Exploring Properties of a Telecommunication Protocol with Message Delay Using Interactive Theorem Prover

Catherine Dubois[1], Olga Grinchtein[2](✉), Justin Pearson[3], and Mats Carlsson[4]

[1] ENSIIE, Samovar (UMR CNRS 5157), Évry, France
catherine.dubois@ensiie.fr
[2] Ericsson AB, Stockholm, Sweden
olga.grinchtein@ericsson.com
[3] Uppsala University, Uppsala, Sweden
justin.pearson@it.uu.se
[4] RISE SICS, Stockholm, Sweden
mats.carlsson@ri.se

Abstract. An important task of testing a telecommunication protocol consists in analysing logs. The goal of log analysis is to check that the timing and the content of transmitted messages comply with specification. In order to perform such checks, protocols can be described using a constraint modelling language. In this paper we focus on a complex protocol where some messages can be delayed. Simply introducing variables for possible delays for all messages in the constraint model can drastically increase the complexity of the problem. However, some delays can be calculated, but this calculation is difficult to do by hand and to justify. We present an industrial application of the Coq proof assistant to prove a property of a 4G protocol and validate a constraint model. By using interactive theorem proving we derived constraints for message delays of the protocol and found missing constraints in the initial model.

Keywords: Testing of telecommunication protocol
Constraint programming · Formal proof · Coq

1 Introduction

We presented in [11] a constraint model of a telecommunication protocol that broadcasts public warning messages [1]. The goal was to check that the message transmission conforms to specification by analysing logs. We used the constraint modelling language MiniZinc [15] to model the protocol and to find solutions that indicated errors in logs. We used this approach to analyze both real and generated logs. Since some messages of the protocol can be delayed, introducing a delay for every message increases the complexity of the model. However, it is possible to derive a formula for some delays, which simplifies the problem to be solved by a constraint solver. We manually derived the delays in [11], but we did not prove the correctness of the derivation.

In this work we use the Coq proof assistant [20] to derive and prove the formula for delays of some messages. By using Coq we found necessary assumptions that should be made on parameters of the constraint model. We also found missing constraints in [11]. By using Coq we are guaranteed that resulting calculations are correct. Furthermore because certain properties of the model had to be proved and derived it was more appropriate to use a proof assistant rather than a computer algebra system. The rest of the paper is structured as follows. Section 2 presents Constraint Programming and the Coq proof assistant. Section 3 is an overview of the telecommunication protocol that we analyse. Section 4 presents the constraint model on which our proofs in Coq are based. Section 5 presents a property of the protocol we explore and a new constraint model for delays. Section 6 describes proof steps in Coq.

2 Preliminaries

In this section we present very briefly, both constraint programming and the proof assistant Coq.

Constraint Programming [18] (CP) is a framework for modelling and solving combinatorial problems including verification and optimisation tasks. A constraint problem is specified as a set of *decision variables* that have to be assigned values so that the given constraints on these variables are satisfied, and optionally so that a given objective function is minimised or maximised. We use italic to distinguish *decision variables* from parameters in the constraints.

MiniZinc [15] is a constraint modelling language, which has gained popularity recently due to its high expressivity and large number of available solvers that support it. It also contains many useful modelling abstractions such as quantifiers, sets, arrays, and a rich set of global constraints. All the constraints presented in this paper are shown in a form that is very close to their MiniZinc version.

Coq [20] is an interactive proof assistant based on constructive type theory, more precisely, the calculus of inductive constructions. It also has a trustworthy kernel [2]. Coq allows the user to state theorems, write proofs that are verified according to the Curry-Howard isomorphism, and thus checking is reduced to type checking. It is also possible to write and verify algorithms. Proofs are written with the help of tactics (a.k.a. proof commands). Coq has many basic tactics, including tactics for unfolding definitions, but also more complex ones, e.g. doing arithmetic reasoning, or applying inductive proof schemes. Coq also provides a rich library of high-level tactics that automates many low level details. Coq has been used successfully in many projects of large scale such a formal proof of the four colour theorem [10] or the construction of an optimising compiler for C [14]. Interactive theorem provers differ from automatic theorem provers, such as SMT or SAT solvers, in that the user has to guide the tool to produce the proof.

3 Protocol Overview

Our case study is the Earthquake and Tsunami Warning System (ETWS) that is a part of the Public Warning System [1]. Its purpose is to broadcast emergency information to all the users in a certain area when earthquake or tsunami is imminent. We do not present all details of the protocol, but only the part that we used in the Coq development.

```
Write − ReplaceWarningRequest{
          WarningType : '0580'H
          rPer : 30
          nBR : 4
          WarningMessageContents : '41424344'H
}
```

Fig. 1. Warning message of combined type

There are three participants in the protocol that we consider: the network entity, a radio base station and the user equipments. By receiving a warning message from a network entity, the radio base station broadcasts paging messages and system information messages to the user equipments. In this work we focus on paging messages. Periodicity of paging messages depends on the type of warning message as shown in Figs. 2 and 4. Warning messages can be of three different kinds depending on its content: messages can be primary notifications, and/or secondary notifications. So the type of a warning message can be primary, secondary, or combined. A primary notification is a very simple message indicating a type of imminent danger, e.g. "earthquake", while a secondary notification message contains more detailed text data. Warning messages of combined type include both. Paging message is used to inform user equipment about the presence of primary notification and/or secondary notification.

In Fig. 1 is shown an example of the content of a warning message, where only information elements relevant to this work are included. The parameter `rPer` is in seconds and is used to calculate periodicity of paging messages of secondary notifications. The parameter `nBR` represents the number of paging messages of secondary notifications.

The periodicity of paging messages of primary notifications is equal to the default paging cycle `dPC` and the number of paging messages of primary notifications is `ndPC` that is configured in radio base station. The periodicity of paging messages of secondary notifications is a multiple of `dPC`.

Figures 2 and 4 illustrate interleaving of paging messages of primary and secondary notifications.

In Fig. 2 is shown the acquisition of paging messages by the user equipment after the radio base station receives a first warning message of primary type and then a warning message of combined type. The user equipment first reads

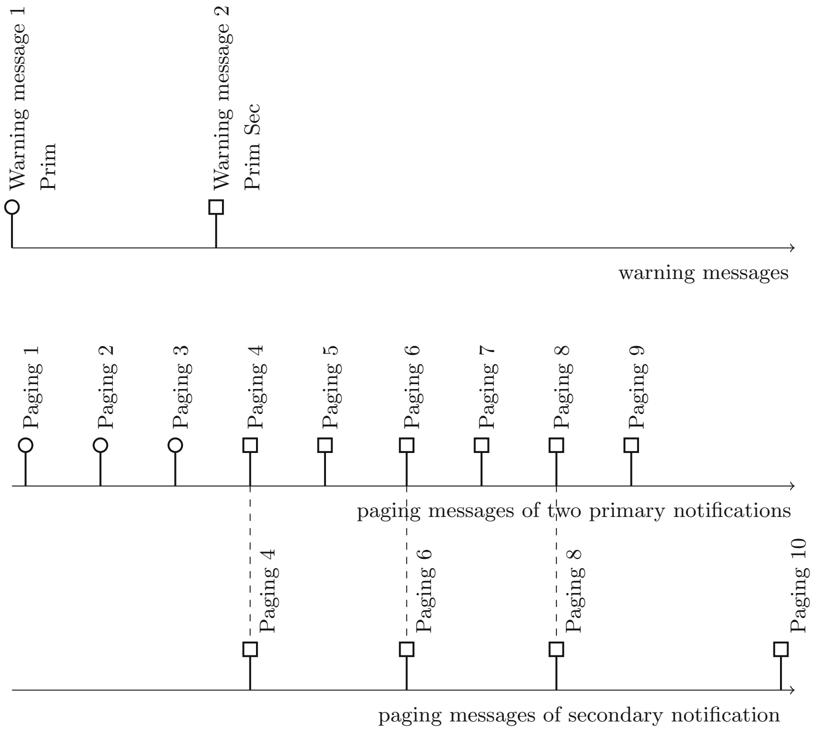

Fig. 2. An example of acquiring paging messages by user equipment transmitted by radio base station after receiving warning message of primary type and warning message of combined type. Different shapes on the top of the vertical lines represent different types of warning messages.

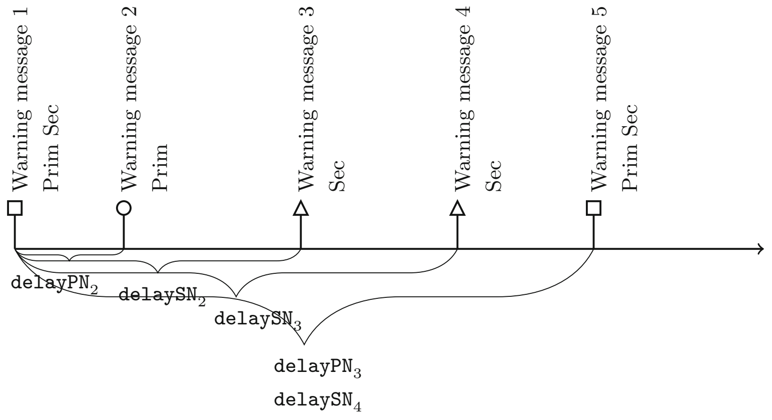

Fig. 3. Transmission of warning messages.

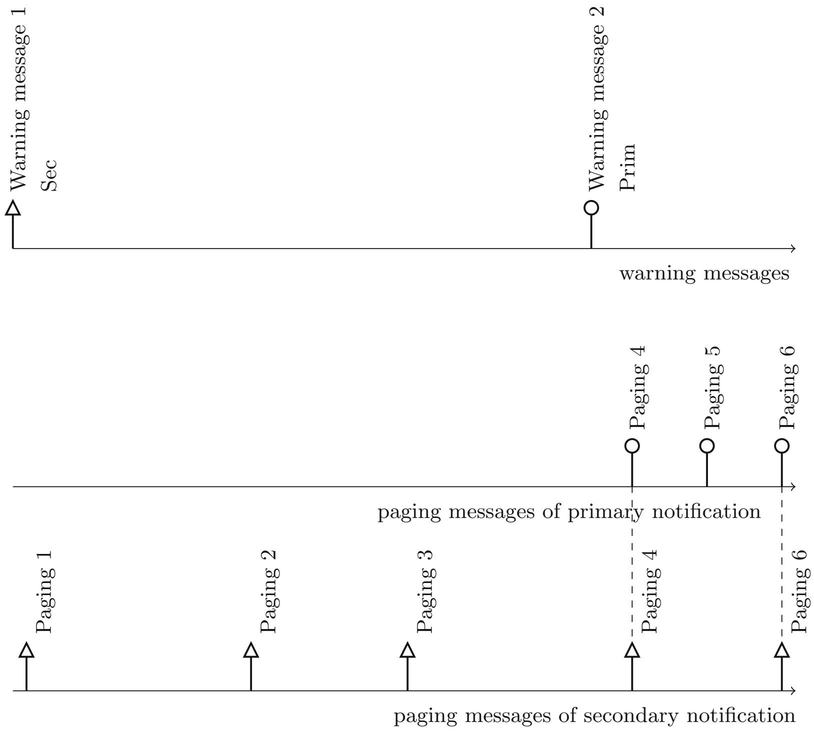

Fig. 4. An example of acquiring paging messages by user equipment transmitted by radio base station after receiving warning message of secondary type and warning message of primary type.

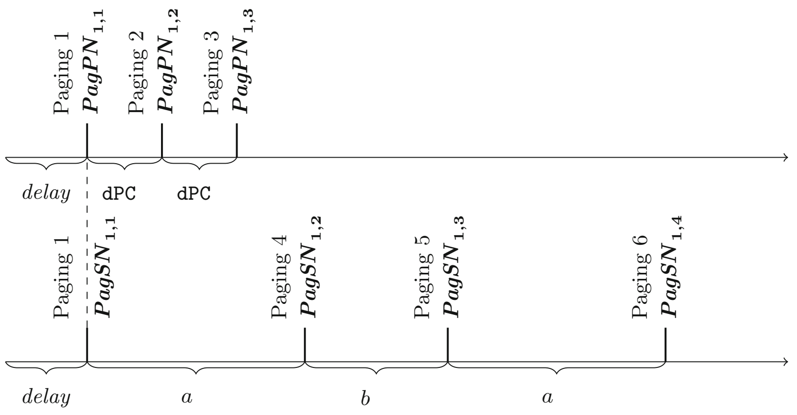
Paging 1 $\boldsymbol{PagPN}_{1,1}$
Paging 2 $\boldsymbol{PagPN}_{1,2}$
Paging 3 $\boldsymbol{PagPN}_{1,3}$
delay
dPC
dPC
Paging 1 $\boldsymbol{PagSN}_{1,1}$
Paging 4 $\boldsymbol{PagSN}_{1,2}$
Paging 5 $\boldsymbol{PagSN}_{1,3}$
Paging 6 $\boldsymbol{PagSN}_{1,4}$
delay
a
b
a

Fig. 5. An example of acquiring paging messages by user equipment transmitted by radio base station after receiving warning message of combined type.

three messages of primary notification corresponding to warning message 1. After warning message 2 is received by the radio base station, it starts to transmit paging messages of primary and secondary notifications, since warning message 2 has combined type. Figure 2 illustrates replacement of warning messages. If the radio base station receives a new warning message of primary type, while transmitting paging messages of primary notification of the previous warning message, then the radio base station starts to transmit paging messages of new primary notification. Similar replacement can occur for secondary notifications.

In Fig. 4 is shown the acquisition of paging messages by user equipment after the radio base station receives first warning message of secondary type and then warning message of primary type. The radio base station transmits paging messages of secondary type and after receiving warning message 2 it starts to transmit paging messages of primary type.

4 Constraint Model

In [11] we introduced a constraint model that had constraints on timestamps and content of messages broadcast by a radio base station, including paging messages. It is a discrete time model, since we deal with timestamps. The goal was to check that timing and content of messages in the logs comply with specification. Logs contain messages of 4 different types including paging messages. We used simulators for both network and user equipment entities, and our system under test is a radio base station. In this section we introduce a constraint model for paging messages in more details than in [11].

For each warning message we know the timestamp corresponding to the sending time. Figure 3 shows transmission of five warning messages. Since warning messages can contain primary and/or secondary notifications, we construct array `delayPN` that defines timestamps of transmission of primary notifications by network entity and array `delaySN` that defines timestamps of transmission of secondary notifications. The array `delayPN` has `nPrim` elements and the array `delaySN` has `nSec` elements. Timestamp of the first warning message is 0 and the message is of combined type. This means that $\texttt{delayPN}_1 = 0$ and $\texttt{delaySN}_1 = 0$. The second warning message is of primary type and has timestamp $\texttt{delayPN}_2$. The third warning message is of secondary type and has timestamp $\texttt{delaySN}_2$. The fourth warning message is also of secondary type and has timestamp $\texttt{delaySN}_3$. The last warning message is of combined type and hence has two equal timestamps $\texttt{delayPN}_3$ and $\texttt{delaySN}_4$.

Paging messages of primary and secondary notifications have different periodicity and we need to distinguish them in order to check that messages in the log comply with specification. Therefore we introduce a two dimensional array of correct timestamps of paging messages of primary notification *PagPN* of size $\texttt{nPrim} \cdot \texttt{ndPC}$, and another two dimensional array of correct timestamps of secondary notification *PagSN* of size $\texttt{nSec} \cdot \texttt{nBRmax}$, where `nBRmax` is the maximum number in the array `nBR` of numbers of paging messages of secondary notifications. We post constraints on these arrays which calculate periodicity of paging messages. Periodicity of paging messages of primary notification is equal to dPC as shown in Fig. 5. The time difference between two consecutive paging messages of secondary notification depends on dPC and the repetition period rPer of the notification, and can take two different values a and b for the same notification as shown in Fig. 5. Integer decision variable *delay* in Fig. 5 represents the delay of first paging message that is the time difference between time when radio base station starts to transmit primary notification and/or secondary notification and the time user equipment reads first paging message.

The arrays delayPN and delaySN represent the timestamps of the warning messages sent to the radio base station by a network entity. Since some variable delay can occur, we introduce arrays of decision variables *delayPN50* and *delaySN50* which represent the delay of each warning message. We assume that delays are between 0 and 50 ms.

We post a constraint to guarantee that if $\mathtt{delayPN}_i = \mathtt{delaySN}_j$ then the equality $delayPN50_i = delaySN50_j$ holds, $1 \leq i \leq \mathtt{nPrim}$ and $1 \leq j \leq \mathtt{nSec}$. Since the exact number of paging messages depends on *delay*, *delayPN50* and *delaySN50*, we set *PagPN* and *PagSN* to -1 to define missing paging messages.

Constraint (1) defines the timestamp of the first paging message of the first primary notification.

$$\begin{array}{l}
\mathtt{IF}\quad (\mathtt{delayPN}_1 = 0)\\
\quad\quad PagPN_{1,1} = 0\\
\mathtt{ELSEIF}\quad (\mathtt{nPrim} > 1)\\
\quad\quad ((r < \mathtt{delayPN}_2 - delay \wedge PagPN_{1,1} = r)\vee\\
\quad\quad (r \geq \mathtt{delayPN}_2 - delay \wedge PagPN_{1,1} = -1)\vee\\
\quad\quad (r \geq \mathtt{delayPN}_2 - delay\wedge\\
\quad\quad\quad r < \mathtt{delayPN}_2 - delay + 50 \wedge PagPN_{1,1} = r\wedge\\
\quad\quad\quad delayPN50_2 > r - \mathtt{delayPN}_2 + delay))\\
\mathtt{ELSE}\\
\quad\quad PagPN_{1,1} = r
\end{array} \tag{1}$$

where $r = \mathtt{roundupdPC}(\mathtt{delayPN}_1 - delay + delayPN50_1)$ and

$$\forall y \in \mathbb{N} \quad \mathtt{roundupdPC}(y) = y + \mathtt{dPC} - 1 - ((y + \mathtt{dPC} - 1) \mod \mathtt{dPC}) \tag{2}$$

Constraint (2) rounds y to the smallest integer that is greater or equal to y and a multiple of dPC. The constraint (2) is used to define timestamps of paging messages which are multiples of dPC.

Constraint (3) defines timestamp of first paging message of ith primary notification, $1 < i < \mathtt{nPrim}$.

$$\begin{array}{l}
(\forall 1 < i < \mathtt{nPrim})\\
\quad ((r < \mathtt{delayPN}_{i+1} - delay \wedge PagPN_{i,1} = r)\vee\\
\quad (r \geq \mathtt{delayPN}_{i+1} - delay \wedge PagPN_{i,1} = -1)\vee\\
\quad (r \geq \mathtt{delayPN}_{i+1} - delay \wedge r < \mathtt{delayPN}_{i+1} - delay + 50 \wedge PagPN_{i,1} = r\wedge\\
\quad\quad delayPN50_{i+1} > r - \mathtt{delayPN}_{i+1} + delay))
\end{array} \tag{3}$$

where $r = \texttt{roundupdPC}(\texttt{delayPN}_i - \mathit{delay} + \mathit{delayPN50}_i)$.

Constraint (4) defines the timestamp of the first paging message of the last primary notification

$$\begin{aligned}&\texttt{IF}\quad (\texttt{nPrim} > 1)\\ &\qquad \mathit{PagPN}_{\texttt{nPrim},1} = r\end{aligned} \tag{4}$$

where $r = \texttt{roundupdPC}(\texttt{delayPN}_{\texttt{nPrim}} - \mathit{delay} + \mathit{delayPN50}_{\texttt{nPrim}})$

By replacing `delayPN` by `delaySN`, *delayPN50* by *delaySN50* and `nPrim` by `nSec`, in (1), (3) and (4) we obtain a formula for calculating timestamps of first paging messages of secondary notification.

Constraint (5) defines the timestamp of $(k+1)$th paging message of ith primary notification, $1 \leq i < \texttt{nPrim}$

$$\begin{aligned}&(\forall 1 \leq i < \texttt{nPrim})(\forall 1 \leq k < \texttt{ndPC})\\ &\quad (\mathit{PagPN}_{i,k} \neq -1 \wedge \mathit{PagPN}_{i,k} + \texttt{dPC} < \texttt{delayPN}_{i+1} - \mathit{delay} + \mathit{delayPN50}_{i+1} \wedge\\ &\qquad \mathit{PagPN}_{i,k+1} = \mathit{PagPN}_{i,k} + \texttt{dPC})\\ &\quad \vee\\ &\quad (\mathit{PagPN}_{i,k} \neq -1 \wedge \mathit{PagPN}_{i,k} + \texttt{dPC} \geq \texttt{delayPN}_{i+1} - \mathit{delay} + \mathit{delayPN50}_{i+1} \wedge\\ &\qquad \mathit{PagPN}_{i,k+1} = -1)\\ &\quad \vee\\ &\quad (\mathit{PagPN}_{i,k} = -1 \wedge \mathit{PagPN}_{i,k+1} = -1)\end{aligned} \tag{5}$$

Constraint (6) defines timestamp of $(k+1)$th paging message of the last primary notification

$$\begin{aligned}&(\forall 1 \leq k < \texttt{ndPC})\\ &\quad \mathit{PagPN}_{\texttt{nPrim},k+1} = \mathit{PagPN}_{\texttt{nPrim},k} + \texttt{dPC}\end{aligned} \tag{6}$$

Constraint (7) defines timestamp of $(k+1)$th paging message of jth secondary notification, $1 \leq j \leq \texttt{nSec}$, $1 \leq k < \texttt{nBR}_j$.

$$\begin{aligned}&(\forall 1 \leq j \leq \texttt{nSec})(\forall 1 \leq k < \texttt{nBR}_j)\\ &\quad \texttt{IF}\quad (\mathit{PagSN}_{j,k} \geq 0 \wedge (j = \texttt{nSec} \vee \mathit{PagSN}_{j,1} + r <\\ &\qquad\qquad \texttt{delaySN}_{j+1} - \mathit{delay} + \mathit{delaySN50}_{j+1}))\\ &\qquad \mathit{PagSN}_{j,k+1} = \mathit{PagSN}_{j,1} + r\\ &\quad \texttt{ELSE}\\ &\qquad \mathit{PagSN}_{j,k+1} = -1\end{aligned} \tag{7}$$

where $r = \texttt{roundupdPC}(\texttt{rPer}_j \cdot k - (\mathit{PagSN}_{j,1} - \texttt{delaySN}_j - \mathit{delaySN50}_j + \mathit{delay}))$
We based our proofs in Coq on the structure of the constraints presented in this section.

5 A New Constraint Model for Delays

In this section we focus on the property of the protocol introduced in Sect. 5.1, which helped us to design a better constraint model presented in Sect. 5.2. A new constraint model includes constraints that were missing in [11]. The constraints presented in Sect. 5.2 were obtained by interactive theorem proving with Coq.

5.1 A Property of the Protocol

Variable message delay increases complexity of the protocol drastically. A radio base station can read a warning message sent by the network entity with some delay. Introducing for each message a variable that represents delay between 0 and 50 would result in combinatorial explosion. Our goal is to find a formula to compute delay based on the structure of the constraint model of the protocol that eliminates some values and make it easier for a constraint solver to find a solution. Constraints in Sect. 4, which calculate periodicity of paging messages, contain delays *delayPN50* and *delaySN50*. If we constrain the values of *delayPN50* and *delaySN50*, the new constraint model of the protocol should have a property that the array of timestamps of paging messages *PagPN* and *PagSN* will not change.

5.2 Constraints for Delay

The major impact of the delays *delayPN50* and *delaySN50* on *PagPN* and *PagSN* is that they can increase timestamps of paging messages by `dPC`. We introduce constraints that define delays *delayPN50constr* and *delaySN50constr* of notification messages. For each possible value of *delayPN50* and *delaySN50* we find corresponding values *delayPN50constr* and *delaySN50constr* such that arrays of timestamps of paging messages *PagPN* and *PagSN* will not change. The following constraints are implicitly universally quantified over $1 \leq i \leq \texttt{nPrim}$, $1 \leq j \leq \texttt{nSec}$ and $1 \leq k \leq \texttt{nBR}_j$.

The timestamp of the first paging message of the primary notification should be equal to the smallest value greater or equal to $\texttt{delayPN}_i - \mathit{delay}$ and divisible by `dPC`. In order to increase the timestamp of the first paging message by `dPC`, the delay *delayPN50constr* should be the smallest value that guarantees $\texttt{roundupdPC}(\texttt{delayPN}_i - \mathit{delay} + \mathit{delayPN50constr}) > \texttt{roundupdPC}(\texttt{delayPN}_i - \mathit{delay})$. This also holds for $\texttt{delaySN}_i$ and *delaySN50constr*.

Constraint (8) defines *delayPN50constr*

$$\begin{aligned}
&(\mathit{delayPN50}_i = 0 \wedge \mathit{delayPN50constr}_i = 0) \vee \\
&(\mathit{delayPN50}_i \geq 1 \wedge (\texttt{delayPN}_i - \mathit{delay}) \mod \texttt{dPC} = 0 \quad \wedge \\
&\qquad \mathit{delayPN50constr}_i = 1) \vee \\
&(\mathit{delayPN50}_i \geq 1 \wedge (\texttt{delayPN}_i - \mathit{delay}) \mod \texttt{dPC} > 0 \quad \wedge \\
&\qquad \texttt{roundupdPC}(\texttt{delayPN}_i - \mathit{delay}) = \\
&\qquad \texttt{roundupdPC}(\texttt{delayPN}_i - \mathit{delay} + \mathit{delayPN50}_i) \wedge \\
&\qquad \mathit{delayPN50constr}_i = 0) \vee \\
&(\mathit{delayPN50}_i \geq 1 \wedge (\texttt{delayPN}_i - \mathit{delay}) \mod \mathit{dPC} > 0 \quad \wedge \\
&\qquad \texttt{roundupdPC}(\texttt{delayPN}_i - \mathit{delay}) < \\
&\qquad\quad \texttt{roundupdPC}(\texttt{delayPN}_i - \mathit{delay} + \mathit{delayPN50}_i) \quad \wedge \\
&\qquad \mathit{delayPN50constr}_i = \texttt{dPC} - ((\texttt{delayPN}_i - \mathit{delay}) \mod \texttt{dPC}) + 1)
\end{aligned} \tag{8}$$

Let $\texttt{rPerCoq}_j = \texttt{rPer}_j \cdot k - 2 \cdot \texttt{dPC}$ where k is index of paging message of jth secondary notification, $1 \leq k \leq \texttt{nBR}_j$. Let $r = \texttt{roundupdPC}(\texttt{delaySN}_j - \mathit{delay} + \mathit{delaySN50}_j)$

The timestamp of the $(k+1)$th paging message of secondary notification depends on $\texttt{rPer}_j \cdot k$ that makes constraints for the delay of secondary notification more complex than for primary notification. In order to define *delaySN50constr* we consider two cases. The first case defines the delay that increases the timestamp of the first paging message of secondary notification. The second case introduces the delay that increases the timestamp of the $(k+1)$th paging message of secondary notification.

The first case requires that the Eq. (9) holds

$$\begin{aligned}&\texttt{roundupdPC}(\texttt{rPerCoq}_j + (2\cdot\texttt{dPC} + \texttt{delaySN}_j - \mathit{delay} - r)) = \\ &\texttt{roundupdPC}(\texttt{rPerCoq}_j + (2\cdot\texttt{dPC} + \texttt{delaySN}_j - \mathit{delay} - r) + \mathit{delaySN50}_j)\end{aligned} \quad (9)$$

Then by replacing the variable *delayPN50* by *delaySN50* and *delayPN50constr* by *delaySN50constr* in the constraint (8), we obtain the first part of the constraint for *delaySN50constr*.

If the Eq. (9) does not hold, then we define *delaySN50constr* as

$$\begin{aligned}
&(\texttt{roundupdPC}(\texttt{delaySN}_j - \mathit{delay}) = \\
&\quad \texttt{roundupdPC}(\texttt{delaySN}_j - \mathit{delay} + \mathit{delaySN50}_j) \quad \wedge \\
&\qquad \mathit{delaySN50constr}_j = \texttt{dPC} - \texttt{rPerCoq}_j \mod \texttt{dPC} - \\
&\qquad (\texttt{delaySN}_j - \mathit{delay} + (\texttt{dPC} - 1)) \mod \texttt{dPC}) \\
&\vee \\
&(\texttt{roundupdPC}(\texttt{delaySN}_j - \mathit{delay}) < \\
&\quad \texttt{roundupdPC}(\texttt{delaySN}_j - \mathit{delay} + \mathit{delaySN50}_j) \\
&\quad \wedge \\
&\qquad (((\texttt{delaySN}_j - \mathit{delay}) \mod \texttt{dPC} = 0 \quad \wedge \\
&\qquad\quad \texttt{rPerCoq}_j \mod \texttt{dPC} = 0 \wedge \mathit{delaySN50constr}_j = 1) \vee \\
&\qquad ((\texttt{delaySN}_j - \mathit{delay}) \mod \texttt{dPC} = 0 \wedge \\
&\qquad\quad \texttt{rPerCoq}_j \mod \texttt{dPC} > 0 \quad \wedge \\
&\qquad\quad \mathit{delaySN50constr}_j = 1 + \texttt{dPC} - \texttt{rPerCoq}_j \mod \texttt{dPC}) \vee \\
&\qquad ((\texttt{delaySN}_j - \mathit{delay}) \mod \texttt{dPC} > 0 \wedge \texttt{rPerCoq}_j \mod \texttt{dPC} = 0 \quad \wedge \\
&\qquad\quad \mathit{delaySN50constr}_j = \texttt{dPC} - (\texttt{delaySN}_j - \mathit{delay}) \mod \texttt{dPC} + 1) \vee \\
&\qquad ((\texttt{delaySN}_j - \mathit{delay}) \mod \texttt{dPC} > 0 \quad \wedge \\
&\qquad\quad \texttt{rPerCoq}_j \mod \texttt{dPC} > 0 \wedge \texttt{rPerCoq}_j \mod \texttt{dPC} \leq \\
&\qquad\qquad \texttt{dPC} - (\texttt{delaySN}_j - \mathit{delay}) \mod \texttt{dPC} \quad \wedge \\
&\qquad\quad \mathit{delaySN50constr}_j = \texttt{dPC} - (\texttt{delaySN}_j - \mathit{delay}) \mod \texttt{dPC} + 1) \vee \\
&\qquad ((\texttt{delaySN}_j - \mathit{delay}) \mod \texttt{dPC} > 0 \quad \wedge \\
&\qquad\quad \texttt{rPerCoq}_j \mod \texttt{dPC} > 0 \wedge \texttt{rPerCoq}_j \mod \texttt{dPC} > \\
&\qquad\qquad \texttt{dPC} - (\texttt{delaySN}_j - \mathit{delay}) \mod \texttt{dPC} \quad \wedge \\
&\qquad\quad \mathit{delaySN50constr}_j = \texttt{dPC} - (\texttt{delaySN}_j - \mathit{delay}) \mod \texttt{dPC} + \\
&\qquad\qquad \texttt{dPC} - \texttt{rPerCoq}_j \mod \texttt{dPC} + 1))
\end{aligned} \quad (10)$$

Constraints (8)–(10) contain decision variables *delayPN50* and *delaySN50*. Our goal is to replace these variables by *delayPN50constr* and *delaySN50constr* in our MiniZinc constraint model, thus *delayPN50* and *delaySN50* should be eliminated. For example, constraint (8) will be replaced by

$$
\begin{aligned}
(\mathit{delayPN50constr}_i = 0 \lor ((\mathtt{delayPN}_i - \mathit{delay}) \mod \mathtt{dPC} = 0 \land \\
\mathit{delayPN50constr}_i = 1) \lor (\mathtt{delayPN}_i - \mathit{delay}) \mod \mathtt{dPC} > 0 \land \\
\mathit{delayPN50constr}_i = \mathtt{dPC} - ((\mathtt{delayPN}_i - \mathit{delay}) \mod \mathtt{dPC}) + 1),
\end{aligned}
\tag{11}
$$

we add this constraint to the model and we replace the variable *delayPN50* by *delayPN50constr* in (1)–(5).

6 Proofs in Coq

We made several assumptions that are formalised in Coq as axioms.[1] Constraint (12) is based on 3GPP standard.

$$\mathtt{dPC} \geq 320 \tag{12}$$

Constraint (13) is based on the property that the user equipment can read first paging messages with *delay* less than dPC and we assume that radio base station can receive first notification message with delay less than 50 ms.

$$0 \leq \mathit{delay} < \mathtt{dPC} + 50 \tag{13}$$

Constraints (14), (15) and (16) are required by proofs in Coq, which use lemmas of natural arithmetics.

$$\mathtt{rPer}_j > 2 \cdot \mathtt{dPC} \tag{14}$$

$$\mathtt{delayPN}_i > \mathit{delay} \tag{15}$$

$$\mathtt{delaySN}_j > \mathit{delay} \tag{16}$$

$\mathtt{delayPN}_1$ and $\mathtt{delaySN}_1$ can be equal to 0, but we do not consider this case, since then we set $\mathit{delayPN50}_1$ and $\mathit{delaySN50}_1$ to 0. Constraint (14) is a stronger version of assumption we made in [11]. We assumed in [11] that $\mathtt{rPer}_j > \mathtt{dPC}$, but we did not take in consideration message delays.

We started by proving correctness of the constraint that was manually derived for delay. However, while applying tactics in Coq, we found that some cases were missing in the constraint. The parameter rPer did not occur in formula for the delay. The constraints described in the previous section were derived by analysing the required assumptions after applying tactics.

We proved in Coq that the delays *delayPN50* and *delaySN50* can be replaced by *delayPN50constr* and *delaySN50constr* in the constraint model, that is the solutions for arrays of decision variables *PagPN* and *PagSN* are not changed after this replacement.

We split the proof into several lemmas and theorems, where we used Coq library for basic Peano arithmetic. We formulated different theorems for paging messages of primary and secondary notifications. Since constraints on timestamps of paging messages of secondary notifications are more complex, they require more lemmas to prove.

The proof consists of several steps.

[1] The Coq model is available at https://github.com/astra-uu-se/SEFM18.

1. We proved general properties of `roundupdPC`. For example, we prove that
$$\begin{aligned}&\forall n, d \in \mathbb{N},\\ &(d \leq 50 \land \texttt{roundupdPC}(n) < \texttt{roundupdPC}(n+d)) \rightarrow\\ &\texttt{roundupdPC}(n+d) = \texttt{roundupdPC}(n) + \texttt{dPC}\end{aligned}$$
2. We proved that $\mathit{delayPN50constr}_i \leq \mathit{delayPN50}_i$ and $\mathit{delaySN50constr}_j \leq \mathit{delaySN50}_j$, $1 \leq i \leq \texttt{nPrim}$, $1 \leq j \leq \texttt{nSec}$.
3. We proved that
$$\texttt{roundupdPC}(t + \mathit{delayPN50constr}_i) = \texttt{roundupdPC}(t + \mathit{delayPN50}_i),$$
and
$$\texttt{roundupdPC}(t' + \mathit{delaySN50constr}_j) = \texttt{roundupdPC}(t' + \mathit{delaySN50}_j),$$
where t, t' are expressions from constraint model.
4. Let $t' \mod \texttt{dPC} = 0$. We proved that
   - if $t' < (\texttt{delayPN}_i - \mathit{delay}) + \mathit{delayPN50}_i$ and $t' \geq (\texttt{delayPN}_i - \mathit{delay})$ then $t' < (\texttt{delayPN}_i - \mathit{delay}) + \mathit{delayPN50constr}_i$.
   - if $t' < (\texttt{delaySN}_j - \mathit{delay}) + \mathit{delaySN50}_j$ and $t' \geq (\texttt{delaySN}_j - \mathit{delay})$ then $t' < (\texttt{delaySN}_j - \mathit{delay}) + \mathit{delaySN50constr}_j$.
5. In constraint (10) $\mathit{delaySN50constr}_j$ depends on k. We showed that we do not need to have two or more values of $\mathit{delaySN50constr}_j$ with different values of k for the same $\mathit{delaySN50}_j$, $\texttt{delaySN}_j$ and delay. We proved that we can always choose the largest value of $\mathit{delaySN50constr}_j$.

From Step 2 we derived that

- $\mathit{delayPN50constr}_i \leq 50$,
- $\mathit{delaySN50constr}_j \leq 50$,
- if $t > (\texttt{delayPN}_i - \mathit{delay}) + \mathit{delayPN50}_i$, then $t > (\texttt{delayPN}_i - \mathit{delay}) + \mathit{delayPN50constr}_i$,
- if $t > (\texttt{delaySN}_j - \mathit{delay}) + \mathit{delaySN50}_j$, then $t > (\texttt{delaySN}_j - \mathit{delay}) + \mathit{delaySN50constr}_j$

Constraint (10) can be simplified by removing $2 \cdot \texttt{dPC}$, since we have expression $\texttt{rPerCoq}_j + 2 \cdot \texttt{dPC} = \texttt{rPer}_j \cdot k - 2 \cdot \texttt{dPC} + 2 \cdot \texttt{dPC}$. We add $2 \cdot \texttt{dPC}$ in order to be able to use lemmas of natural arithmetics in Coq. We have $\texttt{delaySN}_j - \mathit{delay} - r \leq 0$, but $\texttt{rPer}_j \cdot k - 2 \cdot \texttt{dPC} > 0$ by our assumption and $2 \cdot \texttt{dPC} + \texttt{delaySN}_j - \mathit{delay} - r > 0$.

7 Related Work

Because of the importance of network protocols there have been many case studies on the application of formal methods to the verification and study of network protocols. Some of the early work included the use of finite automata, Petri nets and symbolic execution to verify the absence of deadlock and liveness properties, see [19]. It is impossible here to give a complete survey of the field, (see [17] for a recent survey) instead we will concentrate on the use of theorem provers based on type theory, such as Coq or Isabelle [16] to verify non-trivial properties of network protocols.

One advantage of using a theorem prover over a model checker is that it is easier to reason about infinite objects and hence prove properties that are satisfied by every possible run of a protocol. In [9] a novel use of co-inductive types, which correspond to infinite streams of data, was used to model and verify the alternating bit protocol. While [9] used a process calculus that characterises possible computation steps that can be performed in a protocol, the work in [5] uses an algebraic approach that captures when two processes are equivalent. In order to automate a hand-written proof of the alternating bit protocol, an encoding of a rich and widely used algebraic specification language for concurrent systems with data, μCRL [12] was formalised in Coq.

More recent work (see [3] and the references therein) on process calculi in Isabelle has resulted in generic framework, the Psi-Calculus, that captures many different process calculi. The resulting formalisation is over $32,000$ lines of Isabelle code. In [7] an extension of the Psi-calculus was given to capture the broadcast of messages and applied to a non-trivial wireless sensor network protocol.

The use of interactive theorem proving for high-level constraint models in languages similar to MiniZinc has been considered in [4,8]. In [4], the authors show that almost all interesting properties of a constraints model, such as model equivalence, are undecidable in general for languages as expressive as MiniZinc. However, they illustrate that properties can be automatically verified when a restricted language is considered. In [8] interactive theorem proving was used to derive symmetry breaking constraints of models. The work in [4,8] considers the general problem of reasoning about any model, while we consider a specific case study where undecidability is not a problem.

All of the cited work so far has been concerned with protocols that do not have a time component. In our application the timing of messages is of crucial importance to the correct operation of the protocol. In [13] a real time protocol is verified using HOL (closely related to Isabelle [16]). Further resulting formalisation is then used in conjunction with the theorem analyse the qualitative soft real-time behaviour of the protocol. This is similar in spirit our derivation of message delays.

8 Conclusion

We used Coq to discover a formula for message delay in a 4G protocol and proved correctness of the formula.

We analyzed real logs from [11] with derived delays. However, there was an impact on performance of large generated log analysis. We still can analyze logs with large number of errors, but with a size smaller than in [11].

We found that Coq is a useful tool. It would be hard to do such proofs by hand and the tool helped to construct formula for delay. We want to emphasize that interactive feature of Coq was very important, since the formula in the initial model was not correct. Proofs are about 6500 lines, but we believe that number of lines can be reduced by improving our use of tactics. Our proofs are based on the structure of the constraint model used in [11]. The proofs could also be done with Isabelle/HOL or PVS. In this formal development we rely on Coq features related to inductive types and first order logics and last but not least on Coq standard library (for modulo). We also used intensively the omega tactic to solve some arithmetic subgoals. Why3 [6] could be an alternative for our work, offering a large choice of automatic solvers and proof assistants. However it depends on the way these solvers support the modulo operator. A perspective is to apply this approach to other protocols with message delay to create more efficient constraint model. An interesting question to explore is how to convert automatically constraint models into Coq.

Acknowledgments. The second author was supported by Swedish Foundation for Strategic Research. The third author is partially support by the Swedish Research Council VR.

References

1. 3GPP. Public warning system (PWS) requirements. TS 22.268, 3rd Generation Partnership Project (3GPP). http://www.3gpp.org/ftp/Specs/html-info/22268.htm
2. Barras, B., Werner, B.: Coq in Coq. Technical report, INRIA-Rocquencourt (1997)
3. Bengtson, J., Parrow, J., Weber, T.: Psi-Calculi in Isabelle. J. Autom. Reasoning **56**(1), 1–47 (2016)
4. Bessiere, C., Hebrard, E., Katsirelos, G., Kiziltan, Z., Narodytska, N., Walsh, T.: Reasoning about Constraint Models. In: Pham, D.-N., Park, S.-B. (eds.) PRICAI 2014. LNCS (LNAI), vol. 8862, pp. 795–808. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13560-1_63
5. Bezem, M., Bol, R., Groote, J.F.: Formalizing process algebraic verifications in the calculus of constructions. Formal Aspects Comput. **9**(1), 1–48 (1997)
6. Bobot, F., Filliâtre, J.-C., Marché, C., Paskevich, A.: Why3: Shepherd your herd of provers. In: Workshop on Intermediate Verification Languages (2011)
7. Borgström, J., Huang, S., Johansson, M., Raabjerg, P., Victor, B., Pohjola, J.Å., Parrow, J.: Broadcast psi-calculi with an application to wireless protocols. Softw. Syst. Model. **14**(1), 201–216 (2015)
8. Cadoli, M., Mancini, T.: Using a theorem prover for reasoning on constraint problems. Appl. Artif. Intell. **21**(4&5), 383–404 (2007)
9. Giménez, E.: An application of co-inductive types in Coq: verification of the alternating bit protocol. In: Berardi, S., Coppo, M. (eds.) TYPES 1995. LNCS, vol. 1158, pp. 135–152. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61780-9_67
10. Gonthier, G.: The four colour theorem: engineering of a formal proof. In: 8th Asian Symposium of Computer Mathematics, p. 333. ASCM (2007)
11. Grinchtein, O., Carlsson, M., Pearson, J.: A constraint optimisation model for analysis of telecommunication protocol logs. In: Blanchette, J.C., Kosmatov, N. (eds.) TAP 2015. LNCS, vol. 9154, pp. 137–154. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21215-9_9

12. Groote, J.F., Ponse, A.: The syntax and semantics of μCRL. In: Ponse, A., Verhoef, C., van Vlijmen, S.F.M. (eds.) Algebra of Communicating Processes. Workshops in Computing. Springer, London (1995). https://doi.org/10.1007/978-1-4471-2120-6_2
13. Hasan, O., Tahar, S.: Performance analysis and functional verification of the stop-and-wait protocol in HOL. J. Autom. Reason. **42**(1), 1–33 (2009)
14. Leroy, X.: Formal verification of a realistic compiler. Commun. ACM **52**(7), 107–115 (2009)
15. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: MiniZinc: towards a standard CP modelling language. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 529–543. Springer, Heidelberg (2007)
16. Nipkow, T., Paulson, L.C., Wenzel, M. (eds.): Isabelle/HOL: A Proof Assistant for Higher-Order Logic. LNCS, vol. 2283. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45949-9
17. Qadir, J., Hasan, O.: Applying formal methods to networking: theory, techniques, and applications. IEEE Commun. Surv. Tutorials **17**(1), 256–291 (2015)
18. Rossi, F., van Beek, P., Walsh, T. (eds.): Handbook of Constraint Programming. Elsevier, New York (2006)
19. Sunshine, C.A.: Survey of protocol definition and verification techniques. SIGCOMM Comput. Commun. Rev. **8**(3), 35–41 (1978)
20. The Coq Development Team. The Coq proof assistant reference manual version 8.6 (2016)