Analysing the PGM Protocol with Uppaal *

Béatrice Bérard Patricia Bouyer† Antoine Petit

LSV – UMR 8643 CNRS & ENS de Cachan,
61 av. du Prés. Wilson, F-94235 Cachan Cedex, France
{berard,bouyer,petit}@lsv.ens-cachan.fr

Abstract

Pragmatic General Multicast (PGM) is a reliable multicast protocol, designed to minimize both the probability of negative acknowledgements (NAK) implosion and the load of the network due to retransmissions of lost packets. This protocol was presented to the Internet Engineering Task Force as an open reference specification.

In this paper, we focus on the main reliability property which PGM intends to guarantee: *a receiver either receives all data packets from transmissions and repairs or is able to detect unrecoverable data packet loss.*

We first propose a modelization of (a simplified version of) PGM *via* a network of timed automata. Using Uppaal model-checker, we then study the validity of the reliability property above, which turns out not to be always verified but to depend on the values of several parameters that we underscore.

1 Introduction

Since the introduction of timed automata (Alur and Dill (1990)), a lot of work has been devoted to both theoretical studies of timed models and practical issues for their analysis. Verification algorithms have been designed and implemented in so-called real-time model-checkers like HyTech (Henzinger et al. (1997)), Kronos (Bozga et al. (1998)) or Uppaal (Larsen et al. (1997)), with successful results for numerous case studies. In this paper, we propose the verification of two reliability properties for the multicast protocol PGM.

Reliable multicast protocols. Reliable multicast protocols are designed to enable distribution of information from multiple sources to multiple receivers, with reliability requirements. Examples of applications which may benefit from this technology include video broadcasts, data base replication or software downloads. Reliability in unicast protocols (like TCP) is usually achieved by positive acknowledgments (ACK) sent by the receiver to the source. Extending this principle to multicast protocols with a growing number of receivers may result in so-called ACK implosion. For this reason, the development of multicast protocols initially focused on eliminating ACKs, while keeping negative acknowledgements (NAK), invoked by receivers only when some packets are not received. However, multiple redundant NAKs can also be issued if packets are lost during periods of congestion. Besides, NAKs can be followed by redundant retransmissions.

PGM protocol. Pragmatic General Multicast (PGM) belongs to a second generation of reliable multicast protocols, designed to address the problems mentioned above: it is said to minimize both the probability of NAK implosion and the load of the network due to retransmissions of lost packets. The approach taken for buffer management resorts to timeouts at the source, with a new packet type called Source Path Message (SPM). This protocol was developed jointly by Cisco systems and TIBCO, and presented to the Internet Engineering Task Force as an open reference specification (Speakman et al. (2001)). It is currently supported as a technology preview, usually over IP, with which users may experiment, and it enters the longer-term standardization process.

*This work has been supported by the french project RNRT Calife.

†This work has been partly carried out while this author had a post-doctoral fellowship at BRICS, Aalborg University, Denmark

Contribution of the paper. In this paper, we focus on the main reliability property which PGM intends to guarantee (Speakman et al. (2001)): *a receiver either receives all data packets from transmissions and repairs or is able to detect unrecoverable data packet loss.* Section 2 presents the main features of PGM. In Sections 3 and 4, we describe how a simplified model of the protocol is built, *via* a network of timed automata. Section 5 is devoted to a detailed presentation of the verification process and Section 6 concludes the paper.

2 Description of PGM

We first give a brief description of the protocol, with a single source, as proposed in (Speakman et al. (2001)). Since we are only interested in reliability, we omit all mechanisms related to minimization of the load.

The source multicasts sequenced data packets called ODATA (for Original Data), within a transmit window, at a given rate. Those packets are transmitted through network elements, along some path of a distribution tree. If a receiver detects a missing packet from the expected sequence, it repeatedly unicasts a negative acknowledgement (NAK) to the last element on the path. This network element forwards the NAK to the source using the reverse path, and multicasts a NAK confirmation (NCF). The receiver stops sending the NAK upon reception of the NCF. The same operation is repeated in turn at each level of the path toward the source, including the source itself. Repairs (RDATA) must then be provided by the source or by a Designated Local Repairer (DLR). Since the reliability of the protocol mainly depends on the NAK transmissions, PGM defines a network-layer hop-by-hop procedure for reliable forwarding. A similar method is used for NCF multicasting.
In addition to this basic data transfer operation, SPMs (Source Path Messages) are sent by the source at a given rate, thus periodically interleaved with ODATA. Their purpose is twofold:

1. maintaining up-to-date neighbour information, to ensure that the return path for the NAKs is exactly the reverse of the forward path followed by ODATA and
2. delivering information on transmit resources to the receivers.

The architecture of a network on which PGM can be used is depicted on figure 1.

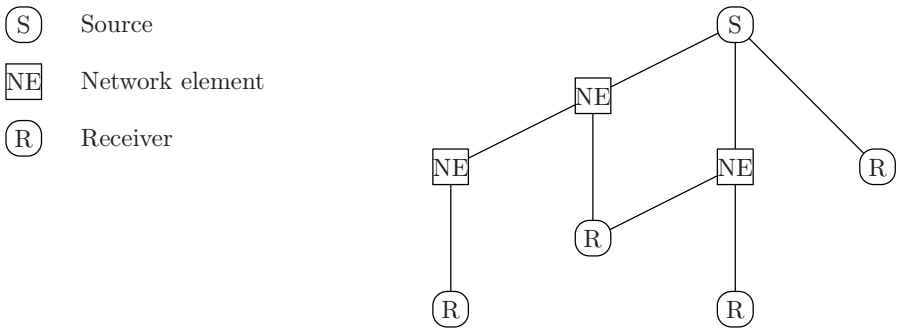

Figure 1: An overview of PGM

Transmit window. More precisely, assuming a constant data packet size, ODATA packets are ordered by the source in unit increments from a circular sequence number space from 0 to $2^{32}-1$, and buffer management relies on the transmit window maintained by the source:

- the left edge of this window, `txw_trail`, is the number of the oldest data packet available for repair,
- its right edge, `txw_lead`, is the number of the most recent data packet transmitted.

The window is considered empty if `txw_trail` = `txw_lead` + 1 and its size is bounded by $2^{16} - 1$. There is no fixed strategy for the source to advance the trailing edge of the window. The edges of the transmit window are contained in SPMs. A receiver also maintains a receive window, with edges `rxw_trail` and `rxw_lead`, which evolves according to the informations received from the source, either by data packets or SPMs.

Rates and priorities. The source must strictly prioritize sending of pending NCFs first, pending SPMs second, and only send ODATA or RDATA when no NCFs or SPMs are pending. The priority of RDATA versus ODATA is application dependent, with a possible sharing strategy. Before the source multicasts some RDATA upon reception of a NAK, there may be some bounded delay to wait for other NAKs.
There are two types of SPMs: in the presence of O/RDATA, ambient SPMs are transmitted by the source at a regular rate, while heartbeat SPMs are transmitted in the absence of data at a decaying rate, in order to maintain receive windows and assist early detection of lost data.

3 Modelling PGM with timed automata

In this section, we explain how our model of the protocol is built. Since the protocol involves timing constraints, the choice of a timed model-checker is mandatory. Besides, many discrete variables appear, with an array-like organization. This leads to the tool Uppaal (Bengtsson et al. (1998)), which offers a compact description language for this type of variables, thus making modelling easier.

3.1 Modelling with Uppaal

A complete presentation of the model handled by Uppaal can be found in many papers or surveys (see for example (Larsen et al. (1997); Bengtsson et al. (1998); Amnell et al. (2001); Bérard et al. (2001))). Timed automata in Uppaal are a modified version of the original ones from (Alur and Dill (1994)). We briefly recall here in an informal way the main characteristics of the model, illustrated in Section 4 below (see Remark 4.1).

In Uppaal, a system consists of a collection of timed automata, with binary synchronization: two components synchronize through channels with a sender/receiver syntax. For instance, on a given channel `c`, a *sender* emits a signal denoted by `c!` and a *receiver* synchronizes with the sender by the corresponding reception denoted by `c?`. These two actions are called *complementary actions*.
A timed automaton is a finite structure handling a finite set of variables. These variables are either *clocks*, which evolve synchronously with time, or bounded discrete integer variables. A location in an automaton can be labelled by clock conditions, called *invariants*, which must be satisfied as long as time elapses in this location. A *transition* of the automaton is decorated by three types of labels:

- a *guard* expressing a condition on the values of the variables, which must be satisfied for the transition to be fired,
- a synchronization *label* of the form `c!` or `c?`,
- a *clock reset* and an *update* of integer variables.

Note that each type of label is optional, an absence of synchronization meaning that the automaton performs an internal action.

A (global) configuration is of the form (ℓ, v) where ℓ is a location vector (indicating the current state in each component of the timed automata network) and v is a valuation of the variables, that is a function which assigns to each clock a real value and to each discrete variable an integer value. An execution in the network starts in initial locations of the different components with all the clocks and variables set to zero. The semantics of this model is expressed by moves between the configurations. Three types of moves can occur in the system: delay moves, internal moves and synchronized moves.