Towards a Communication Framework based on Balanced Message Flow Distribution

Mugurel Ionut Andreica, Iosif Charles Legrand, Nicolae Tapus

To cite this version:

Mugurel Ionut Andreica, Iosif Charles Legrand, Nicolae Tapus. Towards a Communication Framework based on Balanced Message Flow Distribution. IEEE International Conference on "Computer as a Tool" (EUROCON), Sep 2007, Warsaw, Poland. pp.495-500, 10.1109/EURCON.2007.4400619 . hal-00873788

HAL Id: hal-00873788
https://hal.science/hal-00873788

Submitted on 16 Oct 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Towards a Communication Framework based on Balanced Message Flow Distribution

Mugurel Ionut Andreica[*], Iosif Charles Legrand[†] and Nicolae Tapus[*]
[*] Polytechnic University of Bucharest/Department of Computer Science, Bucharest, Romania
[†] California Institute of Technology/Division of Physics, Mathematics and Astronomy, Pasadena, CA, USA

Abstract— **Applications running in Grids and other large scale distributed systems have several communication requirements which are not easy to satisfy by the existing communication middleware, like high throughput data transfer, reliable point-to-point messaging and unreliable multicast streaming. In this paper we propose a scalable, fault tolerant communication framework based on the peer-to-peer paradigm. The framework's design is oriented towards high volume point-to-point and multicast data delivery. The proposed system uses balanced message flow distribution over multiple paths in order to achieve high throughput and the lexicographic distance between the peer's names in order to guarantee that the message flow converges to destination.**

I. Introduction

Grids and large scale distributed systems are execution environments for many data-intensive applications. Simulations or scientific instruments and sensors generating large amounts of information, transmission of audio and video streams from a provider to multiple consumers, scientific computations performed upon large quantities of data, video conferences, Grid monitoring tools sending data to storage and processing applications, instant messaging and massively multiplayer online games are just a few situations where high throughput unicast and multicast communication capabilities are required. There are currently several approaches for offering this kind of communication services.

One possibility is based on establishing direct TCP connections between the sender and the receiver for the case of unicast messages. Multicast 1-to-N communication requirements are satisfied either by using IP multicast or by sending N unicast messages to each of the destinations. This approach has the advantage of being very simple to use, but has several drawbacks. First, it may not be possible to establish direct TCP connections between machines located behind NATs (Network Address Translators) or firewalls. Second, there aren't too many multicast-capable routers in the Internet, so IP multicast cannot be used on a large scale.

A second approach is based on the use of central servers. If two client applications need to communicate with each other, they do this by connecting to one of the servers, which will route all the messages exchanged between them. This solution is suitable for applications running behind NATs and firewalls. Its only drawbacks are that the servers represent central points of failure and that they may become communication bottlenecks in the case of many clients.

Another approach is given by the peer-to-peer paradigm. Peers form a decentralized overlay network similar to an undirected graph, where the edges are represented by direct connections between peers. The peer-to-peer model could use efficiently both the available bandwidth and the computational power. However, such a decentralized approach brings new challenges regarding peer lookup and message routing. Many peer-to-peer communication architectures have been proposed, but they focus on different communication issues, not necessarily on achieving high throughput data transfer. Some of them consider network latency a major factor, others consider the number of hops in the overlay network, while another part of them is more interested in the amount of consumed resources.

The novelty of our communication framework is represented by its focus on high throughput message routing, at the expense of other network metrics. We will consider the process of routing messages in the context of a flow of messages from a source to one or several destinations. Both the source and the destinations have an address representing coordinates in a metric space associated with the overlay network. In order to make sure that any message will eventually reach its destination, each message will be routed on a path consisting of intermediate peers in such a way that the distance towards the destination, computed in the associated metric space, decreases constantly. Naturally, there could exist many such paths. The communication framework distributes the message flow over all these paths in such a way that the total size of the messages forwarded on any path is proportional to the available bandwidth of the path. The offered communication services are reliable point-to-point and unreliable multicast messaging.

The rest of this paper is organized as follows. In Section II we present related work. In Sections III, IV, V and VI we provide an extensive presentation of the concepts and algorithms used by the proposed communication framework. In Section VII we present experimental results and in Section VIII we conclude and present future work.

II. Related Work

Among systems trying to achieve high throughput data transfer are the peer-to-peer file sharing applications like Bittorent[1] or Kazaa[2]. Bittorent uses swarming content delivery. All the peers trying to download the same file are part of

[1] http://www.bittorent.com

© 20xx IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

the same swarm. In order to download a file, a peer has to contact a central tracker. After that it connects to several randomly chosen peers in the swarm. Files are decomposed into smaller pieces and a client may start sharing the downloaded pieces before downloading the entire file. Bittorent uses a tit for tat scheme which encourages peers to upload files, not just download them. Kazaa's peers are differentiated into clients and super-peers. Super-peers are chosen automatically based on their computational power, storage capacity and bandwidth. Clients connect to the closest super-peer, using it for file search and download.

DHTs (Distributed Hash Tables) are decentralized distributed systems that partition the keys inserted into the hash table among the participating nodes. They usually form a structured overlay network in which each communicating node is connected to a small number of other nodes. Any DHT could be easily turned into a system offering communication services. Ref. [1] presents CAN (Content Addressable Network), a DHT which uses a virtual coordinate space represented by a d-dimensional torus. Each node has a unique ID in this space. Considering that there are N nodes in the system, a message reaches its destination after $O(d \cdot N^{1/d})$ hops. The amount of routing information stored by each node is $O(2 \cdot d)$. In [2], the authors introduce Chord. The identifiers of the participating nodes are 160-bit numbers ordered on a circle modulo 2^{160}. Each node x is connected to its numerical successor and stores a list of O(log N) "fingers" to other nodes. The i^{th} element of this list contains the identifier of the peer which succeeds x by at least 2^{i-1}. A node can reach any other node in O(log N) hops using this list. Pastry, presented in [2], uses prefix-based routing in order to build an overlay network and route messages within this network. Each node has a 128-bit identifier in a circular space. The identifiers and the keys are interpreted as base B numbers. Any message reaches its destination after $O(\log_B N)$ hops. The amount of routing information stored by each node is $O((B-1) \cdot \log_B N)$. DHTs are very efficient because they provide a small number of hops with only a small amount of routing information. But they are not the most suitable choice for high volume transmission of data. Moreover, most of them require connectivity capabilities between any pair of nodes, so they are less likely to deal appropriately with machines located behind NATs and firewalls.

Most DHTs support only point-to-point communication, but multicast extensions have been built upon them. Ref. [4] presents Scribe, which uses a single multicast tree built over Pastry. Ref. [5] introduces Splitstream, in which multicast data transfer is based on multiple Scribe trees. Another multicast data delivery system, based on multiple, concurrently used, balanced trees, is presented in [6].

III. Name Space and Topology

We consider that the communication network is composed of one or more communication nodes (or cells). Each cell may have zero or more names. A name is simply a string of characters which must not contain the special character '*'. The names are coordinates in a metric space where the distance between two names is the lexicographic distance. The names assigned to a cell are important in establishing the topology of the overlay network. For each name it has, the cell tries to establish direct connections to the K closest cells having a name lexicographically smaller and the K closest cells having a name lexicographically larger. Because some of these cells might not be directly accessible, because of NATs or firewalls, the K closest directly accessible cells are chosen. A cell also tries to connect to M other cells, chosen randomly from the set of known cells, and to P of the known cells which are very distant in the overlay network, in order to keep a well-connected network. Fig. 1 shows a possible network topology for 8 cells, K=2, M=1 and P=0.

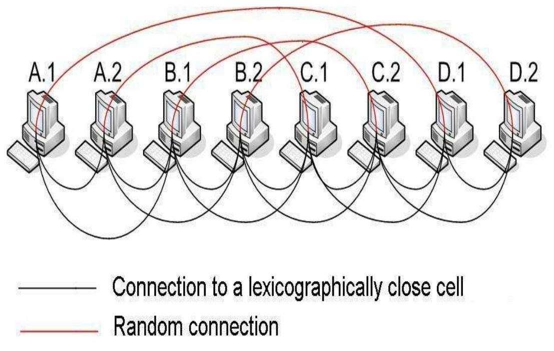


Fig. 1. A possible overlay network topology for K=2, M=1 and P=0.

The applications using a given cell for communication purposes may add a new name or remove one of the old names. They can do this based on their current communication interests. If a group of applications expects to communicate intensely, they may add names which are close to each other in the metric space, thus bringing the corresponding cells closer in the overlay network. These names may also be added in order to reflect geographic proximity and make the cells establish low latency connections, by using the reversed DNS name or a hierarchically structured name (for instance, "country.city.institution.machine_name").

IV. Routing Information

Routing information is exchanged between cells by sending routing update messages at regular intervals. These messages are sent to all the cells located at a distance of at most R hops in the overlay network, where R is the

[2] http://www.kazaa.com

© 20xx IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

neighborhood radius. There are two types of routing update messages: updates which advertise the state of the direct connections and updates advertising other known peers in the network.

The first type of messages contains information about the measured available bandwidth of each direct connection. This information is used by each cell in order to build a detailed view of their neighborhood.

The second type of routing update message contains information about other cells in the overlay network that the sending cell knows about. For each known peer, also called destination peer, a set of cells from the neighborhood of the sending cell is sent. The peers in this set are candidate nodes when choosing an intermediate cell in the routing process. For each peer in the set an estimation of the overall available bandwidth of all the paths towards the destination peer is also sent. Based on this information, the receiving cell computes the available bandwidth on all the paths from the sending cell to the advertised cell which do not contain any other node in the neighborhood of the receiving cell. The names of the advertised cells and the computed information are stored in a trie.

In order to avoid the propagation of every piece of information to every node in the system, each cell allows only a maximum number of T entries in the trie. Once this threshold is reached, either new entries or already existing entries should be discarded. The algorithm we used allows, in fact, for a maximum of $f \cdot T$ ($f>1$) entries in the trie and periodically selects only the T most significant such entries, deleting the others. Newer entries, entries which have a shorter common prefix with all other entries at the moment of insertion and entries which are frequently used in the routing process are preferred.

V. The Routing Algorithm

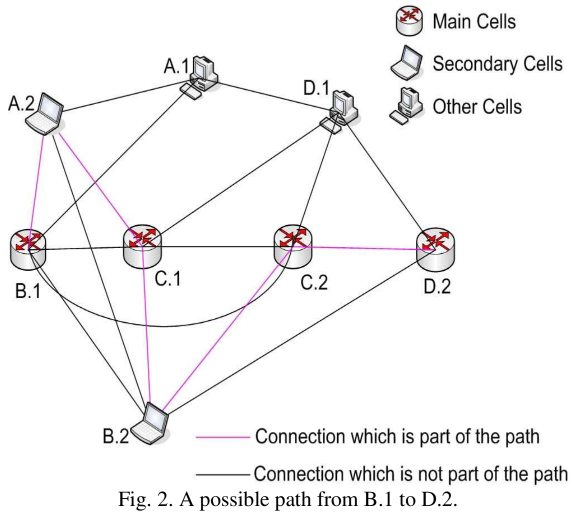
A. *Routing Unicast Messages*

Routing unicast messages is a two step process. First, the name of the destination is looked up in the trie. Here one of the peers which are located in the neighborhood of the cell routing the message and which advertised paths towards the destination is chosen. This choice is non-deterministic. A peer i is chosen as an intermediate cell according to the formula

$$P_i = AB_i / SAB \ . \qquad (1)$$

P_i is the probability to choose the i^{th} peer, AB_i is the advertised available bandwidth of peer i towards the destination and SAB is the sum of the available bandwidths of all the candidate peers. Only peers having a name which is lexicographically closer to the destination than any of the names of the cell making the routing decision are considered as candidate peers, in order to make sure that the lexicographic distance towards the destination decreases. In case the name is not found in the trie, the name which is lexicographically closest to the destination is chosen. This operation can be performed very efficiently using the trie data structure.

The second step consists of choosing a path towards the intermediate peer selected in the first step. In order to make this decision, the knowledge about a cell's neighborhood is interpreted as a directed graph. The vertices of the graph are represented by the cells of the network and the edges of the graph are represented by connections between these cells. Each edge has an associated capacity equal to the measured available bandwidth of that edge. The graph is directed because the measured available bandwidth might differ when measured from each of the two end points of each edge. Considering, in turns, each cell in the neighborhood as a sink and the current cell as a source, the Edmond-Karp algorithm is used in order to compute the maximum flow in the corresponding flow network. After computing the flow, a maximum of D paths from the source to the sink are selected and stored. Each path is chosen considering only the graph's edges which have a positive amount of flow. For each such path, the minimum amount of flow f on any of the edges on the path is computed. This will be the capacity of that path. The quantity f is then subtracted from the amount of flow existing on each of the graph's edges which are part of the path. The next path is computed considering the new amounts of flow on the graph's edges, and so on. The process of selecting a path from the current cell to an intermediate cell located in its neighborhood consists of non-deterministically choosing one of the paths. Each path i is chosen with a probability computed according to the formula


Main Cells
Secondary Cells
Other Cells
A.1
A.2
D.1
B.1
C.1
C.2
D.2
B.2
Connection which is part of the path
Connection which is not part of the path
Fig. 2. A possible path from B.1 to D.2.

© 20xx IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

$$P_i = Cap_i / SumCap . \qquad (2)$$

P_i is the probability to choose the i-th path, Cap_i is the capacity of the i-th path and SumCap is the sum of the capacities of all the selected paths towards the intermediate cell. The message is then routed along this path. The next cell making a routing decision is the last cell of the selected path.

The path a message follows to the destination is composed of two types of cells, chosen in the two steps of the routing process. The first type is represented by the main cells and they are the ones making routing decisions. The names of these cells get increasingly closer to the destination's name, as the cells are located further away along the path. Between two consecutive main cells on the path there may be several secondary cells. The secondary cells only pass the message forward and are not subject to naming restrictions. However, all the secondary cells between two consecutive main cells A and B are in the neighborhood of the cell A. The separation into main and secondary cells is based only on the actions taken when routing a message along a path. Any cell could be a main cell for some messages and a secondary cell for others. Fig. 2 shows a possible path a message could follow between the cells B.1 and D.2. The path contains the cells B.1, A.2, C.1, B.2, C.2, D.2, in this order. The network topology is the one presented in Fig. 1 and the radius R of a cell's neighbordhood was considered to be 1 hop.

B. *Routing Multicast Messages*

A multicast message is sent to a destination of the type "PREFIX*", meaning that the message must reach all the cells having a name starting with the prefix "PREFIX". Routing such a message consists of two stages. In the first stage, the message is routed as a unicast message. The message is being routed towards cells having an increasingly longer common prefix with the destination. Eventually, the message will reach a cell having the prefix "PREFIX". When such a cell is reached, the second stage begins. The multicast message is broadcasted towards all the neighbors of the cell which have a name starting with the prefix "PREFIX". These neighbors will broadcast the message further to other peers having a name beginning with the string "PREFIX", and so on. Because of the way the topology is built, all the cells sharing the same common prefix will try to form a connected subgraph in the overlay network and the message will reach all of the destinations.

VI. Measuring the Available Bandwidth

We need to measure the available bandwidth for every direct connection. In order to do this, we will consider every connection to be a cylindrical pipe, having a certain length and a certain cross section. The "length" L of the pipe will be measured in milliseconds. The cross section of the pipe has an area A and will be measured in kilobytes per second. The volume of the pipe, which is the length multiplied by the area, is measured in kilobytes. The capacity of the pipe is its cross-section A and is the value we want to determine. In order to do this, L is periodically estimated based on the round-trip times of several small messages (a few bytes in size). Then, when sending a "large" message having X bytes through the pipe, its "length" will be X/A. The time it takes for the whole message to reach the other side will be (L+X/A). Fig.3 shows a message having X bytes being sent on the connection. The measured RTT (round-trip time) will be equal to twice the value of (L+X/A) and the desired value of A will be

$$A = \frac{X}{\frac{RTT}{2} - L} . \qquad (3)$$

However, if X/A is less than 1, the RTT will be approximately equal to $2 \cdot L$ and the denominator in (3) would be close to 0. In order to achieve a good result, we would need a value of X for which X/A is significantly larger than 1. We can use binary search for X, between a lower limit of 1 byte and some specified upper limit. An upper limit of 3 megabytes for X is enough even for connections having a bandwidth of 10 Gbps. With the binary search, we are looking for values of X for which the RTT is not too close to $2 \cdot L$, aiming for an RTT value close to $F \cdot L$ ($2.2 < F < 4$). Binary search messages are sent at regular intervals.

The value of the estimated available bandwidth is computed with the following formula:

$$AB_{new} = t \cdot AB_{old} + (1-t) \cdot AB_{computed} . \qquad (4)$$

t is a real number between 0 and 1, $AB_{computed}$ is the value of the available bandwidth computed based on the size and RTT of the most recent message, according to (3), AB_{old} is the previous estimation of the available bandwidth and AB_{new} is the new estimation of the available bandwidth.

Because the value of the available bandwidth might largely fluctuate over time, we do not use a tight binary search. Instead, when we need to increase the lower limit of the binary search, we will slightly increase the upper limit, too. Similarly, when decreasing the upper limit of the binary search, we will also decrease the lower limit a bit. Doing so, we provide a solution for the case when the value of the available bandwidth suddenly moves outside the current range of the binary search.

The algorithm estimates the available bandwidth of the TCP connection, as seen from the application level. It does not try to measure the end to end available bandwidth outside the context of the running application. The measurements are affected by the TCP buffer sizes and by the time the messages spend in the message queues. In order to achieve higher throughput, several TCP tuning techniques should be used, as mentioned in [7]. These techniques are outside the scope of this paper.

© 20xx IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

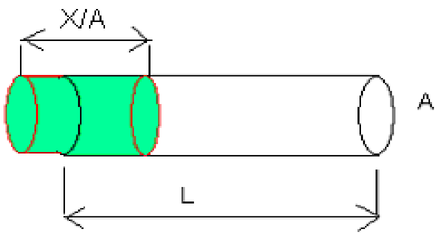


Fig. 3. Sending an X bytes message on the connection.

VII. Experimental Results

At the time of writing this paper, we have implemented only a prototype of the proposed communication framework, using the Java programming language. This prototype implementation has been tested using only a small number of cells. Our test scenarios consisted of 8 cells running on 8 different machines, located at 3 different sites. 2 machines were located at the California Institute of Technology (Site 1), 3 machines were located at CERN (European Center for Nuclear Research), in Switzerland (Site 2) and 3 machines were located at the Polytechnic University of Bucharest, in Romania (Site 3). 2 machines (one at Site 2 and one at Site 3) were located behind firewalls. At the beginning, every machine had a single name and knew the IP and port of one or two other peers. The neighborhood radius was chosen to be 2 hops.

The 2 communication cells located at the first site were named: Site1.Host1 and Site1.Host2. The 3 cells located at the second site had a similar naming model, with names ranging from Site2.Host1 to Site2.Host3. The machines at Site 3 were named following the same pattern, ranging from Site3.Host1 to Site3.Host3. Every communication cell tried to connect to the 2 closest cells having a name lexicographically smaller, the 2 closest cells having a name lexicographically larger and another randomly chosen cell. All the communication cells were started roughly at the same time.

The test cases we used consisted of trains of packets. We considered 3 test scenarios. In the first one, 10.000 messages each having 25.000 bytes were sent from Site1.Host1 to Site3.Host1. Fig. 4 shows the way the total amount of bytes received varied in time. As can be noticed, all the 250 million bytes were received within a time interval of approximately 3 minutes, achieving an average transfer rate of nearly 1.4 MB/s, with a peak rate of about 2.3 MB/s during the first minute. For comparison purposes, we also transferred 250 million bytes using SCP (Secure copy), which achieved a peak rate of 210 KB/s. Except for the SCP test, we also wanted to know what the transfer rate would be if only the direct connection between Site1.Host1 and Site3.Host1 was used. In order to achieve this, we generated a second test scenario, where only Site1.Host1 and Site3.Host1 were started. Fig. 5 shows the variation with time of the total amount of bytes received by Site1.Host1. We achieved a constant transfer rate of approximately 280 KB/s. The transfer rate was noticeably higher when using all the 8 cells, proving that our balanced message flow distribution could be a feasible technique for achieving high throughput data transfers.

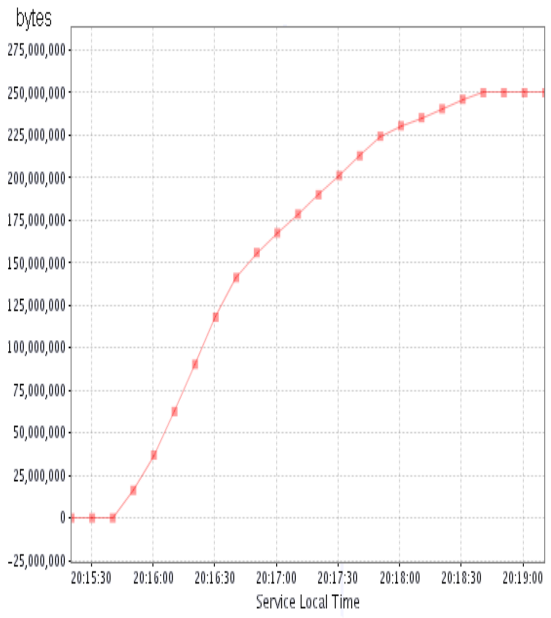


Fig. 4. The variation with time of the total amount of bytes received by Site3.Host1 during the first test scenario.

In the third test scenario we wanted to understand if the message routing overhead was significant. For this, we sent 10.000 messages each having 25.000 bytes between two communication cells located on two machines at Site 2 (Site2.Host1 and Site2.Host2). The transfer rates ranged from 6 MB/s to 9 MB/s. We also transferred 250 million bytes between the 2 machines using SCP. With SCP, the transfer rate reached a peak of approximately 11.3 MB/s. Since in this

© 20xx IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

scenario most of the messages were routed along the direct connection between the 2 machines, the difference between the transfer rates of our system and those achieved by SCP was caused by the routing overhead.

Other test scenarios consisted in testing the multicast message delivery, but the amount of transferred data was quite small and the tests were only intended to verify the correctness of the implementation.

The test results showed that our implementation worked well in a small distributed system. However, the good behavior cannot be extended to large distributed systems without further testing. The test results were also insightful, because we noticed that the routing overhead may be significant in some situations.

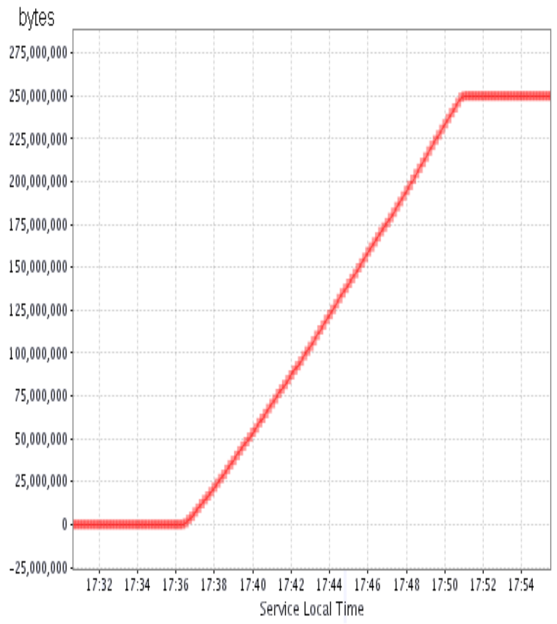


Fig. 5. The variation with time of the total amount of bytes received by Site3.Host1 when using only the direct connection to Site1.Host1.

VIII. Conclusions and Future Work

The test results were not conclusive about the behavior of our system in large distributed systems, but they did show that the framework can behave well in small, although geographically sparse, distributed systems. As part of our future work, we will deploy the communication framework in a large network in order to further test our system. An architecture for generating realistic and significant test cases is already under development.

By having no control over the assignment of names, the communication framework does not fully support peers located behind NATs and firewalls. If there are two cells having names where one is the lexicographic successor of the other, but a direct connection between the cells cannot be established in any of the two directions, the routing algorithm will not work properly. Unicast routing would still work, however, if the two peers are located in the same neighborhood (at most R hops away) and multicast routing would work if all the peers having a prefix equal to the longest common prefix of the two cells form a connected subgraph in the overlay network. We are considering the development of an automatic method of assigning names to the cells, in such a way that the lexicographic distance is related to connectivity capabilities, network latency and available bandwidth.

Other aspects we seek to improve or reconsider are the algorithm used to determine the available bandwidth of a connection, the second stage of the multicast routing algorithm and the enforcement of flow control.

Also as part of our future work, the system will be integrated in the MonALISA framework, presented in [8]. This integration will represent the most important test for our communication framework, because it will have to deal with real-life traffic and not just artificially created environments.

References

[1] S. Ratmasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A Scalable Content Addressable Network,” in Proceedings of the ACM SIGCOMM Conference, 2001

[2] 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 the ACM SIGCOMM Conference, 2001

[3] A. Rowstron, and P. Druschel, “Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems,” in Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms, 2001

[4] A. Rowstron, A. M. Kermarrec, M. Castro, and P. Druschel, “Scribe: The Design of a Large Scale Event Notification Infrastructure,” in Proceedings of the 3rd International Workshop on Networked Group Communication, 2001

[5] M. Castro, et al., “Splitstream: High-Bandwidth Multicast in Cooperative Environments,” in Proceedings of the 19th ACM Symposium on Operating Systems Principles, 2003

[6] M. den Burger, T. Kielmann, and H. E. Bal, “Balanced Multicasting: High-Throughput Communication for Grid Applications,” in Proceedings of the ACM/IEEE Conference on Supercomputing, 2005

© 20xx IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

[7] J. Lee, et al., “Applied Techniques for High Bandwidth Data Transfers across Wide Area Networks,” in Proceedings of the International Conference on Computing in High Energy and Nuclear Physics, 2001

[8] I. C. Legrand, et al., “MonALISA: An Agent based, Dynamic Service System to Monitor, Control and Optimize Grid based Applications,” in Proceedings of the International Conference on Computing in High Energy and Nuclear Physics, 2004

© 20xx IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.