

Composition of Self Descriptive Protocols for Future Network Architectures

Dennis Schwerdel, Zornitsa Dimitrova, Abbas Siddiqui, Bernd Reuther and Paul Mueller
Integrated Communications Systems Group
University of Kaiserslautern
Kaiserslautern, Germany
{schwerdel,dimitrova,siddiqui,reuther,pmueller}@cs.uni-kl.de

Abstract

The network protocols we use today have been introduced decades ago. Since then the whole Internet came to existence and with it a single protocol stack: TCP/IP. What was a good solution back then, is no longer appropriate to fulfill the emerging demands of applications. Newer protocols have been created as solutions for the problems, but replacing TCP/IP requires a complicated deployment and migration phase. The problems with the current Internet architecture and its fixed structure have triggered a discussion on a Future Internet architecture. We propose a way to dynamically select and compose protocols based on principles of service oriented architectures. The goal is a network architecture where new protocols could be easily added and are automatically and transparently used by applications. In this paper we present a way to describe protocols and their effects and dependencies between them. We also present a method to select and compose protocols.

1. Introduction

The Internet is a tremendous success story and with it the TCP/IP protocols suite, which is the core technology of the Internet. Using the Internet requires to use the TCP/IP protocols. Thus it is hard to change or even modify these protocols. Driven by the demands of ever emerging applications and the capabilities of new communication networks, many workarounds have been introduced, like sub-layer proliferation (e.g. MPLS at layer 2.5, IPsec at layer 3.5, and TLS at layer 4.5), and erosion of the end-to-end model (middle-boxes, such as firewalls, NATs, proxies, caches, etc.). This results in increasing complexity and unpredictable vulnerabilities making it even harder to introduce new technologies.

As a consequence the introduction of a *Future Internet* with a newly designed architecture is discussed in the research community (e.g. [1], [2]). One of several requirements, that should be fulfilled by a future network architecture, is the ability to evolve, i.e. to add, change and remove functionality more easily than today. Just defining a new set of protocols for a future Internet is not sufficient, because it will be impossible to take into account all future demands. In

consequence even a completely new designed future Internet will be subject to ongoing evolution. This demand can not be achieved by a single component or protocol, but must be supported by a new architecture that defines the fundamental organization of a system, the relationship of components, and design and evolution principles according to [3]. Especially the definition of evolutionary principles allowing deliberate extensions and replacement of functionality are important to avoid an architectural patchwork similar to today's Internet.

In order to enable evolution of network functionality in large scale networks it must be possible to change functionality of an individual system (network node) and in addition there must be concepts and methods to handle the resulting heterogeneity. The heterogeneity will be inevitable, as in large scale networks it is not possible to synchronize changes because of technical, logistic and even political constraints. We expect that achieving both goals will be the key for a flexible network architecture that is able to evolve.

We propose a network architecture that focuses on reducing the coupling of protocols. The Internet can be considered as a large, distributed (software) system. Hence we address a new inter-network architecture by using software engineering methodology. A promising methodology is the service-oriented architecture (SOA) paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains [4] [5].¹

In order to achieve loose coupling, protocols must not expose any internal mechanisms. This requires well-defined interfaces, which reflect only the service, i.e. the visible result, of a protocol. Then a higher degree of flexibility can be achieved by using micro-protocols [9], which interact dynamically. The interaction can be supported by techniques like message passing, publish/subscribe for notifications, generative communication, which are common in the context of SOA and avoid hard coded interaction (see [6], [7] for details). In this paper we present basic considerations on service oriented descriptions of protocols and for composing micro-protocols using such descriptions. This work is part of our ongoing research in the context of future network

1. Note: SOA is a paradigm, and thus it does not rely on a specific technology (e.g. Web Services) and is not limited to specific application areas (e.g. enterprise information integration).

architectures.

The next section 2 provides a brief overview of related work and outlines the novelties of our approach. Section 3 presents specific terminology used in this work and section 4 describes our approach. Section 5 describes elements for a service oriented description of protocols and 6 describes how to combine them. Finally the planned future work is described in section 7

2. Related Work

Developing concepts and techniques for more flexible networks have been a research topic since many years now, but with varying focuses. In the 90's, there have been several publications about automatic protocol configuration based on micro-protocols. Examples of these are the "Dynamic Network Architecture" [9], DaCaPo [10] and FuKSS [11]. All these approaches compose complex protocols using micro-protocols. In our approach we also use micro-protocols, but use more abstract descriptions and different methods for composing and controlling interactions of protocols.

The research field of programmable networks [12] also aims to achieve higher degrees of flexibility of networks. Especially the separation of the communication plane from the control plane is a major concern. Such a separation is difficult to realize in today's Internet. This is because the network nodes are vertically integrated and content providers have no direct access to the control plane. Our approach is based on horizontally distributed functionalities without layering (see section 4) and thus also simplifies the separation of control and data plane, but uses different concepts than programmable networks.

The current debate about a *Future Internet* is driven by several large initiatives world wide, like GENI [13] and FIND [14], the Clean-Slate Program at Stanford University [15] or the european Fire initiative [16].

Interesting clean-slate approaches which are related to the work presented here can be seen in the Role-Based Architecture (RBA) [17], the Service Integration, control and Optimization (SILO) [18], the Recursive Network Architectures (RNA) [19], the Autonomic Network Architecture (ANA) [20] and the 4WARD project [21].

The RBA approach introduces a non-layered architecture to the design of network protocols and organizes communication in functional units referred to as roles. Roles are not hierarchically organized and thus may interact in many different ways. The main motivation for RBA is to address the frequent layer violations that occur in the current Internet architecture, the unexpected feature interactions that emerge as a result, and to accommodate middle boxes. The SILO approach also introduces a non-layered design based on silos of services. Furthermore it offers a more flexible header structure than the RBA approach. The overall goal of the SILO architecture is to facilitate cross-layer interactions in

a manner that meets the user requirements accurately and optimizes performance. The RNA approach examines the implications of using a single, tunable protocol for different layers. RNA reuses basic protocol operations across different protocol layers, avoiding redundancy of implementation as well as encouraging cleaner cross-layer interaction. It allows protocols and protocol stacks to adjust at runtime. This results in a more dynamic composition of services, both within stacks and in the way networks combines the stacks of individual hops into an overall network architecture. The ANA project aims to design and develop a novel autonomic network architecture. ANA tries to enable a flexible, dynamic, and fully autonomous formation of network nodes as well as whole networks, based on fine grained functional block. The approach of 4ward is to create specific networks for classes of applications or even single application. The functionality of each network is defined by so called Netlets[22] which are predefined compositions of micro-protocols. 4WARD supports the definition and deployment of netlets and fosters the reuse of micro-protocols.

All these projects are similar to our approach in respect to avoid layering and aim at defining a highly flexible architecture. The main motivation for RBA, SILO, and RNA was to address the frequent layer violations and cross-layer interactions that occur in the current Internet architecture. Whereby ANA focuses on autonomic creation of networks and 4WARD aims to support a high diversity of specific networks. Even though the overall goal of all these projects is to develop a highly flexible network architecture, the strategies are different. The specific goal of our approach is to enable networks to evolve without central coordination. Our specific approach is to achieve loose coupling between micro-protocols, which will be extended by concepts to handle heterogeneity of networks.

3. Terminology

This section provides the definitions of terms important for the understanding of the proposed approach.

3.1. Mechanism

A *mechanism* provides the description of how to achieve desired functionality. A particular mechanism can be implemented by different pieces of software. Two mechanisms are considered equivalent if they attain the same functionality and obey the same specification regardless of their implementation techniques and code intricacy. In network architecture, mechanisms are normally protocols or micro-protocols. A good example for a mechanism is the TCP protocol. It is defined by a specification that describes every detail which is necessary for two peers to communicate using the TCP protocol. But the protocol does not define a specific implementation, there exist various implementations

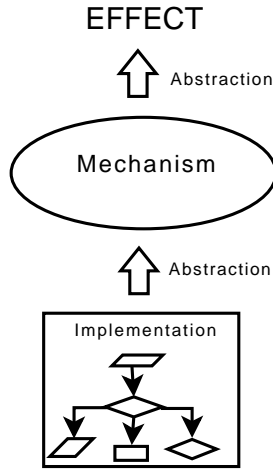


Figure 1. Effect, Mechanism, Implementation

which are all compatible because they all implement the same mechanism. Mechanisms exist at any granularity. TCP is a very complex protocol that combines a lot of different functionality. A mechanism can as well be a micro-protocol like the TTL² method. Such tiny mechanisms might be used to create a compound mechanism.

3.2. Effect

An *effect* describes the desired outcome of the use of a mechanism. An example of an effect is *Order Preservation*. There may be several different mechanisms, which produce the same effect. For example one mechanism uses sequence numbers to keep the order. Another mechanism for reliable communication could wait for an acknowledgment after sending a packet before sending the next one and thereby providing that effect as a side-effect. A mechanism may produce several effects as does the second mechanism in the example. In general, an effect is independent of the mechanism and its implementation, i.e. it should be possible to change the mechanism without changing the effect.

The term *meta-effect* is introduced in order to address effects, which comprise more than one effect, i.e. a meta-effect is generated by the cooperation of several effects. For example the meta-effect *Reliability* consists of the effects *Data Correctness* *Completeness* and *Order Preservation*.

A *service* is specified by a set of effects or meta-effects.

3.3. Correlation between Effects and Mechanisms

In this subsection the possible correlations between effects and mechanism are described.

As stated above, a single effect may be generated by different mechanisms, i.e. several mechanisms provide the

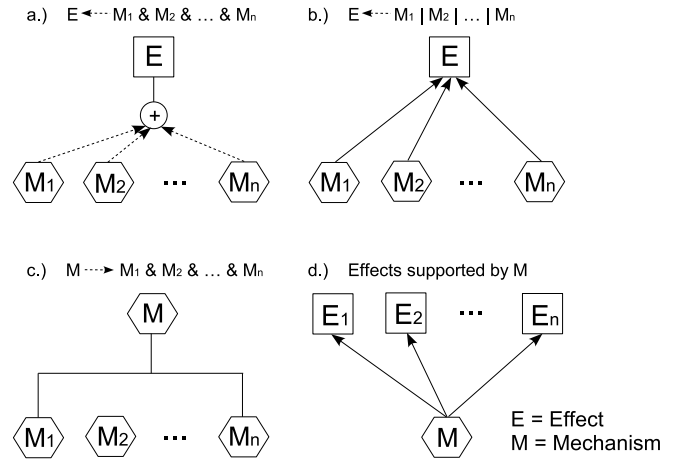


Figure 2. Dependencies

same effect (see figure 2b). Those mechanisms can differ in their implementation, and the implementations themselves can also differ in their performance. For example both *Is-Alive* and *Heartbeat* mechanisms provide the effect *Status of Communication Partner*.

Figure 2a illustrates the case where an effect is produced by the combination of different mechanisms. The effects produced by the individual mechanisms are combined into one meta-effect. For example, the effect *Order Preservation* is generated by the mechanisms *Reordering of Data* and *Detection of Duplication*.

A single mechanism may also participate in the generation of more than one effect (see figure 2d). For example the *Checksum* mechanism supports both *Integrity* and *Data Correctness*.

A mechanism can also rely on other mechanisms. In this case, the availability of those mechanisms is crucial for the functioning of the composite mechanism. Figure 2c shows this correlation.

Effects may be grouped into two categories – necessary and desired effects – depending on that, whether they are mandatory or optional for an application to work properly [8].

4. Approach

The goal of service-oriented networking is to offer fine-grained functionality to applications to choose from. A way to achieve this, is to create a list of effects and a library of mechanisms which provide these effects. When the mechanisms are encapsulated and do not depend on certain internals of other mechanisms they can be nearly arbitrarily combined. For each set of requirements of an application, a matching set of mechanisms could be assembled. Those mechanisms then form a custom micro-protocol stack (which is more like a mesh in fact).

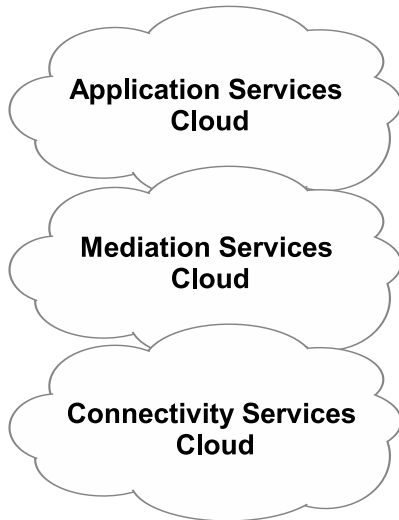


Figure 3. Service Clouds

To dynamically build these micro-protocol stacks mechanism will have to describe their capabilities and requirements in a way that an algorithm can predict the outcome of any combination. To achieve that each mechanism has a set of requirements, a set of provided effects, a description of the costs of the mechanism and a formula how these costs “add up” with the costs of other mechanisms. An algorithm selects mechanisms so that all requirements are met, all desired effects are provided and the combined costs are low. Section 6 describes this in detail.

Inside one stack, protocols communicate using notifications and a common tuple space. Two end-points exchange TLV-headers³ to communicate. A dispatcher manages the distribution of notifications and protocol headers to the micro-protocols. The features of the dispatcher highly depend on the requirements of the mechanisms, so identifying mechanisms and their effects is a critical task in designing the dispatcher.

4.1. Future Internet Architecture

In the presented approach the network architecture is not organized into layers like the architecture of the current Internet, but consists of many stand-alone, self-contained services, which communicate, cooperate, and inter-operate with each other. According to their scope, the services are conditionally grouped into three service clouds (Figure 3).

Application Services Cloud. The application services cloud offers services related to application functionality. Examples of such services are authentication, application notification service, application information exchange service, etc.

Mediation Services Cloud. The scope of the mediation services cloud incorporates services related to network functionality. Typical example of mediation services are connection establishment and release, flow control and congestion control, etc. The presented approach focuses on this cloud.

Connection Services Cloud. The connection services cloud encompasses services related to data transport. Such services are, for example, modifying network data, signaling, signaling error correction, etc.

5. Identification

In order to identify mechanisms, effects and their dependencies the TCP/IP protocol stack has been examined. It is possible to differentiate mechanisms and effects with different granularity, e.g. TCP could be one mechanism with the effect *Reliable Communication*. On the other hand, the Decrement of a TTL value and the packet dropping can be different mechanisms. As a general rule we select mechanisms and effects at such a granularity, so that an application could reasonably require the effect, and a mechanism uses only a few headers and could be legitimately called a micro-protocol.

The following effects have been identified mainly from the provided functionality of TCP/IP, despite the fact that to completely describe the functionality of TCP/IP it may take more than only these effects. The reason to identify those effects is to cover provided TCP/IP functionality in our approach towards novel Internet architecture.

Forwarding. Forwarding is a method to deliver data from source to destination such as, transferring data to the correct outgoing port within a node. There can be different mechanisms to achieve this effect, but it is more considerable how much effective a mechanism is with respect to various efficiency and performance parameters. In the existing Internet architecture this is obtained with the help of different identification and marking techniques.

Communication Supervision. The effect *Communication Supervision* concerns monitoring communication data and connection status. Two major effects, which generate the meta-effect *Communication Supervision*, are *Status of Communication Partner* and *Sorting out foreign Data*, i.e. data, which is not related to a particular communication. Examples of mechanisms for monitoring the status of the communication partner are the *Heartbeat* and *Is Alive* mechanisms. A way to sort out or to differentiate communication traffic is to obtain the partner’s identification related information.

Connection Management. The effect *Connection Management* refers to operations that are performed in order to maintain a connection-oriented communication between

3. Type-Length-Value encoded headers

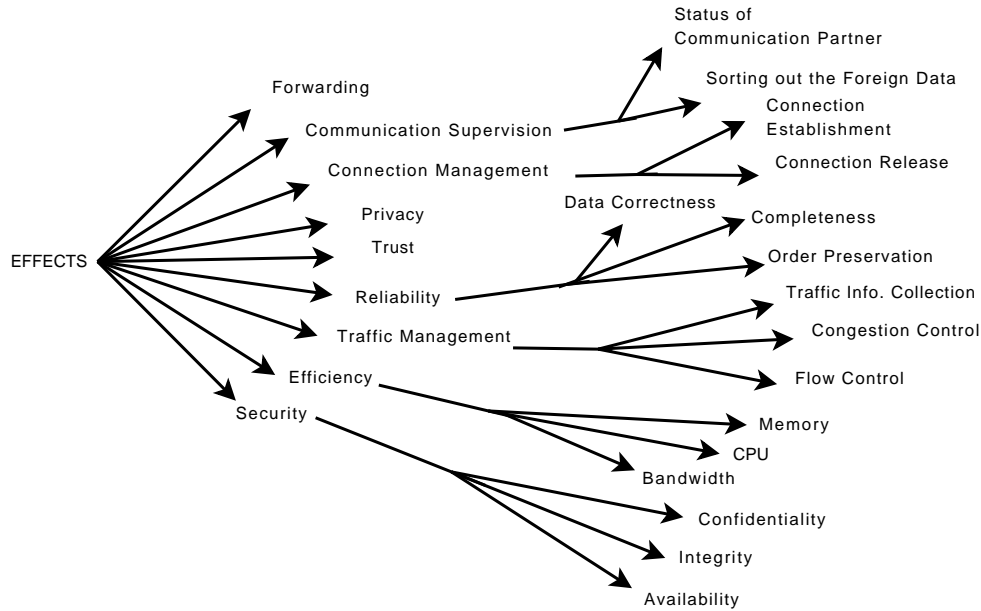


Figure 4. Hierarchy of Effects

the communication partners. The two effects fulfilling this meta-effect are *Connection Establishment* and *Connection Release*. An example of a mechanism for *Connection Management* is the *Three Way Handshake* mechanism used by TCP. The *Three Way Handshake* mechanism uses ACK, SYN, FIN, and RST bits and the sequence number and acknowledgment number field of the TCP-Header.

Privacy. The effect *Privacy* is guaranteed if the data is protected against unauthorized access. TCP/IP does not provide this effect, but protocols like SSL⁴ do.

Trust. *Trust* refers to data communication situations where the identity of the communication partner or the origin of the transmitted data has to be assured. In a distributed network like the Internet it is important to have mechanisms, which can guarantee the desired trust level. Examples of such mechanisms currently used are *Public-Private Key Pairs* and *Identity Certification*.

Reliability. *Reliability* concerns the consistency of communication data. There are three effects which generate the meta-effect *Reliability*, namely, *Data Correctness*, *Completeness*, and *Order Preservation*. The current Internet uses various mechanisms to assure the reliability of data like *Data Retransmission*, *Data Acknowledgment*, *Checksum*. The mechanisms *Data Acknowledgment* and *Checksum* mechanisms use sequence number and checksum fields of TCP header respectively.

Traffic Management. *Traffic Management* is the process of monitoring and controlling the communication traffic in order to avoid undesired situations and overhead on the communication link which can result in slow processing, delay, performance decline, or traffic congestion. The meta-effect *Traffic Management* contains the three effects *Traffic Information Collection*, *Congestion Control*, and *Flow Control*. In the existing Internet, the window field of the TCP header is being used for flow control and ECN field of IP header for *Congestion Control*.

Efficiency. *Efficiency* refers to different communication performance metrics such as, bandwidth, processing power, buffer size, etc. These parameters play an important role on the communication quality.

Security. *Security* refers to protection of information from theft or tampering. There are various ways to corrupt the data such as, appending data, alteration of existing data, or deletion of data. The meta-effect *Security* comprises the three effects *Confidentiality*, *Integrity* and *Availability*. TCP uses the checksum and sequence number fields of the TCP header to cover *Integrity*.

6. Combination of mechanisms

A network architecture is described as a 5-tuple $(\mathcal{E}, \mathcal{M}, \mathcal{M}_f, P, R)$.

\mathcal{E} and \mathcal{M} are sets of all available effects and mechanisms. For example \mathcal{E} contains *Completeness* and *Order Preservation* and \mathcal{M} contains *Sequence Numbers*, *Acknowledgments*

4. Secure socket layer

and *Header Checksums*. $\mathcal{M}_f \subseteq \mathcal{M}$ is the set of mechanisms that represent lower network layers and therefore it is fixed.

P is a function $P : \mathcal{M} \rightarrow 2^{\mathcal{E}}$ that gives for each mechanism a set of effects that it provides. In the example *Sequence Numbers* provides *Order Preservation* and *Acknowledgments* provides both *Order Preservation* and *Completeness*.

R is a function $R : \mathcal{M} \rightarrow 2^{\mathcal{M}}$ that gives for each mechanism a set of other mechanisms that it requires. In the example both mechanisms require the mechanism *Header Checksum* to work properly.

For simplicity the functions

$$P^*(M \subseteq \mathcal{M}) = \bigcup_{m \in M} P(m)$$

and

$$R^*(M \subseteq \mathcal{M}) = \bigcup_{m \in M} R(m)$$

are defined.

A given set of mechanisms $M \subseteq \mathcal{M}$ is consistent iff all requirements are met:

$$\text{consistent}(M \subseteq \mathcal{M}) = R^*(M) \subseteq P^*(M) \cup M$$

In the example that means, that using *Sequence Numbers* alone is consistent since that mechanism requires the mechanism *Header Checksum* which is not present. Using a set of *Sequence Numbers* together with *Header Checksum* is consistent.

To compare different consistent sets of mechanisms a cost-function is used. C is a function $C : 2^{\mathcal{M}} \rightarrow \mathbb{R}$ that values the combined costs of a set of mechanisms. Since different types of costs “add up” differently (e.g. latencies sum up but loss ratios are inverse multiplicative) and the factors with which these types of costs are combined depend on the needs of the application layer, the inner workings of that function are outside of the scope of this model.

To improve the model meta-effects – effects entirely composed of other effects and meta-effects – is introduced. \mathcal{E}_M is the set of meta-effects and *ComposedOf* is a function $\text{ComposedOf} : \mathcal{E}_M \rightarrow 2^{\mathcal{E} \cup \mathcal{E}_M}$ that gives for each meta-effect a set of effects and other meta-effects that it is composed of.

$$\text{ComposedOf}^*(X \subseteq \mathcal{E}_M) = \bigcup_{x \in X} \text{ComposedOf}(x)$$

is defined for simplicity and

$$\begin{aligned} \text{ComposedOf}'(X \subseteq \mathcal{E}_M) &= \text{ComposedOf}^*(X) \cup \mathcal{E} \\ &\cap \text{ComposedOf}'(\text{ComposedOf}^*(X) \cup \mathcal{E}_M) \end{aligned}$$

is the transitive hull of *ComposedOf* that only contains effects as result. With this extension to the model meta-effects like *Reliability* can be constructed with the effects *Data Correctness*, *Completeness*, *Order Preservation*.

6.1. Selection of Mechanisms

Given a cost-function $Cost$ and a set of desired effects D the function

$$S(D, Cost) = \arg \min_{M \subseteq \mathcal{A}} Cost(M)$$

with

$$\mathcal{A} = \{M \subseteq \mathcal{M} \mid \mathcal{M}_f \subseteq M \wedge \text{consistent}(M) \wedge D \subseteq P^*(M)\}$$

returns the best set of mechanisms to satisfy the requirements. For example if $D = \{\text{OrderPreservation}\}$ is desired and $\mathcal{M}_f = \{\text{Ethernet}, \text{HeaderChecksum}\}$ is given, the evaluation of the function S above returns *Sequence Numbers* (assumed $Cost(\{\text{SequenceNumbers}\}) < Cost(\{\text{Acknowledgments}\})$).

Calculating the result of $S(D, Cost)$ is a task of optimization. Many optimization algorithms have been presented by different research groups and are outside the scope of this model, so only a very basic algorithm will be introduced.

The number of consistent mechanism sets will surely be very large as mechanisms are fine-grained, so an algorithm to select mechanisms must be able to handle that.

A simple approach would be:

- 1) Select mechanisms to provide the requested effects. The base technology already provides some mechanisms \mathcal{M}_f . This will yield a number of different premature solutions. At this step the premature solutions are not checked for consistency, they provide the requested effects, but their requirements might be missing.
- 2) Create consistent solutions and rate them. Mechanisms are added to the premature solutions to make them consistent. After this step a number of consistent solutions is known and compared using the cost function.
- 3) Improve solutions with additional mechanisms. In this final step the best solutions are improved by adding mechanisms to them. The goal is to improve the cost of the solution without losing the consistency.

Another possible solution is to have a pre-calculated list of solutions and to adapt them in a predefined way. In the 4WARD project those pre-calculated solutions are called “netlets”. Or maybe multiple heuristics can calculate solutions concurrently and then the best one is chosen.

6.2. Mechanism dependencies

While identifying mechanisms certain dependencies become obvious. Those dependencies exist when one mechanism uses a value created by another mechanism or when a mechanism relies on an effect provided by another mechanism. In those cases the dependencies also define the order in which the mechanisms handle data. Which kinds of dependencies exist and what actions and constraints they require is the scope of future research.

7. Future Work

In this work we have presented a concept to describe protocols or micro-protocols using effects and mechanisms. Using effects allows abstract descriptions which represent the needs of users and thus are appropriate to describe services. Kinds of dependencies between effects and specific mechanisms have been outlined. Such dependencies must be taken into account for the composition of complex protocols. Several possible effects have been presented, which were derived from TCP/IP as an example. Finally a model for the composition of protocols has been presented which is only based on the description of effects and mechanisms.

The goal of this work is to foster loose coupling of micro-protocols, i.e. mechanisms. Effects allow describing mechanisms independently of internal details, implementation, and independently of other mechanisms. It has been shown that in principle it is possible to compose complex protocols based on these individual descriptions, enabling flexible network architectures.

We already have a prototype, that enables micro-protocols to interact indirectly via messages, notifications and generative communication. The presented approach will be used to extend the prototype by dynamical composition of protocols. Therefore the description of mechanisms and effects will be refined and formalized. For verification purposes we aim to implement the mechanisms found in TCP/IP.

References

- [1] D. Clark, et al: New Arch: Future Generation Internet Architecture. Final Technical Report <http://www.isi.edu/newarch/>
- [2] Roscoe T.: The end of Internet architecture. In *Proceedings of the fifth workshop on hot topics in networks (HotNets-V)*, Irvine, USA, November 2006.
- [3] IEEE Std. 1471-2000 *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems - Description*.
- [4] OASIS Reference Model for Service Oriented Architecture 1.0, Official OASIS Standard, Oct. 12, 2006
- [5] T. Erl: Service-Oriented Architecture Concepts, Technology, and Design. *Prentice Hall, 2005*
- [6] B. Reuther J. Götze: An Approach for an Evolvable Future Internet Architecture. *1st Workshop, New Trends in Service & Networking Architectures*, November 2008.
- [7] B. Reuther, P. Müller: Future Internet Architecture - A Service Oriented Approach. In *it - Information Technology Jahrgang 50 (2008) Heft 6*, S. 383-389, Oldenbourg Verlag, Muenchen, 2008.
- [8] B. Reuther, D. Henrici: A model for service-oriented communication systems (Journal Version). In *Journal of Systems Architecture*, June 2008.
- [9] S. W. O'Malley, L. L. Peterson: A Dynamic Network Architecture. In *ACM Transactions on Computer Systems*, Vol 10, No 2, May 1992, Pages 110-143.
- [10] M. Zitterbart, B. Stiller, A. N. Tantawy: A Model for Flexible High-Performance Communication Subsystems. In: *IEEE Journal on Selected Areas in Communications*, Vol. 11, No. 4, May 1993
- [11] T. Plagemann, B. Plattner, M. Vogt, T. Walter, "Modules as Building Blocks for Protocol Configuration", *Proceedings International Conference on Network Protocols (ICNP'93)*, San Francisco, USA, October 1993, pp. 106-115.
- [12] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, G.J. Minden: A survey of active network research. In: *IEEE Communications Magazine*, Jan 1997, Volume: 35, Issue: 1
- [13] L. Peterson, T. Anderson, D. Blumenthal, D. Casey, D. Clark, D. Estrin, J. Evans, D. Raychaudhuri, M. Reiter, J. Rexford, S. Shenker, and J. Wroclawski: GENI design principles. In: *IEEE Computer Magazine*, 2006
- [14] Future INternet Design (FIND) <http://www.nsf.gov/pubs/2007/nsf07507/nsf07507.htm>.
- [15] Cleanslate <http://cleanslate.stanford.edu/>
- [16] Future Internet Research & Experimentation (FIRE) <http://cordis.europa.eu/fp7/ict/fire/>
- [17] R. Braden, T. Faber, M. Handley: From Protocol Stack to Protocol Heap - Role-Based Architecture. In: *Proceedings of the First Workshop on Hot Topics in Networking (Hotnets-I)*, *ACM SIGCOMM*, Princeton, NJ., October 2002.
- [18] R. Dutta, G.N. Rouskas, I. Baldine, A. Bragg, D. Stevenson: The SILO Architecture for Services Integration, control, and Optimization for the Future Internet. In: *IEEE International Conference on Communications*, ICC apos 2007.
- [19] J.D. Touch, Y.S. Wang, V. Pingali: A Recursive Network Architecture. *ISI Technical Report ISI-TR-2006-626*, December 2006.
- [20] ANA: Autonomic Network Architecture <http://www.ana-project.org/>
- [21] 4WARD <http://www.4ward-project.eu/>
- [22] L. Völker, D. Martin, I. El Khayat, C. Werle, M. Zitterbart: An Architecture for Concurrent Future Networks *2nd GI/ITG KuVS Workshop on The Future Internet*, GI/ITG Kommunikation und Verteilte Systeme, Karlsruhe, Germany, Nov 2008