Forum "Math-for-Industry" 2015
October 26-30, 2015, Ito Campus, Kyushu University, Fukuoka, Japan

Using Process Algebra to Design Better Protocols

Peter Höfner
NICTA[1] and UNSW, Sydney, Australia

"Despite the maturity of formal description languages and formal methods for analyzing them, the description of real protocols is still overwhelmingly informal. The consequences of informal protocol description drag down industrial productivity and impede research progress". Pamela Zave [18]

1. Designing Protocols: State of The Art

Protocols are mainly specified in natural languages, such as English, without presenting a formal specification or some sort of pseudo code. At first glance this seems to be an advantage: everybody can easily read and understand the specification, and hence, the protocol is easy to implement. However, looking at contemporary protocol developments more closely, it turns out that natural languages are not proper specification languages at all. They may be easy to understand, but this comes at a price.

(1) *Specifications are (excessively) long.* The description of the Session Initiation Protocol [15], for example, is 268 pages long (and is not even self-contained); the IEEE Std 802.11™-2012 [9] standard, which contains a set of media access control (MAC) and physical layer (PHY) specifications for wireless networks, is $2,793$ pages long.
(2) *Specifications are ambiguous.* It is hard—maybe impossible—to write precise and unambiguous specifications using natural languages only. Ambiguities in the Ad hoc On-Demand Vector (AODV) protocol [12], for example, yielded 5 open-source implementations to behave differently, although all following the standard closely. [7]
(3) *Specifications are underspecified, and erroneous.* A famous example is the Border Gateway Protocol (BGP) [14], which is provably incorrect. [17]

These conclusions are neither new nor surprising, and documented in several research papers, e.g. [18] or [16, Chap. 9]. It is the purpose of this abstract to provide further evidence that formal methods in general and process algebra in particular can overcome these problems. They provide powerful tools that help to analyse and evaluate protocols, already during the design phase. I will illustrate this by a formal analysis of AODV [12], a routing protocol currently standardised by the IETF MANET working group. I will report how a combination of pen-and-paper analysis, model-checking and interactive theorem proving has helped to carry out the analysis. This case study shows (again) that formal methods are mature enough to support protocol design from the beginning. It is my belief that the use of formal methods could have found and prevented limitations in AODV-like protocols as reported in [10].

2. Formal Specification Languages

I believe that formal specification languages and analysis techniques are now able to capture the full syntax and semantics of reasonably rich protocols. They are an indispensable augmentation to natural language, both for specification and analysis.

[1]NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

Even when formal analysis is not the final aim, the use of formal languages is useful: they are unambiguous, reduce significantly the number of misunderstandings, and clarify the overall structure. By this, they almost always avoid underspecifications. Obviously, formal specification languages cannot prevent errors a priori, but they will make them less likely. If no formal analysis is required, it does not really matter which formalism is used. The choice of formal specification languages is numerous: it ranges from timed automata, which offer tool support via model checking (e.g. [3]), via the inductive approach, which offers interactive theorem proving support [11], to algebraic characterisations such as semirings (e.g. [8]) and process algebra (e.g. [5]). For our case study (see below), process algebra was chosen as specification language. It has the advantage that it is closely related to programming languages, and hence specifications are easy to understand by software engineers as well, not only by theoreticians.

3. Case Study: The AODV Routing Protocol

Together with my colleagues R. van Glabbeek, M. Portmann, W.L. Tan, A. McIver and A. Fehnker, I used the process algebra AWN [5] for wireless mesh networks to obtain the first rigorous formalisation of the specification of the AODV routing protocol [12]. In contrast to the 30+ pages of the ambiguous RFC, the created precise, yet very readable specification consists of roughly 200 lines. An analysis of this specification revealed that under a plausible interpretation of the original specification of AODV, the protocol admits routing loops [7]; this is in direct contradiction with popular belief, the promises of the AODV specification, and the main paper on AODV [13] (with over 13000 citations). However, we also proved loop-freedom of AODV under a subtly different interpretation of the original specification. [6]

AWN was initially developed for wireless networks, and has therefore in-built support for node mobility, broadcast/multicast communication etc. However, AWN allows modelling of any type of communicating concurrent processes, and can be used for a wide range of networks and protocols. The syntax of the AWN language is simple and reads much like a programming language, but it is implementation independent and has all the required properties to be able to formally reason about protocol and network properties, and to provide mathematically rigorous proofs. It has been combined with the model checker Uppaal [4] and with the proof assistant Isabelle/HOL [1, 2].

4. Looking Ahead

State-of-the-art formal description languages can be used to specify and analyse rather complicated protocols. To achieve more automation in the analysis, they often offer tool support, such as model checking. So, the question remains why these methods are not used in industrial production? I believe that three ingredients are still missing: (1) *Better (easy to use) tool support*: better tools and faster computers allow more and more automation. However, the use of tools often requires special knowledge (how to use the tool) or a special input format (e.g. timed automata). (2) *Code generation:* it is often believed that the combination of formal specification followed by implementation requires more time (and hence more money) than just implementing the protocol straight away. If entire (or at least parts of) implementations could be generated out of formal specifications automatically, one could gain even more advantages from formal methods. (3) *Training:* to use formal methods, engineers working in industry must be aware of them; this can only be achieved by training. Current research tackles the first two items, the last one may be the hardest to achieve.

References

[1] T. Bourke, R.J. van Glabbeek & P. Höfner (2014): *A mechanized proof of loop freedom of the (untimed) AODV routing protocol*. In F. Cassez & J.F. Raskin, editors: *Automated Technology for Verification and Analysis (ATVA'14)*, *Lecture Notes in Computer Science* 8837, Springer, pp. 47–63, doi:`10.1007/978-3-319-11936-6_5`.

[2] T. Bourke, R.J. van Glabbeek & P. Höfner (2014): *Showing Invariance Compositionally for a Process Algebra for Network Protocols*. In G. Klein & R. Gamboa, editors: *Interactive Theorem Proving (ITP'14)*, *Lecture Notes in Computer Science* 8558, Springer, pp. 144–159, doi:`10.1007/978-3-319-08970-6_10`.

[3] S. Chiyangwa & M. Kwiatkowska (2005): *A Timing Analysis of AODV*. In: *Formal Methods for Open Object-based Distributed Systems (FMOODS'05)*, *Lecture Notes in Computer Science* 3535, Springer, pp. 306–322, doi:`10.1007/11494881_20`.

[4] A. Fehnker, R.J. van Glabbeek, P. Höfner, A.K. McIver, M. Portmann & W.L. Tan (2012): *Automated Analysis of AODV using UPPAAL*. In C. Flanagan & B. König, editors: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS '12)*, *Lecture Notes in Computer Science* 7214, Springer, pp. 173–187, doi:`10.1007/978-3-642-28756-5_13`.

[5] A. Fehnker, R.J. van Glabbeek, P. Höfner, A.K. McIver, M. Portmann & W.L. Tan (2012): *A Process Algebra for Wireless Mesh Networks*. In H. Seidl, editor: *European Symposium on Programming (ESOP '12)*, *Lecture Notes in Computer Science* 7211, Springer, pp. 295–315, doi:`10.1007/978-3-642-28869-2_15`.

[6] A. Fehnker, R.J. van Glabbeek, P. Höfner, A.K. McIver, M. Portmann & W.L. Tan (2013): *A Process Algebra for Wireless Mesh Networks used for Modelling, Verifying and Analysing AODV*. Technical Report 5513, NICTA. Available at `http://arxiv.org/abs/1312.7645`.

[7] R.J. van Glabbeek, P. Höfner, W.L. Tan & M. Portmann (2013): *Sequence Numbers Do Not Guarantee Loop Freedom —AODV Can Yield Routing Loops—*. In: *Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '13)*, ACM Press, pp. 91–100, doi:`10.1145/2507924.2507943`.

[8] T.G. Griffin & J. Sobrinho (2005): *Metarouting*. *SIGCOMM Computer Communication Review* 35(4), pp. 1–12, doi:`10.1145/1090191.1080094`.

[9] IEEE (2011): *IEEE Standard for Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. (Revision of IEEE Std 802.11-2007).

[10] S. Miskovic & E.W. Knightly (2010): *Routing Primitives for Wireless Mesh Networks: Design, Analysis and Experiments*. In: *Conference on Information Communications (INFOCOM '10)*, IEEE, pp. 2793–2801, doi:`10.1109/INFCOM.2010.5462111`.

[11] L.C. Paulson (1998): *The Inductive Approach to Verifying Cryptographic Protocols*. *Computer Security* 6(1-2), pp. 85–128.

[12] C.E. Perkins, E.M. Belding-Royer & S. Das (2003): *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC 3561 (Experimental), Network Working Group.

[13] C.E. Perkins & E.M. Royer (1999): *Ad-hoc On-Demand Distance Vector Routing*. In: *Mobile Computing Systems and Applications (WMCSA '99)*, IEEE, pp. 90–100, doi:`10.1109/MCSA.1999.749281`.

[14] Y. Rekhter, T. Li & S. Hares (2006): *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271 (Draft Standard), Network Working Group (Errata Exist).

[15] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley & E. Schooler (2002): *SIP: Session Initiation Protocol*. RFC 4728 (PROPOSED STANDARD), Network Working Group (Errata Exist).

[16] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe & A. Roscoe (2010): *The Modelling and Analysis of Security Protocols: The CSP Approach*, (first published 2000) edition. Pearson Education.

[17] K. Varadhan, R. Govindan & D. Estrin (2000): *Persistent Route Oscillations in Inter-domain Routing*. *Computer Networks* 32(1), pp. 1–16, doi:`10.1016/S1389-1286(99)00108-5`.

[18] P. Zave (2011): *Experiences with Protocol Description*. In: *Rigorous Protocol Engineering (WRiPE' 11)*.