Attachment to a Native Publish/Subscribe Network

Jimmy Kjällman
Ericsson Research, Finland
Email: jimmy.kjallman@ericsson.com

***Abstract*—**
In this paper we examine how network attachment can be handled in a system where all communication, both on a data link and in a network, is based on the publish/subscribe paradigm instead of the traditional send/receive model. We present and discuss an early clean-slate solution to the problem of establishing and maintaining network connectivity between nodes in a secure and efficient way. The solution includes a basic protocol for pub/sub-based attachment, which addresses certain pertinent security challenges and conforms with the principle of receiver-driven communication. In addition, we report initial experiences from implementing the concepts outlined in our protocol design.

I. Introduction

In order to communicate in a network, a node must be attached to at least one other node that can provide access to further nodes. We use the term *network attachment* of the procedure, and the related function, that deals with setting up, maintaining, and renewing this kind of state, and thus enables network connectivity for a node. Operations handled in this procedure include discovery of potential attachment points, authorization of joining nodes to use services in the network, as well as configuration of entities involved in attachment.

In currently prevalent networks, data link layer technologies are used for network access, and on top of those, the Internet Protocol (IP) handles network layer connectivity. On each layer in these existing systems, communication is usually bootstrapped using mechanisms that are based on the traditional communication model with *send* and *receive* primitives. In this work, however, we investigate network attachment in a novel networking architecture where protocols in the network layer, and partially also in the data link layer, have been designed according to the *publish/subscribe* (pub/sub) paradigm [1].

A major difference between the send/receive and pub/sub models is that in the former paradigm, senders can by default get any data sent to any destination on the link or in the network, while in an ideal pub/sub system, a published piece of data reaches only recipients that have explicitly requested it by subscribing to it. A consequence of the send/receive model is that senders can easily and with a very low cost generate network traffic that is unwanted by receivers, and which can be used for malicious purposes, such as creating denial-of-service (DoS) attacks. This is the situation currently in the Internet as well. But by designing a network that is natively based on pub/sub, it could be possible to avoid such problems [2]. Another important benefit of pub/sub is also that it can be used for enabling information-centric networking.

The Publish-Subscribe Internet Routing Paradigm (PSIRP) EU FP7 is a project that aims at building this kind of a complete internetworking architecture based on using pure pub/sub and principles that mandate information-centric and receiver-driven design [3]. In this system, most protocols and components found in current IP-based networks have to be completely redesigned. One of these components is, of course, the network attachment function that we investigate in the following sections.

This paper is structured as follows. In Section II, we briefly present current attachment mechanisms. In Section III, we describe in more detail the problem field, and specify the scope of the design and implementation which we will present and discuss in Sections IV and V. Finally, we conclude our results and discuss potential future work items in Section VI.

II. Related Work

Existing network attachment mechanisms can be found in current IP-based networks and in link layer access technologies. Especially the latter ones have a major role in performing authentication of nodes and authorizing them to use networks through attachment points. Examples of such systems are IEEE 802.11 [4] with EAP-based 802.1X [5] authentication, 3G/UMTS networks where AKA is used (e.g. [6]), and mechanisms based on PPP [7] together with CHAP [8].

For example, if we take a closer look at attachment procedures in IEEE 802.11, especially in the infrastructure mode, we see that a node seeking access first scans for access points, and thus learns basic information about them. Then, it initiates attachment by sending requests for authentication and association to a chosen access point. The latter phase implies configuring the access point to send and receive frames in its local network on behalf of the wireless station attached to it. After this, stronger authentication of the joining node, as well as an exchange of cryptographic keys, can be performed. The authentication phase may involve an AAA backend that decides if data traffic between the joining node and the network is allowed. Notably, control traffic between the joining node and the AAA system can be securely tunneled (e.g. RADIUS over EAP-TTLS).

As a result of the above procedure, a node gets limited network access on the link layer. The next step is usually then to increase communication scope to other networks, such as the Internet. In IP hosts, a minimum requirement for network layer communication is to have an IP address for a network interface, but also other addresses and parameters are usually needed. In IPv4 networks this information can be

acquired by using DHCP [9]. In IPv6, on the other hand, router advertisements [10] can provide most configuration data. A related issue is translation of network layer addresses to link layer addresses in order to send data to the next hop. The Address Resolution Protocol (ARP) [11] and IPv6 neighbor discovery are normally used for dynamically coupling the two layers with each other.

An alternative to employing separate attachment protocols in each layer is to utilize a *cross-layer* design. This approach, presented in [12], and further explored in a number of other papers (e.g. [13]–[15]), has some advantages compared to using separate mechanisms on each layer. Firstly, it can be used to reduce the number of roundtrips in an attachment procedure by allowing several operations to run in parallel within the same messages. Delegation of some tasks from a client to an access node can also be used for achieving better efficiency. Secondly, security is enhanced if, for instance, authentication of an entity providing configuration information covers both the link and network layer instead of being handled by two unrelated mechanisms. Moreover, cryptographic puzzles can be used for coping with DoS, and a Diffie-Hellman key exchange in the beginning of a negotiation yields keys that can be used for confidentiality and integrity protection. Notably, these protocols can be used as frameworks for existing protocols, such as DHCP, AAA protocols, MobileIP, and MLD.

We can also note that in pub/sub overlay networks, the attachment issue has been discussed in relation to mobility of nodes. In essence, this means that subscription and publication states need to be updated when a node changes its point of attachment. This problem field has been examined for example in [16]–[18].

III. Defining the Problem

In the following sections we will explore how network attachment can be handled in a novel networking architecture. The basic building blocks used in the design are the publish and subscribe primitives, in addition to a simple link layer mechanism that provides the capability to send and receive messages over a physical medium.

The goal of network attachment is to make it possible for nodes to use publish and subscribe functions on data in a network through attachment points. Hence, the focus of this work is on how to use the pub/sub communication model between nodes on the same data link, as well as internally in a node, in designing a network attachment protocol that can be used to fulfill that goal. Data communication in a network once attachment is done is out scope for this paper, as are all details related to the physical, MAC, and LLC layers.

Next, we identify a few basic attachment scenarios and operations involved in a typical attachment procedure. In addition, we describe what security features and other properties we consider important in this context, and conclude this discussion with a problem statement.

A. Basic Attachment Scenarios

In the simplest case, only two nodes are involved in network attachment, with one node providing the other one access to a network. A situation where a single node, for example a laptop computer, accesses a local network or the Internet through a dedicated access node, such as a wireless access point, can be described as an *asymmetric* attachment scenario. A different kind of setting is, for instance, an ad hoc network. There, nodes form networks that can be joined together at attachment, or split as nodes detach. This model we call a *symmetric* scenario. Furthermore, an attachment procedure can occur over either a wired or wireless link. The former ones are often of a switched type, while the latter ones are broadcast media.

For simplicity, we will here focus on the asymmetric scenario and also put more weight on the broadcast case, although our concepts could be usable in other settings as well. We will also concentrate on communication between two nodes attaching directly to each other, and leave external entities, such as nodes belonging to authentication, out of our scope.

We should also note that in the situation when a node changes from one attachment point to another, we face the problem of mobility. However, this is another issue which is out of scope for this paper.

B. Operations in Attachment

The following operations are commonly found in attachment procedures, and therefore they are also taken into account in this work.

- **Discovery and selection of attachment points.** Especially in wireless scenarios, a node needs to find other nodes that can be used for network access, and learn some basic configuration data and other useful information about them.
- **Authentication and authorization.** Authorization to use a network can be based on verifying the identity of a node, or on negotiations about a contract defining services and compensation for them.
- **Configuration.** A central operation in network attachment is to exchange information that is used for setting up entities in order to enable communication. This information includes both communication parameters and data related to setting up security associations.

C. Security and Other Properties

The network attachment mechanism should also provide some degree of protection against security threats that are relevant in this context. Since preventing DoS attacks is a major argument for choosing the pub/sub model in the first place, this issue must also be considered in the attachment design.

Another significant threat is the possibility of receiving malicious configuration data from unauthentic messages. However, we have made the assumption that *opportunistic* security is employed by default. In other words, a node should be able to communicate with any attachment points even without being able to initially verify their identities.

In addition, we want our mechanism to allow autoconfiguration of nodes, and the design should to some extent address demands for resilience, efficiency, and extensibility.

D. Problem Statement

In conclusion, we focus on designing a network attachment protocol using the pub/sub communication model between two nodes, namely an access node and a client, directly over a broadcast data link. We also want to convey essential configuration data, as well as information needed for authorization, as part of the attachment process. Moreover, we explore some submechanisms that aim at making the design more resistant to denial-of-service and message forgery attacks.

IV. Design

A. Communication Model and Architecture

To begin with, we outline the publish/subscribe based communication model and architecture which we utilize when designing the network attachment function. This architecture is similar to the one found in [2].

In our pub/sub model, single pieces of data, as well as channels through which more than one data item can be conveyed, are represented by publications tagged with identifiers (IDs). An entity, for example an application, can thus subscribe to an ID corresponding to the data it wants. If another entity has published, or will publish, data with that identifier, it is delivered to the subscriber.

In addition, the identifier space is expected to be large enough and that IDs can be generated randomly. These IDs can be carried as labels in the headers of packets sent on links.

Another assumption is that each node participating networking has a mechanism for dispatching publications both between node-internal entities, as well as interfaces to external entities. The former group includes, for instance, application processes, while send and receive functions provided by network interface cards belong to the latter group. This mechanism can store publications and subscriptions, and push publications to subscribers. In other words, it represents a *blackboard* design pattern. For example, it can be implemented in a node's operating system kernel, as a user space application, or partially even inside a network interface card.

Further on, we assume that this kind of a system is used for delivering messages between two network attachment protocol machines located in two nodes on the same link.

B. Bootstrapping Communication

Next we discuss how a control channel for a network attachment procedure can be set up, starting from the very first protocol message.

1) Delivering the first message: The first issue to be handled in a network attachment procedure is bootstrapping communication between two nodes. In our case, it means that a node needs to be able to publish data subscribed by another one, thus making it possible to deliver the first message.

An intuitive approach to bootstrapping is that nodes providing access to others initially need to reveal some information about themselves to clients. This information can include basic configuration parameters, a description about services provided and policies mandated by the access node, and the identity of the access node or network behind it. In addition, it should contain an identifier that can be used by clients for initiating an attachment procedure with the access node.

Two different approaches for conveying this information to clients can be identified:

1) An access node publishes information about itself, but does not spontaneously send it out on the link. Instead, it subscribes to subscription messages published by clients. Clients seeking attachment points publish and broadcast explicit subscriptions, tagged with an ID known to access nodes, who then send their information in response.
2) An access node publishes information about itself with a well-known or preshared identifier, and continuously broadcasts it onto a link with certain intervals. Clients seeking attachment points subscribe to this identifier internally and receive the data from the link.

The first approach is similar to sending probe requests and responses in IEEE 802.11, or to router solicitations in IPv6. The second method is, in turn, analogous to sending 802.11 beacons and unsolicited IPv6 router advertisements.

2) Dealing with denial-of-service: The first method above, which is based on explicit subscriptions, has the benefit that a node can acquire the information it needs immediately, without having to wait for the next spontaneous broadcast from an access node. Compared to the second solution, it also implies that less traffic is sent on a link when no nodes need information from the access node. However, on the downside, it requires that an access node has a subscription to an identifier all the time, which could be utilized in denial-of-service attacks. Hence, additional protection, such as limiting the amount of resources an access node uses for serving subscribers of the initial data, might be needed.

In contrast, the second approach addresses the DoS problem mentioned above by allowing clients to subscribe to network information messages only when they have a need to find new attachment points. Unfortunately, this still does not completely remove the need for a common predistributed or global ID to bootstrap communication.

The temporary channel ID to which multiple nodes can send attachment initiation messages can also open up another possibility for DoS. Clients could make an access node create a lot of state by publishing a large number of initiation messages using that ID. In this case, one possible countermeasure could be to include a cryptographic puzzle in the first message sent by an access node. The conceptual idea here is that a client should invest some computational resources by solving the puzzle before being able to even learn what the identifier subscribed by the access node is. In this way, the access node should be able to know that a client is committed to attachment when it receives an initiation message and creates state specific to the client. In existing protocol designs, puzzles are used for example in [19] and [14].

However, since a puzzle introduces a delay in the attachment procedure, and since different devices have different computing abilities [13], the use of puzzles may need to be limited. For example, if an access node is capable of serving new clients without problems, puzzles might not be needed at all.

But if it detects that an attack might be going on, puzzles (or other countermeasures) can be introduced and their difficulty can be increased. At the same time, IDs used for initiation also need to be updated frequently. On the other hand, this could in turn be used by attackers to increase resource usage in both access nodes and clients.

Nevertheless, if puzzles are employed, a client may choose to solve a puzzle after making a decision to initiate attachment with a particular access node. Alternatively, it can compute solutions opportunistically for faster attachment. Again, the resources used for that should be limited, because otherwise a malicious access node could send out very difficult or unsolvable puzzles and make a client do unnecessary work. An additional issue here is also that puzzle solutions and initiation IDs can expire, so clients should ensure that they have fresh enough data available when they decide to join a network.

C. The Attachment Procedure

1) Initiating attachment: So far we have discussed how a client gets the first message from an access node. In fact, a single message could in some cases carry all relevant configuration information a node attaching to a network needs to know in order to be able to publish and subscribe data in a network. If attachment to a network does not require authentication, compensation contract negotiations, key exchanges, or other operations that take place over several messages, only this first step would be required. However, we assume that in many scenarios an exchange that comprises several messages need to be supported, and therefore we must also define how more messages can be delivered between two nodes.

From the first message published by the network attachment function in an access node, a client can learn a channel identifier that can be used for contacting that attachment point. Next, the client should publish a message with that identifier in order to initiate attachment.

This initiation message can contain a request to become authorized to use network services and to get configuration parameters from the network, as well as information about the node's identity, timestamps, proof that a puzzle was solved by the joining node, etc. In addition, it contains a subscription to a return identifier. Correspondingly, the access node that receives this message sends its own return ID in its first response.

2) Subscribing to return identifiers: As a conclusion, return IDs are required for creating two-way communication channels, since network attachment functions need to know what IDs have been subscribed by their counterparts. Consequently, a couple of design alternatives can be identified:

1) Return identifiers can be static channel IDs, i.e., each protocol message in one direction has the same ID.
2) Return identifiers can be different in each message, and the next ID a node is willing to receive is explicitly included in every message.
3) Return identifiers can be channel IDs that are computed algorithmically and change as a function of time.
4) Return identifiers can be different in each message, and the next IDs are computed algorithmically.

Option 1 above is the most simple solution, but using a static identifier to which data can be sent by any node at any time can again cause DoS problems. On a broadcast link, the channel ID is very easy to learn through passive eavesdropping, even if it is originally sent encrypted from one endpoint to another. Then, it could be used by other nodes to send traffic directly to the subscriber of that ID.

Another simple approach is found in option 2. In this case, each published message includes a return identifier that the publisher subscribes to, and that then the publisher of the next message could use. While this would mean that a node would have to store only one ID that its counterpart has subscribed to, it would also introduce an overhead in every message. This mode also assumes that nodes take turns in sending messages, which is not always true, especially in the case of spontaneous data request and updates that might take place after running the initial attachment exchange.

In option 3, proposal 1 has been modified so that a channel ID is valid only for a limited period of time. In that case, a temporary ID could still be eavesdropped and possibly used for malicious purposes, but during the lifetime of a long-lived and low-traffic control channel, probably most of the time IDs learned in this way are invalid. A requirement for using this approach is that publishers and subscribers are sufficiently synchronized. They must also update their subscriptions even when there is no traffic on the control channel.

Finally, option 4 is a modification of proposal 2. Here, each message has its own, unique identifier, but these IDs are computed algorithmically when needed.

Notably, the latter two proposals utilize the concept of *algorithmically generated identifiers*. In order to bootstrap such an ID sequence, an algorithm and some parameters to it, such as seeds for random number generators or keys for cryptographic functions, are included in the payload of an initial message instead of an explicit identifier.

In previous work, algorithmic pseudo-random identifiers have been used for instance for protecting the privacy of users in network communication [20]. In that context, pseudo-random IDs should be possible to produce and tie to each other only by a limited set of entities. We use similar techniques here for creating unique IDs for messages, and to achieve some level of protection against denial-of-service. The latter goal is reached if only a small number of nodes, preferably only the ones participating in the attachment procedure, know the exact identifier sequence. This implies that the message in which the data needed for computing the sequence should be encrypted with a secret shared by only those two nodes. Alternatively, in some situations we could just accept that some nodes present in a broadcast medium when that message is exchanged might know that data, but those who have appeared later do not.

3) Using the data exchange framework: We have now described how an access node and a client can establish a two-way channel for communication between each other. This channel can be used for exchange of information in a stateful configuration procedure. In order to achieve extensibility, the framework should allow different operations to put their data

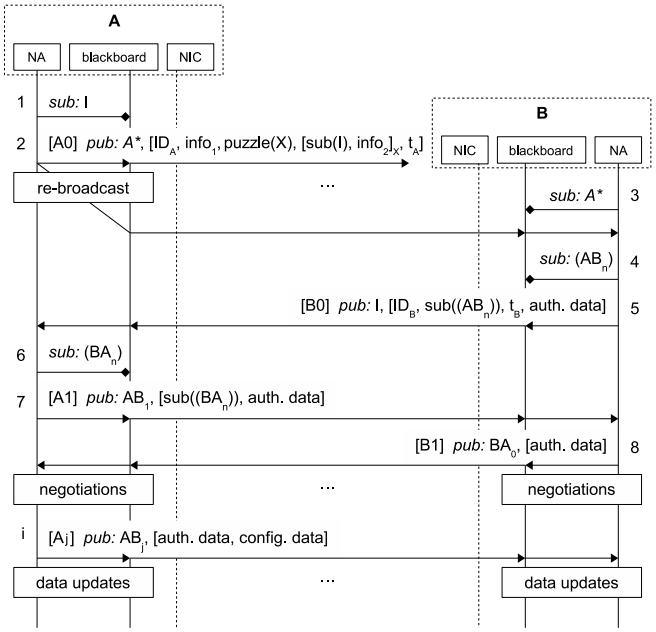

Fig. 1. Network attachment message exchange example.

in custom fields in attachment protocol messages.

One particular function that can be part of an attachment procedure is authorizing a node, and a user behind that node, to use services in the network. A common way to do this is through authentication, that is, verification of the node's (or user's) identity. Another possibility is to allow previously unknown entities to negotiate contracts about services and compensation for them. A contract can then be referred to later, when publishing and subscribing data in the network.

Another operation that we also have already mentioned is providing configuration information to a client. In the PSIRP architecture [3], identifiers used by the *rendezvous* and *forwarding* functions are examples of that kind of data.

In conclusion, an initial message exchange can be performed over the control channel in order to authorize a node for access and provide it with configuration data. Later on, this data can be updated if both the access node and the client maintain state for the channel.

D. A Sample Message Exchange

Figure 1 shows an example of a network attachment message exchange between two nodes. In the figure, *A* represents an access node and *B* a client. Here, the selected bootstrapping scheme is based on periodic broadcasting, and the return ID subscriptions use algorithmically generated sequences. Below we explain what operations take place in this procedure.

In step 1, the network attachment (NA) function in node *A* subscribes internally to identifier *I* used for attachment initiation. In step 2, *A* publishes an advertisement message (*A0*) with identifier *A**. If needed, some of the information in that message, especially the subscription to *I*, can be encrypted using data (*X*) protected by a cryptographic puzzle. Message *A0* is then periodically broadcasted on the link, possibly with updated data (e.g. timestamps, puzzle). This also implies that *A*'s blackboard has been configured so that network interfaces (NIC) subscribe to messages tagged with *A**.

Step 3 shows node *B*'s NA function subscribing to identifier *A**, which can occur for instance when it learns that a link has become available. Based on the information received from node A, *B* can then choose to start an attachment negotiation with A. Subsequently, *B* subscribes to a return ID sequence (AB_n) in step 4, and publishes an initiation message (*B0*) using identifier *I* in step 5. In the message payload, the subscription to (AB_n) is indicated, as well as data that *A* can use for making decisions about authorization.

When node *A* receives the message *B0*, it can create state for the new node, i.e., B. Similarly to B, *A* subscribes to a return identifier sequence (BA_n) in step 6. In step 7, *A* publishes a message (A1) tagged with AB_0, i.e. the return identifier currently subscribed by *B*. In this message, the *A* includes its own return identifier(s), as well as other data that is part of the initial negotiation.

Now *B* can send the following message (*B1*) in the message exchange using identifier (BA_0) (step 8). This initial attachment procedure can then continue in a number of protocol messages, depending on what operations take place.

Finally, when the initial exchange is complete, *A* can send configuration data, as well as, e.g., a service and compensation contract to *B* (message *Aj* in step i). After that, the established channel may remain and be used for data updates.

E. Fault Tolerance and Security

What is still missing from the above model is fault tolerance and additional security features. Firstly, in order to cope with packet loss, message retransmissions are needed. For instance, when publishing a attachment protocol message, a timer can be set, and if there is no reply to it even though an operation expects one, the same message is sent out again.

Secondly, for additional security, one option could be to employ asymmetric cryptography. A Diffie-Hellman key exchange could be performed early in the protocol, with appropriate signatures [21]. The benefit of this would be that encryption and message authentication could be easily handled using the shared secret provided by the exchange. By encrypting the payload in protocol messages, for example configuration parameters and subscription data could be protected from unauthorized parties. And with integrity protection of messages, malicious nodes would not be able to inject their own information into the control channel.

Using public key cryptography is, however, computationally quite expensive. Alternative protection methods might therefore be needed. For example, preshared keys could be utilized similarly as in many existing systems. On the other hand, if we want to avoid off-band data exchanges, and can do with only ensuring that our counterpart remains the same during communication, hash chain based approaches for message authentication present an attractive option. Such methods include for example TESLA [22] and its variations. The problem with those is that they imply delayed authentication, that is, a hash

chain value used for creating a message authentication code for some messages is revealed in a later message. The state of a network attachment function still should not be changed using information from messages not yet authenticated. This can be particularly problematic in a setting where messages are exchanged in a request-response fashion, because then messages buffering is not an option.

V. Implementation

In order to test the concepts described in previous sections, a network attachment daemon has been implemented in Python. In that implementation, link layer communication is based on broadcasting Ethernet frames through raw sockets. Messages are conveyed between applications and link interfaces via an internal blackboard that can store subscriptions, cache publications, and dispatch data to registered entities. That dispatching system is implemented as a hash table that maps identifiers to function pointers. These simple mechanisms are then used by a network attachment protocol implementation that follows the message sequence in Fig. 1. ID sequences are in this implementation created by hashing a concatenation of a shared key and a previous ID.

Currently the performance and architecture of this prototype are suboptimal. Nonetheless, the experiences from developing and testing show that the concepts presented in Section IV are can indeed be realized in actual mechanisms. Compared to existing protocols, the amount of data that the exchange require seems to be relatively high, much due to the use of long identifiers. On the other hand, the number of roundtrips can indeed be reduced significantly by performing attachment operations, such as authentication and parameter configuration, inside a single initial message exchange instead of doing them in separate steps on each layer. The exact message counts still depends on the operations that actually takes place over the message exchange framework, which currently makes a comparison to existing systems difficult.

VI. Conclusion and Future Work

In this paper we presented a novel design for network attachment using the publish/subscribe paradigm. We identified a set of operations that usually need to be handled in an attachment procedure, and created a message exchange framework over which those operations can be performed.

Some challenges were encountered that are related to performing node-to-node communication using principles for information networking and receiver-driven design. Other issues included how to prevent opportunities for denial-of-service, as well as how to deal with other aspects of security. This lead to discussions about alternatives for solutions to these issues. A further analysis on these topics is provided in [23].

Because the designed mechanism was designed to be used in conjunction with a complete pub/sub-based networking architecture, such as PSIRP, future work in this area consists of aligning the design with the PSIRP [3] architecture, and integrating the current implementation with the lower layer PSIRP prototype [24]. This means defining interfaces to other components, and using actual PSIRP identifiers instead of just generic labels. In addition, issues including security, mobility, ad hoc networking, compensation-based authorization mechanisms, and implementation performance will be addressed more thoroughly.

Acknowledgment

Thanks to Sasu Tarkoma, Jukka Ylitalo, Christian Esteve, Teemu Rinta-aho, Pekka Nikander, people in the PSIRP project, and others who provided support during this work. This work was supported by the EU FP7 PSIRP project under grant INFSO-ICT-216173.

References

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

[2] M. Särelä, T. Rinta-aho, and S. Tarkoma, "RTFM: Publish/subscribe internetworking architecture," in *ICT-MobileSummit 2008*, Sweden, 2008.

[3] M. Ain *et al.*, "PSIRP deliverable D2.2: Conceptual architecture of PSIRP including subcomponent descriptions, version 1.1," 2008.

[4] "IEEE Std 802.11-2007: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," 2007.

[5] "IEEE Std 802.1X-2004: Port-based network access control," 2004.

[6] "3GPP TS 33.102 V8.0.0, security architecture (release 8)," 2008.

[7] W. Simpson, "The point-to-point protocol (PPP)," RFC 1661, 1994.

[8] W. Simpson, "PPP challenge handshake authentication protocol (CHAP)," RFC 1994, 1996.

[9] R. Droms, "Dynamic host configuration protocol," RFC 2131, 1997.

[10] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor discovery for IP version 6 (IPv6)," RFC 4861, 2007.

[11] D. Plummer, "Ethernet address resolution protocol: Or converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware," RFC 826, 1982.

[12] J. Arkko, P. Eronen, P. Nikander, and V. Torvinen, "Secure and efficient network access," in *DIMACS Workshop on Mobile and Wireless Security*, USA, 2004, (Extended abstract).

[13] J. Arkko, P. Eronen, H. Tschofenig, S. Heikkinen, and A. Prasad, "Quick NAP - secure and efficient network access protocol," in *ASWN 2006: 6th International Workshop on Applications and Services in Wireless Networks*, Germany, 2006.

[14] S. Heikkinen, M. Priestley, J. Arkko, P. Eronen, and H. Tschofenig, "Securing network attachment and compensation," in *WWRF#15: Wireless World Research Forum Meeting*, France, 2005.

[15] U. Meyer, *Ambient Networks Phase 2, Technical Annex to D3-G.1: Design of Composition Framework*. Ambient Networks, 2006, ch. Report 4: SNAP: A symmetric version of QNAP.

[16] L. Fiege, F. C. Gärtner, O. Kasten, and A. Zeidler, "Supporting mobility in content-based publish/subscribe middleware," in *ACM/IFIP/USENIX International Middleware Conference*, Brazil, 2003.

[17] V. Muthusamy, M. Petrovic, D. Gao, and H.-A. Jacobsen, "Publisher mobility in distributed publish/subscribe systems," in *4th International Workshop on Distributed Event-Based Systems (DEBS'05)*, USA, 2005.

[18] S. Tarkoma and J. Kangasharju, "On the cost and safety of handoffs in content-based routing systems," *Computer Networks*, vol. 51, no. 6, pp. 1459–1482, 2007.

[19] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host identity protocol," RFC 5201, 2008.

[20] J. Arkko, P. Nikander, and M. Naslund, "Enhancing privacy with shared pseudo random sequences," in *13th International Workshop on Security Protocols*, UK, 2005.

[21] H. Krawczyk, "SIGMA: the 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols," in *23rd Annual International Cryptology Conference*, USA, 2003.

[22] A. Perrig, R. Canetti, J. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, no. 2, pp. 2–13, 2002.

[23] J. Kjällman, "Network attachment to a publish-subscribe link," Master's thesis, Helsinki University of Technology, 2009.

[24] P. Jokela *et al.*, "PSIRP deliverable D3.2: Implementation plan based on conceptual architecture," 2008.