

# Efficient Real-time Information Delivery in Future Internet Publish-Subscribe Networks

Christos Tsilopoulos, Ioannis Gasparis, George Xylomenos and George C. Polyzos

Mobile Multimedia Laboratory, Department of Informatics

Athens University of Economics and Business, Athens, Greece

Email: tsilochr@aueb.gr, ioannis.gasparis@cs.aueb.gr, xgeorge@aueb.gr, polyzos@aueb.gr

**Abstract**—In this paper we explain how efficient delivery of real-time information can be supported in the Publish Subscribe Internet (PSI), a network architecture proposal for the Future Internet. PSI departs from IP thinking with respect to the core abstractions made and the functional organization of the system. PSI places information at the heart of the network layer and decouples the forwarding, path formation and topology management functionalities. This design approach can be highly beneficial for real-time communications, as it enables the network to apply sophisticated mechanisms for multicast tree construction, such as delivery of information over optimal (minimum cost) Steiner trees. Initial experiments with a proof-of-concept implementation of PSI indicate the feasibility of realizing such optimization policies. Our results show that significant bandwidth savings can be achieved at the cost of small, un-noticeable to the end-users, delays in flow establishment.

**Index Terms**—Future Internet design; Information-Centric Network architectures; real-time delivery; multicast

## I. INTRODUCTION

Information-Centric networking (ICN) is increasingly attracting the attention of the networking research community. Motivated by the observation that Internet usage is much more content-centric than its initial design had ever anticipated, ICN sees internetworking as a means to interconnect information, rather than interconnecting physical machines [1], [2].

Ongoing ICN efforts have mostly focused on optimizing information delivery through caching [1]. In this direction, researchers study how to adjust routing and forwarding protocols in order to further exploit in-router memory and in-network storage. The merits of caching however, apply to a certain type of applications: pull-based bulk data transfers. On the other hand, the Internet is widely being used for disseminating real-time information such as live media streaming (voice and video conference applications, web radio and live TV) and real-time notifications (sensor measurements, notification alerts, twitter updates etc). This is a class of applications in which the benefits obtained from caching are questionable. For example, the existence of large in-network caches has little impact on the quality of a voice conversation, since the short-lived voice packets may not be worth caching. Real-time applications are publish/subscribe in nature (a data source transmitting data to a set of synchronized receivers) and can highly benefit from efficient multicast delivery schemes. We therefore argue that ICN proposals should widen their application domain beyond caching services -which are suitable for bulk-data transfers- and consider the realization of

efficient multicast schemes -which are suitable for real-time publish/subscribe applications.

In this paper we examine the support for efficient multicast delivery in the Publish-Subscribe Internet (PSI) architecture [3], an ICN proposal for the Future Internet. PSI departs from IP thinking with respect to the core abstractions made and the functional organization of the system. PSI introduces information resolution functionalities as a core component of the network layer and promotes a clear separation between packet forwarding, path formation and topology management functionalities. The design approach makes it feasible to realize sophisticated mechanisms in multicast tree construction, specifically by delivering information over optimal (minimum cost) Steiner trees with minimal signaling overhead. We support our arguments with preliminary evaluations through a proof-of-concept prototype deployed in PlanetLab, as well as in an emulated network. Experimental results with a live streaming application indicate about 30% savings in bandwidth usage, with a small trade-off in increased, but un-noticeable to the user, delay in flow-establishment.

The remainder of the paper is organized as follows: in Section II we present the functional aspects of the PSI architecture and discuss the design differences compared to IP. In Section III we describe in more detail the operation of PSI in a WAN environment and the mechanisms for constructing the Steiner trees. In Section IV we briefly describe our implementation, while section V presents our preliminary evaluation results. We conclude and present plans for future work in Section VI.

## II. PSI: PUBLISH-SUBSCRIBE INTERNET ARCHITECTURE

### A. Network service model

PSI models information items as publications, information producers as publishers and information consumers as subscribers. The network provides users with two basic primitives: (i) *publish*, used for announcing the availability of publications to the network and (ii) *subscribe*, for expressing interest in receiving publications (note that publish does not involve data transmission; it only advertises data availability). The granularity of publications is not mandated by the network; on the contrary, it is left to applications. Publications may represent (large) files, chunks of files, network layer packets, services or channels of live streaming media.

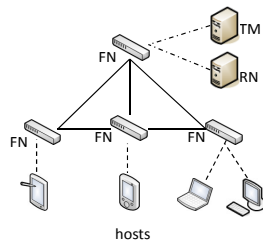


Fig. 1. PSI network elements: Rendezvous Nodes (RN), Topology Managers (TM), Forwarding Nodes (FN) and hosts.

### B. Functional Organization

The operation of a PSI network is the synthesis of three core functions [1]: (i) Rendezvous, (ii) Topology Management and (iii) Forwarding. The Rendezvous function tracks available publications in the network and resolves user subscriptions. The Topology Management function serves a dual cause. First, it monitors network topology and link conditions. Second, it computes optimal delivery paths for disseminating publications from publishers to subscribers. The Forwarding function undertakes the actual data transmission, i.e. packet forwarding. Figure 1 shows the four kinds of network elements in PSI, with respect to the system's functional organization: (a) Forwarding Nodes (b) Rendezvous Nodes, (c) Topology Managers and (d) user Hosts.

Information delivery requires a close interaction between the three core functions. Publication announcements issued by users are handled by the Rendezvous function which stores a mapping between publishers and available publications. Subscriptions are also handled by the Rendezvous function which performs the publication-subscription matching. Once a publisher is found, the Rendezvous function requests the Topology Management function to compute a *suitable* forwarding path for disseminating the requested publication. The path is encoded into a *Forwarding Identifier* and sent to the publisher in the form of a notification. Publishers receive these instructions from the network and transmit the requested publications over the specified paths.

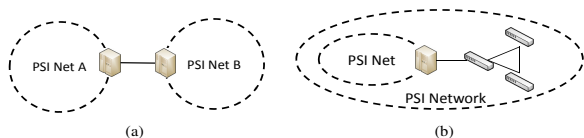


Fig. 2. Interconnection of PSI networks: (a) neighboring PSI networks, (b) nested PSI networks.

### C. Multicast Delivery

Multicast delivery requires some network entity to keep track of users subscribed to information items (e.g. RSS feeds) in order to construct and establish the respective forwarding trees. In PSI, the task of tracking multicast receivers is assigned to the Rendezvous function. When a user issues (withdraws) a subscription, the Rendezvous function adds

(removes) the user from the list of item receivers and requests the Topology Management function to create a source-specific forwarding tree, rooted at the publisher and with item subscribers as leaves. The details of how these mechanisms are implemented vary according to the function implementations used. In IP multicast for instance, all routers implement both the Rendezvous and Topology Management functionalities as they participate in distributed multicast routing protocols. We describe how multicast is implemented in a WAN setup in PSI in Section III-B.

### D. Internetworking in PSI

PSI networks interconnect with each other through border gateways for internetworking purposes. PSI networks may be connected with other PSI networks forming neighboring relations (Figure 2a) or a network may contain nested PSI networks (Figure 2b). Complex network organizations may use a mixed combination of the two interconnection types.

Each PSI network runs its own domain-specific implementations of the core functions, carefully selected for the particular networking environment it operates in (e.g., home, enterprise, mobile, wireless ad-hoc, data-center, etc). PSI networks communicate with each other through inter-PSI implementations for Rendezvous, Topology Management and Forwarding, in the same spirit of the inter-domain routing protocols used in the Internet. In inter-PSI communication, each network aggregates subscriptions issued by domain-local users when these need to be resolved by other PSI networks. Upon the arrival of the requested data packets, the network demultiplexes them and delivers the data to local subscribers. Apart from enhancing the system's scalability, this inter-PSI communication abstraction allows each network to apply its own domain-specific policies in order to optimize its internal operation, without affecting global communication.

### E. Discussion

The functional organization of PSI diverges from IP thinking in two major design aspects. First, network functionalities are centered on what is being transferred instead of who are the communicating end-points. This requires pushing item resolution functionalities down to the network layer. IP, in contrast, does not require such functionalities at its core; existing resolution systems like DNS and DHTs are deployed on top of IP, operating in overlay mode. This layered operation however has a significant drawback: these resolution systems are isolated from the (underlay) network, hence they cannot exploit detailed network information during item/host resolution (say, select a publisher among many possible publishers based on network congestion conditions). In reality, layering principles often have been violated in order to optimize data delivery, e.g. the various DNS tricks applied by CDN operators [8]. However, these violations do not come for free as they introduce anomalies in the system [8]. Second, PSI clearly separates the Forwarding from the Topology Management functionality; more specifically, it decouples packet forwarding, path formation and topology monitoring tasks. Decoupling path formation from topology monitoring and computing paths

on a per-flow basis can be highly beneficial for a variety of applications, as it enables the network to take into account application-specific requirements in path establishment such as QoS restrictions (e.g., end-to-end delay, packet loss).

In this paper we showcase the benefits of separating these functions by delivering real-time information over optimal (minimum cost) Steiner trees. Our aim is not to present a new algorithm for computing Steiner trees; it is to show the simplicity and feasibility of realizing such schemes in PSI. We are motivated by the fact that such algorithms, although extensively studied [5], were not adopted for IP, due to the complexity of constructing a Steiner tree in a fully distributed manner [5]. This occurs because in IP forwarding, topology management and rendezvous functionalities are coupled inside IP routers (e.g. IP multicast routing protocols). We showcase how such optimization policies are feasible in PSI by (i) delegating tree construction to a logically centralized component and (ii) replacing IP forwarding with new stateless multicast forwarding schemes such as LIPSIN [4].

### III. REALIZING PSI IN THE WIDE AREA

In this section we provide a more detailed description of a PSI implementation in the context of a Wide Area Network (WAN) and the specific solutions for the three core functions.

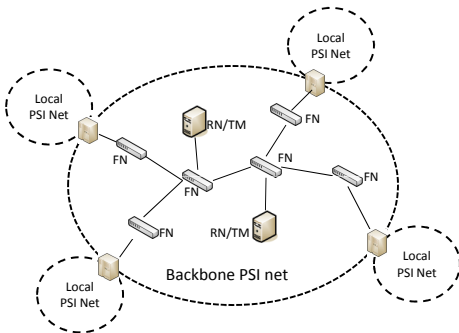


Fig. 3. A PSI WAN that interconnects PSI LANs.

#### A. Core Function Implementation

A PSI WAN consists of a backbone PSI network that provides access to users residing in local area PSI networks (Figure 3). For the forwarding function, we use LIPSIN [4], a source-routing scheme that encodes delivery paths (unicast and multicast) into Bloom filters. Among the features of this technique, LIPSIN supports stateless multicast forwarding, i.e. FNs are not required to store explicit forwarding information per multicast tree. The Rendezvous function is undertaken by several RNs which distribute the load of tracking publications and serving subscriptions in a DHT fashion [3]. When a publication/subscription is issued, it is forwarded to the nearest RN and then routed through the RN-DHT. RNs are co-deployed with FNs, analogous to a DNS server being deployed in an IP-based host. Topology Management functionalities are spread over all nodes. The first task of the TM function - topology discovery and network monitoring - is implemented

through a link-state routing scheme: FNs and border gateways in the backbone participate in the link-state exchange announcing their connectivity information. The second task of the TM function - path computation - is performed inside RNs. When an RN handles a subscription, it extracts the topology information from its co-located FN and computes the publisher-subscriber path. The path is encoded into a LIPSIN source-route forwarding identifier and handed to the publisher.

#### B. Efficient Multicast Delivery

Multicast tree computation is performed at the RNs during subscription handling. The baseline operation is to compute a shortest-path tree, i.e. the union of the shortest paths from the publisher to each subscriber. The centralized nature of tree computation however enables the system to apply more sophisticated algorithms for multicast tree construction. For instance, the RN may compute optimal (minimum cost) Steiner trees.

Multicast trees are constructed with minimal - to the network - signaling overhead. It only requires a subscription to be handled by the RN-DHT and a notification to be sent to the publisher. In addition, due to the stateless multicast capabilities of LIPSIN, no further control plane messages are exchanged, for example, there is no need to explicitly set up multicast forwarding state in FNs [9]. Steiner tree construction requires no further signaling compared to construction of shortest path trees. Compared to IP, the signaling cost for tree construction in PSI is proportional to a DNS/DHT resolution. On the other hand, the operation involves considerable computation overhead in the RNs since they need to handle publications and subscriptions, track active subscribers per item and compute multicast trees. Hence, careful network planning is required when selecting the number and hardware capabilities of the physical nodes for deploying the RNs.

#### C. Discussion

The above design delegates control plane functionalities (multicast group tracking and tree construction) to a *logically* centralized module, the RN-DHT. Centralized schemes are often dealt with skepticism due to scalability constraints. We note however that for the tree formation, the RNs take into account only the WAN topology. Access networks and users are not part of the backbone and they are viewed as neighboring PSI networks. This abstraction highly reduces the computation overhead at the backbone RNs. Moreover, the benefits of delegating control plane functionalities to logically centralized components are increasingly considered as very significant, despite the scalability advantages of fully distributed schemes, strengthening such design approaches [9].

## IV. EXPERIMENTATION

#### A. Network Implementation

We have implemented a proof-of-concept prototype to validate the architecture and explore the feasibility of the aforementioned ideas. The prototype operates as an overlay network. Links between nodes are UDP-tunnels. For topology

discovery, FNs and RNs periodically flood the network with link-state announcements. Unicast paths are computed using Dijkstra's shortest path algorithm with hop-count as the metric. For multicast trees, the baseline is the shortest-path tree, i.e. the union of the shortest paths. For Steiner trees we used the KMB heuristic [6].

### B. Live media streaming applications

We also implemented a simple live media streaming application. Streaming sources act as publishers and stream receivers act as subscribers. Sources first choose a name for the stream and publish it to the network. Subscribers simply subscribe to the stream. We note that subscription handling is required only for flow establishment. Once the tree is constructed and the LIPSIN identifier is passed to the streaming source, subsequent streaming packets are directly delivered from the source to receivers.

Subscribers are expected to obtain the available channel names through an out-of-band mechanism, for example directory services or search engines. In voice/video conversations (e.g. Skype sessions), each user in the session selects a name for her outgoing stream, publishes the stream's name to the network and then hands the name to the receiver through a separate signaling mechanism. Multi-party conferences are decomposed to a number of single-source streams, one per participant. Each user subscribes to each participant's stream, for which the network creates a source-specific multicast tree.

To stop receiving a stream, subscribers issue unsubscriptions. Subscriptions also expire after a time-out, after which the user no longer receives data packets. To stay tuned, users must periodically re-subscribe to the stream. The soft state of subscriptions serves as a precaution mechanism to prevent the network from pushing packets to users that are not interested anymore but did not properly un-subscribe (e.g., due to a node or link failure or, simply, negligence).

## V. PRELIMINARY EVALUATION

### A. Evaluation metrics

Our initial evaluation goal is to measure gains in bandwidth usage when delivering data over Steiner trees compared to using shortest-path trees; we also consider using multiple unicast flows (as is the case in the Internet today) as a baseline. Our second goal is to examine how the processing overheads for computing Steiner trees at the RNs affect the user experience in terms of the increased delay in flow-establishment compared to the shortest-path tree mode.

### B. PlanetLab Experiments

We deployed PSI in 35 *PlanetLab Europe* (PLE) nodes in a dense network topology; Figure 4 depicts our *backbone topology* which consists of 20 nodes. The other 15 nodes (not shown) acted as PSI LANs providing access to users.

In each PSI LAN we instantiated 2 user processes, thus we had 30 users in total. We then injected a synthetic workload emulating a TV scenario, in which a user switches among 6 available streaming channels, all produced by the same source.

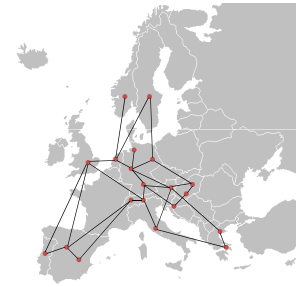


Fig. 4. Network topology of PlanetLab overlay.

User behavior is modeled by two modes: a) the stationary mode, where the user stays tuned to a channel for a duration uniformly distributed between 4 and 8 minutes and b) the channel surfing mode, where the user switches sequentially between channels, staying tuned in each channel for 10-30 seconds. Each user initially starts in the stationary mode and then alternates between the two modes. When in surfing mode, the user changes a channel  $x$  times, with  $x$  being an integer uniformly distributed between 2 and 5 (i.e. at least 2 channel switches and at most 5). After surfing, the user switches to the next channel and goes back to stationary mode, and so on. Each experiment lasted 2 hours and we repeated it once for each delivery scheme: (i) multiple unicast flows, (ii) shortest-path trees and (iii) Steiner trees.

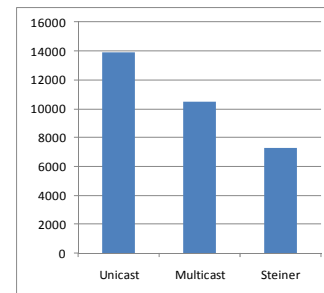


Fig. 5. Total number of bytes transferred between FNs (in GB) in the three delivery modes.

Figure 5 shows the total number of bytes (in GB) transferred in the network, i.e. the number of bytes transmitted by FNs. Shortest-path trees reduced bandwidth consumption by a factor of 25% compared to multiple unicasts, while Steiner trees reduced the amount of transferred bytes by 31% compared to shortest-path trees and by 48% compared to multiple unicasts.

We now focus on the cost of constructing the Steiner tree compared to the shortest-path tree, which is the baseline multicast mode in PSI. As discussed in section III-B, the signaling cost remains the same regardless of the multicast mode used. The cost of the optimization scheme comes in the form of additional processing delay in the RNs, which increases network response when handling a subscription and, therefore, also increases the flow-establishment delay. Due to the public nature of the Planetlab testbed, we could not extract reliable observations regarding end-to-end delays (link delays



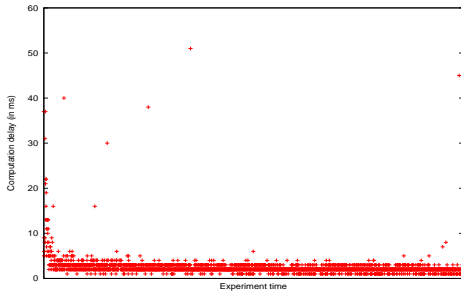


Fig. 6. Delays within the RN for computing Steiner trees.

varied extremely during our experiments). Thus we examined the processing delays in the RNs for computing the multicast trees. In Figure 6 we plot the computation delay in the RN for computing Steiner trees. The Y axis is the time required for running the KMB heuristic (in ms) and the X axis is the time in the experiment. 99% of KMB computations required up to 10 ms while the maximum observed computation delay was 50 ms. These processing delays are almost un-noticeable to users, as they are a small fraction of the end-to-end delay.

### C. Emulated Network Experiments

The PLE deployment reflects a small-scale network setup. To test the system in a larger scale, we moved to an isolated environment and emulated AS224, the Norwegian University & Research Network which consists of 233 routers [7]. We executed 233 instances of our software in a workstation - one instance per network node. Instances communicate with each other exactly as in the PLE testbed. The connectivity between the instances reflects the topology of the network. We considered 75 of the nodes to be PSI LANs, each one providing access to 10 users; hence the network serves 750 users in total. Users follow the same behavior as in the PLE experiments.

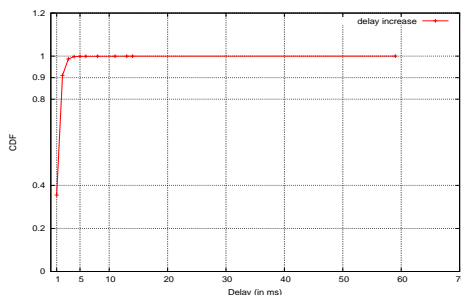


Fig. 7. CDF of additional delay in flow-establishment between Steiner tree and shortest-path tree delivery.

We ran the system using the same synthetic workload as in the PLE experiments, once for each of the two multicast modes. For each subscription issued, we measured the flow-establishment delay and calculated the delay increment in the case of the Steiner tree mode, compared to the shortest-path mode. In 99.6% of the cases the delay was increased by less

than 60 ms, while for the remaining 0.4% it ranged from 200 ms to 32 seconds. We believe that this small portion of outliers is an artifact of the implementation and workstation load, thus we considered the outliers as noise and removed them from the results. In Figure 7, we plot the CDF of the additional flow-establishment delay. In 90% of the requests, flow-establishment in the Steiner-tree mode was increased by only 2 ms compared to the shortest-path tree mode, while for 99.9% the increment is less than 5 ms. The results indicate that - at these scales - Steiner tree construction has negligible costs with respect to user experience.

Another important issue related to system scalability is the request aggregation in PSI LANs and its effect in the subscription load that actually hits the PSI backbone. In our experiments, viewers generated 93,500 subscriptions in the 2 hour experiment, i.e.  $\sim 13$  subs/s. However, due to the request aggregation in PSI LANs, the actual number of subscriptions sent to the backbone was 55,000 ( $\sim 7.6$  subs/s), a reduction of more than 40%. We anticipate that in larger setups with zipf-like distributions in item popularity, the backbone load will not grow linearly, therefore the design would scale to significantly larger deployments. To what extent this will be the case, is an issue of further research.

## VI. CONCLUSIONS AND FUTURE WORK

In this paper we presented how the efficient delivery of real-time information can be supported in the Publish Subscribe Internet architecture. The functional design of PSI, makes the construction of Steiner delivery trees realistic, with minimal signaling costs. We supported our arguments by preliminary experimental results of a real PSI implementation, deployed in PlanetLab as well as in an emulated network. Although tested in a small scale, test results indicated bandwidth savings of 30% at the cost of small, un-noticeable to the user, delays in flow-establishment.

For future work, we plan to further investigate the optimization space introduced in large-scale setups. We expect that the design choice to delegate control-plane functionalities to a logically centralized component (the RN-DHT) will face scalability issues, for example, large processing requirements in the RNs that will lead to increased delays. We plan to examine the scalability constraints of the system compared to the offered optimization opportunities.

## ACKNOWLEDGMENTS

The work reported in this paper was supported by the FP7 ICT project “Publish Subscribe Internet Technology” (PURSUIT), under contract ICT-2010-257217.

## REFERENCES

- [1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. L. Braynard, “Networking named content,” in Proc. ACM CoNEXT, Rome, Italy, Dec. 2009.
- [2] D. Trossen, M. Särelä, and K. Sollins, “Arguments for an informationcentric internetworking architecture,” ACM SIGCOMM *Computer Communication Review*, vol. 40, no. 2, pp. 26-33, 2010.
- [3] 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 Magazine Special Issue on Information Centric Networks*, Vol. 50, no. 7, pp. 52-58, 2012.

- [4] P. Jokela, A. Zahemszky, S. Arianfar, P. Nikander, and C. Esteve, "LIPSIN: Line speed publish/subscribe inter-networking," Proc. ACM SIGCOMM, Barcelona, Spain, Aug. 2009.
- [5] L.H. Sahasrabudde and, B. Mukherjee, "Multicast routing algorithms and protocols: a tutorial," *IEEE Network*, vol.14, no.1, pp.90-102, Jan/Feb 2000.
- [6] L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Tree," *Acta Informatica* 15, pp. 141-145, 1981.
- [7] J.-J. Pansiot, P. Merindol, B. Donnet, and O. Bonaventure., "Extracting intra-domain topology from mrimf probing," Proc. PAM, Zurich, Switzerland, April 2010.
- [8] P. Vixie, "What DNS Is Not," *Communications of the ACM*, vol. 52 no. 12, pp. 43-47.
- [9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," Proc. ACM SIGCOMM, Seattle, USA, Aug. 2008.