nyström approximation [22]. The coordinates of this approximate position is then taken as the node's *virtual landmark vector* to which the Hilbert Curve is applied to produce the 1D node index (peer ID).

The approach is scalable as nodes have to communicate with landmark nodes only. The number and choice of actual landmarks is not critical as already discussed.

[1]The actual number of virtual landmarks corresponds to the maximum between the parameter K and the number of positive eigenvalues.

The VLHC technique can be applied in P2P networks without the need for global communication and synchronisation. This can be done in a fully decentralised approach as outlined in the following steps.

Using a Gossip-based protocol nodes can randomly select the landmarks by exchanging sorted lists of IP addresses and choosing the top L after a suitable convergence period.

The elected landmark nodes collect RTTs to each other and create the matrix needed for LMDS. The landmark nodes distribute the matrix throughout the network at a certain rate to prevent a traffic bottleneck at the landmarks.

When each node receives the matrix, it can then calculate its virtual landmarks vector and from this its node ID.

IV. Comparative Analysis

The comparative analysis of the different methods is based on the average latency to the 30 Nearest Neighbour (NN) nodes. Two methods provided the baseline and the optimal value of the performance index. The ideal performance was computed by searching the 30 actual NNs in the topology for each node. The random selection of 30 nodes provided a baseline value of the worst performance. The performance indices from several random selections of landmark sets were averaged.

In the experimental evaluation four methods were tested to carry out a comparative analysis:

- 1-dimensional Landmark Multidimensional Scaling (1D LMDS),
- Hilbert Curve applied to the landmark vectors (Hilbert),
- the proposed Virtual Landmarks Hilbert Curve (VLHC) and
- 8-dimensional LMDS Network Coordinates (8D Network Coordinates).

The 8D LMDS generates a network coordinates scheme which is suitable to assess the information loss associated with the virtual landmarks. Ideally a loss-less latency vector projection from the L-dimensional space to the K-dimensional space would produce a performance index comparable to the ideal NN method. It should be noticed that this method does not produce a scalar index which can be used as peer ID and it serves as reference only.

In each test L landmark nodes were chosen randomly and RTTs between them were used to create the matrix for the LMDS calculations with the Nyström approximation. RTTs from each node i to the landmark nodes were determined, v_i^L, and were input into the LMDS algorithm to create a vector of distances to K virtual landmarks v_i^K.

The Douge Moore's C implementation [25] of a recursion free algorithm of the Hilbert's space filling curve [26] was adopted. It was applied to v_i^K to produce the 1D index $H(v_i^K)$. The same algorithm was also applied to the vector v_i^L to produce the 1D index $H(v_i^L)$ for the classical Hilbert Curve method.

 ISBN: 978-1-61208-173-1

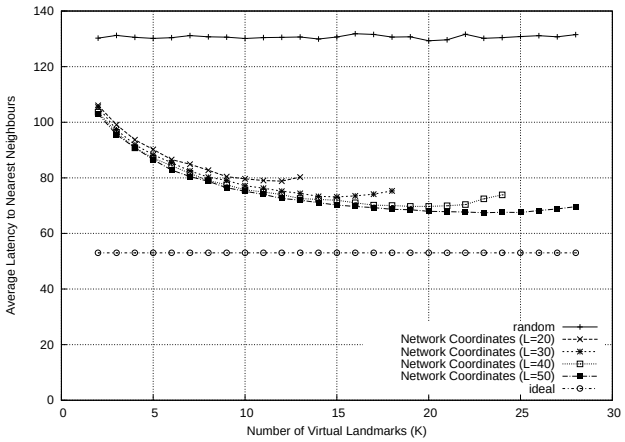


Figure 1. Average latency to nearest neighbours vs. the number of virtual landmarks: reference methods only

LMDS with $K = 1$ was applied to vectors v_i^L to produce a 1D index $L(v_i^L)$. Each of the three methods generates a different set of peer IDs, respectively, $\{H(v_i^K)\}$, $\{H(v_i^L)\}$ and $\{L(v_i^L)\}$.

Each generated index set was used to search the set of 30 nearest neighbours nodes for each node. Multiple sets of random nodes were generated and the exact NNs in the topology were computed using the Floyd Warshals algorithm [27]. The set of NNs in the 8D network coordinates scheme was determined by means of the Euclidean distance.

A set of NNs was used to compute an accuracy index. The distances between every node and its NNs was used to produce an average NN latency for each method.

The accuracy index is a normalized average NN latency ($\in [0, 1]$) which is designed to be relevant across heterogeneous topologies. An accuracy index of 1 is assigned to the ideal average NN latency; 0 is assigned to the worst average NN latency average associated with the random method. The average NN latency of each method and on each topology was normalized between these two indices. Using this accuracy measure results close to 1 indicate the method was almost as good as was possible and results which were closer to 0 imply that the method was almost as bad as a purely random selection of neighbours.

V. Simulations and Results

We have tested the algorithms in two different types of network topologies:

- 10 Internet-like topologies with 1000 nodes generated by the topology generator BRITE [28] with a Waxman model [29] to simulate a flat level Autonomous System;
- 10 2D mesh topologies with dimensions 40×25 (1000 nodes).

In the first test the average NN latency of the three reference methods (ideal, random and Network Coordinates) was compared to verify the intrinsic dimensionality of the Internet-like topologies. In Figure 1 the average NN latency for these three methods is shown for a varying number of virtual landmarks K. The number of actual landmarks L is fixed to 20, 30, 40 and 50. For the Network Coordinates method and for a given value L, the test was executed till the eigendecomposition would return enough positive eigenvalues to generate the desired number K of virtual landmarks. For $K < 10$ the average NN latency of the Network Coordinates scheme clearly improves as K increases. For $K > 10$ it does not improve further or it worsens. The test was run on a single Internet-like topology; similar results were obtained for the other topologies. This experiment shows that a choice of 8 virtual landmarks is a good trade off, confirming the intrinsic dimensionality of the Internet-like topologies as determined in [17]. In all other simulations we have fixed the number K of virtual landmarks to 8.

All methods were tested on each network topology multiple times. For each topology, the resulting performance indices were averaged over 20 tests with different random selections of landmarks. For each topology, the number of landmarks was increased from $L = 3$ to $L = 49$ and for every L value the normalised accuracy index was computed as the average over 20 tests with different random selections of landmarks.

Figures 2(a) and 2(b) show the accuracy of the methods when varying the number of the actual landmark nodes.

The curve for the Hilbert method is truncated at $L = 20$. Within Douge Moore's C implementation of the Hilbert Curve, the number of input bits (which defines the granularity of the curve) multiplied by the number of input dimensions cannot exceed the value of 8 times an unsigned 'long long' data type (which is 64 bytes in the used CPU architecture). A curve with 3-byte granularity was used to give sufficient precision, so when the input dimensionality increased beyond 21, the curve would no longer compute as $22 \times 3 > 64$.

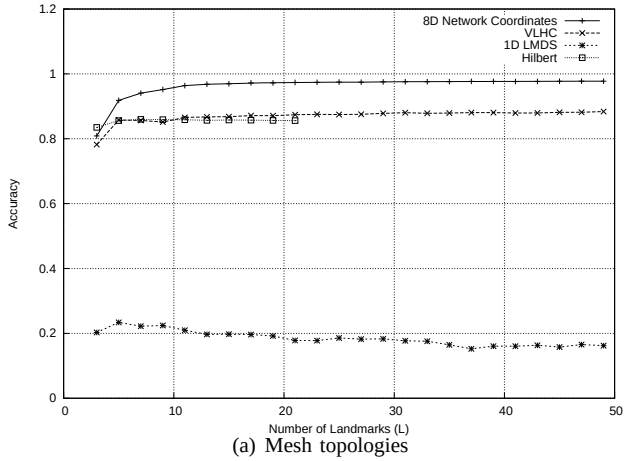
A. Discussion

The VLHC technique performs better than either the 1D LMDS or stand alone Hilbert curve on both mesh and Internet like topologies. Its accuracy plateaus at around 0.87 on the mesh topology, and around 0.41 on the Internet like topology. It achieves this after 18 or more landmarks are used, only improving marginally after this point as more landmarks are added.

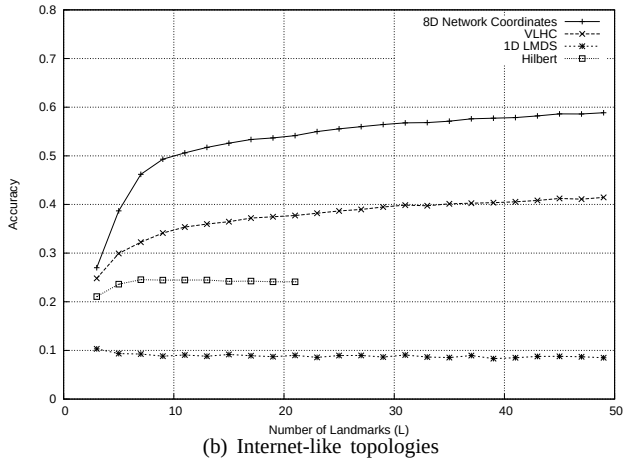
On the mesh topologies, where the intrinsic dimensionality is 2, the standard Hilbert Curve computed over all landmarks was comparable to VLHC. However, implementation issues did effect its actual scalability in terms of the number of landmarks.

On the Internet like topologies, the standard Hilbert Curve achieved a maximum accuracy of around 0.24 after 6 landmarks but could only maintain this until the implementation failed at 22 landmarks. On this type of topologies, where the

 ISBN: 978-1-61208-173-1



(a) Mesh topologies



(b) Internet-like topologies

Figure 2. Accuracy for mesh (a) and Internet-like topologies (b)

intrinsic dimensionality is about 8, it performed constantly worse than the VLHC.

On the mesh topologies the 1D index produced by the LMDS was the worst performer with a peak of 0.23 accuracy at 5 landmarks. As more landmarks were added the accuracy fell slowly. For the Internet like topology the 1D LMDS index was again the worst performer for any numbers of landmarks we have tested, getting a peak accuracy score of 0.1 which again fell slowly as the number of landmarks increased.

The 8D network coordinates produced by LMDS showed almost optimal performance in the mesh topologies, with accuracy rising up to 0.97 after 17 landmarks. In the Internet like topologies, however, it performed significantly worse, achieving a maximum of 0.58 at 49 landmarks. The curve shows a steep improvement up to around 8 landmarks, at which point the rate of improvement slows considerably. This is also another indirect confirmation of the intrinsic dimensionality of this type of topologies.

The non optimal performance of the 8D network coordinates system in the Internet-like topologies may be due to using a triangulation based technique (which is essentially how LMDS works) in a network space which violates the triangular inequality (the communication latency from A to B might be more than from A to C to B). The mesh topology however is much closer to Euclidean space and so the inequality would hold true in most cases.

The effectiveness of the VLHC 1D node indexing scheme is considerably greater in the mesh topology than in the Autonomous Systems topology. Indeed it scores only 12% less than a brute force approach in the mesh topology which makes it suitable for most applications. The utility in an Internet like topology will be considerably more dependant on the problem domain, as the peak accuracy of 41% may not be sufficient for some applications where a high degree of geographical accuracy is needed, but rather for ones where a notion of locality in a scalar index outweighs the overheads of adopting a more accurate but more complex multidimensional network coordinate scheme.

VI. Conclusion

This work has presented the application of the concept of virtual landmarks to the problem of generating locality aware peer identifiers by means of space filling curves. Quantitative evidence has suggested that applying LMDS in conjunction with a Hilbert Curve produces a superior mapping to a 1D index in terms of locality preserving properties, when compared to either technique applied independently. This agrees with the postulation that a small number of virtual landmarks are sufficient to capture the intrinsic dimensionality of Internet-like networks. More experimental work is required to investigate the sensitivity of the proposed method with respect to this parameter in real network topologies.

In contrast to previous applications of the Hilbert Curve, a larger numbers of landmarks can be employed by exploiting a decentralised LMDS algorithm. This not only increases accuracy but also allows for more robustness.

The experimental analysis has shown that, as expected, 1D indices are less accurate than multidimensional network coordinates. In general, applications should consider what accuracy level of nearest neighbour preservation is needed before adopting either a 1D scheme or a multidimensional scheme, as the cost of simplicity is still substantial.

Further work will include using graphs extracted from real network topologies to support the results obtained via simulation, the implementation and testing of the outlined Gossip-based approach and the adaptation of the algorithm to handle a dynamic environment with churn rate for nodes and landmarks and with time-varying latencies.

References

[1] H. Shen and C. Xu, “Hash-based proximity clustering for load balancing in heterogeneous dht networks,” *Journal of Parallel and Distributed Computing*, vol. 65, no. 5, pp. 686–702, May 2005.

 ISBN: 978-1-61208-173-1

[2] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting network proximity in peer-to-peer overlay networks," Microsoft Research, Cambridge, England, Tech. Rep. MSR-TR-2002-82, May 2002.

[3] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings*, New York, USA, Jun. 2002, pp. 1190–1199.

[4] Y. Zhu and Y. Hu, "Towards efficient load balancing in structured P2P systems," in *18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, Santa Fe, USA, april 2004, p. 20.

[5] M. Cai and M. Frank, "RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 650–657.

[6] S. Savarimuthu, M. Purvis, M. Purvis, and B. Savarimuthu, "Mechanisms for the self-organization of peer groups in agent societies," *Multi-Agent-Based Simulation XI*, pp. 93–107, 2011.

[7] Z. Li, G. Xie, and Z. Li, "Locality-aware consistency maintenance for heterogeneous P2P systems," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2007, pp. 1–10.

[8] M. Gharib, Z. Barzegar, and J. Habibi, "A novel method for supporting locality in peer-to-peer overlays using hypercube topology," in *International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, 2010, pp. 391–395.

[9] A. Allan and G. Di Fatta, "Effectiveness of landmark analysis for establishing locality in P2P networks," in *The Second International Conference on Advances in P2P Systems (AP2PS 2010)*, October 2010.

[10] C. Gotsman and M. Lindenbaum, "On the metric properties of discrete space-filling curves," *IEEE Transactions on Image Processing*, vol. 5, no. 1, pp. 794–797, Jan. 1996.

[11] B. Moon, H. Jagadish, C. Faloutsos, and J. Saltz, "Analysis of the clustering properties of the Hilbert space-filling curve," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 1, pp. 124–141, Jan. 2001.

[12] J. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. C-18, no. 5, pp. 401–409, May 1969.

[13] L. Tang and M. Crovella, "Geometric exploration of the landmark selection problem," *Passive and Active Network Measurement*, pp. 63–72, 2004.

[14] V. De Silva and J. Tenenbaum, "Sparse multidimensional scaling using landmark points," *Dept. Math., Stanford University, Stanford, CA, Tech. Rep*, 2004.

[15] C. Faloutsos and Y. Rong, "Spatial access methods using fractals: Algorithms and performance evaluation," University of Maryland, Maryland, USA, Tech. Rep. UMIACS-TR-89-31, Mar. 1989.

[16] H. Jagadish, "Linear clustering of objects with multiple attributes," *ACM SIGMOD Record*, vol. 19, no. 2, pp. 332–342, Jun. 1990.

[17] L. Tang and M. Crovella, "Virtual landmarks for the internet," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM, 2003, pp. 143–152.

[18] W. Torgeson, "Multidimensional scaling of similarity," *Psychometrika*, vol. 30, pp. 379–393, 1965.

[19] R. Shepard, "Analysis of proximities: Multidimensional scaling with an unknown distance function I & II," *Psychometrika*, vol. 27, pp. 125–140, 219–246, 1962.

[20] J. B. Tenenbaum and V. de Silva, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[21] V. de Silva and J. B. Tenenbaum, "Sparse multidimensional scaling using landmark points," University of Stanford, Standford, USA, Tech. Rep. CSE-TR-456-02, jun 2004.

[22] J. C. Platt, "Fastmap, metricmap, and landmark mds are all nystrom algorithms," in *In Proceedings of 10th International Workshop on Artificial Intelligence and Statistics*. IEEE, 2005, pp. 261–268.

[23] S. Baskakov, "Landmarks selection algorithm for wireless sensor networks," in *Self-Adaptive and Self-Organizing Systems, 2008. SASO'08. Second IEEE International Conference on*. IEEE, 2008, pp. 361–369.

[24] Q. Cao and T. Abdelzaher, "Scalable logical coordinates framework for routing in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 4, pp. 557–593, 2006.

[25] D. Moore. (2011) Fast hilbert curve generation, sorting, and range queries. Rice University. Texas, USA. [Online]. Available: http://www.tiac.net/∼sw/2008/10/Hilbert/moore/hilbert.c

[26] A. Butz, "Alternative algorithm for Hilbert's space-filling curve," *IEEE Transactions on Computers*, vol. C-20, no. 4, pp. 424–426, april 1971.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (3rd Ed.)*. MIT Press, 2009.

[28] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," in *Proc. of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'01)*, 2001, pp. 346–353.

[29] B. Waxman, "Routing of multipoint connections," *Selected Areas in Communications, IEEE Journal on*, vol. 6, no. 9, pp. 1617–1622, 1988.

 ISBN: 978-1-61208-173-1

Leveraging Social and Content-based Recommendation in P2P Systems

Fady Draidi, Esther Pacitti, Michelle Cart, Hinde Lilia Bouziane
INRIA & LIRMM, Montpellier, France
{fady.draidi, esther.pacitti, cart, hinde.bouziane}@lirmm.fr

***Abstract*-We focus on peer-to-peer (P2P) content recommendation for on-line communities, where social relationships between users can be exploited as a parameter to increase the trust of recommendation. Most of the existing solutions establish friendship relationships based on users behavior or declared trust. In this paper, we propose a novel P2P recommendation approach (called F2Frec) that leverages content and social-based recommendation by maintaining a P2P and friend-to-friend network. This network is used as a basis to provide useful and high quality recommendations. Based on F2Frec, we propose new metrics, such as usefulness and similarity (among users and their respective friend network), necessary to enable friendship establishment and to select recommendations. We define our proposed metrics based on users' topic of interest and relevant topics that are automatically extracted from the contents stored by each user. Our experimental evaluation, using the TREC09 dataset and Wiki vote social network, shows the benefits of our approach compared to anonymous recommendation. In addition, we show that F2Frec increases recall by a factor of 8.8 compared with centralized collaborative filtering.**

***Keywords*-P2P systems; personalization; recommendation; gossip protocols; social networks.**

I. INTRODUCTION

We focus on Peer-to-Peer (P2P) large scale content sharing for on-line communities. For instance, in modern e-science (e.g., bio-informatics, physics and environmental science), scientists must deal with overwhelming amounts of contents (experimental data and documents, images, etc.) produced and stored in his workspace that they are willing to share within a community or with specific friends without relying in a centralized server.

Peer-to-Peer (P2P) networks, offers scalability, dynamicity, autonomy and decentralized control. Locating contents based on contents ids in a P2P overlay network is now well solved (see [4]). However, the problem with current P2P content-sharing systems is that the users themselves, i.e., their interest or expertise in specific topics, or their rankings of documents they have read, are simply ignored. In other words, what is missing is a recommendation service that, given a query, can recommend relevant documents by exploiting user information.

Sinha et al. [11] have shown that users prefer the advices that come from known friends in terms of quality and trust, because users typically *trust* their friends' advices. The emersion of Web2.0 and the growing popularity of online social networks have encouraged exploiting users' social data in P2P systems. In existing P2P solutions, friendship links are extracted from users' behaviors [6] or are established based on explicit trust declaration [8]. To enrich these solutions, we consider that users that store similar contents may be potentially friends with a specific declared trust level with respect to the relevance of a user in a specific topic. Thus, our decentralized recommendation approach leverages content-based and social-based recommendation over a distributed graph, where each node represents a user labelled with the contents it stores and its topics of interests. As a basis for recommendation, we propose new social metrics such as similarities (among users and their respective friend network) and usefulness of a user with respect to a friend or query taking into account the declared trusts. These measures are defined based on user topics of interest and relevant topics that are automatically extracted from the contents they store. Notice that a user is considered *relevant* in a specific topic *t* if it has a sufficient amount of content with high probability related to *t.* Then this user will be relevant to serve queries related to *t.* also a user *v* is considered *useful* to a user *u*, if *v* is relevant in topics that *u* is interested in.

We implement friendship networks using concepts from the Friend-Of-A-Friend (FOAF) project. FOAF provides an open, detailed description of profiles of users and the relationships between them using a machine-readable syntax. We use FOAF files to support users' queries. To establish friendship and disseminate recommendation, we rely on gossip protocols [3] as follows: At each gossip exchange, each user *u* checks its gossip *local-view* to enquire whether there is any relevant user *v* that is useful to *u*, and whether its friendship networks have high overlap with *u*'s friendship network. If it is the case, a demand of friendship is launched among *u* and *v* and the respective FOAF files are updated accordingly.

Whenever a user submits a keyword query, its FOAF file is used as a directory to redirect the query to the top-k most adequate friends taking into account similarities, relevance, usefulness and trust. In our previous work [3], we focused on P2P anonymous recommendation exploiting different types of gossip protocols.

In this paper, we propose F2Frec, a new social-based approach for recommendation that facilitates the construction and maintenance of P2P social network and exploits social metrics to provide recommendations. Our major contributions are:

- We introduce new social metrics to suggest friends and detect if a friend is relevant and useful to provide recommendations.
- We propose an efficient query routing algorithm that takes into account the social metrics to select, in a top-k approach, the most appropriate friends to provide recommendation.

 ISBN: 978-1-61208-173-1

- Once the best recommendations are provided, we propose to rank them by taking into account the semantic similarities, content popularity, distance and trust between query's initiator and responders.
- We provide an experimental evaluation using real data sets that demonstrates the efficiency of F2Frec over the TREC09 [10] and Wiki vote social network [12] compared to anonymous P2P recommendations and centralized recommendation.

The rest of this paper is organized as follows. Section II provides a general overview of F2Frec. Section III presents our social metrics and how we manage friendship establishment. Section IV describes our solution for retrieving recommendations over F2Rrec given a key-word query. Section V gives our experimental validation that compares F2Rrec with centralized collaborative filtering. Section VI discusses related work. Section VII concludes.

II. General Overview of F2F Recommendation

Our recommendation model is expressed based on a graph $G = (D,U,E,T)$, where D is the set of shared documents, U is the set of users $u_1,...u_n$ corresponding to autonomous peers $p_1,...p_n$, E is the set of edges between the users such that there is an edge $e(u,v)$ if users u and v are friends, and T is the domain of topics. Each user $u \in U$ is associated with a set of topics of interest $T_u \subset T$, and a set of relevant topics $T_u^r \subset T_u$ extracted locally from the documents u has rated. The rating that has been given by a user u on document doc is denoted by $rate_{doc}^{u}$.

In our approach, we use Latent Dirichlet Allocation (LDA) [2] to automatically extract the topics in the system, which in turn are used to extract users' relevant topics of interest. In F2Frec, LDA processing is done in two steps: the training at a global level, and inference at the local level. The global level is given to the bootstrap server (BS) that aggregates a sample set of M documents from F2Frec participant peers. Then, BS runs the LDA classifier to get a set $T = \{t_1,..t_k\}$ of topics, where k is the number of topics. Each topic $t \in T$ contains a set of Z words, where Z is the number of unique words in M, and each word $z \in Z$ is associated with a weight value w_z^t between 0 and 1. The w_z^t represents how much the word $z \in Z$ is related to t. At the local level, user u performs LDA locally to extract the topics of its local documents, using the same set of topics T that were previously generated at the global level. LDA provides a vector of size k for each document doc, $V_{doc} = [w_{doc}^{t1},...,w_{doc}^{tk}]$, where w_{doc}^{t} is the weight of each topic $t \in T$ with respect to doc.

Users' relevant topics of interest are extracted based on a combination between documents' semantics and ratings. Since we focus on on-line communities, we safely assume that users are willing to rate the documents they store. Once a user u extracted the V_{doc} for each $doc \in D_u$, it multiplies the $V_{doc} = [w_{doc}^{t1},..,w_{doc}^{tk}]$, by the rating $rate_{doc}^{u}$. Then, user u identifies for each topic $t \in T$ only the documents that are highly related to t. A document doc is considered highly related to topic t, if its weight in that topic w_{doc}^{t} multiplied by its rating $rate_{doc}^{u}$ exceeds a threshold value. Next, u counts how many documents are highly related to each topic $t \in T$. User u is considered interested in topic $t \in T_u$ if a percentage y of its local documents are highly related to topic t. Finally, u is considered a *relevant user* in topic $t \in T_u^r$ if it is interested in t and has a sufficient amount x (system-defined) of documents that are highly related to topic t.

Each user $u \in U$ maintains locally a FOAF file that contains a description of its personal information, and friendship network, denoted by $friends(u)=\{f_1, f_2,...f_n\}$. Personal information includes the extracted topics of interest, where each topic of interest $t \in T_u$ is associated with a Boolean value that indicates whether u is relevant in that topic. Friends' information includes friends' names, links (URI) to their FOAF files, relevant topics of interest, and trust levels. The trust level between user u and a friend v, denoted by $trust(u,v)$, is a real value within [0, 1] and represents the faith of user u in its friend v. The trust level between user u and its friend v can be obtained explicitly [8] or implicitly [7].

Furthermore, each user $u \in U$ establishes new friendships with users that are *useful* to u's demands or have friendship networks with high overlap with u's friendship network. A user v is considered *useful* to a user u, if v is a relevant user and a certain amount of v's relevant topics T_v^r are of interest for u. User u exploits its useful friends (of friends) for recommendations. Notice that, if a friendship acquaintance exists between users u and v, u implicitly recommends its documents to v and vice-versa, in related topics. More precisely, if there is a friendship path between users u and v, $path(u,v)=\{(u, v_i), (v_i,v_j),...,(v_k, v)\}$, then u can recommend its documents related to their topics of interest to v and vice-versa.

Queries are expressed through key-words, and mapped to topic(s) T_q using LDA. Moreover, queries are associated with a TTL (Time To Live), and routed recursively on a distributed top-k algorithm: Once a query is received by any user, it is forwarded to its top-k best friends by taking into account usefulness and trust. A response to a query q is a recommendation provided in a ranked list and defined as:

$$recommendation_q = rank(rec_q^1(doc_1),... rec_q^n(doc_j)) \quad (1)$$

Different recommendations may be given for the replicas of a document doc_i. The $recommendation_q$ is ordered based on a ranking function, that ranks each $rec_q^v(doc_i)$ according to its relevance with q, its popularity, and the distance and trust between the q initiator and responder v. More details on query processing and recommendations ranking are given in Section IV.

The trust value between a query's initiator u and a responder v, denoted by $trust_q(u,v)$, is computed during query processing. The path of a query q between u and v can be represented as $path_q(u,v)=\{(u, v_i), (v_i,v_j), (v_j, v)\}$, and the trust value between u and v can be computed by multiplying the trust values among directs friends along the $path_q(u,v)$, which is:

$$trust_q(u,v) = \prod_{v_i,v_j \in path_q(u,v)} trust(v_i, v_j) \quad (2)$$

III. Friend to Friend Recommendation

The goal is to let each user explicitly establish friendship with useful users, so that it can exploit them for recommen-

dation. First, we present the similarity metrics we propose. Then, we present the data structures and algorithms for friendship establishment.

A. Metrics

We compute the similarity distance between *u* and *v* based on their friendship networks and relevant topics of interest. We measure the similarity distance between *u* and *v* based on their friendship networks, denoted by $distance_{fri}(u,v)$, by counting the overlap of their friends. We use the dice coefficient, which is:

$$distance_{fri}(u,v) = \frac{2|friend(u) \cap friend(v)|}{|friend(u)|+|friend(v)|} \quad (3)$$

We could also use other similarity functions such as cosine, jaccard, etc. We use $distance_{fri}(u,v)$ as a measure for the implicit trust between *u* and *v*.

We measure the common interest of topics between user *u* and *v*, denoted by $distance_{intr}(u,v)$, by counting the overlap of their topic of interests. We use the dice coefficient, which is:

$$distance_{intr}(u,v) = \frac{2|T_u \cap T_v|}{|T_u|+|T_v|} \quad (4)$$

Notice that user *u* and *v* may be similar in terms of topics of interest. However, *v* may not be useful for *u*, because the topics of interest of *u* are not related to *v*'s relevant topics. Therefore, we measure how much *v* is useful to *u*, denoted by *useful(u,v)*, by counting the overlap between *u*'s topics of interest T_u and *v*'s relevant topics T_v^r. Similarly, we use the Dice coefficient to measure *useful(u,v)*:

$$useful(u,v) = \frac{2|T_u \cap T_v^r|}{|T_u|+|T_v^r|} \quad (5)$$

We measure the final similarity distance between *u* and *v*, denoted by *sim(u,v)*, by combining $distance_{fri}(u,v)$ with *useful(u,v)* in a weighted approach as follows:

$$sim(u,v) = \alpha * useful(u,v) + (1-\alpha) * distance_{fri}(u,v) \quad (6)$$

The parameter α is used to adjust whether *u* prefers to establish friendship with users that are highly useful to its queries, or with users that their friendship networks are highly overlapped with *u*'s friendship network. As α values become close to 1, the usefulness of users play a more important role in the final similarity distance *sim(u,v)*.

Also, we use the Dice coefficient to measure how much a relevant user *v* is useful to a query *q*:

$$useful(q,v) = \frac{2|T_q \cap T_v^r|}{|T_q|+|T_v^r|} \quad (7)$$

If $useful(q,v) \neq 0$, then the relevant user *v* can give recommendations for *q*.

B. Friendship Establishment

Each user *u* exploits its gossip *local-view* to establish friendship. For each gossip cycle, *u* goes through each user entry $v \in local\text{-}view_u$, and evaluates whether *v* may be suggested for friendship as follows: User *u* computes the similarity distance *sim(u,v)* as described in Section III.A. User *v* is suggested to *u* for friendship under some conditions, taking into account the degree of similarity *sim(u,v)*, the $distance_{fri}(u,v)$, the $distance_{intr}(u,v)$, *useful(u,v)*, and *v*'s relevant topics, etc. If *u* has accepted to establish friendship with *v*, user *u* sends a message to *v*, denoted by msg_{req}, asking *v* for a friendship. Then, *u* adds *v* to a *waitList* list, waiting for friendship confirmation.

Afterwards, user *u* receives a reply message, denoted by msg_{rep}, from each user $v \in waitList$. If user *v* has accepted to establish friendship with *u* i.e., $msg_{rep} = accept$, *u* stores *v*'s information in its FOAF file. The information for the new friend *v* includes *v*'s relevant topics of interest, a trust value *trust(u,v)* between *u* and *v*, and link to *v*'s FOAF file. Notice that the *trust(u,v)* is assigned explicitly by *u* [8].

IV. Query Processing Based on FOAF File

In this section, we describe our query processing algorithm to generate recommendations. Next, we describe the ranking model we use to order the returned recommendations.

A query is defined as $q(word_i, TTL, V_q, T_q, trust_q(u,v), k)$, where $word_i$ is a list of keywords, TTL is the time-to-live value, V_q is query *q*'s topic vector. Query *q*'s topic vector, $V_q = [w_q^{t1},..,w_q^{tk}]$, is extracted using LDA. Then, query topic(s) $T_q \subset T$ are computed, where *q* is considered to belong to a topic $t \in T_q$ if its weight w_q^t in that topic exceeds a certain threshold (which is system-defined). The $trust_q(u,v)$ is the trust level between *u* and a responder *v*. The value *k* is the parameter for top-k redirection.

Each time, a user *u* issues a query *q*, it proceeds as follows: First, it computes how much each useful friend $v \in friend(u)$ is useful to *q*. Then, *u* computes the rank of *v*, denoted by *rank(v)*. The rank of a useful friend *v* for *u* depends on the usefulness of *v* for *q*, and the trust level between *u* and *v*. Accordingly the *rank(v)* is defined as:

$$rank(v) = trust(u,v) * useful(q,v) \quad (8)$$

Once *u* has computed the rank of each useful friend *v*, it adds *rank(v)* to a *RankList* that contains the useful friends' addresses along with their ranks. Then, it selects the top-k useful friends from the *RankList* with highest rank, and adds them to *topkList*. Then, *u* forwards *q* to each useful friend $v \in topkList$, attaching to *q* the trust value $trust_q(u,v)$, and reducing the query TTL by one. Note that the value of $trust_q(u,v)$ is equal to the value of *trust(u,v)*, because *v* is a direct friend of *u*. Also the useful friend *v* with the highest rank is the useful friend that is most useful to *q*, and has the highest trust level with *u*.

Once user *u* receives the recommendation information from the responders, it ranks those recommendations and presents them in an ordered list (see Section IV.A).

When a user *v* receives a query *q* that has been initiated by a user *u*, it processes *q* as follows: First, it measures the similarity between query *q* and each document *v* has locally. The similarity between a document *doc* and *q*, denoted by *sim(doc,q)*, is measured by using the cosine similarity between the document topic vector $V_{doc} = [w_{doc}^{t1},...,w_{doc}^{tk}]$ and the query topic vector $V_q = [w_q^{t1},...,w_q^{tk}]$, which is:

$$sim(doc,q) = \frac{\sum_{i=1}^{d} w_q^{t_i} * w_{doc}^{t_i}}{\sqrt{\sum_{i=1}^{d} w_q^{t_i} * w_q^{t_i} * \sum_{i=1}^{d} w_{doc}^{t_i} * w_{doc}^{t_i}}} \quad (9)$$

Second, *v* returns to the query's initiator *u* the recommendations for the documents whose similarity exceeds a given (system-defined) threshold.

Finally, *v* selects from its friends the top-k useful friends that have the highest rank, and adds them to the *topkList* if the query's TTL is not yet zero. Then, *v* computes the trust value $trust_q(u,x)$ for each useful friend $x \in topkList$ based on Equation 2. Then *v* attaches $trust_q(u,x)$ to *q*, and forwards *q* to *x* after reducing TTL by one.

With such query routing, we avoid sending *q* to all friends, thus minimizing the number of messages and network traffic for *q*. In addition, we send the query to friends that are most useful and trustful.

A. Ranking Recommendations

Recall that the result of a query *q* submitted by a user *u* is $recommendation_q = rank(rec_q^{v1}(doc_1),\ldots, rec_q^{v}(doc_i))$, where $rec_q^{v}(doc_i)$ is the recommendation that has been given for a document doc_i from a responder *v*. We rank $rec_q^{v}(doc_i)$ based on the semantic similarity between *q* and doc_i, the popularity of doc_i, and the distance and trust between *u* and the responders of doc_i. Accordingly, $rec_q^{v}(doc_i)$ that has been received from responder *v* includes $sim(doc_i,q)$, *v*'s topics of interest T_v and the $trust_q(u,v)$. The rank of a $rec_q^{v}(doc)$, denoted by $rank(rec_q^{v}(doc))$, is defined as:

$$rank(rec_q^{v}(doc)) = a * sim(doc, q) + b * \frac{\sum_{v \in R} distance_{intr}(u,v) * trust_q(u,v)}{|R|} + c * pop(doc) \quad (10)$$

Where *a*, *b* and *c* are scale parameters, *pop(doc)* is the popularity of *doc*, and |*R*| is the number of responders that have recommended *doc* to the initiator *u*. The popularity is equal to the number of replicas of *doc* in F2Frec. The user can specify whether it prefers highly popular documents, documents that are highly semantically relevant to *q*, or documents that come from highly similar users, by adjusting parameters *a*, *b* and c. Upon receiving the recommended documents, user *u* can download a copy of a document, rate and include it in its document set D_u.

V. EXPERIMENTAL EVALUATION

In this section, we provide an experimental validation of F2Frec to assess the quality of recommendations, search efficiency (cost, and *hit-ratio*), and the average number of friends. We conducted a set of experiments using TREC09 [10] and the Wiki vote social network [12]. We first describe the experimentation setup. Then, we evaluate the effect of friendship establishment on the performance of F2Frec. Finally, we compare F2Frec with centralized collaborative filter.

A. Experimentation Setup

We use the classical metric of *recall* that is used in information retrieval and recommender systems to assess the quality of the returned recommendations. Recall represents the system ability to return all relevant documents to a query from the dataset. Thus, in order to measure recall, the relevant documents set for each query that have been issued in the system should be known in advance. Data published by TREC have many relevance judgments. We use TREC09 filtering track [10], a set of 348566 references from MEDLINE, the on-line medical information database, consisting of titles and abstracts from 270 medical journals over a five year period (1987-1991). It includes also a set *Q* of 4904 queries. The relevant documents for each query $q \in Q$, denoted by R_q, were determined by TREC09 query assessors.

In the experiments, user *u* issues a query $q \in Q$ and uses F2Frec to possibly retrieve the documents that are in R_q. The set of documents returned by F2Frec for a user *u* and a query *q* is denoted by P_q. Once a user *u* has received P_q from F2Frec, it can count the number of common documents in both sets P_q and R_q to compute recall. Thus, recall is defined as the percentage of *q*'s relevant documents $doc \in R_q$ occurring in P_q with respect to the overall number of *q*'s relevant documents $|R_q|$:

$$recall = 100 . \frac{|P_q \cap R_q|}{|R_q|} \quad (11)$$

In addition we use the following metrics to evaluate F2Frec.

- **Communication cost:** the number of messages in the P2P system for a query.
- **Background traffic:** the average traffic in bps experienced by a user due to gossip exchanges.
- **Hit-ratio**: the percentage of the number of queries that have been successfully answered.
- **Average number of friends in the network:** the total sum of the number of friend of all users divided by the size of the network (total number of users).

We extracted the titles and abstracts of TREC09 documents and removed from them all the stop words (e.g., the, and,..). Then, we fed them to the GibbsLDA++ software [9], a C++ implementation of LDA using Gibbs sampling, to estimate the document topic vectors V_{doc}. With $|T|=100$ as the number of topics. To estimate the query topic vectors V_q, we removed the stop words from queries keywords, fed the query keywords left to GibbsLDA++, and computed the topics T_q of each query $q \in Q$. For ease of presentation, we consider that each query $q \in Q$ has one topic $t_q \in T$.

We use the Wiki vote social network [12] to give randomly each user a set of documents from TREC09. Wiki vote considers that two users are considered friends if one votes for the other. It consists of 7115 users connected together by 103689 links with an average of 14.57 links per user. After distributing the TREC09 documents over the Wiki vote users, we get a total of 6816170 documents, with an average of 958 documents per user.

We generate a random rating between 0 and 5 for each document a user has and compute the users' topics of interest from the documents they have rated. We consider that each user *u* is interested at least in one topic and relevant at least for one topic. Also *u* is interested in at most 10 topics and relevant for 5 topics at most.

F2Frec is built on top of a P2P content sharing system that we generated as an underlying network of 7115 nodes, which is equal to the number of users in the Wiki vote net-

 ISBN: 978-1-61208-173-1

work. We use PeerSim for simulation. Each experiment is run for 24 hours, which are mapped to simulation time units.

In order to evaluate the quality of recommendations, we let each user *u* issue a query after computing the previous query or after a system-specified timeout. Then we obtain the result for each query and compute the respective metric values. In order to obtain global metrics, we average the respective metric values for all evaluated queries. We let each user *u* establish new friends after each time it performs a gossip.

B. Experiments

We first investigate the effect of friend establishment on the performance of F2Frec over the respective metrics. Second, we compare F2Frec with a centralized collaborative filter. For the gossip parameters (gossip period C_{gossip}, gossip message L_{gossip}, and *view-size*), we use 30 minutes for C_{gossip} (simulation time units), 10 for L_{gossip}, and 50 for *view-size* (in [3] we showed that this setting provided good quality of recommendations with acceptable network traffic). We use 1 for the TTL of the query, and query is forwarded to each friend *v* that is useful to query, in order to measure the quality and effectiveness of friendship establishment.

Then, we collect the results for each experiment after 24 simulation hours. We set TTL to 1 to measure the quality and effectiveness of friendship establishment. All experiments are performed under churn i.e., the network size is changed during the run due to the joining and leaving of users. The experiments start with a stable overlay with 355 users. Then, as experiments are run, new users are joining and some of the existing users are leaving.

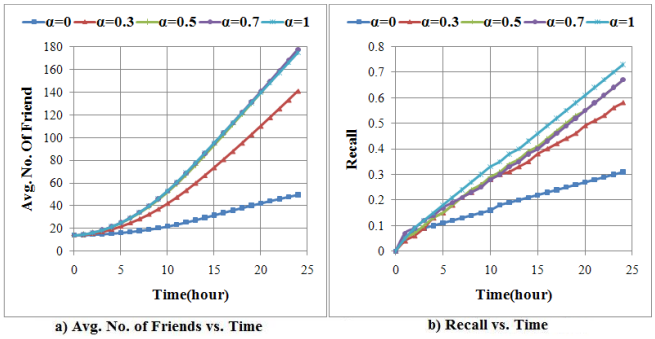
Friendship Establishment. In this experiment, we vary the value of α between 0 and 1, in order to investigate the trade-off of usefulness and friendship distance based on the Equation 6. In addition, we investigate the effect of friendship establishment on the performance of F2Frec over the respective metrics. In each experiment, each user $u \in U$ gets its initial friends from the Wiki social network, then *u* runs the F2Frec algorithm to establish new friends.

Table 1 shows the results obtained after 24 hours of running the F2Frec algorithm. We can see that the average number of friends increases from 49.7 to 174.6 when increasing α from 0 to 1. Combining users' usefulness with friend networks increases the likeness between users. Thus more new friends are added to users' FOAF files. We also observe that recall, communication cost, hit-ratio and background traffic are correlated to the average number of friends. The communication cost increases because more useful friends are visited. Visiting more useful friends increases the relevant documents returned, and thus greater recall is achieved. Also, hit-ratio increases as long as the average number of friends also increases, because there is a higher probability to find a useful friend to serve a query. However, bandwidth consumption increases because increasing the number of friends implies the increase of the size of the gossip entries, increasing the size of the gossip messages. As a result the bandwidth consumed is increased.

In Figure 1, we show the variation of average number of friends and recall versus time under different values of α. We observe that combining the usefulness of users with friendship networks increases the possibility of finding new friends (Fig. 1(a)). When the value of α is equal to 0, the final friendship establishment depends on the overlap between users' friends only. This depends on the density of the links in the network graph. In our benchmark, the overlap between friend networks is low, and thus the average number of friends is low, which causes low recall. However, the recommended documents in this case have more confidence and quality, and users are more satisfied with those recommendations. This is because they are recommended by trusted friends.

TABLE 1. RESULTS OBTAINED BY F2FREC OVER THE RESPECTIVE METRICS

α	Max. recall	Max. Com. cost	Max. Hit-ratio	Max. Avg. background traffic (bps)	Max. Avg. Friend
0	0.31	20	0.61	12.4	49.7
0.3	0.58	38.3	0.94	17.4	141.1
0.5	0.67	46	0.977	19	177.6
0.7	0.67	47	0.98	18.7	177.6
1	0.73	46.5	0.98	18.5	174.6



a) Avg. No. of Friends vs. Time b) Recall vs. Time

Figure 1. F2Frec performance over respective metrics

When the value of α is equal to 1, friendship establishment depends on the usefulness of users only. Each time a user *u* performs gossip, new relevant users are added to its *local-view*. Thus, *u* finds new relevant users that are useful to its demand, and then establishes friendship with them. Therefore, more friends are added at *u*'s FOAF file. As a result, the average number of friends is increased. While the values of α increase between the two extremes, *u* finds new relevant users that are useful to its demand, and establishes friendship with them. Accordingly, its friend list is increased. Then, the possibility of overlap between users' friends increases as well. As a result, the possibility of establishing new friendship increases.

We observe that the recall achieved by α=1 is greater than that with α=0.7 or 0.5, even though the average number of friends are almost identical (Fig. 1(b)). When α=1, friendship establishment depends on users usefulness only. Accordingly, each user *u* establishes new friendship with relevant users that are more useful to its demands.

For the other simulations, we set α=0.5, because this setting leverages users' usefulness and friendship networks, and provides reasonable results with acceptable overhead in terms of background traffic.

 ISBN: 978-1-61208-173-1

Social Effect. We compare F2Frec with a centralized collaborative filter [1] with respect to recall. In order to recommend a set of documents that a user *u* may like, we compute the similarity between a user *u* and all the users in the system. Then, we select a set of users, noted *neighbors(u)*, which are the top-k similar to *u*. We use the cosine similarity to extract a user's *neighbors*, based on the ratings that are given by the users over the documents they have seen or created. Once the similarity between *u* and each user *v* has been computed, we select the top 178 similar users as the *neighbors(u)*. Once users' *neighbors* are extracted, we run the system and generate recommendations for each user from its *neighbors*, and then compute the average recall of all users. The recommendations for each user *u* are generated as follows: First, *u* randomly selects a query $q \in Q$ s.t. $t_q \in T_u$. Then *u* forwards *q* to each member in its neighbors. Each *neighbor* receives *q*, returns to *u* all the documents that their similarity with *q* exceed 0.5. We select the top 178 similar users as the *neighbors(u)*, and $t_q \in T_u$, to be identical with F2Frec.

We observe that the similarity measure is time consuming as it takes about 38 hours to compute the similarity between the 7115 users with 6816170 documents. This time increases exponentially as the numbers of documents and users increase.

Results. We observe that the average recall achieved by collaborative filter is equal to 0.076. F2Frec increases the recall by a factor of 8.8 in comparison with the centralized collaborative filter. A major reason behind this significant gain is the friendship establishment in F2Frec that relies on usefulness of users and friend networks. In contrast, centralized collaborative filter aggregates neighbors based on document ratings only. Unfortunately, this kind of similarity does not capture the contents of the documents.

VI. Related Work

In this section, we discuss the previous works that are most related to F2Frec. Previous works such as [3] and [5] exploit specific gossip protocols to aggregate the *neighbors* of each user. Then users use those *neighbors* (of *neighbors*) to serve their demands. However, these systems do not exploit users' social data and explicit friendship for recommendations.

In [6] and [8], users' social data (friends, trust, etc.) are exploited in computing recommendations. In [6], users' preferences are extracted from user's behaviors and asserted in users' FOAF files. Then the system aggregates users' FOAF files and clusters the users with similar preferences in one group. Then, when a user *u* in a group rates an item, the system determines whether the item is good enough to be recommended to other users in the group. However, aggregating users' FOAF files increases network traffic. In contrast, F2Frec lets each user maintain locally its FOAF file, and uses it to support its queries.

In [8], trust in users is also used as a basis for recommendations. The system lets each user *u* express its level of trust to each user it has interacted with. Then *u* measures the trust level between itself and each user in the network, and selects the most trustful as its neighbors. Those neighbors are used to compute the recommendations. However, inferring the trust relationships between users is time consuming and increases network traffic. In contrast, F2Frec computes the trust between indirect friends during query processing. Thus we do not need extra data and information to propagate and aggregate the trust network.

VII. Conclusion

In this paper, we proposed F2Frec, a P2P recommender system that leverages content and social-based recommendations by maintaining P2P social networks. The basic idea of F2Frec is to exploit the users' relevant topics of interest and friends' networks, in order to get high quality recommendations. F2Frec relies on gossip protocols to disseminate relevant users and their information, in order to let users establish friendship with new useful friends.

We use FOAF files to store users' friendship networks and their relevant topics of interest, and as a directory to redirect a query to the appropriate trustful and useful friends in a top-k approach.

In our experimental evaluation, using the TREC09 dataset and Wiki vote social network, we showed that F2Frec increases recall by a factor of 8.8, compared with centralized collaborative filter.

ACKNOWLEDGMENT

We would like to thank Bettina Kemme for her insightful discussions.

References

[1] J.-S., Breese, D., Hecherman, and C., Kadie, Empirical analysis of predictive algorithms for collaborative filtering. *Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence*, 1998, pp. 43–52.

[2] D.-M., Blei, A.-Y., Ng, and M.-I., Jordan, Latent Dirichlet Allocation. *Journal of Machine Learning*, 2003, vol. 3, pp. 993–1022.

[3] F., Draidi, E., Pacitti, and B., Kemme, P2Prec: a P2P Recommendation System for Large-scale Data Sharing. *Tran. on Large-Scale Data- and Knowledge- Centered Systems, LNCS*, 2011, vol. 6790, No. 3, pp. 87-116.

[4] M., El Dick, E., Pacitti, R., Akbarinia, and B., Kemme, Building a peer-to-peer content distribution network with high performance, scalability and robustness. *Information Systems*, 2011, vol. 36, No. 2, pp. 222-247.

[5] A.-M., Kermarrec, V., Leroy, A., Moin, and C., Thraves, Application of Random Walks to Decentralized Recommender Systems. *OPODIS*, 2010, pp. 48–63.

[6] H.-J., Kim, J.-J., Jung, and G.-S., Jo, Conceptual framework for recommendation system based on distributed user ratings. *LNCS*, 2003, vol. 3032, pp. 115-122.

[7] L., Lacomme, Y., Demazeau, and V., Camps, Personalization of a trust network. *IEEE/ACM*, 2009, pp. 408–415.

[8] P., Massa and P., Avesani Trust-aware Collaborative Filtering for Recommender System. *LNCS*, 2004, vol. 3290, pp. 492-508.

[9] X.-H., Phan, October 2011, http://gibbslda.sourceforge.net

[10] S., Robertson and D.-A., Hull, The TREC-9 filtering track final report. *Proc. of 9th Text REtrieval Conf. (TREC-9)*, 2001, pp. 25-40.

[11] R., Sinha and K., Swearingen, Comparing Recommendation made by Online Systems and Friends. *Proc. of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, 2001

[12] Wikipedia vote network, October 2011, http://snap.stanford.edu/data/wiki-Vote.html

 ISBN: 978-1-61208-173-1

Web Service and Business Process Execution on Peer-to-Peer Environments

Marco Pereira, Marco Fernandes and Joaquim Arnaldo Martins
DETI - Department of Electronics, Telecommunications and Informatics
IEETA - Institute of Electronics and Telematics Engineering of Aveiro
University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal
{marcopereira, marcopsf, jam}@ua.pt

***Abstract*—Service oriented environments and peer-to-peer networks are on the forefront of research. This paper addresses the issues that arise when attempting to integrate these technologies, while at the same time makes explicit the benefits that can be gained from this integration. We propose the creation of a proxy for web services that allows the deployment of multiple instances of the same traditional web service in a peer-to-peer network. This proxy handles service discovery in the peer-to-peer network and can be used by existing clients with no modifications, thus offering a transparent way access to resource replication and decentralisation benefits that are traditionally associated with peer-to-peer networks. We then proceed to adapt a business process execution engine to be peer-to-peer aware, allowing the implementation of process partition and delegation techniques that can result in reductions in the network traffic required to execute a business process, as well as in a more efficient distribution of the service load through available peers.**

Keywords-Peer-to-Peer; Web Services; Service Oriented Architecture; Business Process Management.

I. INTRODUCTION

Access to computational resources is a key requirement of most modern organisations. This requirement arises from the need to produce text documents, to process employees' salaries, or to provide complex services. The typical response to this requirement leads to the proliferation of hardware throughout organisations, and even among individual users. While some of it is specialised hardware (such as dedicated servers) most takes the form of personal computers that are mainly used to perform simple tasks, wasting potential computational resources in the form of processing cycles and storage space. Using peer-to-peer (P2P) technology it is possible to tap into these otherwise wasted computational resources [1]. A possible use for these "*recovered*" computational resources is service deployment, particularly web services. If web services are themselves seen as resources it is possible to apply to them the same philosophy that is applied to files in traditional P2P networks, where availability and resilience is improved by the existence of multiple copies distributed throughout the network. By having multiple copies of a web service present in a P2P network one can avoid what can be seen as a potential centralised point of failure and provide an alternative way to implement a Service Oriented Architecture (SOA) [2]. While SOA is one of the most popular research topics, and has been a driving force in the software industry [3] to fully explore its advantages it is not possible to ignore the importance of service orchestration, that can be used to create composite services from the individual services available, thus creating a business process. The usage of SOA, business process orchestration and web services can bring numerous advantages for organisations [4] such as higher automation and process integration. Unfortunately the tools that perform service orchestration are not expecting the services to be available from multiple providers as it would be the case if the services were available as resources in a P2P network and do not take advantage of this fact. In this work we describe how to use a P2P network in combination with orchestration tools in order to reap benefits from making web services behaviour more akin to the one exhibited by files in file-centric P2P networks.

This work focuses on the development of integration strategies that explore the synergies that exist between web services, business process execution and P2P environments. We believe that better integration between these technologies will lead to improved performance and added robustness when executing business processes or individual web services. These improvements are achieved by using process delegation to reduce the overall network traffic generated by the execution of a business process and by allowing the replication of individual web services through multiple peers to ensure that a service can be executed even if some of the service providers become unreachable.

While designing a P2P based service-oriented environment we have established a few pre-requisites. First unless absolutely required existing standards should be used. Using standards compliant approach will enable us to accommodate already existing services, clients and business processes within the P2P environment, and does not impose any additional burden to developers. Second we make no particular assumptions about the underlying network. This means that our environment should transparently accommodate different topologies and be able to execute the services and business processes in a non-optimised fashion if the available peers do not offer specific capabilities.

The structure of this work is as follows: in Section II we review existing related work while in Section III and Section IV we describe how web services and business process execution can be made P2P-aware while the benefits and caveats that

 ISBN: 978-1-61208-173-1

can follow from having P2P-aware business process execution environment are analysed in Section V with a case study. Finally in Section VI we present the conclusion of this work and explain our plans to further improve the presented work.

II. Related Work

Exposing services as part of a P2P network can be seen as one of the achievements of the JXTA framework [5]. The JXTA framework provides the necessary protocols to create a P2P overlay network, establish connections between peers, and to discover resources in the network. JXTA is able to create unstructured P2P overlays that can be configured to form either a pure P2P system that resorts to flooding (using multicast where available) to perform network queries or an hybrid P2P system where queries can be directed to infrastructure peers (in addition to any local caching performed by client nodes). It should be noted that the use of an hybrid topology is mandatory in case the actual overlay needs to be extended past any type of network boundaries (such as NATs or firewalls). In JXTA every resource (be it a communication channel, a peer or a service) is represented by an advertisement [6]. An advertisement is a small XML document with information about a particular resource that possess a pre-determined lifetime that will expire if not explicitly renewed during that lifetime. The first step to locate a resource in a JXTA based P2P network is always to discover a corresponding advertisement by querying the network or the local cache. Of particular relevance is a family of advertisements, *Module Advertisements* that can be used to represent and discover services. This family possesses two advertisement types, *Module Specification Advertisement* and *Module Implementation Advertisement*, representing respectively the expected behaviour and protocol of a given service and a concrete implementation of the corresponding service. It should be noted that although the publication of a *Module Specification Advertisement* is optional, its publication is considered a good practice and has an advantage over the publication of a *Module Implementation Advertisement* alone. The advantage is that a *Module Specification Advertisement* is allowed to carry within it a *Pipe Advertisement*, which can be used to locate a peer that is able to execute the service. As it would be expected, services constructed in this way are deeply intertwined with the P2P network and difficult to expose to the outside, thus creating an impedance mismatch when trying to use them in SOA.

Another project, also based on JXTA took a different approach. Instead of creating pure JXTA services, JXTA-SOAP [7] allowed developers to create web services, that can then be deployed in the P2P network. Services created with JXTA-SOAP can be reached within the P2P network by discovering a *Module Specification Advertisement* that contains the WSDL of the web service and a *Pipe Advertisement* to contact the peer to execute the service. JXTA is used as a transport protocol instead of HTTP (handled automatically by the JXTA-SOAP library). Developing a service using the JXTA-SOAP approach requires the service to be developed in Java, and the implementation of a specific interface. It also requires the service to be deployed using the first generation of the Apache Axis platform [8]. As with native JXTA services, services developed with JXTA-SOAP are also difficult to expose to the outside.

The default approach to web service discovery is to rely on UDDI (Universal Description, Discovery and Integration protocol). UDDI provides a centralised broker that can be queried by a client to discover a provider of a given service, yet this centralised approach creates a single point of failure. To tackle this problem it was proposed in [9] that UDDI brokers could be federated using a P2P approach, where each UDDI broker acts as a super-peer for a group of peers that have shared interests (in this case they either require or provide similar services). Replacing UDDI with a completely decentralised P2P approach was proposed in [10]. In this scenario peers publish a semantic description of each provided service (based on OWL-S). When queried, the network can return the description to a particular service (or a semantically equivalent one that is currently available) by automatically producing a service composition described using Business Process Execution Language (BPEL). The resulting service composition can be later used in any application as a regular web service.

The issue of web service replication is approached in [11]. This work assumes an ad-hoc network scenario (similar to P2P networks) where frequent node disconnections and failures make traditional static binding unreliable. To increase the reliability and availability of services in those types of networks, it introduces an active monitoring scheme, based on a global view of the network that can be used to determine if a service is still available. To cope with expected service failure it allows dynamic deployment of replicas of web services (the web services must be Java based). To invoke a service a node must at first discover an available instance of the service (it is stated that this a responsibility of the client, not of the system and the suggested means to achieve this are described in [12][13]). After discovering this initial instance, it passes it to a tool called "*WSDL-finder*" that must be called every time before the service is actually invoked in order to discover and invoke the service from an available replica.

An alternative to the use of P2P networks for service replication can be found in SmartWS [14]. It relies on client side "*smart proxies*" that intercept the original web service call and redirects it to a service provider that at the moment offers optimal performance (based on a series of tests). All the service providers must be known before generating the client side "*smart proxy*", which means that any new provider that appear after the proxy creation will not be taken into consideration.

On the business process side there have been several proposals that attempt to leverage the existence of multiple distinct providers, particularly of the orchestration engine itself. The proposed techniques can be applied to business process described using BPEL, and mainly deal on how to divide the business process in order to distribute the execution of a single business process by multiple BPEL engines in order to improve throughput. To achieve this goal it was proposed

 ISBN: 978-1-61208-173-1

in [15] the partition of the BPEL instruction sequence into a set of distributed processes (that can be reordered, but whose final output will always be the same as the original sequence). BPEL activities are divided into fixed (receive, reply invoke) and portable, where each fixed activity is aggregated with a process service (receive/reply pair with the entry point) and the portables can be moved. This approach allows the automatic extraction of parallelism from the flow activity and results in partitions containing one fixed activity and zero or more portable ones. According to the authors this approach leads to a projected throughput increase of 30% under normal system load and by a factor of two under high load, yet it assumes that every service-providing node has BPEL runtime capabilities.

Another approach can be found in [16] where it is proposed to decentralise the flow control and dynamically select the role that a given node should take. After executing an activity each node transfer all the generated state information to the following node (thus the participating nodes can be seen as stateless). It should be noted that this approach allows the dynamic discovery of business partners, yet it still requires the presence of a BPEL engine in every node and only considers simple flows without any type of synchronisation, restrictions or error handling. Alternative approaches to business process partitioning can be found in [17][18]. These works propose extensions to the existing BPEL standard in order to make the data flow (expressed in the form of shared process wide variables) as explicit as its control flow. Business processes using the proposed extensions can be partitioned taking into account both control requirements and shared data requirements. It should be noted that the partitioning process takes place before deploying the business process for execution.

III. Web Services in P2P Environments

Our starting point is a previously developed P2P framework designed to support digital libraries [19]. The P2P component is based on a JXTA unstructured hybrid overlay comprised of both infrastructure peers and client peers. In this overlay infrastructure peers are responsible for gathering advertisements that are sent periodically by client peers and other infrastructure peers. Client peers can contact infrastructure peers in order to discover and receive new advertisements that are stored in a local cache. When client peers need to locate a resource they first issue a query to their local advertisement cache and only in the event of not having a matching advertisement they issue the query to infrastructure peers. It was decided that the services should be standard web services, given the ubiquity and consequent familiarity of that technology. This decision lead us to use JXTA-SOAP to provide the bridge between web services and the P2P network, in an attempt to avoid exposing the details of the P2P network directly to service developers. While this initial approach allowed us to take advantage of JXTA built-in resource discovery mechanism (advertisements), it also possessed a number of shortcomings. Its use of JXTA-SOAP created a technological limitation to service developers by requiring that services to be deployed using Apache Axis. Additional requirements of the creation of services using JXTA-SOAP include the need to implement a specific interface and the creation of service descriptor (one for each service), which requires details from the P2P network itself, thus breaking the illusion that developers are creating standard web services. Furthermore the services created were only available within the P2P network, and making them available to the outside required a manually created per service proxy.

Given the limitations of our initial approach the need for a complementary solution became clear. Instead of creating our services inside the P2P network we decided that it would be better if we created regular web services. This allows developers to design new services without having to worry about implementing specific interfaces to allow the network to be service aware while also freeing them from having to create services using a particular technology or application server. In order to make the network aware of the existence of these external web services (some whose access might only be possible from the *localhost*) the local P2P client can be configured to fetch the WSDL service descriptions from either a set of addresses or a system folder. The same local client can be deployed and configured to run in tandem with already existing web services, exposing them to the P2P network. Services are then described using a specially crafted advertisement (based on JXTA *Module Specification Advertisement*) that uses the information available from the service WSDL. This advertisement will carry three pieces of information that allows the identification of both service and providing peer: its namespace, methods and address. Two distinct peers can deploy a copy of the service in their own application server and automatically generate advertisements for each service from its corresponding WSDL. Two advertisements will describe the same service if they possess the same namespace and method collection. With this strategy web services' clients are created in the traditional way, and if nothing is done we could run the risk of creating bindings that use the same service-providing peer, ignoring any replica present in the P2P network. To avoid this we chose to create a transparent web services proxy. Each peer can be configured to provide a small HTTP server whose main task is to capture SOAP messages. These messages have the required information (namespace and method) to locate a service in the network by searching for its advertisement, generating a list of potential providers. Another task of the HTTP server is to publish a modified version of the original WSDL of each service. This modified WSDL is identical to the original except that the <*soap:address*> of the binding will point to an address configured in the web service proxy instead of pointing to the service location directly, thus ensuring that clients generated from this modified WSDL version will be transparently using the P2P network. Assuming that previously deployed public services remain public, clients generated from the original WSDL will not be affected, ensuring that legacy applications will continue to work while still offering a clear upgrade path. As was said before each peer can be configured to act as a proxy regardless of the existence of other peers that are performing the same task. This enables us to provide multiple entry points into the P2P network or even to apply

 ISBN: 978-1-61208-173-1

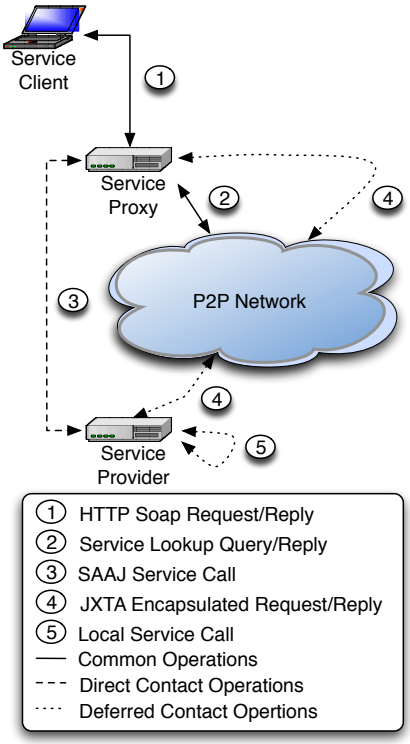


Figure 1: Direct contact and deferred contact.

load balance techniques between entry points.

It should also be noted that peers that act as service proxies have two distinct service invocation methods that they can use: direct contact and deferred contact (as seen in Figure 1). Direct contact can be used when the service is directly accessible using the standard HTTP protocol, while deferred contact first transfers the raw SOAP message that the service proxy received to the peer that is actually going to perform the service using the P2P network. The decision to use one or another method (illustrated in Figure 2) depends of whether a service can be reached using the standard HTTP protocol or not, and determining this requires resolving the service address. A successful resolve indicates that the service is directly accessible and is available, while a failure can indicate that the service provider is no longer available, or that the service is only accessible from the P2P network. Since both types of failures are indistinguishable we make the assumption that the requested service will be available from the same provider through the use of the P2P network (thus adopting a deferred contact strategy). If this leads to a new failure while contacting the peer, a new one can be selected from those that offer the same service.

A disadvantage of this approach when compared with both native JXTA services and JXTA-SOAP based services is that it will require manual service replication. JXTA services were designed to be portable across machines that are running the same JXTA version (JXSE or JXTA-C), with their runtime requirements being described in *Module Advertisements*. A similar philosophy can be applied to JXTA-SOAP services,

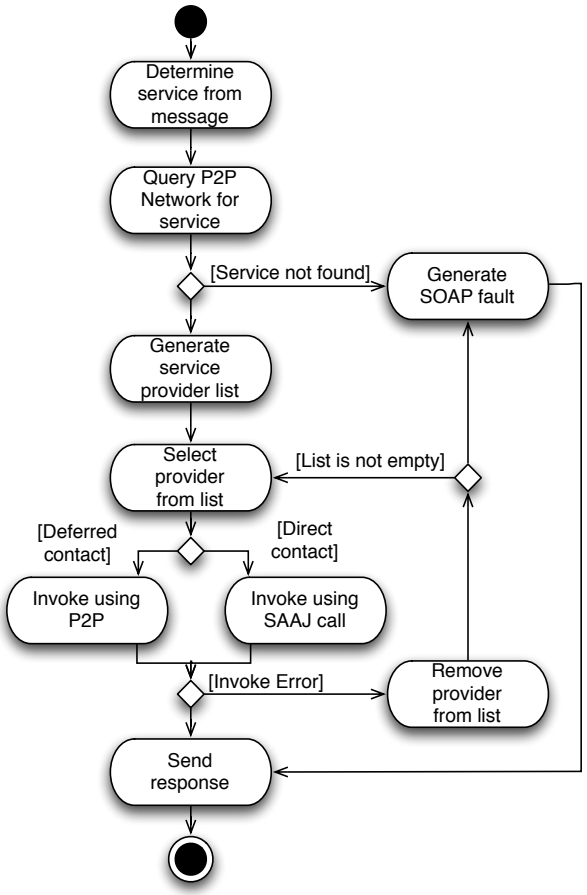


Figure 2: Decision model.

since their base requirements (an Axis application server and java based JXSE JXTA version) is known a priori, and their runtime dependencies can be bundled in a package. With our proposed approach the only pieces of information that we have about a service is its WSDL and providing peer. Since we have no information about the application server that they require, or about their runtime dependencies it is not yet possible to devise an appropriate and completely automated service migration/replication policy. On the other hand our strategy does not mandate the use of any specific technology for the creation of services, unlike previously referenced approaches [6][7][11].

IV. Business Process Execution In P2P Environments

While the strategy detailed in the previous section regarding web services deployment in P2P environments can be used transparently in the context of business process execution, we believe that a BPEL engine can benefit from the fact of being P2P-aware. The most obvious benefit is that it can use the service discovery mechanism directly, avoiding having to go through the web services proxy for each service that it intends to invoke from the P2P network. Having direct access to the discovery mechanism means that the BPEL engine can act as a simple load balance mechanism, exploring service

 ISBN: 978-1-61208-173-1

replication to avoid sending too many service requests to the same service-providing peer. This can help to alleviate the fact that the traditional approach to business process execution is a centralised one, in which service calls are dispatched to partner links (usually generated from a WSDL and thus bound to a single provider) and where state is centrally managed, by replacing those pre-bounded calls with dynamically discovered service providers. Going one step further, the ability to execute a BPEL process is in itself a service, which can be replicated and advertised by the P2P network. It should also be noted that the composition of multiple services executed by a BPEL engine is itself exposed as a web service described by a WSDL. This fact can be explored in a P2P environment in several ways: by replicating business processes (seen as regular web services) throughout the network; by allowing peers that provide an entry point into the P2P network to perform load balance between multiple composite service providing peers and more importantly by realising that the network may have multiple peers that can provide BPEL execution as a service. Being able to discover other available BPEL engines opens the door to distributing the orchestration process through multiple service providers (as opposed to execute the entire process in a single centralised provider). Distributing the orchestration through multiple peers has several advantages, particularly in high load scenarios or in scenarios where there is the need to transfer large amounts of data between service providers and consumers. Achieving this goal requires a careful partitioning process in order to reduce the number of messages and the amount of data transferred, thus increasing throughput.

Regarding the partitioning process, previous work assumes that every partner node will have BPEL capabilities, which in a P2P network designed to take advantage of already existing computational resources might not be the most convenient approach. It is possible to safely alleviate this assumption when using a BPEL engine that is P2P aware, since before executing a business process we can discover not only the required service providers but also any other available BPEL engines (since BPEL execution itself is a service). If no other engine is found then business process execution will proceed in the traditional centralised way, yet if one or more engines are found the BPEL process definition can be partitioned and parts of it delegated to other engines. If those engines are themselves P2P aware it is possible to continue the partition process. It should be noted that the absence of this "BPEL in every node" assumption means that some of the previously proposed partition mechanisms can not safely be applied to this scenario, yet some of the previously proposed design principles remain valid. When there is a parallel execution (a flow activity), an entire branch can be still be partitioned if the first invoke service activity exists at a BPEL-capable peer. Furthermore having access to the service discovery mechanism means that we can eventually use information about the services themselves both to decide what will be the more adequate service provider to use and to infer the best tasks to be delegated.

As was asserted before, process delegation has the potential to greatly reduce the amount of data that must be transferred through the network, mainly by eliminating round trips in the invocation calls. Since this effect can be seen more clearly when delegating services that require the transmission of large message or variables (particularly large response messages), our main concern should be to provide a way to identify those types of service. While there is no standard way to know a priory which services will generate a large response message, we can use the return type as a telltale of those services. It is safe to assume that any efficiency gain will likely be much smaller when delegating the process if services are going to return an integer when compared with services that return an array of bytes. As such we suggest the usage of a simple rule: perform no process delegation if the next service return messages with simple types (numeric, boolean, strings) or complex types based on these types.

In line with previous work inner process delegation presents some difficulties when dealing with process monitoring. While keeping track of the progress of a business process in a centralised scenario is a simple task, doing so in a decentralised orchestration environment is not as trivial. This is a non-critical issue that only occurs for BPEL engines that support process delegation; nevertheless one should be aware of this limitation. Furthermore the delegation of branches that contain shared process variables can also become a source of problems.

V. Case Study

We present as an example the case of a digital newsstand website that allows registered users to view a range of newspapers as they are published. The website receives PDF files from the publishers which need to be converted into an image format (in the case JPEG) for display purposes and whose text must be extracted for search purposes. As part of the submission process there is the need to invoke several services (image conversion, resize and white space cropping, text extraction, OCR and storage). The sequence of services to be performed can be organised as a business process (a functional diagram of it can be found in Figure 3). The initial input of this process is a PDF file and an XML document with associated metadata. The process starts with two parallel branches. The first branch extracts text from the initial PDF, while the second branch converts the PDF into an array of PNG files and crops the white space around the generated PNG files. From this point on the process once again splits into two parallel branches, one that is responsible to convert the PNG files to TIFF format (the OCR service requires that the input files to be in TIFF format) with the resulting files being fed to an OCR service. Meanwhile the other branch converts the PNG files to the final JPEG format (with the appropriate screen resolution). The final activity consists on the use of a service that will store all non-intermediate files that were generated by the process.

In the worst case scenario each of the blocks in Figure 3 represents a service in a different peer. In a centralised orchestration this represents a significant amount of data that must be sent through the network. The total amount can be calculated by $T = 3S_{PDF|} + 5S_{PNG} + 2S_{TIF} + 2S_{OCR} +$

 ISBN: 978-1-61208-173-1

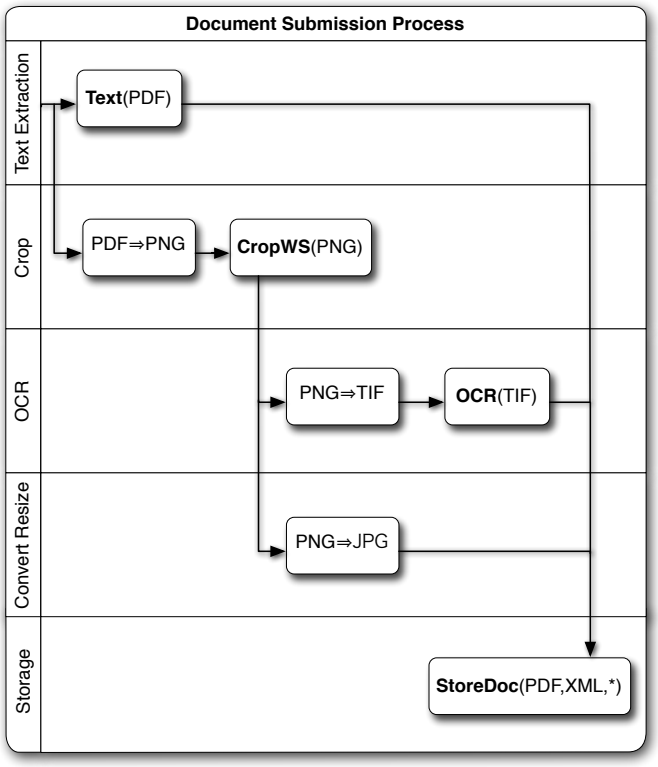


Figure 3: Functional diagram of the document submission process.

$2S_{TXT} + 2S_{JPG} + S_{XML} + S_{ID}$ where S_x represents the message size of the transmission of X. In a branched process such as this one it is possible to perform a simple optimisation by delegating an entire branch of activities. If one of the peers that provide an image conversion service also provides BPEL execution capabilities, we can reduce data that must circulate through the network by delegating all the activities in the "OCR" band of Figure 3. The call to the image conversion service would be a local one, thus avoiding sending the intermediary TIFF files that result from the image conversion back to the original caller trough the network. In this particular digital newsstand application the intermediary TIFF are about 3MB each, which for a 40-page newspaper would result on not having to send 120MB of data through the network if this optimisation is applied.

In an optimal scenario where all peers have BPEL execution capabilities it would be possible to apply the partition algorithms that were previously mentioned in Section II. It should be noted that some delegation could prove to be counter-productive. If there were services just before the storage service, dedicated to provide unique identifiers, produce checksums or calculate hashes based on the metadata of the new document, delegating the orchestration of one of those services and the storage to those providers would actually increase the network usage since instead of invoking the first service, receive its results and send everything to the storage service, both the initial PDF and final JPEG images would have to be sent first to the new service provider and only after to the storage service. In this scenario instead of transferring $T_{final} = 2S_{XML} + 2S_{ID} + S_{PDF} + S_{JPG}$ we would be transferring $T_{final} = 2S_{XML} + S_{ID} + 2S_{PDF} + 2S_{JPG}$. If we apply the criteria that was proposed earlier, since the id/checksum service return type will probably be of a simple type (that we can safely predict to be small when compared with byte arrays that hold the original PDF or JPEG files) no delegation would occur, thus avoiding generating extra network traffic. Other optimisations could be considered, such as trying to merge activities in peers that provide multiple consecutive services. This optimisation could greatly reduce network traffic but it would be difficult to analyse its beneficial impact if factors such as throughput were also to be considered. In this particular example it could also be possible to further explore the P2P network by using it as a storage medium, which would allow the storage service to be executed by any available peer.

VI. Conclusion and Future Work

In this work we have discussed and implemented strategies to better integrate web services, SOA, and business process execution in peer-to-peer environments. Our proposed strategy allows the deployment of replicated web services in multiple peers without requiring any major change to the services themselves or to the clients. It accomplishes this with the use of a small proxy that allows access to services hosted on the P2P network to clients that are not aware of the presence of the P2P network providing the following benefits:

- Does not require modifications to existing services or clients.
- Does not mandate a specific technology for the development or deployment of new services.
- Provides a way to tap into otherwise wasted computational resources.
- Transparently manages access to replicated web services.

While it can be argued that the adopted strategy adds an additional step that has the potential of slowing the access to a given service in low load scenarios, it also has the potential to shield clients in high load scenarios, provided that multiple peers provide an entry proxy to the P2P network and that popular services are properly replicated. Business process execution engines can use the replicated services transparently, providing an added layer of reliability to business process execution, yet a P2P-aware business process engine can obtain the following additional benefits:

- Delegate parts of a business process to other engines.
- Directly select service providers to distribute workload.
- Reduce the amount of data that must be sent through the network.

Of the previously stated benefits, process delegation and reduced network data transfers depend directly on the existence of additional business process execution engines in the P2P network. Since we are able to discover them in runtime we can alleviate the "BPEL in every node" assumption present

 ISBN: 978-1-61208-173-1

in previous works [15][16] and avoid the need to perform the partition process before actually deploying the business process [17][18] thus ensuring that there is an "always working" solution for the execution of the business process, even when no other business process execution engines are available in the P2P network.

A point to be improved in the service discovery mechanism is that the current approach is still based on the traditional WSDL, which only provides a syntactic description of the service. While this description provides enough information to discover and execute a service based only on an incoming request, the use of a semantic description would enable more refined queries. One of the goals of a semantically improved discovery mechanism would be to further improve service execution resilience by allowing the exchange of a missing or faulty services with semantically equivalent ones (or composition of multiple services if applicable) in an automated way (a similar approach can be seen in PANIC [20]). This requires knowledge about what the service does that cannot be obtained from the current WSDL description, but might be available with the introduction of WSDL-S or OWL-S. The additional knowledge gathered about each service could also be used to improve peer selection process, and open the way to more efficient process delegation strategies. Two peers can be providing the exact same set of services, yet due to differences in hardware the performance obtained from each one can be very different, making one of those peer a less desirable choice to perform some classes of services. As an example a storage service would benefit from being executed on a peer with more available storage space while a video conversion service would benefit from being executed in a peer with dedicated encoding hardware. By taking into account the requirements of the service when selecting the service-providing peer it is possible to promote an even more rational use of available hardware resources. It should be noted that having advanced peer/service selection algorithms is a an important step that to achieve further performance gains, and is an important research topic [21][22][23].

As was said before, the proposed P2P service discovery mechanism assumes that a web service is going to be described by a WSDL. While this assumption holds true for SOAP based web services, it collapses when dealing with REST services. This has a double impact since it prevents the use of REST services in BPEL processes and prevents proper integration of REST services with our P2P network. Taking into account the work described in [24] where REST services have been described in WSDL and in [25], where REST services were composed into BPEL processes (with the use of extensions) we believe that it will be possible to support REST services in parallel with SOAP based web services using our proposed P2P architecture with only minor modifications.

ACKNOWLEDGEMENT

This work was funded in part by the Portuguese Foundation for Science and Technology grant SFRH/BD/62554/2009.

REFERENCES

[1] I. J. Taylor and A. Harrison, *From P2P to Web Services and Grids. Peers in a Client/Server World*. Springer, 2005.

[2] E. A. Marks and M. Bell, *Service-Oriented Architecture: A Planning and Implementation Guide for Business and Technology*. John Wiley & Sons, June 2006.

[3] M. Bichler and K.-J. Lin, "Service-oriented computing," *Computer*, vol. 39, pp. 99–101, 2006. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MC.2006.102

[4] O. Zimmermann, V. Doubrovski, J. Grundler, and K. Hogg, "Service-oriented architecture and business process choreography in an order management scenario: rationale, concepts, lessons learned," in *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, ser. OOPSLA '05, 2005, pp. 301–312. [Online]. Available: http://doi.acm.org/10.1145/1094855.1094965

[5] JXTA Community Board, "Jxta homepage," 2010. [Online]. Available: http://jxta.kenai.com/

[6] ——, *JXTA v2.0 Protocol Specification*, 2007. [Online]. Available: http://jxta.kenai.com/Specifications/JXTAProtocols2_0.pdf

[7] M. Amoretti, "Enabling peer-to-peer web service architectures with jxta.soap," in *IADIS e-Society 2008*, 2008.

[8] A. S. Foundation, "Web services - axis," 2005 Published. [Online]. Available: http://axis.apache.org/axis/

[9] M. P. Papazoglou, B. J. Krämer, and J. Yang, "Leveraging web-services and peer-to-peer networks," in *Proceedings of the 15th international conference on Advanced information systems engineering*, ser. CAiSE'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 485–501.

[10] Z. Zhengdong, H. Yahong, L. Ronggui, W. Weiguo, and L. Zengzhi, "A p2p-based semantic web services composition architecture," in *IEEE International Conference on E-Business Engineering*, oct. 2009, pp. 403–408. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ICEBE.2009.63

[11] S. Dustdar and L. Juszczyk, "Dynamic replication and synchronization of web services for high availability in mobile ad-hoc networks," *Service Oriented Computing and Applications*, vol. 1, no. 1, pp. 19–33, 2007. [Online]. Available: http://dx.doi.org/10.1007/s11761-007-0006-z

[12] L. Juszczyk, J. Lazowski, and S. Dustdar, "Web service discovery, replication, and synchronization in ad-hoc networks," in *Proceedings of the First International Conference on Availability, Reliability and Security*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 847–854. [Online]. Available: http://dx.doi.org/10.1109/ARES.2006.143

[13] S. Dustdar and M. Treiber, "Integration of transient web services into a virtual peer to peer web service registry," *Distributed and Parallel Databases*, vol. 20, pp. 91–115, September 2006. [Online]. Available: http://dx.doi.org/10.1007/s10619-006-9447-1

[14] J. G. R. Jr., G. T. do Carmo, M. T. Valente, and N. C. Mendonça, "Smart proxies for accessing replicated web services," *IEEE Distributed Systems Online*, vol. 8, no. 12, 2007.

[15] M. G. Nanda, S. Chandra, and V. Sarkar, "Decentralizing execution of composite web services," *SIGPLAN Not.*, vol. 39, pp. 170–187, October 2004. [Online]. Available: http://doi.acm.org/10.1145/1035292.1028991

[16] F. Montagut and R. Molva, "Enabling pervasive execution of workflows," in *Collaborative Computing: Networking, Applications and Worksharing, 2005 International Conference on*, 2005, p. 10 pp. [Online]. Available: http://dx.doi.org/10.1109/COLCOM.2005.1651227

[17] R. Khalaf, O. Kopp, and F. Leymann, "Maintaining data dependencies across bpel process fragments," *International Journal of Cooperative Information Systems.*, vol. 17, no. 3, pp. 259–282, 2008. [Online]. Available: http://dx.doi.org/10.1142/S0218843008001828

[18] R. Khalaf, "Supporting business process fragmentation while maintaining operational semantics: a bpel perspective," Ph.D. dissertation, University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology, 2008. [Online]. Available: http://elib.uni-stuttgart.de/opus/volltexte/2008/3514/

[19] M. Pereira, M. Fernandes, J. A. Martins, and J. S. Pinto, "Service oriented p2p networks for digital libraries, based on jxta." in *ICSOFT 2009 - Proceedings of the 4th International Conference on Software and Data Technologies*, B. Shishkov, J. Cordeiro, and A. Ranchordas, Eds. INSTICC Press, 2009, pp. 141–146.

[20] J. Hunter and S. Choudhury, "Panic: an integrated approach to the preservation of composite digital objects using semantic web services,"

 ISBN: 978-1-61208-173-1

International Journal on Digital Libraries, vol. 6, no. 2, pp. 174–183, 2006. [Online]. Available: http://dx.doi.org/10.1007/s00799-005-0134-z

[21] F. Xhafa, L. Barolli, T. Daradoumis, R. Fernández, and S. Caballé, "Jxta-overlay: An interface for efficient peer selection in p2p jxta-based systems," *Computer Standards & Interfaces*, vol. 31, no. 5, pp. 886–893, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYV-4S2TRXK-1/2/ac83a5f48d7beeac9f84cb21e6182d83

[22] N. C. Mendonça and J. A. F. Silva, "An empirical evaluation of client-side server selection policies for accessing replicated web services," in *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2005, pp. 1704–1708. [Online]. Available: http://doi.acm.org/10.1145/1066677.1067062

[23] S. Dykes, K. Robbins, and C. Jeffery, "An empirical evaluation of client-side server selection algorithms," in *Proceedings of INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, March 2000, pp. 1361–1370.

[24] L. Mandel, "Describe rest web services with wsdl 2.0: A how-to guide," IBM, 2008. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-restwsdl/

[25] C. Pautasso, "Bpel for rest," *Business Process Management*, vol. 5240, pp. 278–293, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85758-7_21

 ISBN: 978-1-61208-173-1

Symmetric Push-Sum Protocol for Decentralised Aggregation

Francesco Blasa, Simone Cafiero, Giancarlo Fortino,
Department of Electronics, Informatics and Systems
University of Calabria, 87036 Rende (CS), Italy
{*checco84,simone.cafiero*}*@gmail.com, g.fortino@unical.it*

Giuseppe Di Fatta
School of Systems Engineering, The University of Reading
Whiteknights, Reading, Berkshire, RG6 6AY, UK
G.DiFatta@reading.ac.uk

***Abstract*—Gossip (or Epidemic) protocols have emerged as a communication and computation paradigm for large-scale networked systems. These protocols are based on randomised communication, which provides probabilistic guarantees on convergence speed and accuracy. They also provide robustness, scalability, computational and communication efficiency and high stability under disruption. This work presents a novel Gossip protocol named Symmetric Push-Sum Protocol for the computation of global aggregates (e.g., average) in decentralised and asynchronous systems. The proposed approach combines the simplicity of the push-based approach and the efficiency of the push-pull schemes. The push-pull schemes cannot be directly employed in asynchronous systems as they require synchronous paired communication operations to guarantee their accuracy. Although push schemes guarantee accuracy even with asynchronous communication, they suffer from a slower and unstable convergence. Symmetric Push-Sum Protocol does not require synchronous communication and achieves a convergence speed similar to the push-pull schemes, while keeping the accuracy stability of the push scheme. In the experimental analysis, we focus on computing the global average as an important class of node aggregation problems. The results have confirmed that the proposed method inherits the advantages of both other schemes and outperforms well-known state of the art protocols for decentralized Gossip-based aggregation.**

***Keywords*-peer-to-peer computing; distributed aggregation algorithms; gossip protocols; extreme scale computing.**

I. INTRODUCTION

Nowadays, highly distributed systems such as P2P networks, large scale sensor networks, grids and ubiquitous systems, enable a broad range of applications [1]. Centralized paradigms are not suitable for distributed large-scale scenarios as they introduce bottlenecks and failure intolerance. In particular, applications require atop such systems a protocol layer that can cope well with the highly dynamic and decentralized nature of these infrastructures. Locality has been a key element to successfully deploy applications into large-scale networked systems. However, computing and spreading global information is still necessary for a wide range of applications and is a particularly challenging task when considering dynamic, highly distributed, large and extreme scale systems.

This work was partly carried out while Francesco Blasa and Simone Cafiero were at the University of Reading, UK, for a work placement of the Erasmus Training Programme (June-Sept. 2010).

Aggregation protocols represent a decentralized paradigm for computing global properties of distributed systems. Several distributed aggregation protocols have been proposed in the last years. They can be divided into two main classes: Tree-based protocols and Gossip-based protocols.

The former performs a tree-based communication throughout a tree overlay structure (e.g., [2]–[4]). Tree-based protocols support a minimum number of communications but require the construction of a hierarchical communication structure among nodes and can be affected by the presence of single points of failure.

The second class includes Gossip (or Epidemic) protocols, which are a robust and scalable communication paradigm to disseminate information in a large-scale distributed environment using randomised communication [5], [6]. Although Epidemic protocols have communication costs usually greater than tree-based protocols, they are intrinsically fault tolerant. Gossip-based communication can use push, pull or push-pull schemes.

P2P applications based on Gossip protocols are emerging in many fields. In [7], a global load monitoring service for P2P overlay networks has been proposed. In [8], a decentralized dynamic load balancing algorithm for a desktop Grid environment is presented. The work in [9] introduces an epidemic content search mechanism in unstructured P2P overlay over intermittently connected mobile ad hoc networks. The work in [10] studies Gossip-based message dissemination schemes to be employed for content and service dissemination or discovery in unstructured P2P and ad hoc networks. In [11], authors define a protocol to achieve mutual anonymity in unstructured P2P networks, which deals with high churn rates by means of an epidemic-style data dissemination. A number of Gossip-based protocols for sensor networks have also been proposed (e.g., [12]–[14]). Gossip protocols have also been adopted to solve the general data aggregation problem [15]–[17].

In this work, we present a new algorithm for Gossip-based aggregation named Symmetric Push-Sum Protocol (SPSP). The proposed algorithm is fully decentralized and suitable for large scale networks. We evaluate performances of our algorithm w.r.t. Push-Sum Protocol (PSP) [15] and Push-Pull Gossip Protocol (PPG) [16], [17]. SPSP preserves the mass conservation invariant, i.e., at any time the sum of all

 ISBN: 978-1-61208-173-1

values in the network is constant. This invariant guarantees the correctness of aggregation algorithms [18]. In particular, among the various aggregation functions in the experimental analysis we focus on the average. The simulations have confirmed the quality and the consistency of the results. In particular, the results show that the proposed approach always performs much better than the state of the art aggregation protocols for large-scale distributed systems.

The rest of this paper is organized as follow. Section II reviews related work. Section III presents the proposed aggregation protocol. Section IV presents an experimental comparative analysis. Finally, Section V provides some conclusive remarks and future research directions.

II. Related Work

Several Gossip-based aggregation protocols have been proposed. In [15], a push scheme protocol named Push-Sum Protocol (PSP) is proposed. It is a simple aggregation protocol for computing several aggregation functions (e.g., sum, average, count). In PSP, the local scalar value is represented as a pair (v, w), where v is initialised with the local value x and the initial weight w depends on the global aggregate function to be computed as shown in Table I. The global aggregate value is given by v/w after a fixed number of communication cycles. At each cycle, each node halves its local value and weight $(v, w) = (v/2, w/2)$ and sends the new obtained pair to a randomly selected node according to a uniform probability density function (pdf). The global mass is guaranteed to be conserved in case a reliable communication protocol is used. A number of messages equal to the number of nodes in the network is sent in total at each cycle. The *diffusion speed* is the minimum number of protocol cycles required to achieve a good approximation of the true value of the global aggregate function with high probability:

$$Prob(e_i < \varepsilon) \geq 1 - \delta, \quad \forall\, i = 1, \ldots, n, \tag{1}$$

where n is the network size, e_i the approximation error at node i and ε and δ two arbitrary small positive constants. The *diffusion speed* of the PSP has been shown to have a complexity $O(\log(n) + \log(\varepsilon^{-1}) + \log(\delta^{-1}))$ [15].

In [19], authors discuss some issues of PSP when used as summarisation algorithm. The sum function in [15] requires a leader election; this represents a non trivial task and could introduce single points of failure. In [19], the authors provides a scalable and fault tolerant solution to this problem by incorporating a leader election mechanism in the aggregation protocol.

In [16], [17], two similar algorithms are proposed; they can be both referred to as Push-Pull Gossip protocol (PPG). PPG uses a push-pull scheme that improves the diffusion speed w.r.t. PSP. In PPG at each cycle a node i randomly chooses a node j to perform an averaging operation and to update their local values $\frac{x_i+x_j}{2}$. In PPG, $2 \times n$ messages are sent in total at each cycle. In [17], authors focus on the design of Gossip algorithm by defining a method to obtain the fastest Gossip algorithm over a given distributed network. In particular they find out that the averaging time (which is directly related to diffusion speed) depends on the eigenvalue of a doubly stochastic matrix characterizing the algorithm. The fastest Gossip algorithm is obtained by minimizing the eigenvalues in a distributed fashion.

In [18], authors state that the correctness of such algorithms depends on the mass conservation invariant. They show that PPG could violate this fundamental invariant if an *atomic violation* happens. An *atomic violation* occurs on node i when i receives a push while it is waiting for a pull. Therefore two versions of PPG are proposed: Push-Pull Back Cancellation and Push-Pull Ordered Wait. The first algorithm adopts a simple message cancellation mechanism to guarantee atomicity and avoid mass conservation violation. However, the cancellation method decreases the diffusion speed. The second approach adopts a buffer for storing push messages that are received while a push-pull operation is being executed. This mechanism could introduce a deadlock across the network. To avoid deadlocks they introduce a total order among nodes and a constraint in the nodes selection mechanism, which penalizes selection of some nodes.

In general, the use of synchronous cycles simplifies the analysis and the implementation of Gossip-based aggregation protocols. Nevertheless, the protocols can be implemented in completely asynchronous environments. Independent local Poisson clocks can be used to generate synchronous cycles in asynchronous distributed environments (e.g., [20]). A second interesting alternative is the use of an exact global estimation of the right termination time, similar to the median-counter algorithm [6] for rumour spreading.

III. Symmetric PSP

We propose a novel Gossip-based aggregation protocol, the Symmetric Push-Sum Protocol (SPSP), which combines the simplicity of PSP and the convergence speed of PPG.

We assume that the transport protocol is reliable. This assumption is not strictly necessary and could be relaxed as Gossip protocols are intrinsically fault tolerant. However, in this work the effect of packet loss is not investigated.

SPSP adopts an asynchronous push-pull communication scheme. Push-pull schemes are expected to converge faster than push schemes with the same number of exchanged messages [6].

Let consider a distributed system composed by n peers $P = \{P_1, \ldots, P_n\}$. Each node i holds a local value v_i and a local weight w_i ($w_i \geq 0$) and needs to compute a global aggregation function $F(v_1, w_1, \ldots, v_n, w_n)$. Similarly to PSP, SPSP can perform several aggregation functions, some of which are shown in Table I.

 ISBN: 978-1-61208-173-1

Table I
SETTINGS FOR SEVERAL AGGREGATION FUNCTIONS

Function	Description
Sum	$v_i =$ local value $w_i = 1$ at a single node, 0 at all other nodes
Count	$v_i = 1$ $w_i = 1$ at a single node, 0 at all other nodes
Average	$v_i =$ local value $w_i = 1$
Weighted Average	$v_i =$ local value $\times$ local weight $w_i =$ local weight

At each node i
Require: v_0, w_0
The initial local value, v_0;
The initial local weight, w_0.
Initialisation:
1: $(v, w) = (v_0, w_0)$
At each cycle:
2: $j \leftarrow$ *getNode*()
3: $v \leftarrow v/2$, $w \leftarrow w/2$
4: send an *aggregation message* to j, $\langle (v, w), \textbf{true} \rangle$
At event: received an *aggregation message* $\langle (v', w'), r \rangle$ from j
5: **if** r is **true then**
6: $v \leftarrow v/2$, $w \leftarrow w/2$
7: send an *aggregation message* to j, $\langle (v, w), \textbf{false} \rangle$
8: **end if**
9: $v \leftarrow v + v'$, $w \leftarrow v + w'$

Figure 1. The pseudocode of the Symmetric Push-Sum Protocol

As shown in Figure 1, at each cycle a node i randomly selects a communication partner j according to a uniform pdf. This selection is provided by the service *getNode()* (line 2). Then i halves its local value and weight (line 3) and sends them to j (line 4). At the reception of the message, the node j will asynchronously perform a symmetric push operation: it halves its local value and weight and sends them to node i (lines 6-7). Then, it adds the received value and weight to its own value and weight (line 9). In case of *atomic violation*, a node i receives the symmetric push from $k \neq j$ immediately after its push operation, the symmetric push mechanism guarantees the *mass invariant*.

At each random push an asynchronous reply follows as a symmetric push: at each cycle $2 \times n$ messages are sent.

A. Node Cache Protocol

In uniform Gossip protocols the random node selection is a critical operation. In general, the global network topology is not known or is not available at each node.

Figure 2 describes the node selection algorithm adopted in SPSP, i.e. the Node Cache Protocol. The protocol only requires a few assumptions: the network is a connected graph, each node knows its physical neighbours ($Neighbours$), a multi-hop routing protocol is available.

At each node i
Require: $Q_{MAX}, Neighbours$
The maximum size of local cache, Q_{MAX};
The initial set of physical neighbours nodes, $Neighbours$.
Export: *getNode*()
Let *getNode*() return and remove a random node ID from the local node ID cache Q_i
Initialisation:
1: $Q_i \leftarrow Neighbours$
2: randomly trim Q_i such that $|Q_i| \leq Q_{MAX}$
At each cycle:
3: $j \leftarrow$ *getNode*()
4: send a *cache message* to j, $\langle Q_i, \textbf{true} \rangle$
At event: received a *cache message* $\langle Q_j, r \rangle$ from j
5: **if** r is **true then**
6: send a *cache message* to j, $\langle Q_i, \textbf{false} \rangle$
7: **end if**
8: $Q_i \leftarrow Q_i \cup Q_j \cup \{ j \}$
9: randomly trim Q_i such that $|Q_i| \leq Q_{MAX}$

Figure 2. The pseudocode of the Node Cache Protocol

The node selection protocol maintains a local cache Q of node identifiers (IDs), with $|Q| = Q_{MAX}$. The cache is initialised with the physical neighbours (lines 1-2). At each protocol cycle the local cache is sent to a node randomly chosen from the cache according to a uniform pdf (lines 3-4). When a remote cache is received, it is merged with the local one and trimmed to the maximum size by randomly removing a number of IDs exceeding Q_{MAX} (lines 5-9). The procedure can be considered a practical implementation of multiple random walks. After a sufficiently large number of cycles, the entries in the local cache are uniformly distributed. In regular connected graphs, random walks converge to uniform independent samples of the node set in a polynomial number of steps. In expander graphs, i.e., sparse graphs that are very well connected, random walks converge to the uniform distribution in $O(\log(n))$ [21].

The node cache protocol provides a local service *getNode()*, which removes and returns a random node from the cache.

IV. PERFORMANCE EVALUATION

In this section the proposed SPSP is evaluated and compared with PSP and PPG protocols for the decentralised approximate computation of a global average. At each cycle c of the aggregation protocol each node i computes an estimate $\tilde{m}$ of the global true average m:

$$\tilde{m}_i(c) \approx m = \frac{\sum_{j=1}^{n} v_j}{n} \quad (2)$$

A. Experimental setting

We implemented the three protocols, SPSP, PSP and PPG, in an ad hoc simulator based on discrete events [22]. The simulator has an event scheduler, a set of processes, which simulate network nodes, a topology manager and events, which represent operations such as initialisation, messages, computation, etc. The simulations assume that a reliable point-to-point communication protocol is available in the network.

We have tested the distributed algorithms in two different types of network topologies:

- two Internet-like topologies were generated using BRITE [23] with a Waxman model to simulate a flat level Autonomous System with 1000 and 5000 nodes;
- two 2D mesh topologies were also generated with dimensions, respectively, 40×25 (1000 nodes) and 100×50 (5000 nodes).

The algorithms were evaluated according to the peak data distribution, where only a node i holds as local value $v_i = N$, and all others j hold as local value $v_j = 0$. As shown in Table I to compute the average each node i holds weight $w_i = 1$. According to this setting, m is equal to 1. We have tested all the discussed algorithms with two different peak data distribution randomly generated.

Each one of the tested protocols uses the Node Cache Protocol reported in Section III-A. In particular, each node has its own node cache with Q_{MAX} equal to 20.

In order to simulate the algorithms and to collect relevant performance indices, we have adopted an opportune cycle structure of fixed length where the aggregation is carried out. The cycle structure guarantees that there is no overlap in the communication of different cycles and provides a simple mechanism for varying the *atomic violation* percentage (AVP).

Each cycle is composed of four intervals as shown in Figure 3. The four intervals have a fixed length of, respectively, $d1$, $d2$, $d2$ and $d3$:

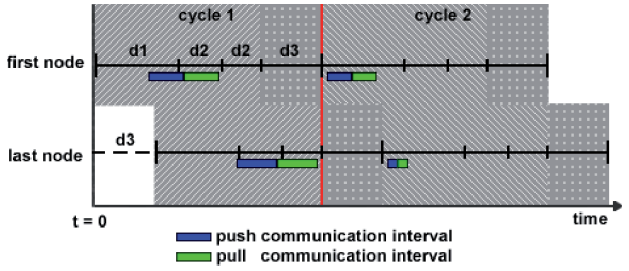


Figure 3. Cycle structure

- $d1$ is the length of the interval where nodes start push operations;
- $d2$ is the maximum propagation delay between any pair of nodes in the network;
- $d3$ is the maximum synchronisation offset between any pair of nodes in the network.

We assume a uniformly distributed synchronisation offset for the start of the aggregation process at different nodes. In all experiments the maximum synchronisation offset ($d3$) between any pair of nodes is set to 10 msec. This value is similar to a clock synchronisation offset, which can be obtained using e.g., NTP [24] or PariSync [25].

Each node (source) initiates a push operation at a random instant of the first interval ($d1$). In particular, $d1$ is the simulation parameter through which the AVP can be varied. By decreasing the value of $d1$ the AVP increases and vice versa.

After a propagation delay the push message is received at the destination node, which asynchronously replies with a symmetric push (pull) message. After the corresponding propagation delay the reply is received at the source node. The second and third intervals account for these propagation delays. Two intervals of length $d2$ are necessary to guarantee that a symmetric push operation (push-pull) is completed. The value of $d2$ is a property of the network topology.

Finally a padding interval ($d3$) is required to ensure that communications of two different cycles do not overlap because of the synchronisation offset of the nodes.

B. Analysis

The performance of the three methods (SPSP, PSP and PPG) are compared in terms of accuracy and convergence speed at different AVP levels. The accuracy is computed in terms of the mean percentage error of the estimated average among all nodes, as shown in equation 3.

$$\text{MPE}(c) = \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{m - \tilde{m}_i(c)}{m} \right| \quad (3)$$

The convergence speed is evaluated by means of the variance of $\tilde{m}_i$ among all nodes over time, as indicated in equation 4.

$$\text{VAR}(c) = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (m - \tilde{m}_i(c))^2 \quad (4)$$

As shown in Figure 4, PPG is sensitive to AVP. PPG always reaches a non-null error for $AVP > 0\%$. With a very large $d1$ interval the smallest AVP level (0.3%) is obtained. Even in this case PPG does not converge to the true average (MPE $\neq 0$). PSP preserves the *mass conservation* invariant and is guaranteed to converge to the true average. However it has a slow convergence speed compared to the other protocols. SPSP is not sensitive to the AVP level and is guaranteed to converge to the true average. Moreover, SPSP provides the best accuracy in all simulated scenarios.

 ISBN: 978-1-61208-173-1

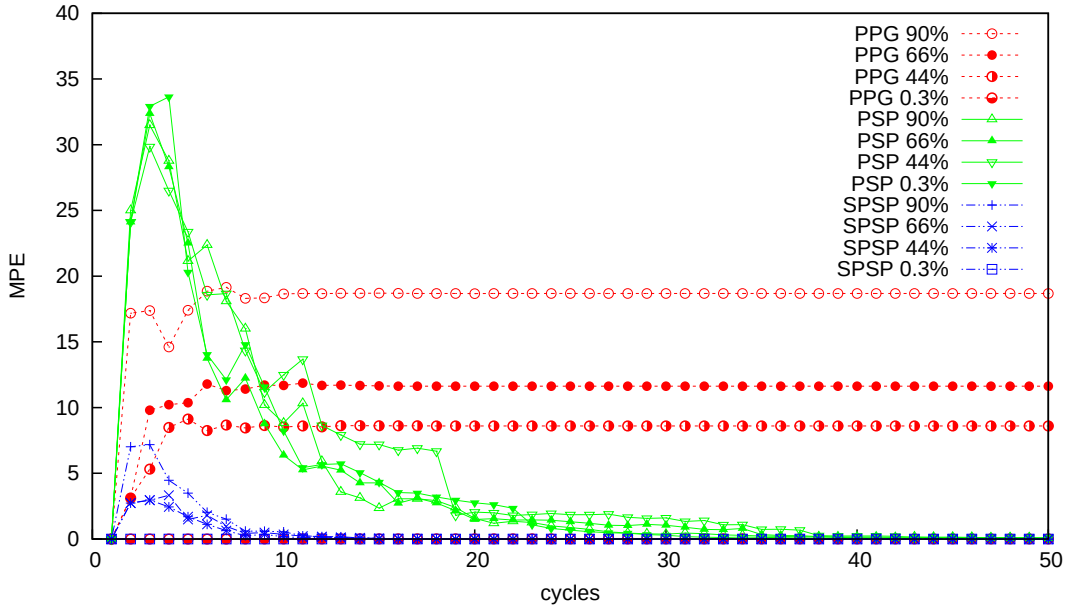


Figure 4. MPE varying the AVP w.r.t. 100 different simulations: BRITE and Mesh topologies, 1000 and 5000 nodes, 2 different Peak Data Distribution.

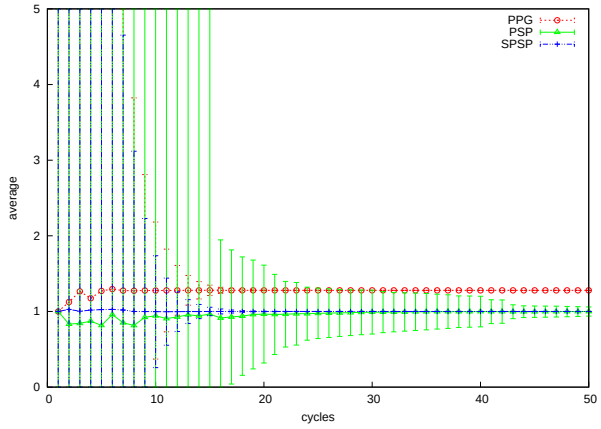


Figure 5. Average and standard deviation of the estimated aggregate over the network nodes. BRITE topology with 5000 nodes. AVP equal to 90%.

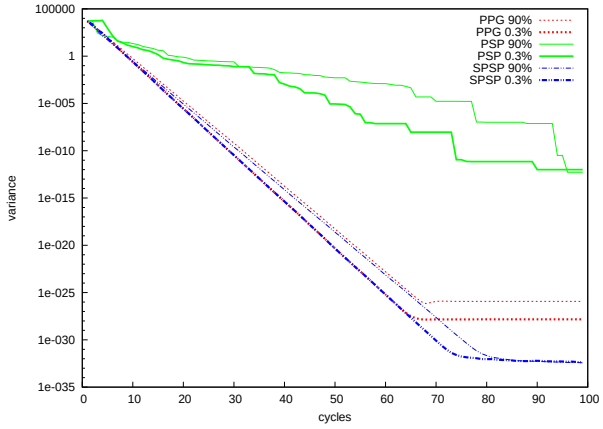


Figure 6. Convergence speed (variance). BRITE topology with 5000 nodes. AVP equal to 0.3% and 90%.

In Figures 5 and 6, PPG and SPSP have a similar variance trend, however PPG converges to an incorrect estimate $\tilde{m}$.

V. Conclusion

The Symmetric Push-Sum Protocol (SPSP) is a novel Gossip-based aggregation protocol that is suitable for computing aggregation functions on networks of any scale. The proposed algorithm is totally decentralized and robust and preserves the *mass conservation* invariant. The experimental analysis has confirmed that the algorithm outperforms the state of the art protocols (i.e., PSP and PPG) for the average aggregation function. In particular, SPSP does not violate the *mass conservation* invariant, similarly to PSP, and is much faster than PSP. SPSP and PPG have similar convergence speed; SPSP guarantees a convergence to the true global aggregate, while PPG does not. Future research directions will focus on the evaluation of the protocol in dynamic environments with churn rate and with node and link failures.

 ISBN: 978-1-61208-173-1

REFERENCES

[1] F. Cappello, S. Djilali, G. Fedak, T. Hérault, F. Magniette, V. Néri, and O. Lodygensky, "Computing on large-scale distributed systems: Xtremweb architecture, programming models, security, tests and convergence with grid," *Future Generation Comp. Syst.*, vol. 21, no. 3, pp. 417–437, 2005.

[2] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani, "Estimating aggregates on a peer-to-peer network," Stanford InfoLab, Technical Report 2003-24, April 2003.

[3] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 131–146, December 2002.

[4] M. Dam and R. Stadler, "A generic protocol for network state aggregation," in *In Proc. Radiovetenskap och Kommunikation (RVK)*, 2005, pp. 14–16.

[5] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proc. of the sixth annual ACM Symposium on Principles of distributed computing*, ser. PODC '87. ACM, 1987, pp. 1–12.

[6] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 2000, pp. 565–574.

[7] B. Ghit, F. Pop, and V. Cristea, "Epidemic-style global load monitoring in large-scale overlay networks," *International Conference on P2P, Parallel, Grid, Cloud, and Internet Computing*, pp. 393–398, 2010.

[8] D. H. H. Sheng Di, Cho-Li Wang, "Gossip-based dynamic load balancing in an autonomous desktop grid," in *Proc. of the 10th International Conference on High-Performance Computing in Asia-Pacific Region*, 2009, pp. 85–92.

[9] Y. Ma and A. Jamalipour, "An epidemic P2P content search mechanism for intermittently connected mobile ad hoc networks," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2009, pp. 1–6.

[10] S. Tang, E. Jaho, I. Stavrakakis, I. Koukoutsidis, and P. Van Mieghem, "Modeling gossip-based content dissemination and search in distributed networking," *Comput. Commun.*, vol. 34, pp. 765–779, May 2011.

[11] N. Bansod, A. Malgi, B. K. Choi, and J. Mayo, "Muon: Epidemic based mutual anonymity in unstructured P2P networks," *Computer Networks*, vol. 52, no. 5, pp. 915–934, 2008.

[12] L. Chitnis, A. Dobra, and S. Ranka, "Aggregation methods for large-scale sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, pp. 1–36, April 2008.

[13] N. Marechal, J.-M. Gorce, and J. Pierrot, "Joint estimation and gossip averaging for sensor network applications," *IEEE Transactions on Automatic Control*, vol. 55, no. 5, pp. 1208–1213, may 2010.

[14] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, nov. 2010.

[15] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, oct. 2003, pp. 482–491.

[16] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Transactions on Computer Systems*, vol. 23, pp. 219–252, August 2005.

[17] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, june 2006.

[18] P. Jesus, C. Baquero, and P. Almeida, "Dependability in aggregation by averaging," in *1st Symposium on Informatics (INForum 2009)*, sept. 2009, pp. 482–491.

[19] W. Terpstra, C. Leng, and A. Buchmann, "Brief announcement: Practical summation via gossip," in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing (PODC)*. ACM, August 2007, pp. 12–15.

[20] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, sep 1986.

[21] D. Gillman, "A chernoff bound for random walks on expander graphs," *SIAM Journal on Computing (Society for Industrial and Applied Mathematics)*, vol. 27, no. 4, pp. 1203–1220, 1998.

[22] G. Fortino, C. Mastroianni, and W. Russo, "A hierarchical control protocol for group-oriented playbacks supported by content distribution networks," *Journal of Network and Computer Applications*, vol. 32, no. 1, pp. 135–157, 2009.

[23] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in Internet topologies," *SIGCOMM Comput. Commun. Rev.*, vol. 30, pp. 18–28, April 2000.

[24] D. L. Mills, "On the accuracy and stability of clocks synchronized by the network time protocol in the Internet system," *ACM Computer Communication Review*, vol. 20, pp. 65–75, 1990.

[25] P. Bertasi, M. Bonazza, N. Moretti, and P. E., "PariSync: Clock Synchronization in P2P Networks," *ISPCS 2009 Internation IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pp. 12–16, October 2009.

 ISBN: 978-1-61208-173-1

A Data Aggregation System using Mobile Agents on Integrated Sensor Networks

Yuto Hamaguchi*, Tomoki Yoshihisa*, Yoshimasa Ishi*,Yuuichi Teranishi†, Takahiro Hara*, and Shojiro Nishio*
*Department of Multimedia Engineering,Graduate School of Information Science and Technology, Osaka University
†National Institute of Information and Communications Technology, Japan
Email: [hamaguchi.yuto,yoshihisa,ishi.yoshimasa]@ist.osaka-u.ac.jp, teranisi@nict.go.jp, [hara,nishio]@ist.osaka-u.ac.jp

***Abstract*—Due to the recent development of sensor networks, integrated sensor networks, in which some sensor networks are managed systematically, have attracted considerable attention. To implement various applications, such as weather forecasting and environmental observation, with integrated sensor networks, users (clients) usually aggregate and collect the sensor data obtained from all the sensor networks. After this, the clients execute some operations, such as averaging and analysis, on the aggregated sensor data. However, such processes cause heavy network traffic in aggregating data from all the sensor networks. Hence, in this paper, we propose a data aggregation system using mobile agents. Mobile agents are programs that migrate among sensor networks. Since the mobile agents migrate while executing operations on the sensor data, the clients do not need to aggregate the sensor data by collecting it from all the sensor networks, and the network traffic can be reduced.**

Keywords-wireless sensor networks; mobile agents; peer-to-peer networks.

I. INTRODUCTION

Due to the recent development of sensing technologies, sensor networks, in which sensors construct information networks and communicate with each other, have attracted considerable attention. A sensor network typically has a sink node, which collects the sensor data generated from the sensors connected to the sensor network. However, sensor networks have limits regarding the geographical area they can cover, since the sensors must function within communication ranges of other sensors. Therefore, we need to integrate local sensor networks in order to construct broad sensor networks that can cover a wide area. We call the constructed sensor networks as *integrated sensor networks*. With integrated sensor networks, users can collect sensor data obtained from various sensor networks. In the following examples, there are three sensor networks deployed respectively at Osaka, Kyoto, and Nara, which are notable Japanese cities in the Kansai area, located near one another.

- For weather forecasting, a user calculates the average temperature of these cities. To calculate the average value, the client aggregates the temperature sensor data obtained from all the sensor networks.
- For environmental observation, a user finds the hottest city among the three cities. To find the city that has the maximum temperature value, the client calculates the temperature sensor data obtained from all the sensor networks.
- To find broken temperature sensors, a user finds the sensors generating abnormal values, by analyzing the sensor data generated from all the sensor networks.

To aggregate sensor data generated from integrated sensor networks, several data aggregation systems have been proposed ([1], [2]). In these systems, users (clients) must aggregate the sensor data obtained from all sensor networks for their clients. After this, the clients execute some operations, such as averaging and analysis, on the aggregated sensor data. However, such processes cause heavy network traffic in aggregating data from all the sensor networks. To solve this problem, data aggregation operations should be performed only on necessary data at each sensor network. For example, to determine the average temperature for weather forecasting, the clients do not need to collect all the sensor data, but only the average temperature and the number of sensors for each sensor network.

Hence, in this paper, we propose a data aggregation system using mobile agents on integrated sensor networks. Mobile agents are computer programs that migrate among sensor networks. In our proposed system, clients generate mobile agents. The mobile agents execute the user-written programs migrating among sensor networks, and finally return to the clients. Since the mobile agents migrate while executing operations on the sensor data, the clients do not need to collect and aggregate the sensor data generated from all the sensor networks, and thus network traffic can be reduced.

The rest of this paper is organized as follows. We introduce related work in Section II. In Section III, we discuss the requirements for integrated sensor networks, and explain the system designed to satisfy these requirements. We describe the implementation of our proposed system in Section IV, and discuss its merits and demerits in Section V. Finally, in Section VI we conclude the paper.

II. RELATED WORK

A number of sensor data aggregation systems for integrated sensor networks have been developed. LiveE! collects the sensor data observed by digital instrument shelters ([3]). The shelters are managed by 11 decentralized servers, and

 ISBN: 978-1-61208-173-1

have temperature, humidity, pressure, wind-direction, wind-speed, and rainfall sensors. Since the servers are decentralized, LiveE! can relieve the servers' load. However, users must access all the servers to collect all the sensor data.

X-Sensor 1.0 is a sensor network testbed which our research group has developed ([4], [5]). X-Sensor 1.0 has several sensor networks that are deployed in major Japanese universities. By registering a sensor network with the X-Sensor 1.0 testbed server, the sensor network can be managed by the centralized server. Users can collect the sensor data from the X-Sensor 1.0 system via the centralized server. Compared with X-Sensor 2.0, it is difficult to aggregate sensor data from multiple sensor networks with the X-Sensor 1.0 since we have to submit aggregation queries to sink nodes for each sensor network.

IrisNet [2] stores the sensor data in the distributed sensor databases. In IrisNet, SAs (Sensing Agents) collect the sensor data from several sensors. The SAs aggregate the collected sensor data for nearby OAs (Organizing Agents). The OAs select the most relevant database in which to store the aggregated data, from the viewpoint of overall system performance, and transfer their data to this database.

Environmental Monitoring 2.0 is a data-sharing and visualization system using Sensormap [6] and GSN (Global Sensor Network) [7]. Sensormap provides data-sharing services, and GSN provides data-management services. In contrast to the three systems above, Environmental Monitoring 2.0 focuses on the visualization system. It can show the sensor type, sensor data, and so on, visually.

However, in these systems users must collect the sensor data for their clients; and after this, the clients execute operations on the collected sensor data. In our proposed system, the clients do not need to collect all the sensor data, since mobile agents migrate among the sensor networks while executing operations on the sensor data.

Some systems designed to manage mobile agents, such as AgentTeamwork [8], PIAX [9], and MADSN [10], have been proposed. AgentTeamwork uses mobile agents for grid computing. The mobile agents migrate to the computers that have the necessary data, and execute the required tasks there. PIAX is a P2P-based mobile-agent system. However, AgentTeamwork and PIAX do not focus on sensor data aggregation. MADSN, on the other hand, does not focus on integrated sensor networks. A mobile-agent system customized for integrated sensor networks offers users easy and efficient aggregation of sensor data generated from such networks.

III. Requirements and Design

As noted in Section I, the network traffic required to aggregate the sensor data from conventional integrated sensor networks is large, since the clients must often collect all the sensor data from all the sensor networks. By executing operations on the sensor data and aggregating only the necessary data, the network traffic can be reduced. Mobile-agent systems are suitable for executing operations on respective sensor networks, since they can migrate among the sensor networks, executing operations. Therefore, we use mobile agents.

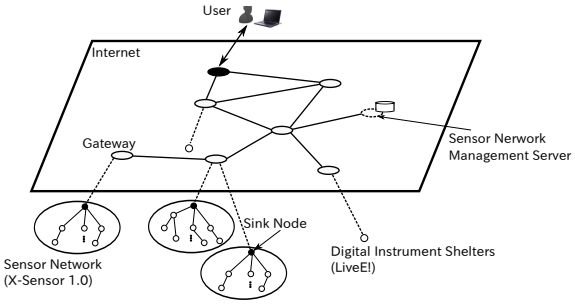


Figure 1 An Integrated Sensor Network

A. Requirements

In this subsection, we describe the requirements for data aggregation systems using mobile agents.

1) Management of Mobile Agents: To enable mobile agents to aggregate the sensor data, we need to manage the mobile agents. That is, the system generates the mobile agents and controls them according to users' operations. For example, users select the sensor networks which they want to aggregate the sensor data, and write a data aggregation program for the mobile agents. After this, the clients generate the mobile agents, and the mobile agents begin to migrate. The written program is executed in each sensor network when the mobile agents migrate to it. In addition, users can control the mobile agents by sending operations such as 'move', 'stop', and 'destroy', and debug the program in the course of the mobile agents' execution of their tasks. Finally, the mobile agents return to the client. The system must manage the mobile agents in order to accomplish such operations.

2) Visual Interface: To facilitate control of the mobile agents, it is useful to determine where they are and what they are doing. With visual access to their locations, users can intuitively determine where the mobile agents are. In addition, visual interfaces facilitate users' selections of mobile agents. Users select the mobile agents using a visual interface when they wish to confirm their status (such as waiting or running). Thus, a visual interface is required for data aggregation systems using mobile agents.

3) Physical Sensor Network: To aggregate the sensor data generated from the sensor networks, the system obviously needs to connect to the physical sensor networks. The sink nodes of the sensor networks collect the sensor data generated from the connected sensors. In addition, the system must accommodate different sensor database schemas, since

 ISBN: 978-1-61208-173-1

the sensor networks are managed by different organizations. Thus, a schema-management function is required for managing the physical sensor network.

B. Design

Figure 1 shows the network architecture for our model integrated sensor network. Each sensor network has a gateway, and can connect to the Internet via the gateway. The gateway has rich computing resources, and has no difficulties with battery lifetime as it is connected to an external power source. In the figure, two different types of sensor networks, X-Sensor 1.0 and LiveE!, are included in the integrated sensor network. To integrate different types of sensor networks, we use the sensor network management server, which manages the metadata, such as the ID, location, and database schema of each sensor network. Users can determine these by consulting the sensor network management server. Since the sensor network management server needs to maintain the latest metadata, the metadata are sent to the sensor network management server when the gateways update their metadata. Below, we explain our system design based on the requirements described in the previous subsection.

1) Use of the Mobile Agent System: Various systems to manage mobile agents have been developed. Among these, P2P-based systems are suitable for integrated sensor networks that include many sensor networks. By using a P2P-based mobile-agent system, the system load is not concentrated on the server, and thus it can run the mobile agents effectively. Although our system has a sensor network management server, the load is low since the clients communicate with it only when they start running the system. To facilitate the collection of sensor data generated from integrated sensor networks, we prepare two kinds of special mobile agents: (a) gateway agents which are stationed at the gateways to access sensor networks, and collect sensor data obtained from sink nodes via the gateways; and (b) user agents which are generated by the clients and execute the programs written by users. These latter cannot access the sensor data obtained from the gateway directly, since they do not know the access method for the sensor database of the sink node connected to the gateway. Therefore, they communicate with the gateway agents in order to receive the sensor data.

By stopping the user agents at the gateways, users can check their status. Otherwise, users cannot check their status, since the user agents usually continue to migrate. We defined four user-agent statuses: *READY* means the mobile agents are ready to run programs; *WAIT* means the mobile agents are waiting at each sensor network for the user's commands; *RUNNING* means the mobile agents are executing the user's programs; *FINISH* means the mobile agents have completed the program.

2) Web Browser Interface: To provide visual interfaces, we use web browsers. This is because users do not need to install new software in order to visualize the state of sensor data aggregation, since most of the clients have web browsers. In addition, we can employ various web applications such as maps and databases through the Internet, and exploit these, through web browsers, for the purposes of visualization. By clicking and sending the commands to the indicators for the mobile agent on the map, users can control the user agent based on its status. Furthermore, in the case of an abnormal stop, or the deletion of the mobile agents, the messages are displayed on the web browser.

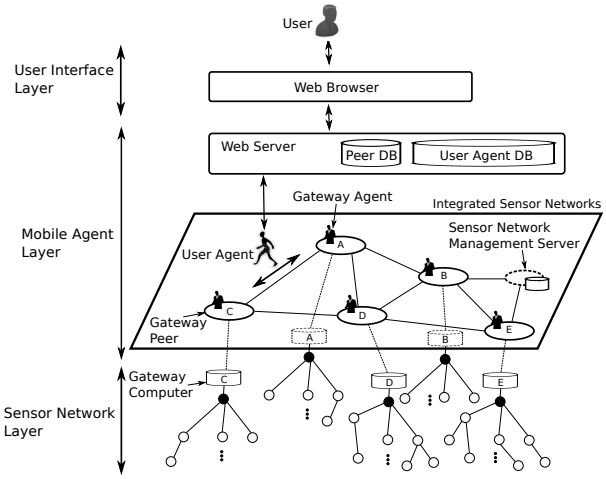


Figure 2 The X-Sensor 2.0 Architecture

3) Integrated Sensor Networks: Since the sensor network management server has the metadata for the sensor networks, we can employ different types of sensor networks that have different sensor database schemas. In our design, the sink nodes must connect to the gateways, and the mobile-agent management system must be installed in the gateways.

IV. Implementation

In this section, we describe our implementation of the data aggregation system using mobile agents. We call the implemented system X-Sensor 2.0. The X-Sensor 2.0 architecture is shown in Figure 2. The architecture has three layers. In Subsection IV-A, we explain the mobile-agent layer. In Subsection IV-B, we explain the user-interface layer; and finally, we explain the sensor-network layer in Subsection IV-C. The X-Sensor 2.0 essentially extends the X-Sensor 1.0 to incorporate the use of mobile agents.

A. Implementation of Mobile Agents

We exploit PIAX ([9]) for managing the mobile agents in the mobile-agent layer. PIAX is a peer-to-peer (P2P)-based mobile-agent system, suitable for aggregating data from integrated sensor networks. The mobile agents are implemented by Java. Our implemented mobile agents have some APIs (Application Programming Interfaces). Users

 ISBN: 978-1-61208-173-1

Table I X-SENSOR 2.0 APIS (PARTIAL LIST)

Class	Main Method	Arguments	Return Value
PeerAccess	getPeerInfo	none	PeerInfo
SQLProcessor	getSensorData	id,site, query	none
	getSensorResult	id	ResultSet
	returnAsText	id	none
	returnAsImage	id, GraphOptions	none
Agent Communication	sendMessage	agentid, message	none
AgentControl	getAgentStatus	none	AgentStatus
	move	peername	none
	stop	none	none

write mobile-agent programs using these APIs. For non-expert users, we can prepare various template programs. The implemented APIs are shown in Table I (X-SENSOR 2.0 APIS). The PeerAccess class provides users access to information about the sensor networks. For example, users can obtain the database schemas and the location of the mobile agents at runtime by using the getPeerInfo method. The SQLProcessor class provides the query processing functions. Users write SQL queries and make the user agents execute them by using the getSensorData method included in the SQLProcessor class. Users can obtain the result by using the getSensorResult method, and then show the result on the web browser by using the returnAsText or returnAsImage methods. The returnAsText method is used to check the results of data aggregation by downloading the text files. The returnAsImage method is used to return the JPEG images to clients. The AgentCommunication class effects communication between the user agents and the gateway agents. For example, when users wish to send messages to a user agent, they use the sendMessage method. Users write the receiveMessage method to receive the messages. The AgentControl class provides the getStatus method and some agent-control methods. By using the getStatus method, users can determine the status of the user agents at runtime, and control them by the move method or the stop method based on their status.

B. Visualization of Sensor Networks

We use web browsers for the user-interface layer, as explained in III-B (2). Figure 3 shows a screenshot of the web browser interface for the X-Sensor 2.0. Users log into the X-Sensor 2.0 system. Then, the map is shown on the web browser. To render the map, we use MSN Virtual Earth [11]. In the figure, we can see sensor networks at Italy, India, China, and Japan. A red square indicates a single sensor network. When the mouse cursor points at the indicator, the detailed information for the sensor network pops up. If there are a number of sensor networks in a narrow area, it may be difficult to recognize each indicator. In this case, the indicators are bundled up in one square indicator. By

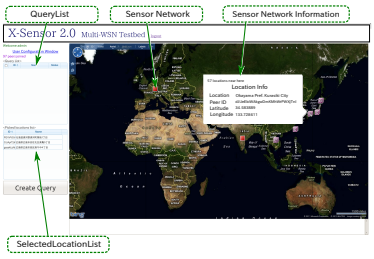


Figure 3 The X-Sensor 2.0 Web Interface

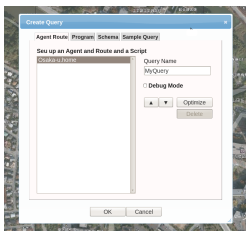


Figure 4 Query Creation Dialogue

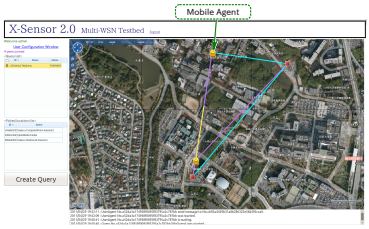


Figure 5 Visualization of Mobile Agents

Figure 6 Sensors for X-Sensor 2.0 in Our Laboratory

zooming the map, the bundled indicator is divided into the respective squares. To display the information for each sensor network when users access the web site, the web server accesses the sensor network information from the sensor network management server.

To aggregate the data from integrated sensor networks, users first select the respective sensor networks. The selected sensor networks are listed in the *SelectedLocationList*, which is shown on the left side of the web page. When users click the selected sensor network name on the *SelectedLocationList*, the location is shown on the map. By double-clicking the sensor network name, the sensor network is removed from the *SelectedLocationList*.

After selecting the sensor networks, users push the "Create Query" button to create queries. Here, "query" means a query to aggregate sensor data, including the program for the mobile agents. To answer the queries, the system generates the user agents. Unless the user writes the mobile-agent generation function in the program for the user agent, one query generates one user agent. Figure 4 shows the query creation dialogue. With the *AgentRoute* tab, users enter the query name. Then, users select the agent mode. The agent modes include a normal mode and a debug mode. In the normal mode, the mobile agents do not wait for user commands before migrating through sensor network. Thus, users cannot control the mobile agents until they return to the client. In the debug mode, the mobile agents wait for the 'move' command before they begin to migrate to other sensor networks. The program for the mobile agents is written in the *Program* tab. In the *Schema* tab, users can

 ISBN: 978-1-61208-173-1

confirm the metadata for each sensor network. When users finish creating the query, they push the "OK" button. Then, the generated user agent begins to migrate among the sensor networks. The created queries are listed in the *QueryList* located on the top left of the web page.

To facilitate their visualization, the user agents register their events, such as beginning migration or beginning communication with the web server. When the web server receives an event, it sends an asynchronous message to the users' web browser using Reverse Ajax. When a mobile agent migrates to a peer, the location is shown in the map, as in Figure 5.

The X-Sensor 2.0 can show the aggregation results by means of a graph image, as shown in Figure 7. Users write in the programs how to visualize the results of data aggregation and generate the graph image from query results; then download the results. We explain the detailed program in Subsection IV-D.

Users can check the results of the data aggregation by double-clicking the query name listed in the *QueryList*. In addition, since the mobile agents can return the results at runtime, users can check the intermediate results when needed.

C. Implementation of Integrated Sensor Networks

We need only two steps to add a new sensor network to the X-Sensor 2.0. For example, first, users create a configuration file for the new sensor network. Next, users install and activate the PIAX on the gateway computer, and connect the sensor network to the integrated sensor network. Figure 6 shows the physical sensors that compose the X-Sensor 2.0. We deployed more than 100 sensor nodes in our laboratory.

D. Application Scenario

In this subsection, we describe how to aggregate the sensor data obtained from the X-Sensor 2.0. The X-Sensor 2.0 is available as web site (see [12]). Suppose the case that the user obtains the average temperature from an integrated sensor network. The integrated sensor network consists of three sensor networks, *Osaka-u.NishioLab.Xsensor1*, *Osaka-u.NishioLab.Xsensor2*, and *CyberMedia.Center*. These sensor networks have temperature sensors, and periodically collect the sensor data into their databases. First, the user logs into the system and consults the map. The user selects some of these networks by clicking their indicators shown on the map. Then, the user clicks the "Create Query" button, and chooses a template program for the mobile agents. The template program for this example is shown in Figure 8.

Lines 1 and 2 are required to employ the user agent and the SQL query API respectively. The userProcessing method beginning with line 6 is executed when the mobile agent migrates to other sensor networks. In line 8, the variable *sp* is initialized and manages the query results of each sensor network. In line 9, the mobile agent aggregates the *timestamp* and the temperature

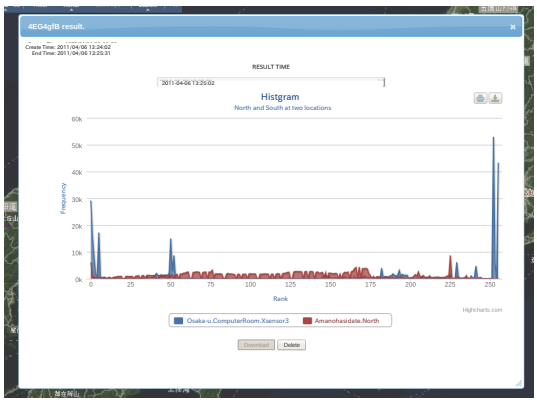


Figure 7 Visualization of Results

```
1  import org.xsensor2.agent.TraverseAgent;
2  import org.xsensor2.dbaccess.SQLProcessor;
3
4
5  public class UserAgent extends TraverseAgent{
6    public void userProcessing() {
7
8      SQLProcessor sp = this.getSQLProcessor();
9      sp.getSensorData("key1","select timestamp,value from temperature
           limit 20","Osaka−u.NishioLab.Xsensor1");
10     sp.getSensorData("key1","select timestamp,value from temperature
           limit 20","Osaka−u.NishioLab.Xsensor2");
11     sp.getSensorData("key1","select timestamp,value from temperature
           limit 20","CyberMedia.Center");
12   }
13
14   public void finalProcessing() {
15
16     SQLProcessor sp = this.getSQLProcessor();
17     sp.getSensorData("avg","SELECT sensortype,AVG(value) FROM key1
           ");
18     sp.returnAsImage("avg");
19
20   }
21 }
```

Figure 8 A Program for User Agents

data as a *value* from the *Osaka-u.NishioLab.Xsenseor1* sensor network. Also, in lines 10 and 11, the mobile agent aggregates the same sensor data obtained from *Osaka-u.NishioLab.Xsensor2* and *CyberMedia.Center*. The finalProcessing method beginning with line 14 is executed when the mobile agent returns to the client. In the method, the program calculates the average of the aggregated sensor data. At this time, *key1* has the query results obtained from the getSensorData method. After writing the program, the user creates the mobile agent, which migrates among the sensor networks. After a while, the mobile agent returns to the client and the result is shown on the web browser.

E. System Evaluation

We measured the network traffic for data aggregation using X-Sensor 2.0 implementation and server client implementation, as a comparison. At the measurement, 5 sensor networks are connected, and the size of each sensor data is 10 bytes. There are over 20 sensors in each sensor network,

 ISBN: 978-1-61208-173-1

Table II THE EVALUATION RESULT

Aggregation Method	Network Traffic	
	Total 500 records	Total 50,000 records
Server Client	24,414 bytes	217,450 bytes
Mobile Agent	45,695 bytes	45,730 bytes

and user submits the query to calculate average value for all sensor data. We created the query and the mobile agent program to get 100 or 10,000 records at each sensor network. The size of the query and the mobile agent program are 618 and 9013 bytes respectively. The network traffic includes the data for mobile agents. The result is shown in Table II. In server-client method, the server aggregates all sensor data. Then, it calculates the average value. In mobile-agent method, the mobile agent calculates the average value while migrating among gateways successively.

V. DISCUSSION

We can see that the mobile agent method can reduce the network traffic compared with the server-client method. The server-client method requires more network traffic than the mobile agent method as the sensor data size increases. However, when the mobile agent can not aggregate sensor data effectively, the network traffic increases. Therefore, the effectiveness of using mobile agents depends on the application of data aggregation.

X-Sensor 2.0 is suitable for sensor data aggregation using mobile agents, since the system is designed to make the mobile agents aggregate the sensor data. Other mobile-agent systems such as AgentTeamwork or the original PIAX are designed for, and can be applied to, various mobile-agent applications. However, a mobile-agent system customized for integrated sensor networks offers users easy and efficient aggregation of the sensor data generated from integrated sensor networks, since the mobile agent migrates among multiple sensor networks. A further merit may be seen in the X-Sensor 2.0's ability to visualize the locations of sensor networks and the mobile agents. Therefore, it is easy to determine the status of the mobile agents. Regarding the fault tolerance for X-Sensor 2.0, by programming the mobile agents so that they can find unusual response from sensors, we can make the system be tolerant for troubles.

On the other hand, one of the demerits of the X-Sensor 2.0 is that it is hard for non-experts users to write the agent programs. We can solve this problem by improving the script language for programming the mobile agents, and supporting more intuitive GUI. In addition, since users have flexibility in writing mobile-agent programs, they can create malicious agents such as computer viruses. However, we can detect these by checking the user-written mobile-agent programs at the web server.

VI. CONCLUSION

In this paper, we described the design and implementation of the X-Sensor 2.0, which is a data aggregation system using mobile agents on integrated sensor networks. In the system, the mobile agents migrate among the sensor networks, executing operations on the sensor data. Since the mobile agents aggregate only necessary sensor data, network traffic can be reduced.

In the future, we will define a more intuitive script language for programming the mobile agents.

ACKNOWLEDGMENT

This research was supported by collaborative research of NICT·Osaka University (research and development, validation of integrated management technique for heterogeneous, wide-area sensor networks) and Grants-in-Aid for Scientific Research (S) numbered 21220002.

REFERENCES

[1] S. Michel, A. Salehi, L. Luo, N. Dawes, K. Aberer, G. Barrenetxea, M. Bavay, A. Kansal, K. A. Kumar, S. Nath, M. Parlange, S. Tansley, C. van Ingen, F. Zhao, and Y. Zhou, "Environmental monitoring 2.0.", In Proc. of the International Conference on Data Engineering (ICDE 2009), pp. 1507-1510, 2009.

[2] P. Gibbons, B. Karp, Y. Ke, S. Nath, and S.n Sechan, "IrisNet: An Architecture for a Worldwide Sensor Web.", IEEE Pervasive Computing, Vol. 2, No. 4, pp. 22-33, 2003.

[3] Live E!: Live environmental information of the earth, http://www.live-e.org/en, Sep 3, 2011.

[4] A. Kanzaki, T. Hara, Y. Ishi, T. Yoshihisa, Y. Teranishi, and S. Shimojo, "X-Sensor: Wireless Sensor Network Testbed Integrating Multiple Networks.", Wireless Sensor Network Technologies for the Information Explosion Era Studies in Computational Intelligence, Vol. 278, pp. 249-271, 2010.

[5] X-Sensor Multi-WSN Testbed, http://www1.x-sensor.org, Sep 3, 2011.

[6] S. Nath, J. Liu, and F. Zhao, "Sensormap for wide-area sensor webs.", IEEE Computer, Vol. 40, No. 7, pp. 90-93, 2008.

[7] K. Aberer, M. Hauswirth, and A Salehi, "Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks.", In Proc. of the International Conference on Mobile Data Management (MDM 2007), pp. 198-205, 2007.

[8] M. Fukuda, C. Ngo, E. Mak, and J. Morisaki, "Resource management and monitoring in AgentTeamwork grid computing middleware.", In Proc. of the IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (PacRim 2007), pp. 145-148, 2007.

[9] Y. Teranishi, "PIAX: Toward a Framework for Sensor Overlay Network.", In Proc. of the International IEEE Consumer Communications and Networking Conference Workshop on Dependable and Sustainable Peer-to-Peer Systems (CCNC 2009), pp. 1-5, 2009.

[10] H. Qi, S. S. Iyengar, and K. Chakrabarty, "Distributed Multi-Resolution Data Integration Using Mobile Agents.", In Proc. of the IEEE Aerospace Conference, Vol. 3, pp. 1133-1141, 2001.

[11] MSN Virtual Earth, http://virtualearth.msn.com, Sep 3, 2011.

[12] X-Sensor 2.0, http://www2.x-sensor.org, Sep 20, 2011.

 ISBN: 978-1-61208-173-1

Modular P2P-Based Approach for RDF Data Storage and Retrieval

Imen Filali, Laurent Pellegrino, Francesco Bongiovanni, Fabrice Huet and Françoise Baude

INRIA-I3S-CNRS, University of Nice-Sophia Antipolis
2004 route des lucioles, Sophia Antipolis, France
imen.filali@inria.fr, laurent.pellegrino@inria.fr, francesco.bongiovanni@inria.fr
fabrice.huet@inria.fr, francoise.baude@inria.fr

***Abstract*—One of the key elements of the Semantic Web is the Resource Description Framework (RDF). Efficient storage and retrieval of RDF data in large scale settings is still challenging and existing solutions are monolithic and thus not very flexible from a software engineering point of view. In this paper, we propose a modular system, based on the scalable Content-Addressable Network (CAN), which gives the possibility to store and retrieve RDF data in large scale settings. We identified and isolated key components forming such system in our design architecture. We have evaluated our system using the Grid'5000 testbed over 300 peers on 75 machines and the outcome of these micro-benchmarks show interesting results in terms of scalability and concurrent queries.**

***Keywords*- Semantic Web; Peer-to-Peer (P2P); Resource Description Framework (RDF); RDF data indexing; RDF query processing**

I. INTRODUCTION

The Semantic Web [1] promises to deliver a new experience of the Web through the usage of more structurally complex data based on the Resource Description Framework (RDF) data model [2]. Realising this vision in large scale settings will be hardly feasible without proper and scalable infrastructures such as the ones proposed by the Peer-to-Peer (P2P) community in the last decade. More specifically, Structured Overlay Networks (SONs) such as CAN (Content Addressable Network) [3] and Chord [4] have proved to be an efficient and scalable solution for data storage and retrieval in large scale distributed environments [5], [6]. These overlays, which offer a practical Distributed Hash Table (DHT) abstraction, use a variant of consistent hashing [7] for assigning keys to nodes. Consistent hashing distributes the keys uniformly among all nodes, which provides a lookup performance of $O(\log N)$ where N is the total number of nodes in the network. However, such protocols can not handle more advanced queries such as partial keywords, wildcards, range queries, etc. because consistent hashing is not order-preserving; it randomly distributes lexicographically adjacent keys among all nodes. More advanced SONs such as P-Grid [8] and PHT [9] introduced the capability to handle more complex queries (e.g., range or prefix queries) but are still limited regarding the expressiveness of the queries they support.

The need to specifically manage a large amount of RDF data has triggered the concept of *RDF store*, which can be seen as a kind of "database" allowing to query RDF data using advanced query languages such as SPARQL [10]. The first generation for RDF data storage systems has spawned centralized RDF repositories such as RDFStore [11], Jena [12] and RDFDB [13]. Although these RDF stores are simple in their design, they suffer from the traditional limitations of centralized systems such as single point of failure, performance bottlenecks, etc. The Semantic Web community can benefit from the research carried out in P2P systems to overcome these issues. As a result, the combination of concepts provided by the Semantic Web and P2P together with efficient data management mechanisms seems to be a good basis to build scalable distributed RDF storage infrastructure. SPARQL is a very expressive language and supporting it in a distributed fashion is challenging. Various solutions based on P2P solutions have been proposed to process RDF data in a distributed way [14]–[16], but their architectures are rather complex and lack flexibility.

To meet the storage and querying requirements of large scale RDF stores, we revisit, in this paper, a distributed infrastructure that brings together RDF data processing and structured P2P concepts while keeping simplicity, reusability and flexibility in mind. The proposed architecture, based on a modified version of CAN [3], does not rely on consistent hashing. We chose to store the data in a lexicographical order in a three dimensional CAN which *(i)* eases RDF query processing, *(ii)* reflects the nature of RDF triples *(iii)* retains the benefits of deterministic routing [17].

The contributions of this paper are *(i)* the design of a fully decentralized P2P infrastructure for RDF data storage and retrieval, based on three dimensional CAN overlay, written in Java with the ProActive [18] middleware, *(ii)* the implementation of the proposed design with clear separation between the sub-components of the whole API (e.g., storage component, query processing, etc.). This modular architecture means it is possible to substitute sub-components (e.g., using another local RDF store). Finally, *(iii)* the evaluation of the proposed solution through micro-benchmarks carried out on the Grid'5000 test bed.

The remainder of the paper is organized as follows. In Section II, we give an overview of the related work for RDF data storage and retrieval in P2P systems. In Section III, we

 ISBN: 978-1-61208-173-1

introduce the proposed distributed infrastructure for RDF data storage and retrieval and present our data indexing and query processing mechanisms. The experimental evaluation of our approach is reported in Section IV. Finally, Section V concludes the paper and points out future research directions.

II. Related Work

Many P2P-based solutions have been proposed to build distributed RDF repositories [20]. Some of them are built on top of super-peer-based infrastructure as in Edutella [14]. In this approach, a set of nodes are selected to form the super-peer network. Each super peer is connected to a number of leaf nodes. Super-peer nodes manage local RDF repositories and are responsible for query processing. This approach is not scalable for two main reasons. First, the super-peer nodes are a single point of failure. Second, it uses the flooding-like search mechanism to route queries between super-peers.

By using DHTs, other systems, such as RDFPeers [15], address the scalability issue in the previous approach. RDFPeers is a distributed repository built on top of Multi-Attribute Addressable Network (MAAN) [24]. Each triple is indexed three times by hashing its subject, its predicate and its object. This approach supports the processing of atomic triple patterns as well as conjunctive patterns limited to the same variable in the subject (e.g., $(?\mathtt{s}, \mathtt{p}_1, \mathtt{o}_1) \wedge (?\mathtt{s}, \mathtt{p}_2, \mathtt{o}_2)$). The query processing algorithm intersects the candidate sets for the subject variable by routing them through the peers that hold the matching triples for each pattern.

From a topology point of view, the structure that comes closest to our approach is RDFCube [16], as it is also a three dimensional space of subject, predicate and object. However, RDFCube does not store any RDF triples. It is an indexation scheme of RDFPeers. RDFCube coordinate space is made of a set of cubes, having the same size, called *cells*. Each cell contains an *existence-flag*, labeled *e-flag*, indicating the presence (*e-flag=1*) or the absence (*e-flag=0*) of a triple in that cell. It is primarily used to reduce the network traffic for processing join queries over RDFPeers repository by narrowing down the number of candidate triples so to reduce the amount of data that has to be transferred among nodes. GridVine [19] is built on top of P-Grid [8] and uses a semantic overlay for managing and mapping data and meta-data schemas on top of the physical layer. GridVine reuses two primitives of P-Grid: `insert(key, value)` and `retrieve(key)` for respectively data storage and retrieval. Triples are associated with three keys based on their subjects, objects and predicates. A lookup operation is performed by hashing the constant term(s) of the triple pattern. Once the key space is discovered, the query will be forwarded to peers responsible for that key space.

From the data indexing point of view, almost of the proposed approaches for RDF data storage and retrieval use hashing approaches to map data into the overlay. However, even if this indexing mechanism enables the efficient key-based lookup, resolving more complex queries such as conjunctive queries may led to an expensive query resolution process.

III. CAN-Based Distributed RDF Repository

The aim behind the P2P infrastructure proposed in this work is the RDF data storage and retrieval in a distributed environment.

At the architectural level, it is based on the original idea of CAN [3]. A CAN is a structured P2P network based on a d-dimensional Cartesian coordinate space labeled $\mathcal{D}$. This space is dynamically partitioned among all peers in the system such that each node is responsible for storing data in a *zone* of $\mathcal{D}$. To store the $(\mathtt{k}, \mathtt{v})$ pair (`insert` operation in Figure 1), the key `k` is deterministically mapped onto a point `i` in $\mathcal{D}$ and then the value `v` is stored by the node responsible for the zone comprising `i`. The lookup (`retrieve` operation in Figure 1) for the value corresponding to a key `k` is achieved by applying the same deterministic function on `k` to map it onto `i`. The query is iteratively routed from one peer to its adjacent neighbors, with closest zones'coordinates to the searched key, until it reaches the node responsible for that key.

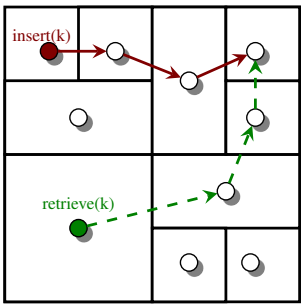


Figure 1. Routing in CAN: data storage (`insert`$(\mathtt{k}, \mathtt{v})$) and retrieval (`retrieve`$(\mathtt{k})$).

At the data representation level, data is presented in the RDF format [2]. RDF is a W3C standard aiming to improve the World Wide Web with machine processable semantic data. RDF provides a powerful abstract data model for structured knowledge representation and is used to describe semantic relationship among data. Statements about resources are in the form of <subject,predicate,object> expressions which are known as `triples` in the RDF terminology. The subject of a triple denotes the `resource` that the statement is about, the predicate denotes a `property` or a characteristic of the subject, and the object presents the `value` of the property. These triples, if connected together, form a directed graph where arcs are always directed from resources (subjects) to values (objects).

When designing an RDF data storage and retrieval, a set of key challenges have to be taken into account in order to come up with a scalable distributed RDF infrastructure. From the system scalability point of view, and unlike centralized solutions for massive RDF data storage

 ISBN: 978-1-61208-173-1

and retrieval which raise several issues (e.g., single point of failure, poor scalability), we argue that the use of a structured P2P overlay, at the architectural level, ensures the system's scalability. It also offers location transparency, that is, queries can be issued by any peer without any knowledge regarding the location of the stored data. Scalability needs to be achieved also at the query level by providing the ability to perform concurrent complex queries. As the data is expressed in RDF format, we use the SPARQL query language, which is another W3C recommendation [10], used to query RDF data. SPARQL queries could be in the form of:

- **atomic queries** are triples where the subject, the predicate and the object can either be variables or constant values. As an example, the query $\mathtt{q} = (\mathtt{s_i}, ?\mathtt{p}, ?\mathtt{o})$ looks, for a given subject $\mathtt{s_i}$, for all possible objects and predicates linked to $\mathtt{s_i}$. These kinds of queries are also called triple patterns.
- **conjunctive queries** are expressed as a conjunction of a set of atomic triple patterns (subqueries).
- **range queries** have specified ranges on variables. For instance, we consider the following query $\mathtt{q} = (<\mathtt{s}><\mathtt{p}>?\mathtt{o}\ \mathtt{FILTER}\ (\mathtt{v_1} \leq ?\mathtt{o} \leq \mathtt{v_2}))$ with a given subject s and a predicate p. It looks for a set of objects, given by the variable ?o, such as $\mathtt{v_1} \leq \mathtt{o} \leq \mathtt{v_2}$.

As stated earlier, the intrinsic goal behind a distributed RDF storage is to search for data provided by various sources. As a first step towards this direction, we would like to guarantee that the data can be found as long as the source node responsible for that data is alive in the network. This can be guaranteed by using a structured overlay model for distributed RDF data storage and retrieval. In this work, our distributed RDF storage repository relies on a three dimensional coordinate space where each node is responsible for a contiguous zone of the data space and handles its local data store. In the following section, we detail the data storage and retrieval process.

A. RDF Data Organization

The RDF storage repository is implemented using a three dimensional CAN overlay with lexicographic order. The three dimensions of the CAN coordinate space represent respectively the subject, the predicate and the object of the stored RDF triple. From now, let us denote by $\mathtt{z_{smin}}$,$\mathtt{z_{smax}}$, $\mathtt{z_{pmin}}$,$\mathtt{z_{pmax}}$, $\mathtt{z_{omin}}$ and $\mathtt{z_{omax}}$, the minimum (min) and the maximum (max) borders of a peer's zone according to the subject axis ($\mathtt{z_{smin}}$, $\mathtt{z_{smax}}$), the predicate axis ($\mathtt{z_{pmin}}, \mathtt{z_{pmax}}$), and the object axis ($\mathtt{z_{omin}}, \mathtt{z_{omax}}$). We say that a triple $\mathtt{t} =< \mathtt{s}, \mathtt{p}, \mathtt{o} > \in \mathtt{z}$ only if $\mathtt{z_{smin}} \prec \mathtt{s} \prec \mathtt{z_{smax}}$, $\mathtt{z_{pmin}} \prec \mathtt{p} \prec \mathtt{z_{pmax}}$ and $\mathtt{z_{omin}} \prec \mathtt{o} \prec \mathtt{z_{omax}}$; where s, p, o present respectively the subject, the predicate, and the object of the triple t and z is a zone of the CAN overlay. Doing so, a triple represents a point in the CAN space without using hash functions.

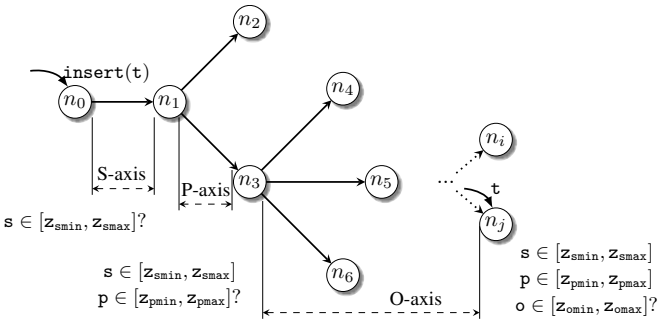


Figure 2. Insertion of RDF triples.

For better understanding, consider the example presented in Figure 2. Suppose that node n_0 receives an insert(t) request aiming to insert the RDF triple t in the network. Since no element of t belongs to the zone of n_0, and as s fits into the zone of n_1, n_0 routes the insert message to its neighbor n_1 according to subject axis ($\mathtt{S-axis}$). The same process will be performed by n_1, by means of the predicate axis ($\mathtt{P-axis}$) which in turn, will forward the message to its neighbor n_3. Once received, n_3 checks whether one of its neighbor is responsible for a zone such as o belongs to. Since the target peer is not found, the message will be forwarded at each step according to the object axis ($\mathtt{O-axis}$) to the neighbor with object coordinates which are closest to o. The idea behind this indexing mechanism is sketched in Algorithm 1.

Algorithm 1 Indexing algorithm

```
1:                                              ▷ Code for Peer P_i
2: upon event ⟨Insert | t⟩ from P_j
3:     if t ∉ Z_i then
4:         if s or p or o closer to one of my Neighbors' zones then
5:             send ⟨Insert | t⟩ to Neigh_dim
6:         end if
7:     end if
8: end event
```

This indexing approach has several advantages. First, it enables to process not only simple queries but also range queries. Using hashing functions in a DHT approach makes the management of such kind of queries expensive or even impossible. Moreover, in contrast to hashing mechanism that destroys the natural ordering of the stored information, the lexicographic order preserves the semantic information of the data so that it gives a form of clustering of triples sharing a common prefix. In other words, this approach allows that items with "close" values will be located in contiguous peers. As a result, range queries, for instance, can be resolved with a minimum number of hops. The routing

 ISBN: 978-1-61208-173-1

process of an `insert` operation consists in finding the peer managing the zone where the triple falls to. Routing `query` messages is slightly more complex and will be explained later in this section.
A closer look reveals that one downside with this approach is that it is sensitive to the data distribution. RDF triples with common prefixes might be stored on the same peer, i.e., a node can become a `hot zone`. In the case where an element is common to many triples, such as a frequently occurring predicate (e.g., $< \texttt{rdf} : \texttt{type} >$), the triples can still be dispatched on to different peers, depending on the values of the other elements. However, when some elements share the same namespace or prefix, the probability that they end-up on a very small subset of all available peers is very high. To avoid this potential issue, we try to automatically remove namespaces or prefixes and only use the remaining part for indexing and routing. Some care has to be taken when doing this because if done too aggressively, we might lose the clustering mentioned earlier. Note that this issue also appears in other P2P implementations which rely on prefix-based indexing with order-preserving hash functions [19].

In the general case, there are other solutions that can be used to mitigate the impact of skewed data. First, one can limit the CAN space if some specific information is known about the data distribution. For instance, if it is known that all subjects will have a prefix falling in a small interval, then it is possible to instantiate the overlay with the specified interval, avoiding empty zones. Second, if at runtime some peers are overloaded, it is possible to force new peers to join zones managing the highest number of triples, hence lowering the load. Some more advanced techniques have been proposed to deal with imbalance such as duplicating data to underloaded neighbors or having peers manage different zones [3].

B. RDF Data Processing

From the data retrieval perspective, efficient data lookup is at the heart of P2P systems. Many systems, such as Chord, relies on consistent hashing to uniformly store $(\texttt{key}, \texttt{value})$ pairs over the key space. However, consistent hashing only supports key-based data retrieval and is not a good candidate to support range queries since adjacent keys are spread over all nodes as stated earlier. Therefore, efficient lookup mechanisms are needed to support not only simple atomic queries but also conjunctive and disjunctive range queries. Hereafter, we detail how the queries are supported by our routing process:

- **atomic queries** are routed on the $\texttt{subject} - \texttt{axis}$ of the CAN overlay looking for a match on the subject value $\texttt{s}_\texttt{i}$. Once a peer responsible for the specified value $\texttt{s}_\texttt{i}$ is found, it forwards the query through its neighbors in the dimension where peers are most likely to store corresponding triples based on their zones's coordinates
- **conjunctive queries** are decomposed into atomic queries and propagated accordingly.
- **range queries** are routed by first identifying the constant part(s) in the query. Then the lowest and the highest values are located by going over the corresponding axis. If all results are found locally, they are returned to the query initiator. Otherwise, the query is forwarded to neighbors that may contain other potential results.

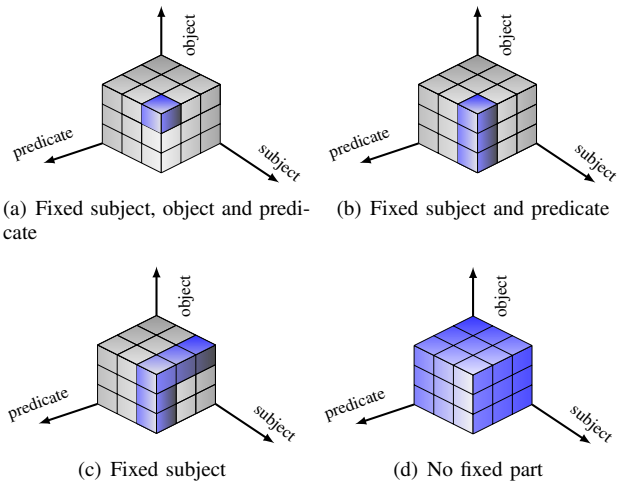
In order to route a query, a client sends it to a peer inside the overlay, the *query initiator*, as mentioned in Algorithm 2 at line 2. Once received, this query will be transformed, i.e., the initiator creates a message with additional information used for routing purposes (line 3), notably a *key* corresponding to the coordinates the message must be routed to. The next step consists of decomposing a complex query, a conjunctive query for instance, into atomic queries (line 5). Once we have these atomic queries, the peer sends messages, in parallel, to its neighbors accordingly, that is, if through them it can reach peers responsible for potential matches (lines 6 - 8). Whenever a peer has to propagate the message in different dimensions, it will de facto become a synchronization point for future results, that is, it waits for the results to come back and will merge the results before sending them to the client node (lines 15 - 21). In parallel of sending messages to its neighbors, the initiator will also check its local datastore in case it has potential matches for the query (line 10). Once neighbors receive a routing message (line 23), they will check their local datastore in case they can match the query (line 24) and return possible results to the initiator (line 26) otherwise they propagate the message to their neighbors accordingly (line 28). In order to ease the routing of the results, each message will embed the list of visited peers. This technique ensures that the forward path is the same as the backward path, avoiding potential issues related to NAT traversal, IP filtering, etc. that may happen in case we want to establish a direct connection to the initiator peer.

Figure 3 depicts various routing scenarios depending on the parts within the triple pattern. If subject, predicate and object are fixed, e.g., when performing an *add*, then the only peer which potentially holds matching results will be involved 3(a). In case the subject and the predicate are fixed, the message will have to traverse the object dimension in order to collect matching triples 3(b). When only the subject is fixed, the routed message will have to cross the object and predicate dimensions 3(c). Note that whenever a query with only variables is processed, our approach naively uses message flooding through each peer's neighbors. Hence, it may happen that a peer receives a message multiple times from different dimensions as shown in Figure 3(d). These duplicate messages will be ignored. Thanks to the way data is indexed and stored, queries are restricted to a specific subspace where candidate results are more likely to be

 ISBN: 978-1-61208-173-1

Algorithm 2 SPARQL queries routing algorithm

```
1:                                          ▷ Code for the Query Initiator
2:  upon event ⟨Query | Q⟩ from client
3:      RQ ← transformIntoRoutableQuery(Q)
4:      if RQ is a complex query then
5:          List of sub_q_i ← decomposeQuery(RQ)
6:          for each sub_q_i do
7:              send ⟨SubQuery | sub_q_i⟩ to Neigh_dim
8:          end for each
9:      end if
10:     if local RDFs matches Q then
11:         MergedRes ← MergedRes ∪{local matched RDFs}
12:     end if
13: end event
14:
15: upon event ⟨SubQueryResults | Res⟩ from Neigh_dim
16:     MergedRes ← MergedRes ∪ {Res}
17:     Pending_Sub_q ← Pending_Sub_q \ {Neigh_dim}
18:     if Pending_Sub_q == ∅ then
19:         send ⟨FinalRes | MergedRes⟩ to client
20:     end if
21: end event
22:                                  ▷ Code for the Neighbors_dim
23: upon event ⟨SubQuery | sub_q_i⟩ from Initiator or Neigh_dim
24:     if local RDFs matches sub_q_i then
25:         Res ← matched RDFs
26:         send ⟨SubQueryResults | Res⟩ to Initiator
27:     else
28:         send ⟨SubQuery | sub_q_i⟩ to Neigh_dim
29:     end if
30: end event
```



(a) Fixed subject, object and predicate

(b) Fixed subject and predicate

(c) Fixed subject

(d) No fixed part

Figure 3. Example of message scope depending on constant parts in the query.

found.

C. Modular Architecture

One of the goals when designing this distributed storage was to be able to easily change or modify some parts. A modular architecture is at the heart of the design, clearly separating the infrastructure (a CAN overlay), the query engine (using Jena [12]) and the storage system (a BigOWLIM [21] repository) as depicted in Figure 4. However, these elements do not work in isolation. Rather, they require frequent interactions. In this section, we will outline the different parts of our architecture, explaining their functions and showing their relations.

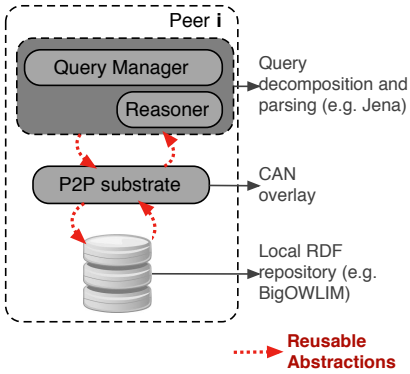


Figure 4. Modular architecture overview.

Query manager. Although the routing of the query is a P2P substrate's responsibility, part of the process requires the analysis of the query in order to extract atomic queries and their constant parts. This is performed using the Jena Semantic Web Framework [12] which provides dedicated operations. When a query returns data sets from multiple peers, the merge/join operation also relies on Jena. In order to experiment with the modularity aspect of our implementation, we have successfully swapped the query engine for Sesame [22] without impacting the other parts of the architecture.

P2P substrate. This "layer" is responsible for maintaining the CAN infrastructure, routing messages and accessing the local repository. The 3D CAN overlay is managed through an *Overlay* object which is responsible for maintaining a description of the zone managed by the current peer and an up-to-date list of its neighbors. Changing the number of dimensions of the CAN, e.g., to handle meta-data, requires providing a modified implementation of the *Overlay* object. To route a query, we first analyze it to determine its constant parts, if any, which will be used to direct it to the target peer. When there is not enough information to make a routing decision, it is broadcasted to the neighboring peers which will perform the same process.

Local storage abstraction. The local storage is ultimately responsible for storing data and processing queries locally.

 ISBN: 978-1-61208-173-1

It is important for the P2P infrastructure to be independent from the storage implementation. All references are isolated through an abstraction layer whose role is to manage the differences between data structures and API between the P2P and the storage implementations. Some requests require the access to the local repository to read or write some information. Although this is rather straightforward, some care has to be taken regarding the commit of data to the storage. With some implementations like BigOWLIM, committing can take some time and thus should not be done after each write operation. The peer can implement a policy to only perform them when a threshold is reached (e.g., the time passed since last commit aka *commit interval*, number of write done, etc.) or when a read query has to be processed.

IV. Experimental Evaluation

In order to validate our framework, we have performed micro-benchmarks on an experimental testbed, Grid'5000. The goal was twofold. First, we wanted to evaluate the overhead induced by the distribution and the various software layers between the repository and the end user. Second, we wanted to evaluate the benefits of our approach, namely the scalability in terms of concurrent access and the overlay size. All the experiments presented in this section have been performed on a 75-nodes cluster with 1Gb Ethernet connectivity. Each node has 16GB of memory and two Intel L5420 processors for a total of 8 cores. For the 300 peers experiments, there were 4 peers and 4 BigOWLIM repositories per machine, each of them running in a separate Java Virtual Machine.

A. Insertion of random data

Single peer insertion. The first experiment performs 1000 statements insertion and we measured the individual time for each of them, on a CAN made of a single peer. The two entities of this experiment, the caller and the peer, are located on the same host. The commit interval was set to $500\ ms$ and 1000 random statements were added. Figure 5(a) shows the duration of each individual call. On average, adding a statement took $2.074\ ms$ with slightly higher values for the first insertions due to cold start.

In a second experiment, the caller and the peer were put on separate hosts in order to measure the impact of the local network link on the performance. As shown in Figure 5(b), almost all add operations took less than $9\ ms$ while less than 6.7% took more than $10\ ms$. The average duration for an add operation was $6\ ms$.

Multiple peer insertion. We have measured the time taken to insert 1000 random statements in an overlay with different number of peers, ranging from 1 to 300. Figures 6(a) and Figure 6(b) show respectively the *overall* time when the calls are performed using a single or 50 threads. As expected, the more peers, the longer time is taken to add statements since more peers are likely to be visited before finding the target

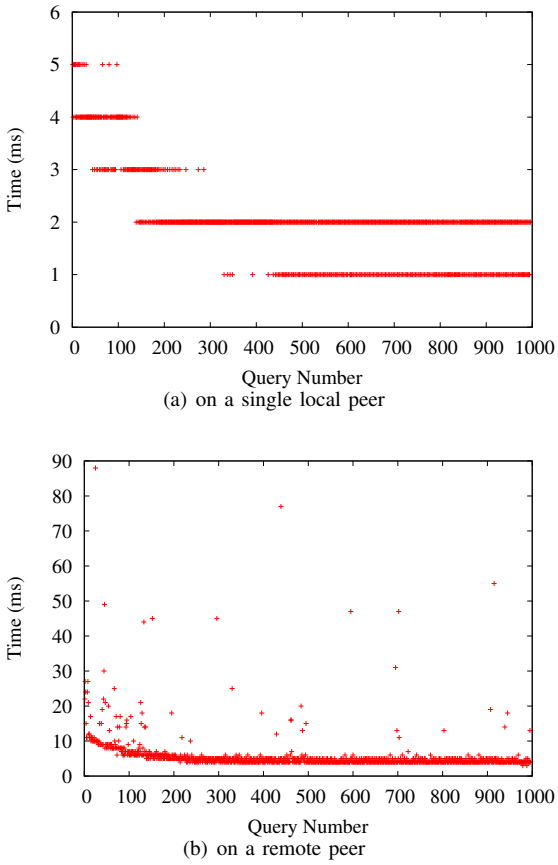


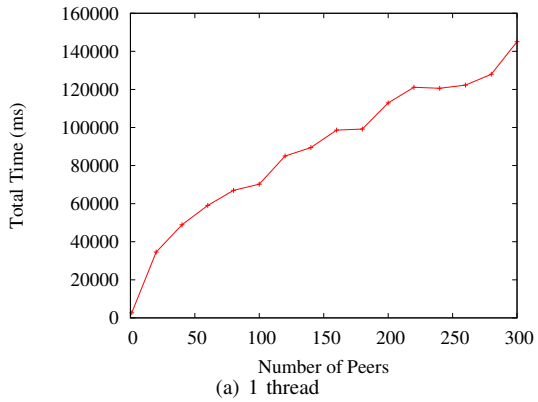
Figure 5. Insertion of 1000 statements with one peer.

one. However, when performing the insertion concurrently, the total time is decreased but still depending on the number of peers. Depending on the various sizes of the zones of the global space and the first peer randomly chosen for triple's insertion, the performance can vary, as can be seen with the small downward spike on Figure 6(b) at around 80 peers. To measure the benefits of concurrent access, we have measured the time to add 1000 statements on a network of 300 peers while varying the number of threads from 1 to 50. Results in Figure 7 show a sharp drop of the total time, clearly highlighting the benefits of concurrent access.

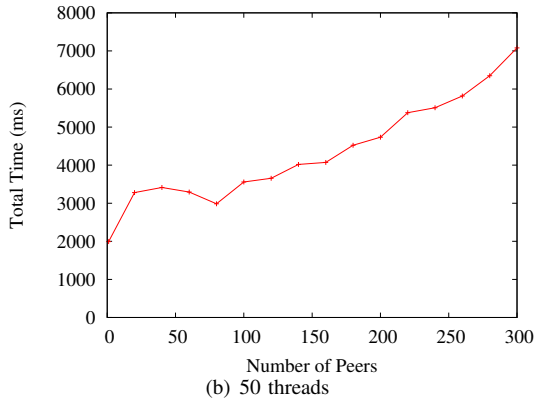
B. Queries using BSBM data

The *Berlin SPARQL Benchmark* (BSBM) [23] defines a suite of benchmarks for comparing the performance of storage systems across architectures. The benchmark is built around an e-commerce use case in which a set of products is offered by different vendors, with given reviews by consumers regarding the various products. The following experiment uses BSBM data with custom queries detailed below. The dataset is generated using the BSBM data generator for 666 products. It provides 250030 triples which are organized following several categories: 2860 Product Features, 14 Producers and 666 Products, 8 Vendors and

 ISBN: 978-1-61208-173-1



(a) 1 thread



(b) 50 threads

Figure 6. Insertion of 1000 statements for a variable number of peers.

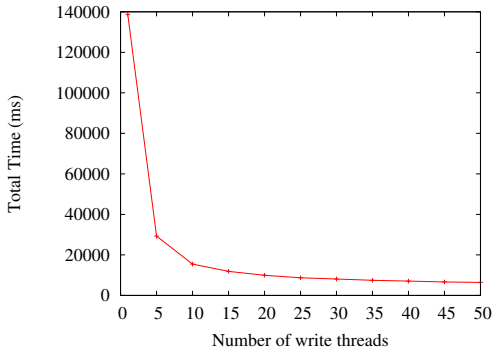


Figure 7. Evolution of the time for concurrent insertion with 300 peers.

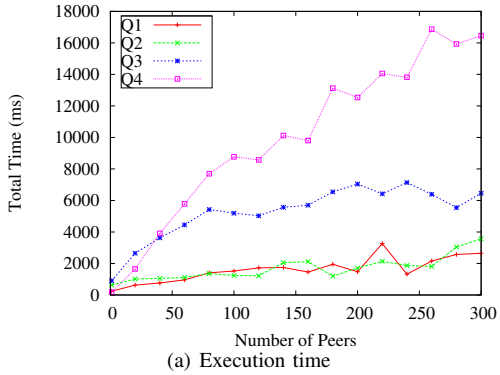
13320 Offers, 1 Rating Site with 339 Persons and 6660 Reviews. Out of this benchmark, we chose four queries to execute on our infrastructure:

- Q1 finds all the producers from Germany
- Q2 retrieves triples having "purl:Review" as object
- Q3 retrieves triples having "rdf:type" as predicate
- Q4 returns a graph where "bsbm-ins:ProductType1" instance appears

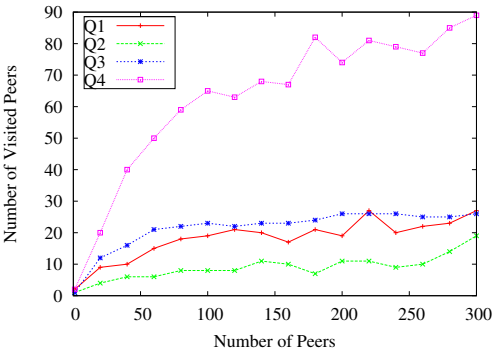
Q1 and Q4 are complex queries and will be decomposed into two subqueries. Hence, we expect a longer processing time for them. The number of matching triples for each query is as follows:

Query	Q1	Q2	Q3	Q4
# of results	1	6660	25920	677

Figure 8 shows the execution time and the number of visited peers when processing Q1, Q2, Q3 and Q4. Note that when a query reaches an already visited peer, it will not be further forwarded, therefore we do not count it. Q1 is divided into two subqueries with only a variable subject. Hence, it can be efficiently routed and is forwarded to a small number of peers. Q2 also has one variable and thus exhibits similar performance. Q3 has two variables so it will be routed along two dimensions on the CAN overlay, reaching a high number of peers. Since it returns 25920 statements, the messages will carry a bigger payload compared with other queries. Finally, Q4 generates two subqueries with two variables each, making it the request with the highest number of visited peers. In the network of 300 peers, the two subqueries have visited more than 85 peers.



(a) Execution time



(b) Messages overhead

Figure 8. Custom queries with BSBM dataset on various overlays.

Summary. Regarding the statement insertion into the distributed storage, although a single insertion has a low performance, it is possible to perform them concurrently, leading to a higher throughput. The performance of the query processing phase strongly depends on the number of subqueries, the payload carried between peers and the number of visited peers. While the payload depends on the complexity

 ISBN: 978-1-61208-173-1

of the query itself (conjunctive/atomic query, number of variables in a triple pattern, etc.), the number of visited peers depends not only on the structure of the overlay but also on the randomly chosen peer for initiating the query.

V. Conclusion

In this paper we have presented a distributed RDF storage based on a structured P2P infrastructure. RDF triples are mapped on a three dimensional CAN overlay based on the value of its elements. The global space is partitioned into zones and each peer is responsible for all the triples falling into it. We do not use hash functions, thus preserving the data locality. By removing constant parts such as prefixes from when indexing elements, we lessen bias naturally present in RDF data. The implementation has been designed with flexibility in mind. Our modular design and its implementation is abstracted away from the local storage and the query decomposer, thus they can swapped with other ones with minimal efforts. It also relies on standard tools and libraries for storing triples and processing SPARQL queries. We have validated our implemention with micro-benchmarks. Although basic operations like adding statements suffer from an overhead, the distributed nature of the infrastructure allows concurrent access. In essence, we trade performance for throughput. On a 75 nodes cluster, we have deployed an overlay of 300 peers. The time taken for query processing depends on the number of variable parts in the query and the size of the result set.

Future work. When queries have to be multicasted along different dimensions, the number of visited peers increases significantly, lowering the global performance. In this regard, we are currently working on an optimal broadcast algorithm, such as the one proposed in [25], which we adapt to CAN. This will allow us to decrease the number of redundant messages in case no constant parts are specified within the triple patterns of the query. Finally, we will investigate the impact of churn and node failures in our future experiments.

Code availability: The implementation mentioned in this paper is available at: http://code.google.com/p/event-cloud/.

Acknowledgment. The presented work is funded by the EU FP7 STREP project PLAY (http://www.play-project.eu) and French ANR project SocEDA (http://www.soceda.org)

References

[1] "W3C Semantic Web Activity," http://www.w3.org/2001/sw/, last accessed: July 2011.

[2] G. Klyne, J. Carroll, and B. McBride, "Resource description framework (RDF): Concepts and Abstract Syntax," *Changes*, 2004.

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content addressable network," in *Proceedings of SIGCOMM '01*, vol. 31, no. 4. ACM Press, October 2001, pp. 161–172.

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of SIGCOMM '01*. New York, NY, USA: ACM, 2001, pp. 149–160.

[5] A. Lakshman and P. Malik, "Cassandra: A Decentralized Structured Storage System," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, p. 3540, 2010.

[6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's Highly Available Key-Value Store," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 205–220, 2007.

[7] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of Computing*, 1997, pp. 654–663.

[8] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt, "P-Grid: a Self-Organizing Structured P2P System," *ACM SIGMOD Record*, vol. 32, no. 3, p. 33, 2003.

[9] S. Ramabhadran, S. Ratnasamy, J. Hellerstein, and S. Shenker, "Prefix Hash Tree: An Indexing Data Structure Over Distributed Hash Tables," in *PODC*, 2004.

[10] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," W3C Recommendation, January 2008, `http://www.w3.org/TR/rdf-sparql-query/`, last accessed: July 2011.

[11] "RDFStore," http://rdfstore.sourceforge.net/, last accessed: July 2011.

[12] J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: implementing the semantic web recommendations," in *World Wide Web conference*. ACM, 2004, pp. 74–83.

[13] R.V.Guha, "rdfDB: An RDF Database," http://guha.com/rdfdb/, last accessed: July 2011.

[14] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, "EDUTELLA: A P2P Networking Infrastructure Based on RDF," in *Proceedings of the 11 International World Wide Web Conference*, Honolulu, USA, May 2002.

[15] M. Cai and M. R. Frank, "RDFPeers: a Scalable Distributed RDF Repository Based on a Structured Peer-to-Peer Network," in *WWW*, 2004, pp. 650–657.

[16] A. Matono, S. Pahlevi, and I. Kojima, "RDFCube: A P2P-Based Three-Dimensional Index for Structural Joins on Distributed Triple Stores," *DBISP2P*, pp. 323–330, 2007.

[17] T. Schutt, F. Schintke, and A. Reinefeld, "Structured overlay without consistent hashing: Empirical results," in *Cluster Computing and the Grid Workshops, 2006. Sixth IEEE International Symposium on*, vol. 2, 2006, p. 8.

[18] "The ProActive middleware," http://proactive.inria.fr/, last accessed: July 2011.

[19] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt, "GridVine: Building Internet-Scale Semantic Overlay Networks," in *International Semantic Web Conference*, 2004.

[20] I. Filali, F. Bongiovanni, F. Huet and F. Baude, "A Survey of Structured P2P Systems for RDF Data Storage and Retrieval," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems III*, 2011, pp 20–55.

[21] D. M. Atanas Kiryakov, Damyan Ognyanov, "OWLIM : a Pragmatic Semantic Repository for OWL," 2005.

[22] A. K. Jeen Broekstra1 and F. van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," in *The Semantic Web - ISWC 2002 In Proceedings of the first Int'l Semantic Web Conference*, 2002, pp. 54–68.

[23] C. Bizer and A. Schultz, "The berlin sparql benchmark," 2009.

[24] M. Cai, M. Frank, J. Chen, and P. Szekely, "MAAN: A Multi-Attribute Addressable Network for Grid Information Services," in *Journal of Grid Computing*, vol. 2, 2003.

[25] S. El-Ansary, L. Alima, P. Brand, and S. Haridi, "Efficient broadcast in structured P2P networks," in *Peer-to-Peer Systems II*, 2003, pp. 304–314.

 ISBN: 978-1-61208-173-1

Formal Analysis and Verification of Peer-to-Peer Node Behaviour

Petter Sandvik[1,2] and Kaisa Sere[1]
[1]Department of Information Technologies, Åbo Akademi University
[2]Turku Centre for Computer Science (TUCS)
Joukahaisenkatu 3–5, 20520 Turku, Finland
{petter.sandvik,kaisa.sere}@abo.fi

***Abstract*—As services and applications move away from the one-to-many relationship of the client-server model towards many-to-many relations such as distributed cloud-based services and peer-to-peer networks, there is a need for a reusable model of how a node could work in such a network. We have constructed a reusable formally derived and verified model of a node in a peer-to-peer network for on-demand media streaming, validated and animated it, and then compared the results with simulations. We have thereby created an approach for analysing peer-to-peer node behaviour.**

Keywords-formal modelling; peer-to-peer; BitTorrent; on-demand streaming.

I. Introduction

There has been a trend in computer software towards a "utility computing vision" [1] in which computer services are accessed without needing to know the specific underlying structure. Rather than the traditional client-server architecture of network services, this vision is largely dependent on many-to-many relations such as distributed cloud-based services and peer-to-peer systems. However, the recent increase in peer-to-peer usage has highlighted a few issues when it comes to development of such services. Testing a peer-to-peer system can be difficult and cumbersome, due to the often large scale and heterogeneous nature of the system. In some cases simulations can be used, but designing a thorough simulation is not an easy task. Furthermore, both testing and simulating these types of systems may require us to emulate the whole network of interacting nodes even if our interest would lie with only one of them, such as when developing a new application designed to interact with a network of existing ones.

Our background in formal methods made us wonder if this problem could be approached from the opposite direction. By this we mean that instead of testing and simulating the whole network to confirm the correct behaviour of one node, we create a formally verified model of one node and then use that model for analysing peer-to-peer node behaviour in general. We will here look at a peer-to-peer on-demand media streaming system, in which content is divided into pieces, distributed between peers using piece selection methods based on BitTorrent on-demand media streaming [2], and then played back in-order. We describe the creation of models for three specific piece selection methods, based on a common reusable formally derived and verified model in Event-B [3]. We then show how ProB [4] can be used to animate our Event-B model, giving results that we can compare with results from simulations. Hence, we show how these different techniques and tools complement each other in the design task.

The rest of this article is organised as follows: In Section II, we describe the Event-B formalism and the tools we have used, and in Section III, we give an overview of on-demand streaming. Section IV details the creation of our formal model. In Section V, we show the results of animation, and compare them with results from previous simulations. We conclude this article in Section VI with discussion and future work.

II. Event-B, the Rodin Platform and ProB

Event-B [5] is a formalism based on Action Systems [6], [7] and the B Method [8]. The primary concept in formal development with Event-B is *models* [5]. A model in Event-B consists of *contexts*, which describe the static parts such as constants and sets, and *machines*, which contain the dynamic parts such as variables, invariants (boolean predicates on the variables), and *events*. An event contains *actions*, which describe how the values of variables change in the event, and *guards*, which are boolean predicates that all must evaluate to `true` before the event can be enabled, i.e., able to execute.

In Event-B development starts from an abstract specification, and the model is then *refined* stepwise into a concrete implementation. In order to achieve a reliable system we use superposition refinement [9], [10] to add functionality while preserving the overall consistency, which means that we add new variables and functionality in such a way that it prevents the old functionality from being disturbed [11]. In order to prove the correctness of each step of the development, we rely on the Rodin Platform [12] tool, which automatically generates *proof obligations*. These proof obligations, which are mathematical formulas that need to be proven in order to ensure correctness, can then be proven either automatically or interactively with the Rodin Platform tool. The choice of Event-B as the formalism to use for a model of this kind was largely due to this integrated tool support.

While the Rodin Platform tool is good for modelling and proving, we would also like to animate, or "execute" our models. This is because the mathematical correctness does not prove that our model does what we wanted it to do [5], and we would also like results that can be compared to those from simulations. ProB [4] is a free-to-use animator

 ISBN: 978-1-61208-173-1

and model checking tool, and supports models from both the B Method and Event-B. While ProB is available as a plug-in for the Rodin Platform tool, we have used the standalone, fully featured version for animation.

III. On-Demand Streaming

Streaming can be described as the transport of data in a continuous flow, in which the data can be used before it has been received in its entirety. There are two different approaches to streaming content; live streaming and on-demand streaming. From an end user perspective live streaming is similar to a broadcast; that is, everyone who receives the media is intended to receive the same content at the same time. On-demand streaming is different, in that it is "essentially playback, as a stream, of pre-recorded content" [13]. This makes on-demand streaming more similar to traditional file transfer. However, on-demand streaming is still "play-while-downloading" and not "open-after-downloading" [14], and traditional file sharing protocols can therefore not be used without modifications. This holds true especially if we look at peer-to-peer file sharing, where content is often transferred out-of-order.

The basis for the peer-to-peer media streaming solution we will look at is the file sharing protocol BitTorrent [15]. In BitTorrent, content is partitioned into pieces of equal size, and by default these pieces are requested out-of-order. By modifying the algorithms used to select the order in which the pieces of the content is requested, i.e. piece selection, BitTorrent can be made to work for streaming content. Several different modifications to the BitTorrent protocol to enable streaming media have been proposed, for instance BiToS [16] and Give-to-Get [17]. We have here chosen to model three different piece selection methods; *sequential*, *rarest-first with buffer* (RFB) and *distance-availability weighted* (DAW). Sequential represents a straightforward streaming solution, requesting pieces in their original order. While this is used for instance when streaming content in the OneSwarm friend-to-friend sharing application [18], BitTorrent contains a tit-for-tat incentive mechanism that requires data to be out-of-order to function as intended, and the sequential method may therefore be of limited use when unknown peers are involved. RFB is a modification of the rarest-first method used in BitTorrent file sharing, where the piece held by the fewest other peers is requested. The addition of a buffer means that a specific number of pieces after the piece currently being played back are requested with the highest priority, thus striving towards always having a certain amount of the content immediately available for playback. and only after that will the rarest piece be requested. DAW [2] tries to strike a balance between requesting rare pieces and pieces that are close to being played back, by calculating priority using the distance (i.e., difference in sequence number between a specific piece and the currently playing one) multiplied with the availability. If there is more than one piece with the same priority, the piece with the lowest piece number, i.e. closest to being played back, will be chosen in both RFB and DAW.

For streaming to work, we note that data cannot be received slower than it should be played back, and this is something that we must take into consideration when creating our model. In the following section, we describe a common Event-B model for the piece selection methods, and refine that model into three specific ones corresponding to the three mentioned piece selection methods.

IV. Modelling with Event-B

Entire peer-to-peer systems and other distributed architectures have been formally modelled [19], [20], [21]. We have created a reusable Event-B model for a node in an on-demand content streaming network [3]. The difference between the two approaches is that instead of looking at the whole network of peers, we model just how one peer looks at the system. Our idea when creating a model is to build it in separate layers, separating the functional parts from each other so that the model could easily be adopted for use with different functionality. Here we focus on modelling the piece selection methods.

As we model our peer-to-peer client as a client for streaming media, we see that three major functions are needed; piece selection (possibly out-of-order), piece transfer (possibly out-of-order) and playback (always in-order). These three functions are independent of each other, but must be performed in this sequence. Hence, pieces must be selected before they are transferred, and pieces must be transferred before they can be played back. An example situation is shown in Fig. 1. As mentioned previously, the content must be transferred at least as fast as it should be played back in order to ensure that streaming works. Therefore, we require that selection will always take place at the same rate as playback or faster; that is, for each time we advance playback we will have selected at least one additional piece.

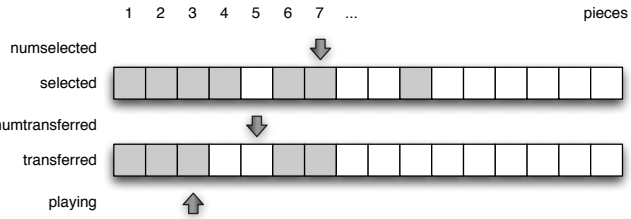


Fig. 1. *The relation between* `selected`, `transferred` *and* `playing`. *The arrows indicate the number of pieces (7 selected, 5 transferred and 3 playing), while the grey squares indicate the specific pieces.*

A. Common Model

We will start with a common model for all three piece selection methods [3], and here we will briefly describe the features of this model. We have two constants, `pieces` and `simreq`, which define how many pieces the content is divided into and the number of simultaneous requests, i.e., the maximum amount of pieces that can be selected but not yet transferred. We have variables for how many pieces we have selected (`numselected`) and exactly which pieces have been selected (`selected`), and similar variables for pieces that have been requested, i.e., transfer started, and for which the transfer has completed. We also keep track of which piece we are playing back and the priority for all pieces, as well as which piece we last updated priority for (`priupd`). The type restrictions of these variables are defined by invariants [3].

 ISBN: 978-1-61208-173-1

machine PieceSelect_M
variables playing completed numselected selected numtransferred transferred numrequested requested priority priupd
invariants
...
events
...
end

We initialise our common model with having zero pieces, and therefore not having selected, requested or transferred anything. Priorities are set initially for all pieces, but before the priorities are actually used they will be updated. This is done by an abstract event called CHANGE_PRIORITIES. which will be refined later. Initially, this event is enabled as long as we have not updated priorities for all possible pieces to request (@*grd4_1*) and for any p which is a non-zero natural number (@*grd4_2*). The priority of the piece after the previously updated one is then set to p (@*act4_1*), while the value of priupd is set to that piece (@*act4_2*). This means that we will update priorities of all pieces from the currently playing one to the last one.

event CHANGE_PRIORITIES ≙
any p
where
@*grd4_1* priupd < pieces
@*grd4_2* p ∈ ℕ1
then
@*act4_1* priority(priupd+1) ≔ p
@*act4_2* priupd ≔ priupd + 1
end

As the main focus in this model is piece selection, we will now look at the piece selection events. Because we require that content must be transferred faster than it should be played back, we also require that piece selection happens faster than playback. We have modelled this by separating the main flow of the program into two events: SELECT and SELECT_AND_ADVANCE. This means that the action taken in each step can be that of selecting a piece, or selecting a piece and advancing playback. In other words, every time something happens in our model we will select a piece, and some of those times we also advance playback. Naturally, after initialisation we always start with selecting a piece and only after that piece has been transferred could it be possible to advance playback.

The SELECT event is enabled when we have not yet selected as many pieces as we can (@*grd0_1* and @*grd2_4*) and when we have updated priorities for all pieces (@*grd4_5*). The parameter n must also be such that it can represent a piece we have not yet selected (@*grd1_2* and @*grd1_3*) and the priority for piece number n must be less than or equal to the priorities of all other possible pieces (@*grd4_6*). In practice, this means that the maximum priority that can be given to any piece is a numerical value of one, with higher numerical values being less prioritised. It also means that if there is more than one piece with the same priority, and that priority has the lowest numerical value of all priorities given to valid pieces, which one of these pieces to select is not determined. In our case, we will in the next refinements specify which of these pieces to select. However, the way the piece selection is modelled in this common model is actually consistent with the original BitTorrent specification, which does not specify an order when two or more pieces have the same availability [22].

What the SELECT event actually does is to increase the number of selected pieces (@*act0_1*), indicate that the specific piece has been selected (@*act1_2*) and reset the priupd variable so that we can update priorities before the next piece is selected (@*act4_3*).

event SELECT ≙
any n
where
@*grd0_1* numselected < pieces
@*grd1_2* n ∈ playing+1..pieces
@*grd1_3* selected(n) = FALSE
@*grd2_4* numselected − numtransferred < simreq
@*grd4_5* priupd = pieces
@*grd4_6* ∀k · (k ∈ playing+1..pieces ∧ k ≠ n ∧ selected(k) = FALSE ⇒ priority(n) ≤ priority(k))
then
@*act0_1* numselected ≔ numselected + 1
@*act1_2* selected(n) ≔ TRUE
@*act4_3* priupd ≔ playing
end

The SELECT_AND_ADVANCE event is very similar to the SELECT event, with the addition of guards and action concerning advancing playback. Thus, for this event to be enabled we require that we have not played all selected and transferred pieces (@*grd0_a* and @*grd2_c*), and that we have already selected and transferred the piece following the one currently being played back (@*grd1_b* and @*grd3_d*). The actions of this event are identical to the SELECT event, except for the addition of an action increasing the number of the currently playing piece (@*act0_a*) and therefore also requiring the increased value in the action that resets priority updates (@*act4_3*).

event SELECT_AND_ADVANCE ≙
any n
where
@*grd0_1* numselected < pieces
@*grd0_a* playing < numselected
@*grd1_2* n ∈ playing+1..pieces
@*grd1_3* selected(n) = FALSE
@*grd1_b* selected(playing+1) = TRUE
@*grd2_4* numselected − numtransferred < simreq
@*grd2_c* playing < numtransferred
@*grd3_d* transferred(playing+1) = TRUE
@*grd4_5* priupd = pieces
@*grd4_6* ∀k · (k ∈ playing+1..pieces ∧ k ≠ n ∧ selected(k) = FALSE ⇒ priority(n) ≤ priority(k))
then
@*act0_1* numselected ≔ numselected + 1
@*act0_a* playing ≔ playing + 1
@*act1_2* selected(n) ≔ TRUE
@*act4_3* priupd ≔ playing + 1
end

 ISBN: 978-1-61208-173-1

Our model also contains events which are not interesting in this context and therefore not shown here. We have events for pieces being requested and transferred, and in both we require that it is possible to perform the actions enabled by the event, which increase the number of requested or transferred pieces and mark the specific piece number as requested or transferred, respectively. The transfer event thereby updates variables that can be seen in the guards of the SELECT_AND_ADVANCE event. There is also an event that only advances playback after all pieces have been selected. We also have a final event which represents the conditions that must be true for the execution to terminate, which is that all pieces must have been selected, requested, transferred and played back. Only then will we set our variable completed to TRUE.

B. Three Piece Selection Models

Now that we have described our common model, we will take a look at our refined models which represent the use of three different piece selection methods.

1) The Sequential Piece Selection Method: The sequential piece selection method is very simple. Essentially, pieces are selected in order by setting the priority for a piece to its piece number. To model such a piece selection method we can use our common model as a basis, without needing any new variables, constants or events. In fact, the only change is refining the CHANGE_PRIORITIES event. The parameter p from the abstract event is here replaced by its concrete representation, priupd+1, which is the piece number of the piece for which we are changing priority. This necessitates the addition of a witness (@*p*), and we also remove the type guard for p.

event CHANGE_PRIORITIES_SEQUENTIAL ≙
  refines CHANGE_PRIORITIES
  when
    @*grd4_1* priupd < pieces
  with
    @*p* priupd+1 = p
  then
    @*act4_1* priority(priupd+1) ≔ priupd+1
    @*act4_2* priupd ≔ priupd + 1
end

2) The Rarest-First Method with Buffer: To model the rarest-first method with buffer (RFB) based on our common abstract model, we need to refine the abstract priority into a concrete one. As described in Section III, the priority in RFB is highest in the buffer, which consists of a fixed number of pieces after the playing one. Outside the buffer, the priority of each piece is set to the availability of that piece. Thus, we add a constant buffersize to describe the size of the buffer, and constants minavail and maxavail to describe the minimum and maximum values for piece availability. Availabilities must be larger than zero, because allowing zero availability for a piece would introduce additional complexity in piece selection and uncertainty as to whether all pieces could actually be transferred. We also need a new variable, availability, to describe the availability of each piece. We also add the following invariant, which states that when we have updated priorities for some but not all pieces, the pieces that we have updated priorities for and that are outside the buffer will have their priorities equal to their availability.

@*inv5_23* ∀t · (t ∈ playing+1..priupd ∧
  priupd < pieces ∧ t > playing+buffersize
  ⇒ (priority(t) = availability(t)))

Initially we set availability to minavail for all pieces. Because the availability is not controlled by us, we need an abstract event which changes the availability of a piece. This event should be enabled independently of piece selection, but not when updating priorities because they depend on the availability. The new event CHANGE_AVAILABILITY is enabled for any valid piece (@*grd5_1*) and availability (@*grd5_2*) whenever we have updated priorities for all pieces (@*grd5_3*), and sets the availability of that piece (@*act5_1*).

event CHANGE_AVAILABILITY ≙
  any n a
  where
    @*grd5_1* n ∈ 1..pieces
    @*grd5_2* a ∈ minavail..maxavail
    @*grd5_3* priupd = pieces
  then
    @*act5_1* availability(n) ≔ a
end

For changing priorities, we refine our abstract event into two separate events, CHANGE_PRIORITIES_BUFFER for setting priorities for pieces in the buffer, and CHANGE_PRIORITIES_RFB for the other pieces. The guard (@*grd5_3*) separates the two different events, ensuring that only one of them is enabled at a time. The parameter p from the abstract event is changed into a concrete one, necessitating the witness (@*p*) and removal of the guard stating the type of p. As can be seen both in the witnesses and in the actions (@*act4_1*), in these events the replacement for p is 1 and the availability of the piece, respectively.

event CHANGE_PRIORITIES_BUFFER ≙
  refines CHANGE_PRIORITIES
  when
    @*grd4_1* priupd < pieces
    @*grd5_3* priupd < playing + buffersize
  with
    @*p* 1 = p
  then
    @*act4_1* priority(priupd+1) ≔ 1
    @*act4_2* priupd ≔ priupd + 1
end

event CHANGE_PRIORITIES_RFB ≙
  refines CHANGE_PRIORITIES
  when
    @*grd4_1* priupd < pieces
    @*grd5_3* priupd ≥ playing + buffersize
  with
    @*p* availability(priupd+1) = p
  then
    @*act4_1* priority(priupd+1) ≔
      availability(priupd+1)
    @*act4_2* priupd ≔ priupd + 1
end

 ISBN: 978-1-61208-173-1

The SELECT and SELECT_AND_ADVANCE events both gain one guard. This guard corresponds to the requirement that if two pieces have the same priority, the one with the lowest piece number is selected.

```
@grd5_7 ∀j · (j ∈ playing+1..pieces ∧ j ≠ n
    ∧ selected(j) = FALSE ∧
    (priority(n) = priority(j)) ⇒ (n < j))
```

The remaining events do not need refining.

3) The Distance-Availability Weighted Method: When modelling the distance-availability weighted piece selection method (DAW), we can use our experience with modelling RFB as many parts are similar. In this refinement, the contexts of RFB and DAW are identical, and so are the variables. However, the invariant concerning priority (*@inv5_23*) is different. Here, the priorities we have updated outside the buffer should be set to distance times availability [2].

```
@inv5_23 ∀t · (t ∈ playing+1..priupd ∧
    priupd < pieces ∧ t > playing+buffersize
    ⇒ (priority(t) = (t − (playing+buffersize))
    ∗ availability(t)))
```

The CHANGE_AVAILABILITY event introduced in the refinement for RFB is abstract enough that it can be used as-is for DAW as well, but the big difference lies in the CHANGE_PRIORITIES event. Like in RFB, we refine the abstract event from our common model into two different events; one for pieces in the buffer and one for pieces outside the buffer. The CHANGE_PRIORITIES_BUFFER event for pieces in the buffer is, again, identical to the RFB one, as they both assign the highest priority to buffersize pieces after the playing one. However, the event that changes priorities for pieces outside the buffer is different. The parameter p from the abstract event is here replaced with a witness stating the corresponding concrete priority according to the DAW piece selection method.

```
event CHANGE_PRIORITIES_DAW ≙
  refines CHANGE_PRIORITIES
  when
    @grd4_1 priupd < pieces
    @grd5_3 priupd ≥ playing + buffersize
  with
    @p ((priupd+1) − (playing+buffersize))
        ∗ availability(priupd+1) = p
  then
    @act4_1 priority(priupd+1) ≔
        ((priupd+1) − (playing+buffersize))
        ∗ availability(priupd+1)
    @act4_2 priupd ≔ priupd + 1
end
```

The remaining events are identical to the RFB ones, which in some cases means that they are unchanged from the common model. The requirement that if two or more pieces have identical priority the one with the lowest piece number must be selected is also present in DAW.

C. Proof Obligations

Event-B models have certain properties, and when refining a model these properties need to be preserved in order for the model to remain correct. The Rodin Platform tool generates proof obligations, which are the mathematical formulas that need to be proven in order to ensure this correctness, including preserving the invariants and strengthening of the guards of our Event-B models. The Rodin Platform tool can automatically discharge most of these proof obligations by means of automatic provers, but some may need to be proven interactively, which the Rodin Platform also provides the means for. Table I shows the amount of proof obligations generated for each machine by version 2.0.1 of the Rodin Platform tool, and how many of those that needed user interaction. The Rodin Platform tool was used on a computer with a 2.4 GHz Intel Core 2 Duo processor running Mac OS X 10.5.8.

TABLE I
Proof Obligations of our Event-B Model.

Machine	Total Proofs	Interactive Proofs
PieceSelect_M	71	2
PieceSelect_M_SEQ	72	2
PieceSelect_M_RFB	92	3
PieceSelect_M_DAW	93	4

V. Validation, Animation and Comparison

As mentioned in Section II, we have used the standalone, fully featured version of ProB [4]. Due to the way our models are created and ProB interacts with them, memory and processing time constraints have forced us to use smaller values than we would in a real-life situation. Combined with limitations from simulations, this means that we look at the case when the content is divided into 20 pieces, only one request can be outstanding at any time, and the buffer size for RFB and DAW is set to 3. Although smaller than what would be used in a real-world situation, we believe that this is large enough to be noticeable but small enough not to impact the results.

We have compared the results from animating our models in ProB with the results from simple mathematical simulations [2], [13]. These simulations show the behaviour of the piece selection methods for the whole network, and here we chose a network of ten peers starting from scratch and one seed holding all the content. The results show how many of the peers hold each piece after twelve pieces have been selected, when selection happens twice as fast as playback.

Because our Event-B model looks at the network from the point of one node only, we have completed 40 random runs of the animation in ProB, and present the average results from these runs. The minimum availability was set to one and the maximum availability was set to five, and as in the simulation we have stopped to look at the situation after twelve pieces have been selected. Fig. 2 shows the results from both simulation and ProB animation for RFB and DAW piece selection methods.

The results for the sequential piece selection method are not shown, because the results from the simulation are identical

 ISBN: 978-1-61208-173-1

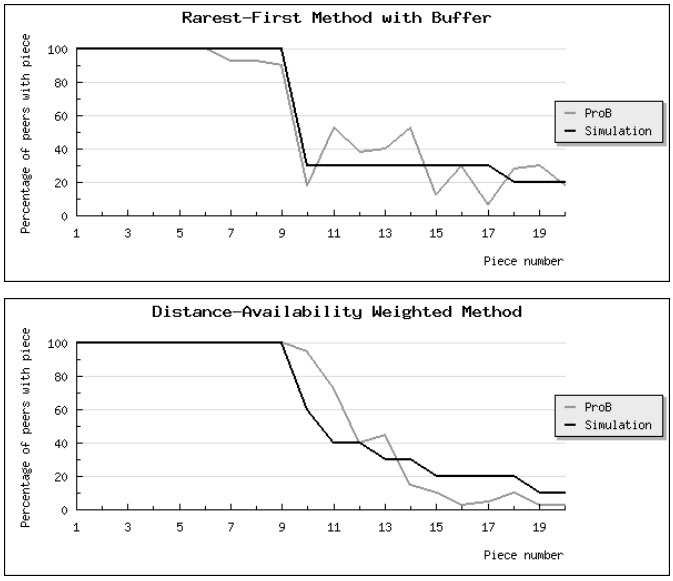


Fig. 2. *The availability of each piece after twelve pieces have been selected, when using RFB and DAW.*

to the ProB results, as should be expected. In both cases, the twelve first pieces are always selected after twelve pieces in total have been selected. For RFB and DAW, we note that the results from simulation and ProB animation are very similar. Some of the differences that exist are due to the fact that the simulations use the actual availability for each piece when calculating priority, while the animation uses random values corresponding to the nondeterminism in our Event-B model. Another difference regards the playback position. In the simulations, playback has always reached exactly piece number six after twelve pieces have been selected, because playback is advanced exactly every other time selection is done. In the animation, we start with selecting a piece, and after that we either select a piece or select a piece and advance playback with equal probability, which leads to variation in playback position but a mathematical average of 5.5. This means that unlike in the simulation, in the ProB animation an RFB node may not always have selected the 9 first pieces, and this is visible in Fig. 2. The very nature of DAW makes it unlikely, although not impossible, for the same thing to happen with DAW.

VI. Conclusions and Future Work

We have created a formally constructed and verified Event-B model of a node in a peer-to-peer content streaming network. This model we have then refined and animated, and the results have been compared with simulations. Using the same piece selection methods, our formal model of one peer has given us similar average results as a simulation of the whole network of peers. While we ran the ProB animation using smaller figures than would be used in a real-world situation, we still believe that our results show the added value that our method creates. By itself or together with simulations, animation of a formally derived and verified model can be used as an approach to analysing and verifying peer-to-peer node behaviour.

As our model is reusable, future work could include refining our model for other piece selection methods and extending the models and comparisons to other peer-to-peer systems besides an on-demand streaming one. Another possible direction would be refining our formal models to include the network structure in order to facilitate analysing aspects that require knowledge of more than just one node.

References

[1] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, pp. 599–616, 2009.

[2] P. Sandvik and M. Neovius, "The Distance-Availability Weighted Piece Selection Method for BitTorrent: A BitTorrent Piece Selection Method for On-Demand Streaming," in *Proceedings of AP2PS '09*, October 2009.

[3] P. Sandvik, K. Sere, and M. Waldén, "An Event-B Model for On-Demand Streaming," Turku Centre for Computer Science (TUCS), Tech. Rep. 994, December 2010, http://tucs.fi/publications/insight.php?id=tSaSeWa10a (Accessed September 2011).

[4] "The ProB Animator and Model Checker," http://www.stups.uni-duesseldorf.de/ProB/ (Accessed September 2011).

[5] J.-R. Abrial, *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.

[6] R.-J. Back and K. Sere, "From Modular Systems to Action Systems," *Software - Concepts and Tools*, vol. 13, pp. 26–39, 1996.

[7] M. Waldén and K. Sere, "Reasoning About Action Systems Using the B-Method," *Formal Methods in Systems Design*, vol. 13, pp. 5–35, 1998.

[8] J.-R. Abrial, *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[9] R.-J. Back and R. Kurki-Suonio, "Decentralization of Process Nets with Centralized Control," in *Proceedings of the 2nd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 1983, pp. 131–142.

[10] S. Katz, "A Superimposition Control Construct for Distributed Systems," *ACM Transactions on Programming Languages and Systems*, vol. 15(2), pp. 337–356, April 1993.

[11] K. Sere, "A Formalization of Superposition Refinement," in *Proceedings of the 2nd Israel Symposium on the Theory and Computing Systems*, June 1993.

[12] "Event-B and the Rodin Platform," http://www.event-b.org/ (Accessed September 2011).

[13] P. Sandvik, "Adapting Peer-to-Peer File Sharing Technology for On-Demand Media Streaming," Master's thesis, Åbo Akademi University, May 2008.

[14] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava, "On Peer-to-Peer Media Streaming," in *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, 2002.

[15] B. Cohen, "BitTorrent - A New P2P App," Yahoo eGroups, http://finance.groups.yahoo.com/group/decentralization/message/3160 (Accessed September 2011).

[16] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for Supporting Streaming Applications," in *9th IEEE Global Internet Symposium 2006*, April 2006.

[17] J. Mol, J. Pouwelse, M. Meulpolder, D. Epema, and H. Sips, "Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems," in *Multimedia Computing and Networking 2008, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 6818*, 2008.

[18] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Privacy-Preserving P2P Data Sharing with OneSwarm," in *SIGCOMM'10*, August–September 2010, pp. 111–122.

[19] L. Yan and J. Ni, "Building a Formal Framework for Mobile Ad Hoc Computing," in *Proceedings of the International Conference on Computational Science (ICCS'04)*, June 2004.

[20] L. Yan, "A Formal Architectural Model for Peer-to-Peer Systems," in *Handbook of Peer-to-Peer Networking 2010 Part 12*, X. Shen, H. Yu, J. Buford, and M. Akon, Eds. Springer US, 2010, pp. 1295–1314.

[21] M. Kamali, L. Laibinis, L. Petre, and K. Sere, "Self-Recovering Sensor-Actor Networks," in *FOCLASA*, 2010.

[22] B. Cohen, "Incentives Build Robustness in BitTorrent," in *1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

 ISBN: 978-1-61208-173-1

Video Quality Assurance for SVC in Peer-to-Peer Streaming

Mikko Uitto and Janne Vehkaperä
VTT Technical Research Centre of Finland
Kaitoväylä 1, 90571 Oulu, Finland
mikko.uitto@vtt.fi, janne.vehkapera@vtt.fi

***Abstract*—The Scalable Video Coding (SVC) has begun to arouse interest as a considerable alternative when streaming compressed video over wireless link. The main advantage of SVC comes with the scalability: one encoded sequence can contain multiple decodable sub-streams that allow adaptation to bandwidth fluctuation as well as terminal capabilities. Another growing phenomenon in video streaming is the utilization of peer-to-peer (P2P) technology, which benefits from its reduced costs and load on servers. However, similarly as in non-P2P networks, packet losses and transmission errors are possible, which sets specific need for error resilience especially in the decoder in order to provide sufficient Quality of Experience (QoE). This paper focuses on SVC transmission in error-prone P2P networks and represents our quality assurance strategies focusing mainly to the error concealment for the SVC decoder. The paper also evaluates the effectiveness of the proposed method via simulation setup where the decoder receives incomplete SVC streams that simulate the packet losses in P2P network.**

Keywords-SVC; scalable video coding; error concealment; P2P; peer-to-peer; QoE

I. INTRODUCTION

Video streaming due the fast development of Internet and video technology demands new ways of streaming high quality video to users with different network and terminal device capabilities. Indeed, in the side of need for better transmission technologies the increasing number of different end terminals has awaken the need for dedicated video compression technologies. One of the strong candidates is Scalable Video Coding (SVC), which requires only one encoding for the video, but multiple sub-streams can be decoded from the single stream. This allows not only considerable bandwidth adaptation but also excellent suitability into the receiving terminal. This means that the same video can be streamed both to high quality television with extremely high quality as well as low resolution mobile phones with lower quality.

Alongside the development of video compression, better ways for streaming real-time video are needed that reduce the load of the dedicated video servers [1]. Peer-to-peer (P2P) technology has aroused interest as an alternative transmission gateway when streaming video content among several users. One of the advantages comes with the non-dedicated server implementation since each user works as a server as well as a client. Peer-to-peer structure also reduces maintenance costs and provides simplicity although more work needs to be done in future with the security issues [2]. Content Delivery Network (CDN) support often large number of users likewise in P2P, but they require the deployment of special infrastructure [3].

Despite the fast development of powerful video codecs and streaming techniques a chance for transmission errors is always present due e.g., to network congestion, delay requirements and high video bitrate [4]. Additionally to these issues, temporary link failure can cause significant QoE degradation or even crash the decoder. Some of the research done for SVC quality assurance in P2P relies on controlling the errors via Forward Error Correction (FEC), Flow Forwarding (FF) or retransmission [1] without considering the loss potential still in the decoder. Available tools for maintaining the SVC video quality for spatially oriented streams via error concealment have been investigated [5] and also implemented to the old SVC reference software [6],[7]. However, these are mainly developed for individual frame losses and therefore require, for example, complex memory usage in order to maintain the previous pictures as a reference to the following pictures. Additionally, complex data structures in the error concealment can set hard limitation for the real-time performance, also for low-resolution streams. Furthermore, none of the existing quality assurance and error concealment techniques are designed especially to P2P streaming.

In this paper, we describe some of the work done in P2P-Next project, which is a research project funded partially by European Commission in the context of Framework Program 7 [8]. One of the goals in this project is to develop a P2P content delivery platform, the NextShare system with SVC support. Without going into deep in SVC integration into P2P architecture of the NextShare system we focus in this paper how to maintain the satisfying quality among the end users when packet losses are possible during the SVC transmission concentrating especially on the error concealment possibilities in the decoder.

The paper is organised in the following way. In Section 2, we describe SVC delivery in P2P architecture and present the NextShare platform. In Section 3, we provide our approach how to maintain video quality on a satisfying level for the end user. In Section 4, we evaluate our approach and compare it to the reference cases via simulation setup. Finally, Section 5 concludes the paper.

 ISBN: 978-1-61208-173-1

II. SVC IN PEER-TO-PEER ARCHITECTURE

The cost-effective solution of P2P has aroused widely interest as an alternative gateway for real-time video streaming. However, the number of such systems with full SVC support [9] is rare although the recent research has investigated this to some extent [10]. The majority of the research in this area, such as LayerP2P [11] does not consider SVC as the applied video codec but rely on non-layered codecs, such as H.264/AVC.

A. *SVC advantages*

The MPEG-4 Scalable Video Coding standard is an extension of the H.264/AVC standard (AnnexG) and provides a number of different layers within one encoded bitstream. While the H.264/AVC compliant base layer of a scalable bitstream provides the minimum quality, the enhancement layers are used to further increase the quality, resolution or frame rate of the bitstream [12]. Thus, a client only needs to receive a small part of the scalable bitstream to consume the data in low quality, while it has to receive and decode the complete scalable bitstream to consume the data in best quality. The usage of scalable codecs simplifies the adaptation of bitstream significantly, as an adaptation of such a bitstream can be performed by simply skipping some or all of the data related with enhancement layers.

The SVC base layer may be enhanced in three dimensions: the temporal dimension (frame rate), the spatial dimension (resolution) and the quality dimension (SNR) [12]. When considering networks with fluctuating bandwidth, especially temporal and SNR scalabilities enable powerful adaptation by diminishing the video bitrate. However, when several terminals with unique device capabilities exist also spatial scalability is a considerable alternative for saving the encoding time of various different types of sequences.

An essential feature of the design of the SVC extension is that the majority of the components of the H.264/AVC standard were adopted. This implies that transform coding, entropy coding, motion compensation, intra-prediction, the deblocking filter or the structure of the NAL units (NALU) are used as intended for the H.264/AVC standard [12]. One advantage of this approach is that the base layer of an SVC-encoded bitstream can generally be processed by a H.264/AVC compatible decoder, as the extensions of the H.264/AVC standard are only used to support spatial and signal-to-noise ratio (SNR) scalability.

B. *NextShare*

The NextShare, an open-source system, the next generation P2P content delivery platform, is developed in the P2P-Next project [8] and it has a fully support also to SVC. Basically, it follows the foundation of BitTorrent, but thanks to the NextShare development of state-of-the-art scientists, it can be now used not only for single layer streams, but also to multi-layered SVC streams. The basic principle and also a benefit in this platform is that the core won't require any changes if the video codec e.g., the decoder needs to be replaced to another. Additionally, the following error concealment as well as quality assurance technique presented in the next section is so called stand-alone algorithm that is not decoder dependent. This means that its integration is done basically to the decoder-player interface without requiring any major changes to the decoding process.

The simplified overall model of the producer-consumer side architecture can be seen in Fig. 1. The current SVC implementation in NextShare is designed to support both spatial and SNR layers. Likewise in the evaluation section of this paper, we have modelled the system with 4-layer mixed scalabilities where both the base layer BL and the spatial enhancement have one additional SNR layer (see TABLE 2). In the NextShare setup [9], we use 64 frames in a piece with 25 fps, with 3072 Kb/s for the highest VGA high-quality layer. Naturally, all the layers are mapped to pieces separately. The SVC encoder is optimised to have a constant bit rate with only one slice in a picture, because the coding efficiency suffers from using multiple slices [13].

Furthermore, the P2P engine that is responsible for creating and injecting the content into the network will not send the upper layer before the corresponding lower layer is sent [9]. In addition, the pieces are sent forward only if all the frames are received, which means that individual frame losses are not possible. Since the decoder will receive only "complete" group of pictures (GOPs) it guarantees in theory that the decoder should never crash. However, problems arise especially when spatially scalable video is streamed. First, the user may experience that the resolution varies in the player, which can be a very annoying phenomenon. Second, it is not always certain that the decoder is able to survive from the layer switching, especially if no Intra Decoding Refresh (IDR) pictures are used. This means that error concealment is needed to assure the video quality.

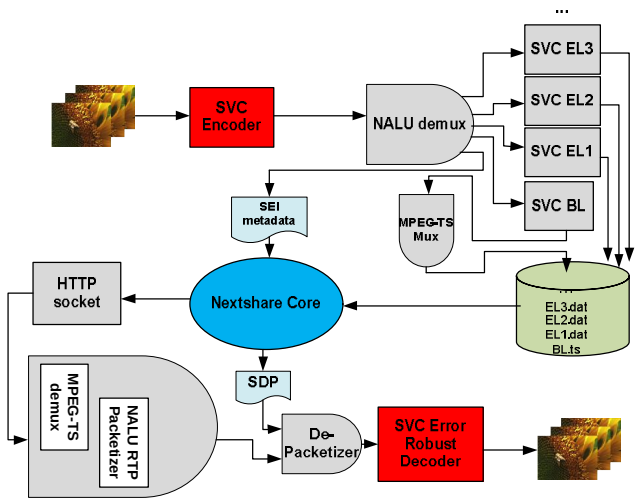


Figure 1. Simplified model of the producer-consumer side architecture.

 ISBN: 978-1-61208-173-1

III. PROPOSED QUALITY ASSURANCE TECHNIQUES

The available bandwidth does not always guarantee lossless transmission of the video stream. Especially in P2P networks the number of peers can vary causing total enhancement layer GOP losses in the receiving terminal. This means that the decoder must take these losses into an account and provide not only a stable and steady decoding process but also a satisfying video quality for the end users.

The error concealment in H.264/SVC decoder is to ensure complete decoding without crashes and to provide sufficient quality of experience. In some cases the spatial enhancement layer(s) cannot be received within the defined time slots, especially with high data rate videos. One problem in SVC decoder is that the first IDR packet usually defines the target resolution to be decoded: if the highest layer of spatially scalable stream cannot be received, it can crash the decoder or the resolution may vary from high to low, which can be very annoying phenomenon for human eye. Another viewpoint is that the hierarchical prediction structures in SVC can cause extensive error propagation.

We took the SVC reference decoder (version 9.15) [6] as a starting point and implemented error concealment for the decoder in order to provide good error robustness. Second, we concentrated on implementing picture upsampling techniques because the varying resolution in the player, such as in VLC, is a very provocative quality of experience.

Currently, JSVM reference software provides four separate upsampling algorithms with the picture resampling tool [14]. The first upsampling method is based on integer-based 4-tap filters that are originally derived from Lanczos-3 filter and arbitrary upsampling ratios are supported. On the contrary, the second method supports only dyadic upsampling ratios where the actual upsampling process is performed with several dyadic stages using also interpolation for the missing luminance and chrominance samples. The third method applies three-lobed Lanczos-windowed sinc functions and finally, the fourth method is the combination of AVC-half sample interpolation filter and bi-linear filter. [14]

We implemented the first method, integer-based 4-tap upsampling filter, as a separate function after the decoding process in order to enable an easy integration into different decoders. We optimised the time consuming blocks via benchmarking and achieved real-time algorithm, at least for VGA resolution video.

The use of IDR pictures is one easy way to break the decoding chain and check whether all the spatial layers are received. Therefore, we monitor the resolution of the first IDR picture, taking place as a first picture in each GOP. As was presented in Section II, the P2P engine sends only full GOPs to the consumer. So we basically upsample the whole GOP until the next IDR is received. Fig. 2 clarifies the upsampling process.

The actual upsampling process is simple; the algorithm solely takes the decoded picture as an input and upsamples it into the target resolution, which is defined in the Sequence Parameter Set (SPS) NAL packet. After this, the upsampled picture is directed either in the file writing process or to video output player, such as VLC. On this work, we focused principally implementing “portable” upsampling routine that guarantees satisfying end quality. The next step will be to apply and/or develop even better filters, which are state-of-the-art [15]. However, in this work we also benchmarked the JSVM filters and implemented the one providing the best end quality. This can be seen TABLE 1, which illustrates the sequence average Peak-Signal-to-Noise-Ratio (PSNR) comparison results in decibels (dB). Clearly our choice, the 4-tap filter, provides the best end quality both when upsampling the spatial base layer (BL) or its first quality enhancement layer (EL1) where Coarse-Grain Scalability (CGS) is used. The three test sequences will be introduced later in Section IV.

TABLE 1. JSVM upsampling filter comparison.

	PARKRUN		*SUNFLOWER*		*CREW*	
	BL	EL1	BL	EL1	BL	EL1
4-tap	21,80	23,04	28,46	33,02	29,48	32,50
Dyadic	21,43	22,39	27,51	30,27	29,20	31,68
Lanczos	21,43	22,37	27,52	30,28	29,20	31,66
Half-pel +bilinear	21,43	22,39	27,51	30,27	29,20	31,68

We did not want to focus only on simple spatial scalability when outlining the upsampling implementation. Instead of this, we used mixed spatial and quality layer scenario that was defined already in the project [9]. This enables a configuration where multiple receiving terminals with different device capabilities exist.

 ISBN: 978-1-61208-173-1

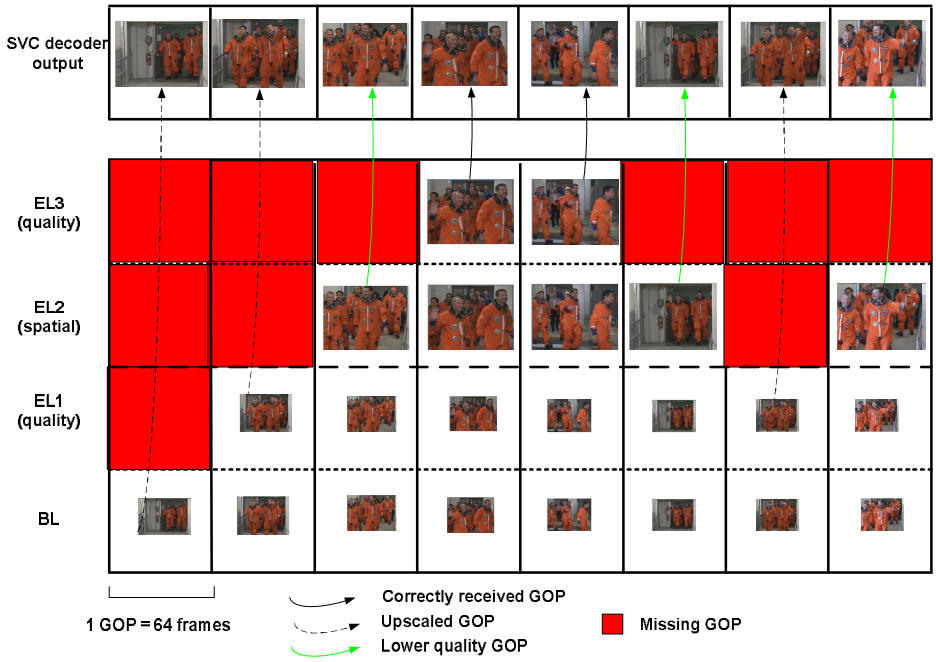


Figure 2. SVC upsampling process.

IV. EVALUATION

The evaluation section consists of the description of the simulations and their results. In addition, the corresponding SVC decoder is also integrated into the SVC prototype and the functionality of the error concealment is confirmed with the actual P2P setup as well [9].

A. Simulation setup

In order to evaluate the effectiveness of our quality assurance technique for the SVC decoder we developed also a packet loss generator as separate software that reads the SVC stream and drops packets with a certain loss ratio. For the purpose of P2P packet loss simulations we modified the software to drop whole GOPs from the stream. Basically the GOPs were dropped randomly but in a manner where the missing GOP and its higher enhancements were also discarded. Consequently, we were able to replicate comparable model for the P2P video decoder in the aspect of transmission errors. Once generating the GOP losses we decoded the output file with our modified SVC decoder and then measured the output PSNR. We repeated this simulation chain 50 times for each GOP loss ratio (2%, 5%, 10% and 20%) in order to average the PSNR values for each frame.

We chose three sequences with different characteristics mainly to have variety in the results (see Fig. 3). The *Parkrun* illustrates a running person both with steady slow motion, moving camera as well as static scenes with zero motion. The second sequence, *Sunflower*, contains only sharp motion in a small area both from the camera and the bee. The final sequence, *Crew*, contains lot of motion, bright lights and colors.

Figure 3. Test sequences.

The encoding parameters can be seen in TABLE 2. In order to have a variation to PSNRs and bitrates we encoded the test sequences without enabling the rate control. However, in the real demonstrator constant bit rate is applied but for our simulations for the error concealment it is not needed. We used CGS for quality enhancement laeyrs. As was illustrated in Fig. 2, we upsample the highest received GOP. If the GOP loss generator drops only the highest layer, then the output GOP is decoded with the quality from EL2. Furthermore, if EL1 is the highest received layer, we decode this one and use it as a reference for the upsampling algorithm.

TABLE 3 presents the encoded PSNRs for each layer. Naturally, when e.g., upsampling the EL1 (*Parkrun*) the end PSNR is not anymore 33.65 decibels for the high-resolution

 ISBN: 978-1-61208-173-1

image, because upsampling causes blurriness to some extent.

TABLE 2. SVC encoding parameters.

Number of layers	4
BL & EL1 resolution	QVGA (320x240)
EL2 & EL3 resolution	VGA (640x480)
GOP size (IDR period)	64
B frames	yes
Frame rate	30 fps
BL bitrate (*Parkrun*;*Sunflower*;*Crew*)	100; 70; 200 kb/s
EL1 bitrate (*Parkrun*;*Sunflower*;*Crew*)	700; 600; 800 kb/s
EL2 bitrate (*Parkrun*;*Sunflower*;*Crew*)	1300; 800; 900 kb/s
EL3 (full) bitrate (*Parkrun*;*Sunflower*;*Crew*)	4500; 2000; 2400 kb/s

TABLE 3. Encoded PSNRs for each layer.

	Parkrun	*Sunflower*	*Crew*
BL	27,96	29,55	30,80
EL1	33,65	35,61	35,50
EL2	26,87	32,51	32,49
EL3	32,96	37,31	36,08

B. Results

This section presents the results of the simulations where random GOP losses were injected to the three 1800-frame sequences. Fig. 4 – Fig. 6 show the PSNR-Y curves as a function of GOP loss ratio %. As can be seen in all simulation cases our proposed SVC quality assurance as well as error concealment technique outperforms the reference case, which was the so called “frame freeze” technique that can be widely seen in various video players. We can observe that the PSNR difference between the proposed and the reference case is only 2 dB (for 20% loss) for the *Parkrun*. This can be explained by the sequence characteristics where basically the video background and main target remain the same all the time creating smaller gaps between BL and EL3 PSNRs.

For the other two test sequences, the PSNR variation at 20% ratio is approximately 3-4 dBs better and it is clearly seen that the PSNR difference would increase for greater GOP loss ratios. Despite the fact that the end quality is significantly better as the PSNR values indicate, the visual quality, especially jerkiness, is extremely smooth without any freeze states in the video playback. As can be seen for the *Crew* sequence in Fig. 7 the overall quality improvement with the proposed method is significant.

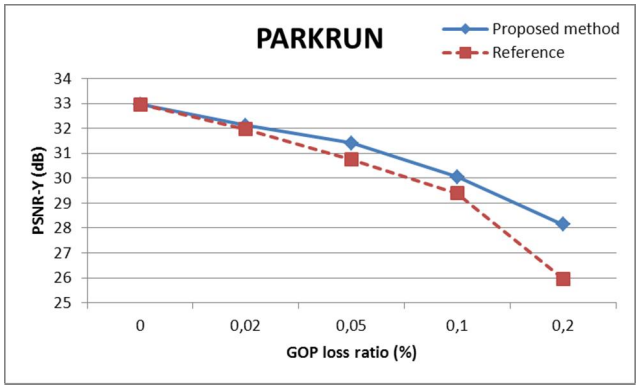


Figure 4. Average PSNR results for the *Parkrun* sequence.

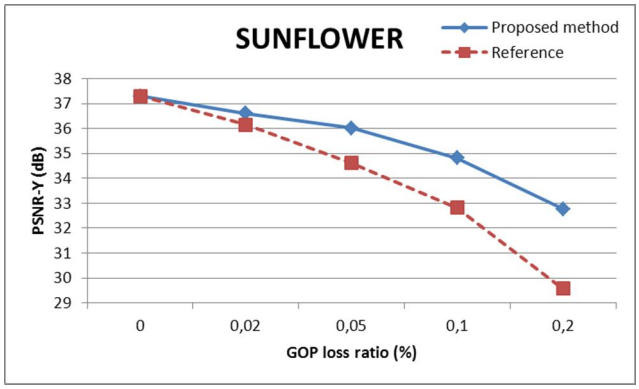


Figure 5. Average PSNR results for the *Sunflower* sequence.

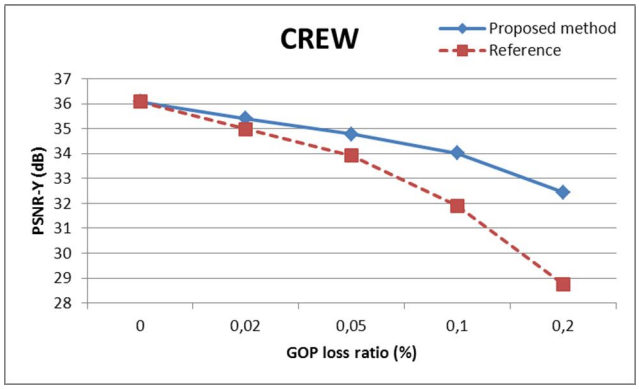


Figure 6. Average PSNR results for the *Crew* sequence.

 ISBN: 978-1-61208-173-1

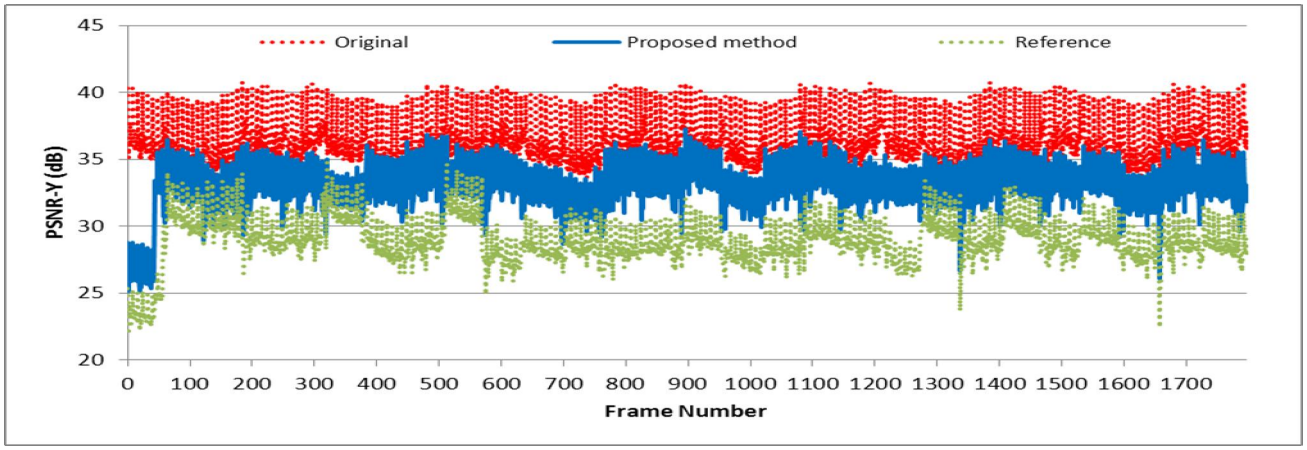


Figure 7. PSNR curve for Crew – sequence (1800 frames).

V. CONCLUSIONS

This paper investigated how to maintain the quality of experience in a good level for SVC streams in P2P streaming. The paper introduced the actual P2P platform developed in the P2P-Next project and its SVC relevance. We presented our implemented technique for video quality assurance focusing on the SVC decoder-side error concealment possibilities. In addition to the fact that the upsampling implementation is also running in the Nextshare SVC platform, we made our own simulation setup in order to evaluate the goodness of the quality assurance technique. The results show inevitably that our approach provides a lot smoother visual quality of experience compared to the traditional frame-freeze technique and also the computational values via PSNR curves proves that our method is applicable algorithm to be used in the SVC decoder. The proposed algorithm will be a portable block between any video decoder and player in future.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union's Seventh Framework Programme (P2P-Next) under grant agreement n° 216217. The authors would like to thank for the support.

REFERENCES

[1] T. Schierl, Y. Sanchez, C. Hellge, and T. Wiegand, “Improving P2P Live-Content Delivery using SVC”, Proceedings of IEEE Visual Communications and Image Processing (VCIP'10), Huang Shan, An Hui, China, pp. 1-10, July 2010.

[2] P. Sanjoy, “Digital Video Distribution in Broadband, Television, Mobile, and Converged Networks: Trends, Challenges and Solutions”. John Wiley & Sons Ltd, West Sussex, United Kingdom, pp. 1-384, 2011.

[3] P. Bacciechet, T. Schierl, T. Wiegand, and B. Girod, “Low-delay Peer-to-Peer Streaming using Scalable Video Coding”, Packet Video 2007, Nov. 2007, pp. 173-181, doi: 10.1109/PACKET.2007.4397039

[4] B. Li and H. Yin, “Peer-to-Peer Live Video Streaming on the Internet: Issues, Existing Approaches and Challenges”, IEEE Communications Magazine, Vol. 45, June 2007, pp. 94-99, doi: 10.1109/MCOM.2007.374425

[5] Y. Guo, Y. Chen, Y.-K. Wang, H. Li, M. M. Hannuksela, and M. Gabbouj, “Error Resilient Coding and Error Concealment in Scalable Video Coding”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 19, pp. 781-795, Jun. 2009, doi: 10.1109/TCSVT.2009.2017311

[6] SVC Reference Software (JSVM software). [Online]. Available:
http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm, accessed on 23/06/2011

[7] Y. Chen, K. Xie, F. Zhang, P. Bandit, and J.Boyce,” Frame loss error concealment for SVC”, Journal of Zhejiang University – Science A 7, pp. 677-683, December 2006.

[8] The P2P-Next Project, FP7-ICT-216217, http://www.p2pnext.org, accessed on 23/06/2011

[9] N. Capovilla, M. Eberhard, S. Mignanti, R. Petrocco, and J. Vehkaperä, “An Architecture for Distributing Scalable Content over Peer-to-Peer Networks”, Proc. Advances in Multimedia (MMEDIA), 2010, pp.1-6, June 2010, doi: 10.1109/MMEDIA.2010.17

[10] M. Mushtaq and T. Ahmed, “Smooth Video Delivery for SVC based Media Streaming over P2P Networks”, Proc. Consumer Communications and Networking Conference 2008 CCNC 2008 5th IEEE, pp. 447-451, Jan.2008.

[11] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, “LayerP2P: Using Layered Video Chunks in P2P Live Streaming”, IEEE Transactions on Multimedia, Vol. 11, pp. 1340 – 1352, August 2009.

[12] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard", IEEE Trans. on CSVT, vol. 17, no. 9, pp. 1103-1120, September 2007.

[13] M. Uitto, J. Vehkaperä, and P. Amon, ” Impact of Slice Size for SVC Coding Efficiency and Error Resilience”, 6th International Mobile Multimedia Communications Conference (MobiMedia 2010, SVCVision Workshop). Lisbon, Portugal, pp. 1-15, 6-8th September 2010.

[14] JSVM 9.15 Software Manual, 2008.

[15] Q. Shan, Z. Li, J. Jia, and C.-K. Tang, “Fast Image/Video Upsampling”, ACM Transactions on Graphics (SIGGRAPH ASIA 2008), Vol. 27, pp. 1-8, November 2008.

 ISBN: 978-1-61208-173-1

Pair-wise Similarity Criteria for Flows Identification in P2P/non-P2P Traffic Classification

José Camacho, Pablo Padilla, F. Javier Salcedo-Campos, Pedro García-Teodoro, Jesús Díaz-Verdejo
Dpt. of Signal Theory, Telematics and Communications,
CITIC - Faculty of Computer Science and Telecommunications - University of Granada,
C/ Periodista Daniel Saucedo Aranda s/n 18071 GRANADA (Spain).
josecamacho@ugr.es, pablopadilla@ugr.es, fjsalc@ugr.es, pgteodor@ugr.es, jedv@ugr.es

***Abstract*—There is a growing interest in network traffic classification without accessing the packets payload. A main concern for network management is *peer-to-peer* (*P2P*) traffic identification. This can be performed at several levels, including packet level, flow level and node level. Most current traffic identification approaches rely on flow level identification, being highly demanding and time consuming procedures. This paper introduces a similarity-based method to pair flows up, which is aimed at reducing the cost of identifying P2P/non-P2P traffic flows. For that, different similarity measures for flows pairing are proposed and analyzed.**

***Keywords*-Traffic classification; peer-to-peer; k-Nearest Neighbors**

I. INTRODUCTION

The increasing popularity and expansion of peer-to-peer (P2P) networks and applications has raised some engineering issues related to traffic and security. On the one hand, Internet service providers need to handle the large volume of traffic yielded by P2P activities to assure the minimal impact to other network services. Moreover, the exchange of any kind of information between the so-called peers, most of them anonymous, is a security risk. This risk affects users in particular, since the information exchanged might contain viruses, worms and malware. It also affects the network infrastructure, since P2P applications can be used to support other harmful activities such as coordinated DoS attacks, botnets, etc.

In this context, there is a clear interest in P2P traffic identification. This paper introduces a new method aimed at reducing the cost of identifying P2P/non-P2P traffic flows. The rest of the article is organized as follows: Section II reviews the state of the art of traffic classification and P2P traffic identification. Section III introduces the datasets used in the experimentation. Section IV motivates the use of macro-flows built upon pairs of flows. In Section V some strategies for flows pairing are presented. Section VI is devoted to compare the results obtained by these different strategies, and finally, the conclusions are drawn in Section VII.

II. STATE OF THE ART

The recognition of P2P traffic is part of a more general problem, namely the identification of network traffic [1]. Three main problems arise in the identification of the traffic on a network:

1) Characterization: There are many features that have been proposed in the literature to represent and classify network traffic. The information used includes a wide variety of parameters, from statistical data of connections from SNMP routers reports [2] (low granularity) to information obtained from TCP headers, including the signaling bits and the first bytes of payloads (high granularity) [3].
2) Identification level: Once the traffic has been parameterized, three levels are considered to perform the identification [1], [4]: node level, packet level and flow level. In the first case, the objective is to identify nodes that generate a certain type of traffic [5]. The aim of packet-based identification is to classify each packet individually. In the flow-based identification, the goal is to determine the application protocol that generates each traffic flow.
3) Identification process: A wide variety of recognition systems are used to perform the identification, ranging from heuristic or signature-based [1], [6], [7] to data mining or pattern recognition algorithms [4], [8].

P2P flow recognition has been attempted by using a number of techniques. Among them, the *k*-Nearest Neighbors (kNN) technique is remarkable because of its simplicity and high recognition rate reported. Jun et *al.* [9] performed a comparison between a number of techniques including Naïve Bayes, decision trees, kNN and other methods to classify flows from 12 different application protocols, where some of them are P2P (BitTorrent and Gnutella) and the rest non-P2P (HTTP, DNS, POP3, etc.). The results show that kNN is the best technique in terms of precision rate. Lim et *al.* [10] proposed a discretization of standard parameters of traffic flows (ports, package sizes, number of packets, duration of flow, etc..) and assessed four classification techniques: support vector machines (SVMs), kNNs, Naïve Bayes and

 ISBN: 978-1-61208-173-1

decision trees. The results indicate that the performance of kNN is similar to SVMs, which yielded the best performance. Salcedo-Campos et *al.* [11] proposed a kNN-based technique called MVC (Multiple Vector Classification) for P2P traffic identification. This method combines three kNNs applied over different sets of parameters obtained from the flows.

Most current traffic classification approaches rely on flow level techniques. Despite the good classification performance usually obtained by them, the general process is highly time consuming. In order to overcome such limitation, this paper introduces several similarity measures for flows pairing in order to identify groups of flows likely to be generated by the same protocol/service. This way, once a flow is identified with a well-known procedure (e.g., DPI tools), all the flows which are similar to it according to the flows pairing will also be (quickly) identified. The proposed pairwise approach for flows classification takes advantage of the good performance exhibited by kNN classifiers.

III. Network traffic data for the experimentation

In order to evaluate the approach and methods described in this work, an experimental setup with two steps has been considered. The first one includes the capture of a great amount of real network traffic, in this case, acquired in an academic institution network. The second one consists of the automatic classification of all the captured traffic packets and flows by means of a deep packet inspection (DPI) tool. In this scenario, the *ground truth* data-set is constituted taking into account the analysis and identification of each traffic flow and its associated traffic packets with a DPI tool, in this case openDPI [12], with a negligible percentage of classification errors.

The database used in this work contains the data captured during three days of network inspection in an academic institution. The acquisition was performed in the access router in order to control the incoming and outgoing traffic of the inner nodes of the network. The traffic flows and their packets are captured in both communication ways.

The original data-set has been divided into a calibration subset of 100,000 flows and a test subset with 100,000 flows. It should be remarked that the flows are sequentially organized so that the period corresponding to the calibration subset is previous to the one of the test subset, with no time period overlapping. Table I shows the amount of P2P traffic in both calibration and test subsets. OpenDPI tool found 35 and 41 different protocols in the calibration and test subsets, respectively.

The openDPI classification shows that HTTP is the protocol with the highest number of flows, while the portion of P2P protocols is close to 9%. Although this P2P fraction could be considered reduced, the P2P traffic volume associated is high, due to the size of each P2P flow. A more detailed analysis shows that only a reduced number of network nodes generate or receive P2P traffic, being more relevant videostreaming related protocols, which contribute to the HTTP traffic (i.e., YouTube traffic). The rest of non-P2P flows include mainly habitual protocols, such as DNS, SSL or email protocols. The majority of the P2P flows are related to BitTorrent, meanwhile Gnutella and others are found in a lower proportion. This proportion may be considered a consequence of the particular features of the protocols. The relation between the P2P traffic and the non-P2P traffic is similar in both calibration and test data-sets. Please, refer to [11] for a more detailed description of the protocols in the data set.

Table I
BASIC TRAFFIC DESCRIPTION OF THE CALIBRATION AND TEST SUBSET.

	Flows		
Subset	Total	P2P flows	non-P2P flows
Calibration	100,000	8,897	91,103
Test	100,000	8,916	91,084
Total	200,000	17,813	182,187

The feature vector representing each flow is composed of 61 variables, as Table II depicts. The feature vectors contain all the information needed for posterior analysis, including the flow identification label in the database, the protocol detected by openDPI and some traffic information concerning the flow. The IP addresses of each flow have been sorted by number. The term UP (ascending) points out that the packet is going towards the machine with the highest IP, and the term DOWN (descending) indicates that the packet is going towards the machine with the lowest IP.

In the rest of the text, the terms observation and feature vectors are used interchangeably. Two levels of classification are established: the first one considering all the protocols in the subsets and the second one considering only two classes indicating if the flows are related to P2P or non-P2P protocols.

IV. Motivation

In kNN classification, an object is assigned to the most common class amongst its k nearest neighbors. In this section, the kNN technique in its simplest form (k=1) will be applied to the calibration data-set in order to motivate the approach adopted in this paper. From here onwards, let us call this the NN technique. The NN classifies an observation (feature vector or flow) within the same class of the nearest observation in the calibration data. To establish the nearest observation to a given one, a closeness functional needs to be defined, typically based on well-known distances. An often used distance is the normalized Euclidean distance, where all variables have been normalized in variance:

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^t \cdot \mathbf{S}^{-1} \cdot (\mathbf{x}-\mathbf{y})} \qquad (1)$$

 ISBN: 978-1-61208-173-1

Table II
VARIABLES OF THE FEATURE VECTOR FOR EACH FLOW.

Value	Description
	Flow identification
ID_FLOW	Flow ID
IP_LOW	Lower IP of the session tuple
IP_UPPER	Highest IP of the session tuple
PORT1	Port related to the lowest IP (IP_LOW)
PORT2	Port related to the highest IP (IP_UPPER)
PROT_UDP	Transport protocol UDP
PROT_TCP	Transport protocol TCP
PROT_UNK	ICMP
DIR	Direction of the first observed packet (UP or DOWN)
FIRST_TIME	Timestamp of the first packet (μs)
LAST_TIME	Timestamp of the last packet (μs)
	Related to transfer
NPACKETS	Number of packets in flow
NPACKETS_UP	Idem way UP
NPACKETS_DOWN	Idem way DOWN
PACKETS_SIZE	Complete size of all the packets in the flow
PACKETS_SIZE_UP	Idem way UP
PACKETS_SIZE_DOWN	Idem way DOWN
PAYLOAD_SIZE	Complete size of payloads
PAYLOAD_SIZE_UP	Idem way UP
PAYLOAD_SIZE_DOWN	Idem way DOWN
MEAN_PACK_SIZE	Mean packet size
MEAN_PACK_SIZE_UP	Idem way UP
MEAN_PACK_SIZE_DOWN	Idem way DOWN
SHORT_PACKETS	Number of short packets
SHORT_PACKETS_UP	Idem way UP
SHORT_PACKETS_DOWN	Idem way DOWN
LONG_PACKETS	Number of large packets
LONG_PACKETS_UP	Idem way UP
LONG_PACKETS_DOWN	Idem way DOWN
MAXLEN	Maximum packet size
MAXLEN_UP	Idem way UP
MAXLEN_DOWN	Idem way DOWN
MINLEN	Minimum packet size
MINLEN_UP	Idem way UP
MINLEN_DOWN	Idem way DOWN
	Related to time
DURATION	Flow duration (μs)
MEAN_INTERAR	Mean time between consecutive packets
MEAN_INTERAR_UP	Idem only UP
MEAN_INTERAR_DOWN	Idem only DOWN
MAX_INTERAR	Maximum time between consecutive packets
MAX_INTERAR_UP	Idem only UP
MAX_INTERAR_DOWN	Idem only DOWN
MIN_INTERAR	Minimum time between consecutive packets
MIN_INTERAR_UP	Idem only UP
MIN_INTERAR_DOWN	Idem only DOWN
	Signaling
N_SIGNALING	Number of packets containing flags
N_SIGNALING_UP	Idem way UP
N_SIGNALING_DOWN	Idem way DOWN
NACKS	Number of packets with ACK flag active
NFIN	Idem FIN
NSYN	Idem SYN
NRST	Idem RST
NPUSH	Idem PSH
NURG	Idem URG
NECE	Idem ECE
NCWD	Idem CWD
NACK_UP	Number of packets UP with ACK flag active
NACK_DOWN	Idem way DOWN
NFIN_UP	Idem FIN & UP
NFIN_DOWN	Idem FIN & DOWN
NRST_UP	Idem RST & UP
NRST_DOWN	Idem RST & DOWN

where $\mathbf{S}$ is a diagonal matrix containing the sampling variances of the variables.

In Figure 1, the performance of the NN technique for traffic classification (Figure 1(a)) and P2P traffic identification (Figure 1(b)) is assessed with the calibration data following two different approaches. The first approach considers the traffic corresponding to the first hour as the calibration data for NN. Then, traffic classification and P2P identification are performed over the rest of the flows up to the 20th hour. Notice that the first 20 hours correspond to the calibration subset introduced in the previous section. The test subset is only employed in the experiments of Section V. The second approach considers a sliding window of one hour as the calibration data for NN. Thus, to classify an observation the nearest neighbor is obtained from the immediate preceding observations within one hour interval. Both methods are compared to a 95% confidence level for statistical significance, computed using permutation tests, a.k.a. randomization tests [13], [14]. The confidence level is useful to assess the expected performance of a random classifier in a given data-set, in order to test whether the performance of the present classifier is beyond what it is expected just by chance. Thus, in Figure 1(b), the expected accuracy of a random classifier is high (between 50% and 95%) due to the low percentage of P2P flows in the data in comparison with non-P2P flows. The random performance also changes over time due to changes in the percentage of P2P traffic. The good performance of NN is evidenced in the figures since both approaches are far above the confidence level. Also, the sliding window approach outperforms the static window in the first hour.

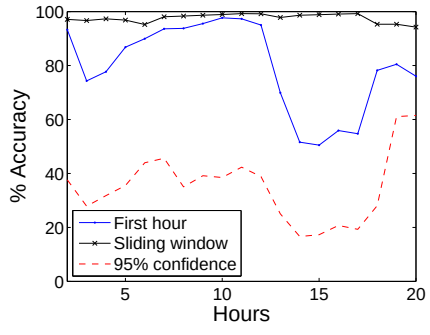
Another interesting question is how the similarity between flows is affected by the coincidence of IP addresses. An experiment to check this is shown in Figure 2. The 20th hour interval of traffic from a specific IP (the most common one) was classified using the NN technique from two different data-sets obtained from the previous 19 hours: traffic from the same IP and traffic from the rest. For a fair comparison, both data-sets had the same number of flows and non statistical differences on time stamp. According to the figure, most of the correct traffic classification and P2P traffic identification is obtained for traffic with the same IP.

V. STRATEGIES FOR FLOWS PAIRING

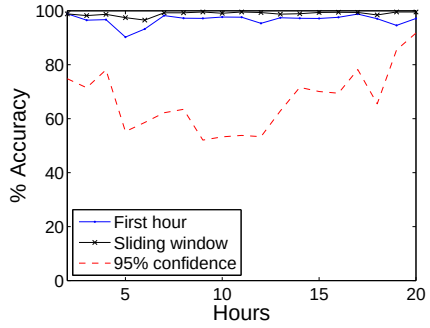
The results in the previous sections show that the good performance of NN is almost restricted to traffic with the same IP. This represents a severe limitation for the general application of NN to on-line traffic classification, since it cannot be applied to traffic coming from new IPs not previously considered. Furthermore, the performance of NN is expected to degrade with the time separation between calibration flows and test flows. Finally, taking into account that calibration flows need an additional classification mechanism to perform NN, for instance payload-based classification, the direct application of NN in traffic classification is not recommended.

Nevertheless, flows identification methods based on pairing can take advantage of this good performance of NN. From the previous results, a convenient approach for

 ISBN: 978-1-61208-173-1



(a)



(b)

Figure 1. Percentage of accuracy for the calibration data-set in (a) traffic classification and (b) P2P traffic identification. The confidence level is computed using randomization tests.

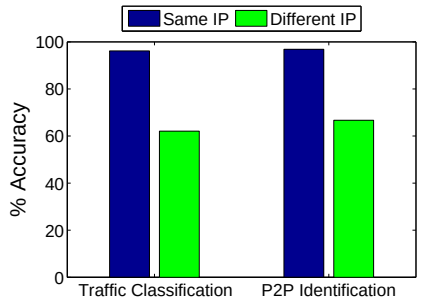


Figure 2. Percentage of accuracy in traffic classification and P2P traffic identification for the 20th hour traffic from a given IP. The performance of NN using past traffic from the same IP is compared to that of NN using past traffic from different IPs.

flows pairing is to use a time sliding window, where only those flows which share at least one IP with the current flow are considered as potential candidates for pairing. This approach has been combined with payload-based classification methods by the authors in some preliminary experiments, yielding less than 5% of payloads inspection to identify correctly close to 100% of flows. This low payload inspection level and the fact that only a time window of traffic data is stored for classification, makes this approach specially suited for on-line traffic classification in network monitoring.

A main decision within this approach is the similarity or closeness functional considered in NN for flows pairing. Here, two types of functional are compared: those based on traditional distances and a parametric functional, referred to as similarity rule. The similarity rule has been designed from first principles by the authors, taking into account the general behavior of network protocols.

A. Distance-based approaches

Two distances have been considered: the normalized Euclidean distance in Eq. (1) and the Mahalanobis distance [3]:

$$d_M(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^t \cdot \mathbf{\Sigma}^{-1} \cdot (\mathbf{x}-\mathbf{y})} \qquad (2)$$

where $\mathbf{\Sigma}$ stands for the covariance matrix. The difference between normalized Euclidean and Mahalanobis distances is that in the latter the weight of the eigenvalues of the covariance matrix in the resulting distance are normalized. This may be convenient when eigenvectors of low variance (low eigenvalue associated) contain relevant information for classification.

B. Similarity rules

The most similar flow to a given one can be found as the one which maximizes a similarity functional. The proposed parametric definition of the similarity functional for a pair of flows is the following:

$$F = |N_{IP} - 1| + \frac{1}{d_{p1} + k_1} + \frac{1}{d_{p2} + k_1} + \frac{1}{d_t + k_2} \qquad (3)$$

where N_{IP} is the number of coincident IPs between the two flows, which is at least 1 (Recall that at least one coincident IP is assumed for flows pairing), d_{p1} and d_{p2} are the 1-norm distances between ports (ordered according to the coincident IP), measured in tens of ports, d_t is the 1-norm distance between time stamps at the beginning of the flow (first packet), measured in seconds, and k_1 and k_2 are the functional parameters.

The definition of the functional in Eq. (3) answers to the behavior of typical network protocols. Thus, servers typically use one or a reduced number of ports to accept service requests. Also, the Operating Systems in the clients typically use consecutive dynamic port numbers for the consecutive connections established. For example, this would be the behavior of a web client when connecting to a number of web pages. In particular, if these pages are hosted in the same server, the flows share the same two IPs and close ports. Finally, related flows should be close in time. Eq. (3) has been designed so that close ports and time stamps have a significant impact on the functional but it is not so much penalized by large distances. For this, 1-norm distances

 ISBN: 978-1-61208-173-1

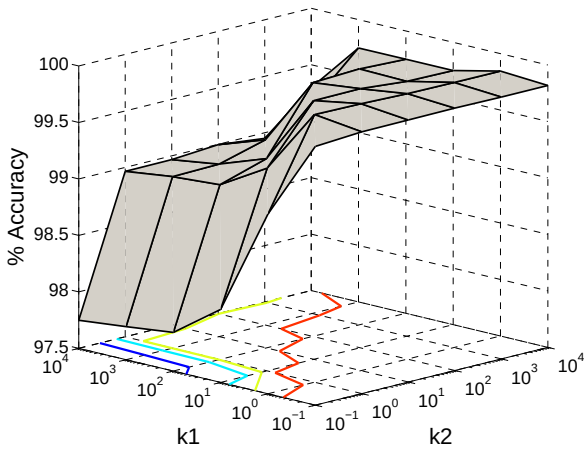


Figure 3. Parameters fitting for the similarity functional using the calibration data. Parameters k_1 and k_2 take values between 0.1 and 10.000 and are presented in a logarithmic scale

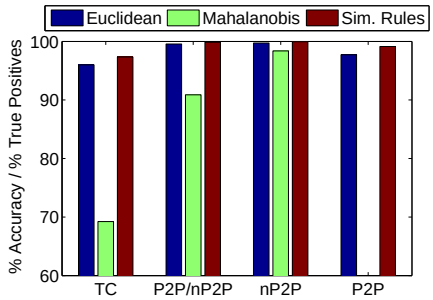
are considered instead of 2-norm distances, and they are included in inverse form in the functional.

The definition of the functional is also convenient from the practical point of view. The five variables (2 IPs, 2 ports and beginning time stamp) considered in the functional are obtained from the first packet in a flow. Therefore, one single packet is enough for flows pairing. Unlikely, distance-based approaches with the feature vector in Table II can only be applied once the flows have finished.

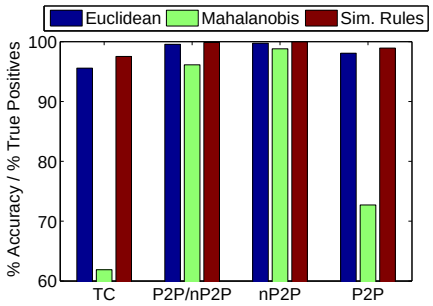
The calibration data will be used to fit the parameters of the similarity functional. Figure 3 shows the result of the calibration for k_1 and k_2 values between 0.1 and 10,000, in logarithmic scale. According to the results, the parameters are set to $k_1 = 1$ and $k_2 = 1$. It should be noted that the results are quite stable for a large interval of the parameters. In particular, the time closeness (k_2) does not seem to be relevant or even positive for certain values of k_1.

VI. Comparison

This section is devoted to compare the performance of distance-based approaches and similarity rules for flows pairing. The accuracy of each approach is defined as the percentage of flow pairs belonging to the same class. Although this work is focused on P2P classification, the pairing strategy can be used for traffic classification in general. This accuracy for traffic classification and P2P traffic identification for the calibration and test data-sets is presented in Figure 4. Notice that all the calibration decisions, such as the normalization in Euclidean distance, the covariance in the Mahalanobis distance, and the values of k_1 and k_2 parameters in the similarity rules, are set from the calibration data and then applied to the test data.



(a) Calibration data



(b) Test data

Figure 4. Comparison of strategies for flows pairing in terms of the coincidence of classes within a pair. Percentage of accuracy in traffic classification (TC) and P2P traffic identification (P2P/nP2P) and percentage of true positives in no P2P (nP2P) and P2P (P2P) traffic. The percentage of true positives of P2P traffic in the calibration data is 13%.

Figure 4 shows that the similarity rules outperform the other two approaches, being the Mahalanobis distance the worst choice. Figure 5 shows the first 30 eigenvalues of the covariance matrix in the normalized calibration data. The first four eigenvalues contain more than the 50% of the variability within the data, which evidences the collinearity of the variables considered in the feature vectors (Table II). The Mahalanobis distance normalizes the weights of the eigenvectors in the distance. This is negatively affecting the performance, showing that the eigenvectors of highest eigenvalue associated contain the relevant similarity information for classification. This is also convenient from the practical point of view, since it means that the useful similarity information is manifesting in a high number of variables. This result is coherent with those in [11]. In particular, the similarity information useful for classification is manifesting in the five variables considered in the similarity rules, which yield the best performance. This is especially convenient considering that a flow can be paired from the first packet using the similarity rules.

Finally, a comparison of the mean time between pairs has been performed for Euclidean-based pairs and similarity rules pairs. A t-test showed that this mean time is lower for

 ISBN: 978-1-61208-173-1

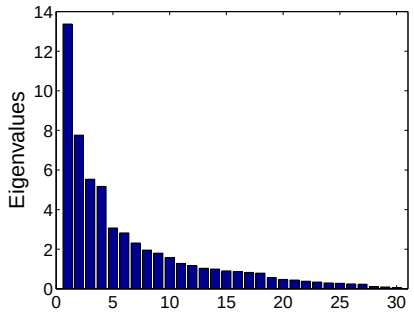


Figure 5. First 10 eigenvalues of the covariance matrix in the calibration data.

the similarity rules pairs (p-value $< 1^{-12}$) for both calibration and test data-sets. Pairs of flows with less difference in time are expected to be more reliable.

VII. Conclusion

This paper is devoted to introduce and compare different strategies for traffic flows pairing based on similarity measures. This strategy is used for fast P2P traffic classification in network monitoring, although it can be applied to traffic classification in general.

According to the results presented, flows pairing can be effectively performed using only five parameters for each flow: the IPs and port numbers and the beginning time stamp. These five parameters are combined in what has been named similarity rule. The pairing based on similarity rules outperforms the application of other traditional distances, such as the Euclidean distance, in several ways:

- The parameters in the similarity rule are available from the first packet in a flow, so that a flow can be paired only with the information in the first packet. Distance-based pairing needs the completeness of the flows.
- Similarity rules are faster to compute than distance-based pairing, since only 5 parameters are used. Also, they require less storage space.
- Classification based on similarity rules outperforms classification based on traditional distances.
- Similarity rules provide closer flow pairs in time than distance-based pairing.

Acknowledgment

Research in this paper is partially supported by the Spanish Ministry of Science and Technology through grant TEC2008-06663-C03-02.

References

[1] A. Callado, C. Kamienski, G. Szabo, B.P. Gero, and J. Kelner, "A Survey on Internet Traffic Identification," *IEEE Communications Surveys & Tutorials*, vol. 11, n. 3, pp. 37-52, 2009.

[2] S. Sen and J. Wang, "Analyzing Peer-to-Peer TrafficAcross Large Networks," *IEEE/ACM Transactions on Networking*, vol. 12, n. 2, pp. 219-232, 2004.

[3] A. Madhukar and C. Williamson, "A Longitudinal Study of P2P Traffic Classification", *Proc. of Int. Symposium on Modeling, Analysis and Simulation*, pp. 179-188, 2006.

[4] R. Keralapura, A. Nucci, and C. Chuah, "A Novel Self-Learning Architecture for P2P Traffic Classification in High Speed Networks," *Computer Networks*, vol. 54, pp. 1055-1068, 2010.

[5] L. Xuan-min, P. Jiang, and Z. Ya-jian, *"A New P2P Traffic Identification Model Based on Node Status"*, In Int. Conference on Mangement and Service Science, pp. 1-4, 2010.

[6] X. Li and Y. Liu, "A P2P Network Traffic Identification Model Based on Heuristic Rules,". *Int. Conference on Computer Application and System Modeling*, vol. 5, pp. 177-179, 2010.

[7] W. JinSong, Z. Yan, W. Qing, and W. Gong, "Connection Pattern-based P2P Application Identification Characteristic," *Proc. of Int. Conference on Network and Parallel Computing Workshops*, pp. 437-441, 2007.

[8] M. Soysal and E.G. Schmidt, "Machine Learning Algorithms for Accurate Flow-Based Network Traffic Classification: Evaluation and Comparison," *Performance Evaluation*, vol. 67, n. 6, pp. 451-467, 2010.

[9] L. Jun, Z. Shunyi, L. Yanqing, and Z. Zailong, "Internet traffic classification using machine learning," *Second International Conference on Communications and Networking in China (CHINACOM'07)*, pp 239-243, 2007.

[10] Y. Lim, H. Kim, J. Jeong, C. Kim, T.T. Kwon, and Y. Choi, "Internet traffic classification demystified: on the sources of the discriminative power," *Proceedings of the 6th International Conference On Emerging Networking Experiments And Technologies (CoNEXT'10)*, 2010.

[11] F.J. Salcedo-Campos, J.E. Díaz-Verdejo, and P. García-Teodoro, "Multiple Vector Classification for P2P Traffic Identification", In Proc. of Int. Conference on Data Communications and Networking (DCNET), 2011.

[12] OpenDPI, 2011. Available at http://www.opendpi.org

[13] F. Lindgren, B. Hansen, W. Karcher, M. S. ostr om, and L. Eriksson, "Model validation by permutation tests: Applications to variable selection," *Journal of Chemometrics*, vol. 10, pp. 521–532, 1996.

[14] S. Wiklund, D. Nilsson, L. Eriksson, M. S. ostr om, S. Wold, and K. Faber, "A randomization test for pls component selection," *Journal of Chemometrics*, vol. 21, pp. 427–439, 2007.

 ISBN: 978-1-61208-173-1

An Empirical Study of MPI over PC Clusters

Fazal Noor, Majed Alhaisoni*, Antonio Liotta+*
*Computer Science and Software Engineering Department
University of Hail, Saudi Arabia
+Department of Electrical Engineering and
Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
The Netherlands
f.noor@uoh.edu.sa, majed.alhaisoni@gmail.com, a.liotta@tue.nl

***Abstract* — Message Passing Interface (MPI) is an important mechanism in P2P. Herein we assess how different types of MPI collective communication functions perform on a Gigabit Ethernet Homogeneous Beowulf PC cluster. In this way we provide an insight on the factors that affect P2P applications over an enterprise context such as the emerging Cloud-based services. By contrast to the literature, which includes mostly theoretical studies, we carry out an empirical study. We show that the bahaviour of scatter and gather are most unpredictable in comparison with other collective functions.**

Keywords - MPI benchmark, Homogeneous PC Cluster, Ethernet, Collective Communications, Latency, Bandwidth, All-to-all, Gather, Scatter, Broadcast.

I. INTRODUCTION

In recent years Peer-to-Peer (P2P) networks have become an active area of research [1]-[4]. Traditional networks use the client/server paradigm where dedicated servers offer clients services. P2P networks are characterized by all peers having the capability of both being a client and a server. P2P networks can support many applications such as sharing of resources, e.g. communication services, file sharing, query search, distributed computing, etc.

In P2P topology, MPI (Message Passing Interface) is considered as a common communication protocol for various P2P systems. Therefore, it is considered as a good mechanism with its goals are to have high performance, scalability, and portability. Having low delay with reasonable throughput is important for computing clusters, due to a lack of shared memory implies large amounts of network data transfer. However, portablility is very important for MPI. The scalability of MPI is mainly due to MPI being the real standard in distributed computing.

In this paper, a simulation of P2P is done using MPI collective communications routines on a PC cluster to measure and evaluate the performance of P2P system. A homogeneous PC cluster is defined as one having identical hardware (including network hardware such as switches) and operating system on all the machines in the network. It is considered heterogeneous if PC hardware and/or software is different from each other in a cluster. One of the reasons to study homogeneous PC cluster is to gain insight into the behaviour of collective communication models used as models usually are made under the assumption of homogeneity. The objective of this paper is twofold, first to study how a variety of MPI communication models perform over PC clusters. Second, we define and measure in practical settings the execution time of Ethernet networks. We pinpoint the overheads and how these affect link efficiency.

Among all technologies, including Infiniband [5], Quadrics [6], and Myrinet [7], we have decided to focus on Ethernet which is readily available for experimentation.

The paper is organized as follows, in Section II some related work is presented. In Section III experimental methods are presented and in Section IV the results are presented. Section V contains discussion of the results presented. Finally in Section VI conclusion and future work is presented.

II. RELATED WORK

There has been a lot of work on MPI communications performance of PC cluster. Most of work is performed on a heterogeneous Beowulf PC clusters. In our work we focus on studying the performance of MPI collective communications on homogeneous Beowulf PC cluster consisting of 20 identical machines. In [9] the authors have used MPIBench a software for benchmarking the performance of MPI functions using a highly accurate, globally synchronized clock. In [10] the authors have developed a MagPIe library which optimizes MPI's collective communication and have used a LogP model for short messages and LogGP model for long messages (Table 1). In [11] a nice comparison is made among the common

 ISBN: 978-1-61208-173-1

parallel communicaton models appearing in the vast literature, such as Hockney, LogP, LogGP, and PlogP. All these models appearing in the literature make an assumption of a homogeneous environment and are listed in Table 1 [11].

Table 1: Common Communication Models

Model	***Time***	***Parameters***
Hockney	$T = l + m/b$	l : is latency of network b: is bandwidth
Log P	$T= o + (k/w -1)*max (g,o) + L + o$	o: overhead time to transmit or receive g: gap min time interval between message p: number of processors l: upper bound on latency k: number of bytes in a message w: size of the network package in bytes
Log GP	$T= o + (k-1)G + L + o$	G: gap per byte for long messages
P Log P	Same as Log P but with each parameter being dependent on message size.	Same as Log P but dependant on message length m L: end to end latency

However analytical models can not truly replace actual performance measurements. In our work we present empirical results of the MPI collective communication routines. Such functions provide insight to communicating on both wireless and wired Peer-to-Peer systems.

III. EXPERIMENTAL METHODS

A. Performance Measurement

The factors which affect performance are many and may be listed as:

1. Hardware related: CPU clock speed and number of CPUs on motherboard, memory, and network adapters.
2. Network related: Type of hardware, cable, fast-ethernet, switches, and routers. Protocols used TCP/IP, UDP/IP and others.
3. Software related: Operating system type, user buffering, kernel buffering of data, types of MPI routines used for example collective communications. MPI eager and rendezvous protocols, efficiency of algorithms, and polling or interrupts.

From the above we can define execution time as a function of both hardware and software. Execution Time will be a function of topology, number of nodes, message size, switch, router, network adapter, type of algorithm, link type, overhead computer and operating systems, physical medium, MPI related, TCP/IP related, and Ethernet related. This is indeed a very complex function in which latencies of both hardware and software have to be considered. Hard disk, RAM, and cache access times with network interface card, PCI, and PCI Express transportation times need to be considered in calculation of execution time. Also the latencies of network devices should be considered such as the switches which are in the 2 to 20 µsec range [8]. In routers the processing delay due to software processes would be considerably higher. DSL or Cable internet connections have less than 100 milliseconds (ms) delays but less than 25 ms are desired. Satellite Internet connections have an average of 500 ms or higher latency. The peak theoretical bandwidth of a network connection is fixed by the technology used but the actual throughput (bandwidth) varies with time and is affected by high latencies. Excessive latencies on the network causes bottlenecks that hinder flow of data therefore decreasing effective bandwidth.

The total time of sending a message from one peer to another peer computer can be represented in terms of execution times and communication time as,

$$Time = t_{execA} + t_{comm} + t_{execB} \ , \qquad (1)$$

where t_{exec} time can be defined as,

$$t_{exec} = K \cdot C / f \ . \qquad (2)$$

where K is instructions per program, C is clock cycles per instruction and f is CPU frequency. The time involves the message's journey from the transmitting computer's memory, user space to kernel space to the network interface, through the physical medium, to the switch, and then to the receiver computer's network interface, and up to the application.

In Peer-to-Peer applications collective operations are rampant. Broadcast, scatter, and gather routines are common and their communication time depends on the size of each message, number of messages, interconnection structure, and network contention. The communication time can be written as

$$t_{comm} = t_s + m \cdot t_t + t_c \ . \qquad (3)$$

In the above equation t_s is the message latency assumed constant and includes the overhead time at the source and destination; t_t is the transmission time computed as $1/B$ where B is link bandwidth given in Q bits/sec; and m represents message to send. The contention time t_c is burst dependent and usually removed for simplicity. The time complexity is $O(m)$ for m data items. Usually one sends messages from one computer to multiple destinations. The 1-to-N fan-out broadcast is when the same message is sent to N destinations sequentially, then the communication time is

 ISBN: 978-1-61208-173-1

$$t_{comm} = N(t_s + m \cdot t_t), \quad (4)$$

and for the *scatter* and *gather* communication models is,

$$t_{comm} = N(t_s + m \cdot t_t / n). \quad (5)$$

In scatter a unique message is sent from source to every other destination and in gather a unique message is received from every other nodes.

The time complexity is $O(Nm)$ for one source connecting to N destinations. In (5), n is the total number of nodes and $N = n$-1. For a tree type structure the time complexity of 1-to-N fan-out will depend on number of nodes at each level and the number of levels. The disadvantage of a binary tree implementation is if one node fails then all the nodes below it will not receive the message.

For an Ethernet LAN network, communication time can be defined as:

$$t_{comm} = t_m + m_E \cdot t_t + t_q + t_p + t_E \quad (6)$$

where all terms depend on message size and message time at Media Access Control (MAC); t_m is dependant on message size and would be same for a homogeneous network and would be some factor multiplied by message size, $\alpha \cdot m_{size}$; m_E represents the number of bits in an Ethernet packet; t_q is the queuing delay; t_p is the propagation delay defined as d/c, where d is the length of the link and c is speed of light in medium having a value less than $3x10^8$ m/sec (e.g. Copper wire .77c); t_E is the Ethernet interframe gap which is 12 bytes (96 bits).
The traffic intensity can be defined as

$$T_i = (m_E \cdot p) / B \quad (7)$$

where p is the average packet arrival rate. Therefore T_i approaching close to 0 will indicate small delay; T_i approaching to 1 is an indication of large delay.

The communication overhead can affect transmission efficiency. The transmitted Ethernet packet has a payload which is TCP/IP encapsulated in addition to the application header. TCP header consists of 20 bytes and the IP header consists of at least 20 bytes. The transmitted Ethernet frame has a preamble of 8 bytes and an interframe gap of 12 bytes. The number of Ethernet packets per second on the link will be,

EthernetPacketsPerSecond = B / [8 *(*FrameSize* + *Interframe gap* 12 bytes + *Preamble* 8 bytes)] (8)

The Ethernet protocol efficiency is defined as,

Efficiency = Payload size/ Frame size (9)

and the throughput is

Throughput = Efficiency x B (10)

Therefore for every Ethernet packet on the link, a 96 bit interframe gap and 64 bits of preamble would be overhead. If the link has the capacity of 1 Gbps then for a minimum Ethernet frame size of 64 bytes transmitted, the link will consist of 7.62 x 10^8 bits/sec due to Ethernet frame and an overhead of 2.38 x 10^8 bits/sec due to interframe gap and preamble combined. For a Gigabit Ethernet the minimum frame size would be 512 bytes when operating in half-duplex mode. The Ethernet protocol efficiency is low for small packets (e.g. 54.76% for 64 bytes) and high (e.g. 97.53% for 1518 bytes) for large packets and hence the throughput is low for small packets and high for large packets.

Latency can have detrimental affects lasting few seconds or can be persistent depending on source of delays. Both bandwidth and latency are two main entities to measure network performance. Since software related latencies are hard to measure and define, one resorts to empirical methods as done in the next section.

B. Beowulf PC cluster Specifications

Our testbed consists of a PC cluster including 20 Lenovo machines with the following specifications: Intel Core™ 2 Duo CPU, E4400 2.00GHz, 1.00 GB of RAM. Network Card: Broadcom Netlink, Gigabit Ethernet, Driver date 8/28/2006 version 9.81.0.0. The PC are connected to a Gigabit D-Link Ethernet switch. Each machine has RedHat Enterprise AS Linux operating system installed, and use LAM 7.0.6/MPI 2.

C. MPI Benchmarks

The Message Passing Interface (MPI) is a standard interface that is broadly used with distributed computing applications [12]-[15]. The following MPI-based benchmarks are used to test the communication performance of the nodes:

a) *All-to-all*: every node sends a message to every other node.
b) *Broadcast*: one node sends one message to every other node.
c) *Gather*: all nodes send a different message to a single node.
d) *Scatter*: a single node sends a different message to every other node.
e) *Point-to-Point*: a single message is sent/received between 2 specific nodes.

The implementation details of the above collective communications are usually unknown to the programmer.

 ISBN: 978-1-61208-173-1

The MP1-1 standard specified the "blocking collective communications" only while the MPI-2 standard defines "non-blocking" routines which perform better with some applications. MPI related software performance will depend on the type of message passing protocols, *eager* (asynchronous communication) or *rendezvous* (synchronous), type of message buffering (user and system), sender-receiver synchronization (e.g. polling or interrupt), and efficiency of the algorithms used to implement the collective communication routines.

There are many benchmarking software available on the internet such as MPIBench or mpptest. However, most benchmarks available for collective communication basically use the following procedure to measure the execution time.

1. All processes arrive at Barrier
2. Start time
3. MPI_collective_fn
4. All processes arrive at Barrier
5. End time
6. PTime = End_t ime – Start_time

where MPI_collective_fn is one of the MPI functions all-to-all, gather, scatter, broadcast, and Point-to-Point.

D. Analysis of Execution Time (PTime)

In the benchmark procedure above, PTime consists of time to execute the MPI_collective_fn function and twice the time of Barrier.

First, the time spent at the transmitting computer would involve sending data by the kernel of system to the network interface card (NIC) and the time NIC takes to pack bytes in an Ethernet frame to send the frame on the physical wire. Depending on type of NIC architecture, a typical Ethernet NIC would have specifications as given in Table 2.

Table 2. Gigabit Ethernet Network Interface Specifications.

Speed	Interface	Data Width	Clock	Time for 1 byte transfer
1 Gbps	GMII	8 bits	125 MHz	8 ns
10 Gbps	XGMII	32 bits	156.25 MHz	.8 ns

To transmit an Ethernet frame the time to transmit from Medium Access Control (MAC) to Physical (PHY) for a 1 Gbps link would be

$$T_t = 8 \text{ ns} * Ethernet\_Frame\_size \quad (10)$$

One thing about Gigabit Ethernet is its clock rate at 125 MHz but more data is transmitted per time. The transfer rate is higher since 125 MHz x 2 bits per signal (i.e. per wire pair in Cat 5E cable) x 4 signals per time = 1 Gbps. Therefore, on the motherboard if PCI Express is available then with a maximum transfer rate of up to 250 MB/s then full speed of Gigabit Ethernet is achievable.

Next, the switch receives the Ethernet frame, and processes the frame with a typical delay of 2 to 20 µsecs [8]. It is then sent out to the destination computer where again the NIC on receiving computer processes it (PHY taking anywhere from 200 to 300 nanoseconds depending on technology) and sends it to the MAC layer, then onto the TCP/IP layers up to the application. The process of transmittance, reception, and acknowledgement is repeated according to TCP/IP, Ethernet framing, and depending upon the application's instructions, i.e. MPI collective_fn function and time of Barrier. The Barrier is used to explicitly control the flow of execution. There are at least 3 types of Barrier implementations [9], namely,

a) Counter implementation (linear barrier)
b) Tree implementation
c) Butterfly barrier

The time complexity of barrier with counter implementation is $O(n)$. For both the tree and the butterfly implementations the time complexity is $O(\log n)$, where n is the number of nodes. From the above, PTime depends on how MPI collective_fn and Barrier are implemented in LAM 7.0.6/MPI 2, as summarized in Table 3 (for large number of nodes).

Table 3. Time Complexity

Linear Model	PTime	
	Barrier Implementation	
	a) Counter	*b) Tree and c) Butterfly*
MPI_alltoall	O(nNm) + O(n)	O(nNm) + O(log n)
MPI_Bcast	O(Nm) + O(n)	O(Nm) + O(log n)
MPI_Gather	O(Nm) + O(n)	O(Nm) + O(log n)
MPI_Scatter	O(Nm) + O(n)	O(Nm) + O(log n)

IV. RESULTS

The MPI benchmarks are run on a PC cluster by first fixing the number of nodes in a communication group to 2 and varying the size of messages from x Kbytes to y Mbytes (2^n where n = 0,1,2,3,4…,). Then the number of nodes in communication group is iteratively incremented up to 20 nodes.

The figures show Minimum Round Trip (MRT) time measured in granularity of µsecs for messages of sizes ranging from 256 KB to 2 MB.

In Figure 1 all-to-all minimum round trip time is plotted versus number of nodes in a PC cluster. Fig. 1 show MRT is almost constant (with little variation) for a communication group consisting of anywhere from 2 to 9 nodes in comparison with 10 to 20 nodes which shows MRT linearly increasing.

In Figure 2, broadcast MRT time is plotted versus number of nodes. From the figure we observe MRT increases with increase in the number of nodes again with a steeper slope for large message sizes within each group. Note, from the figure it shows somewhat a step wise increment in MRT values.

 ISBN: 978-1-61208-173-1

In Figures 3 and 4, gather and scatter MRT time is plotted versus number of nodes, respectively. Both gather and scatter have a similar shape for MRT. The shape is more pronounced for large message sizes within each group. For gather and scatter communications it is seen from the figures MRT for small number of nodes e.g. 2, 3 and for large number of nodes, e.g. 16-20 takes on values which are much larger than the MRT of the number of nodes in between. MRT takes on a minimum for 8, 9 nodes and slightly higher for 10 nodes. It is also interesting to note that skewed shape flattens out as the message size is reduced. The figures of scatter and gather show unexpected behavior in MRT for nodes up to 9. The reason for this is presented under the discussion section.

In Figure 5, Point-to-Point benchmark is plotted with logarithmic scale for MRT and messages sizes ranging from 4 bytes to 8 Mbytes.

Comparing all-to-all communication with the others it is seen all-to-all has the highest MRT as expected.

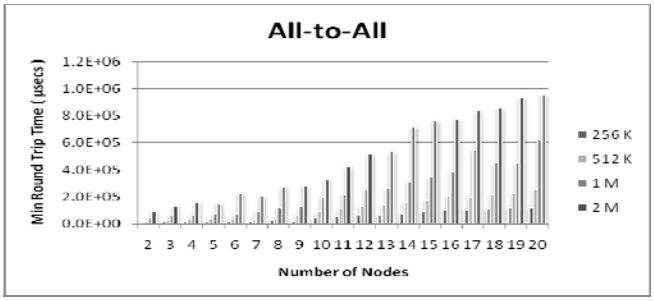


Figure 1. All-to-all benchmark showing MRT for message sizes of 256K B, 512KB, 1MB, 2MBand for different size of PC cluster.

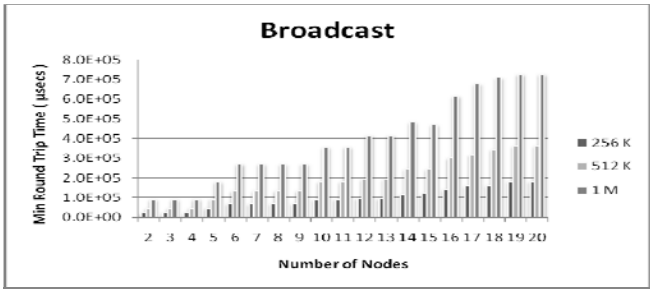


Figure 2. Broadcast benchmark showing MRT for message sizes of 256KB, 512KB, 1MB and for different size of PC cluster.

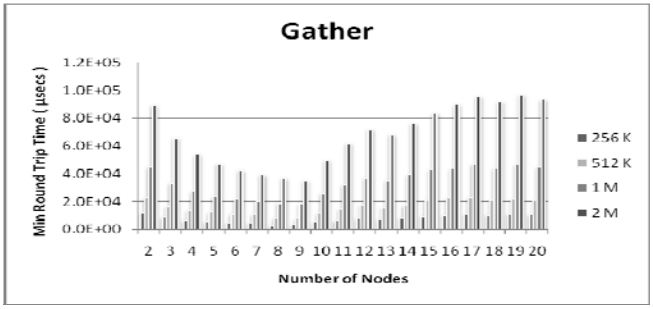


Figure 3. Gather benchmark showing MRT for message sizes of 256KB, 512KB, 1MB, 2MB and for different size of PC cluster.

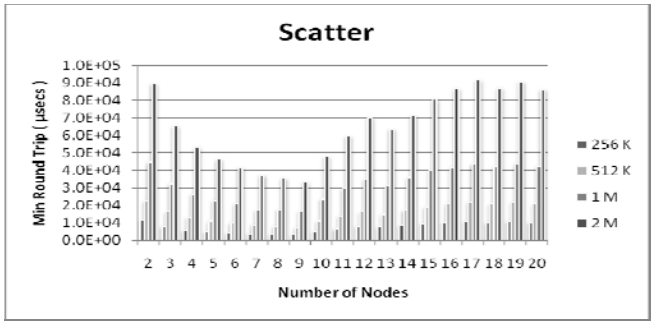


Figure 4. Scatter benchmark showing MRT for message sizes of 256KB, 512KB, 1MB, 2MB and for different size of PC cluster

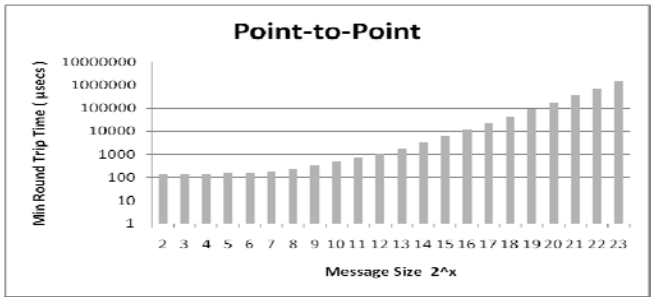


Figure 5. Point-to point benchmark showing MRT log scale versus message sizes of 4 bytes to 8 Mbytes.

V. DISCUSSION

In this section we discuss the results of all-to-all, broadcast, gather, scatter, and Point-to-Point functions used in the benchmarks. First note, the message size being transmitted is fixed in the case of broadcast and Point-to-Point. In the second case of all-to-all, gather, and scatter, the message size is divided equally within the communication group and depends on the number of participating nodes. Let m denote the message size and n denote the number of nodes,

$$s_i = m / n_i \quad \text{for } i = 2,3,...,20 \tag{11}$$

where s_i is the actual message size being transmitted or received by each communicating node. Therefore, $s_1 > s_2 > s_3 \cdots > s_{20}$ since $n_i < n_{i+1}$. As the number of nodes are increased the message size being sent or received goes down. From (5) one knows that latency depends on message size (and of course is a function of time): as the message size s_{i+1} is less than s_i therefore the per message latency of s_i is larger than s_{i+1}. However this is not always the case as seen in the scatter and gather routines. When a host application transmits to its destination, non-blocking sends are posted by MPI,

 ISBN: 978-1-61208-173-1

reducing latency for certain nodes (e.g. 9 in the scatter figure) and the TCP protocol is used for reliability. It must wait for a period of time to receive an acknowledgment. If the reply does not arrive within the expected period the data is retransmitted. On Ethernet LAN the wait time is not more than a few µsecs. Thus, the overhead time has a major affect on latency.

A number of factors together are involved in having a major affect on MRT as shown in the scatter and gather plots. First, varying the size of the message has affect. Second, the implementation of LAM-MPI routines are not known to the programmer, which model is being used whether linear or tree type. Third, which protocol MPI is using either eager or rendezvous protocols. The LAM-MPI constructs the message and sends it through the network to the destination computer which must accept and act upon the message contents. LAM-MPI uses either the eager protocol or the rendezvous protocol depending on message size. With the eager protocol, as soon as a message is posted both the envelope and data are sent to the destination. If on the destination the receive operation is not posted then buffering has to be done; this buffering involves an additional data duplication. In the rendezvous protocol, when a message is posted the envelope is sent to the destination and buffered. As soon as a receive is posted the destination sends an acknowledgement to the sender which then only will be the data send by the sender In this case buffering of the data is avoided and used for large messages.

Fourth, TCP/IP protocol is being used by LAM-MPI protocols on top of TCP/IP protocols which cause higher latencies in communicating a message from sender to destination. The actual application bytes packed in an Ethernet frame are much less due to MPI application, TCP/IP, and Ethernet headers. In some cases, as depicted in the figures of scatter and gather, when the message size decreases among the communication nodes it can happen that MRT decrease down to a certain value and, as the message size is further decreased beyond the minimum MRT point, the MRT will start to increase again. The explanation of such a behavior is due to the factors mentioned above (i.e. the implementation of the scatter and gather algorithms in MPI, eager and rendezvous protocol switching depending on message size, plus TCP overhead and the increase of overhead ratio as the Ethernet frame size decreases).

VI. CONCLUSION AND FUTURE WORK

In this paper we have studied the performance of MPI collective communications routines on a gigabit Ethernet LAN. The experiments were performed to represent a Peer-to-Peer scenario in which one has different sizes of nodes in a group and variations in message size. All the benchmark routines presented closely represent typical communications of a Peer-to-Peer system i.e. broadcasting, gathering, scattering, all-to-all, and point-to-point. The results show in the case of gather and scatter that as the message size is decreased among the increasing nodes there is no set predictable pattern MRT takes. We have seen that many factors affect the performance of collective communications in a wireline (Ethernet) environment. Our next target is to extend our study to the area of Peer-to-Peer over wireless networks.

ACKNOWLEDGMENT

The authors would like to thank CSSE Research Center for carrying out experiments on the PC cluster. The authors would also thank the anonymous reviewers for their valuable and insightful comments.

REFERENCES

[1] S. Lin, A. Pan, R. Guo, and Z. Zhang., “Simulating Large-Scale Peer-TO-Peer Systems with WiDS Toolkit”, White Paper, Microsoft, Jan. 2008.

[2] M. Li, W. Lee, and A. Sivasubramaniam, “Efficient Peer-to-Peer Information Sharing over Mobile Ad Hoc Networks”, In MobEA, 2004.

[3] X. M. Huang, C.Y. Chang, and M.S. Chen, “PeerCluster: A Cluster Based Peer-to-Peer Sytem”, IEEE Transactions on Parallel and Distributed Systems”, vol. 17, No. 10, Oct. 2006, pp. 1110-1123.

[4] B. Parviz and K. Miremadi, “Building a Peer to Peer Message Passing Environment by Utilizing Reflection in .NET.”, In Proceedings of PDPTA'2006. pp.1096~1102.

[5] InfiniBand Trade Organization., http://www.infinibandta.org/

[6] http://en.wikipedia.org/wiki/Quadrics, Sept. 3, 2011.

[7] Myricom Inc., http://www.myri.com, Sept 3, 2011.

[8] CISCO Inc., http://www.cisco.com, Sept 3, 2011.

[9] F. A. Vaughan, D. A. Grove, and P. D. Coddington, “Communiation Issues for Two Cluster Computers,” ACSC ’03 Proceedings of the 26th Austrailasian computer science conference, vol 16, 2003.

[10] T. Kielmann and H. E. Bal, “Fast Measurment of LogP Parameters for Message Passing Platforms”, 4th Workshop on Runtime Systems for Parallel Programming (RTSPP), pp. 1176-1183, held in conjunction with IPDPS 2000, Cancun, Mexico, May 1-5, 2000. Lecture Notes in Computer Science, Vol. 1800.

[11] J. P. Grbovic, et al, “Performance Analysis of MPI Collective Operations”, Journal Cluster Computing, Vol 10, Issue 2, June 2007, pp. 127-143.

[12] R. Riesen, “Communicaton Patterns”, Parallel and Distributed Processing Symposium, 25-29 April 2006.

[13] A. Leko, et al, “Practical Experiences with Modern Parallel Performance Analysis Tools : An Evaluation”, Parallel and Distributed Processing, IPDPS 2008 IEEE Symposium 14-18 April 2008, Miami, Fl, pp. 1-8.

[14] B. Wilkinson and M. Allen, Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers, Second Edition, Pearson Prentice Hall, 2005.

[15] F. Noor and S. Misbahuddin, “Using MPI on PC Cluster to Compute Eigenvalues of Hermitian Toeplitz Matrices”, Lecture Notes in Computer Science, 2010, vol 6081, pp 313-323.

 ISBN: 978-1-61208-173-1

Coalitions and Incentives for Content Distribution over a Secure Peer-to-Peer Middleware

Maria-Victoria Belmonte, Manuel Díaz and Ana Reyna
Department of Languages and Computer Science
E.T.S.I. Informática. Bulevar Louis Pasteur, N.35
University of Málaga (UMA), 29071, Málaga, Spain
e-mail: {mavi, mdr, reyna}@lcc.uma.es

***Abstract*— Nowadays, Peer-to-Peer is responsible for more than 60% of Internet traffic. These protocols have proved to save bandwidth and computing resources in content distribution system. But, problems related to user behaviour, such as free riding, still persist, and users must be motivated to share content. In previous work, we have designed and simulated a coalition and incentive theoretical mechanism for content distribution that aims to fight against problems in user behaviour. In this paper, we present a real implementation of it. Since developing a peer-to-peer application from scratch is a laborious and error prone task, we use SMEPP, a middleware that aims to ease the development of secure distributed application, to implement it.**

Keywords-coalitions; incentives; peer-to-peer; middleware; overlay.

I. INTRODUCTION

Nowadays, Peer-to-Peer (P2P) protocols are responsible for more than 60% of Internet traffic, in spite of anti-piracy laws [1]. Many Internet applications are taking advantage of P2P architecture, since P2P paradigm abandons central servers to give way to a network where all nodes play the role of server and client simultaneously. This brings new perspectives to application scalability; where an excess of nodes in the client-server paradigm could lead to saturation or even a system crash, in the P2P paradigm it means greater capacity.

P2P protocols enable content distribution in a cost-effective way, as they do not require a centralised provider to handle all the demands. Instead, a P2P protocol can use its clients' bandwidth for content distribution, saving the bandwidth and computing resources of the system. However, the performance and availability of these systems relies on the voluntary participation of their users, which is highly variable and unpredictable. Empirical studies have shown that a large fraction of the participants share little or no files. For instance, in [1], the authors affirm "*in Gnutella 25% of the users do not share any files, Furthermore, about 75% of the clients share 100 files or less*" (including the 25% that do not share) "*and only 7% of the clients share more than 1000 files. This 7% of users together offer more files than all of the other users combined*". More recently, Handurukande et al. [3] also observed the same behaviour in the eDonkey P2P network and concluded that this is common to most P2P file sharing systems. This phenomenon is known as "free-riding", and is still an open issue on content distribution systems [4]. P2P content distribution systems need mechanisms that motivate peers to share their content.

In [5], we presented a new coalition formation scheme based on game theory concepts which formally prove how coalitions improve P2P systems performance, encouraging participants to contribute resources, receiving in return a better quality of service. Empirical results obtained through simulations illustrated how our approach encourages collaborative behaviour, preventing the free-riding problem and improves the overall performance of the system. Until now, this mechanism has been a theoretical proposal, whose features have been demonstrated only through simulations. In this paper, we present a real distributed implementation of the mechanism.

The development of distributed applications in general, and concretely P2P, is a laborious and error prone task, since many issues must be considered, from network protocols, to security. In order to facilitate the software development of this kind of system, new tools and methodologies capable of abstracting all the underlying complexity should be used. A middleware can simplify and reduce the development time of the design, implementation and configuration of applications, thus allowing developers to focus on the requirements of their applications. In [5], we presented SMEPP (Secure Middleware for Embedded Peer to Peer systems), a new middleware especially, but not exclusively, designed for Embedded Peer to Peer (EP2P) systems. This middleware was designed to overcome the main problems of existing domain specific middleware proposals [6]. The middleware is secure, generic and highly customisable, allowing it to be adapted to different devices, from PDAs and new generation mobile phones to embedded sensor actuator systems, and domains, from critical systems to consumer entertainment or communication [8].

In this paper, we take advantage from SMEPP middleware to implement our coalitions and incentives mechanism for distributed content distribution. On the one hand we prove the suitability of using a P2P middleware, and on the other, we demonstrate that our mechanism can be developed as a real distributed application.

 ISBN: 978-1-61208-173-1

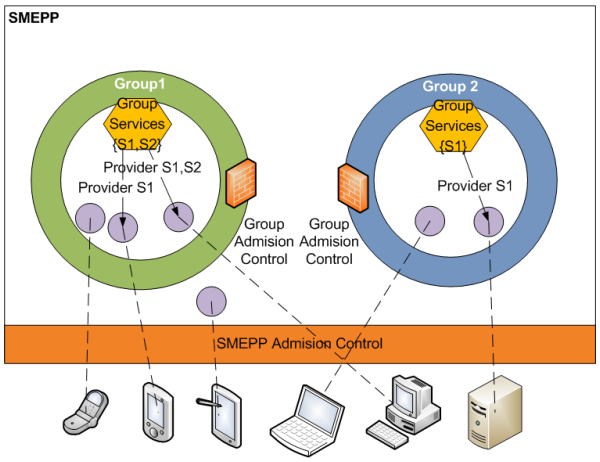


Figure 1. SMEPP abstract model

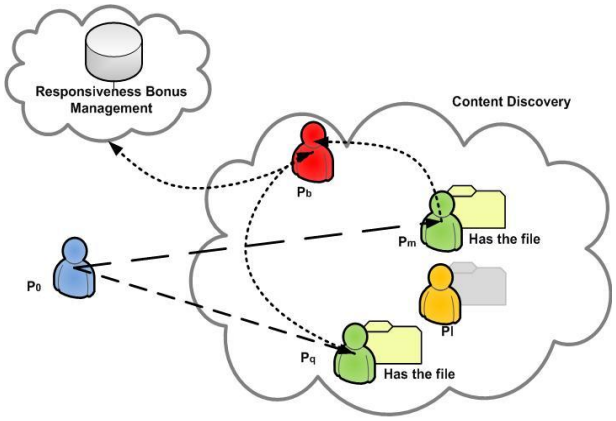


Figure 2. Coalitions and incentives mechanism

The structure of this paper is as follows. The following section presents an overview of SMEPP middleware. In Section III, the main features of our content distribution mechanism are introduced. Section IV focuses on the implementation issues. Finally, conclusion is presented in Section V.

II. SMEPP MIDDLEWARE OVERVIEW

SMEPP middleware is based on three main pillars, its abstract model, its reusable and flexible architecture and its built in security. A detailed description of SMEPP middleware is beyond the scope of this paper, nevertheless we believe it is essential to introduce its main features. In addition, in Section IV, some details, required for the implementation, will be given. More details of SMEPP can be found in [5] .

The abstract model defines the entities involved and how they relate in P2P environments (illustrated in Figure 1). It defines the concepts of peer, group and service. The functionality of the application is offered in the form of services, which can be published or consumed only inside groups of peers. The group definition determines the level of security inside a group. To access a group the peer has to provide the suitable credentials, this is internally managed by the middleware. The service discovery is effectively performed thanks to the underlying structured overlay network that implements CHORD protocol [9]. In addition, the abstract model also defines the API, which is generic and language independent, and defines the functionality exposed by the middleware with a high level of abstraction, this is the way the programmer can interact with the middleware.

The architecture of SMEPP is based on software components. Component-oriented paradigms have proved to be a good approach to designing a middleware. Software components offer several features (reusability, adaptability, etc.) which are particularly suitable for dynamic environments and rapidly changing situations that a middleware has to face. This is especially interesting for our application. A specific component framework has been designed for the implementation of the middleware. The developer has several tools which allow the tuning of the middleware for a specific platform, device or communication protocol.

Security is the most distinctive feature of SMEPP. Since its conception the security aspect was considered, and tackled transversally on the architecture and on the service model definition, this ensures that the middleware is capable of providing a high level of customisable security.

The SMEPP performance results showed that overall resource consumption of the middleware was relatively small, the overall memory consumption peak being 1,83MB and the highest average memory consumption being 670kB. Moreover regarding the usage of CPU, the middleware uses relatively little CPU time (max being 2% of CPU capacity on a 2GHz Intel Core2 Duo). Taking into consideration that the middleware is designed to work on small capability devices, this is a good result. Furthermore, the suitability of SMEPP was demonstrated by the development of two different innovative real-life applications in the domains of Context Aware Mobile Telephony and Environmental Monitoring in Industrial Plants [8]. This was formerly implemented in JXTA [10], where a new version of the application over SMEPP proving the benefits of using middleware was developed. A comparison between JXTA and SMEPP can also be found in [11].

In this paper, we demonstrate how our coalitions and incentives mechanism for content distribution can be easily implemented over SMEPP, taking advantages of its features, such as the built in security and the structured overlay look up mechanism for content discovery.

III. COALITION AND INCENTIVES MECHANISM OVERVIEW

The central idea of our mechanism is sharing the task of downloading a file between a set of peers making up a coalition. On the one hand, the downloader benefits as the total download time is reduced. On the other hand, the burden on the uploader (or provider) peer is also alleviated, since the total task is divided between the members of the coalition. And in addition, providers are rewarded for their participation in the coalition.

More concretely, these rewards aim to encourage participants to contribute resources, receiving in return a better quality of service. In this way, each peer that participates in a coalition is lending "bandwidth" to other coalition peers, in

 ISBN: 978-1-61208-173-1

exchange for profit or *utility*. Each participant or provider receives a reward each time it participates in a coalition, and is penalised each time it downloads. The reward that a provider obtains by performing a task inside a coalition is calculated using the game-theory concept of *core* [12]. The *core* ensures that each coalition participant receives a fair utility in return for the bandwidth that it supplies. In our model, these utilities are used to compute the *Responsiveness bonus* (*Rb*), which represents the overall contribution of the peer to the system. Therefore, this value will determine the quality of service of each peer. The higher *Rb* the better quality of service, this is the key to encouragement.

Each peer can play three different roles: downloader, participant or manager (In Figure 2, P_b is the downloader, P_0 is the manager, and P_q and P_m are the participants). To sum up: the download process starts when a peer decides to download a file. In order to download this file, the downloader has to find file providers in the network (discovery). Once the providers are found, a coalition manager is elected. The manager selection does not imply centralization, because any potential participant can become the manager with equal probability. Next, the manager sends offers to the potential candidates (the rest of the providers). Each provider answers the offer, and once the manager has received all of them (or a timeout is reached), it has to divide up the task between the potential coalition members (those participants who answered). This process, called *Task Assignment*, establishes the coalition itself, and after this, the download itself starts. During the download, the downloader periodically sends acknowledgement information to the manager, who runs a *checking mechanism* to guarantee the quality of service in the coalition, adapting to network traffic and helping to avoid some attacks of malicious peers (such as free-riding). After these checks, the *Rb* of all the members of the coalition is updated using the utilities obtained after the *Coalition Payment Division*.

A. Task Assignment

Given a collection of providers, the task assignment has to determine the task that each provider will be responsible for, this is the input bandwidth that each participant will provide to the coalition. If there are few participants (under a threshold) no selection has to be done, otherwise only some providers will be chosen for the coalition.

To do so, and to determine the input bandwidth of a participant, *the progressive filling algorithm* is used. This algorithm provides the *max-min fairness* [13]. A bandwidth allocation is max-min fair if and only if an increase of the input bandwidth of a peer *x* within its domain of feasible allocation is at the cost of decreasing some other input bandwidth of a peer *y*. So, it gives the peer with the smallest bidding value the largest feasible bandwidth.

B. Checking Mechanism

The checking mechanism makes the system less vulnerable to peer failures, churns and network congestion problems, while it ensures the quality of service of the coalition. The mechanism works as follows, during the download of a file; the downloader sends acknowledgement information to the manager with a predefined frequency. The manager calculates the difference between the bytes sent and the ones which should have been sent (according to the task assigned to each participant). If this difference exceeds a predefined threshold, the coalition is reconfigured in order to provide better quality of service. Moreover, the manager also checks that the downloader *Rb* is high enough to keep downloading. The central idea is that if the coalition is not working as it was expected or the downloader is abusing the system, the coalition is cancelled.

Since the update of *Rb* values are calculated by the manager and are based on the acknowledgement sent by the downloader, the downloader could avoid the penalty if it sends faked acknowledgement. But the checking mechanism performed by the manager will stop the coalition if the acknowledgement is too small, so the downloader will not be penalised, but neither will they receive the file.

C. Coalition Payment Division

The hallmark of our mechanism is that the coalition payment division ensures fairness, thanks to the game theory concept of *core*. This means that peers won't be negatively affected if they have lower capacity. The details of this are explained in the following paragraph.

Let's call coalitional value *V(S)*, to the total utility or profit of a coalition *S*. For every peer in the coalition, $Pi \in S$, we must distribute *V(S)* between the peers, and assign an amount or utility (x_i) to every peer $Pi \in S$. The problem is to distribute *V(S)* in a stable and fair way so the coalition peers have no reason to abandon it.

Firstly, we must calculate *V(S)*. The profit obtained by *S* is calculated as the difference between the time required for the download with just one uploading participant (only P_0, the manager) minus the time it takes with the coalition *S* (all the participants, including the manager). Then the coalitional value is given by the following equation:

$$V(S) = t_0 \frac{\sum_1^n b_i^{in}}{\sum_0^n b_i^{in}} \quad where\ t_0 = \frac{File\ size}{b_0^{in}} \qquad (1)$$

where t_0 is the time that it would take the P_0 to upload the whole file (being P_0 the only uploader or provider), b_0^{in} the upload bandwidth of P_0 and b_i^{in} the upload bandwidth of the remaining participants of *S*.

Secondly, we use the *core* to distribute *V(S)* between the coalition members. A utility distribution belongs to the *core* if there is no other coalition that can improve on utilities of all of its members. The stable utility division (x_i) to every peer $Pi \in S$ is given, then, by the following equation (in detail in [5]), where b_0^{out} is the download bandwidth of P_0 .

 ISBN: 978-1-61208-173-1

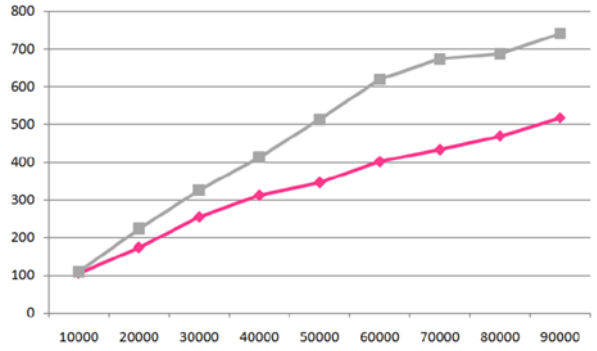


Figure 3. Average download time using coalitions (diamonds) and not using them (squares) (simulation time vs bytes)

$$x_i = \begin{cases} t_0 \dfrac{\left(\sum_1^n b_i^{in}\right)^2}{\left(\sum_0^n b_i^{in}\right)^2} & if\ i = 0 \\ t_0 \dfrac{b_0^{out}\ b_i^{in}}{\left(\sum_0^n b_i^{in}\right)^2} & if\ i \neq 0 \end{cases} \quad (2)$$

D. Responsiveness Bonus Computation

As it has been said, peers with higher utility will get a better quality of service. In our approach the utility accumulated by each peer (Rbi) is proportional to the resources that it supplies, and it is calculated as a heuristic function of x_i. The value of Rb_iwill be reduced when P_i acts as a downloading peer, and incremented when it is a provider or uploading peer. The heuristic uses the x_i values obtained by Pi by means of U_{pi} (Upload points) and D_{pi} (Download points). U_{pi} and D_{pi} accumulate the utility obtained by each coalition formation process in which P_i participates.

Let us call Fsi to the number of files shared (the total size in bytes) by a peer Pi. The Rb_i value of the peer is calculated using the following equation:

$$Rb_i = \begin{cases} 1 & if\ (U_{pi} - D_{pi}) \geq 0 \\ 0 & if\ (U_{pi} - D_{pi}) < 0 \wedge U_{pi} = 0 \wedge F_{si} = 0 \\ 1 & if\ (U_{pi} - D_{pi}) < 0 \wedge U_{pi} = 0 \wedge F_{si} > 0 \\ \dfrac{U_{pi} \cdot \gamma}{D_{pi}} & if\ (U_{pi} - D_{pi}) < 0 \wedge U_{pi} > 0 \end{cases} \quad (3)$$

The Rb_i values are between zero and one. The interpretation of this formula is that if the peer uploads more than downloads, it gets the maximum value, also true when, it is not uploading but sharing. If neither uploading nor sharing; its Rb_i is set to zero. In any other case, it is calculated as the ratio between the upload and the download points (the γ parameter allows us to regulate the relation to increase/decrease the penalty/reward).

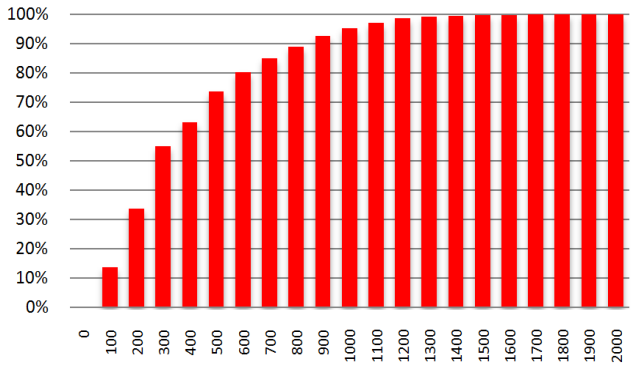


Figure 4. Free tider detection (% detection vs simulation time)

Therefore, we use this value to decrease the download bandwidth $Rb_i \cdot b_b^{out}$ (using it as a multiplier of the download bandwidth of the peer Pb when it wants to download a file). Initially, the Rb_i of the peers is 1, a higher responsiveness bonus (Rb_i closer to 1) will mean that P_i will be able to use most of its bandwidth capacity. Otherwise, a Rb_i closer to 0 will reduce its bandwidth capacity, (in fact, it could even avoid creating the coalition for the download when it is 0). Thus, our incentive mechanism penalises the selfish behaviour of the peers, and provides incentives for collaborative behaviour.

E. Experimental Results

In [5], we presented some simulation results. These experiments confirmed the benefits of using our mechanism. On the one hand download times are improved, and on the other hand, free riders are stopped, this lead to an improvement of the system's effectiveness.

Our own simulator was used to run the experiments. It was configured to simulate a P2P network of 1000 peers during 2000 units of simulated time (steps). All peers had the same bandwidth capabilities. The collection of files shared in the network was defined with different sizes (from 10000KB to 90000 KB), and a random number of copies (between 5 and 500) of these were delivered through the network at the start of the simulation. Each peer had a random number of initially stored files, and the objective of the simulation was that every peer download the files that were not initially stored. Our simulations, considered three types of users (or behaviours): free riders (FR), collaborative (C) and adaptive (A). The first, do not share at all, the second share as much as possible, and the last, only share if they want to download. Depending on the behaviour of each peer (that is randomly assigned in each simulation) it will face its downloads in different ways.

 ISBN: 978-1-61208-173-1

TABLE I. BYTES DOWNLOADED IN SIMULATIONS

	No coalitions		*Coalitions*	
	Pop. 1	***Pop. 2***	***Pop. 1***	***Pop. 2***
FR	157 Gb	123 Gb	24 Gb	20 Gb
C	156 Gb	92 Gb	130 Gb	82 Gb
A		91 Gb		84 Gb
Total	313 Gb	306 Gb	154 Gb	186 Gb

To analyse the impact of the different behaviours on the system the experiments were run with two different populations. The first one without adaptive users: 50% FR, 50% C and 0% A, called Population 1. And the second with adaptive users: 40% FR, 30% C and 30% A, called Population 2. In addition, to analyse the impact of the use of coalitions, simulations were run with and without incentive policies: No Coalitions (NC), where no incentive mechanism was considered and Coalitions (C), which implemented our proposal. After repeating the simulation experiments 100 times we took the average to give the results. Two main metrics were considered: downloaded bytes and average download time.

In Table I the bytes downloaded per populations and per behaviour for both scenarios, with and without coalitions, is shown. In Population 1, when coalitions were used the total amount of bytes downloaded was reduced to 50% with respect to NC, but 84% of this reduction was due to the free riders detection. This showed how the algorithm prevents free riders from abusing, avoiding the overhead of the system resources. In Figure 4, the free rider detection effectiveness of our approach is shown (Population 2). More than 50% were detected at step 300 and the 100% were stopped at step 1500.

In Population 2, when adaptive users were introduced, the benefit of using of coalitions was higher (than in Population 1). The total amount of bytes was reduced by 39% with respect to NC, where 83% was due to the free rider's detection. In addition, comparing coalitions in both populations, the total amount of downloaded bytes were increased by 20% using Population 2, proving that adaptive users benefit the system. Note that in Population 2 there were fewer free riders and collaborative users, therefore, less shared files in the network, this justifies the smaller amount of total bytes downloaded with respect to Population 1.

In addition to the analysis of the downloaded bytes, the average download time offered even better results. In Figure 3, the average download time using and not using coalitions is shown for Population 2. Experiments showed that using coalitions the average download time was smaller. As expected, the benefit of using coalitions is increased as the file size grows. When adaptive users were introduced the download times were improved compared with NC, what demonstrated the effectiveness of our incentive mechanism. More details about the configuration and results of the experiments can be found in [5].

F. Related Works

The incentive mechanisms in P2P networks for content distribution [4] have been classified in different categories. Our approach belongs to reciprocity based mechanisms: peers that contribute more get a better quality of service. Other publications also included in this category are [14][15][16][17][18][19].

From the approaches above, those based on mutual reciprocity, like Bit Torrent [16], Emule [15] or [14], do not fit the asymmetric nature of a collaborative relationship, since the peer's decision to upload to another peer is based on the direct exchange of data/services/credits between two peers that have mutual interests (same content). However, our approach, unlike the ones above, encourages cooperative behaviour by forming coalitions of peers that help each other to download files. So any peer can participate in a coalition increasing its *Rb*, and this will lead to a higher download bandwidth for further downloads from any other peer in the system.

The indirect reciprocity-based approaches, like [17][18][19] or our approach, consider peers' overall contribution to the network, and so they encourage cooperation.

2Fast [17] is also based on creating groups of peers that collaborate in order to download files. However, the system does not enforce fairness and does not specify how the helper may reclaim its contributed bandwidth in the future. Again in [19], where the peer contribution is based on the number of peer uploads and downloads, the computed peer contribution does not guaranty that the peers receive fair utility in return for the bandwidth that they supply. Finally, Karakaya et al. [14] propose a distributed framework in which each peer monitors its neighbours (recording the number of messages coming or going), and the free-riders are located and isolated. However, and unlike this approach, stopping free riders is not the only goal of our approach, indeed we also wish to increase the effectiveness of the download mechanism.

IV. IMPLEMENTATION

When faced with the implementation of our content distri-

TABLE II. SMEPP API

Group management	***Service management***
createGroup	*publish*
joinGroup	*unpublish*
leaveGroup	*getServices*
getGroups	*getServiceContract*
getGroupDescription	*startSession*
getPeers	***Peer management***
getIncludingGroups	*newPeer*
getPublishingGroup	*getPeerId*
Message handling	***Event handling***
invoke	*event (raise)*
receiveMessage	*receiveEvent*
reply	*subscribe*
receiveResponse	*unsubscribe*

 ISBN: 978-1-61208-173-1

bution mechanism, first, the functionality has to be modelled as a service, since SMEPP is service oriented. Then, two main implementation issues must be tackled: first, the content discovery, and second the query and update of the Responsiveness bonus. Both issues will take advantage of the structured overlay network offered by SMEPP.

As we have already stated, SMEPP is based on the concept of peer, group and service. Its API offers primitives to abstract the peer management, group management, and the service management, but also for events and message handling. The SMEPP API is summarised in Table II.

A. Content Discovery

The content discovery process is responsible for finding the peers that provide a specific file in the network. This task is tackled differently in popular P2P systems. In Napster, this responsibility was delegated to central servers which were also responsible of storing the files; In Emule, the task is delegated to many different servers that only store provider's references, not the files. Gnutella implements a pure distributed algorithm to find providers, forwarding query messages through the neighbours. BitTorrent or One-Click Hosting (like megaupload, rapidshare, etc.), do not provide any mechanism for content discovery. In BitTorrent, torrent files containing the description of the shared file are published through webs, mail, forums, etc so, the user must find this torrent file in order to join or to start a download. The same goes for one-click hosting, where search engines or webs specialised in download links are used to find the sources of the content.

As SMEPP integrates Service Discovery functionality, we take advantage of this in order to be able to efficiently discover the file a peer wants to download. We could also opt for an implementation based on events, but the file search would be less efficient and the implementation effort would be bigger. SMEPP defines the services through *contracts* (A XML File). The contract provides descriptive information on the service, while the implementation is the executable service (e.g., a Java service) exposed to the middleware through grounding. A service contract describes "what the service does" (viz., the service *signature*), "how it does it" (viz., the service *behaviour*), and it may include other extra-functional service properties (e.g., QoS). When a service is published, SMEPP generates a *key* from the contract; this *key* determines which peer in the group is responsible for this service. The structured SMEPP overlay network (which uses CHORD protocol [9]) determines the range of *keys* a peer is responsible for, and enables a fast search mechanism thanks to the *key space* defined within the group.

To take advantage of this effective search mechanism, we define a contract for each shared file. This way if two peers share the same file, they will publish the same contract, which will result in the same key, and therefore, the same responsible peer. Thus, peers which share the same file, will publish the same service on the same peer.

B. Responsiveness Bonus Management

In addition to the content discovery, the storage (query and update) of the *Rb* is another important implementation issue. The *Rb* represents the overall contribution of a peer, so it has to be *updated* every time a peer participates in a coalition (as downloader or as participant or manager, typically the former will decrease the *Rb* and the latter will increase it.). Every time a manager set up a coalition for a downloader, it has to *query* the downloader's *Rb*, in order to determine the bandwidth that the coalition will provide.

There are several choices to implement this. On the one hand we can opt for a centralised storage server. This would simplify the update and query processes, and would require less communication effort than in a pure distributed scenario. But this came at a price: that of a single point of failure. On the other hand, the pure distributed scenario, requires a complex algorithm for the calculation of the *Rb* value, such as the ones used in distributed consensus systems [20] which requires a lot of effort to maintain consistency. Finally, we can opt to take advantage of the structured overlay network in a similar way as for the approach of the implementation of content discovery. As foreseen, in this paper we choose this last option for the implementation of our mechanism.

The central idea is that every peer has to delegate the task of storing and updating its *Rb* value to another peer, like in [21]. Using the unique identifier of the peer and a hash function, a peer can find the peer responsible for storing the value of any other peer in the network. This functionality is offered by the overlay network

The *Rb* management functionality will be encapsulated into a service, this service will be responsible for storing and updating a peer's *Rb*. When a peer joins the group, it must publish an *Rb* service with its id (as was proposed for files in content discovery). To update or query an *Rb* a peer just need to invoke the middleware primitive *getServices*, specifying the id of the peer it wants to update or query in a contract template.

TABLE III. MECHANISM MESSAGES

Message	***Sender***	***Receiver***
Manager offer	Downloader (C)	Manager (C)
Offer answer	Participant (S)	Manager (C)
Acknowledge	Downloader (C)	Manager (C)
Cancel	Manager (C)	Participant (S)
New Download	*user*	Downloader(C)
Task	Manager (C)	Participant (S)
Query Rb	Manager (C)	*Rb manager(R)*
Update Rb	Manager (C)	*Rb manager(R)*

 ISBN: 978-1-61208-173-1

```
// CREATE NEW SMEPP peer
 String configFile = args[0];
 Credentials myCredentials = new Creden-
tials("");
 PeerManager peer =
   PeerManager.newPeer(myCredentials, config-
File);
//Join P2P group (find and join)
 GroupDescription myP2PGroup =
   new GroupDescription("P2PGroup ",
   new SecurityInformation(1), "P2PGroup");

GroupId[] groupIds =
   peer.getGroups(myP2PGroup);
//(...)
 GroupId gid = groupIds[0];
 peer.joinGroup(gid);
//PUBLIHS SERVICES
//Main Service
 PeerServiceId psid = peer.publish(gid,
ContractLoader.loadFromFile("MainService.xml"),
   new SMEPPServiceGroun-
ding(MainService.class),
   null,
   null);
//Rb management service
   psid =
   peer.publish(gid,
   ContractLoader.loadFromFile("RbMgr.xml"),
   new SMEPPServiceGrounding(RbMgr.class),
   null,
   null);
//File Sharing services
foreach (file f in sharedFiles){
  String fContract =
   GenerateContract(f,"SharingS.xml");

  psid = peer.publish(gid,
   ContractLoader.loadFromFile(fContract),
   new SMEPPServiceGrounding(SharingS.class),
   null,
   null);
  //Invoke local service to start downloads
 }
}
```

Figure 5. SMEPP peer code

C. Final Implementation

As has been stated, in our approach each peer can work as participant, manager or downloader. The functionality required is summarised in Table III. The table shows the messages sent between the different services. At least three services need to be defined in order to fulfil the requirements of this application:

- *Main Service (C)*: This service enables the download and the manager functionality.
- *Sharing Service (S)*: This service encapsulates the functionality of participants.
- *Rb Management (R):* This service is responsible for enabling the update and the query of the *Rb* value of each peer.

Each peer must at least publish one *Main Service*, one *Rb Management* service and as many *Sharing* services as files it shares.

The code of a SMEPP peer running our mechanism is illustrated in the Figure 5. For the sake of simplicity we skip base cases, the steps are the following: first we use the *newPeer* primitive to create the SMEPP peer (this connects the peer to the SMEPP network and assigns it an Id). For security, SMEPP requires the provision of valid *Credentials* in order to successfully join the network. The *PeerManager* object allows us to invoke peer's primitives. Next the peer has to find the concrete group where our P2P content distribution mechanism is running, to do this, it performs a search of a group providing its description, using *getGroups* primitive. Once the group is found, the peer joins it with *joinGroup* primitive. Next, the peer publishes all the, previously explained, services. This is performed using the *publish* primitive. Up to this point, we have a peer in the group sharing files, to start a download the user will invoke the local service specifying the file info (*NewDownload* message), this will start the exchange of our mechanism's message between the different peers in the group, performing the coalition and incentives mechanism (as explained in Section III).

To summing up, SMEPP simplifies the implementation of this kind of application, as the above code shows. Not only abstracting the underlying complexity but also offering an efficient look up mechanism for file discovery. Moreover, it tackles the security issues internally, without additional effort.

V. CONCLUSION AND FUTURE WORK

In this paper, we have presented the implementation of a coalition and incentive based P2P content distribution system.

Our mechanism is based on game theory and takes into account the rational and self-interested behaviour of the peers. The central idea is that incentives encourage participation; each time a participant contributes in a coalition they receive a reward. The fairness of the rewards division within a coalition is guaranteed by means of the game theory concept of *core*. These rewards are accumulated in the *Responsiveness bonus*, which represents the overall contribution of the peer to the system, and this is used to increase or decrease the quality of service of the downloads the peer performs. This way our approach manages to promote the cooperation, and therefore, reduces the free riding phenomenon. Moreover, simulations showed that download times are improved.

When dealing with the development of real distributed applications, is has been proven that to use a middleware simplifies the implementation issues. In this paper, we proposed to use SMEPP middleware to ease the distributed implementation of the above mechanism. SMEPP is a Secure Middleware for Embedded Peer to Peer Systems and another of our publications.

 ISBN: 978-1-61208-173-1

For the real implementation of our content distribution system different issues have been taken into account. Mainly, two issues must be addressed: content discovery (taking advantage of the middleware overlay), and the storage, query and update of the Responsiveness bonus. Taking advantage of the middleware, the complexity of the development of the distributed application is abstracted, moreover the process of content discovery is delegated in the middleware, what greatly eases the implementation, as foreseen.

As future work, we plan to implement our coalition approach over Gnutella protocol [22]. The objective would be to compare and analyse the performance of these two implementations.

ACKNOWLEDGMENT

This work is partially supported by EU funded project FP6 IST-5-033563 and Spanish projects TIN2008-01942 and P07-TIC-03184.

REFERENCES

[1] Ipoque Internet Study 2008-2009. http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009. 26.09.2011.

[2] Saroiu, S., Gummadi, P. K., and Gribble, S. D, "Measurement study of peer-to-peer file sharing system". In Kienzle, M. G. and Shenoy, P. J., Multimedia Computing and Networkin, pp. 156–170. SPIE, San Jose, CA, USA. 2002.

[3] Handurukande, S. B., Kermarrec, A.-M., Le Fessant, F., Massouli´e, L., and Patarin, S. , "Peer sharing behaviour in the edonkey network, and implications for the design of server-less file sharing systems". SIGOPS Oper. Syst. Rev., vol. 40, pp. 359-371, 2006.

[4] Karakaya, M., Korpeoglu, I., and Ulusoy, O. , "Free riding in peer-to-peer networks". Internet Computing, IEEE, vol. 13, pp. 92–98, 2009.

[5] Belmonte M.V., Díaz M., and Reyna A., "A Coalition based incentive mechanism for P2P content distribution systems" Proceedings of the 3rd. international conference on agents and artificial intelligence. pp. 15-24. ICAART 2011. Rome, Italy, January - 2011.

[6] Díaz M., Garrido D., Reyna A., and Troya J.M.,"SMEPP: A Secure Middleware for P2P Systems". Horizons in Computer Science. Volumen III. Nova Publisher 2010.

[7] Díaz M., Garrido D., and Reyna A. "SMEPP and the internet of things". In Workshop on Future Internet of Things and Services. CD.ROM, 2009.

[8] Caro, R.J., Garrido, D., and Plaza Tron, P. , "SMEPP: A Secure Middleware For Embedded P2P", 2009. ICT-MobileSummit Conference Proceedings. IIMC International Information Management Corportaion, 2009.

[9] Stoica I., Morris, R., Karger, D.R., Kaashoek, M.F., and Balakrishnan, H., "Chord: A scalable peer-to-peer lookup service for internet applications". In SIGCOMM, pp. 149–160, 2001.

[10] The JXTA home page. www.jxta.org. 26.09.2011

[11] Deliverable 6.4 SMEPP Validation. www.smepp.net 26.09.2011

[12] Kahan, J P. and Rapoport, A., "Theories of coalition formation", L. Erlbaum Associates, Hillsdale, New Jersey London,1984.

[13] Bertsekas, D.P. and Gallager, R.G., and Humblet, P., "Data networks" Prentice-Hall, New York, NY, USA, 1987.

[14] Karakaya, M., Korpeoglu, I., and Ulusoy, O. "A connection management protocol for promoting cooperation in Peer-to-Peer networks". Computer Communications, 31, pp. 240-256. 2006.

[15] Kulbak, Y., Bickson, D., et al.(2005). The emule protocol specification http://www.cs.huji.ac.il/labs/danss/p2p/resources/emule.pdf; 15.06.2011.

[16] Cohen, B. "BitTorrent protocol specification". http://www.bittorrent.org/beps/bep_0003.html 26.9.2011

[17] Garbacki, P., Iosup, A., Epema, D., and van Steen,M. "2fast : Collaborative downloads in p2p networks". In Peer-to-Peer Computing, IEEE International Conference on, pp. 23–30. IEEE Computer Society, Los Alamitos, CA, USA. 2006.

[18] Karakaya, M., Korpeoglu, I., and Ulusoy, O. "Counteracting free riding in Peer-to-Peer networks". Computer Networks, 52, pp. 675–694. 2008.

[19] Mekouar, L., Iraqi, Y., and Boutaba, R. "Handling Free Riders in Peer-to-Peer Systems". Agents and peer-to-peer computing: 4th international workshop, AP2PC 2005, Utrecht, The Netherlands, July, pp. 58–69. Springer-Verlag, New York, NY, USA. 2006.

[20] Zhou, R., Hwang, K., and Cai, M. "GossipTrust for Fast Reputation Aggregation in Peer-to-Peer Networks" IEEE Transactions on Knowledge and Data Engineering, CA, USA, pp.1282-1295. 2008.

[21] Kamvar, S.D., Schlosser, M.T., and Garcia-Molina, H.. "The Eigentrust algorithm for reputation management in P2P networks". In Proceedings of the 12th international conference on World Wide Web (WWW '03). ACM, New York, NY, USA, pp. 640-651. 2003.

[22] "Gnutella Protocol Specification". Online. http://www. stanford.edu/ class/cs244b/gnutella_protocol_0.4.pdf.26.09.2011.

 ISBN: 978-1-61208-173-1

On the Performance of OpenDPI in Identifying P2P Truncated Flows

Jawad Khalife, Amjad Hajjar
Faculty of Engineering, IT department
Lebanese University
Beirut, Lebanon
jawad_khalife@hotmail.com, arhajjar@idm.net.lb

Jesús Díaz-Verdejo
Dept. Signal Theory, Telematics and Commun.
University of Granada
Granada, Spain
jedv@ugr.es

***Abstract*—This paper aims to show the impact on classification accuracy and the level of computational gain that could be obtained in applying deep packet inspection on truncated peer to peer traffic flows instead of complete ones. Using one of the latest open source classifiers, experiments were conducted to evaluate classification performance on full and truncated network flows for different protocols, focusing on the detection of peer to peer. Despite minor exceptions, all the results show that with the latest deep packet inspection classifiers, which may incorporate different helper technologies, inspecting the first packets at the beginning of each flow, may still provide concrete computational gain while an acceptable level of classification accuracy is maintained. The present paper discusses this tradeoff and provides some recommendations on the number of packets to be inspected for the detection of peer to peer flows and some other common application protocols. As such, a new sampling approach is proposed, which accommodates samples to the stateful classifier's algorithm, taking into consideration the characteristics of the protocols being classified.**

Keywords-IP traffic classification; p2p; peer to peer; deep packet inspection; DPI optimization

I. INTRODUCTION

Traffic identification is a hot research topic, especially when it comes to complex Internet applications, such as P2P (peer to peer), using port obfuscation, encryption, and tunneling [1]. As they inspect the full packet payloads to match specific protocol patterns or signatures, DPI (Deep Packet Inspection) based methods [2], are characterized by the high level of classification accuracy they provide. However, the associated high computational cost and some user privacy issues arise some concerns regarding their use in real environments, especially in high-speed networks.

Optimizing DPI based methods, is thus becoming an important research trend attempting to enhance the classifier performance in terms of low computation, while maintaining an acceptable level of accuracy.

Reducing the input size required by the classifier through sampling can be considered one of DPI optimization means, which is not only supposed to reduce computational requirements, but also to ensure an acceptable level of user privacy. While different sampling policies exist, traffic sampling could be applied on two different levels: on the packet payload level, through partial packet payload inspection, and on the flow level, through inspecting only a few packets from within the complete traffic flow.

In [3], we studied how far DPI could be optimized through packet level sampling. Results showed that, unless just few bytes (not more than 128 Bytes) were truncated from the end of the packet payload, a sharp decrease in the accuracy will be apparent.

As part of our work in progress, this paper attempts to optimize DPI classifiers through flow sampling to which we will refer as flow truncation. It is important to note that we consider truncation as a particular case of the sampling technique, by inspecting a certain number of elements at the beginning of a stream (first bytes in a packet payload, first packets in a flow). We focus on P2P applications as we consider that identifying p2p traffic is one of the most complex classification tasks, especially when compared with other common application protocols. In fact, and as shown in [1], most works were emphasizing on their ability to detect p2p traffic as a key indicator of the quality of the classifier, and most importantly, were providing a better understanding of P2P traffic characteristics as part of the detection mechanism.

While flow sampling is supposed to decrease the classification time, it is still a lossy process, which would affect accuracy. *What impact would the flow truncation process have on DPI accuracy and at which computational gain?* This is the question we are trying to answer in this work through our conducted experiments and the obtained results.

Our goal(s) through this work can be defined as follows:

1) To determine a minimum number of packets to be inspected within a p2p flow to be classified with an acceptable level of accuracy. To generalize for other protocols.

2) To discover to what extent per-flow sampling would optimize DPI classifiers in the sense of decreasing computational costs while maintaining an acceptable level of classification accuracy.

The remaining of this paper is organized as follows: Section II provides an overview of DPI optimization means and parameters for their evaluation, focusing on flow sampling techniques and putting our work in perspective with previous works in the literature. Section III describes the dataset we used for the experiments, and the OpenDpi [4] tool in the way it analyzes and labels packets and flows. Section IV provides a description of the way we used to truncate the flows. Section V shows our conducted experiments' results in terms of accuracy and computational cost and highlights on some important results and special

 ISBN: 978-1-61208-173-1

cases. Finally, Section VI presents the conclusion and future work.

II. STATE OF ART IN DPI OPTIMIZATION

DPI [2] based identification methods have no restrictions on inspecting the full contents of the payloads. As such, there are some concerns related to both user privacy and computational performance.

In this paper, we will focus on software based optimizations which concept is to reduce the size of DPI input through sampling techniques [5] that are considered as a general mean of input reduction. As many works [6] [7] [8] show, sampling techniques can be integrated within the traffic classification process.

Amongst the ways of categorizing sampling techniques we have the *per-packet* payload sampling [3] [9] [10], i.e., sampling bytes from within the packet payload, and the *per-flow* packet sampling [7] [8] [11] [12], i.e., sampling a subset of packets from within the whole traffic flow or a combination of both [13].

Per-flow packet sampling for DPI classification is shown in many papers with different sampling policies: Bloom filters in [8], Deep Packet Inspection using Parallel Bloom Filters [14] [15], k-ary sketch [16], Related sampling [12], and Mask-match sampling in [17]. Chen et al. [7] suggested six sampling strategies and showed how they affect DPI identification systems.

In [3], we tested *per-packet* payload sampling. In this paper, we consider *per-flow* packet sampling. However, we did not follow any specific sampling policy. Instead, given an accuracy level to be maintained, *we simply recommend to only inspect a predetermined number of packets N_{min} at the beginning of a traffic flow (p2p or other protocols).* Thus, the sampling rate will be *N_{min} packets per flow.*

Although an in-depth comparison of different sampling methods is beyond the scope of this paper, we will briefly compare the most relevant ones to our work. Our optimization concept is detailed in Subsection II.C.

Some of the sampling modes discussed in [7] can timely investigate the traffic load conditions of the links. However, their results were difficult to generalize since they were affected by many factors as they concluded.

Sampled NetFlow was mentioned in [11], where Carela-Español et al. find that packet sampling has a severe impact on the performance of the classification method. They were able to achieve an overall flow classification accuracy of 85% for a sampling rate of 1/100.

RelSamp (Related Sampling) [12] proposes that flows, parts of the same application session, are given higher probability. However, in [12] RelSamp was compared to Sampled NetFlow, which is a sampling technique for network monitoring rather than traffic classification.

Mask-match sampling method discussed in [17], provided 94% accuracy for UDP flows for a sampling rate of 0.1. However, this method focuses mainly on long flows, and the validity of the samples is related to the randomness of the ID field of the IP packets headers.

Similar to our work, Canini et al. [8] used Bloom filter to sample the first 10 packets of each flow while a negligible loss in the accuracy was encountered (0.0047%). However, they used L7 filter and they encountered some false positive figures due to Bloom Filters. Fernandes et al. [13] also proposed a similar work, where a combination of per-flow and per-packet sampling was used to capture only few packets (7 packets) per flow and a fraction of its payload without a significant impact on accuracy. However, the analysis in [13] did not provide protocol oriented results (such as p2p) and experiments were based on L7-filter tool.

III. OPTIMIZATION THROUGH FLOW TRUNCATION

Our goal in optimizing DPI is to maintain classification accuracy level as high as possible while trying to decrease the required computational cost.

In this subsection, we will focus on both the theoretical concept of optimizing the processing time through flow truncation, and on the quasi-theoretical estimation for this parameter as well.

Theoretically, the model of the DPI classifier proposed in [18] includes 5 processing blocks. Cascarano et al. studied the cost of each block and concluded that the most important is the cost of the pattern-matching block, which according to the study, has a linear dependence on the number of input characters.

For simplicity reasons, and according to the target of the experiments, we will model individual packet classification as a process composed of just two modules or steps:

- A packet-handling module, mainly devoted to preprocess the packet and identify the flow the packet belongs to (reading the packet, getting flow identifiers, searching for the flow in active flow list, etc.). This processing is mainly related to packet header.
- A packet-inspection module (referred to as "pattern matching block" in [18]), devoted to the classification of the packet. This module handles both the header and the payload of the packet.

Thus, the time cost related to individual packet classification t_{pc}, can be approximated as:

$$t_{pc}=t_h + t_i \quad (1)$$

where t_h is the packet handling time, and t_i is the packet inspection time.

In our concept of optimization through flow truncation, we emphasize on t_i as we consider it to be the only sensitive term to flow truncation. Through flow truncation, we intend only to classify packets which ordinal number inside the flow is lower than a predefined threshold (N_{min}) within each flow. However, this does not mean that packets over N_{min} are not be parsed at all. On the contrary, these packets still have to be handled by the classifier simply for determining to which flow they belong. The difference is that for these packets the inspection part is to be omitted. So, it is important to note that, for these packets, t_h cannot be avoided by the classifier, as it is evidently impossible to know if a given packet belongs to an existing flow without parsing its header at least. The decision to inspect the packet or not

 ISBN: 978-1-61208-173-1

depends on its ordinal number being lower or higher than N_{min}.

As a result, with flow truncation, we are supposed to eliminate packet inspection time t_i for packets which position in the flow is over N_{min}. This is supposed to decrease the global classification time for the whole traffic according to the following simple approximations:

- With *no flow truncation*, the time for analyzing a flow t_{fc} can be approximated as:

$$t_{fc} = N_p \cdot E[t_h + t_i] \quad (2)$$

where N_p is the number of packets in the flow and $E[t_h + t_i] = E[t_{pc}]$ is the average packet classification time. $E[t_{pc}]$ depends on the flow size and the length of the payloads, and on the protocol to which the flow belongs.

- With *flow truncation*, assuming $N_p > N_{min}$, the time for analyzing a flow, t'_{fc}, becomes:

$$t'_{fc} = N_p \cdot E[t_h] + N_{min} \cdot E[t_i] \quad (3)$$

Our proposal, based on N_{min}, and to the best of our knowledge, is different from the other sampling methods in the following points:

- Most sampling policies were general with no specific attention on the detection of a particular protocol (or class of protocols), while our aim is to find the number of packets N_{min} necessary for the detection of each particular protocol. This approach fits with our long-term objective, which is the detection of P2P flows.
- Most sampling policies provide sampling rates, which implies that the number of sampled packets will increase as long as the flow is under course while in our proposal, it is fixed to N_{min} *per flow*, which is an efficient method that yields higher computational gains for large flows.
- Most sampling policies neglect the effect of non sampled packets, while in our proposal we have to parse all packets.
- In some papers, experiments were restricted to DPI as implemented by L7-filter tool while in the latest classifiers, such as OpenDPI, the DPI technology is being enhanced through integration with other helper methods such as behavioral and statistical analysis.

To conclude with common features for all sampling methods, Guo et al. [6] showed that as the packet sampling probability decreases, the false negative rates become higher. Therefore, it can be stated that: the maximum value of sampling rate is limited by the affordable computational cost and the minimum value of sampling rate is limited by the acceptable accuracy value.

As our sampling rate is defined as: "*N_{min} packets per flow*", what is the recommended value for N_{min} to identify p2p and other protocols? The following sections will help in answering this question through experimental results.

IV. OPENDPI AND TESTBED

For the experiments, we used OpenDPI, which is derived from the commercial PACE product [19]. The core of OpenDPI is a software library designed to classify internet traffic according to application protocols. In its current version, up to 101 different protocols can be identified, including some P2P protocols. In addition to pattern matching, OpenDPI incorporates different techniques such as behavioral (by searching for known behavioral patterns of an application in the monitored traffic) and statistical analysis (by calculating some statistical indicators that can be used to identify transmission types, as mean, median and variation of values used in behavioral analysis and the entropy of a flow). Our experimental setup is described next.

First, we use a customized tool based on the OpenDPI library, which is able to follow and differentiate the packets in each flow and to provide both flow and packet based outputs. Second, we used a dataset of real traffic captured at the access link of a medium size institution over 3 days. Complete flows in both directions were captured at a border router. We have chosen a subset of randomly selected files (totaling 3 GB) from our original dataset on which we run classification experiments. Then, by using the customized OpenDPI tool over the database subset and using complete flows with full packet payloads, we have built the "ground truth", i.e., the set of correctly labeled flows and packets (without truncation) that will be used as the reference for the analysis of flow truncation as described in the following section.

V. FLOW TRUNCATION RESULTS AND ANALYSIS

To be able to generate accuracy results without truncating flows, we customized OpenDPI to output the packet ordinal number inside the flow the packet belongs to at which detection is achieved. As described in OpenDPI documentation, the flow is classified according to the first recognized packet. In what follows, we will refer to this number as packet detection number or flow detection number. Then, to truncate flows, we have customized the code to be able to classify, within each flow, only packets with numbers less than N_{min}, and as mentioned in Section II, to solely handle remaining packets just for determining to which flow they belong. Note that with this customization, we could obtain computational measurements and validate accuracy results obtained previously through the flow detection number.

A. Accuracy Results

As shown in [1], a common framework is not yet defined for traffic classification methods. Therefore, we referred to existing works in the literature most of which commonly considered values above 90% as acceptable levels of classification accuracy. The distribution (histogram) of the flow detection number for both P2P and non-P2P protocols is depicted in Figure 1 for the full dataset, which contains 40340 P2P flows out of 4859208. As shown, there is a big number of flows for which the detection is achieved with just a few packets. Proportionally, the number of flows with high

 ISBN: 978-1-61208-173-1

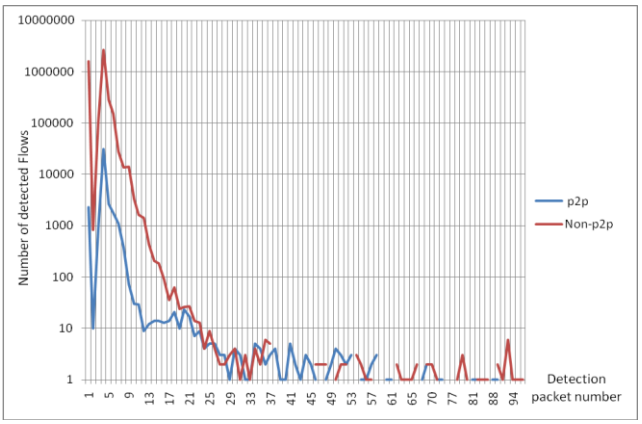


Figure 1. Histogram showing the number of detected p2p and non p2p flows in function of the flow detection number. Data for detection packet number over 100 is negligible and is not shown.

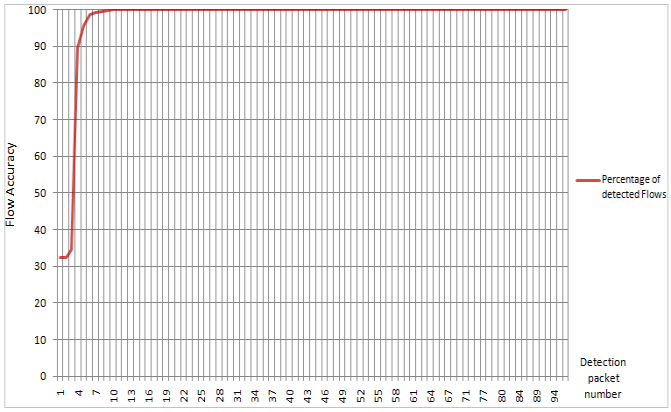


Figure 2. Global flow accuracy as a function of the packet detection number.

(i.e., greater than 50) flow detection number is almost negligible.

In fact, if we represent the percentage of flows that have been classified vs. the flow detection number (Figure 2), we can see that an important fraction of the flows can be classified with N_{min} below 20. Moreover, all protocols are mainly being detected within the first ten packets with a flow accuracy value of 99.90%.

As depicted also in Figure 2, for N_{min}=4, flow accuracy degrades to 84.35% while it jumps to 99.15%, for N_{min}=10. It is noticed that for N_{min} values greater than 10, only a very slight increase in accuracy is obtained. For example, for N_{min}=20, accuracy becomes 99.54%, with an increase of 0.39% compared to N_{min}=10.

Regarding the behavior on a per protocol basis, Figure 3 shows the average flow detection number for most common protocols in the dataset. Although some protocols like iMESH and Bitorrent show higher average values, most protocols averages were below 10 packets. If we further analyze the results on a per protocol basis, similar results could also be obtained. As an example, Figure 4 shows the histogram of the flow detection number for two relevant protocols: Bittorent (Figure 4.a) and http (Figure 4.b). Thus,

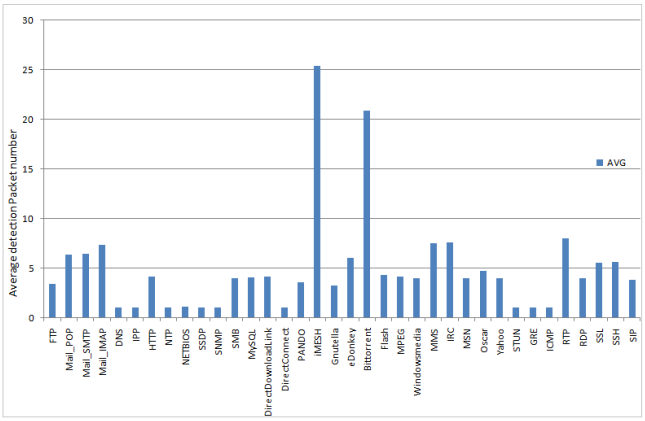


Figure 3. Average detection packet number for different protocols in the dataset.

an accuracy value of 99.91% for http and 99.18% for Bittorrent can be achieved if the 10 packets rule is still being respected for both protocols.

It is worthy to note that according to [19], OpenDPI in bandwidth management systems only scans for patterns in the first 1-3 packets for unencrypted and 3-20 packets for encrypted communication protocols: A fact which does not apply to OpenDPI in traffic classification systems, as shown in our experiments.

As it was clearly noticed from the previous results, in order to reach 99.15% of p2p flow accuracy, the classifier should inspect at least 10 packets. The same applies for remaining protocol. As a result, one optimal value to be recommended for N_{min} is 10. However, this is not mandatory as it is related to the required level of accuracy.

B. Computational Cost Results

Different set of evaluation tools and parameters were proposed for comparing classifiers, such as, Netramark [20] TIE [21] and perfprofiling for Snort [6]. However, according to [22], a commonly agreed upon workload for the evaluation of deep packet inspection architectures is still missing. In our case, we needed higher level evaluation parameters regardless of the pattern matching technique, therefore, we have chosen simple processing time as in [7] [18] using Linux monitoring tools [23]. For this purpose, we evoked the insertion into the classifier code at the proper places, of time related function calls providing granular results at the microseconds' level. We compiled the classifier code with GCC v4.4.3 with -O3 optimization level, on the testing server having the hardware specifications of 8 GB of memory, 2 Intel(R) Xeon(R) 2.66GHz processors with 4 cores each.

Tests were performed over one of the captured files for N_{min}=20. The most relevant features are shown in Table I, while Table II shows the results.

Flow and packet classification times were calculated by dividing the classification time for all the flows respectively by the number of flows for t_{fc} and by the number of packets for t_{pc}. Consequently, the gain in processing time when using truncated flows is 9.63%. .

 ISBN: 978-1-61208-173-1

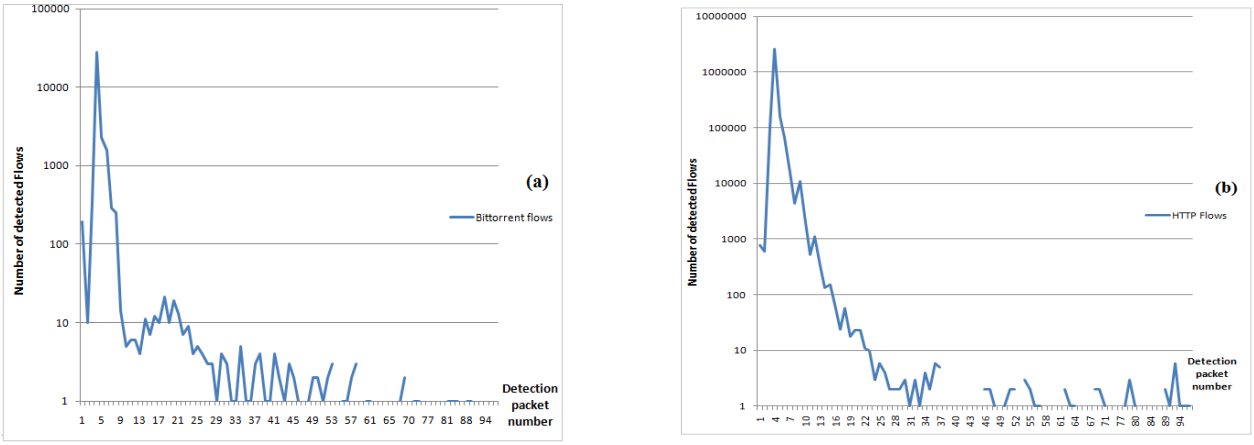


Figure 4. Histogram showing the number of detected flows in function of packet detection number for: (a) Bittorrent protocol (b) HTTP protocol.

As shown through the previous results, a *computational time gain* of 9 % can be obtained by classifying the first only 20 packets while maintaining an accuracy level of more than 99 %.

In Table I, the percentage of flows with N_p less than N_{min} is important for the effectiveness of the truncation. In fact, the smaller this value is, the bigger the time gain would be and the flow truncation will become more effective. In showing the importance of this parameter, we have chosen N_{min}=20 instead of N_{min}=10 as the accuracy is slightly different between these values, as depicted in Figure2. Then, we referred to comparing the quasi-theoretical and experimental measured values of the time gain. We measured the average for t_i=5.35µs, N_p=42, with N_{min}=20, then, when multiplied by the total number of flows, (3) gives the total processing time for all the flows. The obtained result is a quasi-theoretical value of 11.8 % for the time gain instead of 9.63%, which is the measured one. Effectively, this difference is due to the assumption we took that all of the flows should have theoretically an average number of packets (N_p=42) more than N_{min}=20, which is not exactly the case for the capture file we tested, having 30% flows with N_p less than N_{min} as shown in Table I.

C. Comparison with other sampling schemes

In comparing our approach (for N_{min}=20), with EIM (Equidistant Invariable Mode) [7] having a sampling rate of 7/13, 164084 packet samples are to be inspected, which means 36% increase in the classification time due inspecting additional 140,634 packets. Figure 1, shows that, for N_{min}=7, 97.96% of accuracy could be obtained, which applies to EIM scheme only if the first 7 packets were sampled, otherwise, EIM accuracy will drop to an unacceptable value, due to the stateful inspection of OpenDPI, as explained next.

TABLE I. CHARACTERISTICS OF THE CAPTURE FILE USED FOR TEST

Total # flows	**# P2P flows**	**# Flows N_p< N_{min}**	***Number of packets***	
			Under N_{min}	***Over N_{min}***
7337	188	30%	104678	200040

TABLE II. THE OBTAINED COMPUTATIONAL RESULTS

File N_{min}=20	**Classification time** (in µs)		
	All flows	***Flow (t_{fc})***	***Packet (t_{pc})***
Full flows	7300301	995	24
Truncated flows	6596881	899	21

D. Classifying the DPI Classifier

Although a concrete gain in classification time was obtained through flow truncation, still the 9.63% value did not meet with our expectations. In fact, if we assume that OpenDPI is inspecting all packets over N_{min} the gain has to be theoretically higher. However, this fact helped in reverse engineering an important aspect of the classifier itself, which in turn could interpret the moderate gain we obtained. In fact, OpenDPI seems to be incorporating DFI (Deep flow inspection) beside the DPI classification mechanism. Specifically, OpenDPI shows a stateful or PBFS like (Packet Based per Flow State) classification behavior. As per the taxonomy presented in [24], PBFS based DPI classifiers focus on the first packets of each session. Thus, they have a built-in feature of requiring less input than other classifiers.

In this context, the sampling scheme we proposed seems to be more convenient to PBFS classifiers as it focuses on packets where the PBFS classification decision is being made, whereas in most sampling methods, packets have to be continuously inspected as long as the flow is under course. In this regard, it may be required that all sampling works joint with classification should be reconsidered, especially when used with PBFS based classifiers.

E. Special cases

During experiments, some exceptional cases were noticed. Though these cases had no significant impact on the presented results, it is worthy to provide a preliminary interpretation while detailed explanations should be left for future work.

1) Deviators Flows

As shown in Figure 2, we calculated the average *detection packet number* for each protocol. However, we found some flows which detection number is much deviated

 ISBN: 978-1-61208-173-1

from the calculated average, we call these flows *deviators*. We validated the fact that the presence of most deviators is due to flows that were under course during the start of the capture. In addition, this highlights the importance of the first flow packets to OpenDPI, which when lost, will cause flows to be detected at packet numbers deviated from the average.

2) Packets Changing Protocol

Some minor flows were noticed to be changing their protocol even after the first time detection. This may lead to an error in classification, and can be considered as one of the weaknesses of our approach to be dealt with in future enhancements.

VI. CONCLUSION AND FUTURE WORK

This paper aims to optimize DPI-based classifier by decreasing the required computational cost while maintaining acceptable levels of flow classification accuracy for p2p and other application protocols. As opposed to many sampling works, our approach is to reduce input requirement through inspecting only a fixed number of packets from within the flow beginning. Our conducted experiments show that when inspecting the first 10 packets for all protocols, including p2p, more than 9% of classification time can be saved while a flow accuracy level over 99% can be still maintained. However, future enhancements may have to deal with some weaknesses and special cases detected within our approach such as, deviators flows, packets changing protocol within the same flow, studying the effect of modifying the first packets, distinguishing between packets in the uplink and downlink directions and their contribution in the classification process. Finally, it is important to note that when used jointly with stateful and PBFS based classifiers, sampling methods should accommodate to the importance of the first packets in classifying the whole flow. This fact highlights the importance of our sampling approach which accommodates samples to the stateful classifier's algorithm by focusing on the first packets, and takes into consideration as well, the characteristics of application protocols being classified by sampling a convenient number of 10 packets sufficient for identifying most application protocols including p2p. In this context, accommodating enhancement means to the classifier's algorithm from one hand, and to the classified traffic characteristics, on the other, would be a good practice for any related future work.

ACKNOWLEDGMENT

This work has been partially supported by Spanish MICINN under project TEC2008-06663-C03-02.

REFERENCES

[1] T. Nguyen and G. Armitage, " A Survey of Techniques for Internet Traffic Classification using Machine Learning", IEEE Communications Surveys & Tutorials, v. 10, pp. 56-76, 2007.

[2] Allot Communications "Digging Deeper Into Deep Packet Inspection (DPI)." White paper. Available at https://www.dpacket.org 14.09.2011

[3] J. Khalife, A. Hajjar, and J. Díaz-Verdejo, "Performance of Opendpi To Identify Truncated Network Traffic", In Proc. DCNET 2011, pp. 51-56, Seville.

[4] http://www.opendpi.org 14.09.2011

[5] R. Jurga and M. Hulbój, "Packet Sampling for Network Monitoring", Technical Report, CERN | HP Procurve openlab project. Available at http://www.zdnetasia.com 14.09.2011

[6] Z. Guo and Z. Qiu, "Identification Peer-to-Peer Traffic for High Speed Networks Using Packet Sampling and Application Signatures", In Proc. ICSP2008, pp. 2013-2019.

[7] H. Chen, F. You, X. Zhou, and C. Wang, "The study of DPI identification technology based on sampling", ICIECS 2009, 2009, pp. 1-4.

[8] M. Canini, D. Fay, D. Miller, A. Moore, and R. Bolla, "Per Flow Packet Sampling for High-Speed NetworkMonitoring", In Proc. COMSNETS'09, 2009, pp. 1-10.

[9] D. Ficara, G. Antichi, A. Di Pietro, S. Giordano, G. Procissi, and F. Vitucci "Sampling Techniques to Accelerate Pattern Matching in Network Intrusion Detection Systems", In Proc. ICC2010, 2010, pp. 1-5.

[10] G. Aceto, A. Dainotti, W. de Donato, and A. Pescapé, "PortLoad: taking the best of two worlds in traffic classification", In Proc. of INFOCOM 2010,2010, pp. 1-5.

[11] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the impact of sampling on NetFlow traffic classification", Computer Networks, Volume 55, Issue 5, 1 April 2011, pp. 1083-1099 .

[12] M. Lee, M. Hajjat, R. Kompella, and S. Rao, "RelSamp: Preserving Application Structure in Sampled Flow Measurements", In Proc. INFOCOM 2011, 2011, pp. 2354-2362.

[13] S. Fernandes, R. Antonello, T. Lacerda, A. Santos, D. Sadok, and T. Westholm, "Slimming Down Deep Packet Inspection Systems", In Proc. INFOCOM Workshops 2009, 2009, pp. 1-6.

[14] S. Dharmapurikar, P. Krishnamurthy, T.Sproull, and J. Lockwood, "Deep Packet Inspection using Parallel Bloom Filters", In Proc. High Performance Interconnects 2003, 2003, pp. 44-51.

[15] Y. Li "Memory Efficient Parallel Bloom Filters for String Matching", In Proc. NSWCTC 2009, 2009, pp.485-488.

[16] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications", In Proc. of ACM SIGCOMM Internet Measurement Conference IMC'03, October 2003.

[17] R. Cong, J. Yang and G. Cheng, "Research of Sampling Method Applied To Traffic Classification", In Proc. ICCT 2010, 2010, pp. 112-115.

[18] N. Cascarano, A. Este, F. Gringoli, F. Risso, and L. Salgarelli, "An Experimental Evaluation of the Computational Cost of a DPI Traffic Classifier", Proc. GLOBECOM'09, 2009, pp. 1-8.

[19] http://www.ipoque.com 14.09.2011

[20] S. Lee, H. Kim, D. Barman, S. Lee, C. Kim, and T. Kwon, "NeTraMark: A Network Traffic Classification Benchmark", ACM SIGCOMM Computer Communication Review, Volume 41 Issue 1, January 2011.

[21] http://www.grid.unina.it 14.09.2011

[22] M. Becchi, M. Franklin, and P. Crowley, "A Workload for Evaluating Deep Packet Inspection Architectures", In Proc. IISWC 2008, pp.79-89.

[23] http://www.tldp.org 14.09.2011

[24] F. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus "Lightweight, Payload-Based Traffic Classification: An Experimental Evaluation", In Proc. ICC 2008, 2008, pp. 5869-5875.

 ISBN: 978-1-61208-173-1

Applying Certificate-Based Routing to a Kademlia-Based Distributed Hash Table

Michael Kohnen, Jan Gerbecks, Erwin P. Rathgeb
University of Duisburg-Essen
Computer Networking Technology Group
Essen, Germany
{Michael.Kohnen, Erwin.Rathgeb}@iem.uni-due.de, Jan.Gerbecks@stud.uni-due.de

Abstract–Most Distributed Hash Table (DHT) algorithms have proven vulnerable against a multitude of attacks. Countermeasures using reputation systems to generate trust values have been developed and analyzed. These analyses mostly refer to unstructured peer-to-peer (P2P) networks. In this paper, we present our concept for applying trust values to the bootstrap, lookup, PUT and GET processes of structured P2P networks and evaluate it using the Kademlia DHT algorithm in a binary trust environment created by certificates.

Keywords–DHT; Security; Kademlia; Trust; Reputation; Certificates

I. INTRODUCTION

Research has proven that Distributed Hash Table (DHT) algorithms are vulnerable to different kinds of attacks [1]. These attacks include the Sybil Attack [2], eclipse attacks and attacks on the routing and storage mechanisms. As one possible solution against those threats, trust-based systems have been invented to improve security. A lot of the existing research about trust and reputation management in a peer-to-peer (P2P) environment focuses on unstructured networks [3] [4]. However, considering real-world implementations, the structured networks prevail: Popular P2P applications such as BitTorrent [5] and eMule [6], used by millions of users, implement the Kademlia algorithm.

We, therefore, aim to analyze the feasibility of trust and reputation mechanisms in structured P2P networks. To test the general functioning, we use a simplifying assumption of binary trust created by certificates. In the following section, we present the related work. In Section III, we explain our concept, followed by Section IV with its evaluation. Section V concludes the paper.

II. RELATED WORK

Marti and Garcia-Molino [7] categorize P2P reputation systems and divides them into the three functionalities "information gathering", "scoring and rating" and "response", each having several sub functions. Furthermore, the authors define factors influencing reputation systems and discusses them.

Gomez Marmol and Martinez Perez [8] offer an overview of the current state of P2P reputation systems. EigenTrust [3] is one of the popular ones. It uses a rating system similar to eBay's: A node can receive either a positive or a negative rating after a transaction. The EigenTrust algorithm then defines how the ratings from different nodes can be combined and normalized.

EigenTrust and the other algorithms presented in [8] either have been tested using unstructured P2P networks or mention structured networks only for storing the trust information. They do not analyze the specific impact of using trust information for routing and storing in structured networks.

Therefore, we aim to analyze the consequences of using trust information in structured P2P networks during the joining, routing, storing and retrieving processes. According to [9], these processes will be referred to as bootstrap, lookup, PUT and GET process, respectively. In this paper, we present a basic concept of using trust values to enhance the security of a DHT algorithm.

In the following, we discuss the trust values' consequences for bootstrapping and performing lookup, PUT and GET actions. Afterwards, we evaluate our concept using the Kademlia DHT algorithm [10].

III. OUR CONCEPT

We seek an approach that enables a node to determine the authenticity of a result on its own. A node shall be able to decide for itself whether it regards an action as successful. It shall also be able to abort or ignore an action if it does not trust the result.

To achieve this, trust values are used: We have each node assign trust values to each other node it encounters. Then, we define a minimum trust value another node needs to have so that a node uses it for its actions in the network [7]. As a consequence, a node is able to determine whether the result of an action (bootstrap, lookup, GET, PUT) is valid. If it does not find enough trustworthy nodes, it cancels its action in order to protect itself from invalid results.

A. Functioning

Once a node's trust value is known, it is used to determine whether the node should be used. We propose to use a single minimum trust value threshold defining whether another node is used for outgoing requests of all kinds, as a node that does not answer GET requests correctly, for example, should not be used for other purposes.

Trust values are not assigned globally, but individually by each node. Possible reasons for this are, e.g., a result node possessing a certificate issued by another CA or the requesting

 ISBN: 978-1-61208-173-1

node using a lower minimum trust value threshold than the responding node. This individual assignment of trust values leads to their local storage on each node and therefore an extension of the routing table. From this, it follows that…

- … incoming requests shall always be answered in a correct way, regardless of the trust status of the requesting node, and …
- … that the routing table of a node shall not only contain nodes it trusts, but also untrusted nodes.

If nodes would only answer request coming from trusted nodes, a network partitioning could result. The same applies to the routing table: If a node can only answer with next hops that it trusts, the requesting node may not be informed about existing nodes itself may trust, but the responding node does not.

When a node joins the network, it needs to perform a bootstrap procedure to obtain information about other nodes to fill its routing table. A node must send its bootstrap requests only to trusted nodes. As mentioned before, every node has its own view of trust, so the responses may contain all kinds of nodes.

When a node performs a node lookup for a GET or PUT action, it must only query nodes it trusts. During a lookup, newly encountered nodes must be evaluated and the node must determine if they are trustworthy before they are used. At the end of the lookup, the candidate nodes for the GET or PUT action must also be evaluated (if not done so already) so that the action is performed using trusted nodes only.

B. Consequences

Our concept enables a node to determine the trustworthiness of e.g. the retrieved results of a lookup. Every node can individually choose a minimum trust value threshold and decide on its own whether it regards an action or a result as valid. The nodes do not need to refer to general assumptions such as the amount of malicious nodes in the network to evaluate the correctness of actions and results.

As a drawback, our concept decreases the number of nodes that can be used for a node's actions. A certain minimum amount of trusted nodes is therefore essential. Furthermore, the availability of content items cannot be as easily guaranteed as in normal DHT networks: The assignment of content items to nodes is not unique any longer, but it differs due to the different trust values nodes assign to other nodes. It is therefore possible that content inside the network cannot be found by a node despite its existence. Possible reasons are the following:

- The content is only stored on nodes the requesting node does not trust.
- There are not enough trusted nodes on the route to the (trusted) nodes storing the content so that the lookup terminates prematurely.

IV. EVALUATION

In order to analyze to which extent these effects influence the ability of the nodes to use the network, we evaluate our concept in a Kademlia-based DHT using a simulated network of 1,000 nodes.

A. Assumptions

This paper is intended as a proof of concept to show that trust values can improve secure bootstrapping, lookup, PUT and GET in structured P2P networks. For this proof of concept, we assume the following:

- Nodes are either fully trusted or untrusted: We assume the existence of a certification authority. Nodes that possess a certificate are fully trusted, other nodes are untrusted.
- Nodes possessing a certificate are never malicious: We assume that the algorithm generating the trust values (here: certification) determines the trustworthiness correctly.

The first assumption requires a central entity which does not follow the peer-to-peer principle. The second assumption does not necessarily hold for real networks, as also nodes that are regarded as trustworthy may act maliciously. However, if the DHT would not work under these "perfect" conditions, it would not work in reality either.

The simulation scenario differentiates between nodes with and without a certificate: Nodes without a certificate ("No Cert" nodes) use all other nodes, whereas nodes with a certificate ("Cert" nodes) use only other nodes with a certificate for their actions. In this "binary trust" environment, we are able to demonstrate the worst case for the application of our concept. For small fractions of trusted nodes, the absolute number of them is below 100, which is rather low. However, we will demonstrate that even this small number of nodes is able to operate.

B. Choice of Kademlia

We choose the Kademlia algorithm because its routing process does not set hard restrictions on the next hop choice: Kademlia uses the XOR operation to calculate the distance between two IDs. During the routing process, a node uses a list of potential next hop nodes that is ordered by XOR distance with the closest nodes at the top of the list. When the first k nodes on the list do no longer change and have been queried for closer nodes, the lookup terminates and the action is performed on those nodes. This action can either be a PUT action or a GET action. During a PUT action, a node stores a content item on the configured amount of nodes. Using a GET action, a node tries to retrieve a content item.

C. Simulation Environment

Our simulation scenario consists of one large network in which only a subset of the nodes uses trust-based routing. This way, we can compare the performance of nodes applying and not applying our concept. In our simulation, malicious nodes perform a storage attack called "invalid data attack", which means they deliver randomly altered data if asked for a content item. We choose this attack type, because the Kademlia algorithm is rather robust against routing attacks: It has few restrictions regarding the choice of the next hop, so if a node propagates faulty routing information, the information might well be overridden by the responses of other nodes, only

 ISBN: 978-1-61208-173-1

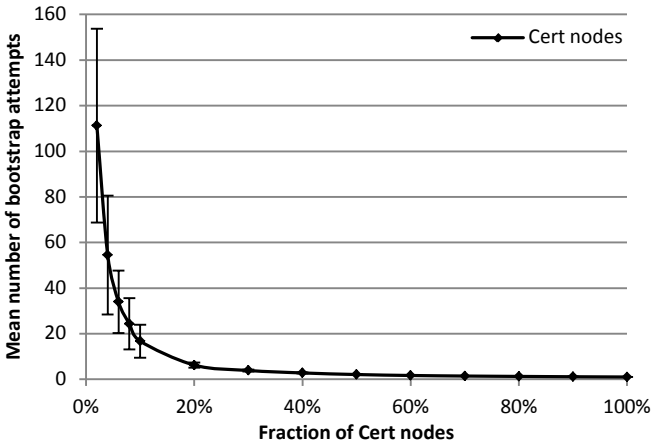


Figure 1. Mean number of Cert nodes' bootstrap attempts

causing higher delays. So storage attacks generally pose a greater threat and therefore show the influence of our concept more clearly.

For simulation, we use the OverSim [11] framework, which bases on OMNeT++ [12]. We vary the fraction of nodes possessing a certificate from 0% to 100% in steps of 10% and conduct additional simulations for fractions of 2% to 8% in steps of 2% for a detailed analysis. For each parameter combination, we perform 30 simulation runs using different seeds for the random number generator.

D. Results

All figures below show the arithmetic mean and the standard deviation of the results.

1) Bootstrap

At the beginning of each simulation run, the nodes perform bootstrap procedures. One node per second is inserted into the network and tries to bootstrap using a randomly chosen existing node. The first node is always a Cert node. As No Cert nodes use every kind of node to bootstrap, their first attempt to bootstrap will always succeed. Cert nodes use only other Cert nodes to bootstrap; their attempts to bootstrap may therefore fail if the randomly selected target node of the bootstrap request is not a Cert node. If the attempt fails, they pause for ten seconds and try to bootstrap again until they succeed.

Figure 1 shows that for very low fractions of Cert nodes, the mean number of bootstrap attempts of the Cert nodes is rather high. However, this value quickly decreases when the fraction of Cert nodes increases.

2) PUT and GET requests and malicious nodes

In order to demonstrate our concept's resilience against attacks, we introduce malicious nodes into the simulation network: We vary the fraction of No Cert nodes which are malicious from 0% to 100% in steps of 10%. As we assume that the trust values are correct, nodes with certificate cannot be malicious.

The measurement phase begins when all nodes have attempted to bootstrap at least once (in our case after 1,000 seconds). During the simulation, the nodes publish content with random identifiers and try to retrieve it. The Kademlia algorithm makes use of replication per definition. Baumgart and Mies argue in their S/Kademlia paper [13] that smaller replication factors than the original Kademlia's 20 are sufficient. So, in our simulation, we use S/Kademlia's default values: Content is stored on 4 nodes during a PUT action and a GET action tries to obtain the content from 4 nodes as well. The GET action is regarded as successful if at least 50% of the responses are identical. The nodes only try to retrieve content IDs that have been published previously.

Content can be published either by Cert or by No Cert nodes. The same also applies to retrieving content, so there are four possible combinations of PUT and GET actions: Cert PUT & Cert GET, Cert PUT & No Cert GET, No Cert PUT & Cert GET and No Cert PUT & No Cert GET.

The results show that our concept allows the Cert nodes to retrieve content published by other Cert nodes regardless of the fraction of malicious nodes. Figure 2 shows the success ratio of GET requests of Cert and No Cert nodes for content published by their respective kinds in dependence of the fraction of malicious nodes. In the absence of malicious nodes, the success ratio for retrieving content by No Cert nodes that was also published by No Cert nodes was the same as for the Cert/Cert

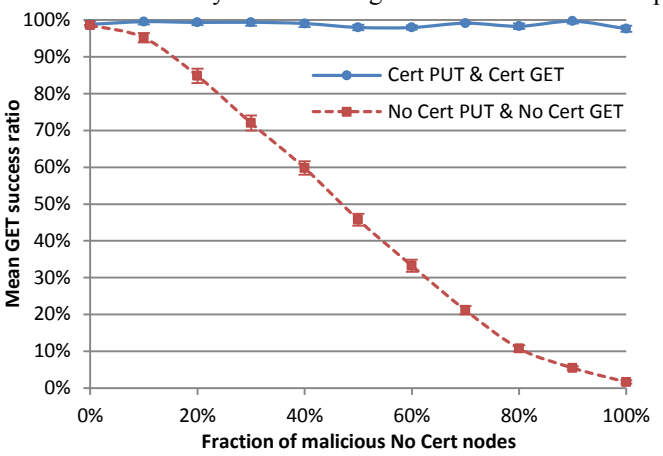


Figure 2. Success ratios of GET requests (10% Cert nodes)

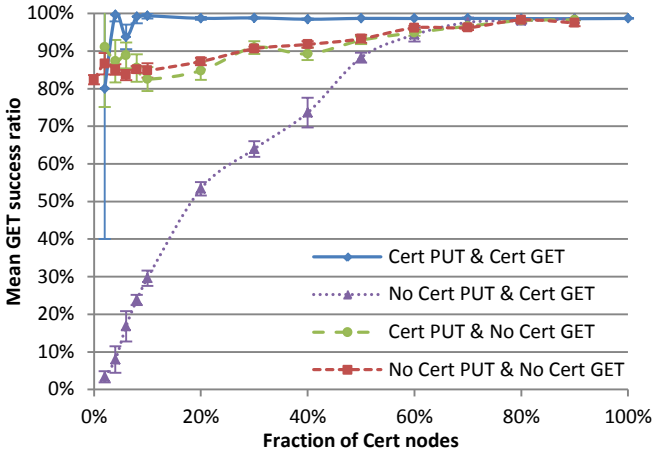


Figure 3. Success ratios of GET requests (20% malicious No Cert nodes)

 ISBN: 978-1-61208-173-1

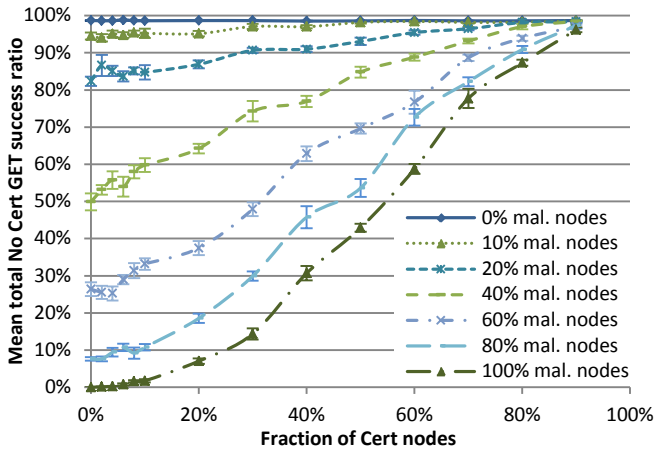


Figure 4. Total No Cert GET success ratio

case. The No Cert/No Cert success ratio decreases if the fraction of malicious nodes increases, whereas the Cert/Cert success ratio stays close to 100%. This demonstrates that the Cert nodes obtain a better performance than the other nodes: They are able to retrieve the desired content successfully despite the presence of malicious nodes. Furthermore, this result is independent of the fraction of malicious nodes.

Figure 3 shows the GET success ratios for all four possible combinations of PUT and GET requests. As an example, the fraction of malicious No Cert nodes in Figure 3 is 20%. The results show that even for small fractions of Cert nodes (2% to 4%, resulting in only 20 respectively 40 nodes), these are able to retrieve content published by other Cert nodes. Content stored by No Cert nodes can be retrieved increasingly successful by Cert nodes if the fraction of Cert nodes increases. However, the success ratio for this case is always higher than the fraction of Cert nodes: There are four replicas of each content item. If only one of them is stored on a Cert node, it can be found by another Cert node. The probability that at least one of the four nodes that store the item is a Cert node can be computed by using the following hypergeometric probability distribution formula:

$$1 - \frac{\binom{Total\ number\ of\ nodes\ -\ Number\ of\ Cert\ nodes}{4}}{\binom{Total\ number\ of\ nodes}{4}}$$

The correlation coefficient of the simulation results and the theoretical values is 0.995, which shows that the results are close to the theory.

In Figure 3, it is also visible that the No Cert nodes benefit from the presence of the Cert nodes, as the success ratio of No Cert GET requests increases when the fraction of Cert nodes increases.

Figure 4 reveals this tendency more clearly: It shows the total GET success ratio of No Cert nodes (for content that is stored on both types of nodes) for different fractions of malicious No Cert nodes. The No Cert nodes benefit from the presence of Cert nodes: The higher the fraction of Cert nodes is, the better the success ratios of the No Cert nodes are as well. This can also be seen in the results in Figure 2: Even with 100% malicious nodes, the success ratio of the No Cert/No Cert case is not 0%: No Cert nodes can still retrieve content items correctly if at least 50% of the responses originate from Cert nodes.

V. Conclusion and Outlook

We have presented a concept that uses trust values to enhance the security of lookups and PUT and GET actions in a structured P2P network. Nodes shall react to incoming requests as usual and use only trusted nodes when performing their own actions. We have shown that this concept works as intended using a Kademlia-based DHT: Despite the presence of malicious nodes, nodes applying our concept are able to continue operation normally as if no malicious nodes were present.

Further research is required to investigate the application of our concept to other DHT algorithms: Other algorithms have stricter requirements regarding the placement of other nodes into a routing table, for example. This may require an extension of our concept.

Our simplifying assumptions regarding certification and maliciousness do not hold in reality, so further research is required to analyze the effects of using a reputation system to generate the trust values. These values are typically not binary, so research regarding the minimum trust value threshold is also required.

Our simulations did not include message exchanges for the determination of the trust values and did not account for additional time required to validate the certificate. Further analyses of performance issues are therefore required.

VI. References

[1] Guido Urdaneta, Guillaume Pierre, and Maarten van Steen, "A Survey of DHT Security Techniques," *ACM Computing Surveys*, vol. 43, no. 2, pp. 8:1-8:49, Jan. 2011.

[2] John R. Douceur, "The Sybil Attack," in *IPTPS '02: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, Cambridge, MA, USA, 2002, pp. 251-260.

[3] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks," in *Proceedings of the 12th International Conference on World Wide Web*, Budapest, Hungary, 2003, pp. 640-651.

[4] Loubna Mekouar, *Reputation-based Trust Management in Peer-to-Peer File Sharing Systems*. Waterloo, Ontario, Canada: University of Waterloo, 2010.

[5] (2011, Sep.) BitTorrent. [Online]. http://www.bittorrent.com/

[6] (2011, Sep.) eMule Project. [Online]. http://www.emule-project.net/

[7] Sergio Marti and Hector Garcia-Molino, "Taxonomy of Trust: Categorizing P2P Reputation Systems," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 50, no. 4, pp. 472-484, Mar. 2006.

[8] Felix Gomez Marmol and Gregorio Martinez Perez, "State of the Art in Trust and Reputation Models in P2P Networks," in *Handbook of Peer-to-Peer Networking*, Xuemin Shen et al., Eds.: Springer, 2010, pp. 761-784.

[9] Frank Dabek, Ben Zhao, Peter Druschel, John Kubiatowicz, and Ion Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Berkeley, CA, USA, 2003, pp. 33-44.

[10] Petar Maymounkov and David Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *IPTPS: Revised Papers from the First International Workshop on Peer-to-Peer Systems*,

 ISBN: 978-1-61208-173-1

Cambridge, MA, USA, 2002, pp. 53-65.

[11] Ingmar Baumgart, Bernhard Heep, and Stephan Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007*, Anchorage, AK, USA, 2007, pp. 79-84.

[12] (2011, Sep.) OMNeT++. [Online]. http://www.omnetpp.org/

[13] Ingmar Baumgart and Sebastian Mies, "S/Kademlia: A practicable approach towards secure key-based routing," in *Proceedings of the 13th International Conference on Parallel and Distributed Systems*, Hsinchu, Taiwan, 2007, pp. 1-8.

 ISBN: 978-1-61208-173-1

New Heuristics for Node and Flow Detection in eDonkey-based Services

Rafael A. Rodríguez-Gómez, Gabriel Maciá-Fernández, Pedro García-Teodoro
Department of Signal Theory, Telematics and Communications,
CITIC - ETSIIT, University of Granada
Granada, Spain
rodgom@ugr.es, gmacia@ugr.es, pgteodor@ugr.es

***Abstract*—The development and use of applications based on peer-to-peer (P2P) networks have exponentially grown in the last years. In fact, the traffic volume generated by these applications supposes almost the 80% of all the network bandwith nowadays. For this reason, the interest of Internet Service Providers (ISPs) for classifying this large amount of traffic has also grown in a considerable manner. In this context, the present paper describes two detection algorithms for eDonkey services. The first one is aimed to detect eDonkey flows. It is based on the hypothesis that clients that begin connections are in charge of sending the information. The second algorithm has been developed to detect nodes that generate eDonkey traffic. It is based on the hypothesis that the up-rate of these nodes follows a constant pattern along the time. Both detection algorithms have been proved in three different groups of network traces. As a result, our detection hypothesis is checked. Additionally, the experiments carried out show that the proposed algorithms have a high classification rate and a low false positive rate.**

Keywords-Traffic classification; flow detection; node detection; P2P; eDonkey.

I. INTRODUCTION

The development and use of applications based on P2P networks have exponentially grown in the last years. Nowadays, several examples can be found: eMule or uTorrent, as file sharing applications, Skype, as voice over IP application, and Spotify, as audio flow sharing.

Traffic generated by P2P applications consumes around 80% of all the network bandwith [1]. This enormous use of available bandwith requires the allocation of a considerable amount of resources to guarantee the quality of the provided services.

The ability of classifying the P2P traffic is a key issue to ISPs, as they are forced to increase the maintenance operations due to this excessive growth in the use of P2P services [2].

Traffic classification methods can be divided in three approaches: *(i)* based on port, *(ii)* based on packet content, and *(iii)* based on flow features. Current P2P applications can receive connections in any port and encrypt the content of its messages. This feature makes the classifications based on port and packet content difficult.

The classification methods suggested in this paper are based on flow features, and are used to detect eDonkey traffic, a communication protocol of P2P networks mainly used in file sharing applications such as eMule or aMule.

Two new detection heuristics are proposed: *node detection based on up-rate*, and *flow detection based on inversion of download direction*.

Node detection based on up-rate relies on the next two assumptions: *(i)* users of eDonkey-based applications limit the up-rate, and *(ii)* the up-rate is approximately constant around this limit established by the user.

The proposed flow detection method assumes that those nodes with eDonkey traffic that, under certain conditions, establish a connection will send the majority of the information. This behavior is radically opposite to the common client-server paradigm, in which clients establish connections and servers send the required information.

The rest of the paper is organized as follow: In Section II, some relevant papers in the field of traffic classification are presented, the novelty of the present contribution being remarked. In Section III, some specially relevant concepts regarding eDonkey are exposed. The proposed heuristics to detect nodes and flows in eDonkey-based services are detailed in Section IV, presenting Section V the experimental framework considered. In Section VI, the experimental results obtained are shown and discussed. Finally, in Section VII, principal conclusions of this work are drawn.

II. RELATED WORK

In the specialized literature, three kinds of classification methods can be found: *(i)* based on well-known ports, *(ii)* based on packet content, and *(iii)* based on flow features. T. Karagiannis et al. [3] assure that classification methods based on well-known ports are not valid to detect P2P traffic nowadays. On the other hand, methods based on packet content imply legal issues related to privacy and thus, their field of application is enormously reduced.

There exists a huge amount of papers proposing classification methods based on flow features analysis. A relevant example is BLINC [4], a classification tool based on the assumption that a host can be associated with an application responsible to generate the majority of its traffic volume. In the same manner as BLINC, we also propose here a node classification method.

 ISBN: 978-1-61208-173-1

Another possibility is to classify only a subgroup of existing protocols. This is the most common approximation in the field of P2P classification. An example of this we present the work of T. Karagiannis et al. [3], this is the first work that tries to classify encrypted P2P traffic in random ports without inspecting packet payload. The classification is based on connection patterns of P2P networks. Our classification methods also detect P2P encrypted traffic in random ports.

Xu et al. [5] propose a method to identify P2P traffic based on the data transfer behavior of P2P applications. The authors assure that the downloaded data by a node will be subsequently uploaded to another node in the network. Thus, flows that download and upload the same data blocks will be identified as P2P flows. The heuristics proposed in the present paper are also based on the data transfer behavior of P2P applications. As we will show in Section IV there exist several differences with respect to the work of Xu et al.

Finally, there exist some targeted at classifying a single protocol. As an example, Bonfiglio et al. [6] detect real time Skype traffic exploiting the randomness introduced at the bit level by the encryption process to detect Skypes fingerprint from the packet framing structure. In our case, the classification also aims at classifying only the eDonkey protocol.

III. General concepts of the eDonkey protocol

The eDonkey protocol was designed to communicate nodes belonging to a hybrid P2P network composed by server and client nodes. Servers are in charge of giving access to the network and managing the information distribution in a similar manner to a dictionary, *i.e.*, they store the correspondence between resources and the nodes sharing them. On the other hand, clients are the nodes that share data, and they store the resources of the network.

In the following, a brief description of eDonkey-based communications, with special interest in the aspects used in the present paper, is provided. To do this in a structured manner, we explain the process followed by a client to download a resource from the network.

A. Client to server connection

The first step to download a resource from the network is to connect to an eDonkey server. This connection can be divided in two steps: *(i)* TCP connection from client to server, and *(ii)* server challenge. To carry out this challenge the server also tries to establish a TCP connection with the client. This connection allows to discover if the client is able to receive connections from other clients of the eDonkey network. If the challenge is passed, the client will receive an identification called *high ID*; otherwise, it will receive a *low ID*. Thus, a high ID client can directly receive connections from other clients in the network, while a low ID client can not.

Once a client has accessed to an eDonkey server, it can look for resources by describing them with certain key words. The requested server will respond with a list of related resources. Subsequently the client requests one or several of them to be downloaded, and finally he will receive a response with a list of clients that share the requested resource.

B. Client to client connection

It is necessary to carry out client to client connections in order to download any resource from the network. However, a client with a low ID can not accept connections. To solve this, there exists a procedure in the eDonkey protocol called `callback`. If a client wants to connect to a low ID node, it has to send a `callback` message to the corresponding eDonkey server. This server will resend, through a previously established connection, the `callback` to the low ID node who will begin a new connection. This mechanism does not solve the case in which both nodes are low ID because none of them are able to accept external connections.

C. Downloading of a shared file

Downloading a shared file in eDonkey protocol consist of two steps: *(i)* entering the reception queue, and *(ii)* starting the download. Firstly, a client A requests a file to a client B. B answers by sending the position in which this request is stored in its queue. Secondly, when this request reaches a position in the queue to be served, B sends a message to A indicating this new state so that the download process starts.

The most common situation in a download process is that a request from client A has to wait a considerable time to be served. Thus, after a fixed time (around 40 seconds in our experiments) client B closes the connection with client A. B will establish a new connection when the request from A is able to be served. As detailed in Section IV, this behavior will be used in our flow detection method.

IV. Detection heuristics

Two detection methods of eDonkey protocol are now proposed: node detection based on up-rate and flow detection based on inversion of download direction. These methods can work together: firstly, a node detection indicates those nodes that generate eDonkey traffic and, subsequently flow detection determines specifically eDonkey file sharing flows of the detected node.

The detection heuristics used in both methods are explained in the following.

A. eDonkey flows detection

The first heuristic is aimed to detect eDonkey flows based on the hypothesis that clients that begin connections also send the majority of the data over that connection. As explained in Section III, this is the most common situation in any download process of a shared file in eDonkey.

 ISBN: 978-1-61208-173-1

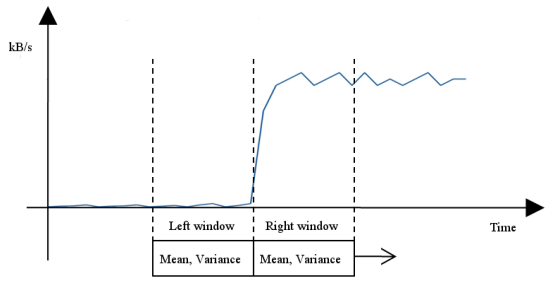


Figure 1. Calculation of Kullback-Leibler distance in time.

In client-server applications, servers usually send the majority of the data after a connection is started by a client. This behavior is reverse to that of eDonkey protocol, and this is the reason because we propose to use the next hypothesis to detect eDonkey flows:

Hypothesis 1. *eDonkey flows are those in which clients who begin the connection send substantially more information than they receive.*

Note that this heuristic is only valid for file sharing flows, and not in the case of signaling flows. File sharing flows are specially relevant because they are the principal responsible of congesting the bandwidth of a network.

The proposed detection method has been developed to be executed over offline traces. eDonkey flows are those where the number of bytes sent by clients (who begin the connection) are greater than the number of bytes received plus a threshold, $Thres_B$. This threshold is experimentally determined by means of a study of the size distributions in file sharing flows of the eDonkey protocol.

The selection of the threshold to be used in this method is not critical, as we argue in Section VI, because the difference between sent and received bytes in file sharing flows is really noticeable.

B. eDonkey nodes detection

Here we expose a method to detect nodes which are generators of eDonkey traffic.

The proposed method is based on the assumption that users of P2P file sharing applications usually limit their up-rate. This is due to the fact that these applications consume the majority of the upload bandwidth, and consequently users that use them without a limitation in the up-rate suffer a decrease in the quality of their normal Web browsing.

The mentioned constant rate supposes a differentiating feature allowing the detection of these nodes. In conclusion, the proposed hypothesis is defined as follow:

Hypothesis 2. *Nodes generators of eDonkey traffic are those whose up-rate is quasi-constant.*

To apply the previous hypothesis it is necessary to detect a quasi-constant level in the up-rate of a node. We take as a reference the work of J. Ramirez et al. [7], in which they use the Kullback-Leibler (KL) divergence to detect voice activity in audio signals. This detection is aimed to determine the specific instant at which an evaluated audio signal changes from only noise to contain human speech, or vice versa. Authors use KL divergence to detect changes in mean and variance of audio signals. In our case the KL divergence is used to detect the absence of significant changes in the up-rate of a node (quasi-constant up-rate).

The KL divergence can be described as an indicator of the similarity between two probability distributions. In the case of two Gaussian distributions p_L and p_R is defined as (taken from [7]):

$$H(p_L||p_R) = \frac{1}{2}[log(\frac{\sigma_R^2}{\sigma_L^2}) - 1 + \frac{\sigma_L^2}{\sigma_R^2} + \frac{(\mu_L - \mu_R)^2}{\sigma_R^2}] \quad (1)$$

where σ_R y σ_L represent standard deviations of p_R y p_L, and μ_R y μ_L their means.

The KL divergence is not symmetric, and thus we will use the KL "distance" $\rho_{I,D} = H(p_R||p_L) + H(p_L||p_R)$. In the case of Gaussian distributions it is defined as (taken from [7]):

$$\rho_{L,R} = \frac{1}{2}\left[\frac{\sigma_L^2}{\sigma_R^2} + \frac{\sigma_R^2}{\sigma_L^2} - 2 + (\mu_L - \mu_R)^2(\frac{1}{\sigma_L^2} + \frac{1}{\sigma_R^2})\right] \quad (2)$$

The proposed detection method can be described as follows (Algorithm 1). Firstly, up-rate values are calculated every t seconds. A median filter [8] of length N is applied to the resulting values. This filter takes a window of length N and sorts some values extracting the central one.

Secondly, the means and variances of the filter values are calculated by means of two consecutive windows (v_I y v_D) of length N (see Figure 1). The Gaussian distributions represented by these means and variances should be very similar if there exists a quasi-constant up-rate. Therefore, the KL "distance" (Equation (2)) computed from these means and variances should be minor than $Thres_{KL}$ to represent a constant up-rate. If a constant rate is detected, our method will classify the corresponding node as an eDonkey traffic generator.

V. EXPERIMENTAL FRAMEWORK DESCRIPTION

Three groups of network traces have been used to carry out the experimentation related to the proposed methods of eDonkey traffic detection. In the following, the principal features of these traces are exposed.

- *Controlled environment traces (CE).* The traffic generated by 5 users during 72 hours were collected. In this period, they used aMule version 2.2.6 and shared the same 10 files. The eDonkey server to which they connected was *se-Master Server 1*. They limited the up-rate of aMule to 30kB/s. All of them used their PCs

 ISBN: 978-1-61208-173-1

Algorithm 1 Node detection

```
for node = 0 while node < num_nodes do
  for i = 0 while i < len(up_rate_node) do
    rate_filt_node ⇐ median_filt(up_rate_node, N)
    v_I ⇐ rate_filt_node[i : i + N]
    v_D ⇐ rate_filt_node[i + N + 1 : i + 2N + 1]
    ρ_{L,R}[i] ⇐ Equation(2)
    if ρ_{L,R}[i] < Thres_KL AND μ_R different to 0 then
      return eDonkey node
    else
      return Not eDonkey node
    end if
    i ⇐ i + 1
  end for
  node ⇐ node + 1
end for
```

and Internet connection without restrictions. Every user generated around 19,000 connections of the eDonkey protocol and more than 7,000 of other protocols like DNS, HTTP, SSH and SMTP.

- *HTTP server traces (HS).* This collection of traces represents the traffic generated by an HTTP server from an European University during 7 days. The server is Apache version 2.2.0 and receives a mean of 8,971 connections per day.
- *University trunk traces (UT).* All the traffic of a trunk from a Middle East University during 48 hours composes these traces. There are around 73,000 IPs, 300 millions of packets transmitted, and the most common protocols that appear are: Bittorrent, HTTP, DNS, SSL, and FTP. After the analysis of the entire database by means of a packet inspector application (`OpenDPI` [9]), very few eDonkey packets have been detected. This is due to the fact that the P2P file sharing applications used in this Middle East University are based on Bittorrent protocol instead of on eDonkey.

VI. Experimental results

The experimentation carried out for both detection methods is focused on a double aim: *(i)* To study if the presented hypotheses are valid through the evaluation of detection and false negative rates obtained in CE traces, and *(ii)* to analyze if these hypotheses present low false positive rates for other protocols through HS and UT traces.

Following, the obtained results for flow and node detection methods are analyzed.

A. eDonkey flows detection

First of all, to execute the eDonkey flow detection method it is necessary to determine the value of threshold the $Thres_B$, which is the maximum difference allowed between the number of the sent and received bytes to consider the evaluated node as not generator of eDonkey traffic. The results of a study of the detection and false positive rates in function of $Thres_B$ in the three groups of traces indicate that there exists a wide range (between 5 and 25kB) to select this threshold in which the success of the method is very similar. Specifically, the threshold selected is 10kB.

Table I
Detection rate of flow detection method in UT traces.

	Detection rate	Detected flows	Total flows
BitTorrent	0.0256	854	33,304
HTTP	0.01691	50,795	3,003,161
FTP	0.01423	35	2,460
SSL	0.01244	2,808	225,685
IRC	0.00213	7	3,281
Oscar	0.00079	2	2,528
DNS	0.00001	8	1,508,413
Mail_POP	0.00000	0	5,208
All	0.01139	54,509	4,784,040

In the present experiment another assumption has been applied. An eDonkey node download or upload files from or to more than one peer simultaneously. Thus, we can suppose that eDonkey flows coincide temporally with other eDonkey flows.

CE traces contain 37,089 file sharing flows of eDonkey protocol. 28,016 of them have been detected, which implies a detection rate of 77.53%. None of the flows belonging to other protocols different to eDonkey have been detected (0% of false positives). Finally, there exists a considerable false negative rate: 22.47%. This is mainly due to two factors:

1) *Low ID in some of the ends.* Nodes with low ID can not accept connections from other peers of eDonkey network and, for this reason, they always begin the connections, independently of being server or receiver of the information. This situation is not valid in the proposed detection hypothesis.
2) *Service without an intermediate close of connection.* A request of a resource could be served without an intermediate close of the initial connection (explained in Section III). In this case, the number of bytes received are greater than the sent.

Therefore, the proposed detection method is able to detect file sharing flows of eDonkey protocol between high ID nodes and with an intermediate close of connection (the most common situation, as we mentioned in Section III).

The results obtained from HS traces show that none of the 62,798 HTTP flows have been detected as an eDonkey flow, which represents a 0% of false positive rate.

Flows from UT have been labeled through the execution of a modification of `OpenDPI`, an application that performs deep packet inspection [9]. We take this labels as ground truth and compare the results of our detection method with it obtaining the results shown in Table I.

The principal contribution in false positve rate corresponds to bitTorrent protocol. This protocol is also used in

 ISBN: 978-1-61208-173-1

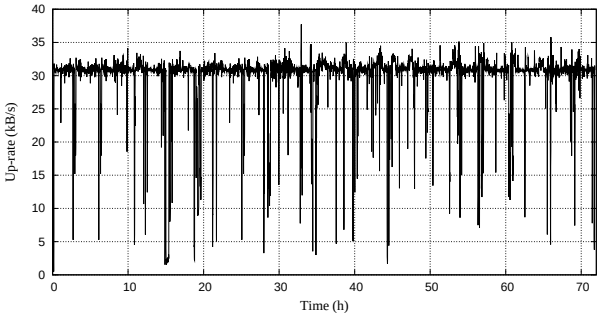


Figure 2. Typical up-rate of a monitored node from CE traces.

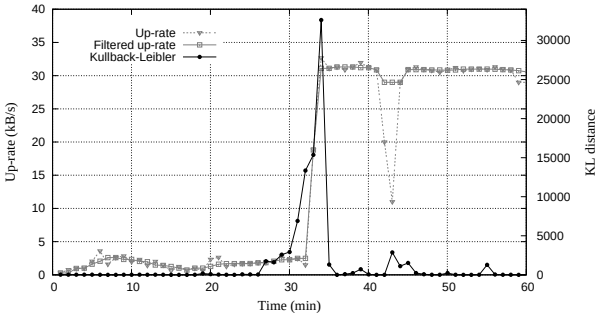


Figure 3. Evolution of Kullback-Leibler distance in one hour of CE traces.

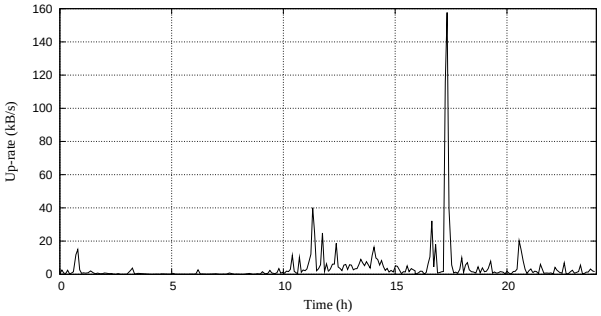


Figure 4. Up-rate of HTTP server (HS trace) during 24 hours.

P2P file sharing applications. However, a principal difference with eDonkey is that bitTorrent flows are bidirectional, so both peers send information to the other in the same flow and it can occur that clients that begin a connection send more than they receive. In the same manner, FTP flows were detected because clients frequently send more data than servers do.

Finally, it is remarkable the false positive rate associated to the HTTP protocol, because none of the flows in HS traces were detected while UT traces represent a 1.69%. After a detailed study we conclude that these false positives are mainly caused by three factors: *(i)* high length of cookie and URL in HTTP GET messages, *(ii)* very short server responses, 304 (Not Modified), and *(iii)* HTTP POST sending a big amount of data. This is the case of using the Web 2.0 philosophy, but it has no significant influence in our detection method nowadays.

B. eDonkey nodes detection

The second detection hypothesis is validated with CE traces, as we can see in Figure 2, in which the up-rate of a monitored node is shown. During the 72 hours of traces there exists a quasi-constant behavior around 30kB/s, the limit fixed in the experiment.

The up-rate suffers several descents of low duration. These descents correspond to instants at which the monitored client stops sending data to a peer in order to begin the transmission with another one. The churn [10] (independent arrival and departure by peers) is extremely high in P2P networks, so the mentioned situation is frequent.

In CE traces, a 38.7% of flows belong to other protocols different to eDonkey, the most common being DNS, HTTP, SSH and SMTP. This is represented in Figure 2 as instants at which the up-rate limit fixed in the experiment is exceeded. So, these instants are due to additional network activity to eDonkey traffic.

In Figure 3, the up-rate of a user in CE traces during the first hour of the capture is shown. The up-rate presents two well defined sections: near to zero, and around 30kB/s. During the first 30 minutes the up-rate is near to zero because only a few nodes know the existence of the monitored peer. After that, we can observe the mentioned constant behavior around 30kB/s. The up-rate filtered is also presented in the same figure, which allows us to appreciate the suppression of values that extremely deviate from the expected ones.

Finally, the KL distance is also presented in this figure and is divided in three parts: two sections near zero, separated by a clear increase of the KL distance. KL distances near zero represent the constant up-rate sections, and a relative maximum of the KL distance indicates a change in the up-rate distribution. The second section of the near zero KL distance will be identified as generated by an eDonkey node because it presents a constant up-rate different to zero.

Additionally, a study using all the network traces has been carried out to select a value for the threshold $Thres_{KL}$, and a length of the window, N, for the median filter and the KL distance. There exists a wide range of $Thres_{KL}$ values that present a high detection rate in CE traces and a low false positive one in UT and HS traces. This range is $[10^3, 10^4]$ and the value for $Thres_{KL}$ finally selected was 8,000. On the other hand, the selected value for N was 5 minutes because this is the minimum burst of constant traffic to be detected as generated by an eDonkey node. Bursts of traffic from eDonkey nodes with a duration less than 5 minutes are not interesting in this work, because we try to detect nodes with a substantial consumption of bandwidth network.

The execution of nodes detection method in CE traces indicates that these nodes are classified as generators of eDonkey traffic during a 86.10% of the monitored time. The rest of the time is mainly constituted of instants at which clients stop to share with certain client and consequently

 ISBN: 978-1-61208-173-1

their up-rates will not be constant.

Finally, the up-rate of HTTP server has also been analyzed. As we see in Figure 4, the up-rate do not present a constant behavior, so this node was classified as eDonkey generator only a 1.687% of the total monitored time (168 hours in total).

VII. Conclusions

In this paper, two methods to detect eDonkey traffic without inspecting packet payload have been proposed: *(i)* an eDonkey flow detection and, *(ii)* an eDonkey node detection.

The experimental results obtained allow us to conclude that the proposed detection hypotheses are acceptable in the case of eDonkey protocol. Moreover, both detection methods present a high classification rate and a low number of false positives. Finally, we specify that the eDonkey flow detection process is valid for file sharing flows from eDonkey nodes with high ID.

Additionally, in a near future we plan to raise two work lines:

- To combine the information obtained by both methods in order to increase the detection rate and to reduce false positive rate.
- To explore the possibility of detecting other P2P protocols used in file sharing applications, *e.g*, Kademlia or BitTorrent, through the execution of the second method (eDonkey flow detection). We think that this method could detect protocols used in P2P file sharing applications because they saturate the up-rate of users and that implies the necessity of a limitation.

Acknowledgement

This work has been partially supported by Spanish MICINN under project TEC2008-06663-C03-02.

References

[1] A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A Survey on Internet Traffic Identification," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 37–52, Aug. 2009.

[2] A. Feldmann, "A possibility for ISP and P2P collaboration," in *Broadband Communications, Networks and Systems, 2008. BROADNETS 2008. 5th International Conference on*, Sep. 2008, p. 239.

[3] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of P2P traffic," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ser. IMC '04. New York, NY, USA: ACM, 2004, pp. 121–134.

[4] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: multilevel traffic classification in the dark," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '05, vol. 35, no. 4. New York, NY, USA: ACM, 2005, pp. 229–240.

[5] K. Xu, M. Zhang, M. Ye, D. M. Chiu, and J. Wu, "Identify P2P traffic by inspecting data transfer behavior," *Computer Communications*, vol. 33, no. 10, pp. 1141–1150, Jun. 2010.

[6] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing skype traffic: when randomness plays with you," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '07. New York, NY, USA: ACM, 2007, pp. 37–48.

[7] J. Ramirez, J. Segura, C. Benitez, A. de la Torre, and A. Rubio, "A new kullback-leibler vad for speech recognition in noise," *Signal Processing Letters, IEEE*, vol. 11, no. 2, pp. 266–269, Feb. 2004.

[8] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proceedings of the IEEE*, vol. 78, no. 4, pp. 678–689, Apr. 1990.

[9] Opendpi. http://www.opendpi.org/ [Last accessed in September 26, 2011].

[10] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ser. IMC '06, 2006, pp. 189–202.

 ISBN: 978-1-61208-173-1