Designing and Running Concurrent Future Networks

Denis Martin, Helge Backhaus, Lars Völker, Hans Wippel,
Peter Baumung, Benjamin Behringer, Martina Zitterbart

Institut für Telematik, Universität Karlsruhe (TH), Germany

I. Introduction

In the future, many new services can be expected to be running on a multitude of devices and networks with diverse requirements. For example, in sensor-actuator networks energy awareness is important; in car-2-car communication one possible goal is reliable low delay data dissemination of critical traffic events; and in inter-planetary networks delay-tolerant delivery of data is one of the main issues. With network virtualization, one could run different networks in parallel that are tailored to the individual needs of the application. It also allows for an easy instantiation of new network architectures.

In the proposed demo, we will show how our concepts support the rapid development of application-tailored communication protocols for such network architectures and their easy deployment within a run-time environment. This way, the development of innovative services is no longer limited to the application, but can also easily happen further down the protocol stack.

The demo features a graphical design tool supporting our design process for protocol development. Using this tool, a simple protocol will be created, which will be subsequently instantiated on nodes running the Node Architecture in a demo-network. The Node Architecture, developed within the 4WARD project[1], is our run-time environment for so-called *Netlets*, which can be roughly described as containers for protocol stacks. The design tool is used to create Netlets by composing basic protocol building blocks.

Using a simple video application, we will show the impact of degrading network conditions that will lead to a deteriorated service quality. This would require further measures to be taken either in the application or in the communication protocol in order to cope with the changed situation. Rather than adapting the application, we will show how we can also re-design the simple Netlet used before to enhance it with further protocol mechanisms. The enhanced Netlet will then be deployed on the Node Architecture and the selection process will decide to use the new Netlet since it provides a more robust data delivery.

The key features presented in our demo include:
- Rapid creation of composed protocols (Netlets) using a graphical design tool.

- Reuse of existing protocol building blocks.
- Automatically selecting an appropriate Netlet based on application requirements at end nodes.

In the following sections we will briefly describe the concepts the demo is based on.

II. Netlets and Node Architecture

Figure 1 presents a simplified version of the Node Architecture. A detailed description of its concepts can be found in [1].
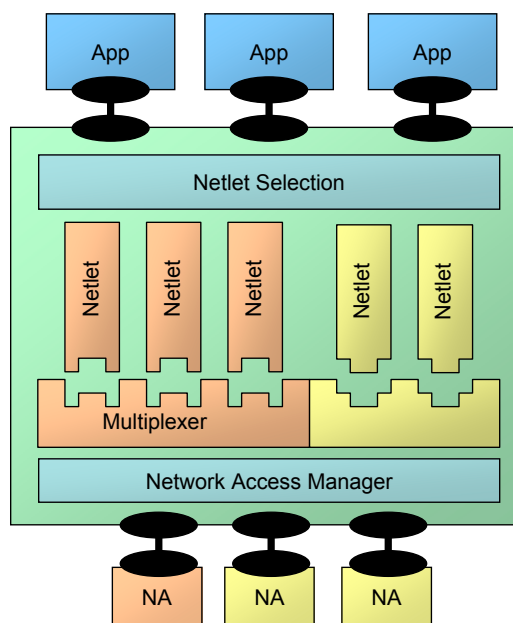

Figure 1. Simplified Node Architecture

The Node Architecture describes a possible design of end-nodes supporting a multitude of current and future protocol stacks. Those protocol stacks are encapsulated in so-called *Netlets*. Within the Node Architecture, Netlets are considered as black-boxes. Thus, they can be created by just writing code, following a protocol composition approach, and/or using a design tool with partial code generation.

When running multiple Netlets in the same network, they need a basic common understanding of, e.g., the data unit format or addressing scheme. The Netlet *Multiplexer* uses

this basic set of network invariants to (de-)multiplex data streams. Since those invariants may differ from network to network (e. g., sensor networks, backbone networks, delay-tolerant networks etc.), multiple Multiplexers are allowed to run in parallel.

In order to communicate through the Node Architecture, applications need to define their communication requirements. Based on those (and the requirements defined by user and/or administrator policies), the *Netlet Selection* component chooses an appropriate Netlet, taking the underlying network's properties into account. This selection process is described in detail in [2] and will also be shown as part of the demonstration.

As an abstraction of the connectivity to the network, the *Network Access* (NA) is used. Essentially, it is similar to today's network interface, but it may provide more elaborate information about the underlying physical or virtual network, and it may trigger events on network property changes. Connecting the Netlet Multiplexer to appropriate Network Accesses is handled by the *Network Access Manager*.

Our Node Architecture prototype can run on Linux as well as inside the OMNeT++ network simulator. The implementation provides a framework for easy protocol development and deployment. It will be used for other concepts developed within 4WARD and other projects.

III. DESIGN TOOL

We aim at a close inter-connection between design-time tools and the run-time environment to eventually come up with an integrated solution for rapid protocol development. Therefore, we are also looking into tool-supported design of protocols and we envision a complete life-cycle process for protocol development and deployment as proposed in [3].
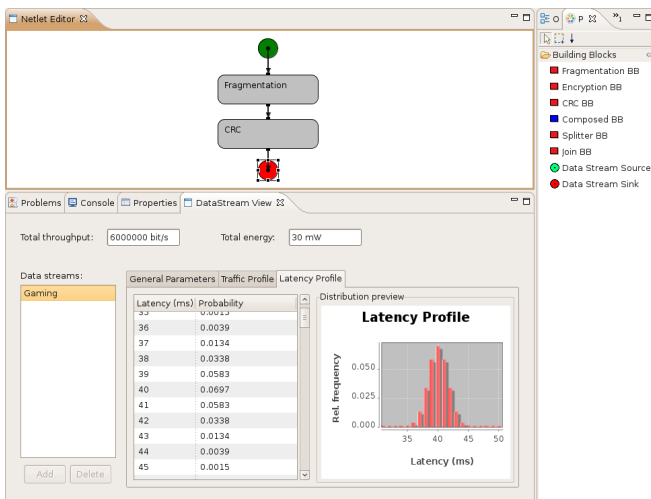

Figure 2. Property Aggregation Tool

The design tool features reuse of basic building blocks for protocols that are composed in order to create new Netlets. Such building blocks provide, for instance, fragmentation of data units, or the computation of an CRC checksum. The

protocol composition tool was implemented as an Eclipse plug-in based on the GEF framework. Its main feature is the automated aggregation of building block properties (e. g., added delay due to processing, energy consumption, etc.) as described in [2]. This allows the protocol designer to estimate how a composed protocol will act and if constraints are met early on, i. e. before implementation and simulation. It also allows him to compare the estimated performance with other, existing solutions, before he actually deploys it. Figure 2 shows the resulting latency profile of a simple example Netlet consisting of a fragmentation and a CRC checksum block.

During the demo, the design tool is used to create an XML file with instructions on how to compose a Netlet. Based on this XML file, the respective Netlet is automatically built from pre-existing building blocks. Those generated Netlets are then instantiated within the Node Architecture and used subsequently without the need to adapt the application.

IV. DEMO REQUIREMENTS

In order to demonstrate the introduced concepts and their features, we will need:

- Tables with about 2-3 m^2 in total to accommodate 3 to 4 laptops.
- Poster space to illustrate internal details of the demo.
- One or more power outlets.

The setup will display 3 interconnected laptops: Two end systems communicating over a simulated network on the third laptop, all running our Node Architecture.

V. RELATED WORK

Regarding the composition of protocols using smaller building blocks, various approaches exist: in the context of Future Internet research [4]–[6], as well as in the past in the context of high-speed transmission protocols [7]–[10]. While some of those approaches can handle run-time composition to a certain extent, all of them have made (partial) composition decisions at design-time in order to tackle the complexity of the problem (realized, e. g., in SILO's ontology, or in RNA's meta-protocol graph). We, in our work, focus completely on design-time composition decisions.

For modelling of protocols, well-known formal description languages that already have been applied to networking include the Specification and Description Language (SDL) [11], Extended State Transition Language (Estelle) [12], and Language Of Temporal Ordering Specifications (LOTOS) [13]. Some newer examples include State Chart XML [14], UML [15], or Cosmogol [16]. An innovative approach that combines SDL with model-driven development methods is described in [17]. Those languages, however, are developed to describe protocol *behavior*, which has been proven difficult, at least with respect to existing protocols like TCP.

In addition, tools like OPNET Modeler have some visual modeling capabilities for the implementation of protocols and topology design, which are mainly used for simulations. Clack [18] is a noteworthy visualization of network traffic flows through a virtual topology and a composed network

stack used for teaching. We, however, extend these visual approaches in order to support an integrated tool chain from visual design towards deployment of network protocols in the most flexible way. The envisioned iterative process of doing so is described in [3].

REFERENCES

[1] L. Völker, D. Martin, I. El Khayat, C. Werle, and M. Zitterbart, "A Node Architecture for 1000 Future Networks", in *Proceedings of the International Workshop on the Network of the Future 2009.* Dresden, Germany: IEEE, Jun. 2009.

[2] L. Völker, D. Martin, C. Werle, M. Zitterbart, and I. El Khayat, "Selecting Concurrent Network Architectures at Runtime", in *Proceedings of the IEEE International Conference on Communications (ICC 2009).* Dresden, Germany: IEEE Computer Society, Jun. 2009.

[3] L. Völker, D. Martin, T. Rohrberg, H. Backhaus, P. Baumung, H. Wippel, and M. Zitterbart, "Design Process and Development Tools for Concurrent Future Networks", in *3rd GI/ITG KuVS Workshop on The Future Internet*, Munich, Germany, May 2009.

[4] J. D. Touch, Y.-S. Wang, and V. Pingali, "A Recursive Network Architecture", ISI, Tech. Rep., Oct 2006, iSI-TR-2006-626.

[5] R. Dutta, G. N. Rouskas, I. Baldine, A. Bragg, and D. Stevenson, "The SILO Architecture for Services Integration, controL, and Optimization for the Future Internet", in *Proc. IEEE International Conference on Communications ICC '07*, G. N. Rouskas, Ed., Glasgow, Scotland, 2007, pp. 1899–1904.

[6] M. Sifalakis, A. Louca, A. Mauthe, L. Peluso, and T. Zseby, "A functional composition framework for autonomic network architectures", in *Proc. IEEE Network Operations and Management Symposium Workshops NOMS Workshops 2008*, 7–11 April 2008, pp. 328–334.

[7] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols", *SIGCOMM Comput. Commun. Rev.*, vol. 20, no. 4, pp. 200–208, 1990.

[8] N. C. Hutchinson and L. L. Peterson, "The x-kernel: An architecture for implementing network protocols", *IEEE Trans. Softw. Eng.*, vol. 17, no. 1, pp. 64–76, 1991.

[9] M. Zitterbart, B. Stiller, and A. Tantawy, "A model for flexible high-performance communication subsystems", *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 4, pp. 507–518, May 1993.

[10] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek, "The click modular router", *SIGOPS Oper. Syst. Rev.*, vol. 33, no. 5, pp. 217–231, 1999.

[11] *Specification and Description Language (SDL)*, International Telecommunication Union (ITU) ITU-T Recommendation Z.100, Aug 2002.

[12] S. Budkowski and P. Dembinski, "An introduction to Estelle: a specification language for distributed systems", *Computer Networks and ISDN Systems*, vol. 14, no. 1, pp. 3–23, 1987.

[13] M. D. P.H.J. van Eijk, C.A. Vissers, *Formal Description Technique Lotos: Results of the Esprit Sedos Project*, P. Van Eijk and Michel Diaz, Ed. New York, NY, USA: Elsevier Science Inc., 1989.

[14] *State Chart XML (SCXML): State Machine Notation for Control Abstraction*, World Wide Web Consortium (W3C) Std., May 2008.

[15] *Unified Modeling Language*, Object Management Group (OMG) Std., Rev. 2.1.2, Nov 2007.

[16] S. Bortzmeyer, "Cosmogol: a language to describe finite state machines", Nov 2006, expired IETF Draft.

[17] T. Kuhn, R. Gotzhein, and C. Webel, "Model-Driven Development with SDL - Process, Tools, and Experiences", *Model Driven Engineering Languages and Systems*, vol. 4199/2006, pp. 83–97, 2006.

[18] D. Wendlandt, M. Casado, P. Tarjan, and N. McKeown, "The clack graphical router: visualizing network software", in *SoftVis '06: Proceedings of the 2006 ACM symposium on Software visualization.* New York, NY, USA: ACM, 2006, pp. 7–15.

[19] G-Lab Project. http://www.german-lab.de/.