Exploring the Pub/Sub Routing&Forwarding Space

András Zahemszky, András Császár, Pekka Nikander
Ericsson Research
Email: {firstname.lastname}@ericsson.com

Christian Esteve Rothenberg
University of Campinas (UNICAMP)
Email: chesteve@dca.fee.unicamp.br

***Abstract*—We envision an information-centric future Internet where the network is built around named pieces of data instead of explicitly addressable hosts. One clear way of implementing information-centric networking is using publish and subscribe (pub/sub) operations instead of the send and receive primitives. Internet-like pub/sub networking requires completely different routing protocols and forwarding mechanisms compared to those that are extensively used today. Consequently, we are facing a clean-slate design exercise, where we should start our adventure by exploring the new design space. We identify four key metrics (signalling overhead, state in nodes, information in packets and routing stretch) to help us evaluating the different proposals. We present a general five-step approach for routing in pub/sub networks. The presented approach is recursive, so it can be repeated as many times as necessary until we reach manageable sized problem instances. The final part of the mechanism is to glue together the created and assigned forwarding structures to the publication to ensure that all interested subscribers at any domains in the network will get the requested data.**

I. Introduction

In addition to the prominent World Wide Web, new forms of Internet usage have appeared rapidly in the last years. People read news feeds, listen to webradios, watch (RT or non-RT) video streaming, and download huge amount of data using BitTorrent. A common characteristic of these applications is that the users indicate their wish to get some specific pieces of information and they are not interested in the sources of data as long as the data comes in its original, unmodified form.

The first one to prominently advertise this shift in the application space from connections to information was Van Jacobson in his talk in 2006 [1]. Since that, it has become a common direction to propose clean-slate solutions for solving the problems of IP [2]. Researchers are aiming for new internetworking solutions that have native support for mobility, multi-homing, privacy, accountability, or other clearly missing features. Furthermore, many of the proposed solutions use data instead of the hosts as the basis for the design [3].

As a specific example, the EU FP7 PSIRP project [4] has a goal of building a pub/sub-based network, where the architecture uses pieces of data as the first-class citizens. With the *publish* operation, an endpoint can indicate that it wants to associate a document or a one-way channel with the given (possibly randomly looking) identifier. With the *subscribe*, an endpoint can signal its desire to get (read only) access to the named document or channel. Based on the subscriptions, the network is responsible for delivering the document or any data appearing on the channel, to all the subscribers. As typical to pub/sub-, multicast is the natural mode of communication.

In this paper, we explore the new design space we are facing when attempting to design the routing and forwarding components of the PSIRP architecture [4]. First, we discuss the problem of designing a scalable routing mechanism for multicast and multicast-based publish/subscribe (Sec. II). Next, we briefly overview the RTFM architecture [5], the variant of the PSIRP architecture that our work is based on, including various possible forwarding components (Sec. III). The core contribution of this paper is our Divide&Conquer approach to tackle the problems of scalable routing (Sec. IV). We divide the routing problem in two dimensions: first, vertical and second, horizontal. The former is hierarchical division and the latter is dividing the overall problem of multicast routing in each area of the hierarchy to smaller subproblems, each of which is easily manageable as such. Finally, we discuss related work (Sec. V) and provide our tentative conclusions (Sec. VI).

II. The problem

Our aim is to create an information-oriented network based on the pub/sub paradigm. The data transmission with unicast in such networks is clearly not optimal w.r.t. transport efficiency, while the same data is delivered over the same connections multiple times. Thus, we have selected multicast as the primary transport mechanism. As a consequence, it is more natural to consider delivery trees rather than forwarding paths.

A forwarding tree for a given publication contains the publisher and all subscribers. It may be optimal w.r.t. an appropriate metric, e.g. delay. We call such trees as *ideal* trees.

It appears that the routing&forwarding solution space for multicast tree implementations is 4-dimensional, as they can be evaluated by four rather orthogonal metrics. The first is the transport efficiency measured in *stretch*, meaning that the packets follow the shortest possible paths (in any appropriate metric). The second concerns the amount of *network state* needed in the forwarding nodes. This must grow sublinear w.r.t. the number of nodes and subscribers and strongly sublinear w.r.t. the number of publications. The third aspect involves the number of bits included in the *packet headers*, encoding the information that helps the forwarding node to determine the outgoing interface. The final one is the *signalling overhead* caused by routing and other related control protocols.

Assume two examples: a) *source routing* requires a high amount of information in the packet headers but relatively little state in the forwarding nodes. The downside is the signaling overhead in scenarios under subscribers' churn, as the sender of the data should be always up-to-date on the information

encoded in the headers. Moreover, the transport utilization depends to a large extent on the algorithms the overall system uses to compute the forwarding trees for data delivery.

In the second example, b) *network state* installed to describe directly the ideal trees into the forwarding nodes provides high levels of transport efficiency and minimal number of bits in packet headers (just a tree identifier). Potentially, each forwarding node may need to have a separate entry for each single active publication tree in the globe. Moreover, every change in any tree would need to be signalled to a potentially large number of forwarding nodes.

To avoid potential state explosion (as our estimation total number of possible publications $\approx 10^{15}$), we are looking for solutions that use *good enough* forwarding trees, i.e., which are close to ideal but require only a reasonable amount of state in the network. When trying to save state, we may end up with a little higher overhead, for example in terms of unnecessarily sent packets. However, this may be remedied by other means, for instance by utilising *opportunistic caching*.

In this paper, our goal is to work towards a solution that finds a good balance among the contradictory aspects. To get there, we first present the overall PSIRP architecture, and then briefly discuss our ongoing work in the forwarding domain.

III. The PSIRP components

We start by briefly outlining the RTFM architecture [5], an early design choice from the PSIRP project. Our present contribution can be regarded as an evolutionary step building upon RTFM. The RTFM consists of recursive functional building blocks:

1) A *Rendezvous* system is in charge of matching subscriptions to publications within information scopes.
2) The *Topology* system, our main focus in this paper, is responsible for collecting and managing network topology information, as well as creating and maintaining the required multicast delivery trees both proactively (establishment, optimization) and re-actively (on-demand).
3) The *Forwarding* functions perform the actual datagram delivery, based on the forwarding identifiers in the packet headers and the state installed in the forwarding nodes by the topology system.
4) Finally, *More* refers to the additional data mediation functions at forwarding time, such as opportunistic caching or network coding.

At the bottom of the architecture we have the forwarding layer. Above it, the structure can be divided into a data and control plane (see Fig. 1). The functions are present seperately in different network levels (e.g. in domain(AS)-level, core network level). The *control plane* consists of the topology and rendezvous systems. At the *data plane*, in addition to the traditional transport functions, we envision a number of new network functions, e.g. opportunistic caching and lateral error correction. The transport functions will work in concert, utilising each other in a *component wheel* [4], similar to the way Haggle managers are organised [6].

In our design, the basic communication scheme is functionally similar to IP-based Source Specific Multicast (SSM) [7], with the IP multicast groups having been replaced by an identifier identifying the publication, similar to the topic identifier described in [8].

Each publication, be it a single packet, a one-way multicast channel or a document, is identified with a Rendezvous Identifier (RId). More precisely, each publication is identified with a $< SId, RId >$ pair, where the Scope Identifiers $< SId >$ are just specific RIds, identifying *scopes*, which help the rendezvous system to scale and to organise the publications.

When a publisher wants to publish a new piece of information, it picks up a RId and hands the publication data to the system. Correspondingly, subscribers acquire the RIds through application-specific means and ask the system to arrange the data to be received. Once the rendezvous system has identified a publication that has both a publisher (or an up-to-date cache) and one or more subscribers, the topology system is requested to build a forwarding tree from the present location of the data to the subscribers. The high-level operations of this routing function are the main concern of this paper.

A. Forwarding components

First, we outline four solution components: using in-packet Bloom filters to encode delivery trees, using Merkle trees to represent partial forwarding trees, and scaling these approaches up to more dense trees through installing explicit state within the forwarding nodes, and finally, we briefly outline how the latter approaches could work together.

1) Encoding delivery trees with Bloom filters: Bloom filters (BFs) [9] are data structures for representing subsets of a given maximum size without listing the individual elements. BFs are applicable in any situation where a small number of false positives is acceptable. When used to take forwarding decisions, false positives are translated into packets being transmitted over additional links than the ones originally programmed. As long as the false positive rate is low enough, we consider that acceptable due to active caching and the decreasing probability of concatenated false positives over multiple hops; for the details, see [10].

To encode delivery trees, we form a set L of directed links. That is, for the forwarding nodes A and B, we represent the

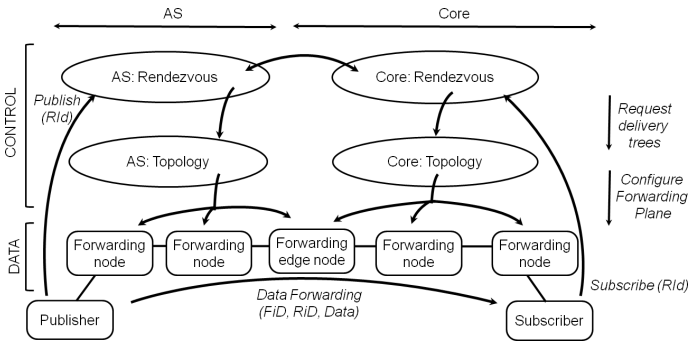

Fig. 1: High level arquitectural overview.

link from A to B as $\overrightarrow{AB}$, and the link from B to A as $\overleftarrow{AB}$. So, any forwarding tree can be seen as a set of unidirectional links. For example, encoding a simple tree with links from A to B, and from B to C, we should add $\overrightarrow{AB}$ and $\overrightarrow{AC}$ into the BF.

We place the BF describing the delivery tree into the packet header. Checking the header, each forwarding node tests which of its outgoing links are included in the set. Since this is a simple binary `and` operation, the checks can be done parallel in hardware, producing a very fast forwarding plane.[1]

As we show in [10], using 248-bit BFs, it is possible to encode up to about 40 link names (out of trillions) into a single BF, while still having an acceptable false positive rate (of around 4%.) That means that we can forward unicast packets over 40 links, well over the maximum practical hop count in the current Internet. For multicast, if we assume Internet-like AS topologies, we can use a single BF to address up to 20 receivers scattered all over a large AS topology. If more is needed, state can be added into the network in the form of virtual links as shown below.

Hence, using BFs in packets seems to provide us with an efficient way of source routing in arbitrary trees over Internet-like topologies.Our approach aims to work in intra- and inter-domain scenarios over any transport technology.

2) Merkle tree representation: Merkle trees [11] are data structures that contain summary information about a larger tree-like data structure. We propose to use them as an alternative compact form to represent forwarding trees and to derive chained forwarding labels.

Every node in the Merkle tree has a view of its vicinity and knows about the active trees it participates in. The active tree structures need to be consistent among the domains participating in the inter-domain routing, requiring the existence of a routing protocol to maintain the tree structures. The actual labels forming the hash trees can be constructed for each active tree. The node compares the received label with the compiled root hashes of its active network trees to resolve the next hop(s) and include the updated forwarding identifiers. Hence, the chained hashing mechanism of Merkle trees provide an uniform way to derive compact labels that aggregate paths recursively.

3) Utilising subtrees as virtual links: When the capacity of a single BF is not sufficient, the system can be extended by representing some subtrees as a *virtual link*. Basically, a virtual tree is a subtree that has a distinct name and associated state in the network nodes. That is, while we use BFs to encode the links of a tree in the packet header, we can also include trees as virtual links in this Bloom-filter-based representation.

Given the scale-free characteristics of the Internet, a major part of the data travels through a very small hub (tier-1). With virtual links, we can define forwarding subtrees that take advantage of the common path towards the center of the network that can be assumed for the vast majority of the traffic from any source. By defining virtual links spanning over multiple hops towards the domain edges, more stable fast forwarding paths can be installed explicitly in the network.

Hence, with this construction, the problem of mapping rendezvous identifiers to forwarding identifiers can be reduced, deeper in the network, to a smaller mapping problem of active flows (partly defined by virtual links), which is well within the scope of feasibility.[2]

4) Approximate fast stateful edge switching: False-positive-prone fast forwarding decisions can also be performed through stateful operations between domain boundaries. At the edges, making a switching or mapping decision between the large rendezvous identifier space and the smaller forwarding identifier space needs to be efficient both in space (small amounts of high-speed memory) and time (few computation cycles per packet).

The *SPSwitch* in [2] is a promising approach to solve the problem: with only a few bits per entry (40-50 bits), switching of packets identified by long flat identifiers (e.g., 256-bit) can be performed in a fast and efficient way. Again, the price is a fairly small amount of traffic delivered over unrequested paths.

B. Putting the components together

As described above, we have different forwarding mechanisms (in-packet BFs and Merkle-trees). The idea of virtual links combines both approaches by allowing explicit stateful trees to be included into the in-packet Bloom filters. Finally, the SPSwitch approach tackles the mapping problems we are likely to have at domain boundaries.

We need to combine these mechanisms so that we create state on-demand and only where required, adapting to the actual traffic patterns. This is achieved by relying on semi-stable subtrees represented by virtual links spanning multiple hops. In addition, by employing false-positive-prone forwarding schemes over suboptimal trees, we achieve state reduction and line speed at the cost of some extra delivery. It is a fair and currently necessary trade-off to reach the required levels of scalability. Indeed, a clever design can convert this apparent bandwidth waste into a strength of the architecture (e.g. opportunistic caching, resilience etc.).

IV. Divide and conquer

We present our Divide&Conquer-based approach to tackle the scalability problem of massive amount of overlapping multicast trees for efficient data delivery. We split the problem space along two dimensions: we use hierarchies in the network to achieve scalability and we divide the problem into smaller, more manageable steps.

A. Hierarchical aggregation

As a first approach, we use the traditional and successful way for achieving scalability: hierarchical aggregation. Our requirement can be explained in what we call *scalability*

[1]There is a small possibility that this simple method would create forwarding loops. For loop prevention, see [10].

[2]Consider as a reference the current performance of MPLS/VPN devices under route reflector operations handling up to few millions of flows.

principle: In any given domain *A*, the amount of state corresponding to any remote domain *B* should not depend on the number of subscribers in domain B. The key of this principle is that any node in domain *A* should see domain *B* as only one subscriber, even if there are many real subscribers within *B*. Changes inside domain *B*, therefore, should only cause change in the forwarding states within domain *B*. Topology hiding, like in today's Internet, not only meets the operators' business interest but also helps achieving scalability.

Hierarchy can be implemented recursively on different levels, e.g. consider OSPF areas, autonomous systems (ASes), AS confederations etc. in today's networks. We should, however, not restrict ourselves to this division, as the introduction of the new paradigm may not only change the current AS structure but also the fundamental policies [12] that define what autonomous system means. We should not introduce a constraint in the number of levels to be used: when an operator feels necessary it could introduce a new hierarchy level at any time. For the sake of easier explanation, in the following we will work only with two-level hierarchies: intra-domain and inter-domain levels.

B. A *five-step approach*

Now we turn our attention towards the other dimension of the division. Our approach can be described as taking a few steps, one after each other. All these operations should take place within each single area that we have in the hierarchy. The steps are as follows:

1) Compute an ideal tree.
2) Determine the gaps between the ideal tree and any existing trees.
3) Select tree-creation strategies or gap filling strategy for each gap.
4) Compute the needed changes according to the strategies.
5) Apply the changes to the network.

1) Ideal tree: The first step is to *compute an ideal forwarding tree* that connects the publisher to all the subscribers. We envision that this will be performed in the vicinity of rendezvous and topology layers, in a (possibly) distributed manner. The ideal forwarding tree is the best tree regarding any appropriate metrics (delay, hop count, etc.). The resulting forwarding tree will usually span several different domains. By introducing hierarchies, we can reduce the problem of creating one overall ideal forwarding tree to creating several intra-domain forwarding trees and an inter-domain (AS-level) forwarding tree connecting them.[3] These ideal forwarding trees will guide us as a reference when we assign the actual trees for the ultimate data delivery.

2) Gaps: In the second step, the topology layer *compares the existing forwarding trees to the ideal ones* and selects the best matching existing trees in each domain. There are no restrictions on the number of existing trees used, logically combining several existing trees is allowed. The resulting best-matching tree either covers all the subscribers in the domain, and therefore can be utilized for publication delivery if other criteria are met (e.g delay requirements) or does not cover some subscribers. In the latter case, it cannot be used for publication delivery in its current form, before it gets expanded. However, we may be able to avoid the latter situation by default including trivial broadcast and short-range unicast trees in each domain.

3) Tree-creation and gap filling strategies: The third step is to *select a tree-creation strategy*, separately in each domain, for creating the actual forwarding tree. If the combination of the existing trees already covers all the subscribers inside a domain, then this combination could be used as the actual delivery tree. However, if any of the determined further constraints are not met, more work is needed. For example, if the best-matching tree contains too many subtrees where there are no interested subscribers, an action must be taken, either immediately or later. In general, it seems a viable strategy to first use the existing, non-optimal tree, and build one or more new trees in parallel, and switch over to the new trees once they are ready. An alternative may be to modify some existing trees, provided that the modified trees still fulfill the requirements of their other use. The exact details, however, depend on how the trees are built and installed, i.e. how much the solution is based on source-routing like approaches and how much on actually representing the trees with explicit state in the forwarding nodes. Our present solution is largely flexible here, as it became apparent from the description of the solution components in Section III-A.

If the combination of the existing trees does not cover all the subscribers, the network must *select a gap filling strategy*. For example, it may decide *to modify existing trees or create new ones* in order to reach all the subscribers and meet other constraints. Note that the compulsory requirement for the new combination of forwarding trees is that it must cover all the subscribers for the given publication.

4) Compute the changes: After the routing entity has selected the strategies, it should *compute the new/modified trees* accordingly. This process is eventually mapping the internal representation of the tree to the representation that is used in the network. Here, different forwarding solutions yield different actions: either installing elements to Bloom filters, or encoding tree structures into Merkle trees, or more possibly combining both of them. Finally, the routing entity should identify the places (network elements) where state injection or update is needed.

5) Apply changes: As the last step, the *changes must be signaled* to all affected entities. This involves sending control messages ordering nodes to install or modify forwarding states, as well as notifying the source of the data delivery if a new forwarding identifier is to be placed into the packet header.

C. An example scenario

Figure 2 illustrates the hierarchical aggregation. The entities contain nested entities (e.g. today ASes contain areas and areas contain routers). Consider that on a level of hierarchy Entity

[3]Note that the resulting overall tree may not be ideal but that it will be policy compliant and that each subtree will be ideal.

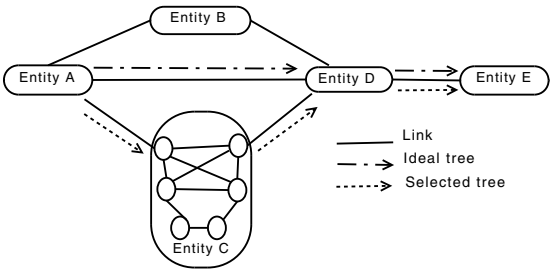

Fig. 2: Example of hierarchical aggregation and tree selection

A is the publisher (which means that it contains the publisher host) and Entity D and E are the subscribers. From the hop count point of view, the tree A-D-E is the ideal. Assume that there exists another publication which has the tree A-C-D-E. The routing system therefore can assign this tree for the new publication as well, as it contains all subscribers. Consequently, we should not add new forwarding state as we have not created any new tree on that level. Finally, when the publication reaches e.g. E, it should switch to an internal tree to reach the subscribers residing inside. However, there will not be any internal forwarding tree for this publication in Entity C. By caching the publication, Entity C may as well increase the network performance by supporting e.g. faster error correction (implementing the network coding *More* functions envisioned in RTFM). Of course, if some policy dictates that the ideal tree should be used or any requirements are not met, the delivery of the publication can be switched to it as soon as all the necessary states are installed in the network.

D. Towards a pub/sub routing protocol

A key to implement the five-step mechanism is to have a (distributed) entity in the network that is topology-aware. We suggest a GMPLS Path Computation Element (PCE)-like [13] topology manager, adopted into our pub/sub world.

The pub/sub PCE collects the topology information of the network domain by subscribing to link advertisements coming from the nodes (this requires to have a default path between any node and the PCE). The content of the link advertisements may vary depending on the actual forwarding mechanism used. Finally, the pub/sub PCE will have a full view of the topology of the domain. Assuming that publish and subscribe messages collect the path between the PCE and the publishers/subscribers, the PCE can locate all the entities and can compute the trees.

The difference of the current practice is the absence of link advertisement flooding, as only the PCE should build the network graph and other participants do not need to subscribe, meaning that changes trigger less control messages than today. Considering the inter-domain case, the cooperation of pub/sub PCEs results in the appropriate inter-domain forwarding structure.

E. Challenges of the division

The operations at the hierarchical boundaries (e.g., edge router mapping of trees) represent the main scalability challenge of our Divide&Conquer approach. Obviously, a full source-route would circumvent the edge-mapping problem. However, strict source-routing has its limitations and price (see Sec. II). Therefore, we need to consider suitable label swapping and stacking mechanisms with an optimal balance of the size of network state and packet headers, and signaling overhead.

During the tree construction phase, the appropriate set of forwarding trees (potentially multi-domain) has to be chosen so that all domain internal subscribers are covered. At the domain boundaries, when an edge forwarding node receives a packet over an inter-domain tree, the edge node must determine the right internal tree; similarly, after receiving packet on an intra-AS tree, it must determine the right inter-domain tree. Obviously, there will be no one-to-one mapping between the trees at different levels..

To change between the hierarchy levels, one possibility is to look inside the packet, using the publication-level identifiers. This approach seems unfeasible from a scalability point of view considering the potential amount of *active* rendezvous identifiers travelling an edge during a certain time window (publication data delivery time).

To benefit this fact that potentially many publication identifiers will require the same mapping of trees, we propose the use of a special set of ”non-routable” link IDs with the goal of triggering the mapping. Typically, we need at least one virtual link from each edge node in the AS to the rest of the edge nodes. The amount of intra-domain virtual links “bridging” edge routers and providing a fast path to each neighbouring AS will be in the order of a few hundreds.[4] On packet reception, the edge node checks for the presence of this special link idenitifiers in the Bloom filter and will push the appropriate forwarding label for intra-domain delivery avoiding the publication identifier look-up.

Another solution space of the mapping problem is determined by the notion of information scoping. While scope identifiers (SIds) are meant to guide the pub/sub directives to the suitable rendezvous points in the network for matching, we consider to include the SId in the actual data packets to take edge filtering decisions at this granularity. Since scopes are aggregating data in a semantic layer, it makes sense to think of re-using this aggregation for communal transport services at the forwarding layer.

We are aware of additional challenges that deserve more attention, including extensive evaluation of the trade-offs, Bloom filter check performance, implications of the delay in the multi-step approach, and specific issues w.r.t. content-orientation of inter-domain routing policies. Each of these challenges will be the subject of upcoming papers.

[4]Given the power-law distribution of an AS degree. In practice, maximal 1024 AS neighbours can be assumed [14], [15].

V. Related Work

We are certainly not the first to work with routing&forwarding problems in future networks. However, to the best of our knowledge no prior work has proposed the adoption of the pub/sub paradigm throughout the stack in Internet-scale, and described its relation to routing.

TRIAD [16] was among the first proposals of a *content-based routing* design in the sense that it routes on URLs by mapping fully qualified domain names (FQDN) to next-hops. While similar in the spirit of data-centrism, our work is more ambitious and aims at a finer granularity of content, namely individual pieces of information objects (pub/sub channels or documents).

IP *multicast* can be viewed as a special case of a data-oriented networking service. In practice, a multicast address is a name (cf. a pub/sub topic) rather than a true network identifier. IP multicast issues w.r.t. complexity and deployability incentives have been widely discussed [15]. The authors of [15] revisit the case of IP multicast and propose Bloom filters to aggregate the active multicast groups inside a domain, piggybacking this information in BGP updates. Their IP multicast design also includes a Bloom-filter- based shim header in packets to represent AS-level paths of multicast packets. Our work handles links as more general destination information than IP prefixes and explores more dimensions of the routing&forwarding space.

ROFL [17] proposes Internet-scale routing scheme on flat host identifiers based on neat DHT constructs, but suffers from policy-compliancy issues [12] and larger stretch. AIP [18] is based on a two-level (domain and host) routing architecture with self-certifying domain and host identifiers to account on an Internet scale. The future internetworking proposal in [19] also combines the notion of separating of routing and forwarding using generic link identities.

DONA [3] employs flat self-certifying labels for data objects operated by *find/register* primitives over the legacy IP network. The main difference in our work is that we do not assume underlying IP forwarding. Besides policy and incentive compatibility issues, DONA suffers from scalability problems [12] near Tier-1 operators as they require an entry for each single registered publication in the global network. This suggests that a new data plane design is required to support global data-oriented internetworking.

VI. Conclusion and future work

This paper explored the routing options and problems of a new (inter-)networking layer based on the publish/subscribe paradigm. We moved a step forward towards pure pub/sub routing by dividing the problem in two dimensions. First, we use hierarchical aggregation; second we describe five steps that together solve the problem of routing: (1) Construct the best possible (ideal) forwarding tree on the known topology. (2) Examine the existing forwarding trees and combine them to best match the ideal tree. (3) If it does not reach all subscribers, or it does not meet any additional requirements, select tree-creation and gap-filling strategies, then (4) compute the modifications. Finally, (5) push the necessary information to the affected entities and update the source of the data delivery, if needed.

The ultimate goal we strive to achieve is to find a solution, which has (A) reasonable signalling overhead in dynamic conditions and (B) fairly low stretch, and which minimizes (C) the unnecessarily used network resources and (D) the per-packet overhead. Finding a right balance will be crucial for future work. Our plan for future work also includes validating the overall solution with implementation (NetFPGA routers, FreeBSD nodes) and ns-3 simulation works.

Acknowledgment

We would like to thank Petri Jokela, Jarno Rajahalme and Somaya Arianfar for their valuable comments. This work was supported by the EU FP7 PSIRP project under grant INFSO-ICT-216173.

References

[1] V. Jacobson, "If a clean slate is the solution what was the problem?" Stanford "Clean Slate" Seminar., Feb 2006.

[2] C. Esteve Rothenberg, F. Verdi, and M. Magalhães, "Towards a new generation of information-oriented internetworking architectures," in *First Workshop on Re-Architecting the Internet*, Madrid, 2008.

[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," *SIGCOMM CCR*, vol. 37, no. 4, pp. 181–192, 2007.

[4] D. Trossen (ed.), "Conceptual Architecture of PSIRP Including Subcomponent Descriptions (D2.2)," http://psirp.org/publications, June 2008.

[5] M. Särelä, T. Rinta-aho, and S. Tarkoma, "RTFM: Publish/subscribe internetworking architecture." ICT Mobile Summit, Stockholm., 2008.

[6] J. Su, J. Scott, P. Hui, J. Crowcroft, E. de Lara, C. Diot, A. Goel, M. Lim, and E. Upton, "Haggle: Seamless networking for mobile applications," in *Ubicomp*, 2007, pp. 391–408.

[7] H. Holbrook and B. Cain, "Source-Specific Multicast for IP," RFC 4607, August 2006.

[8] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, 2003.

[9] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[10] P. Jokela, A. Zahemszky, C. Esteve, S. Arianfar, and P. Nikander, "LIPSIN: Line speed publish/subscribe inter-networking," http://www.psirp.org/files/Deliverables/PSIRP-TR09-0001-LIPSIN.pdf, PSIRP EU FP7 project, Tech. Rep., 2009.

[11] R. C. Merkle, "A certified digital signature," in *CRYPTO '89: Proceedings on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1989, pp. 218–238.

[12] J. Rajahalme, M. Särelä, P. Nikander, and S. Tarkoma, "Incentive-compatible caching and peering in data-oriented networks," in *First Workshop on Re-Architecting the Internet*, Madrid, Dec 2008.

[13] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture," RFC 4655, Aug. 2006.

[14] S. Jiang, "An addressing independent networking structure favorable for all-optical packet switching," *SIGCOMM CCR*, vol. 37, no. 1, pp. 17–28, 2007.

[15] S. Ratnasamy, A. Ermolinskiy, and S. Shenker, "Revisiting ip multicast," in *Proceedings of ACM SIGCOMM'06*, Pisa, Sep. 2006.

[16] D. R. Cheriton and M. Gritter, "Triad: A new next-generation internet architecture," http://www-dsg.stanford.edu/triad/, July 2000.

[17] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, "ROFL: Routing on flat labels," in *Proceedings of ACM SIGCOMM'06*, Pisa, Italy, Sep. 2006.

[18] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable internet protocol (aip)," in *ACM SIGCOMM*, 2008.

[19] L. B. Poutievski, K. L. Calvert, and J. N. Griffioen, "Routing and forwarding with flexible addressing," *Journal Of Communication and Networks*, vol. 9, pp. 383–393, Dec. 2007.