Caching and Mobility Support in a Publish-Subscribe Internet Architecture

George Xylomenos, Xenofon Vasilakos, Christos Tsilopoulos, Vasilios A. Siris, and George C. Polyzos, Athens University of Economics and Business

ABSTRACT

The Internet is straining to meet demands that its design never anticipated, such as supporting billions of mobile devices and transporting huge amounts of multimedia content. The publish-subscribe Internet (PSI) architecture, a clean slate information-centric networking approach to the future Internet, was designed to satisfy the current and emerging user demands for pervasive content delivery, which the Internet can no longer handle. This article provides an overview of the PSI architecture, explaining its operation from bootstrapping to information delivery, focusing on its support for network layer caching and seamless mobility, which make PSI an excellent platform for ubiquitous information delivery.

INTRODUCTION

The current Internet architecture and its fundamental protocol, IP, were designed when networking was oriented toward remote access to mainframe computers. The Internet basically overlaid packet switching on top of the circuit-switched telephone network, but the communication model remained that of connecting two *fixed* endpoints, called *hosts*. While internal network nodes were only concerned with packet switching, hosts were responsible for providing error, flow, and congestion control on an end-to-end (E2E) basis [1]. These design choices led to a reliable and efficient network that expanded from tens to billions of nodes.

Given that user needs have changed dramatically since the advent of the Internet, its host-centric architecture can no longer meet current and emerging demands. Mobile devices are expected to overwhelm the Internet, yet IP has a hard time accommodating them, requiring cumbersome solutions such as *Mobile IP*, which employs tunnels to hide mobile nodes (MNs) behind fixed addresses. The usage model of the Internet has also changed: instead of sharing remote resources, users predominantly want to share information, regardless of the endpoint holding it. This demand is currently met by content distribution networks (CDNs) and peer-to-peer (P2P) overlays. CDNs proactively push content to servers and then manipulate the Domain Name System (DNS) so as to serve users from nearby servers, while P2P overlays allow users to exchange data by forming application-level overlay networks with their own routing and forwarding logic.

Patches to the Internet architecture, including Mobile IP, CDNs and P2P overlays, can only superficially bridge the gap between the host-centric model and current user needs. As data is inherently opaque to the Internet, these solutions have to translate information semantics to endpoint semantics, at a considerable cost. Mobile IP enforces triangular routing, thus injecting extra traffic to the Internet. CDNs misuse the DNS to redirect user requests, thus preventing DNS caching and causing scalability problems [2]. Finally, P2P overlays cause traffic asymmetries and routing policy violations [3].

Recognizing the limitations of the Internet, many research groups have proposed information-centric networking (ICN) architectures, which focus on content access rather than endpoint communication [4]. The publish-subscribe Internet (PSI) architecture, presented in this article, realizes the design principles of the PSIRP and PURSUIT projects [5] for a clean-slate architecture meeting current user needs. PSI focuses on fetching information from wherever it is available, avoiding addressing and routing assumptions that could inhibit scaling and ubiquitous access to information. Moreover, it facilitates native multicast, which can improve information delivery efficiency and scalability.

Information is the main entity in PSI, with users announcing the availability of it or interest in it. Information items are identifiable, and thus possible to cache and replicate in a uniform manner. The asynchronous nature of the PSI architecture simplifies resynchronization after MN handoffs, while its caching abilities further enhance support for the seamless mobility of users and information.

The goal of this article is to explain how PSI operates, from network bootstrapping to information delivery. We specifically focus on how PSI addresses the deficiencies of the Internet in the areas of network layer caching and mobility, thus providing an excellent platform for ubiquitous content delivery. Since different parts of the architecture are at different stages of develop-

0163-6804/12/$25.00 © 2012 IEEE

ment, the article describes a simplified yet integrated plan of how it all fits together.

In the remainder of this article we first outline the PSI architecture and then explain the design of a PSI network. We proceed to explain how caching operates in PSI, and then focus on mobility. We finally conclude and provide a status report.

Architecture Overview

Naming Information

In PSI information items are named by a pair of identifiers, the scope identifier (SID) and the rendezvous identifier (RID). The RID is provided by an application-specific function and is a flat, fixed-length, semantic-free identifier. The SID denotes the scope in which an item belongs. Scoping yields two benefits: first, it allows organizing information items into collections (e.g., a set of photographs from a trip); second, it is a flexible means of enforcing access control and dissemination policies on information collections (e.g., a collection could be visible only to a specific network domain).

Scopes are hierarchically structured, forming parent-to-child relationships. An SID is represented by a variable-length list of fixed-length tokens (same as RIDs), for example, */X/Y/Z*, where *Z* is a subscope of */X/Y* and *Y* is a subscope of *X*. An information item may belong to one or more scopes. For instance, a photograph may belong to a collection from a trip as well as to a collection of beach photographs. An RID must be unique in the scope(s) to which it belongs, and an SID must also be unique within its parent scope. The system rejects duplicate identifier pairs, but it does not otherwise get involved with the way identifiers are generated.

Network Primitives

PSI provides users with two primitives over information items, publish and subscribe. *Publish* is used by information producers to announce the availability of an information item to the network. *Subscribe* is used by information consumers to express their interest in receiving an information item. Users can subscribe to either specific items or scopes. In the latter case the subscriber declares an interest in receiving all items belonging to that scope. The granularity and nature of information items is not mandated by PSI. Announced items can be files of any size, chunks of files, channels of streaming media, or services. The actual information transfer takes place via transport protocols built on top of these basic primitives.

Separation of Functions

Following the design methodology described in [6] the PSI architecture is based on three distinct functions. *Rendezvous* is responsible for matching subscriptions to announcements for information items. *Topology Management* is in charge of discovering network topology and computing delivery paths. Finally, *Forwarding* handles the actual transmission of data. These *logical* functions are implemented by a distributed collection of *physical* nodes, as described in the next section.

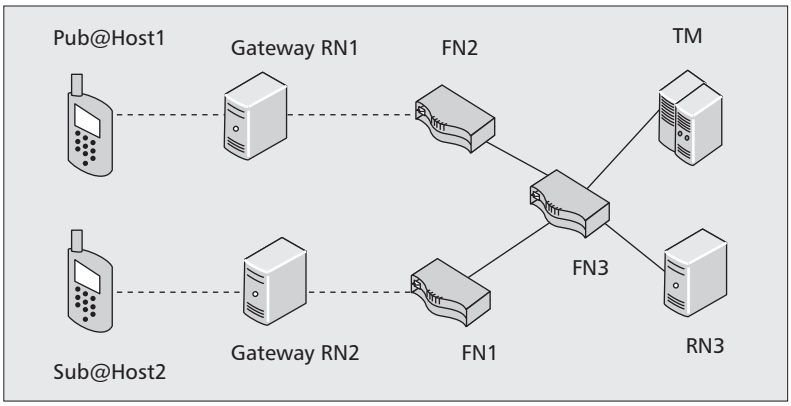

Figure 1. *Basic network elements.*

In order for an information item to be retrieved, three steps are involved. First, an information producer must *publish* the availability of the item to the network. The announcement is propagated by network elements to the Rendezvous function, which tracks available information items and the hosts announcing them. Second, an information consumer must subscribe to the item. The subscription is also propagated to the Rendezvous function, which performs the publication-subscription matching. Once a publisher is found, the Rendezvous function asks the Topology Management function to construct a forwarding path from the publisher to the subscriber and encode it into a forwarding identifier (FID). The FID is then sent to the publisher as a notification. On reception of such a notification, the publisher starts transmitting the requested item by passing it to the Forwarding function with the FID as a parameter.

Network Design

Basic Elements

A PSI network consists of the four types of network elements shown in Fig. 1. Forwarding nodes (FNs) are streamlined elements that perform packet switching only; they may employ a stateless source-routing scheme that supports native multicast. Rendezvous nodes (RNs) are more complex elements that perform publication-subscription matching; they may use a distributed hash table (DHT) to locate the RN designated to handle each (SID, RID) pair. Topology managers (TMs) compute paths between nodes; they may use a link-state algorithm, in which FNs and RNs participate, to construct a view of the network topology.

Hosts are located at the edges of the network, gaining access to it via gateway RNs present in their current location. These RNs act as proxies for the hosts, aggregating subscriptions for the same item and distributing received information items to interested hosts; therefore, they must maintain information about each host attached to them. Having hosts hidden behind gateway RNs has two benefits:

- Increased user security and privacy (as with NATs)
- Reduced control plane overhead in the network, due to a smaller topology (hosts do not participate in routing) and reduced ren-

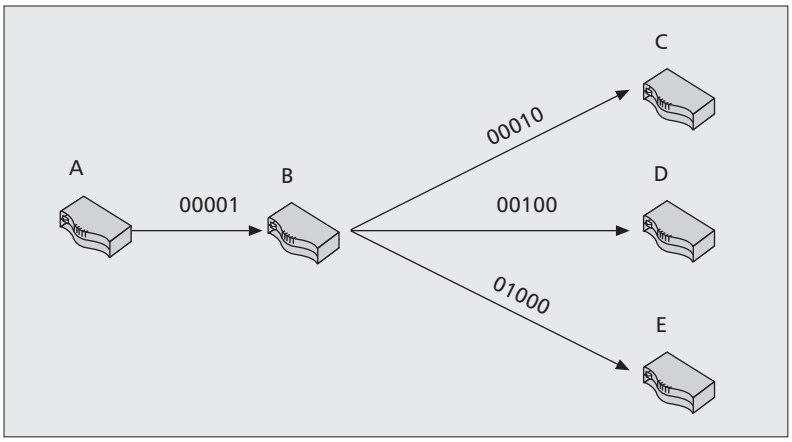

Figure 2. *Source routing example.*

dezvous traffic (concurrent requests for the same item can be merged)

A multihomed host will be located behind (and represented by) multiple gateway RNs; which gateway RNs should be used for each publication and subscription is a matter of current research.

Hosts send their announcements and subscriptions to their gateway RNs, which then forward them to the RN designated to handle the (SID, RID) pair of the information item involved. When an announcement reaches the designated RN, that RN stores the (SID, RID) pair of the information item and enough data to later contact the gateway RN of the publisher. Subscriptions are similarly routed to the designated RN, which then performs the publication-subscription matching to locate a publisher. If a publisher is not found, the subscription is stored at the designated RN, just like an announcement, to be matched by future publication announcements. Once a publisher is found, the RN asks the TM for a *suitable* dissemination path for the requested item, passing to the TM the gateway RNs of the publisher and the subscriber. The TM returns an FID for this path to the RN, which passes it to the publisher's gateway RN. The gateway RN forwards this FID to the publisher, which uses it to transmit the requested item.

In the example of Fig. 1, assume that host 1 announces an item, host 2 subscribes to the same item, and the designated RN for the (SID, RID) pair of that item is RN3. The announcement and subscription would be forwarded via the gateway RNs to RN3. RN3 would find a match between them, ask the TM for a forwarding path from gateway RN1 to gateway RN2, and then return the FID for that path to gateway RN1. Finally, host 1 would start transmitting the information item toward host 2 (via gateways RN1 and RN2) using the returned FID.

Data Forwarding

Packet forwarding in PSI is implemented via LIPSIN [7], a source-routing scheme. In LIPSIN a path is represented as the set of on-path links, encoded into a fixed size bit string using a *Bloom filter*. This bit string serves as the FID of the path and is included in the packet. Each direction of a network link is represented by a link identifier (LID), which is an m-bit string with k bits set to 1 ($k << m$). To construct a source route, we simply OR the LIDs of its constituent links.

For example, consider the three-node path ABC shown in Fig. 2. The FID for this path is 00001 OR 00010 = 00011. This FID is placed in packet headers. When an FN receives a packet, it compares its FID with each LID of its outgoing links. A link j with LIDj is part of the path if FID AND LIDj == LIDj, in which case the FN assumes that LIDj was added to the FID and forwards the packet over link j.

Packets can be delivered over multicast trees to multiple subscribers by simply adding all tree links to the FID; no extra state is required at the FNs, unlike with IP or overlay multicast. In Fig. 2, by ORing the LIDs for BD and BE to the FID for ABC, thus producing 01111, the path is expanded to a multicast tree from A to C, D, and E via B. The forwarding logic relies on bitwise operations on fixed-length bit strings and can be implemented in hardware to support line-speed forwarding.

Since LIPSIN employs the Topology Management function to encode entire paths in FIDs, the network can easily enforce routing policies or provide paths satisfying specific quality of service criteria. FIDs encode unidirectional paths; therefore, by default publishers cannot be reached by subscribers. When bidirectional communication is required (e.g., to return error control feedback to the sender), another FID for a return path must be separately provisioned.

Bloom filters are probabilistic representations of sets, and therefore suffer from false positives: LIDs can falsely match an FID without having been truly added to it, causing the packet to be transmitted over redundant links. A 256-bit FID can handle around 30 links without an excessive rate of false positives, which is sufficient for intradomain forwarding. For interdomain forwarding, each domain can construct its own intradomain FIDs, with packets changing their FID as they enter a new domain. To this end, we are currently exploring the cost and feasibility of FID stacking and FID switching.

Rendezvous Networks and Topology Management

Rendezvous involves publishers, subscribers, and an event notification service, which is provided by the rendezvous network (RENE), consisting of multiple RNs. As previously mentioned, FNs and RNs (but *not* hosts) participate in a link-state routing algorithm in order for the TM to create a topology graph. Each node participating in the link-state algorithm indicates its outgoing LIDs when transmitting link-state announcements (LSAs). In addition, RNs set a special flag in their LSAs to indicate themselves as RNs. This solution is sufficient for intradomain topology management; our current research focuses on combining multiple local TMs to come up with an intradomain global topology management solution that will treat each domain as a simple node.

After discovering the topology, the TM sends

to each RN an FID that can be used to reach the TM. As RNs discover each other via the LSAs, they use the TM to compute paths between them. The RNs jointly implement a DHT (or an equivalent data structure) in order to assign each (SID, RID) pair to one or more RNs, the *designated* RNs for the corresponding information item. A hierarchical DHT structure can be used to provide interdomain rendezvous [8]; our current work focuses on aligning a hierarchical DHT structure with the complex routing policies of the Internet [9].

Data Transport

As discussed above, the PSI network arranges for publishers to receive unidirectional FIDs toward subscribers. This is sufficient to support *channels*, that is, information that can be unreliably delivered over a unidirectional path. The channel abstraction is suitable for live media streams, such as TV and radio. Reliability can be enhanced by adding redundant information into the data stream so as to recover from a target packet loss rate.

Channel sources in PSI announce their availability and wait for notification events. To *tune* in to a channel, a user simply subscribes to it. Rendezvous then takes place and a notification is sent to the channel source. Multicast support for channels requires some extra state: for each channel, the designated RN must maintain a list of subscribers. When a new user tunes in, the RN computes a new FID for multicasting toward all channel subscribers and sends it to the publisher so that subsequent data may also reach the new subscriber without any further action from the network.

In order to support *documents* (i.e., information that requires reliable delivery), the PSI network must also allow feedback to be returned from the subscriber to the publisher. To support this, when users subscribe to documents, the TM computes both publisher-to-subscriber and subscriber-to publisher-paths, and encodes them into FIDs; these paths may be asymmetric if needed. Both FIDs are handed to the publisher, who then passes the FID for the return path to the subscriber, thus establishing bidirectional communication. Since feedback may also be useful for channels (e.g., to adjust transmission rates), we are working on hierarchical schemes for adding feedback paths to multicast channels.

Transport protocols in PSI may employ either push-based or pull-based logic. In the former, the publisher starts sending data packets and waits for acknowledgments from the subscriber, retransmitting lost packets, as in TCP. In the latter, the receiver starts transmitting requests for individual pieces to the publisher, retransmitting requests for packets that do not arrive on time. We are currently researching how these mechanisms can be expanded to cover flow and congestion control, as well as multicast dissemination and multipath transmission. We expect different transport protocols to coexist, to serve the needs of different applications.

While PSI does not prescribe the size of the information items that are entered into the RENE, the rendezvous process is costly, so we expect it to be used only for coarse-grained information items. These items will be subdivided into chunks, identified by (SID, RID) pairs algorithmically generated from the identifier of their parent item. Given a bidirectional path between a publisher and a subscriber produced by the *slow rendezvous* process described above for the parent information item, the subscriber will then be able to directly request its constituent chunks from the publisher; this *fast rendezvous* process *bypasses* the RENE.

Caching and Replication

In PSI any requested information item can be fetched from any possible source, including caches or replication points. In addition, any storage granularity can be accommodated since information items are identified in a uniform way: entire items are identified by an (SID, RID) pair, while their chunks can be identified by algorithmically generated identifiers. Chunks of the same item can even be fetched from different sources via different paths. In contrast, IP packets contain opaque data; therefore, the network cannot match them with information items without resorting to application logic. Caching on the Internet thus requires a different solution for each application.

While caching and replication are equivalent from the viewpoint of subscribers, they reflect different approaches to information dissemination. Replication uses distributed algorithms to proactively push and update information toward network nodes that may later need to serve it, while caching uses local algorithms to opportunistically store received data in order to serve future requests.

PSI supports both replication and caching using the same network primitives. Out of a wide spectrum of possible schemes, we focus here on three scenarios that offer insights into the PSI architecture and can be contrasted readily to what the Internet offers: on-path caching, off-path caching, and content replication.

Out of a wide spectrum of possible schemes, we focus here on three scenarios that offer insights into the PSI architecture and can be contrasted readily to what the Internet offers: on-path caching, off-path caching, and content replication.

On-Path Caching

On-path caching takes place on nodes residing on the path from the publisher to the subscriber, based on their local policies, as shown in Fig. 3 (in the following figures we omit the gateway RNs for clarity). After subscriptions are sent to the local RENE (arrows 1a and 1b), that is, the RNs within the subscriber's network domain, they are forwarded to the global RENE (arrow 2), which notifies an appropriate publisher (arrow 3). The publisher starts transmitting data toward the subscribers (arrows 4 and 5). Any on-path FN can locally store a copy. If a subscriber later requests these data in a pull-based manner, using the fast rendezvous process described above, these requests may be served directly by such an FN if it happens to reside on the path from the subscriber to the publisher.

This mechanism is very lightweight since the RENE is not notified about cached data; hence, it can be implemented at line speed. While on-path caching is only applicable if caches happen to be on the right path, which may not be the case for asymmetric paths, it can be very useful for multicast error control: caches on a multicast

tree can respond to retransmission requests from downstream subscribers, thus avoiding feedback implosion. On-path caching is similar to the approach taken in CCN [4], which explicitly uses symmetric paths. On-path caching is possible on the Internet but only at the application layer, due to the lack of data identification at the network layer; it also requires configuration of caching points, for example, selecting a proxy server for a web browser.

Off-Path Caching

Access networks are strongly motivated to locally cache popular information items so as to avoid using their provider links, thus reducing connectivity costs and improving user experience. Since with PSI any network node can serve as an information provider, subscribers can take advantage of data stored at either hosts or dedicated caches in a uniform manner.

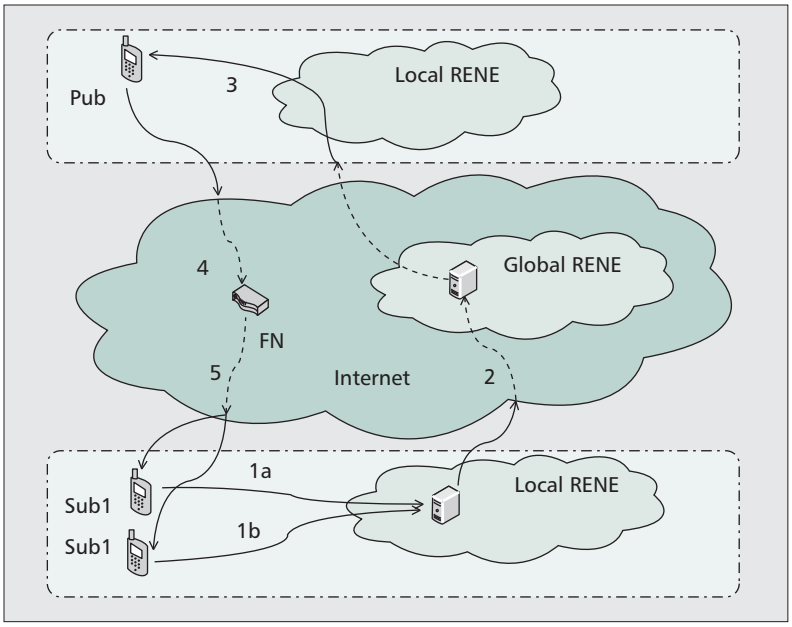

Figure 3. *On-path caching.*

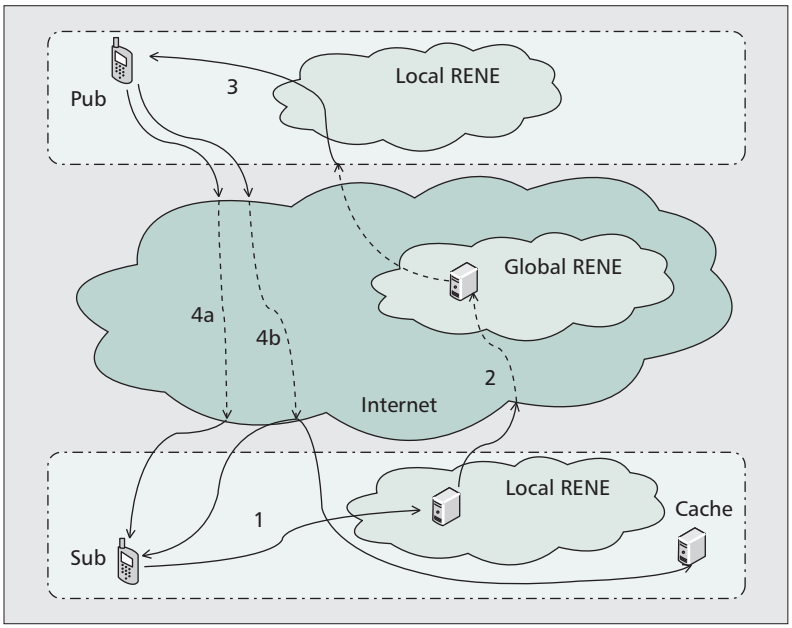

Figure 4. *Off-path caching.*

Figure 4 depicts how an access network can exploit off-path caching to avoid inter-domain rendezvous and data traffic. The subscriber sends a subscription to the local RENE (arrow 1) which checks if the requested information item is locally available. If not, then there are two cases. If the local RENE decides not to cache the information item, it simply forwards the subscription to the global RENE (arrow 2) which notifies the publisher (arrow 3) to start forwarding data (arrow 4a). On the other hand, the local RENE may decide to cache the information item by adding a local caching element to the subscription request; how to decide whether the item should be cached and how to select a caching element is a matter of current study. The modified request is passed to the global RENE and the notification to the publisher, as above (arrows 2 and 3), but now the publisher multicasts the information towards both the original subscriber and the local caching element (arrow 4b). After receiving the information item, the local cache announces its availability to the local RENE. If the subscriber also decides to cache the information item, he can simply announce this to the local RENE.

Unlike on-path caching, off-path caching requires additional overhead, since cached data are announced to the local RENE, but it allows subscribers to be served by off-path caching points. The local RENE may not propagate such announcements to the global RENE; this allows serving local subscribers without incurring inter-domain signaling. Off-path caching is very hard to achieve on the Internet as it requires coordination between many application-level caches; it is in a sense closer to a localized P2P overlay solution.

Content Replication

The PSI architecture can also accommodate dedicated content replication operators, similar to CDNs on the Internet, which mediate between content providers who wish to offload their content distribution and access networks who wish to minimize interdomain traffic and improve user experience. In PSI, these CDNs offer an interdomain solution that can be seamlessly integrated with the intradomain solution described in the previous section.

Figure 5 shows how this can be achieved. After receiving a subscription (arrow 1) and not finding a local match, the local RENE propagates it to the global RENE (arrow 2). The global RENE will then choose a publisher after consultation with the Topology Management function; it may either notify the original content provider to deliver the content (arrows 3a and 4a) or a suitably located CDN node to do so (arrows 3b and 4b). While on the Internet CDN providers rely on DNS tricks to select the CDN node that should serve a client [2], in PSI each CDN node can be announced only to the appropriate network domains. Note that the local RENE can also exploit off-path caching, without interfering with CDN operation.

MOBILITY

MOBILITY SUPPORT

Current trends indicate that mobile hosts will overtake fixed ones, not only in terms of numbers but also in terms of traffic load. PSI natively supports the seamless mobility of MNs, for it decouples request resolution from data transfer in both time and space. Regarding *time*, publishers and subscribers do not need to be simultaneously connected to the network: publications can be issued before subscriptions or vice versa and information transfer will commence when possible. Regarding *space*, subscribers can be served by any publisher, depending on their location, even for different chunks of the same item.

To understand the difference with Mobile IP, consider a MN that performs a handoff while receiving a large information item. With Mobile IP, if we are to avoid triangular routing, the MN must:

- Acquire an ephemeral IP address
- Inform its correspondent hosts of its new address
- Establish new transport connections with these hosts
- Resynchronize with its peers at the application level

For instance, in order to continue viewing a video after the handoff, the MN must ask the video server to start transmitting the video from the last frame received.

In contrast, in PSI the MN can simply reissue subscriptions to any data items it has not received before the handoff; the RENE can direct these subscriptions to any convenient location, which could be a nearby cache rather than the original publisher. In the video example, the MN will ask the RENE for the next video chunks; as these are individually identified, there is no need to resort to the application layer.

EXPLOITING CACHING

While PSI allows MNs to continue exchanging data after a handoff without involving the application layer, this process will be delayed if the requested data are not locally available. We can therefore enhance mobility support by leveraging the native caching options of PSI to ensure that data will be available in a timely manner.

For micro-mobility (i.e., mobility within the same access network), the local RENE can use off-path local caches to support MNs after handoffs. As described above, these caches receive copies of data destined to the MNs; these are registered in the local RENE. After a handoff, when the MN requests the data chunks that it has missed, the local RENE can direct these requests to the local cache. As the local cache keeps receiving data during the MN's disconnection, it can send them quickly to the MN. Caches can be assigned to MNs based on local network topology information and, possibly, by forecasting their future attachment positions.

For macro-mobility (i.e., mobility between access networks), instead of selecting a *single* local cache in the access network, the RENE can select a *set* of caches in neighboring access networks and make them subscribers to the same data as the MN. If the MN moves to a neighboring access network, its data requests will again be served by a nearby cache. While this is ideal for delay-sensitive applications, it imposes bandwidth and storage costs to multiple caches; therefore, it makes sense to only select some neighbors for caching so as to minimize the overall cost. The selection process can consider the probability that the MN will move to a neighboring access network, the storage costs, and the bandwidth/delay for obtaining data from a local cache or a remote publisher [10].

While caching has been proposed as a remedy for handoff induced losses in the context of Mobile IP, in PSI the fact that all information is individually identified allows local caches to be indistinguishable from the original publishers, without having any application awareness. On the Internet, multiple application level caches must be installed in each access network to achieve the same benefits.

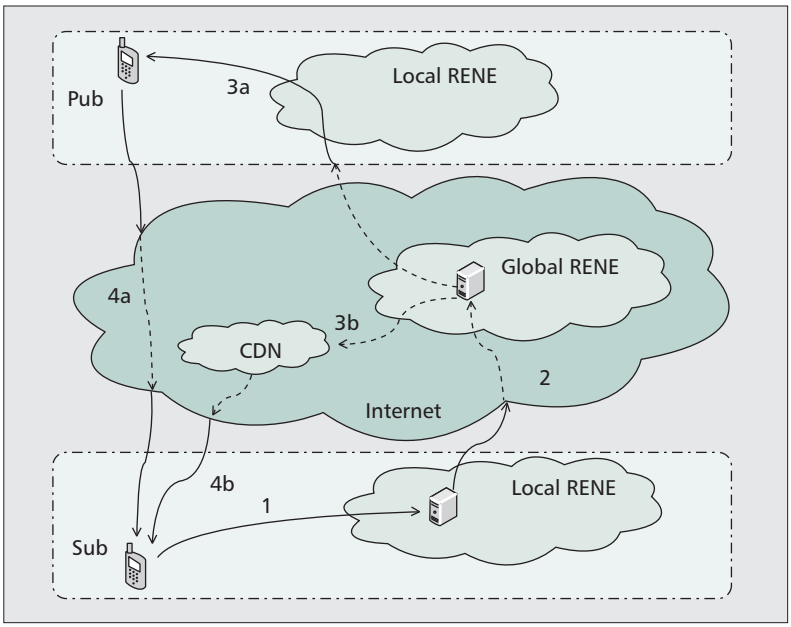

Figure 5. *Content replication.*

CONCLUSION AND STATUS REPORT

We have outlined the PSI architecture for an information-centric future Internet, emphasizing its support for network layer caching and mobility. PSI replaces the host-centric communication model of the current Internet with a publish-subscribe paradigm. Since information items are individually identified in PSI, they can be cached at the network layer. The asynchronous nature of PSI makes it mobile-ready, while caching further enhances support for the seamless mobility of MNs. This makes PSI a far more suitable platform than the current Internet for ubiquitous information delivery to billions of mobile devices.

PSI is a constantly evolving architecture: while the basic requirements are fixed [6], the design choices are being updated regularly [11]. The current implementation prototype of PSI [12] incorporates network bootstrapping, topolo-

The asynchronous nature of PSI makes it mobile-ready, while caching further enhances support for the seamless mobility of MNs. This makes PSI a far more suitable platform than the current Internet for ubiquitous information delivery to billions of mobile devices.

gy discovery, rendezvous, and forwarding for the intradomain case. The prototype is permanently deployed in a dedicated wide-area testbed and ephemerally deployed over PlanetLab. We are currently evaluating solutions for interdomain rendezvous, topology management and forwarding via simulations, analysis and prototype implementations. Finally, network layer caching combined with mobility support is at an advanced design and evaluation stage, while different transport protocols are currently at various design stages.

Acknowledgments

The PSI functional model, architecture and prototype implementation are the joint work of the PURSUIT and PSIRP project partners, which were funded by the EU FP7 program under contracts ICT-2010-257217 and ICT-2007-216173, respectively.

References

[1] D. D. Clark, "The Design Philosophy of the DARPA Internet Protocols," *Proc. ACM SIGCOMM*, Stanford, CA, 1988.
[2] P. Vixie, "What DNS is Not," *Commun. ACM*, vol. 52, no. 12, 2009, pp. 43–47.
[3] S. Seetharaman and M. Ammar, "Inter-Domain Policy Violations in Multi-Hop Overlay Routes: Analysis and mitigation," *Computer Networks*, vol. 53, no.1, 2009, pp. 60–80.
[4] V. Jacobson *et al.*, "Networking Named Content," *Proc. ACM CoNEXT*, Rome, Italy, 2009.
[5] N. Fotiou *et al.*, "Developing Information Networking Further: From PSIRP to PURSUIT," *Proc. 7th ICST Conf. Broadband Commun., Networks, and Systems (BROADNETS)*, Athens, Greece, 2010.
[6] D. Trossen, M. Särelä, and K. Sollins, "Arguments for an Information Centric Internetworking Architecture," *ACM Comp. Commun. Review*, vol. 40, no. 2, 2010, pp. 26–33.
[7] P. Jokela *et al.*, "LIPSIN: Line speed Publish/Subscribe Inter-Networking," *Proc. ACM SIGCOMM*, Barcelona, Spain, 2009.
[8] J. Rajahalme *et al.*, "On Name-based Inter-Domain Routing," *Computer Networks*, vol. 55, no. 4, 2011, pp. 975–86.
[9] K.V. Katsaros *et al.*, "On Inter-Domain Name Resolution for Information-Centric Networks," *Proc. IFIP Networking*, Prague, Czech Republic, 2012.
[10] V. Siris, X. Vasilakos, and G.C. Polyzos, "A Selective Neighbor Caching Approach for Supporting Mobility in Publish/Subscribe Networks," *Proc. 5th ERCIM Wksp. eMobility*, Vilanova i la Geltrú, Spain, 2011.
[11] D. Trossen, Ed., "PURSUIT D2.3; Architecture Definition, Components Descriptions and Requirements," www.fp7-pursuit.eu.
[12] The PURSUIT project, "Blackadder Prototype," https://github.com/fp7-pursuit/blackadder.

Biographies

George Xylomenos (xgeorge@aueb.gr) is an assistant professor of computer science at the Athens University of Economics and Business (AUEB) and a member of the Mobile Multimedia Laboratory. He received his Diploma in informatics (1993) from AUEB, and M.S. (1996) and Ph.D. (1999) degrees in computer science from the University of California, San Diego (UCSD). His research interests include information-centric network architectures and protocols, multicast-based content distribution, and the provision of quality of service over wireless and mobile networks.

Xenofon Vasilakos (xvas@aueb.gr) is a Ph.D. student at AUEB and a member of the Mobile Multimedia Laboratory. He obtained his Diploma in informatics from AUEB in 2007 and his M.Sc. degree in parallel and distributed systems in 2009 from the Vrije Universiteit of Amsterdam. His current research interests include future Internet architecture and protocols, and information-centric networking with an emphasis on rendezvous networks and mobility support.

Christos Tsilopoulos (tsilochr@aueb.gr) is a Ph.D. student at AUEB and a member of the Mobile Multimedia Laboratory. He received his Diploma in informatics from AUEB (2006) and M.Sc. in communication systems and networks from the National and Kapodistrian University of Athens (2009). In his Ph.D. studies he is conducting research in the area of information-centric network architectures and protocols. His research interests include distributed systems and software engineering.

Vasilios A. Siris (vsiris@aueb.gr) is an assistant professor of computer science at AUEB. He received his degree in physics (1990) from the National and Kapodistrian University of Athens, Greece, his M.S. (1992) in computer science from Northeastern University, Boston, and his Ph.D. (1998) in computer science from the University of Crete, Greece. His research interests include resource management in wired/wireless networks and future Internet architectures.

George C. Polyzos (polyzos@aueb.gr) is a professor of computer science at AUEB where he founded and is currently heading the Mobile Multimedia Laboratory. He was previously CSE professor at UCSD, Computer Systems Laboratory co-director, Center for Wireless Communications Steering Committee member, and San Diego Supercomputer Center Senior Fellow. He received his EE Diploma from the National Technical University of Athens (1982), and M.A.Sc. (EE, 1985) and Ph.D. (CS, 1989) from the University of Toronto. His current research interests include Internet architecture and protocols, mobile multimedia communications, wireless networks, ubiquitous computing, and network security.