Towards an Error Control Scheme for a Publish/Subscribe Network

Charilaos Stais, Alexios Voulimeneas, George Xylomenos
Mobile Multimedia Laboratory
Athens University of Economics and Business
Athens, Greece +30 210 8203693
Email: stais@aueb.gr, avoulimeneas@cs.aueb.gr, xgeorge@aueb.gr

***Abstract*—Many proposals for the next generation of the Internet suggest moving from an end-point oriented to an information-centric oriented architecture. Many of these proposals are based on the publish/subscribe paradigm, which lends itself naturally to native multicast support, a key factor for efficient content distribution. However, the design of efficient reliable transport protocols for multicast is a largely open problem, due to the problem of feedback implosion towards the sender as group size grows. In this paper we propose a hierarchical retransmission-based error control scheme for a native publish/subscribe internetwork. We compare our protocol with similar approaches proposed for IP multicast and evaluate its performance against IP multicast with unicast-based error control.**

***Index Terms*—Information-centric networks, error recovery, multicast, transport layer**

I. Introduction

The Internet has proliferated, primarily due to the omnipresent TCP/IP protocol suite. Adhering to the then prevailing principles of telecommunications technology, TCP/IP-based computer networks were designed to forward data traffic among communicating end hosts. In the course of time, the Internet has evolved into a substrate for *information-centric*, or *content-centric*, applications and services such as *peer-to-peer* (P2P) file sharing, *content delivery networks* (CDNs) and cloud computing services. As these applications and services focus on *information*, not on the end hosts producing or consuming it, they are developed as overlays over TCP/IP. This development reveals the disadvantages inherent in TCP/IP. When large numbers of information consumers are served via end-to-end connections to a few information producers, such as popular web sites, a large amount of network resources are needed close to the data sources, as the same packet has to be transmitted multiple times over the same link (once per receiver). A solution to this problem is multicast, i.e. the distribution of data from a single source to a group of receivers via a multicast tree, whereby each packet crosses each tree link only once. Unfortunately, IP multicast turned out to be hard to deploy, for both technical and business reasons [1].

Many researchers have suggested entirely redesigning the Internet with the aim of satisfying the requirements of current and future content distribution applications. Most approaches recommend that *information should be the focus of interest in the internetworking architecture* [2], [3]. The FP7 EU project PURSUIT [4], and its predecessor PSIRP, are concerned with designing and implementing a clean-slate *publish/subscribe internetworking* (PSI) architecture [5], where the information itself is at the core and multicast is natively supported. This model is in sharp contrast to the existing Internet model, which focuses on the location of the information.

A critical question is whether effective information-centric transport protocols can be designed, that would improve application performance over TCP/IP by taking advantage of these new architectures. Although initial work shows that simple bi-directional communication applications can be easily ported to PSI [6], there is practically no work on exploiting the support of PSI for native multicast. In this paper, we take advantage of past work on IP multicast-based transport, in order to design an error control mechanism for reliable multicast content distribution, specifically designed for the PSI architecture, and present a preliminary evaluation of its performance. An obvious use case for our scheme is the reliable delivery of software updates to large numbers of terminals. A less obvious use case is the reliable delivery of sensor data to multiple sinks, for example, data from sound and image sensors sent to many human and robotic rescuers working in a building hit by an earthquake, as envisioned in the DISFER project [7].

The remainder of this paper is organized as follows. In Section II we briefly present the PSI architecture and past work on reliable multicast transport, while in Section III we discuss the error control mechanism that we have designed. Section IV first provides a description of the experimentation environment and then presents the performance results obtained. We finally conclude and discuss our plans for future work in Section V.

II. Background & Related Work

A. The PSI architecture

A publish/subscribe architecture relies on three basic entities: publishers, subscribers and an event notification service, also called a Rendez-Vous network, consisting of *Rendez-Vous Points* (RVPs). When a publisher wants to make content available to the network, it advertises its availability to the responsible RVP, by issuing a publication message. Subscribers are information consumers, who express their interest for content by issuing subscription messages. PSI uses the combination of a *Scope Identifier* (SId) and a *Rendez-Vous Identifier* (RId) to identify content. The SId identifies a collection of items and is mapped to the RVP which is

responsible for that specific collection, while the RId is an identifier derived by the publishing application, indicating a particular item within the collection. The scoping mechanism limits the reachability of information to the parties having access to a particular scope. Within a scope, the architecture is neutral with respect to the semantics and structure of the data. Scopes employ a hierarchical structure, where parent-children relationships exist [5]. Scopes may be both physical, such as a local network, and logical, such as a social network.

When a subscription message arrives at an RVP, the RVP first locates a publisher providing the 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 suitable data forwarding paths. The TM maintains information about the current network topology so as to be able to find paths between the hosts. The TM then calculates a multicast tree containing the publisher and all 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 [8]. Bloom filters are probabilistic representations of sets, where each set element is encoded as a string of zeroes and ones calculated via a number of hash functions. A set is represented as the logical OR of all set elements. Each packet contains in its header an *in-packet Bloom filter* (iBF), 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 examines the iBF in order to determine 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 iBF.

B. Reliable Multicast

Even though IP multicast was not widely deployed [1], considerable work was dedicated to multicast transport layer protocols. The culmination of these efforts is *Pragmatic General Multicast* (PGM) [9], a reliable multicast transport protocol that guarantees that a receiver in the multicast group will either receive all data packets (from their original transmission or a retransmission), or will detect unrecoverable data packet losses. In order to avoid a feedback implosion from the large numbers of receivers towards the sender, PGM relies on two mechanisms: first, only *negative acknowledgments* (NAKs) are used to report missing packets and, second, a hierarchy of PGM-enabled routers, called *network elements* (NEs), are deployed throughout the multicast tree to aggregate feedback from the receivers towards the sender. Essentially, each NE is responsible for the subtree rooted at itself and extending downstream up to either the receivers or the downstream NEs.

When a packet loss is detected by a receiver, it unicasts a NAK towards its parent NE after a random waiting period. The parent NE on reception of the NAK multicasts a *NAK confirmation* (NCF) to its subtree, so as to suppress NAKs for the same lost packet from other receivers, and notes the packet number requested by the NAK. The NE then pushes the NAK towards its own parent, which in turn multicasts an NCF to its own subtree, and so on, until the NAK reaches the source, in which case the missing packet is retransmitted downstream. Only NEs that have noted that they received a NAK for this packet forward it downstream, therefore retransmissions only reach the subtrees where at least one receiver has reported that particular packet to be lost. To further reduce the number of NAKs, an NE can create NAKs indicating multiple losses.

PGM has several other features, such as support for retransmissions from local caching nodes, recovery based on forward error correction and congestion control. These features will not concern us further, as they can be integrated to our own error control scheme in the same manner as with PGM.

III. Multicast Error Control

As explained in [10], iBF-based forwarding suffers from false positives: as more links are added to a Bloom filter, it is more likely that they will match a link not added to them, thus leading to redundant packet transmissions and, even, packet loops. The extent of this problem depends on how many links are encoded into the set. In [10], the authors propose that the number of ones in the iBF should not exceed 40% of the total bits, meaning that with reasonably sized iBFs (since they must fit within packet headers) we cannot represent very large trees.

To allow iBF-based forwarding to support larger multicast groups, we exploit iBF switching [11] at designated relay points. Relay points are routers that replace the iBF inside a packet with a new one before forwarding the packet. When a packet arrives, the relay point checks in its database if an entry for the combination of the SId and the RId of the packet exists and, if so, replaces the iBF with a stored one. When the TM constructs the initial iBF, it pays attention to the ratio of ones in it. If the ratio exceeds 40%, it will have to resort to relay points, which are selected during a breadth-first traversal of the entire multicast tree. The TM constructs iBFs from the publisher to the relay points and from these relay points to the subscribers or to new relay points, if needed. The relay points and the iBFs are returned to the RVP, which is now responsible to inform the relay points how to switch filters when they receive a packet with a specified SId and RId.[1]

Having addressed the iBF scaling issue, we now turn to preventing feedback implosion. Our mechanism is similar to PGM, in that it uses selected routers to aggregate feedback and control the propagation of retransmissions, but adapted for PSI. We reuse the iBF relay points, which are mandated by the forwarding architecture, as feedback concentrators that prevent subscribers from overwhelming the publisher. The target application for our protocol is fully reliable multicast delivery, for example, the distribution of software updates over the network, or the distribution of critical sensor readings to multiple sinks. In these applications each recipient must receive all data correctly. In contrast, PGM targets applications with continuous data transmission, where it is acceptable for some receivers to miss some packets due to persistent losses.

Our scheme operates in three phases: setup, where the multicast tree is created, initial content distribution, where the

[1]We are using the SId and RId instead of the iBF itself to identify the next iBF so as to allow the iBFs between two relay points to evolve (due to link additions and deletions) without updating the lookup tables at all relay points.

publisher sends the content, and recovery, which may include one or more cycles of feedback and retransmissions until all subscribers have successfully received all data. In the setup phase, the iBFs are calculated and distributed to the relay points. After the setup phase, the publisher distributes the content without stopping for retransmissions. Finally, the recovery phase is a loop, where the publisher receives information on lost packets and retransmits them in the next cycle. In contrast, in PGM retransmissions are interspersed with regular packet transmissions; if a packet is not received after some time, the sender abandons the effort.

Since iBFs are unidirectional, we need separate iBFs in order to return feedback from the subscribers to the publisher. We therefore modified the TM to calculate not only downstream iBFs from the root to the leaves of each multicast tree, but also upstream iBFs, by simply ORing the link labels for the reverse direction of the tree; these upstream filters represent a tree from all leaves towards the root, hence they can be used to send upstream messages from any leaf to the root.

The set of forward and reverse iBFs for the sender and each relay point are sent to the RVP, which in turn sends to the publisher and *each* relay point a downstream iBF, used for data forwarding, and an upstream iBF, used for feedback, both within the relay's subtree. The publisher then sends a FIRST_MSG message to initiate content distribution; this message includes the upstream iBF that should be used for feedback and a generation counter set to zero. Upon reception of this message, each relay point stores the upstream iBF used to reach its parent, switches the downstream iBF in the packet and forwards the message, encapsulating inside it the upstream iBF that should be used by its own children for feedback. At the end of this process, each relay point knows the iBF needed to reach its children (from the original RVP message) and each relay point and subscriber knows the iBF needed to reach its parent (from the setup message) for a specific SId/RId pair.

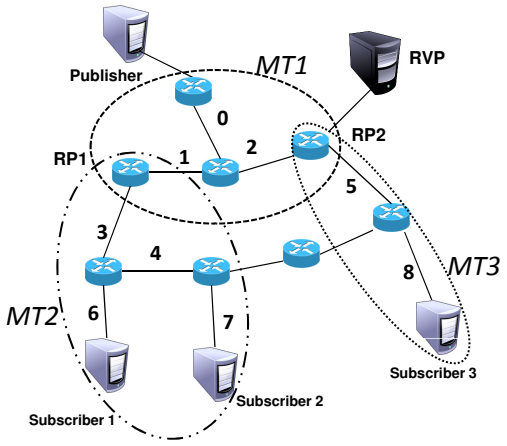

Fig. 1. An example of multicast tree setup.

Figure 1 represents a typical instance of our approach. There is one Publisher, three Subscribers and two Relay Points. Initially, the RVP sends a pair of iBFs (F_{Pub}, R_{Pub}) to the publisher, a pair (F_{RP1}, R_{RP1}) to RP1 and another pair (F_{RP2}, R_{RP2}) to RP2. The FIRST_MSG message from the publisher is forwarded using F_{Pub} and encapsulating R_{Pub}. When it reaches RP1 through the path $\{0,1\}$ and RP2 through the path $\{0,2\}$, each RP stores R_{Pub} as its feedback iBF towards the publisher. Then the message is forwarded from RP1 using F_{RP1} and encapsulating R_{RP1} and from RP2 using F_{RP2} and encapsulating R_{RP2}. Each receiver, upon reception of this message, stores the encapsulated iBF for feedback. In this way the multicast tree from publisher to subscribers is divided into three smaller ones (MT1-MT3).

After setup completes, the publisher starts sending the data packets of the content. At each relay point, the relay looks up the SId/RId in the packet and replaces the forwarding iBF with the one needed to reach the next relay point or subscribers. For example, RP1 would replace F_{Pub} with F_{RP1}. When a subscriber detects that a packet has been lost (based on sequence numbers) or corrupted (based on checksums), it uses its upstream iBF to return a *negative acknowledgment* (NAK) message to its upstream relay point. The relay point holds the packet for a specified interval, waiting for more NAKs to come. If additional NAKs, either for the same or different packets, arrive at the relay point during the waiting period, they are combined into a single NAK which is forwarded upstream, by using the corresponding upstream iBF. Each relay point notes the NAKs arriving from its own subtree, in order to later forward recovery data only where needed.

When the publisher finishes the initial content distribution, it waits for a specified period of time to allow nodes that have received the entire transmission to leave the multicast group and the TM to issue new iBF pairs wherever needed. It then sends a new FIRST_MSG with the generation counter increased by one, so as to distinguish packets from different recovery cycles. This message lets subscribers know that the current transmission round has finished, allowing them to detect lost packets at the end of the current round. It also updates the upstream iBFs throughout the multicast tree and allows relay points to reset any NAK state from previous rounds. Then a retransmission round begins, with the publisher transmitting all packets for which NAKs have been received; each relay point only forwards in its subtree the packets for which it has noted that NAKs had been received. This procedure is repeated at the end of every retransmission round, until all packets are received. Since subscribers leave the multicast group after correctly receiving all data, the tree will eventually be torn down, thus concluding the transfer.

Our approach slightly departs from PGM in the way feedback is aggregated. In PGM, the relay point attempts to suppress further NAKs for a packet by multicasting an NCF, while in our approach each receiver sends NAKs for all missing packets to its upstream relay point. If the loss probability is similar for all links in a subtree, it is unlikely that many receivers will lose the same packet, therefore it is wasteful to multicast an NCF to the entire subtree. The main difference however is that PGM sends new data interspersed with retransmissions, eventually giving up on lost packets, while our mechanism retransmits packets in rounds until all have been received, expecting that in each round some group members will depart, hence reducing the number of links that

each retransmitted packet will cross at every round.

IV. Experimentation and Evaluation

A. Simulator Setup

For our simulation experiments, we used NS-3 [12], where the entire PSI architecture was implemented, including iBF-based forwarding and relay points. We modeled a single publisher reliably distributing content, for example, an operating system patch. The content size is 20MB, composed of 20.000 data packets with a payload of 1KB each. We used randomly generated scale-free network topologies of 200 and 500 routers with 50 and 100 subscribers attached to randomly chosen routers, leading to an average of 15.6 and 28.6 relay points, respectively; smaller topologies can be handled without relay points. Each scenario was executed 5 times with different random positions of attachment to the network for the publisher, resulting in a different tree being generated and different relay points being chosen each time. Finally, we assumed that all losses were random, i.e. packets were lost with probability of $x\%$ or $y\%$ in each link, for the 200 and 500 router topology, respectively. The values of x and y where chosen experimentally so that during the initial file distribution phase 3% of the packets transmitted were reported lost (to the sender) in both topologies, despite the different number of links and receivers involved. This translates to roughly 600 packets that need to be retransmitted during the first recovery round. We did not model the correlated losses that are usually associated with congestion.

B. Experimental Results

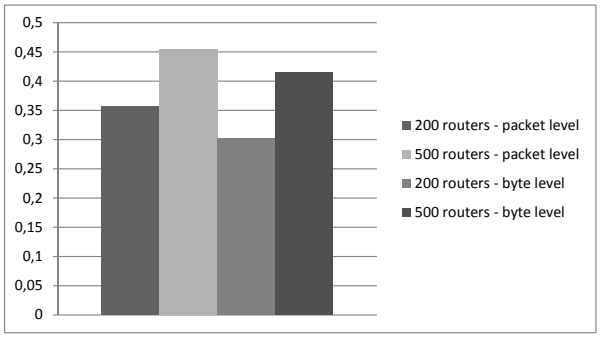

Fig. 2. NAK aggregation rate.

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. Formally, if the subscribers transmit s NAKs and the publisher receives r NAKs, the aggregation rate is defined as $\frac{s-r}{s} = 1 - \frac{r}{s}$, i.e. it shows the fraction of transmitted NAKs that never reach the publisher. As shown in Figure 2, the aggregation rate is higher in the larger topology. This is expected, as a larger network with more subscribers leads to larger trees and the insertion of more relay points, thus increasing the points where NAKs are aggregated. We observe that when counting bytes the aggregation rate is about 5% smaller, since aggregated NAKs are slightly larger.

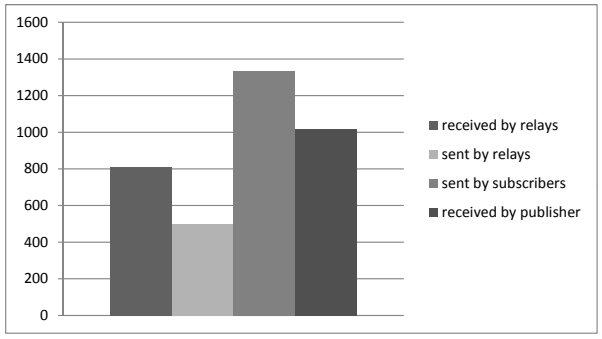

Fig. 3. Number of NAKs handled by architecture entities (200 routers).

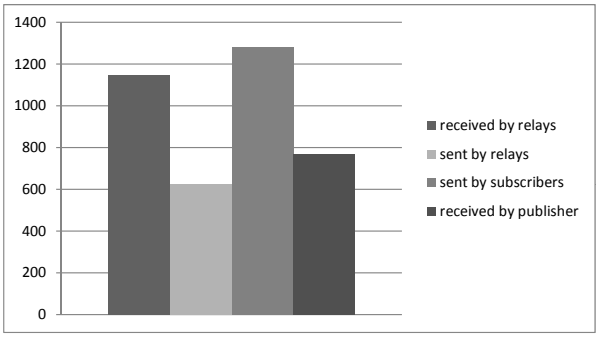

Fig. 4. Number of NAKs handled by architecture entities (500 routers).

In Figures 3 and 4 we provide more details about NAK handling, showing the number of NAKs generated by the subscribers, received and sent by the relays, and received by the publisher *for the entire content transfer*, in the 200 and 500 router topologies, respectively. From these figures we can make multiple observations. First, the number of NAKs generated by the receivers is similar in both topologies, since we ensured that the overall packet loss probability is roughly the same in the initial file distribution phase. 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 lead to more aggregation towards the publisher, as observed in Figure 2.

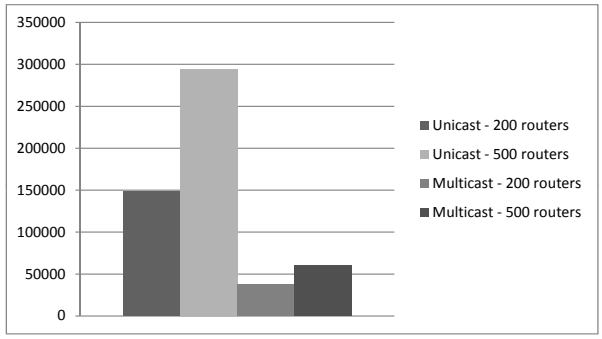

Fig. 5. Amount of feedback traffic handled by network nodes.

Finally, we compare our scheme with unicast error recovery,

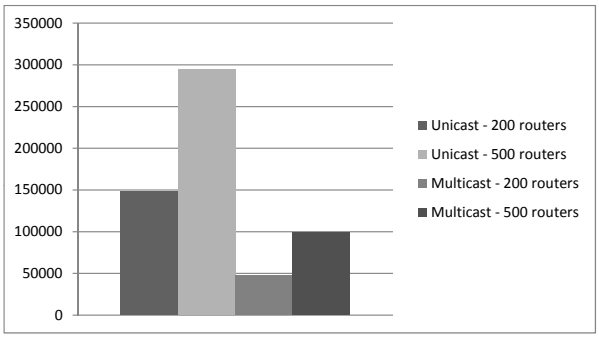

Fig. 6. Amount of recovery traffic handled by network nodes.

i.e. using multicast to deliver the original packets and then switching to unicasting NAKs to the publisher and unicasting recovery packets to each subscriber. Concerning feedback, as shown in Figure 5 the number of NAKs transmitted by all network elements involved (subscriber and all intermediate routers) with our scheme is only 25% and 20% of those needed with unicasting for the 200 and 500 node topologies, respectively. Concerning recovery, as shown in Figure 6 the number of recovery packets transmitted by all network elements involved (publisher and all intermediate routers) with our scheme is 32% and 33% of those needed with unicasting for the 200 and 500 node topologies, respectively. 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 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 per subtree, hence reducing the gap between NAKs and recovery packets. These results show that our mechanism leads to very high gains in both directions compared to unicast recovery.

V. Conclusion and Related Work

In this paper we presented a multicast error control protocol for the reliable delivery of information over a network supporting native multicast, using relay points to extend the reach of the source-routing mechanism used. Our scheme is based on feedback aggregation towards the sender via the relay points and multicast retransmissions of lost data. We explored the performance of the mechanism through simulations using detailed message exchanges, focusing on its feedback aggregation features. Our results indicate that the aggregation mechanism prevents feedback implosion and that multicast retransmissions are far more efficient than unicast ones.

Our mechanism is loosely based on PGM, originally proposed for IP multicast, adapted for the native multicast facility of the PSI architectire. Rather than selecting arbitrary routers for feedback aggregation as in PGM, our scheme exploits the forwarding relays required to scale iBF-based forwarding. Our mechanism targets fully reliable distribution, thus operating in a series of transmission and retransmission phases, unlike PGM which targets mostly reliable distribution, thus mixing new data and retransmissions. Finally, our scheme does not rely on feedback suppression via multicasting NAK confirmations, as PGM does. Most of the extensions proposed for PGM [9] are compatible with our own scheme however.

Future work includes investigating the delay interval for NAK aggregation, which represents a tradeoff between feedback aggregation and completion time. Another direction is extending our scheme with an efficient solution for congestion control over the PSI architecture and studying its performance under correlated losses. Finally, we are planning to examine the effectiveness of caching at relay points to enable local retransmissions of lost data, another idea borrowed from PGM.

Acknowledgment

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: THALIS - Athens University of Economics and Business - DISFER.

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] 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.
[3] 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.
[4] PURSUIT: Publish subscribe Internet technology. [Online]. Available: www.fp7-pursuit.eu
[5] 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.
[6] 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.
[7] DISFER: Distributed sensor systems for emergency response. [Online]. Available: www.aueb.gr/disfer
[8] 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.
[9] J. Gemmell, T. Montgomery, T. Speakman, N. Baskar, and J. Crowcroft, "The PGM reliable multicast protocol," *IEEE Network*, vol. 17, no. 1, pp. 16 – 22, 2003.
[10] 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.
[11] C. Tsilopoulos and G. Xylomenos, "Scaling Bloom filter-based multicast via filter switching," in *Submitted for publication*, 2013.
[12] Ns-3 simulator. [Online]. Available: www.nsnam.org