Athens University of
Economics and Business

B.Sc. Thesis Project report:

"Towards an Error Control Scheme for a Publish/Subscribe Network "

by Alexios Voulimeneas

Supervised by Asst. Prof. George Xylomenos and
Ph.D. Candidate Charilaos Stais

Athens, Greece

June 2012

Abstract

This B.Sc. thesis report demonstrates an error-control protocol for multicast data dessemination over an RTFM [16] based Publish/Subscribe network. To the best of our knowledge, this thesis constitutes the first effort in this direction for Publish/Subscribe networks, and it can form the basis for a one to many (or maybe a many to many) transport protocol over a Publish/Subscribe network architecture. Furthermore, we provide a Ns-3 [15] implementation of our protocol over a PURSUIT [7] Ns-3 prototype implemented in MMLAB by Ph.D. Candidate Christos Tsilopoulos and B.Sc. Alumni Ioannis Gasparis.

Acknowledgment

I would like to thank Dr. George Xylomenos and Ph.D. Candidate Charilaos Stais for supervising my thesis, as well as Ph.D, candidate Christos Tsilopoulos and B.Sc. Alumni Ioannis Gasparis for their collaboration and advice. Working on this thesis has given me important experience on conducting network research.

Contents

1. Introduction

The Internet's success is undeniable. One of the main reasons of this success is the TCP/IP network stack, which led to reliable communication between end hosts. However, the source of Internet's spread may be its bigger disadvantage in the future. Nowadays, we are interested in content and not from where we get it. Applications like Bit-Torrent, P2P and CDNs are examples of this need for content. These applications and services have *information* as their core, not the end hosts consuming or producing it, and thus are developed as overlay solutions.

The existence of such applications shows the disadvantages inherent in TCP/IP-based networking. These networks are focusing on the location of the data. Therefore, in case of popular content producers, a considerable amount of network resources is needed near the data sources, as the same packet has to be delivered many times (one per receiver). Multicast delivery over a multicast tree (with data source as its root and receivers as its leafs) is a good solution, whereby each packet crosses tree link only once. Unfortunately, IP Multicast's deployment was a hard task, due to both technical and business reasons [1].

Consequently, one of the main problems of today's Internet is the lack of information awareness. Many people from the networking research community believe that only a data-centric model will respond to the future needs. There are various research projects based on content such as CCNx [17], SAIL [18] and PURSUIT. However, there are specific problems associated with these projects, i.e. false forwarding positives in PURSUIT's case. A critical question often raised is whether efficient and reliable content delivery can be achieved over a data-centric architecture.

The FP7 EU project PURSUIT for instance is concerned with designing and implementing a clean-slate publish/subscribe internetworking (PSI) archtecture, where information itself is at the core. Although, we can see progress in this field, the problem of designing a TCP/IP equivalent for a PSI architecture has yet to be faced. Furthermore, there is no work on native multicast transport support for PSI, which is one of its strong points. The contribution of this thesis is to provide an efficient way for transport error-control suitable for PSI, which takes advantage of PSI's features and prior work on IP-Multicast (especially PGM [12]). Moreover, we also provide a preliminary evaluation of our mechanism's performance. We strongly believe that our work can be a good basis for the development of an integrated transport layer for a Publish/Subscribe network.

The remainder of this report is organized as follows. In Section 2 we present the PSI architecture, while on Section 3 we describe the error control mechanism that we have designed. In Section 4 we discuss the experimentation environment that we used and present the simulation results. The final Section provides our conclusions and ideas for future work.

2. Background

2.1 A Brief Background on Publish/Subscribe

Publish/Subscribe is a novel future Internet architecture. In this type of network there are three types of network elements: publishers, subscribers and an event notification service, also called a Rendez-Vous network, consisting of Rendez-Vous Points (RVPs) [10]. When a publisher wants to make some content available to the network, it sends a publication message (including all important information: name/location of publisher, content's metadata) to the responsible RVP. On the other hand, subscribers send subscription messages to an RVP to express their interest for specific information items. PSI uses the combination of a Scope Identifier (SId) and a Rendez-Vous Identifier (RId) to identify the content to be published or to be subscribed to. The SId identifies a collection of items and is mapped to the RVP responsible for that specific collection, while the RId is an identifier derived by the publishing application, indicating a particular item within collection. The scoping mechanism limits the reachability of information only to the parties having access to a particular scope. Furthermore, scopes employ a hierarchical structure, where parent-children relationships exist [6].

When a subscription message arrives at an RVP, the RVP initially locates the publishers providing information items that satisfy the subscription. It then communicates with a Topology Manager (TM), which may be a service in the same machine or a stand-alone server, in order to get a suitable forwarding path from the publisher towards the subscribers. The TM maintains information about the current topology (interconnection between routers) in order to find paths between the hosts. The TM then calculates a multicast tree containing the publisher and subscribers, possibly by merging the shortest paths from the publisher to each subscriber.

The multicast tree generated by the TM is encoded in a Bloom filter following the approach of LIPSIN [11]. Bloom filters are probabilistic representations of sets, where each element of the set is encoded as a string of zeroes and ones using a set of hash functions. A set is represented as the logical OR of all elements of the set. Each packet contains in its header a Bloom filter, which encodes the labels of all the links that are part of the path, whether unicast or multicast. When a packet arrives at a router, the router looks at the Bloom filter and tries to find out to which of its outgoing links it will have to push the packet, by performing a logical AND between the label of each link and the in-packet Bloom filter.

2.2 Bloom filter-based Relay Architecture

Transport in Publish/Subscribe is based on source routing using Bloom filters. However, scalability problems occur because Bloom filters lead to false positives when the size of the delivery tree grows beyond a defined limit (as more links are

added to a Bloom filter, it becomes more likely that they will match a link that was not added to them). One solution provided by PURSUIT is Bloom filter-based Relay Architecture (BRA). It divides multicast delivery trees into several smaller subtrees. Generating good subtrees (and consequently good relay points) is still a topic of research, however we will not focus on this and we will use simple techniques based on DFS and BFS algorithms in our approach. Each of these subtrees has its own in-Packet Bloom filter that is used for forwarding packets to the destinations within the given subtree. The subtrees are combined together using relay nodes between a publisher and a set of subscribers. By using BRA we achieve the reduction of false positives because cutting the delivery tree into smaller subtrees decreases the fill factor of each Bloom filter (the amount of ones contained in the Bloom filter).

3. Multicast Error Control

The basic issue in the development of an error control protocol, and especially one suitable for multicast, is the technique the subscribers will use to sent feedback information back to publishers. In PSI architecture communication is based on Bloom filters. Consequently, we need to provide appropriate Bloom filters to the publisher and subscribers with an efficient and accurate way (unicast delivery for each subscriber is not an option because the number of subscribers may be quite large). The TM normally calculates downstream Bloom filters by OR-ing the link labels for the forward direction of the tree. In our scheme, we modify the TM in order to use the reverse direction tree to construct the upstream Bloom filters. Furthermore, for simplicity, we use the same relay points for data forwarding and feedback.

The set of downstream and upstream Bloom filters are sent back to the RVP (if the RVP is not co-located with the TM). The RVP sends a pair of Bloom filters to the publisher and to each relay point, one to be used for delivering data (to subscribers or relay points) and another to be used for feedback information (from subscribers or relay points). The publisher then sends a special setup message called FIRST_MSG in which it encapsulates the upstream Bloom filter used from its children* for feedback. Upon reception of this message, each relay point extracts and stores the reverse Bloom filter used to reach its parent** in the tree and then forwards FIRST_MSG, replacing the reverse Bloom filter with the one used from its children (relay points or subscribers) to reach it. By the end of this procedure, only by sending one multicast message from the publisher, each relay point and subscriber receives the Bloom filters needed to communicate with its parent and children for a specific SId/RId pair.

*Children here refers to subscribers or relay points, for which a downstream path from the current node (relay or the publisher) without indermediate relays exists.
**Parent here refers to the publisher or relay points, for which a downstream path from the current node (relay or subscriber) without indermediate relays exists.

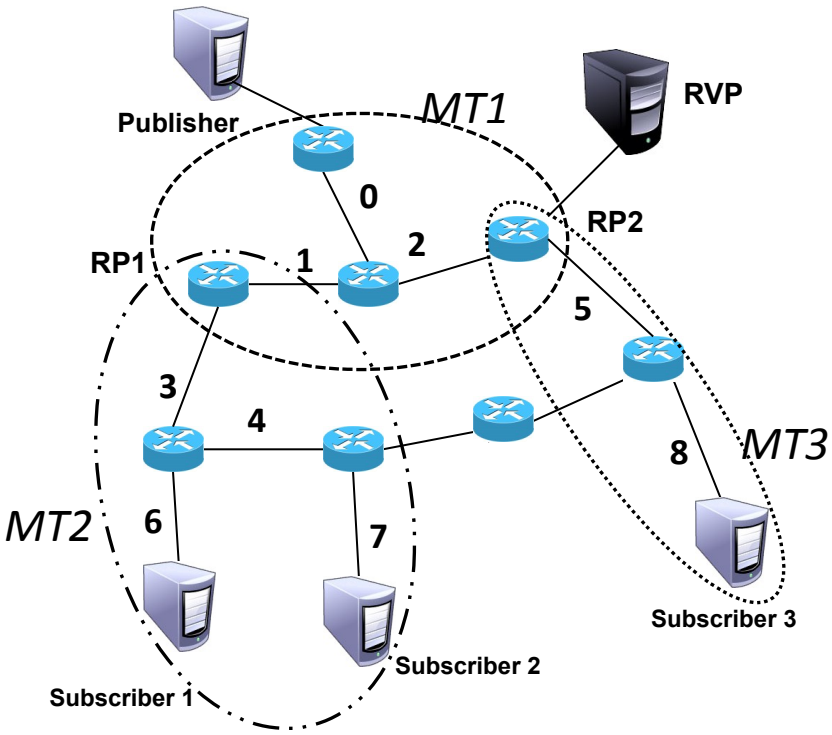

Fig. 1. An example of multicast tree setup

Figure 1 represents a typical instance of our approach. The main network components of this architecture are one Publisher, three Subscribers and two Relay Points. Initially, the RVP sends a pair of filters (PUBup, and PUBdown) to the Publisher, a pair to RP1 (RP1up, RP1down) and a pair to RP2 (RP2up, RP2down). The setup message is forwarding from the publisher using PUBdown and encapsulating PUBup. When it reaches RP1 and RP2 through paths {1} and {2, 5} respectively, they store PUBup as their upstream Bloom filter towards the publisher. Then RP1 replaces PUBup with RP1up and RP2 with RP2up, and they both forward setup message using their own downstream Bloom filters(RP1down and RP2down). Each receiver of the message stores the encapsulating Bloom filter for sending feedback and the procedure continues until all subscribers receive the appropriate upstream Bloom filters.

After setup completes, the publisher starts sending data packets of the content item. At each relay point, the relay looks the Sid/RId in the packet's header and replaces the forwarding Bloom filter with the one needed to reach the next relay points or subscribers. For example RP1 would replace PUBdown with RP1down. When all the data chunks are sent, the content distribution phase is completed.

The above phase is enough for complete data delivery if and only if no packet losses occur, but that is normally not the case. Therefore, our mechanism is completed with a retransmission phase in one or more rounds. When a subscriber detects a packets loss based on sequence number (all packets follow the same path, therefore if they don't come in the right order, we conclude that one or more packet losses occurred) or based on checksum (packet corruption), she uses the stored upstream Bloom filter to send feedback information called a negative acknowledgement (NAK) message for

this sequence number. The NAK is forwarded to the next parent relay point. The relay point holds the packet for a specific time, waiting for more NAKs to come in case more subscribers have lost the same or other packets. For example, if corruption of a packet happens in a relay point near the root of the tree, almost all the subscribers will miss that packet and multiple NAKs will be received. If more NAKs come at the relay point during the waiting period, they are combined into a single "larger" NAK message forwarding upstream by using feedback Bloom filter. This procedure helps us reduce traffic caused from feedback information to the network and its merely a solution to NAK and ACK implosion by other reliable multicast protocols.

When the publisher finishes the initial file distribution, she sends a confirmation acknowledgement (CONFACK) in order to inform the subscribers that the retransmission round has finished. This message is explicitly acknowledged by each subscriber via an empty NAK. The purpose of this message is to let subscribers detect lost packets at the end of the round. Then a retransmission round begins, with the publisher transmitting all packets for which NAKs have been received. This (CONFACK, NAK) procedure continues until all recipients receive the entire file.

4 Experimental Setup and Evaluation

4.1 Simulation Setup

For our simulation experiments we used Ns-3, where the entire PURSUIT architecture was implemented. We built a pub/sub based internetworking architecture called RTFM (for Rendezvous, Topology, Forwarding, and physical Media architecture) and a classic content distribution application (Video-On demand) with a number of subscribers expressing their interest for a specific content.

In our scenarios we used 20MB files, being delivered in 20K data packets of 1 KB payload each. Furthermore, we used randomly generated internet-like (scale-free) topologies of 200 and 500 router with 50 and 100 subscribers respectively, attached to randomly selected routers; smaller topologies can be handled without any relay points. In our experiments, we used the BFS algorithm along the multicast tree in order to select the relay points for forwarding and feedback traffic. For each scenario we used one publisher (per content) randomly located to a topology router and 50 or 100 subscribers according to the topology. We execute this 5 times, in order to generate different multicast trees. Moreover, we used a variety of 200-routers and 500-routers scale free topologies in order to take averages of our measured statistics. Finally, we assumed that losses were random, i.e. packets were lost with a probability of x% or y% for 200 and 500 router topologies respectively. Both values are determined in an experimental manner in order to achieve an overall 3% packet loss rate in our scenarios. Our protocol does not use a congestion control mechanism, therefore congestion control losses are out of topic in our approach (see future work).

In our scenarios the RVP could work in two modes. In the first mode, when a subscriber asks for a specific content, it waits for a specific amount of time before it starts a session. In the second mode it waits for a certain number of subscribers to ask for a content before it starts the session. We can also use a combination of the two modes. In all three cases, a multicast tree is constructed and multicast data delivery starts.

4.2 Simulation Results

Our first metric is the aggregation rate of NAKs achieved by our scheme, as it shows to what extent we have avoided the feedback implosion problem which is the common problem of a multicast transport protocol. The aggregation rate is measured using the type (s-r)/s or 1- s/r where s is the number of packets send from subscribers and r is the number of packets received from the publisher. As shown in Figure 2, we achieve higher aggregation rates in bigger topologies. The explanation is that in bigger topologies larger multicast trees are constructed and consequently BFS algorithm generates more relay points, thus increasing the points where NAKs are aggregated. We also observe that if we apply the above procedure at byte level, the aggregation level is 5% lower in each case, since aggregation leads to fewer but larger NAK messages.

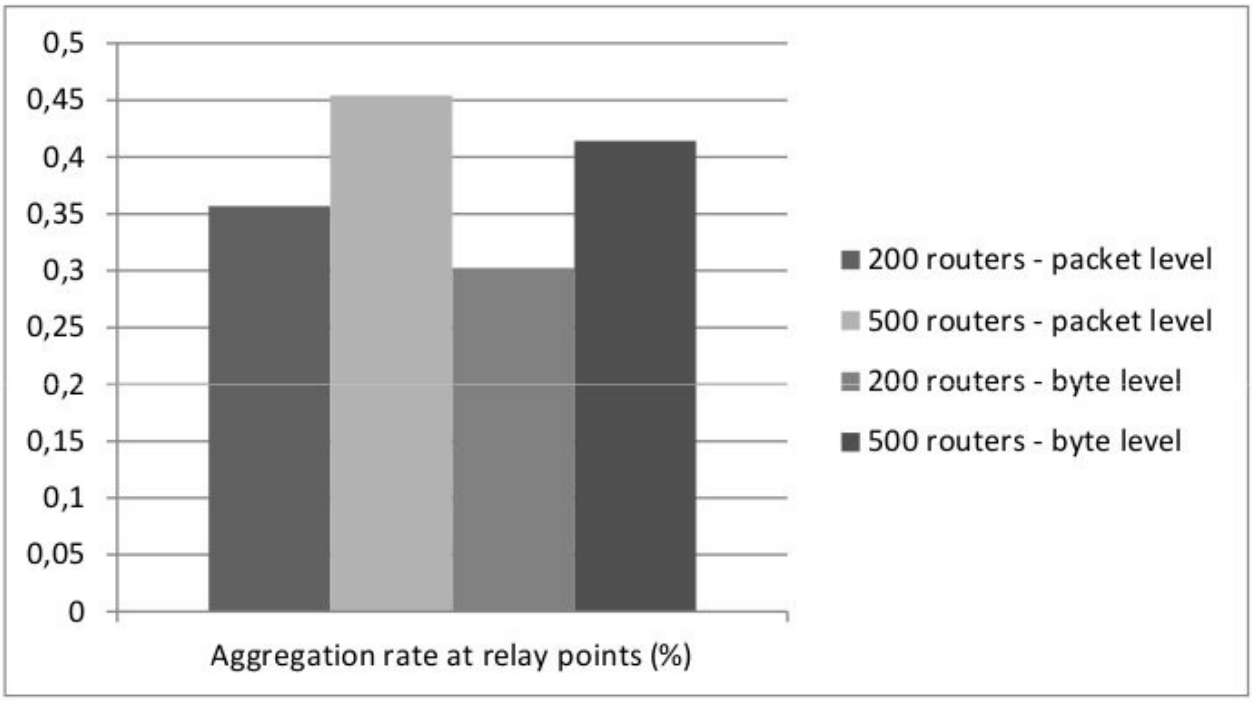

Fig. 2. NAK aggregation rate.

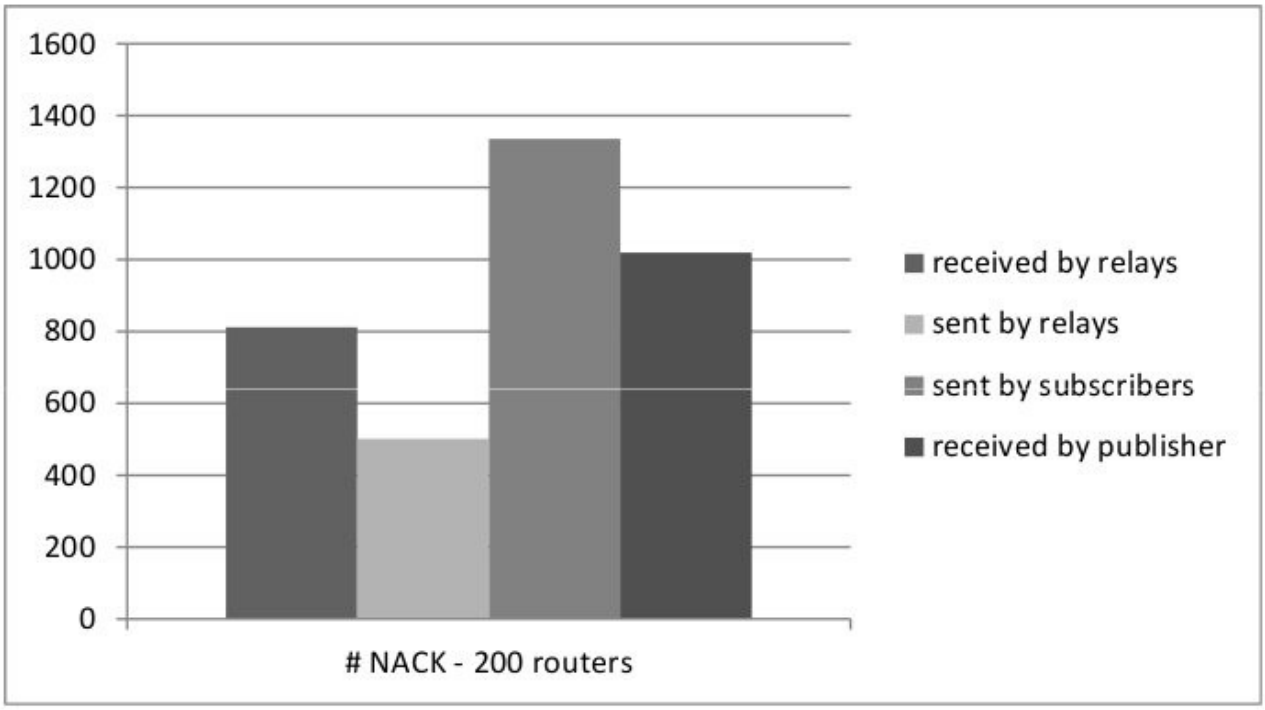

Fig. 3. Number of NAKs handled by architecture entities (200 routers).

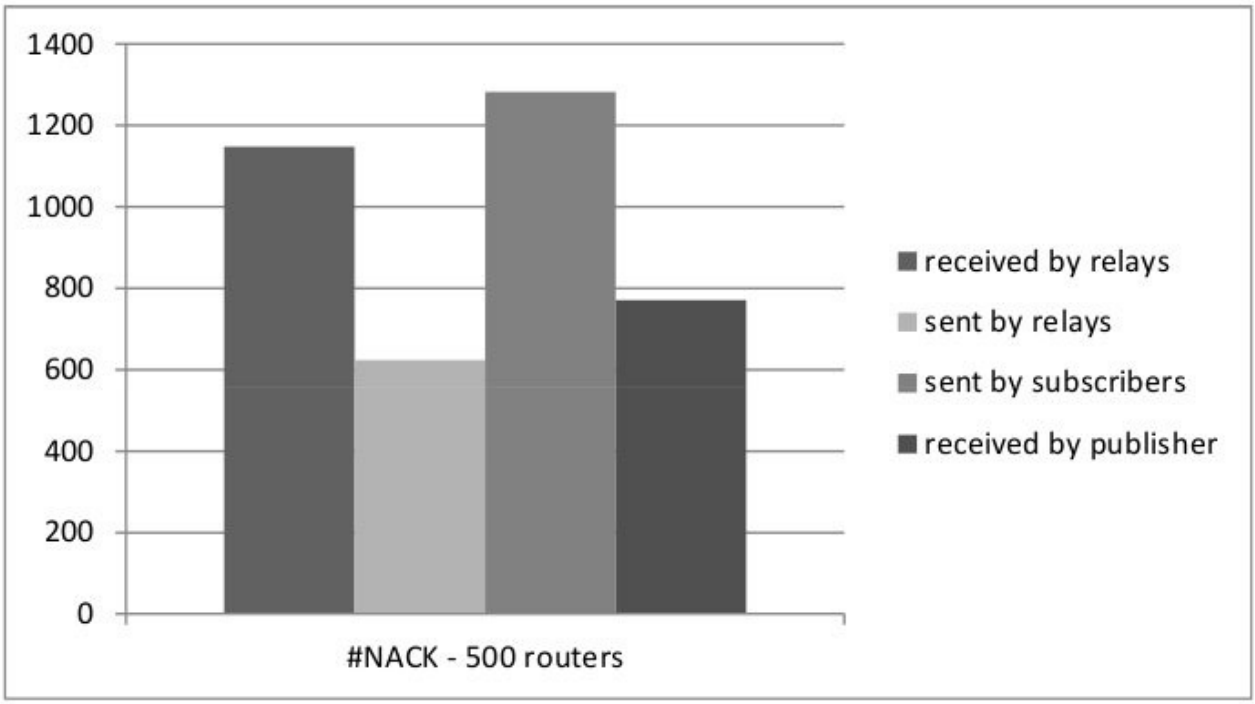

Fig. 4. Number of NAKs handled by architecture entities (500 routers).

In Figure 3 and 4 we provide more details about NAK handling, showing the number of NAKs received and sent by relays, the number generated by the subscribers and the number received by the publisher in both 200 and 500 router topologies. First, the number of NAKs generated by the receivers is roughly the same in both topologies, as we have ensured that the overall loss probability for data packets is roughly the

same. Second, not all NAKs generated by the receivers reach the relay points, as some subscribers reach the publisher directly. Third, while in the larger topology more NAKs are received and transmitted by the relays, the multiple levels of relaying in the larger topology lead to more aggregation, hence fewer NAKs reaching the sender, as observed in Figure 2.

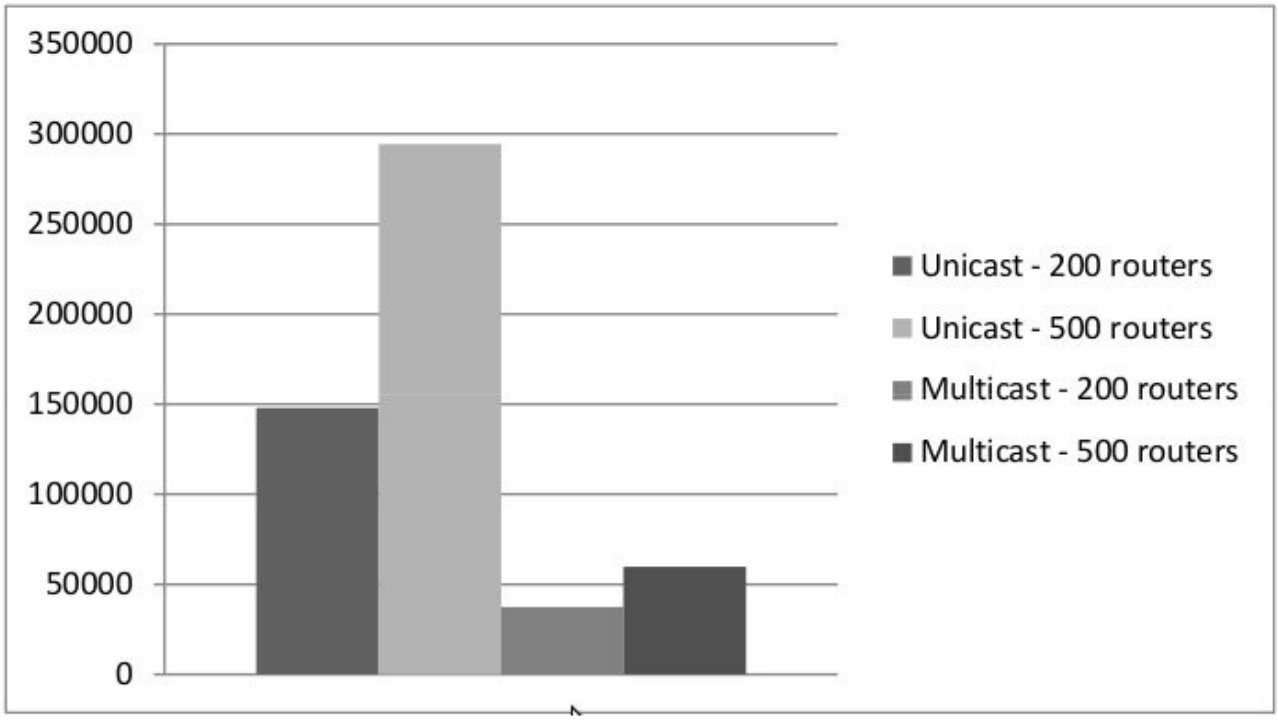

Fig. 5. Amount of feedback traffic handled by network nodes.

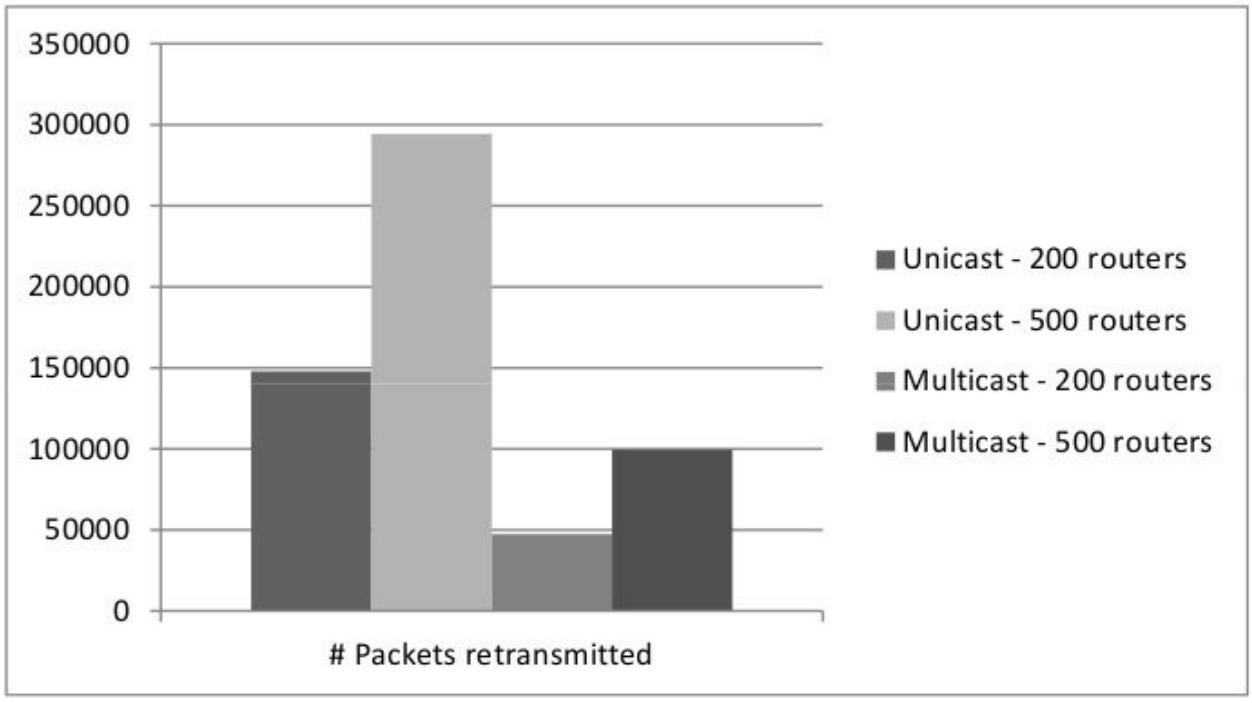

Fig. 6. Amount of recovery traffic handled by network nodes.

Finally, we compared our multicast error recovery scheme with unicast error recovery. As we see in Figure 5, the number of NAKs handled by the network components are about 400-500% (according to the topology) more in the unicast case. Concerning retransmissions, as shown in Figure 6 the number of packets retransmitted by all network elements involved (publisher and all intermediate routers) with our scheme is around 32% and 33% of those needed with unicasting. Note that with unicast recovery the same number of packets are needed in both directions, as one NAK triggers exactly one recovery packet. With our scheme, NAKs are fewer than recovery packets since (a) NAKs for different missing packets can be aggregated and (b) recovery packets are multicasted in every subtree where at least one receiver requested them; higher packet loss rates would lead to more NAKs in each subtree, hence causing the gap between NAKs and recovery packets to close These results show that our mechanism leads to many benefits in both directions compared to unicast recovery.

5. CONCLUSION AND FUTURE WORK

In this report we presented an approach for multicast error control for reliable, on-demand, delivery of information over a Publish/Subscribe network supporting native multicast, using relay points to extend the reach of the source-routing mechanism used. We also provide a preliminary evaluation of our mechanism's performance.

Future work includes to move further and couple the error control scheme with an efficient solution for congestion control over the PSI architecture. Finally, we are planning to examine the effectiveness of caching at relay points in order to enable local retransmissions of lost data.

6. REFERENCES

[1] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, “Deployment issues for the IP multicast service and architecture,” IEEE Network, pp. 78 –88, 2000.

[2] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, “ROFL: routing on flat labels,” Proc. of the ACM SIGCOMM, pp. 363–374, 2006.

[3] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, “A data-oriented (and beyond) network architecture,” in Proc. of the ACM SIGCOMM, 2007, pp. 181–192.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in Proc. of the ACM CoNEXT, 2009, pp. 1–12.

[5] N. Fotiou, D. Trossen, and G. C. Polyzos, “Illustrating a publish-

subscribe Internet architecture," Springer Telecommunication Systems, pp. 1–13, 2011.
[6] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. A. Siris, and G. C. Polyzos, "Caching and mobility support in a publish-subscribe Internet architecture," IEEE Communications, vol. 50, no. 7, pp. 52–58, 2012.
[7] PURSUIT: Publish subscribe Internet technology. [Online]. Available: www.fp7-pursuit.eu
[8] G. Xylomenos and B. Cici, "Design and evaluation of a socket emulator for publish/subscribe networks," in Proc. of the Future Internet Symposium, 2010.
[9] C. Stais, D. Diamantis, C. Aretha, and G. Xylomenos, "VoPSI: voice over a publish-subscribe internetwork," in Proceedings of the Future and Mobile Network Summit, 2011.
[10] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," ACM Computer Survey, vol. 35, pp. 114–131, 2003.
[11] P. Jokela, A. Zahemszky, S. Arianfar, P. Nikander, and C. Esteve, "LIPSIN: line speed publish/subscribe internetworking," in Proc. of the ACM SIGCOMM, 2009, pp. 195–206.
[12] J. Gemmell, T. Montgomery, T. Speakman, and J. Crowcroft, "The PGM reliable multicast protocol," IEEE Network, vol. 17, no. 1, pp. 16 – 22, 2003.
[13] M. Sarela, C. E. Rothenberg, T. Aura, A. Zahemszky, P. Nikander, , and J. Ott, "Forwarding anomalies in Bloom filter-based multicast," in Proc. of the IEEE INFOCOM, 2011, pp. 2399 –2407.
[14] C. Tsilopoulos and G. Xylomenos, "Scaling bloom filter-based multicast with switched-ibfs," in Submitted for publication, 2012.
[15] Ns-3 simulator. [Online]. Available: www.nsnam.org
[16] M. Sarela, T. Rinta-aho, S. Tarkoma, RTFM: Publish/Subscribe Internetworking Architecture, In ICT Mobile Summit, 2008
[17] PARC Content Centric Networking, http://ccnx.org
[18] SAIL: Scalable and Adaptive Internet Solutions, EU FP7 research project, http://www.sail-project.eu/