Invariants – A New Design Methodology for Network Architectures

Bengt Ahlgren[†] Marcus Brunner[‡] Lars Eggert[‡] Robert Hancock[§] Stefan Schmid[‡]

[†]Swedish Institute of Computer Science
Box 1263
SE-164 29 Kista
Sweden
bengt.ahlgren@sics.se

[‡]NEC Network Laboratories
Kurfürstenanlage 36
69115 Heidelberg
Germany
{brunner, eggert, schmid}@netlab.nec.de

[§]Siemens/Roke Manor Research
Old Salisbury Lane
Romsey, Hampshire, SO51 0ZN
United Kingdom
robert.hancock@roke.co.uk

ABSTRACT

The first age of Internet architectural thinking concentrated on defining the correct principles for designing a packet-switched network and its application protocol suites. Although these same principles remain valid today, they do not address the question of how to reason about the evolution of the Internet or its interworking with other networks of very different heritages. This paper proposes a complementary methodology, motivated by the view that evolution and interworking flexibility are determined not so much by the principles applied during initial design, but by the choice of fundamental components or "design invariants" in terms of which the design is expressed. The paper discusses the characteristics of such invariants, including examples from the Internet and other networks, and considers what attributes of invariants best support architectural flexibility.

Categories and Subject Descriptors

C.2.1 [**Computer Communications Networks**]: Network Architecture and Design – *network communications*.

General Terms

Design, Standardization, Theory.

Keywords

Network Architecture Design, Design Methodology, Design Principles, Invariants.

1. INTRODUCTION

In a complex system with components from many different manufacturers, standards play a key role for interoperability. As Tanenbaum observed [9]: "The nice thing about standards is that you have so many to choose from; furthermore, if you do not like any of them, you can just wait for next year's model." This observation has wide applicability. Systems can frequently incorporate new standards during their lifetime to enable new functionality. At some point, however, an existing system simply cannot support desired new functionality and some or even all of its components become obsolete. The characteristics that limit this evolution can be thought of as its *lowest common denominators* or *fixed points*. This paper uses the term *invariants* to denote such limiting properties, which cannot be changed without loss of backward compatibility. It argues that any design includes such invariants, whether they are explicitly stated or not.

One very successful system is the Internet. Its architecture is flexible in many ways. The Internet easily accommodates new applications, protocols and link technologies. Indeed, the principles of protocol layering in general and network transparency in particular practically enforce such flexibility, and have been adopted in many subsequent network architectures. Despite this flexibility, the Internet has invariants. Specifically, the physical architecture of the Internet has always been expressed in terms of interfaces attached to links, generally identified by a unique IP address. The ease with which any entity in the network can be referred to by its IP address resulted in it being used as the sole identifier in most network layer protocols, and also overloaded as an endpoint identifier for the transport layer. Although this behavior is not mandated by the fundamental principles of packet switching, the IP address – right down to its format and allocation policies – is effectively the primary invariant of today's Internet. A trivial change of the address format cannot be accommodated, even though the fundamental principles are unaltered.

This paper identifies architectural invariants as the limiting factors of system designs. Software Engineering uses the term in a similar way [8]. Formal correctness proofs for programs require verification of each instruction according to a predefined proof template. A correctness proof for a loop instruction involves explicitly identifying its variants and invariants, *i.e.*, the pieces of system state that must and must not change across iterations. Both pieces of information are critically important to ensure the correctness of a loop. This paper argues to similarly identify and evaluate both the explicit design choices of a system together with their implicit invariants. Evaluating and designing systems based on both pieces of information enables a more complete investigation of the behavior and limitations of architectures.

SIGCOMM'04 Workshops, Aug. 30 & Sept. 3, 2004, Portland, OR, USA.
Copyright 2004 ACM 1-58113-942-X/04/0008...$5.00.

The invariants of a design are the specific characteristics that limit its future adaptation, flexibility and evolvability. Two different kinds of invariants exist. *Explicit* invariants are planned characteristics resulting from deliberate decisions that are known to limit the flexibility of a design. These characteristics are the desired fixed points that define the option space of a design. Although restricting the potential directions of the evolution of a system, explicit invariants typically do so in a carefully planned way that offers significant benefits in other design areas. If chosen well, invariants can provide a consistent set of building blocks on which new capabilities and concepts can be built without requiring fundamental system restructuring. Although designs can still ultimately fail when explicit invariants were chosen wrongly, this is typically due to planning errors. We believe it is important to establish criteria that allow such errors to be avoided in the future.

Implicit invariants, on the other hand, are the unplanned side effects of deliberate design choices. They can arise where a particular function requires a common network-wide approach, but instead of explicit support, its implementation evolves during deployment and growth, for example, by overloading other network functions. Another source of implicit invariants is where an apparently simple network function acquires unanticipated social (economic or political – *c.f.*, Clark *et al*'s "tussle spaces" [2]) significance, and loses its ability to adapt to changed network requirements. This paper argues that identifying the likely implicit invariants of a system early during a design provides a better understanding of its limitations and can help to correct it. Identifying the invariants of different design approaches for the same problem can aid in comparing and evaluating approaches and establish greater confidence in the final design.

This paper argues that any system will have invariants, regardless of the design process applied. Many approaches to system design exist. Typically, design processes start with identifying explicit requirements and goals, resolve conflicts among them and then focus on deriving design candidates from them. Eventually, system designs emerge that start to conform to the initial requirements. Often, conformance to requirements is the single criterion that decides whether a design is successful.

After implementation and deployment, the environments in which systems operate change over time. A thorough design process will try to predict and accommodate these changes by including them into the requirements. Predicting the future is hard, however, and unforeseen environmental developments can limit a design in unpredictable ways. The concept of architectural invariants and their use to aid the design and evaluation of new systems are the main contributions of this paper.

Section 2 gives a more concrete definition of explicit and implicit invariants, and illustrates their use by identifying some of the main invariants of the Internet architecture and the third-generation wireless network architecture. Section 3 focuses on the concept of evaluating architectures based on their invariants. It proposes to categorize identified invariants according to a set of criteria and compare the results to the underlying design principles. Section 4 describes the use of invariants in creating new designs for a given set of requirements. Identifying invariants early during the design process complements a design process and creates a more complete understanding of the characteristics of a new design. Finally, Section 5 summarizes and concludes this paper.

2. INVARIANTS

Research into network architectures often focuses on architectural principles as a basis for guiding the design [1][2][4][7]. One example is the Internet architecture – although some would argue that today's Internet architecture is an unplanned result, the Internet still *has* an architecture that can serve as an example. The Internet's basic principles include packet switching, global self-addressing of datagrams, and – maybe most prominently – the end-to-end principle. This combination of principles has resulted in a simple service for basic communication that has proven exceptionally flexible. The Internet has incorporated new applications, protocols, node characteristics and link technologies while undergoing an explosive, multi-dimensional growth. Fast optical links, slow modem links, and lossy wireless links all interoperate – maybe not optimally where raw performance is concerned, but that is beside the point of this paper.

Other aspects of the Internet architecture are less flexible. One of its key limitations is the inability for end-to-end addressing of more than 2^{32} nodes due to ubiquitous dependence on the IPv4 address format. Although the declared successor – IPv6 – is readily available, adoption of the new protocol is slow at best. While IPv6 allows addressing of many more nodes, it does so by replacing parts of the existing architecture in ways that are not backwards compatible. Remaining IPv4 nodes cannot communicate with all IPv6 nodes, simply because they cannot address them all. Until all nodes have converted to the new protocol, Internet communication becomes fragmented.

This paper uses the term *invariants* to describe aspects of a design that limit its changeability. They can also be described as a design's *fixed points*. Independent of specific terminology, these characteristics limit backwards-compatible evolution of an architecture and require disruptive changes.

2.1 Invariants vs. Principles

The following definitions may help to clarify the relationship between design principles and invariants. They may further illustrate the benefits of considering invariants as a complement to a principle-based design process:

Design principles specify desirable, but not fully measurable objectives that aid designers during the network design process. Design principles are motivated by their expected positive consequences based either on experience or abstract reasoning.

Invariants of a particular design are specific characteristics that limit its changeability. In other words, they are the *fixed points* that limit the backwards-compatible evolution of a design. Depending on how invariants emerge, they are further categorized as follows:

- *Explicit* invariants are planned or predicted fixed points of a design based on the foreseeable consequences of applied design principles
- *Implicit* invariants are unplanned or unpredictable fixed points of a design. They are the unexpected side effects of a set of desired design principles that limit its evolution.

An example may illustrate the relationship between invariants and design principles. Two of the principles that guided the original design of the Internet architecture were robustness and the use of individually addressed datagrams. One consequence of this deliberate decision is the dependence on globally routable addresses to establish end-to-end communication in the Internet. This constraining characteristic results from the decision to build a system that is robust in the presence of network failures; in the terminology defined above, it is thus an explicit invariant. Global routeability has become a fundamental part of the Internet architecture. Attempts to soften the principle (*e.g.*, with network address translation) have resulted in considerable complications and frequently require changes to deployed applications and protocols.

2.2 Examples

The beginning of this section has defined invariants as the specific characteristics that limit the changeability of a design. This section will clarify this new concept by identifying and discussing the implicit and explicit invariants of two different network architectures, the Internet and the third-generation mobile networks.

2.2.1 Internet Invariants

Section 1 already identified one implicit invariant of the Internet architecture, maybe its most prominent one: the IPv4 address. The semantics of the IPv4 address overload locator and identifier functionality, which are logically separate concepts. When the address is used as a topological locator, it describes a node's point of network attachment. When used as an identifier, it uniquely identifies a node in the network.

This combination of logically separate functions complicates advanced network features such as mobility and multi-homing. Because of the dual use of IP addresses, changes in location implicitly appear as changes in identity. Similarly, multi-homed nodes or sub-networks attach to the network at multiple, separate locations at the same time, and this results in nodes with multiple, concurrent identities. Several proposals introduce new addressing or identity mechanisms that separate the two functions from one another [5][6][10]. Some move the invariant to the naming system [5][6], allowing not only mobility but also enabling the use of multiple, concurrent addressing schemes.

The wholesale transition from today's IPv4-based Internet to one of these new proposals would be as difficult as transitioning to IPv6. However, the new proposals enable advanced networking features compared to the simple address space extension that IPv6 provides. In a sense, IPv6 is just as constrained as IPv4, because it inherits its invariants and IPv6 addresses still combine location and identity. Deployment of one of the new naming schemes would probably result in significantly higher benefits than the current – equally difficult – effort to deploy IPv6. The distinction between these two paths is that, although the introduction of a new naming system represents a much more radical *architectural* shift, it can be achieved incrementally within today's Internet.

The Internet's transport layer also has invariants. One implicit invariant is in the use of port numbers, which act as unique flow identifiers when paired with IP addresses. The local use of flow identifiers to demultiplex concurrent connections is not itself an invariant. A second use of port numbers – the *well-known ports* – is as globally unique service identifiers. Although port numbers were originally only used by transport layer protocols (and thus had only end host significance), they have since become deeply embedded in other aspects of the Internet. Examples include firewalls, the IP security architecture [13], queue management disciplines and network management systems. These mechanisms have become so deeply embedded that more modern transport protocols, [11] and [12], have been driven to adopt similar port naming schemes, although they could have used other – maybe more appropriate – demultiplexing mechanisms. In this way, 16-bit port numbers have become an implicit invariant in their role as service and flow identifiers. RFC 3639 discusses the risks in this process [14], but also notes the commercial and operational pressures that are causing it to take place.

A commonality among the invariants of the Internet architecture may be that they are conceptually simple, but deeply embedded. The IPv4 address invariant, for example, affects *all* nodes, and the port number invariant is nearly as influential. This may signify that although the Internet's variants allow for a great deal of planned flexibility, its invariants are extremely difficult to shift.

2.2.2 Cellular Network Invariants

Modern cellular networks present an interesting contrast to the case of the Internet, both in terms of the types of invariants they expose and the impact they have on network flexibility compared to the underlying architectural principles. This section discusses the *Universal Mobile Telecommunications System* (UMTS) as developed by 3GPP, for which initial studies of evolution exist [16][17]. However, note that nearly identical conclusions would apply both to other contemporary networks and their immediate predecessors.

Like other telecommunications networks, the UMTS design is aimed primarily at the delivery of certain services to users. This is visible not only in the nature of the standardization process and the level at which system requirements are expressed, but also in the style of invariant that the system design exposes. Another difference, rooted in the underlying business models involved, is that networks are defined and evolve on a scale much larger than the individual node. The critical interfaces for evolution and interworking are those between the user and operator, and between operators, but these represent a relatively small proportion of overall network functionality compared to the case of the Internet.

The service-centric origin of UMTS has the result that the important architectural invariants are associated with describing services or their endpoints (*i.e.*, users), rather than describing the supporting network infrastructure. The primary invariant of the 3GPP family of networks is probably the concept of a hardware token, the subscriber identity module (SIM or variants), to represent not only the public identity of the user but also the contractual relationship between the user and service provider. In so far as this is an identifier, it could be compared with the corresponding invariant (the IP address) from the Internet, but the differences are wide ranging. Compared to the IP address, the SIM represents primarily an endpoint rather than a location, locations being transitory in mobile networks. Very few nodes in the UMTS network are directly exposed to the specification details of the token; instead, information derived from it is carried in a variety of higher layer protocols to configure the underlying network. All other identifiers are localized within the lower layers. However, the broad security and contractual implications of the use of identity tokens have a much more pervasive influence on business models and technology evolution, so much so that it is extremely difficult operationally to introduce new access mechanisms even for existing services unless they can be tied to SIM operation. It also imposes a model of user-device relationships that is hard to generalize.

The second significant invariant of UMTS networks is the set of services that is provided, both their definition and their instantiation in particular sessions. As is the case for identifiers, the services (a voice bearer and specific classes of data bearer) are defined at a relatively high level in the protocol stack compared to the supporting infrastructure. Although the layering is still quite strict, transparency to new services is less significant than the ability to offer the existing ones over a variety of network types. Those network types are not exposed to the user, and can change radically from system generation to generation (or even release to release). Adding a new service invariant is a much more significant operation, again as much as anything for non-technical reasons.

Both these invariants are explicit, rather than implicit: they are the result of deliberate design choices, and follow quite directly from the commercial environment within which the technology has been developed. Because the system requirements are based rather directly on end "user" requirements (and also because the development of new applications is rather tightly controlled), few if any *implicit* invariants have emerged as a result of system deployment, at least within the network layer. One example where major efforts have recently been expended is the introduction of multicast capability to the radio access and core networks [20]. Because of the way in which cellular networks have previously focused only on the problem of providing end-to-end unicast links, this has required significant re-engineering inside many elements of the network. (Given that the goals for the introduction of multicast services include charging and security functions, this is hardly surprising.)

Nevertheless, cellular networks continue largely to be used within the constraints that their original designers expected, and so they have not been challenged with the same diversity of new usage models that has exposed the implicit invariants of the Internet. Rather, again in contrast to the case of the Internet, the inflexibilities in evolution and interworking arise from limitations in the underlying architectural principles.

3. INVARIANTS-BASED ARCHITECTURE EVALUATION

This section discusses the use of architectural invariants to evaluate and compare different architecture designs. Section 4 will discuss how this analysis, applied throughout the design process, can aid in finding design alternatives that both fulfill the requirements and remain flexible. If a certain set of metrics could "measure" the invariants of a design, it may result in a method for comparison that should at least be partly objective. What, however, would be the criteria for such comparisons? Which characteristics distinguish good invariants from bad ones? How does one know if a system has enough invariants, or if it has too many?

A first approach is to simply count the number of invariants for a given design. Obviously, this approach is problematic, because a reliable method for discovering and enumerating invariants does not exist. At this point, identification of invariants entirely depends on human experience and skill. However, some general criteria for considering the overall quality of a candidate set of invariants would include at least the following:

- **Is the set complete?** A network design whose set of invariants ignores some communications requirements is vulnerable to the emergence of implicit invariants, usually by the overloading of functions intended for other purposes. An example from the Internet is the case of the continual strain on the DNS [15] caused by its extension into areas of functionality for which it was never intended. If a candidate set of invariants does not have the expressive power to handle the services that will be demanded of the network, it can be expected that implicit invariants will later emerge to fill the gaps in an unplanned way.
- **Is the set independent?** Ockham's razor suggests avoiding to provide multiple ways to express the same concept or achieve the same function within a design. More simply, because invariants imply constraints, it appears beneficial to minimize their number if extra invariants provide no additional functionality. This approach can be reflected practically in a design by the systematic elimination of redundant concepts where they can be expressed in terms of better "quality" candidate invariants, similar to the process of relational database normalization. An idealization of this criterion would be to ask for the set to be "orthogonal", with each invariant having no direct interactions with any other.

These considerations imply that any design must expose a set of invariants; a design that exhibits none (or too few) will be deficient in the long term, despite its superficial flexibility. However, these criteria alone are not enough to judge between all competing designs, and certain characteristics of individual invariants may be good basis for comparison and selection between alternatives. These characteristics include:

- **Does an invariant affect many components or just a few?** The rationale behind this argument is that invariants that are more widespread have a higher potential to cause conflicts. For instance, the IPv4 address invariant affects all nodes, but the SIM card invariant only affects some nodes in their respective networks. Consequently, the SIM card invariant may be less problematic.
- **Does an invariant affect many aspects of an architecture or just a few?** The less a specific invariant affects an architecture, the better. For example, invariants that affect only the control or data plane of a network may be less restricting than invariants that affect both. Invariants that affect multiple layers will be equally problematic, especially as the same property typically increases the number of affected nodes.
- **Does an invariant affect silicon or just bits?** One could argue that an invariant embedded in hardware poses more of a potential issue than an invariant that affects software only. By extension, invariants that affect the data plane are more problematic than those affecting the control plane (if the

latter represents a lower percentage of investment or maintenance cost than the former).

- **Does an invariant have security or privacy implications?** Invariants concerning user characteristics or behavior may be more limiting than others. Examples include the SIM card on which cellular networks depend for user authentication. In the Internet, IPv4 addresses are still frequently used by various security mechanisms, such as in IPsec [13]. Additional problems arise where these invariants become associated with regulatory constraints (location privacy, traffic interception, competition policy through identifier portability).
- **Does an invariant have internal flexibility?** An invariant, especially one used as some sort of identifier, may have the disadvantage of being ubiquitous or deeply embedded in implementations. However, if it has an evolution path of its own (typically some non-trivial syntax) this can mitigate one or both of these problems. An example from the Internet is the evolution from classful to classless IP addressing, where the IP address format could be re-interpreted with a more sophisticated syntax in a way that was transparent to end systems. Telecommunications identifier schemes have often exploited the possibility to do relatively complex processing within the control plane to use variable as opposed to fixed length identifiers.

It is important to note that the goal of evaluating the invariants of various architectural designs is to determine which design is likely to gracefully adapt to future change – in a sense, how far in the future the design's expiration date lays. This evaluation is not concerned with quality, functionality or usefulness. The designers will of course also need to consider all these aspects during the design process. Evaluating competing designs solely based on invariants may result in an architecture with minimal invariants but no useful functionality.

The main Internet invariant, the IPv4 address, is definitely ubiquitous. It affects all nodes, and has security implications. Therefore, it would rate poorly according to the above criteria. On the other hand, one might argue that *one* ubiquitous invariant is better than many "less worse" ones.

A common characteristic of the invariants in the Internet architecture is that the problematic ones are confined to the network and transport layers, *i.e.*, to the "waist" of the hourglass in the protocol stack. In this respect, they get good marks on the second criterion. This classification supports the expectation of flexibility that the architecture has shown with respect to new applications and link technologies. Confining invariants to a limited part of the design, whether physical equipment, logical layers or something else, is important to not unnecessarily limit future change. However, note that confining an invariant to the network layer is still an imperfect state of affairs, given the way in which all nodes of that layer participate in end-to-end data transfer; hence the interest in localization architectures and localized addressing schemes in recent years [3].

Are all invariants bad? No, even though invariants prevent changes to specific parts of the system, their identification is necessary to *enable* changes in the other parts. With clearly spelled out invariants, an engineer can redesign the parts that are not constrained, knowing that the architecture will not fall apart. With this argument, invariants enable evolution in that they, when well-defined, facilitate change. One conclusion that can be drawn from this argument is that an architecture is better with few, strong and well-defined invariants, than many loosely defined ones.

4. INVARIANTS-AWARE ARCHITECTURE DESIGN

One of the main objectives behind the proposed invariant methodology is to facilitate designers to assess an emerging (network) architecture in order to identify potential problems or inconsistencies with the actual architectural objectives as early in the design phase as possible.

For this, we propose that designers follow an iterative design process that requires the architects to revise an evolving architecture several times. This also means that the architecture needs to be analyzed very early in the process in order to reduce the number of redundant artifacts. At each round of the design cycle, the architects use the following process:

- **Identify the architectural invariants.** Review the draft architecture and identify its invariants. Because of various possible viewpoints about an architecture [18], identifying its invariants might involve several iterations. Nevertheless, an evaluation from any viewpoint is expected to yield some set of invariants. A formal way of identifying a complete set of invariants, however, is not likely to exist. Typically, implicit invariants may be hardest to identify. Challenging an architecture with usage scenarios that were not considered during the initial design may be an effective tool to isolate them.
- **Evaluate the invariants** against the set of characteristics given in Section 3 and determine if the invariants only apply where expected. For example, to support terminal mobility host identity should be independent from terminal location. Verify whether all invariants conform to the design requirements. For example, if a design requirement is high scalability, verify that no invariant design decision limits this characteristic.
- **Revise architecture based on invariant evaluation.** If invariants conflict, modify the architecture early to reduce the potential for inconsistencies during the implementation or deployment phases. This iterative design process helps reduce undesired side effects.

Note that evaluation based on invariants also aids in finding consensus on the quality of a given architecture. However, evaluation highly depends on the applied characteristics of an invariant. There is no proven set of characteristics, or a formal way for rating a given invariant against them. Additionally, the main target of evaluation is the degree to which the architecture can evolve over time. Functional requirements on the architecture are not taken into account in that design methodology, but they are rather obviously given by potential users/stakeholders of the architecture.

5. CONCLUSION

This paper proposed a methodology for designing new network architectures, motivated by the view that evolution and interworking flexibility are constrained not so much by the principles, but by the choice of fundamental design invariants. Therefore, the architecture design cycle includes the evaluation of the architectural invariants and tries to adapt the architecture to avoid unwanted invariants. This methodology only targets the aspect of evolution of architecture and not other important questions such as the capabilities or functional distribution of the architecture. However, we believe that an easy to evolve architecture is key for the success of a network in the long term.

This methodology based on invariants is at its early stages of development. Based on the current analysis of case studies from existing networks, there is good reason to hope that it can provide valuable guidance. However, the method can only be developed and proven by its application in the development of concrete new network architectures, which have to meet the challenges of evolution of existing systems, interworking between different network types, as well as the introduction of new network functionality. The first test case will be to apply this design methodology in the context of the European Union's *Ambient Networks* project [19]. Furthermore, we need to enhance the set of characteristics of invariants with the objective to get completeness and a more formal way of using the methodology.

ACKNOWLEDGMENTS

This document is a byproduct of the *Ambient Networks* project, partially funded by the European Commission under its *Sixth Framework Programme*. It is provided "as is" and without any express or implied warranties, including, without limitation, the implied warranties of fitness for a particular purpose. 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 also partly supported by the *Winternet* program which is funded by the Swedish Foundation for Strategic Research.

REFERENCES

[1] Robert Hancock and Josef Urban. Beyond Hosts and Routers: Some Architectural Principles for Future Mobile Networks. Presented at *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA-03)*, Karlsruhe, Germany, August 2003.

[2] David D. Clark, John Wroclawski, Karen R. Sollins and Robert Braden. Tussle in Cyberspace: Defining Tomorrow's Internet. Proc. *ACM SIGCOMM 2002*, Pittsburgh, PA, USA, August 19-23, 2002, pp. 347-356.

[3] 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-03)*, Karlsruhe, Germany, August 2003, pp. 258-266.

[4] David D. Clark, Karen Sollins, John Wroclawski and Ted Faber. Addressing Reality: An Architectural Response to Real-World Demands on the Evolving Internet. Proc. *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA-03)*, Karlsruhe, Germany, August 2003, pp. 247-257.

[5] David D. Clark, Robert Braden, Aaron Falk and Venkata Pingali. FARA: Reorganizing the Addressing Architecture. Proc. *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA-03)*, Karlsruhe, Germany, August 2003, pp. 313-321.

[6] 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.

[7] Joe Touch, Yu-Shun Wang, Lars Eggert and Greg Finn. A Virtual Internet Architecture. ISI Technical Report ISI-TR-570, USC Information Sciences Institue, March 2003. Presented at *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA-03)*, Karlsruhe, Germany, August 2003.

[8] David Gries. The Science of Programming (Monographs in Computer Science). Springer-Verlag Berlin and Heidelberg GmbH & Co. KG, 1987.

[9] Andrew S. Tanenbaum. Computer Networks. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[10] Robert Moskowitz and Pekka Nikander. Host Identity Protocol Architecture. Work in Progress (draft-moskowitz-hip-arch-05.txt), September 2003.

[11] Randall R. Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns J. Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang and Vern Paxson. Stream Control Transmission Protocol. RFC 2960, October 2000.

[12] Eddie Kohler, Mark Handley and Sally Floyd. Datagram Congestion Control Protocol (DCCP). Work in Progress (draft-ietf-dccp-spec-06.txt), February 2004.

[13] Stephen Kent and Ran Atkinson. Security Architecture for the Internet Protocol. RFC 2401, November 1998.

[14] Michael St. Johns and Geoff Huston (eds.) Considerations on the use of a Service Identifier in Packet Headers. RFC 3639, October 2003.

[15] John Klensin. Role of the Domain Name System. RFC 3467, February 2003.

[16] TSG Services and System Aspects: Evolution of 3GPP System. 3GPP TR 21.902, September 2003.

[17] TSG Radio Access Network: Feasibility Study on the Evolution of UTRAN Architecture. 3GPP TR 25.897, August 2003.

[18] Reference Model of Open Distributed Processing. ITU-T Rec. X.901-5, March 2000.

[19] 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. 14-22.

[20] TSG Services and System Aspects: Multimedia Broadcast/Multicast Service; Stage 1 (Release 6). 3GPP TS 22.146, June 2004.