Tools for Parametric Verification. A Comparison on A Case Study.

Petr Matoušek

Faculty of Information Technology, Božetěchova 2
612 66 Brno, Czech republic
matousp@fit.vutbr.cz

Abstract. Protocol analysis involve several parameters in model specification, for instance, transmission delay or the length of the transmitting window. Verification of the model with parameters is semi-decision process that depends on number of clocks, parameters and counters in the model. Using combination of different verification tools for timed models as HyTech, TReX and Uppaal we are able to find relation between parameters satisfying desired property. The paper gives a report on synthesis of parameters of PGM protocol [SFC+01]. We built a formal model based on extended time automata with parameters and verified the reliability property. Our results automatically obtained from the model are consistent with previous results derived manually. The paper describes our experiences with parametric verification of multicast protocol PGM. Results mentioned in the work were made with collaboration with Mihaela Sighireanu[1] from LIAFA, Paris.

Introduction

Model-checking is a popular technique of verification of untimed as well timed systems. In comparison with theorem proving model-checking provides an easy way how to specify and verify a model without profound experiences of logics. For timed systems, model checkers like Uppaal, Kronos, If, TReX can be used. The basic goal of model-checking is to verify a property on a specified model using state space search. In practice, a model checker analyses (explicitly or implicitly) all reachable configurations and tests if the model does not violate desired property.

PGM protocol. In our work we verified a reliability property of PGM protocol. PGM protocol defined in [SFC+01] is a reliable multicast transport protocol for applications that multicast data from multiple sources to multiple receivers. It is a standard defined by Cisco for telecommunication services, for instance, video conferencing. We studied reliability of the transmission - *the full recovery property.* The property states that "PGM either receives all data packets from transmission and repairs, or is able to detect unrecoverable data packet loss".

[1] Mihaela.Sighireanu@liafa.jussieu.fr

Behaviour of a protocol depends on several aspects - the rate of transmission, the speed of packet generation, the length of the transmission window of the source or limits of network elements. Our interest is to find dependecies among these variables with respect to the full recovery property.

Parametric reasoning works on a model with parameters - variables whose values are not changed during analysis. Using parametric reasoning we can either verify that the system satisfies some property for all possible values of the parameters, or find constraints on the parameters defining the set of all possible values for which the system satisfies a property.

In our case study we focus on the synthesis of parameters, i.e., finding constraints on the parameters that ensure satisfaction of the full recovery property. Our model was implemented and verified using different tools: Uppaal [PL00], TReX [BCAS01] and HyTech [HHWT97]. Uppaal is a model checker that does not support parameters so we instantiated variables for some values. HyTech provides parametric verification of hybrid systems. Timed automata are a subset of hybrid automata so we can implement our model in HyTech. HyTech had problem with termination, so we were not able to generate full state space of the model and verify desired property. However, using instantiation of certain parameters and putting constraints on the others we can obtain some relations between parameters as a result. The results were confirmed by similar analysis using TReX. TReX - a tool for parametric verification of extended timed system with counters - implements powerful acceleration that helps to terminate computation. It outputs graphs of symbolic configurations and analysis traces which we used to study system behaviour and detect configurations where the property is satisfied.

Contribution of our work is to show how parametric verification works on a non-trivial example of a real protocol and what kind of tools can be used for parametric verification of timed systems. We identify possible bottlenecks of analysis, especially the features of the system that induces non termination. Our observation and recommendations on parametric analysis are mentioned in this work.

Outline of the paper. In the first section, we introduce the analyzed protocol. We describe a formal model of the protocol based on extended timed automata, timed automata with counters [AAB00]. Then we briefly introduce tools we used for verification - Uppaal, TReX, HyTech and results we achieved. The third section of the paper summarizes our results and experiences with parametric verification and discusses our observation on parametric verification of timed systems.

Related work. There are several works on verification of PGM protocol. [BBP02] verifies a simplified timed version of PGM with a linear topology and a one-placed buffer. Two properties - lost info property and no-loss property are verified in this work. The properties are verified by instantiating parameters using Uppaal. [BL02] validates the sliding window mechanism for any number of data packet sent using LASH. The model used is untimed. A more complex timed model of PGM is considered in [BS03]. This model includes parameters. The constraints on parameters are obtained manually and then verified by instantiation. In contrast to this work, we did synthesis and verification fully automatically.

1 Modelling PGM

PGM protocol works on a network of nodes with multiple senders and multiple receivers. In our approach we abstract the model to a simple one-sender and one-receiver system. Joining and leaving multiple nodes during session can be abstracted like nodes missing data [BS03].

PGM Model. Our PGM model is composed from three automata – a sender, a network and a receiver with six parameters, one finite variable, two clocks, two counters and two communication channels, see Figure 1.

Automata in our model communicate by rendez-vous on gates SN and NR and by shared variables L and *lp*. The states labelled by C are urgent states, i.e., states where the time is not allowed to advance.

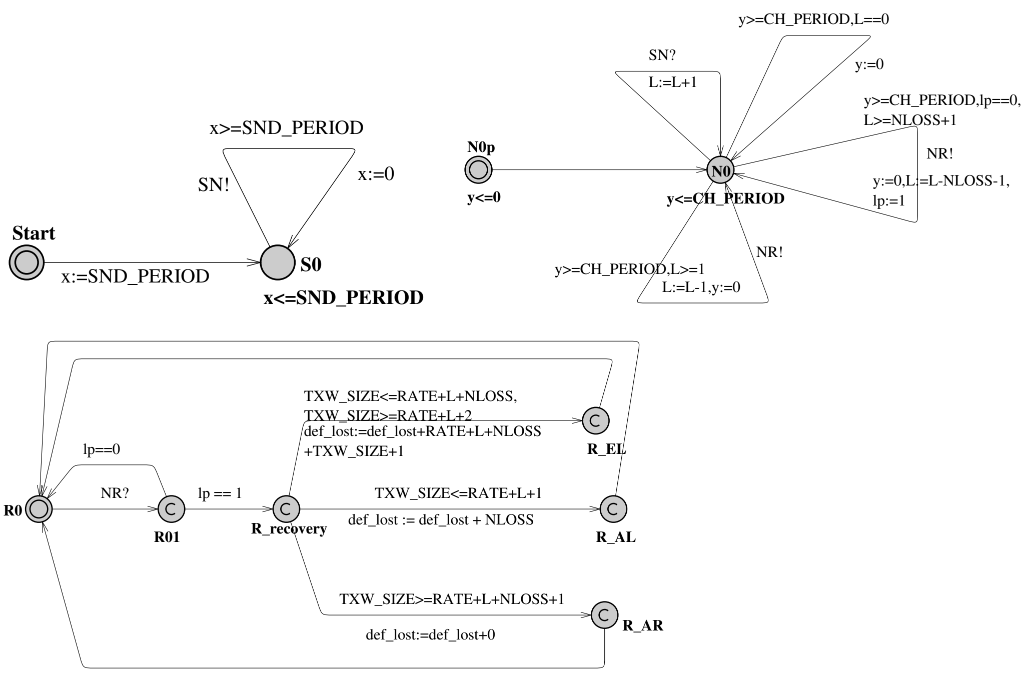

Fig. 1. PGM model - sender, network and receiver

The sender generates a new data each period `SND_PERIOD`. Sent data are stored in the transmitting window that advances each time a new data is sent. The transmitting window is fully opened during the session to recover as many data packets as possible. If data loss is detected, we test if an original appropriate data packet is in the transmitting window. If not, a non-recoverable data loss has happened and the full recovery property is violated.

The network automaton models the transmission channel between the sender and the receiver with transmission delays and non-deterministic losses. The network receives data from the sender and increments the length of a buffer L. The buffer is unbounded, it can grow without limitation. The network element either delivers data to the receiver with the speed defined by parameter `CH_PERIOD` or multiple data are discarded in order to model losses during transmission. The model allow `NLOSS` data packets to be lost, `NLOSS` is a parameter. The initial buffer length is set to `BUFFER_LENGTH`, which means the system is in process of communication - we don't model opening and closing stages of communication.

The receiver is informed about losses using a global variable lp. When a loss occurs the receiver calculates possibility of recovery. The result depends on `TXW_SIZE`, `BUFFER_LENGTH`, `RATE` and the current length of the buffer L'.

By reasoning about the recovery of transmitted data we distinguish three possible cases - every lost packet can be recover, some lost packets can be recover or nothing can be recover. This depends on the size of the sender's transmitting window, the speed of transmission, the delay on network etc. These cases can be described by following manually obtained results:

∀R All lost packets may be recovered (full recovery) if $\texttt{TXW\_SIZE} > \texttt{RATE} + L' + \texttt{NLOSS}$, state R_AR,

∀L None of the `NLOSS` lost packets may be recovered (no recovery) if $\texttt{TXW\_SIZE} \leq \texttt{RATE} + L' + 1$, state R_AL, or

∃R Some of the lost packets may be recovered (partial recovery) if $\texttt{TXW\_SIZE} > \texttt{RATE} + L' + 1$ and $\texttt{TXW\_SIZE} \leq \texttt{RATE} + L' + \texttt{NLOSS}$ (state R_EL) .

The full recovery may be done for the first case if the parameters satisfy constraint $\texttt{SND\_PERIOD} \geq \texttt{CH\_PERIOD} \wedge \texttt{TXW\_SIZE} \geq \texttt{RATE} + \texttt{BUFFER\_LENGTH}$. This constraint on parameters was obtained manually in [BS03] and [MS04]. In this paper we focus on automatical synthesis of parameters. However, it is interesting to compare the manually obtained results with output of verification tools listed in the following section. It is seen that the results are consistent.

2 Tools for parametric verfication

For parametric verification we used three tools - HyTech, TReX and Uppaal. In this part we introduce the tools and our results. As mentioned in [AHV93], a large class of parametric verification problems is undecidable. In [AAB00] the authors introduce a semi-logaritmic approach based on an expressive symbolic representation, parametric DBMs, and extrapolation techniques that allows one to speed up reachability analysis and helps its termination. We will see how important is an effective extrapolation technique in comparison of TReX and HyTech.

2.1 HyTech

HyTech [HHWT95] is a tool for analysis of linear hybrid automata [ACHH93]. A hybrid automaton is a mathematical model for hybrid systems that models both their discrete and continuous behaviour. Hybrid automata can be considered a generalization of timed automata where continuous variables are used for modelling any continuous variable. Timed automata have one type of continuous variables - clocks. Generally, hybrid systems are undecidable [HHWT95]. Linear hybrid systems form a subclass of hybrid systems which can be analysed semi-automatically [ACHH93]. Invariants, guards and actions in linear hybrid systems depend linearly on time and other variables.

HyTech is a symbolic model checker for linear hybrid automata. An important feature of HyTech is its ability to perform parametric analysis. It is able to synthetize parameter values, i.e., find the correct values for the parameters so that the system will satisfy a specified property.

Model description. As input HyTech takes a description of a model in form of linear hybrid system and analysis commands. System description contains variable of several types: *discrete, clock, stopwatch, parameter* and *analog*. Guards and constraints are composed of linear terms and expressions. Each automaton is composed of locations and their transitions, locations are labeled with their invariants. Transitions contain guards with enabling conditions and the successor location. There must be provided an initial state of an automaton and an initial value of the variables.

Model analysis. Analysis in HyTech is specified by two parts: declaration of regions, and a sequence of analysis commands. Analysis commands provide a means of manipulating and outputting regions. At any time instant, the state of a hybrid automaton is specified by a location and constraints on variables. This is called a region. HyTech computes the forward reachable region by finding the limit of the infinite sequence *I, post(I)*, $post^2(I)$, ... of regions. All timed safety requirements including bounded-time response requirements, can be verified using the reachability set. However, the iteration scheme is a semidecision process: there is no guarantee of termination.

In our first approach, we computed the reachability set of the system. The property to be verified of the system was expressed in negative form using a region that violates the property: $final\_reg := def\_lost > 0$. Term $def\_lost > 0$ defines states where the recovery property is not satisfied, i.e., number of definitely lost packets is greater then zero. Firstly, HyTech generates a set of all reachable configurations of the system. Then intersetion with specified property is applied on the set. If the property holds we get non-empty result in form of equations between parameters that satisfies our model and specified conditions. Declaration of analyzed region and analysis commands in HyTech for the first approach is following:

```
-- definition of initial and final region
init_reg, final_reg: region;

-- region inizialization
init_reg := loc[sender] = S0 & x = SND_PERIOD & loc[Node] = N0 & y =
  0 & L = BUFFER_LENGTH & lp = 0 & loc[receiver] = R0 & def_lost = 0
  & RATE >= 1 & TXW_SIZE >= 1 & NLOSS >= 1 & BUFFER_LENGTH >= 1 &
  CH_PERIOD >= 1 ;
```

```
-- a violation state (final_reg)
final_reg :=  def_lost > 0;

-- analysis
reached := reach forward from init_reg endreach;
prints "------------";
print omit all locations
        hide non_parameters in reached & final_reg endhide;
```

For the first approach computation did not terminate. In symbolic model checking are very important techniques like acceleration that help to speed up and terminate analysis. For above written example HyTech had problem to accelerate and after few hours the computation failed because of out of memory.

The second approach analyses the model on-the-fly. At first, the nearest reachable region is computed using *post()* operation and then, immediately intersection of the region and the undesirable property *final_region* is tested. If the intersection is non empty, non-reliable state was reached. If the intersection is empty, we continue iteration. We cannot find all states satisfying the property but we can determine states that violate the property and synthetize parameters for non-allowed states. In HyTech , the second approach is written as follows:

```
init_reg, reached,old, final_reg: region;

init_reg := loc[sender] = S0 & x = SND_PERIOD & loc[Node] = N0 & y =
  0 & L = BUFFER_LENGTH & lp = 0 & loc[receiver] = R0 & def_lost = 0
  & RATE >= 1 & TXW_SIZE >= 1 & NLOSS >= 1 & BUFFER_LENGTH >= 1 &
  CH_PERIOD >= 1 ;

final_reg :=  def_lost > 0;

-- initialize region reached:
reached := init_reg;
prints "------------";
while empty(reached & final_reg) do
  old:= reached;
  reached:=post(old);
  print diff(reached, old);
endwhile;

prints "reached & final_reg:";
print omit all locations hide non_parameters in reached & final_reg
endhide;
```

Results. During analysis of PGM we distinguish four different cases depending on the speed of

- *Case 1:* SND_PERIOD $>$ CH_PERIOD - the rate of arrivals is less than departures, the size of the queue converges to zero. Following constraints on parameters were synthetized:

  ```
  CH_PERIOD < SND_PERIOD & CH_PERIOD >= 1 & NLOSS >= 1 & NLOSS <= BUFFER_LENGTH
  & RATE >= 1 & TXW_SIZE >= 1 & TXW_SIZE + NLOSS <= RATE + BUFFER_LENGTH + 1
    |
  RATE >= 1 & NLOSS <= BUFFER_LENGTH & TXW_SIZE + NLOSS >= RATE + BUFFER_LENGTH + 2
  & CH_PERIOD < SND_PERIOD & CH_PERIOD >= 1 & TXW_SIZE <= RATE + BUFFER_LENGTH
  ```

  The result shows that for TXW_SIZE $\leq$ RATE $+$ BUFFER_LENGTH $-$ NLOSS (first part of the formula) nothing can be recovered and for TXW_SIZE $>$ RATE $+$

BUFFER_LENGTH − NLOSS (second part of the formula) some losses can be recovered. This corresponds with results obtained by TReX - see later.

- *Case 2:* SND_PERIOD = CH_PERIOD - arrivals are the same speed as departures, the size of the queue decreases to zero by number of losses NLOSS that occurs non-deterministic losses in the queue. The constrained obtained are the same as in the previous case.
- *Case 3:* CH_PERIOD/SND_PERIOD > NLOSS - arrivals are faster then departures and losses, the queue grows beyond any limits.
  For this case and case 4, we introduce a new parameter q = CH_PERIOD/SND_PERIOD, and we consider that $q \geq 2$. Parameter synthesis obtained by HyTech for $q = 2$ is:

```
q >= NLOSS + 1 & SND_PERIOD > 1 & BUFFER_LENGTH >= 1 & NLOSS <= BUFFER_LENGTH + 1
& RATE >= 1 & TXW_SIZE + NLOSS <= RATE + BUFFER_LENGTH + 2 &TXW_SIZE>= 1 & NLOSS >= 1
 |
q >= NLOSS + 1 & NLOSS <= BUFFER_LENGTH + 1 & RATE >= 1 & TXW_SIZE + NLOSS >= RATE
+ BUFFER_LENGTH + 3 & SND_PERIOD > 1 & TXW_SIZE <= RATE + BUFFER_LENGTH + 1
```

  This is for $q = 2$. If we set q equal to $\{3, 4, \ldots\}$ we obtain similar results that differ by constants in relation with TXW_SIZE.
- *Case 4:* NLOSS > CH_PERIOD/SND_PERIOD > 1 - arrivals are faster than departures but not enough to fill the losses between two delivery, the size of the buffer does not grow to fast because of losses.
  The experiments and the results are similar to the third case.

2.2 TReX

TReX [BCAS01] is a tool that allows one to analyse automatically automata-based models equipped with variables of different kinds of infinite domain and with parameters. The models are parametric timed automata extended with integer counters and communicating through unbounded FIFO queues.

The verification techniques is improved by efficient extrapolation technique. TReX allows on-the-fly model checking as well as generation of the set of reachable configuration and of a finite symbolic graph.

Model description and analysis. A model of the system is specified using an input language that is a subset of language IF [BFG$^+$00]. A model contains timed automata with counters, parameters and gates for synchronization. In .cnd file we specify initial constraints on parameters to help its termination. The output of verification is a resulting finite graph (.sg), a set of symbolic configurations (.res) and a list of traces/runs (.tr) over a symbolic configuration graph.

Using a set of traces and a graph of symbolic configuration we can observe behaviour of the system and find a relation between parameters satisfying desired property. In our case we search for configurations where number of definitively lost packet is zero. This configuration satisfies the full recovery property.

In HyTech we were able to verify only counter-examples, i.e., configurations where the property was violated. We did not succeed to generate a full set of reachable configuration. In contrary, TReX generate successfully a full graph of all reachable configurations. From this graph we can synthetize parameters satisfying desired property. For instance, in Figure 2 we can see all possible traces (runs) of the model for which the desired property $def\_lost = 0$ holds. The graph was generated from TREX (.tr file) for the full recovery property (def_lost = 0) and `CH_PERIOD`= `SND_PERIOD`. We can observe dependency of an initial value of the buffer `BUFFER_LENGTH` on current length L of the buffer for all recovery property.

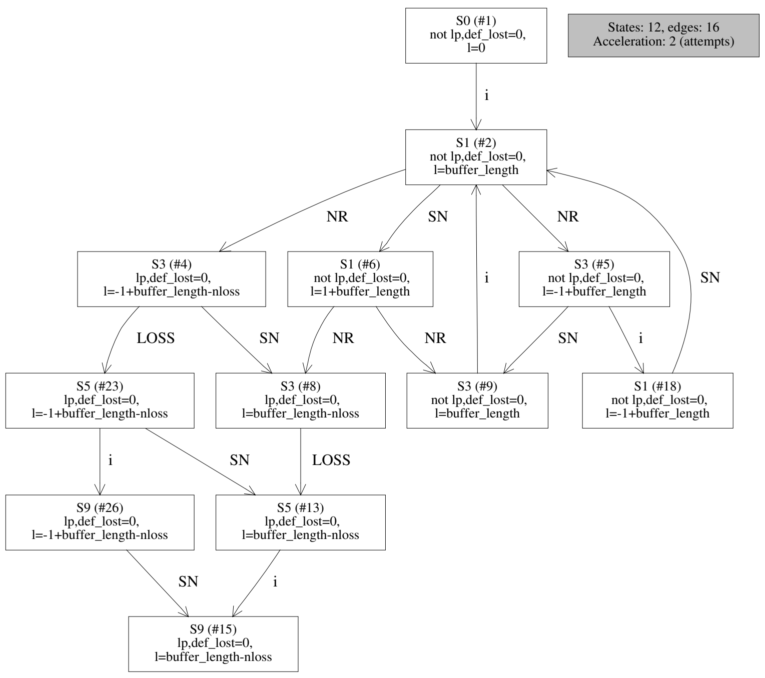
S0 (#1) not lp,def_lost=0, l=0
States: 12, edges: 16 Acceleration: 2 (attempts)
i
S1 (#2) not lp,def_lost=0, l=buffer_length
NR
SN
NR
S3 (#4) lp,def_lost=0, l=-1+buffer_length-nloss
S1 (#6) not lp,def_lost=0, l=1+buffer_length
i
S3 (#5) not lp,def_lost=0, l=-1+buffer_length
SN
LOSS
SN
NR
NR
SN
i
S5 (#23) lp,def_lost=0, l=-1+buffer_length-nloss
S3 (#8) lp,def_lost=0, l=buffer_length-nloss
S3 (#9) not lp,def_lost=0, l=buffer_length
S1 (#18) not lp,def_lost=0, l=-1+buffer_length
i
SN
LOSS
S9 (#26) lp,def_lost=0, l=-1+buffer_length-nloss
S5 (#13) lp,def_lost=0, l=buffer_length-nloss
SN
i
S9 (#15) lp,def_lost=0, l=buffer_length-nloss

Fig. 2. TREX - symbolic reachability graph

Results obtained from a set of symbolic configurations cover all the three cases for recovering losses. We can see that relations for case no recovery and partial recovery correspond with HYTECH results and full recovery with manually obtained relation. The example is for `SND_PERIOD` > `CH_PERIOD`:

- R_AR - full recovery

  txw_size $\geq$ rate + buffer_length and buffer_length $\geq$ nloss + 1
- R_EL - partial recovery

  txw_size $\geq$ rate + buffer_length -nloss - n3 - 1 and twx_size $\leq$ rate + buffer_length - n3 - 3 and buffer_length $\geq$ nloss + n3 - 3 and buffer_length $\geq$ n3 - 2 and n3 $\geq$ 0
- R_AL - no recovery

  txw_size $\leq$ rate -nloss + buffer_length - n3 - 1 and buffer_length $\geq$ nloss + n3 - 2 and buffer_length - n3 - 1 $\leq$ 0 and n3 $\geq$ 0

2.3 Uppaal

Uppaal is a tool for validation and verification of RT systems developed by colaboration of Uppsala University in Sweden and Aalborg University in Denmark [PL00]. A model is described using timed automata. Verifier checks specified properties that are expressed using simple temporal logic with operators `E<>`, `A[]`, `E[]`, `A<>`. Uppaal verifies existence of deadlock using special property `A[] not deadlock`.

Model description. A part of Uppaal tool is a simulator that performs simulation on a specified model. We used the simulator especially in the first stage of the model specification where we tuned our model and compared it with the given protocol. A model wass decribed graphically using a build-in editor. The specification is visual and enables the first check of the consistency of the model. Specification of our model in Uppaal is in Figure 1.

Model analysis. In our project we verified using Uppaal states where `def_lost > 0` and `def_lost = 0`. Uppaal does not support parametric verification, so we instantiated parameters. Using Uppaal we were able to prove that our results obtained by HyTech and TReX are consistent and that our model is deadlock-free.

Results. In this part we briefly show obtained results.

- *Result 1:* relation between current length of the buffer and number of definitively lost packets for different values of parameters.
  constants: `CH_PERIOD`: 10, `SND_PERIOD`: 15, `RATE`: 0

`TXW_SIZE`	10	10	10	10	10	12	11	10	10	12	12	13	14	10	10	2
`NLOSS`	3	3	3	3	3	3	3	3	3	3	4	4	4	4	5	5
`BUFFER_LENGTH`	0	5	8	9	10	10	10	11	12	12	12	12	12	5	5	4
max L	1	6	9	10	11	11	11	12	13	13	13	13	13	6	6	5
def_lost	0	0	0	1	2	0	1	3	3	2	2	1	0	0	0	0

  From the above table we see that:
  1. max L = `BUFFER_LENGTH`+ 1
     Because of `CH_PERIOD` < `SND_PERIOD` the buffer cannot grow except in the initial phase of transmission.
  2. $L \geq$ `NLOSS` + 1
     We can produce losses only if the queue is greater then `NLOSS`+1 packets.
- *Result 2:* relation between `TXW_SIZE` and `BUFFER_LENGTH`.

`TXW_SIZE`	10	7	7	7	7
`NLOSS`	3	3	4	5	5
`BUFFER_LENGTH`	5	5	5	5	6
def_lost	0	0	0	0	1

The second table shows that we have a definitively lost packets if $L >=$ `NLOSS` $+ 1$ and if `TXW_SIZE` $>$ `BUFFER_LENGTH` $+ 1$. Losses occur only if we reach $L \geq$ `NLOSS` $+ 1$, so `BUFFER_LENGTH` $\geq$ `NLOSS` $+ 1$.

- *Result 3:* relation for `CH_PERIOD` $>$ `SND_PERIOD`.
  If we test our model for `CH_PERIOD` $>$ `SND_PERIOD`, i.e, (`CH_PERIOD`$= 2$ `SND_PERIOD`, `SND_PERIOD`$=10$,`CH_PERIOD`$=20$), than we obtain following results:
  1. queue L grows beyond every limit,
  2. def_lost is at most `NLOSS`.

3 Conclusion

Analysis and verification of parametrized models is difficult because verification problem is, in general, undecidable. In this paper, we showed our experiences and results of synthesis of the parameters for PGM protocol. We worked with three different tools - Uppaal, HyTech and TReX. By combined analysis using these tools we were able to find constraints on the parameters that satisfied desired property - the full recovery property. We proved that for `SND_PERIOD` $\geq$ `CH_PERIOD` $\wedge$ `TXW_SIZE` $\geq$ `RATE` $+$ `BUFFER_LENGTH` the property is satisfied. In comparison with previous work [BS03] our result was obtained automatically.

Analysis using Uppaal helped us to visually describe our model and simulate its behaviour. We used its verifier to prove a the full recovery property for a model with instantiating parameters. Verification of dead-lock detection proved consistancy of the model. For parametric analysis we used HyTech and TReX. HyTech had problems with termination so we verified only negation of the property and detected configurations that violates that property, that covered two cases - partial recovery and no recovery. Using TReX we obtained a full graph of symbolic configurations and observed relations between parameters. We syntetized parameters for all three cases - full recovery, partial recovery and no recovery. The results were consistant with those obtained manually or using HyTech and Uppaal.

References

[AAB00] A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th CAV*, volume 1855 of *LNCS*, pages 419–434. Springer Verlag, July 2000.

[ACHH93] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verfication of hybrid systems. In R.L.Grossman, A.Nerode, A.P.Ravn, and H.Rischel, editors, *Hybrid systems*, volume 736 of *LNCS*, pages 209–229. Springer Verlag, 1993.

[AHV93] R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. In *ACM Symposium on Theory of Computing*, pages 592–601, 1993.

[BBP02] B. Bérard, P. Bouyer, and A. Petit. Analysing the pgm protocol with uppaal. In P. Pettersson and W. Yi, editors, *Proceedings of the 2nd Workshop RT-TOOLS, Copenhagen (Denmark)*, August 2002.

[BCAS01] Ahmed Bouajjani, Aurore Collomb-Annichini, and Mihaela Sighireanu. Trex: A tool for reachability analysis of complex systems. In G. Berry, H. Common, and A. Finkel, editors, *Proceedings of Computer Aided Verification*, volume 2102 of *Lecture Notes Computer Science*, pages 368–372. Springer Verlag, June 2001.

[BFG$^+$00] M. Bozga, J.-C. Fernandez, L. Girvu, S. Graf, J.-P. Krimm, and L. Mounier. If: A validation environment for times asynchronous systems. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th CAV*, volume 1855 of *LNCS*, pages 543–547. Springer Verlag, July 2000.

[BL02] B. Boigelot and L. Latour. *ADVANCE Project Deliverable Report*, chapter Verifying PGM with infinitely many packets. LIAFA, 2002.

[BS03] Marc Boyer and Mihaela Sighireanu. Synthesis and verification of constraints in the pgm protocol. In Stefania Gnesi, editor, *Proceedings of International FME Symposium*, pages 264–281, September 2003.

[HHWT95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HYTECH. In *Proceedings of TACAS*, volume 1019 of *LNCS*, pages 41–71. Springer Verlag, 1995.

[HHWT97] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1):110–122, 1997.

[MS04] Petr Matoušek and Mihaela Sighireanu. Parametric analysis of the pgm protocol. submitted, 2004.

[PL00] P. Pettersson and K.G. Larsen. Uppaal2k. *Bulletin of the European Association for Theoretical Computer Science*, 70:40–44, February 2000.

[SFC$^+$01] Tony Speakman, Dino Farinacci, Jon Crowcroft, Jim Gemmell, Steven Lin, Dan Leshchiner, Michael Luby, Alex Tweedly, Nidhi Bhaskar, Richard Edmonstone, Todd Montgomery, Luigi Rizzo, Rajitha Sumanasekera, and Lorenzo Vicisano. PGM reliable transport protocol specification. RFC 3208, IETF, Decembre 2001. 111 pages.