Validating Wireless Congestion Control and Reliability Protocols using ProB and Rodin

Jens Bendisposto, Michael Jastram, Michael Leuschel
Christian Lochert, Björn Scheuermann, Ingo Weigelt[1,2]

Computer Science Department
Heinrich Heine University
Düsseldorf, Germany

Abstract

Implicit hop-by-hop congestion control is a novel congestion control paradigm for wireless multihop networks. Implemented in the CXCC protocol, it has already proven its performance in simulations and measurements. Since CXCC makes extensive use of the properties of an inherently unreliable medium, it is, however, vitally necessary to validate the correctness of the protocol. Indeed, an early version of the CXCC protocol contained deadlocks. CXCC is complemented by an end-to-end reliability mechanism in the BarRel transport protocol. In combination, both protocols offer TCP-equivalent service in dynamic wireless multihop environments, including, e.g., route changes. BarRel relies on properties of CXCC. It therefore likewise deserves validation.
In this work we attempt to validate the CXCC and BarRel protocols using formal methods. To this end, we are developing various models in B and Event-B. We are using the ProB tool to animate and model check the formal models and the Rodin platform to formally prove correctness properties. In this paper we present first encouraging steps towards a full formal validation of the protocols.

Keywords: Protocol Validation, Wireless Networks, Congestion Control, End-to-End Reliability, CXCC, BarRel, B-Method, ProB, Rodin

1 Implicit Hop-by-Hop Congestion Control

In previous work we have introduced the Cooperative Cross-layer Congestion Control (CXCC) protocol and the Backpressure Reliability (BarRel) protocol. CXCC is a congestion control approach, and BarRel is the corresponding end-to-end reliability mechanism. Both protocols in combination provide TCP-equivalent end-to-end service. In this section, we briefly outline CXCC and BarRel. For a more detailed discussion, we refer the reader to [6, 7].

1.1 CXCC

CXCC is a cross-layer approach, encompassing essentially the MAC and transport layers. It uses implicitly obtained information for hop-by-hop backpressure conges-

[1] Email: {bendisposto,jasteam,leuschel,lochert,scheuermann}@cs.uni-duesseldorf.de

[2] The authors are grateful to the German Research Foundation (DFG) and the EU funded FP7 research project 214158 DEPLOY for partially funding this research.

tion control and single-hop reliability. The key concept is that, for each end-to-end connection, an intermediate node may only forward a packet towards the destination after its successor along the route has forwarded the previous one. This yields a backpressure mechanism which reacts very rapidly to varying medium conditions and effectively avoids excessive packet inflow into congested network areas.

The CXCC protocol realizes the implicit hop-by-hop congestion control principle by overhearing the medium after a transmission. No further packets for the same connection may be transmitted, until the packet has been forwarded further by the downstream node. The connection is "blocked" at the first node until the second node forwards the packet further on (see Fig. A.1). In such a setting, overhearing the transmission of the downstream node can serve a dual purpose: it constitutes an implicit acknowledgment, indicating the successful reception of the previous packet by the downstream node, and at the same time is a signal to unblock the connection, allowing the next packet to be transmitted.

Link layer acknowledgments as in IEEE 802.11 are, consequently, not necessary. Only at the last hop the packet is acknowledged explicitly. However, wireless communication is very error-prone. Transmissions may be lost, for instance, due to a collision. CXCC uses Request For Acknowledgment (RFA) packets in order to overcome such situations. An RFA is a small control packet, containing just enough information to identify the packet it refers to. Upon reception of an RFA, a node checks whether it already has received the respective data packet or not. It may then provide appropriate feedback, such that, if necessary, the transmission is repeated. For details see [7].

1.2 BarRel

The CXCC retransmission mechanisms allow to overcome individual transmission errors between adjacent nodes along the route. Because CXCC limits the number of packets per hop to one, there will, by construction, also be no packet loss due to buffer overflows. Hence, there will be no packet loss with CXCC as long as the route to the destination remains stable. In this situation, a source node can implicitly obtain information about successful packet deliveries to the destination node: there is at most one packet per hop in CXCC, and there is neither packet loss nor packet reodering in the network. Consequently, if the route to the destination is currently n hops long, after the $(i + n)$-th packet of a transmission has left the source node, the i-th packet must have arrived at the destination.

In BarRel, this is used in order to realize TCP-equivalent reliable end-to-end data transport without the need for a continuous stream of end-to-end acknowledgment traffic. In wireless multihop environments, such acknowledgment traffic is considered a major problem, because it is forwarded over the same wireless medium as the data traffic (in opposite direction), causing significant overhead and frequent collisions. Therefore, an ACK-free reliable transport protocol is highly desirable.

However, it may also happen that a wireless link fails permanently. Then, the above assumptions do not necessarily hold true, and packet loss *may* happen. If one node along the route is no longer reachable, the routing protocol used in the network will discover a new route. In order to guarantee end-to-end reliability also

in case of link breaks and re-routing, the BarRel mechanism steps into the breach. In these cases, a BarRel source node will go back to the first packet for which it could not yet confirm successful delivery, and retransmit from there.

The delivery of the last n packets of a packet burst can not be acknowledged as described above. BarRel therefore includes further mechanisms to confirm their arrival at the destination. One of them adds a number of *empty* packets at the end of the transmission. By the implicit delivery confirmation mechanism, when the n-th such "capacity refill" (CaRe) packet leaves the source node, the last data packet must have successfully arrived. This allows for a protocol operation without any oncoming end-to-end control traffic. For details see [6].

2 Validating CXCC and BarRel

The B-method [1] is a formal methodology for the systematic development of safety-critical software systems, based on the idea of refinement. Event-B (e. g., [4]) is the successor of the B-method, which is also suitable to model reactive systems. Both are supported by industrial-strength proving tools. Event-B is supported by the Rodin platform [2] which enables integrated editing and proving, i. e., reasoning mathematically about the preservation of system invariants. Our ProB tool [5] can be used to animate B and Event-B models, as well as validate temporal logic formulas via model checking.

In this paper, we decided to use the Event-B method to verify the correctness of BarRel and CXCC; in future work we also plan to investigate the use of the CSP process algebra (which is also supported by our tool ProB). Our goal is to validate the CXCC and BarRel protocols using animation and model checking, but also to try and develop a version of the algorithm which is correct by design. We also intend to evaluate if the B-Method and the tools ProB [5] and Rodin [2] are appropriate for the design of networks protocols. For the verification we developed two different sets of formal models. The first version was a set of models used for animation and model checking rather than for proof. The goal was to get familiar with the problems that arise from the domain. Also we wanted to gain confidence in the correctness of the protocols. The second model is used to obtain correctness by design, by refining a very abstract model towards the CXCC/BarRel protocols.

2.1 First version: models for animation

We specified CXCC and BarRel in two different models. The CXCC model contains a cycle free sequence of nodes, the route, for an end-to-end connection. The nodes are equipped with two buffers, one for incoming packets and one to buffer packets sent. The model can put packets into one end of the route and remove them from the other end. During transmission we can lose packets and acknowledgements and it is also possible that the implicit acknowledgment gets lost. The model does not yet cope with multiple packets on multiple routes. We also developed a graphical representation that could be used to demonstrate the protocol to domain experts who are not familiar with the mathematical notation. The absence of deadlocks was validated using ProB, as were several LTL formulas specifying correct delivery under certain fairness assumptions.

2.2 Second version: models for proof

A flow in a CXCC enabled network can be abstractly seen as a queue. The sender adds data packets to it and the receiver removes them. Another constraint resulting from CXCC and BarRel is that we can divide this queue into different contiguous sections. Each section contains only data-packets, CaRe-packets or duplicates of received data-packets. Furthermore CaRe-packets can only form the last section of the queue and duplicates precede data.

The first model m0 defines a generic queue, mainly modeled after [3]. Methods to enqueue, dequeue and delete the whole queue have been implemented. This queue will then subsequently be refined until the detailed behaviour of CXCC and BarRel is obtained.

m0 is then refined to m1, which maps the queue elements to links along the route holding data packets. Accordingly the maximum size of the queue is limited to the route length. The events enqueue and dequeue now represent sending and receiving data by the source and receiver. The route break event refines the delete event to additionally reset the counter used by the sender. As another step towards realistic network conditions, it also nondeterministically changes the route length. To implement the BarRel mechanism, a number of CaRe packets are appended to the source file. Their number is also changed by the route break event, since it must always correspond to the current route length. The events enqueue and dequeue are also refined by another two events sendCaRe and receiveCaRe, in order to handle transmissions and receptions of CaRe packets

The route break event in m1 removes all elements in the queue and resets the senders counter to retransmit exactly the lost packets. In the next refinement, m2, we remove a number of packets corresponding to the current route length. This reflects that the sender must assume the worst case. As a result, the receiver now needs to cope with duplicate packets arriving.

Fig. B.1 shows a ProB animation of the last model. As far as proving is concerned, m0 still contains three unproven proof obligation (POs) out of 46. Models m1 and m2 are fully proven; 48 of 61 and 18 of 25 POs were automatically proven.

References

[1] Abrial, J.-R., "The B-Book," Cambridge University Press, 1996.

[2] Abrial, J.-R., M. Butler and S. Hallerstede, *An open extensible tool environment for Event-B.*, in: *ICFEM '06*, LNCS **4260** (2006), pp. 588–605.

[3] Abrial, J.-R. and D. Cansell, *Formal construction of a non-blocking concurrent queue algorithm (a case study in atomicity)*, Journal of Universal Computer Science **11** (2005), pp. 744–770.

[4] Abrial, J.-R., D. Cansell and D. Méry, *Refinement and reachability in Event$_B$*, in: H. Treharne, S. King, M. C. Henson and S. A. Schneider, editors, *ZB '05*, LNCS **3455** (2005), pp. 222–241.

[5] Leuschel, M. and M. Butler, *ProB: A model checker for B*, in: K. Araki, S. Gnesi and D. Mandrioli, editors, *FME '03*, LNCS **2805** (2003), pp. 855–874.

[6] Scheuermann, B., "Reading Between the Packets – Implicit Feedback in Wireless Multihop Networks," Ph.D. thesis, Heinrich Heine University, Düsseldorf, Germany (2007).

[7] Scheuermann, B., C. Lochert and M. Mauve, *Implicit hop-by-hop congestion control in wireless multihop networks*, Elsevier Ad Hoc Networks **6** (2008), pp. 260–286.

A Figures

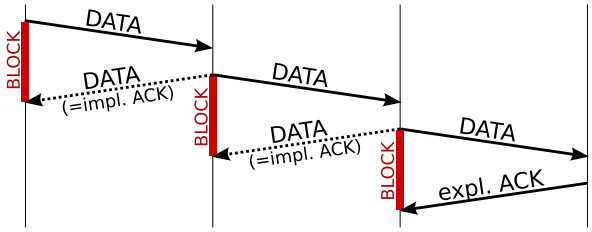

Fig. A.1. Packet forwarding in the CXCC protocol.

B Screenshots

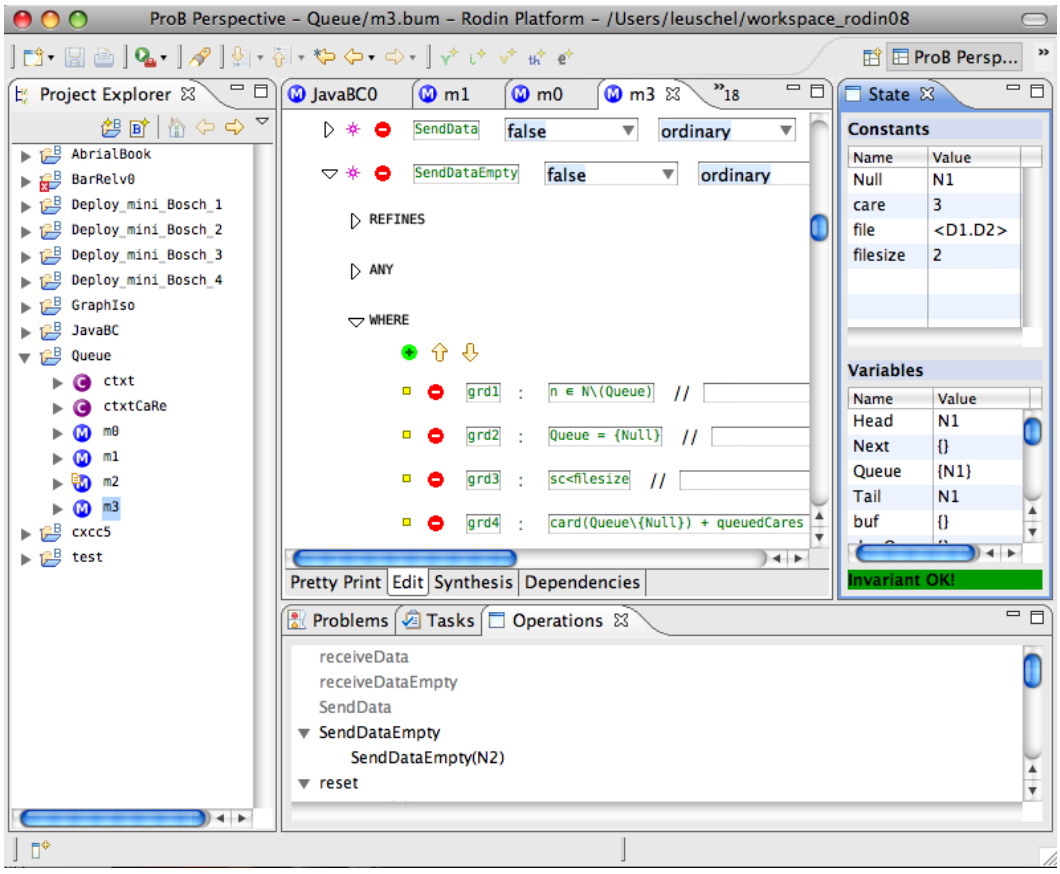

Fig. B.1. Third refinement animated with ProB for Rodin