A Service Oriented Architecture for Multi-domain Mobility

Inge Grønbæk, Teemu Rinta-aho, Tony Jokikyyny.

Abstract— This work applies the design principles advocated in recent publications in an attempt to compile the core of a new Internet architecture. This allows conflict of interest to serve as a driving force for innovation. The cornerstone of the architectural framework is a well designed naming and addressing scheme. Some of the most promising recent proposals in the area of naming and addressing are combined creating a high-level baseline architecture that may serve as a core of the new Internet. An important feature of this architectural framework is the functional layering with a proposed communication substrate API. This API shields the application developer from the complexity involved in handling multi-domain mobility in a QoS controlled environment. It integrates multi-point (e.g. multicast) communications for mobile and stationary endpoints.

It is a working hypothesis of the EU Ambient Networks Project that this coherent framework can be combined with replaceable functional components, at each communications layer, providing a vehicle for handling specific functional requirements and future innovation.

This tentative baseline architecture is proposed by members of the Ambient Networks (AN) project for successive amendment to meet all the AN requirements.

***Index Terms*—Addressing, Architecture, Mobility, Domain.**

I. Introduction

Ambient networking is geared towards increasing competition and cooperation in an environment populated by a multitude of user devices, wireless technologies, network operators and business entities. This architecture aims to extend all IP networks with three fundamental requirements of today's networking world. These requirements include dynamic network composition, mobility and heterogeneity. By encompassing these notions the Ambient Networks (AN) project [1] strives to achieve horizontally structured mobile systems that offer common control functions to a wide range of different applications and air interface technologies. Minimal assumptions are made about user devices in order to arrive at a solution that can operate on terminals with differing capabilities. This leads to a multi-domain environment where none of the existing mobility solutions is sufficient. Additionally, the Internet has moved from a research network to an arena dominated by commercial stakeholders [2]. This has introduced strong and often conflicting commercial interests. These conflicts have severely restricted the technical development of the Internet and partly led to its ossification. Examples include accommodation of well-established requirements like QoS control, native mobility management and multicast. We may experience a gradual migration of new functionality into the next generation of Internet routers. This will cater for efficiency and scalability. However, the evolving network need some fundamental enhancements, e.g. in the area of naming and addressing, together with APIs to access services and network resources. Standardization of such APIs and the corresponding interfaces between different network domains is the key to innovation, allowing new services and technologies to be introduced stepwise without severely impacting existing services and operating domains. The new architecture has to support mobile applications over a multiplicity of domains, including domains for: addressing, administration, mobility, naming, routing, and security.

In the following we start by describing the major architectural principles and requirements that should govern the development of the new Internet architecture, including the layering principles and APIs. Then we describe the proposed naming and addressing framework as a key foundation for all the architectural components. Lastly we briefly discuss some of the required components of the multi-domain service oriented architecture including security, mobility management, QoS control and multicast.

Manuscript received January 20, 2005. This document has been produced in the context of the Ambient Networks Project. The Ambient Networks Project is part of the European Community's Sixth Framework Program for research and is as such funded by the European Commission.

Inge Grønbæk is with Telenor R&D, Snarøyvn. 30, 1331 Fornebu, Norway. (phone: +47 91881777; fax: +47 91881777; e-mail: inge.gronbak@telenor.com).

Teemu Rinta-aho is with Ericsson Research, 02420 Jorvas, Finland. (phone: +358 9 299 3078; fax +358 9 299 3535; e-mail: teemu.rinta-aho@nomadiclab.com).

Tony Jokikyyny is with with Ericsson Research, 02420 Jorvas, Finland. (phone: +358 9 299 3315; fax +358 9 299 3535; e-mail: tony.jokikyyny@nomadiclab.com).

II. Architectural principles and requirements

Innovation with new applications and services has driven the growth of the Internet and the generation of new value. This implies that barriers to new applications are much more destructive than network-centric support for high use applications. Network-centric functionality may even provide valuable building blocks for new services and applications. The requirements that cannot easily be catered for within the existing architecture suggest new design principles that can accommodate the commercial tension while still providing an innovative and competitive environment. The following design principles proposed in [2] are adopted for this purpose:

- Design for variation in outcome, so that the outcome can be different in different places, and the resolution of conflict takes place within the design, not by distorting or violating it.
- Do not design so as to dictate the outcome. Rigid designs will be broken; designs that permit variation will flex under pressure and survive.
- Modularize the design along boundaries of conflict, so

that one tussle does not spill over and distort unrelated issues.

- Design for choice, to permit the different players to express their preferences.

Choice is closely tied to open interfaces. Open interfaces play a critical role in the evolution of networks, by allowing for competition among algorithms, implementations, and vendors, and by enabling rapid technical progress through replacement of modular parts rather than entire systems. Conflict of interests is also focused at interfaces. Exposure of cost of choice at such interfaces is among the important design criteria.

The principles above have to be applied in combination with functional requirements in order to derive a service-oriented architecture with flexibility and room for competition. These requirements ask for scalability in the required functionality as indicated for the Layered IP service model in Fig. 1 and Fig. 2.

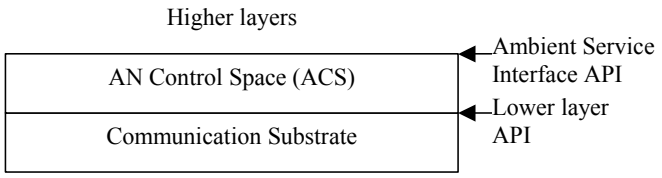

Fig. 1 Network sub-layering.

A. *Services, layering and APIs*

A key architectural goal is to support a rich set of services and building blocks (e.g. a toolbox) for application writers. This set needs to be extensible and flexible to stimulate innovation. The service layering model depicted in Fig. 1 facilitate the definition of a suitable Ambient Network functional layering, and Fig. 2 shows grouping of functions into the two layers in a hierarchy supporting applications. The rationale for proposing such a two-layer model is purely pragmatic, and at some stage the layers may merge.

The extent of interaction or integration with IP-layer routing requires careful consideration, and the working hypothesis proposed here is one sub-layer focusing on end-to-end aspects (ACS) and one path oriented sub-layer (the *communication substrate*) (Fig. 1).

Fig. 2 expands on the sub-layering shown in Fig. 1, suggesting architecture for service layering allowing applications and call/session control to control the IP-bearer and its service levels via a standardized Ambient Service Interface API.

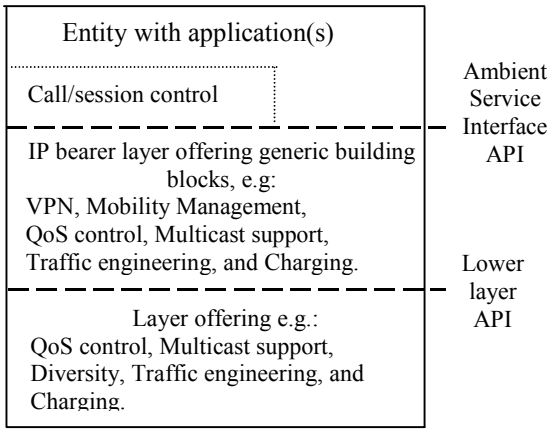

Fig. 2 Layered IP service model.

Some of the functionality may only be provided by one of the sub-layers as a service to the upper layer, while some functions may involve both sub-layers as the upper layer may enhance the functionality provided by the lower layer. Functionality may be passed transparently to the layer(s) above.

The lower sub-layer may, depending on the technologies involved, include the following service functionality that is offered to and used by the IP bearer layer:

- Connection establishment and release, with control of:
  - Reliability/Robustness
  - Bandwidth capacity and capacity modification (when available)
  - Charging information

An example technology for application at the lower layer is MPLS.

The model separates the higher layer service oriented functionality from the basic forwarding at the *communication substrate*.

B. *Entities and Sessions*

Host-to-host communication is replaced conceptually by communication between pairs of *entities* via logical associations called sessions. Packet exchange is handled via the *communication substrate* taking care of Mobility Management and QoS control. An *entity*, as defined in [3], [4] and visualized in Fig. 3, is the generalization of an application that is an endpoint of communication. An *entity* contains states for both application and communication, and it is the smallest unit that can be mobile. A moving *entity* is a unit, carrying its application state and communication state including the endpoints of its sessions. *Entities* operate in interaction with lower-level supporting systems, including operating systems and the Ambient Network Resource Interface. Transport protocols exchange data between two *entities* (Fig. 3), and the topological location of the *entity* is irrelevant to the basic semantics of transport.

The *communication substrate* delivers data on behalf of *entities* and their sessions. When an *entity* wants to send a packet for one of its sessions, it uses the DATA primitive of the API (see Formula 2) to hand the packet to the Ambient Service Interface with parameters defining the destination (*locator*) and the path. The destination *locator* contains the information needed to cause delivery of the packet to the desired destination *entity*. The destination *entity* is dynamically attached to a *slot* at its current point of attachment. The destination locator references this *slot*. A particular management mechanism may rewrite the *locator* en route (e.g. when transiting different addressing or mobility domains). The delivery is all the way via the *communication substrate* to the *slot* where the *entity* is attached, and then to the identified session endpoint.

Mobility management for active sessions is handled below the Ambient Service Interface involving a function called *locator* management. The *locator* management effectively hides the mobility management from communicating *entities*.

III. FUNCTIONAL COMPONENTS OF THE ARCHITECTURE

A. *Naming and addressing for multi-domain mobility*

The current Internet has only two global naming domains, DNS names and IP addresses [5]. These domains are tied to administrative domains and network topology, respectively. The rigidity and limitation of these name domains are responsible for a variety of architectural shortcomings. The Internet therefore lacks a mechanism for directly and persistently naming data and services by relating naming to the hosts on which they reside. This makes it inconvenient to move or replicate service instances and data (*entities*). It is

therefore proposed in the Layered Naming Architecture [6], [7] to introduce a new flat global name domain for services (SID domain). A Service ID (SID) is an identifier assigned to a given service for its unique identification.

Multiple naming domains for services still have to be supported. This applies at least until migration from the existing name spaces to a new flat name space is undertaken. The payoff of such name-spaces may be significant, and recent development in Distributed Hash Tables (DHTs) [8] has made their use practical also in the global scope. Once a flat naming domain is established, it can be used to name any type of entity or data. Additionally the extra naming layer will shield applications from the underlying routers, avoiding dependence on the legacy Internet infrastructure. This separation of location from service identity (SID) allows the network to use any addressing scheme and multiple addressing domains independently of the naming scheme.

This decoupling of identity from location and addressing gives the freedom of choice and adaptability for the delivery path for packets. An *entity* may move within the same system or to a different system. It may also move because the entire system moves, either geologically or topologically (e.g., renumbering, provider-based addressing, etc.). Such movement only requires the system to update the required SID to location bindings e.g. at the involved Mobility Anchor points.

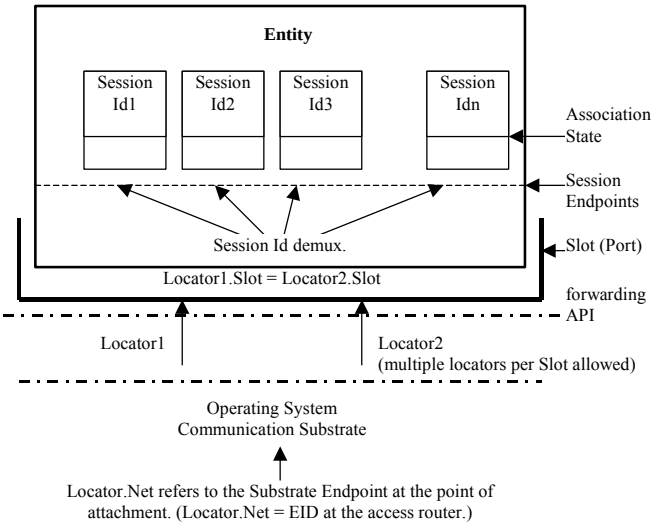

Fig. 3 Entity, slot and the communication substrate.

The following hierarchical naming framework will support the desired separation of identity from location:

- User-level descriptors (keywords, URLs, etc.).
- Host independent Service Identifier (SID – may be interpreted as an *entity* name).
- Locator
  Locator.Slot refers to the *slot* (e.g. port) within the end-system where an *entity* may be attached. Locator.Net refers to the *communication substrate* endpoint at this end-system (e.g. the IP address).
- IP address (Point of attachment).
- Path identifier (Formula 1) will be required to identify physical paths within the network between *communication substrate* endpoints (e.g. point of attachments).

Additionally there is a need for separate identifiers for hosts and network nodes. These may be used in describing network topologies and network related functionality like routing, resilience, traffic engineering etc. The topology independent Endpoint Identifier EID [7] is introduced for this purpose:

- EID.Host identifies the host or network element globally. (It is important to note that a host can also be an *entity*, but in that case it will depend on the context which identifier will be applied e.g. for mobility management.), and
- EID.Interface identifies a specific interface locally scoped at a host or a network element, e.g. at a point of attachment.

It should be noted that EIDs may be used both internal to the network (i.e. for network element identification) as well as for (mobile) host identification. This means that a *communication substrate* endpoint involves two EIDs: one on the network side (e.g. at the access router) and one at the (mobile) host.

Flat name spaces, without external semantic bindings, are proposed in [7] for the SID and the EID. This may require additional association of meta-data of some form (e.g. human understandable text, or subject classifications). This will be analyzed more in detail in the context of the Ambient Network Project. The *locator* mechanism allows communication with an *entity* (or service) also without knowing its SID.

Name resolution can be verified applying cryptographic techniques. The resolution levels supporting the above naming framework are as follows:

1. User-level descriptor → Service Identifier (SID)
2. Service Identifier (SID) → Locator*. *Locators* reference *slots* where a serving *entity* is supposed to be attached. This resolution may e.g. be made source dependent as proposed in [4]. A SID → EID'* resolution, providing the hosts and *slots* for the service, with a following EID' → Locator resolution is less desirable since a service may relocate independent of host mobility.
3. Locator.Net → IP address/EID (point of attachment) A network side EID could replace the IP address, and the addressing and routing could be directly based on this EID. (e.g. Locator.Net = EID)

*) This is intentionally a one to many mapping. (There is a one to many mapping between a SID and *locator* since a service may be replicated at different locations. A *locator* references a selected *entity* providing the service.)

EIDs are not always needed since old parts of the network would need to rely only on IP addresses for a considerable amount of time.

Additionally the following mapping at the substrate level is defined to identify all paths leading to an *entity* referenced by a *locator*:

- Locator → Path Id*

Additionally there is a one to one mapping between the path id and the flow id, and a many to one mapping between flow id and session id (see the sessions and mobility paragraph for a description of paths and flows).

B. Ambient Service Interface

The Ambient Service Interface API will have to offer a set of service primitives. One important primitive specify the characteristics of the requested bearer in terms of QoS, security, resilience, cost etc. This information would be supplied as parameters to the primitive as shown in Formula 1.

PATH (Locator, Path, Policy, QoS,
Resilience, Security, Charge, etc.)

Formula 1 PATH primitive with QoS and bearer parameters.

The *locator* in the PATH primitive is an opaque object that specifies the point of attachment and a *slot* in the correspondent node to which communications is desired (Fig. 3). The path parameter of the primitive is also an opaque object as seen from the service user. This parameter does not require being valued at the invocation of the PATH primitive, but the RESV-COMMIT primitive, giving a response to the PATH (Fig. 4), will return path identification. The PATH primitive can be invoked more than once for the same *locator* to create alternative paths. To sum up, the *locator* defines where to go, while the path identifies how to get there.

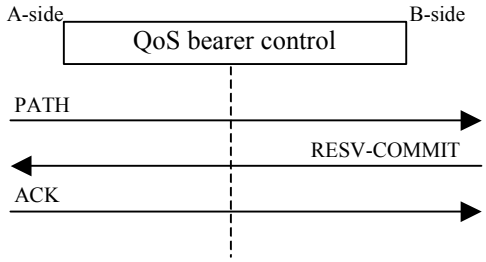

Fig. 4 QoS-controlled bearer service.

The QoS parameter carried in the PATH primitive of the bearer establishment (Fig. 4) defines the initial QoS level. It also defines the limits for QoS modifications that can be allowed without QoS re-negotiation. Re-negotiation is carried out by the applications e.g. by re-invoking the bearer service primitives. The network will initiate such re-negotiation when failing to maintain a level of QoS within the established boundaries.

Short transactions may not benefit from explicit bearer control. It should therefore be possible for an application to request a default QoS level that shall be applied e.g. until an explicit request is submitted overriding the default setting.

When an *entity* wants to send a service data unit for one of its sessions, it hands that payload to its *communication substrate*, via the DATA primitive (Formula 2). The destination *locator* contains the information needed to cause eventual delivery of the packet to the desired destination *entity*. Every *locator* will have a network (Locator.Net) part and a local-delivery (Locator.Slot) part (see Fig. 3). The network part controls delivery of the packet, via the *communication substrate* endpoint (i.e. via. EID.Interface), to a protocol stack in the node containing the *entity*, and the local-delivery part is used to complete delivery to the designated *slot* where the *entity* is attached. This is analogous to an {IP address, port number} pair in the current architecture.

DATA (Locator, Path, Payload)

Formula 2 DATA primitive

The Data service allows communication with a remote *entity*, via the *locator*, without necessarily having to know its name, SID or EID. Furthermore, the mechanism used to learn the content of the *locator* may be global, or it may be local. This asserts the possibility of both local and global addressing domains.

Entities should treat *locators* as opaque objects. The *locator* management functions and the related mobility management is local to the Ambient Service layer, and is accessed via an API that *entities* use for creating, analyzing, saving, and updating the *locators*. When a mobile *entity* moves to a different *slot*, the *locators* for existing sessions and flows must be updated to deliver to the new *slot*.

The *slot* in Fig. 3 is possibly supported by multi-homing, with network attachment points reached via Locator1 and Locator2. This represents an important resilience and load balancing mechanism that can be used on two levels. Firstly two different *locators* (e.g. with a common *slot*) may be applied to force the mobility management to route the flows via different points of attachment at the *communication substrate*. Additionally the *communication substrate* may offer disjoint paths for each of the *locators*. This QoS functionality may be offered via the bearer API for selected paths, and relying on the substrate level routing.

A topology independent *communication substrate* endpoint, called EID here (i.e. a node name and an interface) is identifying itself to the network, to the "authorities", and so on. Such identities are required to allow resilience and load balancing through selection of disjoint routes etc.

In the case when no IP addresses are applied (e.g. in a new architecture after its adoption) there is a need to base the addressing on the EID. This is the major reason for the need of a logical construct of the form EID.Interface additional to the EID.Host. (Such an EID naming structure is also combinable with flat naming).

An Endpoint Identifier (EID) can thus substitute the IP address for routing (to a point of attachment). This may in the long term obviate the need for the Locator.Net → IP address resolution.

Distributed Hash Tables (DHTs) are assumed to be well suited for support of EID based routing, but this needs further verification.

Basing routing on EIDs and DHTs can in principle be done in two very different ways:

1. The first alternative is to use e.g. a DHT to look up an address for the EID.
2. Another alternative is to completely avoid the additional address by using e.g. a DHT for direct routing of each and every packet.

The second alternative is interesting since e.g. multicast and Mobility Management represent a natural extension of DHT based routing. This remains as future work.

Middle-boxes (such as NATs, firewalls and transparent caches) are used as last resorts due to functional limitations in the Internet architecture. This results in violating IP semantics and making the Internet application-specific. Redirection is therefore proposed in [7] to allow a network *entity* to direct resolutions of its name not only to its own location, but also to the locations or names of chosen delegates or proxies. This mechanism is called receiver-controlled intermediaries and is discussed in detail in [7].

The destination abstraction introduced in the Layered Naming Architecture [7] was generalized to represent a sequence of destinations (i.e. a sequence of EIDs, *locators* or SIDs). The intermediary points, reached via the *communication substrate*, do not only forward the packet but may act on them in non-trivial ways (e.g. by carrying out rerouting, address translation, encapsulation etc.). These intermediaries represent source-controlled (as opposed to receiver-controlled) intermediaries. The mechanism of stacked identifiers can be applied for implementation of the mechanism. An API for intermediary control with associated substrate level signalling for path control may be defined. This delegation or redirection at the Ambient Network Resource layer can find more uses, e.g. for load balancing.

C. Sessions and mobility

Entities communicate with each other using logical communication links named sessions. A session implies persistent communication state within the linked *entities*. (A session may be considered to be roughly analogous to a transport-layer connection in the current architecture.) Each packet belongs to exactly one session, and an *entity* may have multiple concurrent sessions that each could involve multiple flows. Sessions are purely end-to-end, they are known only to their *entities* and are invisible to the routers. However, there is a one to one mapping between the flows constituting a session and the paths carried at the *communication substrate*. Paths are under QoS and Mobility Management below the Ambient Service Interface.

Each packet carries a session id that enables the receiving *entity* to de-multiplex the message, within the *slot*, to its session (see Fig. 3). The name domain of session ids is local to an *entity*, and thus unchanged by movement of the *entity*.

Session establishment between *entities* A and B will generally involve some kind of handshake that begins when A sends an initial message to B. *Entity* A must obtain a *locator* to reach B. Subsequent packets from A to B will also carry a session id valid for the B-side of the session, but the initial packet cannot, since session id's are local to *entities*. Thus, there is a need for initialization that may be supported by a Directory System (DS) and some kind of a Rendezvous mechanism [9]. An in depth description of session establishment may be found in [4].

A session is composed by one or more flows. Each of these flows has allocated a distinctly identified path within the *communication substrate*, and each flow may thus be allocated different characteristics (e.g. QoS, resilience, etc.) via the PATH primitive. The level of granularity for the Ambient Networks Mobility Management (MM) is the path (flow).

Session mobility implies MM for all paths/flows within a session. The basic functionality required for session mobility is available for a normal handover initiated on layers below the application. However, session mobility has to be initiated by the session user (e.g. by the application), and may cross device boundaries. The initializing *entity* must therefore specify the *locator* identifying the *slot* of the new *entity* to which the session should be moved. The initiating entity must additionally transfer its association state to the new *entity* by initiating an application level context transfer. This context transfer establishes the session endpoint within the new *entity* and identifies the triggering event for the session handover to be carried out within the Ambient Network Control Space. This assumes that the initiating *entity* requests the Ambient Network Control Space to complete the handover. The path establishment(s) to the new *entity* is implicit in a normal handover. However, the triggering event could carry an identification of selected flows (e.g. a subset of active flows in a multi-flow session) to allow sub-session mobility.

Session handover controlled by a third party is also possible. Such a capability could be an interesting building block of advanced new services.

D. End-to-End Security

At the network level, decoupling location and identity means that using IP routing to send a packet to a given location (via IP) no longer means that the packet is going to the host with the intended identity (EID). On the other hand, because EIDs are flat, they can hold cryptographic meaning. The identifiers could be derived e.g. by hashing a public key. As a result, two communicating parties, given each other's identifiers, can authenticate each other in a way that they could not if hosts were identified only by IP address. The Host Identity Protocol (HIP) [10] is exactly designed to address such security issues and many of its mechanisms can be inherited by the new architecture. A more in depth discussion of security is outside the scope of this paper, and interested readers are referred to [4], or to current work in the context of the Ambient Networks Project [11].

IV. Conclusions and Further Work

We have developed a business oriented architectural framework as a basis for further elaborations, e.g. within the Ambient Network Project. The characteristic of this framework is that it should allow diverse players in the cyber-business to revitalize the innovation that has made Internet such a tremendous success. This architecture fully supports mobility up to the session level and can serve as a vehicle for converging mobile and wire-line communications by treating mobility as the ordinary case allowing potentially all end-systems to move either alone or in groups. Mechanisms for QoS and resilience control are also inherent in the architecture, and the proposed layering with API support ensures freedom of choice and flexibility in innovation both in dimensions of technology and functionality.

The following aspects of the architecture are for urgent further study within the Ambient Network project: Use of flat names, without external semantic bindings, for the SID and the EID; DHT applicability for e.g. name resolution and routing; Details of *locator* based handover management, and last, but probably most important, the security aspects.

Acknowledgment

The authors would like to thank all members of the Ambient Networks work package on mobility and moving networks, and Jukka Ylitalo for providing valuable comments.

References

[1] http://www.ambient-networks.org/.
[2] "Tussle in Cyberspace: Defining Tomorrow's Internet", David D. Clark, John Wroclawski, Karen R. Sollins, Robert Braden, SIGCOMM'02, August 19-23, 2002, Pittsburgh, Pennsylvania, USA, http://www.acm.org/sigs/sigcomm/sigcomm2002/papers/tussle.pdf.
[3] "NewArch project", http://www.isi.edu/newarch/.
[4] "FARA: Reorganizing the Addressing Architecture", D. Clark, R. Braden, A. Falk, V. Pingali, ACM SIGCOMM 2003 Workshops August 25&27, 2003, http://www.isi.edu/newarch/DOCUMENTS/FARA.FDNA03.pdf.
[5] "Naming, Addressing and Identity Architecture", draft IST-2002-507134-AN/ WP1/R04, 2004-12-17.
[6] "IRIS Project", http://project-iris.net/.
[7] "A Layered Naming Architecture for the Internet", Hari Balakrishnan, Karthik Lakshminarayanan, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, Michael Walfish. In the Proceedings of the ACM SIGCOMM Conf., Portland, OR, September 2004, http://project-iris.net/irisbib/papers/naming:sigcomm04/paper.pdf.
[8] "Chord: a scalable peer-to-peer lookup protocol for internet applications", Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan, IEEE/ACM Transactions on Networking (TON), Volume 11, Issue 1 (February 2003), pp. 17-32, ISSN: 1063-6692.
[9] "Internet Indirection Infrastructure", I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, Proc. ACM SIGCOMM 2002, pp 73-86, August 2002, http://i3.cs.berkeley.edu/publications/papers/i3-sigcomm.pdf.
[10] "Host Identity Protocol Architecture", Moskowitz, R. and P. Nikander, IETF work in progress, October 16 2004, http://www.ietf.org/internet-drafts/draft-ietf-hip-arch-00.txt.
[11] "Ambient Networks Intermediate Security Architecture", IST-2002-507134-AN/ WP7/D01, 205-01-10.