Ambient Networks: Bridging Heterogeneous Network Domains

Bengt Ahlgren, Lars Eggert, Börje Ohlman and Andreas Schieder

***Abstract* — Providing end-to-end communication in heterogeneous internetworking environments is a challenge. Two fundamental problems are bridging between different internetworking technologies and hiding of network complexity and differences from both applications and application developers. This paper presents abstraction and naming mechanisms that address these challenges in the *Ambient Networks* project. Connectivity abstractions hide the differences of heterogeneous internetworking technologies and enable applications to operate across them. A common naming framework enables end-to-end communication across otherwise independent internetworks and supports advanced networking capabilities, such as indirection or delegation, through dynamic bindings between named entities.**

I. Introduction

ACCESS to information and the possibility to communicate play an ever-increasing role in people's lives. The communications industry is addressing this need through a large number of different approaches and products. Today, numerous networks that offer diverse services are available to users. However, access to these networks is often restricted due to security and business considerations. Usage requires pre-established, per-network subscriptions; although static, pre-established roaming agreements can extend the scope of these subscriptions to some other networks. In addition, incompatibilities and inconsistencies between network functionality – beyond basic data forwarding – limit the potential usefulness of and the available networks.

The *Ambient Networks* project is currently addressing these challenges [6][7]. The project's main objective is enabling seamless interoperation between heterogeneous internetworks. *Ambient Networks* aim to establish this interoperation through a common control plane distributed across the individual, heterogeneous networks. This new common control plane functionality can be deployed both as an integral component of future network architectures and as an add-on to existing, legacy networks, enabling legacy interoperability. In this case, the common control plane functionality "wraps around" legacy control functionalities, encapsulating and abstracting their individual control idiosyncrasies to provide at least the required subset for future interoperation. This well-defined migration path is first-order priority of the project and enables already deployed, legacy infrastructure to participate in the advanced internetworking capabilities provided by the *Ambient Networks* architecture.

Figure 1 illustrates the logical organization of the "control space" functionality in *Ambient Networks*, illustrating how the common, distributed control space encapsulates both legacy and future internetworking infrastructures ("Ambient Connectivity") and showing example functionality such as support for overlay networks or network context. Figure 1 also highlights the three interfaces that are key features of the *Ambient Networks* control space (hourglass objects in Figure 1). These interfaces – the *Ambient Service Interface, Ambient Network Interface* and *Ambient Resource Interface* – are independent of specific network architectures and network entities interact with the new control space through them.

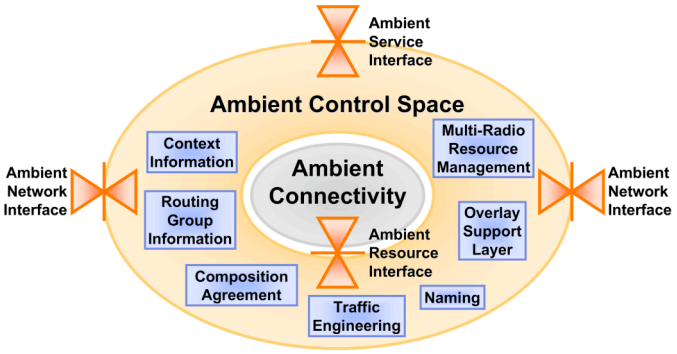

Figure 1: Control space modularization and interfaces.

This paper focuses on mechanisms of the *Ambient Networks* architecture that harmonize heterogeneous connectivity and provide a uniform internetworking environment to services and applications. These mechanisms form a new internetworking concept that is more flexible than traditional IP internetworking. Similar to IP, *Ambient Networks* interconnect independent realms that may use different local network technologies and may belong to different administrative or legal entities. Additionally, *Ambient Networks* enables advanced internetworking capabilities including node, session and network mobility as well as an architecturally clean approach to multihoming.

Note that choosing IPv6 as a new internetworking layer is not a solution. IPv6 has many of the same architectural shortcomings as IPv4. For example, they both lack native support for mobility due to locator overloading for both node identity and location. They also both do not support dynamic internetworking of otherwise independent network domains. Essentially, IPv6 is identical to IPv4 with a larger address space.

Manuscript received March 30, 2005. This work is a product of the *Ambient Networks* project supported in part by the European Commission under its *Sixth Framework Programme*. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the *Ambient Networks* project or the European Commission.

Bengt Ahlgren is with the Swedish Institute of Computer Science (SICS), Kista, Stockholm, Sweden (e-mail: bengt.ahlgren@sics.se). He is also partly supported by the *Winternet* research program funded by the Swedish Foundation for Strategic Research. Lars Eggert is with NEC Europe Ltd, Network Laboratories, Heidelberg, Germany (e-mail: lars.eggert@netlab.nec.de). Börje Ohlman is with Ericsson Research, Kista, Stockholm, Sweden, (e-mail: Borje.Ohlman@ericsson.com). Andreas Schieder is with Ericsson Research, Aachen, Germany (e-mail: Andreas.Schieder@ericsson.com).

This paper describes two mechanisms needed to realize the new *Ambient Networks* internetworking concept. First, it introduces connectivity and resource abstraction frameworks. These abstractions ensure that both control space functions and the applications using them remain independent of underlying infrastructure technologies. Second, *Ambient Networks* introduces a naming framework, consisting of an entity hierarchy with dynamic bindings that support connectivity across heterogeneous network domains and provide advanced capabilities, such as delegation and indirection.

As this paper reports on on-going work, there are still many issues to solve. One important issue is whether we need a new global internetwork namespace, or if name translation at realm borders is sufficient.

Section II describes the connectivity abstractions in more detail and Section III introduces the naming framework. Section IV presents an overview of related work and Section V summarizes and concludes this paper.

II. Connectivity Abstractions

Ambient Networks enable interoperation of legacy networks by abstracting from the intricacies of legacy connectivity planes and by providing a number of common communication primitives to services and applications. These two "connectivity" abstractions are provided through two interfaces of the *Ambient Networks* control space: the *Ambient Resource Interface (ARI)* and the *Ambient Service Interface (ASI)*.

Figure 2 shows a high-level illustration of the two connectivity abstractions and how they integrate with the overall architecture. The *bearer* abstraction is exposed to applications, while the *flow* abstraction allows the control space to remain independent of the underlying network technologies. The flow abstraction enables control space functionality to manipulate and control abstract connectivity plane entities. It can consequently remain technology-independent.

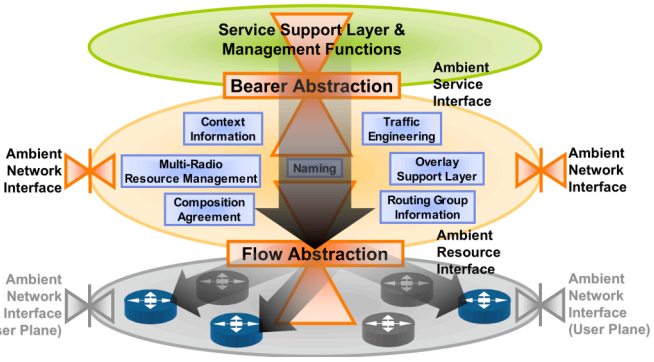

Figure 2: Illustration of the *bearer* and *flow* connectivity abstractions.

The bearer abstraction provides end-to-end communication primitives to applications and services. Examples include simple datagram and flow communication primitives as well as advanced transport overlays that may use network intermediaries to provide customized transport capabilities to applications. The remainder of this section discusses the flow and bearer abstractions in more detail.

Just as general networks, *Ambient Networks* consist of *nodes* and the *links* that connect them (ignoring for a moment the different functionalities provided by different types of nodes). This simple definition, illustrated in Figure 4, is general enough to describe many types of networks, regardless of whether they use circuits, packets or other data transport mechanisms. Networks generally also exhibit some notion of *paths*, which denote the sequence of nodes that traffic between any two nodes passes through at a given time. Paths may be static over the lifetime of a communication session between two nodes, such as in circuit-switched networks, or they may change during the course of the session, such as when routing changes in packet-switched networks. This abstract definition of networks forms the basis for connectivity in *Ambient Networks*.

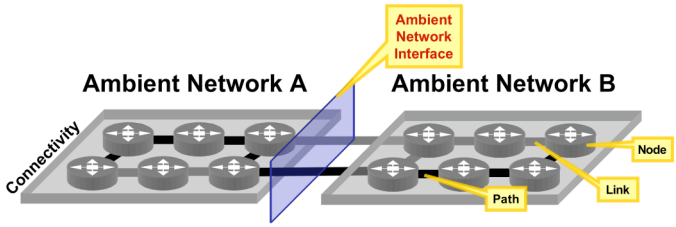

Figure 3: Abstract connectivity planes in *Ambient Networks*.

The control functions of *Ambient Networks*, however, do not operate at the level of nodes, links and paths. The ARI provides a higher-level flow abstraction that only exposes the presence of some of these entities when required. Figure 4 illustrates how the flow abstraction hides details of the underlying connectivity plane and provides *flows*, *flow transits* and *flow endpoints* as entities on which the control space functions operate.

A *flow* is an abstract view of the connectivity provided by the underlying network technology. Flows are constrained to a single network technology; they terminate at technology boundaries, as illustrated with the upward-going lines in Figure 4 (two flow end-points at the middle line). A flow is a transfer of data between two instances of the ARI, where a technology dependent locator labels each *flow endpoint*. Flows are unidirectional, so they at the minimum associated with specific source and destination locators. For some network technologies, flows may require a set-up procedure; for others, that is not necessary. A flow may pass through intermediate resources not explicitly tied to the flow, but controlled through the ARI. These intermediaries are called *flow transits*. The set of intermediaries may change over the lifetime of a flow without changing the flow itself. The flow may also pass other nodes that are invisible and thus not controllable through the ARI.

Flows transfer data between successive nodes using the underlying connectivity functionality. The control space may use the control and configuration capabilities of the ARI to request certain treatments by the connectivity layers. Flows transfer data transparently, with certain performance characteristics, which may include level of integrity of the data. Mobility of a data transfer, if it is not implemented at the link layer, requires modifications to flows due to the inherent changes to locators that node mobility causes.

The flow abstraction can be compared to the *communication substrate* of FARA [3]. However, important and deliberate differences exist. Flows are constrained to single network technology domains, whereas the communication substrate provides end-to-end connectivity across technology domains. The specific definition of a flow allows the internetworking functions to control the mapping at technology boundaries. The flow abstraction is not an interface – it is merely an abstraction of the connectivity provided by the underlying tech-

nology, while the communication substrate is an end-to-end communication primitive in its own right.

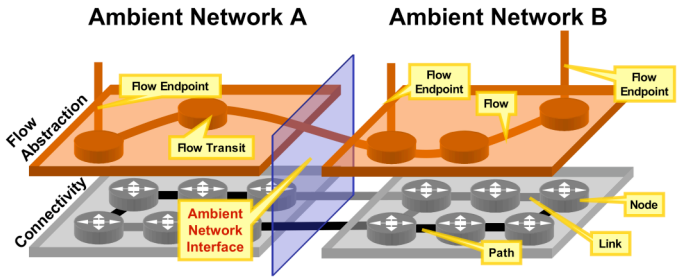

Figure 4: The *flow* abstraction in the *Ambient Networks* connectivity framework.

Whereas the flow abstraction provides an idealized, common connectivity plane to control space functions, the *bearer* abstraction exposed through the ASI provides end-to-end connectivity primitives that may cross many *Ambient Networks* to applications and services. The data transport functions of the control space construct bearers out of sequences of flows. Some flow transits provide bearer-level transport functionality and are visible as *intermediaries*. Figure 5 illustrates the bearer abstraction.

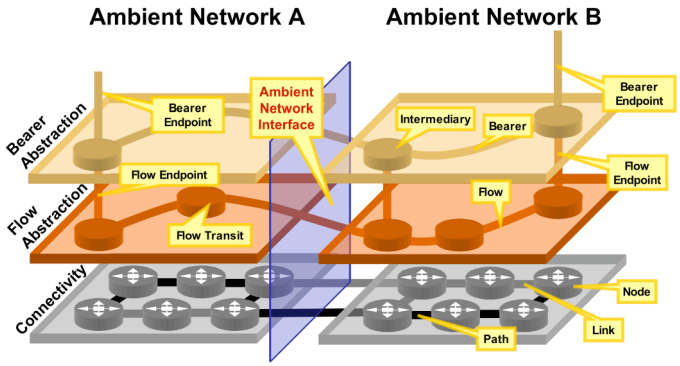

Figure 5: The *bearer* abstraction in the *Ambient Networks* connectivity framework.

Bearers run end-to-end between application peers. They are the means for end-to-end communication that an *Ambient Network* provides to applications through the ASI. Examples include simple datagram and flow communication primitives as well as advanced transport overlays that may use network intermediaries to provide customized transport capabilities to applications.

Bearer endpoints, unlike flows, are not bound to locators but to higher-level entities in the naming framework. This means that different types of bearers can use the functionality provided by the control space, such as mobility, address translation and media adaptation, in different ways to provide their applications with a customized transport service. For example, media delivery bearers may have properties that control content manipulation functions of the control space, such as transcoding. Such a bearer may map to multiple flows that terminate at the endpoints and intermediaries, respectively. The indirection support of the naming framework controls this dynamic mapping of bearers to sets of concatenated flows. Multiple bearers may map to the same flow, *i.e.*, flows can multiplex bearers between the same pair of locators.

On top of the bearer abstraction, applications define *sessions* that may combine multiple bearers into custom transport entities. Application-defined sessions are *not* part of the *Ambient Networks* connectivity framework and are shown here for completeness only. Figure 6 visualizes the use of bearers by an application to define a session between two of its instances. Figure 6 also highlights that not all nodes that exist at lower layers are visible at higher layers. With increasing layer, the view of the underlying network becomes more abstract.

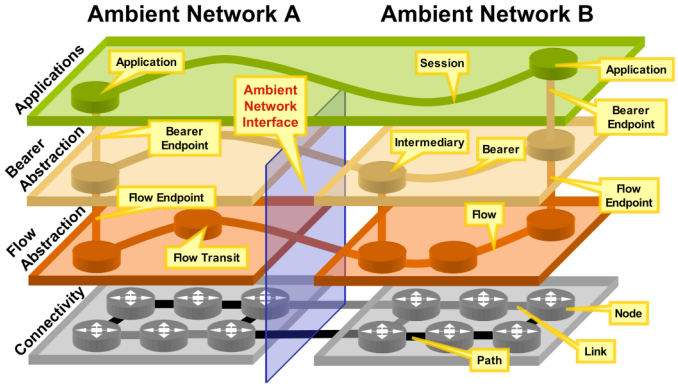

Figure 6: The *Ambient Networks* connectivity framework.

This section has outlined the connectivity framework of *Ambient Networks*, highlighting its flow and bearer abstractions. Flows abstract away from the underlying connectivity technology and provide a uniform view for control space functions. Bearers combine sequences of flows into end-to-end transport primitives for application use. A dynamic mapping between flows and bearers maintained by the naming functions in the control space enables advanced internetwork functionality. The next section focuses on the specifics of these naming functions.

III. Naming Framework

The *Ambient Networks* naming architecture adopts a layered entity model that borrows from several recent proposals [1][5] as well as from some previous ideas [9]. The model deliberately focuses on the *entities* and not the names used for the entities. With a migration perspective in mind, the architecture is designed to not prevent the use of multiple namespaces at a particular layer.

Dynamic bindings at each layer enable inherent support for the mobility of nodes, bearers and applications. This section defines and describes the entities that form the *Ambient Network* naming layers and then discusses their relations.

A. Naming Layers

An *application*, *service* or *data object* is an entity that is either a specific instance of an application service or a specific data object. The identity of the object persists over time and is not tied to the end-system hosting the service/data. Examples are SIP services or web pages. An application is reachable to clients at its *application points of attachment.* These points are located at the ASI and can be compared to standard TCP/IP sockets in the Berkeley API. A bearer, the higher-level connectivity abstraction described in Section II, connects two application points of attachment with each other.

A *host* or *end system* is a node in the network. The identity of the end system is independent of its current location(s) in the network topology and do not depend on the use of a particular communications interface. The term *end system* does not necessarily denote a physical machine – it may identify a

logical entity that may move between physical machines. An end system is the entity "hosting" the ASI and ARI interfaces. Chiappa's concept of an "endpoint" [2] is similar to these end systems, but puts less emphasis on the node aspect.

End systems attach to the network at *network points of attachment.* These entities also define generic locations in the network topology. Locations are identified with some sort of *network address* or *locator*. These locators often depend on the network topology and technology used. Locators are exposed at the ARI. A flow, the lower-level connectivity abstraction, runs between two network points of attachment.

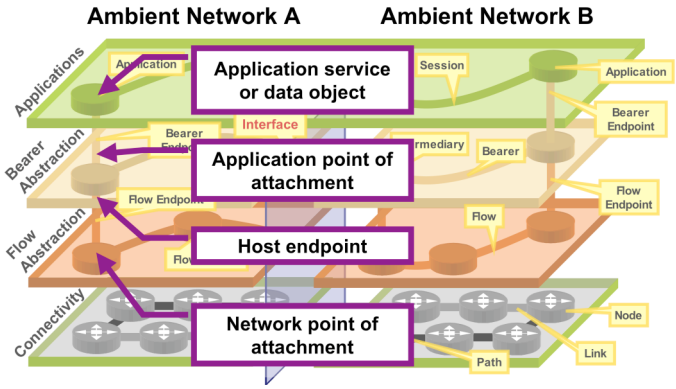

Figure 7: Entities in the naming architecture.

B. Dynamic Bindings Between Levels

The purpose of defining a layered naming architecture is to provide dynamic bindings between entities at different levels. With dynamic bindings, names of entities become location independent. Furthermore, different types of mobility, such as for nodes and services, can be supported natively without resorting to add-on mechanisms.

The control space manages the dynamic bindings between entities and provides resolution mechanisms that map names for entities at higher layers into names for lower-layer entities. Usually, the lower-layer name is the "location" of the named higher-layer entity.

The *Ambient Networks* project is currently investigating which dynamic bindings between entities in its connectivity abstractions are required to support the desired internetworking functions and which management functionality is required to support scalable internetworking following this approach. One example is the dynamic binding between bearers and flows. When nodes change their points of network attachment, their established bearer entities remain logically unchanged. The control space transparently updates the mappings of bearers to flows to support continued transport service at the new location. The details of this and other bindings are the scope of ongoing research in the project.

C. Indirection and Delegation

The *Ambient Networks* naming architecture supports the notions of indirection and delegation [1]. These are extensions to the concept of dynamic bindings that enable advanced mobility schemes and the explicit control of *middleboxes*, such as network address translators (NATs), firewalls or transcoders. An entity can elect to bind its name not only to its own current location, but also to some other location where an intermediary takes care of forwarding the communication to the entity's actual location. A simple application of this mechanism enables servers to operate behind a NAT without explicit configuration.

The concept of indirection also includes the possibility to let the location of an entity be an entity of the same kind. That is, it does not restrict the binding to entities at lower levels, but also allows bindings "horizontally" within one level. One important application is to enable efficient mobility mechanisms for moving networks. A node in a moving network binds its location to a designated gateway node. Only the gateway needs to update its bindings to new network locations as the whole network it gateways for moves. Another example of an application for horizontal bindings is the dynamic creation of virtual links – or "tunnels" – in the network.

D. Bridging Across Different Addressing Realms

Two fundamental alternatives exist for bridging across different addressing (locator) realms or between realms that share an addressing realm but use overlapping regions of it. The methods are *translation* and use of a *common namespace*. These can be applied at different levels in the naming architecture.

With *translation*, the identifiers used at a particular level are translated at gateways between networks. The goal of this translation is to translate unique "foreign" locators of one domain into unique "native" ones of the other, or to establish uniqueness between locators that are reused in both domains. NATs are an example of bridging by translation that is currently in use.

Introducing a *common namespace* is another approach for bridging across different addressing realms. Here, identifiers used at a particular level come out of common namespace shared by all networks. This method inherently establishes a uniform namespace of unique locators; translation is therefore not needed. In the extreme, the common namespace is globally shared, for example, the global IPv4 address space. Note that a common namespace corresponds to the internetworking principle.

It is currently an open question whether *Ambient Networks* need a new internetworking layer that implements a global namespace or whether translation is sufficient. With several layers of entities in the connectivity architecture, it is perhaps possible to avoid exclusive choices at many levels.

IV. Related Work

The Internet Protocol (IP) was originally designed to solve approximately the same problem as the *Ambient Networks* internetworking functions: bridging across heterogeneous network domains. The current Internet, however, does not meet several of the key design assumptions of IP anymore, causing it to fail to support end-to-end communication: IP addresses are no longer sufficient to establish end-to-end communication among all nodes, due to address space reuse enabled by network address translation. Widespread packet filtering has created a network that treats packets differently based on their content. In addition, the Internet has never supported advanced network capabilities such as mobility or multihoming, and currently struggles to incorporate these functions in an architecturally clean way.

Consequently, a number of new internetworking architectures have been proposed. The *Layered Naming Architecture* [1] is one such proposal. It defines four layers of names: user

level descriptors, service identifiers, endpoint identifiers and IP addresses (or other locators). The approach borrows heavily from others – its contribution lies in the combination of existing pieces. The *Ambient Networks* internetworking functions, especially the naming functionality, borrows several of these ideas, for example, the concepts of indirection and delegation.

The *NewArch* project described the *FARA* addressing and routing architecture [3]. Its goal is to cleanly decouple node identity from node location and at the same time avoid the introduction of a new global namespace.

The *Split Naming/Forwarding* (*SNF)* architecture [4] also separates node identities from locations. A unique property of SNF is that the naming layer can also forward data in addition to resolving names.

The *Host Identity Protocol (HIP)* [5] decouples host names and identities by introducing a host identity namespace based on public-key cryptography, securely supporting host mobility and multihoming. The *Host Identity Indirection Infrastructure (Hi^3)* [8], a combination of HIP and the *Internet Indirection Infrastructure (I^3)* [10] uses I^3 for initial rendezvous and HIP for direct, more efficient and more resilient communications afterwards.

TurfNet [11] is a novel internetworking architecture that enables communication among highly autonomous and heterogeneous network domains. The architecture uses on a global identity namespace and does not require global addressing or a shared internetworking protocol.

Plutarch [12] explicitly supports heterogeneity. It introduces the concept of *interstitial functions* to translate communication among heterogeneous networks. It assumes that namespaces differ in every domain and that forwarding is based on sender selection of a context chain, together with configuration of the required interstitial functions.

TRIAD [13] is an internetworking architecture that addresses the lack of end-to-end connectivity caused by NATs through an explicit content layer. It uses identifiers rather than addresses for node identification and routing. TRIAD uses source routing and requires IPv4 in all network domains.

IPNL [14] and 4+4 [15] aim at isolating independent IP subnetworks through loose integration. They use NATs to integrate networks with potentially overlapping address spaces to avoid renumbering.

V. Conclusion

This paper presented the internetworking functions of the *Ambient Networks* architecture, that provide end-to-end communication in heterogeneous internetworking environments. They address the two fundamental challenges – bridging across different internetworking technologies and hiding of network complexity – through connectivity abstractions and naming mechanisms.

Connectivity abstractions hide the differences of heterogeneous internetworking technologies and enable applications to operate across them. A common naming framework enables end-to-end communication across otherwise independent internetworks and supports advanced networking capabilities, such as indirection or delegation, through dynamic bindings between named entities.

This paper described the current state of these mechanisms in the *Ambient Networks* architecture, focusing on the flow and bearer abstractions as well as the required dynamic binding, indirection, delegation and bridging capabilities of the associated naming framework. The project is currently refining and detailing these concepts in conjunction with the production of detailed architecture specification.

Acknowledgment

The authors would like to thank their colleagues in the network architecture group of the *Ambient Networks* project, especially Robert Hancock, Norbert Niebert and Frank Pittmann, who have provided substantial contributions that have shaped the internetworking architecture presented in this paper.

References

[1] Hari Balakrishnan, Karthik Lakshminarayanan, Sylvia Ratnasamy, Scott Shenker, Ion Stoica and Michael Walfish. A Layered Naming Architecture for the Internet. Proc. *ACM SIGCOMM*, Portland, Oregon, USA, August 30 – September 3, 2004, pp. 343-352.

[2] J. Noel Chiappa. Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture. Work in Progress (*Unpublished Internet Draft* http://users.exis.net/~jnc/tech/endpoints.txt), 1999.

[3] Dave Clark, Robert Braden, Aaron Falk and Venkata Pingali. FARA: Reorganizing the Addressing Architecture. Proc. *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, Karlsruhe, Germany, August 2003, pp. 313-321.

[4] Andreas Jonsson, Mats Folke and Bengt Ahlgren. The Split Naming/Forwarding Network Architecture. Proc. *First Swedish National Computer Networking Workshop (SNCNW)*, Arlandastad, Sweden, September 8-10, 2003.

[5] Robert Moskowitz and Pekka Nikander. Host Identity Protocol Architecture. *Work in Progress* (Internet-Draft draft-ietf-hip-arch-02), January 2005.

[6] Norbert Niebert, Andreas Schieder, Henrik Abramowicz, Göran Malmgren, Joachim Sachs, Uwe Horn, Christian Prehofer and Holger Karl. Ambient Networks – An Architecture for Communication Networks Beyond 3G. *IEEE Wireless Communications*, Vol. 11, No. 2, April 2004, pp. 14-21.

[7] Norbert Niebert, Hannu Flinck, Robert Hancock, Holger Karl and Christian Prehofer. Ambient Networks – Research for Communication Networks Beyond 3G. Proc. *IST Mobile Summit*, June 2004.

[8] Pekka Nikander, Jari Arkko and Börje Ohlman. Host Identity Indirection Infrastructure (Hi3). Proc. *Second Swedish National Computer Networking Workshop (SNCNW)*, Karlstad, Sweden, November 23-24, 2004.

[9] Jerome Saltzer. On the Naming and Binding of Network Destinations. In *P. Ravasio et al. (ed.), Local Computer Networks*, North-Holland Publishing Company, Amsterdam, 1982, pp. 311-317. (Reprinted as RFC 1498, August 1993.)

[10] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker and Sonesh Surana. Internet Indirection Infrastructure. Proc. *ACM SIGCOMM*, Pittsburgh, PA, USA, August 2002, pp. 73-88.

[11] Stefan Schmid, Lars Eggert, Marcus Brunner and Jürgen Quittek. Towards Autonomous Network Domains. Proc. *8th IEEE Global Internet Symposium*, Miami, FL, USA, March 17-18, 2005.

[12] Jon Crowcroft, Steven Hand, Richard Mortier, Timothy Roscoe and Andrew Warfield. Plutarch: An Argument for Network Pluralism. Proc. *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, Karlsruhe, Germany, August 2003, pp. 258-266.

[13] David R. Cheriton and Mark Gritter. TRIAD: A Scalable Deployable NAT-based Internet Architecture. *Stanford Computer Science Technical Report*, January 2000.

[14] Paul Francis and Ramakrishna Gummadi. IPNL: A NAT-Extended Internet Architecture. Proc. *ACM SIGCOMM*, San Diego, CA, USA, August 2001, pp. 69-80.

[15] Zoltan Turanyi, Andas Valko and Andrew T. Campbell. 4+4: An Architecture for Evolving the Internet Address Space Back Towards Transparency. *ACM SIGCOMM Computer Communication Review*, Vol. 33, October 2003, pp 43-54.