A Reliable Multicast Transport Protocol for Information-Centric Networks

Charilaos Stais, George Xylomenos, Alexios Voulimeneas
Mobile Multimedia Laboratory, Department of Informatics
Athens University of Economics and Business
Athens 10434, Greece
stais@aueb.gr, xgeorge@aueb.gr, avoulimeneas@gmail.com

Abstract—**In the past few years, many researchers have argued that the Internet should transition from its traditional endpoint-centric architecture to an information-centric paradigm. One of the advantages of the information-centric model is that the network can easily aggregate requests for the same content and serve them via multicast. Indeed, most information-centric architectures proposed to date offer native support for multicast, promising a vast improvement in the efficiency of content distribution. However, designing efficient reliable transport protocols for multicast is a largely open issue, due to the problem of feedback implosion towards the sender as group size grows. In this paper we propose RMTPSI, a retransmission-based reliable error control protocol for multicast communication designed specifically for information-centric networks. We compare RMTPSI with existing approaches proposed for IP multicast and evaluate its performance via simulation, showing that our approach leads to more efficient content distribution and error recovery than previous solutions.**

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

I. Introduction

We have recently experienced a research drive targeting new architectures for the future Internet, aiming to improve the efficiency of large-scale content delivery. Many of these efforts are based on native multicast support, which has long been considered the key for efficient content distribution, but was never widely adopted on the Internet. The design and implementation of the current Internet architecture leans on the traditional telephone network where there are only two parties wishing to communicate. Extending this model to multiparty communication requires considerable engineering effort and costs for network operators. Unfortunately, there is no clear path to multicast adoption aligned with the business models of network operators [1], hence IP multicast is prevalent only inside private networks for specific applications, e.g. IPTV over ADSL networks.

Recent research efforts have tried to move the center of attention from *where* the desired information is located to the *information* itself, since in most applications users are only interested in getting the desired content, not in its location. This direction is evident in *Peer-to-Peer* (P2P) file sharing, *Content Delivery Networks* (CDNs) and cloud computing services, which generally operate as an overlay to the existing Internet. A more radical approach is to introduce a clean-slate *Information-Centric Networking* (ICN) architecture, focusing on content rather than the endpoints hosting and consuming it. By designing a suite of network protocols around information itself, these proposals aim to better satisfy the requirements of current and future content distribution applications [2], [3], [4], [5].

A claimed advantage of the ICN paradigm is that the network can aggregate requests for the same content and serve them via multicast, thus boosting the efficiency of content delivery. While most ICN architectures offer native support for multicast [6], they have not yet addressed the issue of designing efficient reliable transport protocols for multicast, even though considerable work has been performed in this area for IP multicast. In general, reliable multicast can be achieved in two ways: sender-driven with acknowledgments as feedback, and receiver-driven with negative acknowledgments. In sender-driven protocols the sender eventually becomes a bottleneck due to acknowledgment implosion as the number of receivers grows [7]. Therefore, most reliable multicast protocols are receiver-driven, an approach that we also adopt. Our work in this area is based on the ICN architecture of the FP7 EU project PURSUIT [8], referred to as the *Publish/Subscribe Internet* (PSI) architecture. We have previously briefly presented a receiver-driven reliable multicast protocol for the PSI architecture [9]. In this paper, we present our *Reliable Multicast Transport for PSI* (RMTPSI) in detail, contrast it with *Pragmatic General Multicast* (PGM) [10], a reliable multicast transport protocol designed for IP multicast, and evaluate our protocol's performance against PGM over the PSI architecture. Our results show that RMTPSI is more efficient than PGM, while requiring the same time to complete a reliable transfer; specifically, RMTPSI requires 2.9% to 10.2% fewer downstream and 3.5% to 12.1% fewer total transmissions than PGM.

The target application for RMTPSI is fully reliable multicast delivery, for example, distributing OS patches or antivirus updates over the network. In these applications each recipient must receive *all* data correctly, regardless of how long this may take. These applications, besides being extremely common, offer a natural synchronization between senders and receivers: as updates become available, they are transmitted immediately to all waiting recipients. In contrast, in applications such as media distribution, it is either hard to ensure receiver synchronization (e.g. in video on demand) or mostly reliable delivery is sufficient (e.g. in live video streaming).

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

II. Background & Related Work

A. The PSI architecture

A publish/subscribe architecture consists of three elements: publishers, subscribers, and an event notification service, also known as a Rendez-Vous network, consisting of *Rendez-Vous Points* (RVPs) [11]. The publishers are the content owners who offer their content to potential consumers. To announce content availability, publishers advertise it to the responsible RVP by issuing publication messages. The subscribers are the content consumers who express their interest in specific content items by issuing subscription messages. Information indicating the desired content items is included in the publication and subscription messages.

PSI is an instantiation of such a public/subscribe architecture in a networking context: publishers and subscribers are located at network nodes and exchange data via publish and subscribe primitives which are facilitated by a distributed rendezvous function. Data items are identified by a *Scope Identifier* (SId) and a *RVP Identifier* (RId). The SId identifies a collection of content items and is mapped to the RVP responsible for this particular collection, possibly via a *Distributed Hash Table* [12]. The RId identifies a content item within that collection and is determined by the publishing application. The scoping mechanism in PSI is designed to limit access to content, therefore each scope may have different access control rules [13].

A subscriber needs to be aware of the SId/RID pair of a desired content item to issue a subscription message for it. When a subscription message arrives at the RVP corresponding to the SId in the subscription, the RVP checks whether the subscriber can access the scope. If so, it determines which publishers can satisfy the subscriber's request and then communicates with the *Topology Manager* (TM) to request a suitable forwarding path from a publisher to the subscriber. The TM, either a service in the same machine or a stand-alone server, maintains network topology information discovered via a link-state routing protocol. The TM can thus calculate a path between the publisher and the subscriber; when multiple subscribers are interested in the same content item, a multicast tree containing all subscribers is calculated.

The path calculated by the TM is described by a Bloom filter, as in LIPSIN [14]. Bloom filters are probabilistic representations of sets where each element is encoded as a string of zeroes and ones, calculated via a set of hash functions. A set is represented as the logical OR of all its elements. In PSI, each link is labeled with one such string in each direction. A Bloom filter in the header of each packet includes the labels of all the links that are part of the desired path. When a packet arrives at a router, the router determines to which of its outgoing links (possibly, more than one) it will have to forward the packet, by performing a logical AND between the label of each link and the in-packet Bloom filter. This technique supports native multicast, since the Bloom filter in the packet header may represent an entire multicast tree; the Bloom filter is simply a set of link labels. Link labels *must* be unidirectional, as otherwise packets would loop, hence the encoded paths, whether unicast or multicast, are also unidirectional.

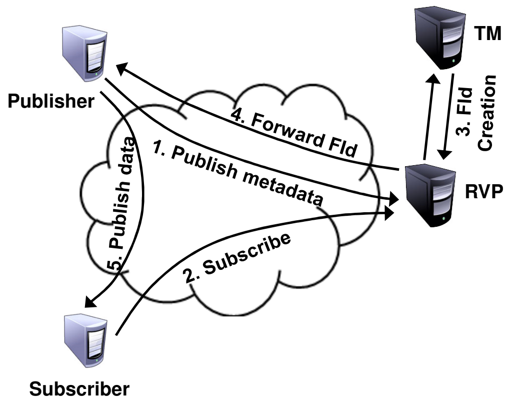


Fig. 1: Communication steps in the PSI architecture.

Figure 1 summarizes the above procedures. First, a publisher issues a publication under a certain SId/RId to the corresponding RVP (step 1). A subscriber that is aware of this SId/RId pair subscribes to it (step 2). After the RVP corresponding to the requested SId receives the subscription, it communicates with the TM in order to retrieve a suitable Bloom filter for data dissemination from the publisher to the subscriber (step 3). Once the RVP gets the Bloom filter, it forwards it to the publisher (step 4). Finally, the data are delivered to the subscriber using the Bloom filter (step 5).

B. Wide Area Multicast in PSI

As more elements are added to a Bloom filter, it becomes more likely that it will match elements not added to it; these are *false positive* matches. When Bloom filters are used to encode routes as in PSI, as more links are added to a route it is more likely that random links may match them. If such a link happens to be on the path taken by a packet, the packet will be needlessly transmitted over that link. The extent of this problem depends on how many links are encoded into the set. In [15], the authors argue that the number of ones in the Bloom filter should not exceed 40% of the total bits, meaning that with reasonably sized Bloom filters (as they must fit within packet headers) we cannot represent very large groups or very long paths.

To scale this scheme to larger multicast groups, we employ Bloom filter switching at designated *relay points* (RPs) [16]. RPs are routers that replace the Bloom filter inside a packet with a new one before forwarding the packet. When a packet arrives, the RP checks if an entry for the SId/RId pair in the packet exists and, if so, replaces the Bloom filter in the packet with a stored one. When the TM constructs the initial Bloom filter, it pays attention to the ratio of ones in it. If the ratio exceeds 40%, it resorts to RPs, selected during a breadth-first traversal of the multicast tree. The TM therefore constructs Bloom filters from the publisher to the RPs and from these RPs to the subscribers or to new RPs, recursively. The RPs and the Bloom filters are returned to the RVP, which then notifies the publisher and the RPs.

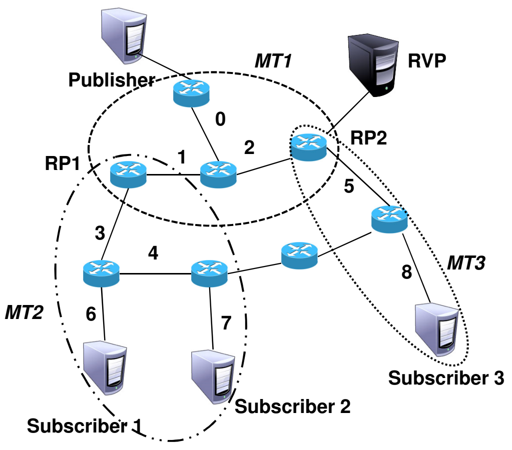


Fig. 2: A multicast tree with two relay points.

Figure 2 presents an example of this approach. The multicast tree is broken in three subtrees, MT1 to MT3, connected via RP1 and RP2. MT1 starts from the publisher and ends at the two RPs, MT2 starts at RP1 and ends at subscribers 1 and 2, and MT3 starts at RP2 and ends at subscriber 3. Messages are initially transmitted by the publisher using the Bloom filter for MT1. Upon arrival at RP1 (RP2), this Bloom filter is replaced by the corresponding one for MT2 (MT3). As a result, each RP needs to store a mapping from a SId/RId pair to a new Bloom Filter. We are using the SId/RId pair to determine the next Bloom filter at RPs, so as to allow the Bloom filters between two RPs to evolve due to link additions and deletions, without updating the switching information at all RPs.

C. Reliable Multicast

Even though IP multicast was never widely deployed, considerable work was dedicated to transport layer protocols for reliable multicast. The *Reliable Multicast Transport Protocol* (RMTP) [17] is a receiver-driven reliable transport scheme for non-real-time multicast content delivery. It relies on selective *acknowledgments* (ACKs), possibly indicating multiple lost packets, which are periodically sent from each receiver towards the sender. In order to avoid sender implosion due to ACKs, a set of receivers, called *Designated Receivers* (DR), aggregate the ACK state and forward it upstream, to higher level DRs or to the sender. Subsequently, retransmissions from the sender are forwarded by each DR towards its group members and lower level DRs. Retransmissions may be either multicasted or unicasted, depending on the number of receivers that have lost a packet. In order to map this scheme to PSI, the TM should use the DRs as RPs in order to extend the reach of the multicast tree, but this would lead to suboptimal routing, due to the need to select actual receivers as RPs; additional RPs are still needed if there are no receivers at appropriate points in the multicast tree.

The *Scalable Reliable Multicast* (SRM) [18] approach on the other hand relies on the fact that in IP multicast anyone can send to a multicast group, hence receivers can multicast their *negative acknowledgments* (NAKs) for missing data so as to reach other nodes that could retransmit them. By limiting the reach of NAKs to a few hops, ideally only nearby receivers will attempt to respond with the missing data. By randomizing the response time to these NAKs and locally multicasting the missing data, the first receiver to respond will silence all others. This approach does not guarantee reliability and is very hard to map to PSI, since it relies on bidirectional multicast trees to allow each receiver to multicast NAKs and retransmissions. Since in PSI multicasting is unidirectional, this would require creating separate Bloom filters from each subscriber to all other nodes within a subtree.

Finally, in the *Pragmatic General Multicast* (PGM) [10] approach, a receiver in the multicast group is guaranteed to either receive all data packets (from their original transmission or a retransmission), or to be able to detect unrecoverable data packet loss. PGM relies on NAKs to report missing packets and a hierarchy of PGM-enabled routers, called *Network Elements* (NEs), 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 in PGM, 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. The NE then pushes the NAK towards its own parent, which in turn multicasts an NCF to its own tree, and so on, until the NAK reaches the source, at which point the missing packet is retransmitted downstream. Only NEs that have received a NAK for this packet forward it downstream, therefore retransmissions only reach subtrees where at least one receiver has reported that packet to be lost. To further reduce the number of NAKs, an NE can create NAK packets which indicate multiple missing packets, thus aggregating individual NAKs.

PGM can be easily mapped to the PSI architecture, by simply using the RPs as the NEs. The only additional requirement is to provide each receiver and RP with a Bloom filter for reaching its parent RP or the publisher, so as to transmit NAKs; NCFs and retransmissions can reuse the multicast Bloom filter used for the original packet transmissions. While PGM in its pure form is most appropriate for semi-reliable continuous data transmission, with minor modifications it can be made to operate in fully reliable mode. Since PGM is a more sophisticated version of RMTP, adding NAK suppression via NCFs, we focus below on comparing our approach with PGM only. 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 for PSI

A. Overview

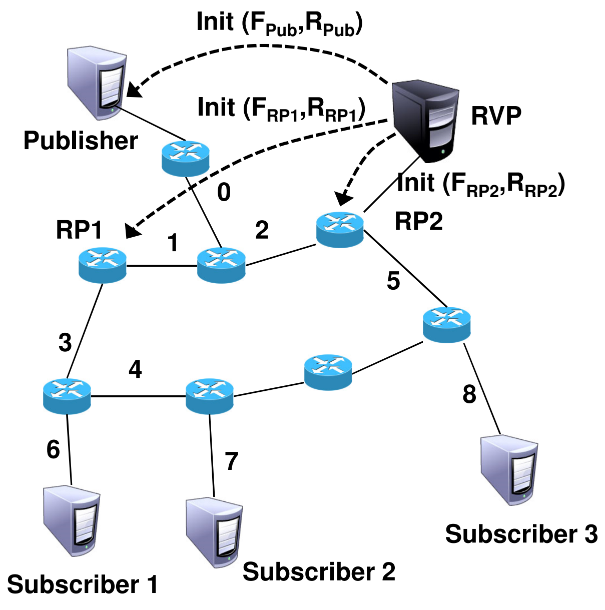
RMTPSI is most similar to PGM, in that it uses selected routers *on the multicast tree*, as opposed to the DRs in RMTP, to aggregate feedback and control the propagation of retransmissions. To minimize the nodes that need to manage multicast state, in place of the NEs used in PGM we reuse the multicast *relay points* (RPs) mandated by the forwarding architecture of PSI when trees grow large (see Section II-B). As a result, each multicast subtree created by the TM independently manages the error recovery process, thus avoiding feedback implosion at the original receiver.

RMTPSI operates in three phases: communication setup, where the multicast subtrees are created and the appropriate Bloom filters are distributed, initial content distribution, where the publisher sends the entire content and collects aggregate feedback, and recovery, which may involve one or more cycles of initiating retransmissions and collecting new aggregate feedback, until all subscribers have completed the download successfully. We elaborate on these phases in the following subsections.

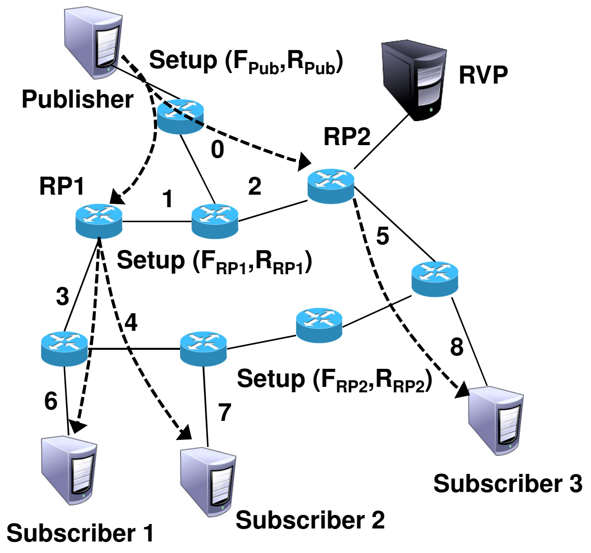
B. Communication setup

In the communication setup phase, the TM creates the multicast delivery tree, splits it into subtrees, creates the appropriate Bloom filters and distributes them to the RPs. As already mentioned, Bloom filters operate only in one direction, therefore we need separate Bloom filters in order to return feedback from the subscribers towards the publisher. We therefore modified the TM to calculate not only downstream Bloom filters from the root to the leaves of each multicast subtree, but also upstream filters, 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, therefore they can be used to send upstream messages from *any* leaf to the root. The advantage of this approach is that only a single Bloom filter needs to be created and communicated to all the receivers in the subtree. The creation of the subtrees is performed via a breadth-first traversal of the overall tree; when the Bloom filter for a subtree contains enough 1's, the subtree stops there and an RP is created. Since the same number of links is entered in both the forward and reverse Bloom filters, the number of 1's and the false positive probability is roughly the same in both directions.

The set of forward and reverse Bloom filters for the publisher and each RP are then sent to the RVP, which in turn sends to the publisher and *each* RP both a downstream Bloom filter, used for data forwarding, and an upstream Bloom filter, used for feedback *within its own subtree*. The publisher then sends a setup message to initiate the content distribution; this message includes the upstream Bloom filter that should be used for feedback within its subtree and a round counter set to zero. Upon reception of this message, each RP stores the upstream Bloom filter used to reach its parent. Then, the RP switches the Bloom filter in the packet with the one it received from the RVP and forwards the setup message, encapsulating inside it the upstream Bloom filter that should be used by its own children for feedback. At the end of this process, each RP knows the Bloom filter needed to reach its children (from the original message of the RVP) and each RP and receiver knows the Bloom filter needed to reach its parent (from the setup message from the parent) for a specific SId/RId pair.


Init (F_{Pub},R_{Pub})
Init (F_{RP1},R_{RP1})
Init (F_{RP2},R_{RP2})
Publisher
RVP
RP1
RP2
0
1
2
3
4
5
6
7
8
Subscriber 1
Subscriber 2
Subscriber 3

(a) Initialization messages from RVP.


Setup (F_{Pub},R_{Pub})
Publisher
RVP
RP2
RP1
Setup (F_{RP1},R_{RP1})
Setup (F_{RP2},R_{RP2})
0
1
2
3
4
5
6
7
8
Subscriber 1
Subscriber 2
Subscriber 3

(b) Setup messages from publisher.

Fig. 3: Communication setup in RMTPSI.

Figure 3 explains how the communication setup phase works, using the same example as in Figure 2. The RVP first sends a pair of filters (F_{Pub}, R_{Pub}) to the publisher, a pair (F_{RP1}, R_{RP1}) to RP1 and another pair (F_{RP2}, R_{RP2}) to RP2, to initialize their state, as shown in Figure 3a. The setup message from the publisher is then 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 filter to 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}, as shown in Figure 3b. Each receiver, upon reception of this message, stores the filter encapsulated inside the message to be later used for feedback. In this way, the multicast tree from publisher to subscribers is divided into three smaller ones (MT1-MT3), and each node knows the downstream and upstream Bloom filters needed for protocol operation.

If a setup message is lost, some of the RPs and subscribers forming the leaves of the subtree where the message was lost will not receive an upstream Bloom filter to reach the publisher or the RP at the root of their subtree. As a result, while RPs will still be able to forward and subscribers will be still able to receive data packets, as these only rely on the forward Bloom filters distributed by the RVP, they will not be able to generate feedback. However, RMTPSI will still operate correctly in this case (see Section III-D).

C. Initial content distribution

After setup completes, the publisher starts sending the data packets, inserting in each header the round counter from the setup message. At each RP, the SId/RId in the packet is looked up and the Bloom filter is replaced with the one needed to reach the next RPs and/or subscribers. For example, in Figure 3 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 the upstream Bloom filter to return a NAK message to its upstream RP, also inserting the round counter in the header. The receiver then enters the missing packet and round counter in a list of pending packets, so as to be able to detect when the transmission is complete. The RP also enters the missing packet and its round counter in a pending packet list, but instead of passing the NAK upstream, it holds it for a specified interval waiting for more NAKs to come. If additional NAKs, either for the same or for different packets, arrive at the RP during the waiting period, they are recorded in the pending packet list and combined into a single NAK which is forwarded upstream, by using the corresponding upstream Bloom filter. Finally, the publisher also enters the information from the NAKs it receives in a pending packet list.

Each RP uses the pending packet list in order to later forward recovery data only where needed and to avoid relaying NAKs for packets that have already been NAKed. For example, say that in Figure 3 RP1 receives a NAK from one of the receivers in its subtree. If a NAK for this packet has not already been received, the packet will be entered in the pending packet list. After the waiting interval, the NAK will be forwarded to the publisher, possibly aggregated with other NAKs. When the publisher later retransmits the missing packet using its forward Bloom filter, the retransmission will reach both RP1 and RP2, but only RP1 will forward it in its subtree as the packet is in its pending list, while RP2 will drop it.

D. Recovery

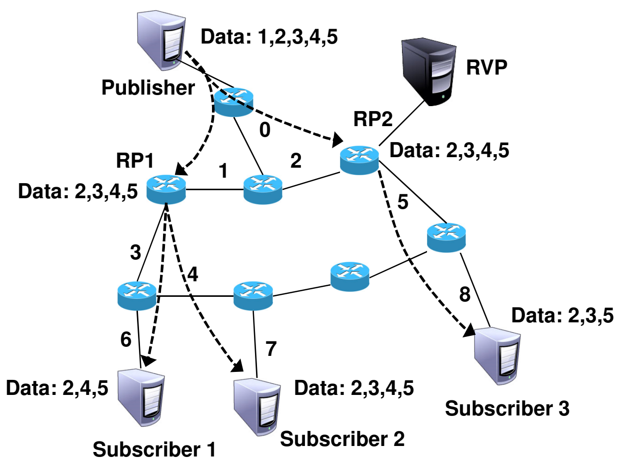
When the publisher finishes the initial content distribution, it waits for a specified period of time in order to allow nodes that have received the entire transmission to leave the multicast group and the TM to issue new Bloom filter pairs wherever needed. The publisher then sends a new setup message with the round counter increased by one, so as to distinguish packets from different recovery cycles. This message lets subscribers know that the current round has finished, allowing them to detect any packets that were lost at the end of the current round; these packets will be requested as part of the next round. In addition, the setup message updates the upstream Bloom filters throughout the multicast tree, as shown in Figure 3b.

Then, a retransmission round begins, with the publisher transmitting all packets for which NAKs have been received, based on its pending packet list. Each RP only forwards in its subtree the packets for which it has received NAKs, as recorded in its own pending packet list. As packets are retransmitted by the publisher and the RPs, the corresponding entries in their pending packet lists are cleared; the same takes place at each receiver as missing packets arrive. Again, receivers send NAKs for missing packets, which are aggregated as previously explained. Both data and NAK packets use the round counter from the most recent setup message. This procedure (setup, transmission, feedback) is repeated at the end of every round, until the download completes at all receivers. A receiver knows that its download is done when a round completes and its pending packet list is empty. Since subscribers leave the multicast group when done, the tree will eventually be torn down, thus concluding the transfer.

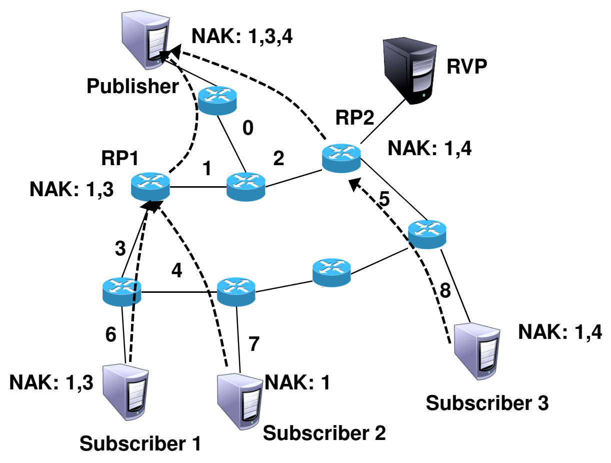
Consider now what happens if a setup message is lost in a subtree. The leaves of that subtree, whether they are RPs or subscribers, will not be able to transmit NAKs upstream but, since they are still able to receive (and forward, if they are RPs) data packets, they can simply note that they have not sent a NAK for some packets in their pending packet list. When a new setup message arrives in a following round, they can send these pending NAKs at that point. Since the multicast tree is only torn down when all receivers leave the multicast group, the publisher will keep generating setup messages until the transmission is complete, even if it has received no NAKs in a round (presumably, due to missing setup messages or NAKs). As a result, missing setup messages will be eventually repaired, RPs and subscribers will send their NAKs, and RMTPSI will work correctly.

E. A detailed example

Consider again the network shown in Figure 2 and assume that the publisher generates 5 packets, numbered 1 to 5, as shown in Figure 4a. Packet 1 is lost over link 0, hence RP1 and RP2 only receive packets 2, 3, 4 and 5. RP1 transmits these packets to subscriber 1 and subscriber 2, but packet 3 is lost over link 6, hence subscriber 1 ends up with packets 2, 4 and 5 and subscriber 2 with packets 2, 3, 4 and 5. RP2 also transmits the same packets, but packet 4 is lost over link 8, hence subscriber 3 ends up with packets 2, 3 and 5. Subscriber 1 thus generates NAKs for packets 1 and 3, subscriber 2 generates a NAK for packet 1 and subscriber 3 generates NAKs for packets 1 and 4, as shown in Figure 4b. RP1 combines the NAKs from subscriber 1 and subscriber 2 to an aggregated



(a) Data packets sent and received.



(b) NAKs sent and received.

Fig. 4: An example of data transmission and recovery.

NAK for packets 1 and 3, while RP2 combines the NAKs from subscriber 3 to an aggregated NAK for packets 1 and 4. As a result, the publisher will receive NAKs for packets 1, 3 and 4, which it will retransmit towards RP1 and RP2. RP1 will only forward packets 1 and 3 to its subtree, while RP2 will only forward packets 1 and 4 in its subtree. Note that RP1 drops packet 4, RP2 drops packet 3 and subscriber 3 drops packet 3, as they have not asked for them. All subscribers will now leave the multicast group, as they have no pending packets, and the tree will be torn down, thus completing the transmission.

As a further example, assume that the retransmission of packet 4 in the previous example was lost again. Since no later messages will arrive, subscriber 3 will not generate a NAK for packet 4 in this round. However, since subscriber 3 still has pending packets, it will not leave the multicast group and the tree will not be torn down. Eventually, the publisher will send a setup message with a new round counter, making subscriber 3 aware that packet 4 was lost again during the previous round. Subscriber 3 will thus send a new NAK for packet 4 with the current round number, and the publisher will eventually retransmit packet 4 during the next round.

F. Comparison with PGM

As mentioned above, RMTPSI is most similar to PGM, in that feedback is aggregated in tree nodes, effectively splitting the multicast distribution tree into subtrees. While in PGM a single IP multicast address is used for the entire tree and the subtrees are used only for feedback aggregation purposes, in our approach we reuse the RPs mandated by the forwarding architecture for this purpose. Also, while in PGM NAKs are sent via unicast to the upstream aggregation point, we instead use reverse Bloom filters to achieve the same result, since our forwarding architecture is based on source routing. Essentially, these modifications are required to map PGM from IP multicast to the PSI forwarding architecture.

A more significant difference is the way feedback is aggregated. In PGM, the RP attempts to suppress further NAKs for a packet by multicasting an NCF, while in RMTPSI each receiver sends NAKs for all missing packets to its upstream RP; there is no NAK suppression within a subtree. 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 for each loss; even if another NAK for the same packet is sent, the multicast NCF may still be more costly, as it crosses all links in the subtree. Another difference is that PGM sends new data interspersed with retransmissions, eventually giving up on lost packets, while RMTPSI retransmits packets in rounds until all have been received. This reflects the fact that RMTPSI is designed for fully reliable transmission, while PGM is geared towards mostly reliable transfers. Since in each round some group members depart, the number of links that each retransmitted packet crosses in RMTPSI is generally lower than that of PGM.

IV. Experimentation and Evaluation

A. Simulator Setup

In order to evaluate the performance of RMTPSI against PGM, we used extensive simulations over NS-3 [19], where the entire PSI architecture was implemented, including Bloom filter-based forwarding [14] and wide-area multicasting based on RPs [16]. We simulated a scenario where a single publisher distributes the desired content (e.g. an OS patch or an antivirus update) to a large set of subscribers. The content size is 20 MB, composed of 20.000 data packets with a payload of 1 KB each. We used randomly generated scale-free network topologies of 200 and 500 routers (generated with the Barabási-Albert algorithm [20]) with 50 and 100 subscribers attached to randomly chosen routers, leading to an average of 15.6 and 28.6 RPs, respectively; smaller topologies can be handled without RPs. Each scenario was executed 7 times, with different random positions of attachment to the network for the publisher, resulting in a different tree being generated and different RPs being chosen each time.

We assumed that all losses were random, i.e. packets were independently lost with the same probability in each link; we did not model the correlated losses usually associated with congestion. The values for the link loss probability were chosen experimentally, so that in each topology the average loss rate *reported to the publisher* would be 3, 6, 9 or 12%. Packet losses affected both data packets and NAKs, but not the initialization packets from the RVP to the publisher and RPs, which are beyond the scope of our work.

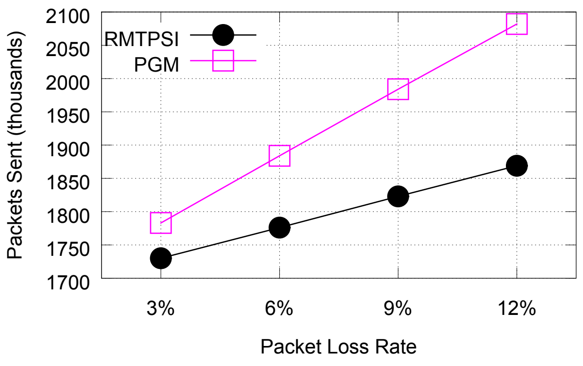
Concerning the PGM protocol, all its features described above have been implemented for the PSI architecture, using RPs to extend the size of the multicast trees and reverse Bloom filters for feedback. PGM uses these RPs as NEs, hence RMTPSI and PGM employ the same multicast trees and subtrees. When a PGM subscriber notices a loss, it picks a random interval from 0 to 1000 ms before sending a NAK, while RMTPSI subscribers send their NAKs immediately. When a NAK is received, the RP in both PGM and RMTPSI waits for 70 ms before passing it upstream, aggregating all NAKs received in the meantime into a single NAK. In order to make PGM fully reliable, each subscriber starts a 500 ms timer upon sending a NAK, waiting for an NCF. If the timer expires before an NCF arrives, the NAK is retransmitted and the timer is restarted. Upon arrival of an NCF, the subscriber starts a new 500 ms timer, waiting for the missing packet. If the timer expires before the packet arrives, the NAK is retransmitted and the timer is restarted, until the packet is eventually received.

B. *Experimental Results*

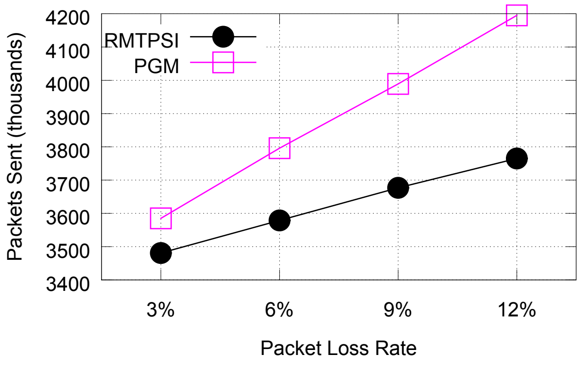
Our first metric is the total number of packets transmitted in the publisher to subscriber direction, for both the initial data distribution and the recovery phases; we count all single hop packet transmissions, that is, a multicast transmission from the publisher or an RP that crosses a tree with n links counts as n transmissions. We did not include the setup messages as they represent a negligible fraction of the total: only one setup message crosses each link in every round, compared to 20.000 data packets sent in the first round only. Figure 5 presents this metric for RMTPSI and PGM, for the 200 and 500 router topologies (the x-axis is in thousands). The size of the network changes the number of total packet transmissions required, as there are both more receivers and more intermediate routers in the larger topology, but the overall trends are the same: RMTPSI requires 2.9% to 10.2% fewer downstream transmissions than PGM, with higher benefits as the loss rate grows.

In order to isolate the performance of the recovery mechanism, in Figure 6 we show the total number of packets transmitted in the publisher to subscriber direction *only for error recovery purposes*, that is, the retransmissions and, in the case of PGM, the NCFs. Again the trends are very similar for both topologies, although the benefits from RMTPSI are far more evident when the regular data transmissions are removed: RMTPSI requires 50.1% to 53.2% fewer recovery packets in the downstream direction than PGM, a nearly constant reduction across the range of loss rates tested.

The reason for this big gap can be understood by Figure 7, which shows the number of packets sent *directly by the publisher* for recovery purposes only. While PGM retransmits roughly the same number of data packets as RMTPSI when NCFs are excluded, the NCFs roughly double the downstream



(a) 200 routers



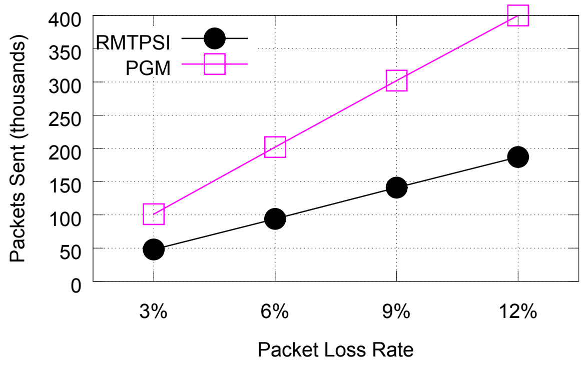
(b) 500 routers

Fig. 5: Number of packets sent downstream by all network elements.

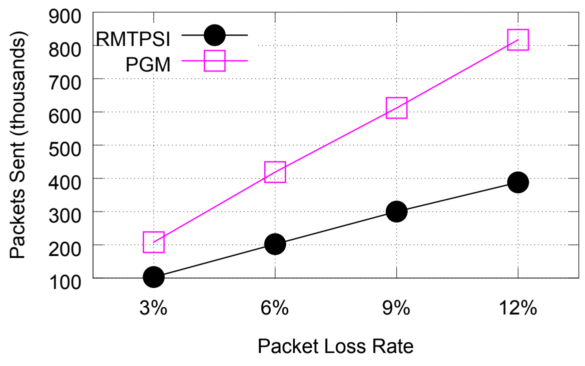
recovery traffic. In PGM, the first NAK for each packet in a subtree triggers an NCF; in most cases, only one leaf of the subtree will have lost that packet, even at the highest loss rates tested in our simulations. As a result, the number of NCFs is roughly equal to the number of NAKs and the number of retransmissions, thus the number of packets sent for recovery purposes nearly doubles in PGM. The same holds for all routers, thus explaining the large difference in favor of RMTPSI in Figure 6.

Since error recovery also requires traffic in the upstream direction, in Figure 8, we show the total number of packets transmitted in both the downstream and upstream directions; again, we count all single hop packet transmissions, as in Figure 5, but this time we also include NAKs. Upstream traffic is equal to 6.7% to 24.8% of the downstream traffic for RMTPSI (7.2% to 27% for PGM), representing a considerable overhead in terms of packets, especially as loss rates grow. RMTPSI requires 3.5% to 12.1% fewer packet transmissions than PGM in both directions, a higher improvement than when only downstream transmissions are counted.

The increased savings with RMTPSI when both directions are taken into account can be explained by Figure 9, which shows how NAK aggregation performs in PGM and RMTPSI; specifically, the figure shows how many NAKs are transmitted by all subscribers and how many NAKs are finally received by the publisher. The reason for the larger number of NAKs generated by the subscribers in PGM is NAK retransmissions

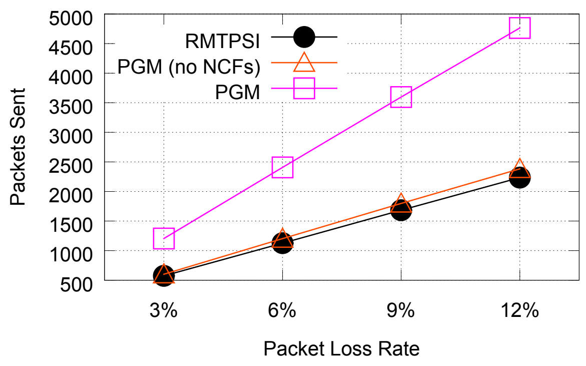


(a) 200 routers

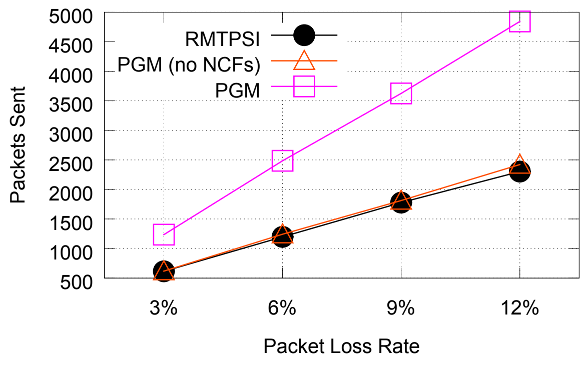


(b) 500 routers

Fig. 6: Number of recovery packets sent downstream by all network elements.



(a) 200 routers



(b) 500 routers

Fig. 7: Number of recovery packets sent downstream by the publisher.

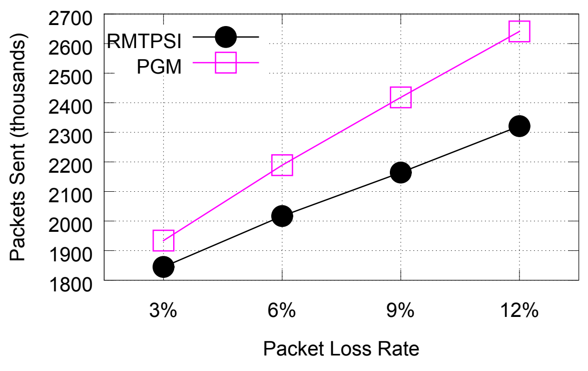
due to timer expirations, when either the NCF or the missing packet is not received on time. Since in PGM both NCF and retransmitted data packets compete with regular data packet transmissions, these delays can be quite large. To reduce NAK retransmissions as much as possible in PGM, in the experiments reported here we increased the timeout delay up to the point where the download duration started to increase. On the other hand, in RMTPSI there is no need for timeouts: if a missing packet has not been received by the end of the round in which the corresponding NAK was sent, a new NAK is sent for that packet in the next round, thus NAKs are retransmitted only when necessary. On the other hand, PGM is more effective than RMTPSI in aggregating these NAKs, since fewer NAKs reach the publisher. Despite this fact, the total number of NAK transmissions is smaller in RMTPSI, as shown above. We also notice that aggregation is more efficient in the larger network topology. This is because a larger network leads to larger trees and the insertion of more relay points. By increasing the points where NAKs are aggregated, both mechanisms work more efficiently.

Finally, we measured the time needed for both mechanisms to complete the transmission to all subscribers, including recovery. While we expected that our approach would need more time to complete than PGM due to our recovery round structure, we observed that both approaches were ending almost simultaneously in all cases. This happens because the publisher in PGM has to interrupt the data transmission in order to send NCFs and retransmissions in parallel with new data; in addition, as noted above, in PGM many NAKs are retransmitted due to timeouts, thus further reducing the useful throughput of PGM as packets are needlessly retransmitted. RMTPSI in contrast always sends at full speed, since NAKs are gathered during one round and their retransmissions are taking place in the next round; NAKs are retransmitted only after the end of a round. In all our tests, full recovery in RMTPSI required two rounds of retransmissions, with three rounds needed only in a few of the repetitions at the highest loss rate.

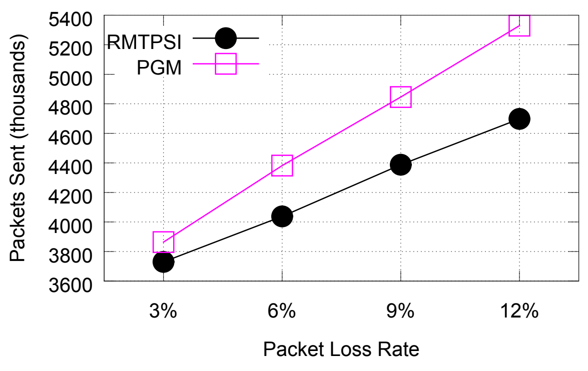
V. Conclusion and Future Work

In this paper we presented an approach for multicast error control for the reliable, on-demand, delivery of information over a network supporting native multicast, using relay points to extend the reach of the source-routing mechanism used. Our RMTPSI scheme is based on feedback aggregation towards the sender via the relay points and multicast retransmissions of lost data. We explored the performance of RMTPSI through simulations using detailed message exchanges, focusing on its feedback aggregation features. Our results indicate that the aggregation mechanism effectively prevents feedback implosion, especially in larger graphs.

RMTPSI is loosely based on PGM, originally proposed for IP multicast, adapted to operate over the native multicast support of the PSI architecture. Rather than selecting arbitrary

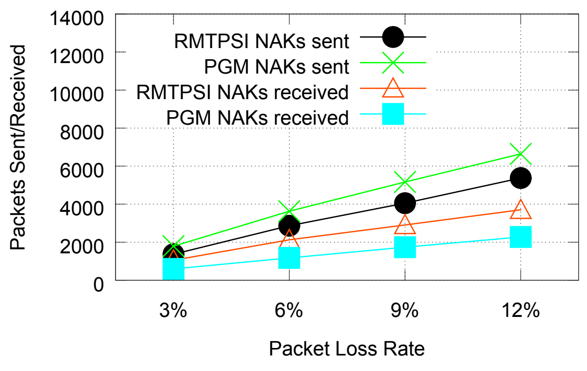


(a) 200 routers

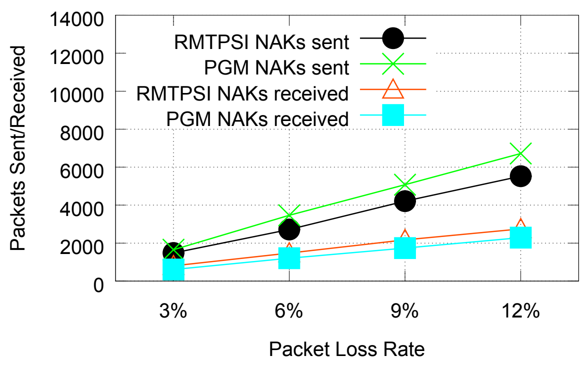


(b) 500 routers

Fig. 8: Total packets transmitted by all network elements.



(a) 200 routers



(b) 500 routers

Fig. 9: Total NAKs sent by subscribers and received by publisher.

routers for feedback aggregation as in PGM, RMTPSI uses the forwarding relays required to scale Bloom filters to large multicast groups for this role. RMTPSI targets fully reliable distribution, thus operating in a series of transmission and retransmission rounds, unlike PGM which targets mostly reliable distribution, thus mixing new data and retransmissions. RMTPSI does not rely on feedback suppression via multicasting NAK confirmations, as PGM does. Most of the extensions proposed to PGM however [10] are compatible with our work. Our results show that RMTPSI is more efficient than PGM, while requiring the same time to complete a reliable transfer as PGM; specifically, RMTPSI requires 2.9% to 10.2% fewer downstream transmissions and 3.5% to 12.1% fewer total transmissions than PGM.

Future work includes an investigation of the delay interval that a relay point should wait for NAK aggregation, which represents a tradeoff between feedback traffic and completion time. We also plan to couple the error control scheme of RMTPSI with an efficient solution for hierarchical 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, another idea borrowed from PGM.

ACKNOWLEDGEMENTS

This research has beenco-financed by the European Union (European Social Fund–ESF) and Greek national funds through the Operational Program “EducationandLifelongLearning” 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] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, “ROFL: routing on flat labels,” in *Proc. of the 2006 ACM SIGCOMM*, 2006, pp. 363–374.
[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 2007 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 2009 ACM CoNEXT*, 2009, pp. 1–12.
[5] G. Parisis, D. Trossen, and D. Syrivelis, “Implementation and evaluation of an information-centric network,” in *Proc. of the 2013 IFIP Networking*, 2013, pp. 1–9.
[6] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, “A survey of information-centric networking research,” *Communications Surveys Tutorials, IEEE*, vol. 16, no. 2, pp. 1024–1049, 2014.
[7] S. Pingali, D. Towsley, and J. F. Kurose, “A comparison of sender-initiated and receiver-initiated reliable multicast protocols,” in *Proc. of the 1994 ACM SIGMETRICS*, 1994, pp. 221–230.
[8] PURSUIT Project, “Home page,” www.fp7-pursuit.eu, 2013.
[9] C. Stais, A. Voulimeneas, and G. Xylomenos, “Towards an error control sceme for a publish/subscribe network,” in *Proc. of the 2013 IEEE ICC*, 2013, pp. 3743 –3747.

[10] J. Gemmell, T. Montgomery, T. Speakman, and J. Crowcroft, “The PGM reliable multicast protocol,” *IEEE Network*, vol. 17, no. 1, pp. 16 – 22, 2003.

[11] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many faces of publish/subscribe,” *ACM Computing Surveys*, vol. 35, pp. 114–131, 2003.

[12] K. V. Katsaros, N. Fotiou, X. Vasilakos, C. N. Ververidis, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, “On inter-domain name resolution for information-centric networks,” in *Proc. of the 2012 IFIP Networking*, 2012, pp. 13–26.

[13] 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.

[14] P. Jokela, A. Zahemszky, S. Arianfar, P. Nikander, and C. Esteve, “LIPSIN: line speed publish/subscribe internetworking,” in *Proc. of the 2009 ACM SIGCOMM*, 2009, pp. 195–206.

[15] 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 2011 IEEE INFOCOM*, 2011, pp. 2399 –2407.

[16] C. Tsilopoulos and G. Xylomenos, “Scaling Bloom filter-based multicast via filter switching,” in *Proc. of the 2013 IEEE ISCC*, 2013, pp. 548 –553.

[17] J. Lin and S. Paul, “RMTP: a reliable multicast transport protocol,” in *Proc. of the 1996 IEEE INFOCOM*, 1996, pp. 1414–1424.

[18] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang, “A reliable multicast framework for light-weight sessions and application level framing,” in *Proc. of the 1995 ACM SIGCOMM*, 1995, pp. 342–356.

[19] NS-3 Simulator, “Home page,” www.nsnam.org, 2013.

[20] A. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, pp. 509 –512, 1999.