Bootstrapping a Publish/Subscribe Information Centric Network

Christos TSILOPOULOS, Dimitris MAKRIS and George XYLOMENOS
Department of Informatics
Athens University of Economics and Business
76, Patission Str., 10434, Athens, Greece
Tel: +30 210 8203693, Fax: +30 210 8203686
{tsilochr, makrhsd, xgeorge}@aueb.gr

***Abstract*—In this paper, we focus on the bootstrap operation of a publish/subscribe information centric network. We consider a set of interconnected network nodes and describe how they organize themselves into a fully functional network by *publishing* and *subscribing* to control plane information. Network bootstrap includes establishing point-to-point communication between network elements, exchanging topological information and setting up the *rendezvous system*. We showcase the network's publish/subscribe service with a complete example.**

***Index Terms*—information centric networks, publish/subscribe, bootstrap, rendezvous system**

I. Introduction

The tremendous growth of the Internet over the last decade has transformed it into a global network of information, diverging from its initial design goal of a resource sharing computer network. Internet users are interested in *what* is available in the Internet rather than connecting to a particular host. The need to provide Internet access to an increasing number of users whilst providing efficient access to information has led the networking research community to reconsider networking fundamentals and principles and propose new, content oriented network architectures [1], [2], [3], [4], [14].

Research initiatives such as CCN [5], PSIRP [6], 4WARD [7] and more recently NDN [8], SAIL [16] and PURSUIT [9] identify the need to further study such networking approaches, promoting the notion of Information Centric Networking. Information Centric Networks (ICNs) place information in the core of the networking protocols, throwing away any host-centric abstractions. In ICNs, datagram packets no longer contain host addresses; instead, the header of a network datagram contains an information label. The network is therefore aware of what is being transferred, allowing core network elements to apply a number of mechanisms that i) increase user application throughput (e.g. using in-network caches and multicast communication), ii) ensure secure access to information (e.g. by applying access control directly on the network packet), iii) reduce unwanted traffic (e.g. preventing spam and DDoS attacks).

PURSUIT, an EU FP7 research project, follows the design guidelines initially set by PSIRP, proposing to build an ICN based on the publish/subscribe paradigm. A publish/subscribe network considers information producers as *publishers* and information consumers as *subscribers*, providing users with two basic network operations: *publish* and *subscribe*. The network's core functionality is to *rendezvous*, i.e. to match subscriptions with publications, notify publishers of these matches and orchestrate the information dissemination from publishers to subscribers.

In this paper, we describe the bootstrapping process for a publish/subscribe network. We consider a set of interconnected nodes and describe how they are organized into a fully functional network using only the *publish* and *subscribe* primitives. We show how network elements *publish* and *subscribe* to control plane information in order to i) establish point-to-point communication with neighbouring nodes, ii) exchange topological information and iii) set up the *rendezvous system* which stores publications and matches them to subscriptions. We then present a complete example of publishing and subscribing in a simple network setup with two users and a single rendezvous point.

II. A publish/subscribe Internet architecture

Our work is based on the Rendezvous, Topology, Forwarding and Media architecture (RTFM) [10]. In RTFM, publishing information to the network does not involve the delivery of information; it is an announcement for the availability of information that can be sent by users. The network tracks the available information and when a user subscribes to it, the network locates the publisher that holds the publication and instructs the publisher to deliver the data to this subscriber. The rendezvous concept of RTFM differs from other rendezvous based networking architectures. In i3 [14] for example, publishing involves transmitting information to the rendezvous point and then forwarding it to subscribers, thus suffering from network path stretch. RTFM orchestrates data dissemination directly from publishers to subscribers over optimal delivery trees, as computed by a topology function.

The architecture separates the networking operations into three distinct functions: Rendezvous, Topology and Forwarding. The Rendezvous function is responsible for matching user subscriptions to publications and locating publishers. The Topology function maintains a topological view of the network and creates optimal delivery paths for disseminating information. The Forwarding function implements the actual data transmission towards the subscribers.

We consider three types of network elements in the system: *routers*, *rendezvous points* and *user hosts*. Routers are simple, low complexity, packet switching elements that maintain a topological view of the network through a link state routing protocol. Routers are assigned a flat, semantically free, topology independent node identifier. Node Ids are unique within a network area, but they do not need to be globally unique, unlike IP addresses. Rendezvous Points (RVPs) are special network nodes that keep track of available publications. Publications and subscriptions are routed to RVPs which perform subscription-publication matching. User hosts gain access to the network through gateway access routers. Users do not have access to topology information; they communicate only with their gateway routers which proxy user publications and subscriptions to the network. User hosts generate network traffic only on demand, as we will see in section IV-C.

Publications are identified by a pair of flat, semantically free identifiers, namely the Scope Id and the Rendezvous Id [11]. The Rendezvous Id (Rid) is a statistically unique identifier used as a label for the publication. The Scope Id (Sid) is used to organize information items into larger collections. The combination of Sid/Rid allows applications to build relationships among information items, for instance, creating information taxonomies, suggesting the location of information or enforcing access control policies.

We use LIPSIN [12] for the forwarding function. LIPSIN is a source routing forwarding fabric that encodes delivery paths into fixed size, in-packet bloom filters called zFilters. In LIPSIN each network link is assigned with a Bloom filter identifier called its Link Identifier (LID). LIDs are unidirectional; each link is identified with a pair of LIDs, one for each direction. To create a forwarding path between two network nodes, LIPSIN encodes all LIDs in the path into a single Bloom filter using binary OR. Packets are transferred in the network using the zFilter as a source route header. When a node receives a packet, it extracts the zFilter and compares it with its outgoing LIDs using binary AND. If the result of the comparison between the zFilter and the LID matches the ANDed LID, then the node assumes that this LID is encoded in the path and transmits the packet on that link. To define the recipients of a packet along a path, nodes are also assigned with a Virtual Link Identifier (VLID) pointing to themselves. Path construction includes also encoding the VLIDs of the destination nodes (one or more). When nodes receive packets, they also compare the zFilter with their VLID and if the result matches that VLID, they deliver the packet to the system's networking software.

III. Network bootstrap

In this section we describe the bootstrap operation of our publish/subscribe ICN following a bottom up approach. We start by describing the publish/subscribe communication among processes in a single node and continue with the description of connectivity establishment between two nodes. Next, we describe a publish/subscribe version of a link state routing protocol that runs in network routers for managing topological information. In section IV we show a publish/subscribe discovery mechanism for locating available Rendezvous Points in the network.

A. A publish/subscribe node

Figure 1 shows an outline of a publish/subscribe node. The heart of the node is the Local Rendezvous Component (LocRC) which provides a publish/subscribe inter process communication (IPC) mechanism to system processes. The LocRC stores subscriptions issued by processes and forwards received publications to processes with matching subscriptions. Network protocols are organized as publish/subscribe processes around LocRC, forming a *protocol circle* instead of a typical OSI protocol stack [15].

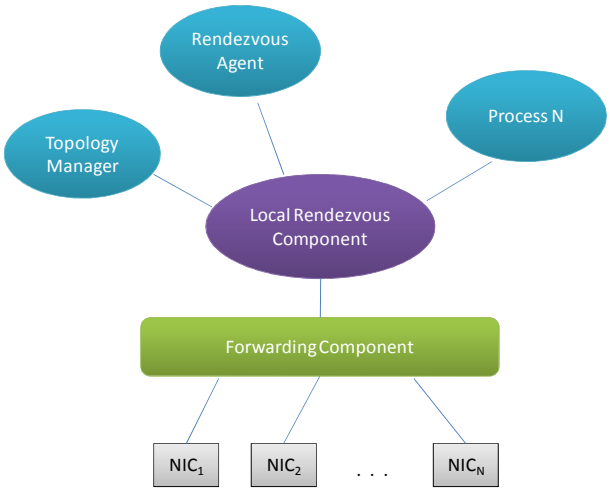

Fig. 1. Internal organisation of a publish/subscribe network node.

Below LocRC lies the Forwarding Component (FwdC) which is directly connected to the node's network interfaces. In order to transmit data to the network, processes have to pass publications to the FwdC via the local publish/subscribe IPC. For example, assume that node X wants to send the publication *(S, R, [data])* to node Y, with *S* being the Scope Id and *R* the Rendezvous Id. X must obtain the forwarding path pointing from X to Y, encapsulate both the path and the publication into a new publication

$$(Fwd\_Sid,\ Fwd\_Rid,\ [path_{XY},\ (S,\ R,\ [data])])$$

and then publish it the LocRC. Node X's FwdC, which is already subscribed to *Fwd_Sid/Fwd_Rid*, receives this publication, extracts the encapsulated publication and transmits the packet according to the embedded forwarding path, which in this case points to node Y. The FwdC at Y receives the packet, removes the path from the publication's data and publishes the original publication *(S, R, [data])* to Y's LocRC. If there are any processes in node Y that had already subscribed to *(S, R)*, Y's LocRC will deliver the publication to them. For the remainder of the paper, communication between processes in different machines embodies this message passing between FwdCs.

B. Point to point communication

When a publish/subscribe node is connected to a communication link, that node's FwdC broadcasts a publication to a well known Sid/Rid pair agreed between the FwdCs, e.g. *Fwd_Sid/Fwd_Link_Connect_Rid* and announces its VLID to neighbouring nodes. When FwdCs receive such publications, they respond back with a new publication containing their own VLID. After the VLID exchange phase is completed, FwdCs set a new LID for the established link and announce the event locally by publishing the LID/VLIDs information under *Fwd_Sid/Link_Established_Rid*. Communication with neighbouring nodes is now feasible as it only requires encoding the LID and VLID of the new node into a forwarding path. To support multi-hop communication, routers need to obtain topology information, as described in the following section.

The VLID exchange is a rather simple approach. [13] presents a thorough study of link establishment between publish/subscribe nodes.

C. Publish/subscribe topology management

The Topology Management Component (TMC) is an implementation of the network's Topology function and is responsible for providing forwarding paths for disseminating information across the network. TMCs manage topological information via a distributed link state routing algorithm. Topology management is required only in switching elements of the network, thus TMCs are installed only in network routers. TMCs in different machines communicate with each other by publishing and subscribing to a Sid/Rid pair agreed a priori, e.g. *TMC_Sid/TMC_Rid*.

When a TMC instance starts up, it subscribes to *Fwd_Sid/Link_Established_Rid* so that it is notified whenever the node establishes a new link connection. Upon link establishment, the TMC creates a *link state connectivity* publication under *TMC_Sid/TMC_Rid*. *Link state* publications contain the router's Node Id, its outgoing LIDs and the neighbouring Node Ids. The TMC computes the path destined to the neighbouring node and sends the publication (as described in section III-A). The neighbouring TMC receives the publication, updates its local network graph and responds back with a new *Link state* publication containing its own *link state connectivity* information. TMCs forward received *link state* publications to their neighbours in a recursive manner so that a *link state* publication reaches all network routers. TMCs track recent *link state* publications and discard duplicates in order to prevent link advertisements from looping infinitely.

Router complexity and CPU requirements are reasonably low. Routers maintain the topology graph in memory, which is proportional to the number of routers in the network and not proportional to the available information items, as in [4]. Network overhead caused by the link state routing protocol remains proportional to the number of routers, which is significantly lower than flooding information about all available information items in the network. Last, but not least, packets are source routed, thus routers do not perform CPU intensive routing table lookups for each incoming packet, as in hop-by-hop switching networks.

IV. Rendezvous System discovery

Publications and subscriptions sent by users are routed to designated network nodes called Rendezvous Points (RVPs). RVPs keep track of available publications and perform the matching between subscriptions and publications. We assume that RVPs are computing systems with enhanced memory and processing capabilities in order to support the *rendezvous* functionality. In order to *rendezvous*, there must be at least one RVP present in the network. RVP presence is published to network routers in the same way that link state publications are flooded. System processes installed in routers and RVPs, called Rendezvous Agents (RVAs), store the location of RVPs. RVAs residing in users' access routers operate as *rendezvous proxies*; they capture publications and subscriptions sent by users and forward them towards the RVP. The coordinated operation of RVAs and the RVP comprises the *Rendezvous System*.

In section IV-A we describe an RVP discovery mechanism with a single RVP in the network. Section IV-B shows an example of two users publishing and subscribing to the network while in section IV-C we discuss an extension of the RVP discovery mechanism that supports multiple RVPs.

A. Rendezvous Point announcement

Figure 2 shows a network of 3 routers, A, B and C. The network operator installs an RVP to A. We call A the *RVP proxy router*. For the rest of the section we consider that RVAs exchange information by publishing and subscribing to a well-known pair of identifiers among RVAs, e.g. *RVA_Sid/RVA_Rid*.

When the RVP establishes a link with A, its RVA publishes the RVP's presence to A's RVA. A's RVA requests its resident TMC to provide a forwarding path to neighbouring routers (B and C) and sends them a new publication indicating that A has an RVP attached to it. RVAs in B and C receive the publication, store the Node Id of A and forward the same publication to their neighbours so that the RVP announcement is pushed throughout the network. Whenever a new router is connected to the network, its RVA requests the RVP location in a pull based manner: the newly attached router sends a publication to its neighbouring RVAs requesting the identity of the *RVP proxy router*. When RVAs receive such *discovery* publications, they immediately respond back with the *RVP proxy router's* Node Id.

To send publications or subscriptions to the network, RVAs in user hosts capture and send them to the RVA in their gateway router. Gateway routers' RVAs operate as *rendezvous proxies* and forward these publications to the *RVP proxy router* which in turn forwards them to the RVP. If no RVP is known, access router RVAs reply to user hosts with an error. When the RVP receives a subscription, it performs a lookup and if a matching publication is found, the RVP creates the forwarding path between the publisher and the subscriber and instructs the publisher to deliver the publication over the specified path. To create paths between network nodes, the RVP needs to have access to topology information. For this purpose, the RVP maintains a limited version of the TMC which is allowed to obtain the topology graph from its proxy router. TMC's in

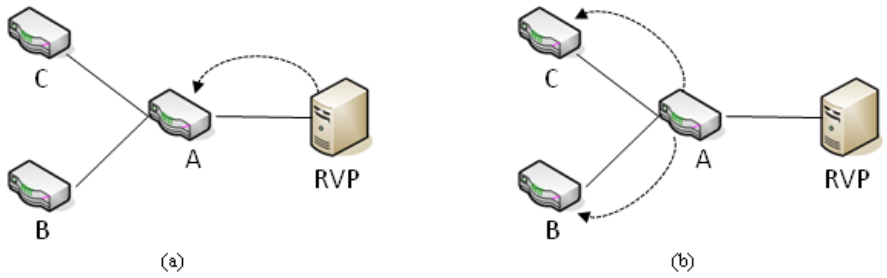

Fig. 2. (a) RVP publishes its presence to A. (b) Router A publishes to neighbouring routers, B and C, that an RVP is attached to A. Routers B and C store the fact that A is a Rendezvous Proxy Router.

RVPs do not publish link state information however and they do not participate in the link state routing protocol.

B. Publishing and subscribing

Figure 3 shows an example of two users, Bob and Alice, connected to a publish/subscribe information centric network with a single RVP attached to router R.

Bob wants to announce a publication labelled *BobSid/BobRid* to the network. Bob's host RVA creates a special publication

(RVA_Sid, Pub_Announce_Rid, [Bob, BobSid/BobRid])

and sends it to its gateway router, G_B (step 1). G_B's RVA receives the publication and records that Bob announced a publication labelled *BobSid/BobRid* to a local database. Then, the RVA at G_B replaces Bob's identity with its Node Id and forwards the publication

(RVA_Sid, Pub_Announce_Rid, [G_B_nodeid, BobSid/BobRid])

to R (step 2) which in turn forwards the announcement to the RVP (step 3). At this point, the RVA running inside the RVP stores the *[G_B_nodeid, BobSid/BobRid]* information to its local database. This record indicates that a host connected to router G_B has announced the availability of data with the label *BobSid/BobRid*.

Next, Alice subscribes to the data labelled *BobSid/BobRid*. Alice's host RVA encapsulates the subscription into a special publication

RVA_Sid, Subcribe_Rid, [Alice, BobSid/BobRid])

and sends it to Alice's gateway router, G_A (step 4). The RVA in G_A receives the subscription, stores it to a local subscription table, replaces Alice's identity with G_A's Node Id and forwards the publication

RVA_Sid, Subcribe_Rid, [G_A_nodeid, BobSid/BobRid])

to the RVP Proxy Router R (step 5) which eventually forwards it to the RVP (step 6).

The RVP matches the subscription sent by G_A with the publication announced by G_B and requests its TMC to create the forwarding path from G_B to G_A. The next step for the RVP is to send a publication to G_B instructing it to deliver the publication to G_A. The RVP creates a new publication

(RVA_Sid, Pub_Instruct_Rid, [path (G_B, G_A), BobSid/BobRid])

and sends it to G_B (step 7). G_B's RVA receives the subscription, looks up in its local database and matches it with Bob's previous announcement. G_B's RVA updates the received forwarding path by adding Bob's outgoing LID. The zFilter now contains the path from Bob's host to Alice's gateway router. The instruction is handed to Bob (step 9) and Bob's host transmits the publication to the received path (step 10). When G_A receives the publication, it searches its local subscription table, matches the publication with Alice's subscription and forwards it to Alice's host (step 11).

C. Multiple RVPs

In section IV-A we presented a Rendezvous Service Discovery scheme with a single RVP, implying a centralized Rendezvous System. Network operators may choose to install many RVPs, building a distributed Rendezvous System. If the RVPs are all connected behind the same router, then there is no difference in the system, since all routers send publications and subscriptions to the same *RVP proxy router*. However, for reasons of efficiency, a network operator may choose to install RVPs in various locations in the network. For example RVPs may be installed near populated areas of the network in order to reduce rendezvous delay. In this case, RVP announcements contain a *hop count* field which is incremented in each hop. RVAs store a list of (*RVP proxy router*, *distance*) pairs. When access routers enter the system, they discover the available RVPs and configure their RVAs to forward publications and/or subscriptions to the nearest RVP.

Once a message reaches an RVP, it is a matter of the Rendezvous System's internal organization how to handle it. For instance, a distributed Rendezvous System implementation may partition the Scope Id space among RVPs, like DHTs partition ID space among peers. In this case, when an RVP receives a publication and/or subscription, it inspects its Scope Id and forwards it to the RVP that handles the particular part of the Sid space. The difference with DHT key based routing is that the Rendezvous System in not an overlay system but has access to topological information. Hence, RVPs know how

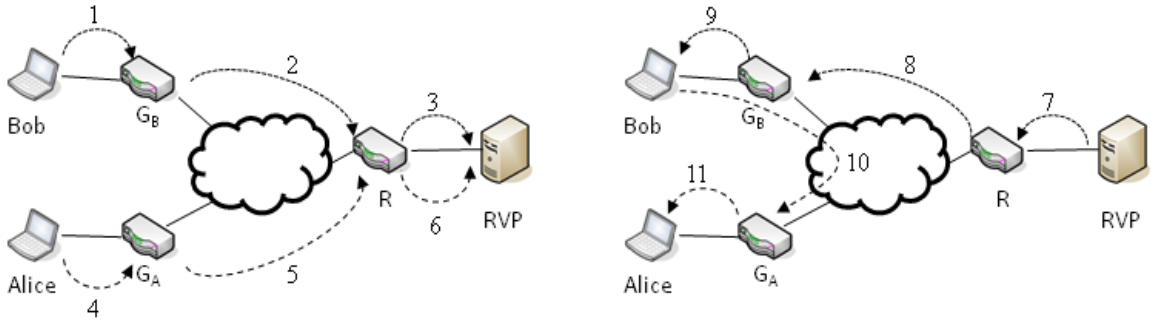

Fig. 3. Bob and Alice are connected to a publish/subscribe network with a single RVP. Steps 1-3: Bob announces data availability to the network. Steps 4-6: Alice subscribes to Bob's data. Steps 7-9: the Rendezvous Point instructs Bob to send the data to Alice. Steps 10-11: Actual data transmission.

to forward messages to other RVPs directly on shortest paths, thus eliminating path stretch.

V. Conclusion and future work

In this paper, we described a bootstrapping scheme for an RTFM based publish/subscribe network. The bootstrapping operation includes link establishment between publish/subscribe nodes, topology management through a publish/subscribe variance of link state routing and rendezvous point discovery. We showcased a complete example of publishing and subscribing to the network.

We have implemented the bootstrap operation in the ns3 simulator [17]. Our plans include building a distributed Rendezvous System, as discussed in section IV-C. We plan to extend the architecture to support the co-existence of multiple Rendezvous Systems provided by different operators. Evaluation will focus - among other metrics - on rendezvous cost and possible solutions for reducing it. Future work also includes interconnecting publish/subscribe networks to a publish/subscribe internetwork and implementing end-to-end transport.

Acknowledgments

The work reported in this paper was supported by the FP7 PURSUIT project under contract FP7-ICT-2010-257217.

References

[1] M. Gritter and D.R. Cheriton, An architecture for content routing support in the internet, in Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems, 2001
[2] H. Balakrishnan, K. Lakshminarayanan, et al, A layered naming architecture for the internet. In ACM SIGCOMM, 2004
[3] T. Koponen, M. Chawla, B.-G. Chun, et al, A data-oriented (and beyond) network architecture, In ACM SIGCOMM, 2007.
[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, Networking Named Content, In ACM CoNEXT, 2009
[5] PARC Content Centric Networking, *http://ccnx.org*
[6] PSIRP: Publish Subscribe Internet Routing Paradigm, EU FP7 research project, *http://psirp.org*
[7] 4WARD, EU FP7 research project, *http://www.4ward-project.eu/*
[8] Named Data Networking, NSF Future Internet Architecture project, *http://www.named-data.net/*
[9] PURSUIT: Publish Subscribe Internet Technology, EU FP7 research project, *http://fp7-pursuit.eu/*
[10] M. Sarela, T. Rinta-aho, S. Tarkoma, RTFM: Publish/Subscribe Internetworking Architecture, In ICT Mobile Summit, 2008
[11] N. Fotiou, G.C. Polyzos and D. Trossen, Illustrating a Publish-Subscribe Internet Architecture, In the 2nd Euro-NF Workshop on Future Internet Architectures
[12] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander, LIPSIN: line speed publish/subscribe inter-networking, In ACM SIGCOMM 2009
[13] J. Kjallman, Attachment to a Native Publish/Subscribe Network, In ICC Workshops, 2009
[14] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, Internet Indirection Infrastructure, In ACM SIGCOMM, 2002
[15] P. Jokela (editor), Progress Report and Evaluation of Implemented Upper and Lower Layer, PSIRP deliverable D3.3, available at *http://psirp.org/publications*
[16] SAIL: Scalable and Adaptive Internet Solutions, EU FP7 research project, *http://www.sail-project.eu/*
[17] ns3 network simulator, *http://www.nsnam.org/*