QoS Evaluation for Real-time Distributed Systems Using the Probabilistic Model Checker PRISM

Takeshi NAGAOKA†, Akihiko ITO†, Kozo OKANO†, and Shinji KUSUMOTO†

†Graduate School of Information Science and Technology, Osaka University
Yamadaoka 1–5, Suita City, Osaka, 565-0871, Japan

The Internet is a best-effort network shared by a number of users, so there is no guarantee on the QoS properties such as network bandwidth, delay and throughput. Therefore, system developers preliminarily have to estimate the QoS by simulation techniques or mathematical analysis. Probabilistic model checking can evaluate performance, dependability and stability of information processing systems with random behaviors. PRISM is one of probabilistic model checkers. It handles automata with probabilities (discrete and continuous time Markov chains) and time elapse. This paper presents a discrete Markov chain for a real-time distributed system, and evaluates its QoS properties using the simulation function of PRISM. As the target real-time distributed system, we use an experimental system whose main subsystem is video data transmission which uses RTSP (Real Time Streaming Protocol) for the streaming protocol. The system has also ftp servers and clients which exploit tcp connections, as well as a packet generator that generates udp packets as background noise. Thus, the system involves several simultaneous connections; it may occur congestion. To validate the correctness of our model, we also model it in a model for the well-known network simulator NS–2. The paper gives the comparison of their simulation results. The comparison shows that the result of PRISM simulation is very similar to that of NS–2. The results show that the proposed method is useful to analyze the network performance.

Keywords: Real-time Distributed System, Model Checking, PRISM, Formal Verification

1 Introduction

Nowadays real-time distributed systems like streaming media systems are widely spreading. These systems require time based transmission such as QoS control to prevent interruption of packet transmission caused by network delay, packet loss, and so on.

Since the Internet is a best-effort network shared by a number of users, there is no guarantee on the QoS properties such as network bandwidth, delay and throughput. Therefore, system developers preliminary have to estimate the QoS by simulation techniques[2] or mathematical analysis[4].

Simulation techniques usually do not guarantee qualitative properties such as the maximum throughput and the minimum jitter, and so on, though they can calculate mean-values along typical traces. In general, these techniques use much resources to simulate accurately the target network systems. On the other hand, mathematical analysis is logically correct and needs fewer computation resources, but in many cases the based models are too simple and ideal; hence it is hard to apply the mathematical analysis to realistic applications.

Formal verification techniques, especially model checking techniques[3] are considered as promising techniques for information system developing due to their ability of exhaustive checking. Among them, probabilistic model checking can evaluate performance, dependability and stability of information processing systems with random behaviors. PRISM[1] is one of probabilistic model checkers. It handles automata with probabilities (discrete and continuous time Markov chains) and time elapse. Therefore, it is suitable for modelling the network systems.

This paper presents a discrete Markov chain for a real-time distributed system, and evaluate its QoS properties using the simulation function and the verification function of PRISM. As the target real-time distributed system, we use an experimental system shown in Fig.1. The main subsystem of the system is video data transmission which uses RTSP (Real Time Streaming Protocol) for the streaming protocol. The system has also ftp servers and clients which exploit tcp connections, as well as a packet generator that generates udp packets as background noise. Thus, the system involves several simultaneous sessions; it may occur congestion.

This paper analyzes newly throughput property. The analysis is performed on two levels; one is detail models, and the other is abstract level. On the detail level, several numeric analysis is performed, whereas on the abstract level, qualitative verification is performed. To validate the correctness of our model, we also model it in a model for the well-known network simulator NS–2[2], and give the comparison of their simulation results. The comparison shows that the result of PRISM simulation is very similar to that of NS–2.

As related works, several case-studies are performed using PRISM [8]–[10]. For example, Paper [9] deals with network protocol.

The rest of the paper is organized as follows. Sec.2 gives some backgrounds, especially overviews of PRISM and NS–2, as preliminaries. Sec.3 describes the example real-time distributed systems which uses streaming media. Explanations on several famous protocols for real-time video systems are also given. Sec.4 and 5 provide concrete models of the example network described in Prism and NS–2, respectively. Sec.6 and 7 show the results of comparison and gives discussions. Finally Sec.8 concludes the paper.

2 Preliminaries

This section simply describes Prism and NS–2 as well as network protocols for net streaming.

2.1 Probabilistic Model Checking tool PRISM

Here, we simply describe overview of probabilistic Model Checking tool PRISM[1].

A model checking tool usually has two inputs, a model $\mathcal{M}$ and a logical expression p. The model is typically a transition system which represents behavior of the system to check; while the logical expression is a temporal logic expression which represents a property to check. The typical output of the model checking tool is whether the logical expression is valid on the model ($\mathcal{M} \models p$). Some model checker outputs a counter example when p is invalid.

The inputs of PRISM include the following three kind of transition system as a model:

Discrete-time Markov chains (DTMC);

Continuous-time Markov chains (CTMC); and

Markov decision processes (MDP).

Each of three systems is a probabilistic transition system (Markov chain). The inputs of PRISM also include Probabilistic Computation Tree Logic (PCTL)[11] for DTMC and MDP, and Continuous Stochastic Logic (CSL)[12] for CTMC. They are CTL based logics enchanted with probability.

PRISM has several analysis modes: simulation mode, numerical analysis mode, and verification mode. Using the simulation mode, we can observe the behavior of the given model system visually. This mode uses only a model as input. Numerical analysis mode can evaluate the value of uncertain variable specified with PCTL or CSL based on the model. Such numerical analysis is considered as a kind of parametric model checking[13]. PRISM can draw a graph with several trials of such numeric analysis. Verification mode is like typical model checking except that PRISM cannot output counter examples.

In this paper, we use DTMC's as the model of the network. Here, we describe more precisely on a DTMC. Formally, a DTMC D is a tuple (S, s_init, P, L), where

S is a set of states ("state space") ;

$s\_init \in S$ is the initial state;

$P : S \times S \to [0, 1]$ is the transition probability matrix where $\Sigma_{s' \in S} P(s, s') = 1$ for all $s \in S$; and

$L : S \to 2^{AP}$ is function labelling states with atomic propositions.

PRISM allows a transition to specify an action and updating expressions on D, where D is a set of variables with finite domain. In other words, a DTMC of PRISM is a kind of extended automaton with probabilities. Usually, one execution of a transition is translated into a unit time of time elapse (a tick event). Such scheme is known as digital clock view of DTMC, however, using an integer variable (with the upper-bound) explicitly as a clock variable, we can also represent a system with discrete time in DTMC. In this paper, we use the latter scheme to avoid the state explosion problem.

2.2 Network Simulator NS–2

The Network Simulator (ns) is a network simulator developed by Virtual InterNetwork Testbed (VINT) project. It has been widely used[16].

NS–2 is the successor version of ns. It is open source software. NS–2 uses two languages: C++ for core libraries; and OTcl (Object-oriented Tcl) for scenario description and network configuration. A new protocol which is not pre-defined in the core libraries has to be written by a user.

2.3 Protocols for Net-streaming

Here, we simply summarize typical protocols used in the Internet. Typical protocols used in the Internet have a congestion control mechanism in order to avoid network congestion. For example, TCP (Transmission Control Protocol) uses AIMD (Additive Increase Multiplicative Decrease) type window-flow control as such the mechanism. It controls the data size of sending packets based on the current available bandwidth. Such scheme has an advantage for the correct data transmission. It, however, allows delays, which is not suitable for real-time data transmission. Therefore, RTSP (Real Time Streaming Protocol) is used for real-time application. RTSP is a protocol for the Internet streaming of voice and movies, on TCP/IP network. Famous congestion control mechanisms for RTSP are RAP (Rate Adaptation Protocol)[5] and TEAR (TCP Emulate At Receivers)[6]. Recently, TFRC (TCP-Friendly Rate Control)[7] attracts attention. Hence, this paper models TFRC.

2.3.1 RTSP

RTSP is one of typical protocols working at end-to-end. RTSP has five states, called SETUP, PLAY, RECORD, PAUSE and TEARDOWN. RTP (Real-time Transport Protocol) is responsible for transmission of stream-data. It determines the throughput of RTP based on rate control scheme of TFRC using the report message of RTCP (RTP Control Protocol).

2.3.2 TCP Friendly Rate Control TFRC

TFRC is a rate control scheme for fairness between RTP and TCP. It controls the rate in order to avoid bad effects on existing TCP flows in the same network, which increases total effectiveness of the whole network. TFRC controls the rate using the report message of RTCP. The report message contains loss of packets and jitters, which can be estimated via the sequence number of received RTP packets and time stamps, respectively. RFC3448 describes the following formula for determining the throughput:

$$X = \frac{s}{R*\sqrt{2*b*p/3} + (t\_RTO*(3*\sqrt{3*b*p/8}*p*(1+32*p^2)))}$$

where the unit of X is byte/seconds. The parameter of the formula is summarized in Table 1.

The calculated throughput is a rate with which a RTSP server should send packets considering the network congestion at the time. Therefore weighted average values of the pa-

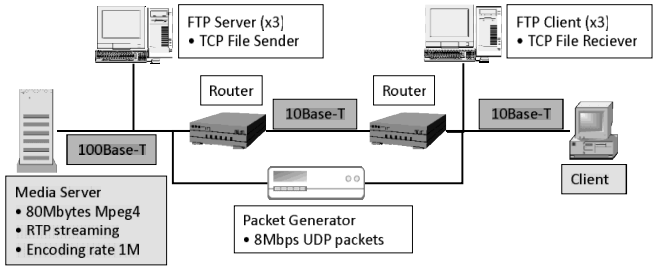

Figure 1: A Configuration of Experimental System

rameters in a short period are applied into the equation. Paper [7] also defines the calculation methods for the parameters.

When the value of X is less than the bandwidth, TFRC lets RTSP set the value as throughput.

3 Example Network Configuration and Protocols

In this section, we introduce an example of a real-time distributed system. As the example system, we select a video data transmission system[14] shown in Fig.1. The system is composed of a pair of a video server and its client, three pairs of FTP servers and their clients, and a packet generator, which are connected to each other through routers located at the middle of Fig.1. The routers are connected through the 10Base–T Ethernet, which is considered as a bottleneck of packet transmission. In the considering scenario, the video server sends 80MB of video data with throughput of 1Mbps using the rate control of TFRC. After 100 seconds from the start of the video streaming, FTP servers and clients start their data transmission through TCP sessions. Also, the packet generator always sends UDP packets with the throughput of 8Mbps as background noise.

4 PRISM Model

We generate two PRISM models for the example of Fig.1. The first one is modelled in detail, and another one is based on the simulation results on the first model but is more simplified model. First, we apply PRISM simulation to the detailed model to show the correctness of our model and to obtain the data to generate the simplified model. Next, the simplified model is used to verify the qualitative properties of the throughput.

Table 1: Parameters of the Throughput Estimation Formula

R[seconds]	Round trip time
p[%]	A packet loss rate
s[byte]	Packet size
b[number of times]	A number of packets acknowledged by a single TCP acknowledgment
t_RTO[seconds]	A TCP retransmission timeout value

4.1 The Detailed Model

In the detailed model, we have to model packet loss rates and round trip time of the RTP packets as well as the buffer control of the routers, the congestion control of transmission protocols, and a time elapsing mechanism.

4.1.1 Time Elapsing

In order to manage time elapsing in the detailed model, we declare an integer variable to represent time. Time elapsing is based on events such as packet transmission. In the model, each module registers time of occurrence of the next event, and when all modules register the time, the model perform time elapsing into the latest time of the registered event from the current time. After time elapsing, a corresponding module performs the registered event and registers time of the next event again.

The module to manage time elapsing contains two variables and is implemented as ten lines of code.

4.1.2 Buffer Control of the Routers

Buffer control of the routers is modelled as a queue which holds transferring packets. In the model, the current queue length is managed with an integer variable. Enqueue and dequeue behaviors are described in the model as operations. Also, regardless of the current queue length, enqueueing packets are dropped with certain probability. In order to construct the module of buffer control, we have to specify maximum length of the queue, a packet transfer rate of the link between the routers, and constant probability to drop enqueueing packets as parameters.

In the enqueue operation, if the current queue length becomes larger than the maximum one, the enqueueing packet is considered to be dropped (drop tail). The dequeue operation is abstracted; together with the time elapsing operation, a number of packets are output from the queue at a time according to the packet transfer rate of the link.

The buffer control module shown in Fig.2 contains ten variables and is implemented as about 80 lines of code, where the module also manages the history of packet loss intervals used for the congestion control. In Fig.2, the ten variables are firstly declared. Several actions are defined in CCS like expressions with probabilities. For example, the expression

```
[ENQFTP1]  (q_len <= MAXQLEN - pnum_ftp1)
  -> 1 - P_LOSS_RATE : (q_len' = q_len + pnum_ftp1);
   + P_LOSS_RATE : true;
```

stands for that when an action $ENQFTP1$ occurs and $(q\_len \leq MAXQLEN - pnum\_ftp1)$ holds, the variables q_len is updated to $q\_len + pnum\_ftp1$ with probability $1 - P\_LOSS\_RATE$, or do nothing with probability P_LOSS_RATE.

4.1.3 Congestion Control

In the model, TCP and RTP sessions can handle congestion control.

The slow-start and congestion avoidance behaviors of TCP are embedded to our TCP model. For each connection of the TCP, we declare two integer variables to manage the slow-start threshold and the window size. For each connection of

```
module Router

  q_len : [0..MAXQSIZE] init 0; //The current queue length

  //The history of packet loss intervals
  int_p_loss0 : [0..10000] init 0;
  int_p_loss1 : [0..10000] init 10000;
  int_p_loss2 : [0..10000] init 10000;
  int_p_loss3 : [0..10000] init 10000;
  int_p_loss4 : [0..10000] init 10000;
  int_p_loss5 : [0..10000] init 10000;
  int_p_loss6 : [0..10000] init 10000;
  int_p_loss7 : [0..10000] init 10000;
  int_p_loss8 : [0..10000] init 10000;

  //A flag to observe whether the packet loss occurs burstly or not
  p_loss_flag : bool init false;


  //When the queue length does not reach to its maximum
  //Transfering packets are dropped with certain probability
  [ENQMS] (q_len <= MAXQLEN - ms_pnum)
    -> 1 - P_LOSS_RATE :
       (q_len' = q_len + ms_pnum) & (p_loss_flag' = false) &
       (int_p_loss0' = int_p_loss0 + 1)
     + P_LOSS_RATE :
       (int_p_loss0' = 0) & (int_p_loss1' = int_p_loss0) &
       (int_p_loss2' = int_p_loss1)&(int_p_loss3' = int_p_loss2)&
       (int_p_loss4' = int_p_loss3)&(int_p_loss5' = int_p_loss4)&
       (int_p_loss6' = int_p_loss5)&(int_p_loss7' = int_p_loss6)&
       (int_p_loss8' = int_p_loss7);

  //When the queue length reaches to its maximum
  [ENQMS] (q_len > MAXQLEN - ms_pnum) & (!p_loss_flag)
    -> (q_len' = MAXQLEN) & (int_p_loss0' = 0) &
       (int_p_loss1'=int_p_loss0+1)&(int_p_loss2' = int_p_loss1)&
       (int_p_loss3' = int_p_loss2)&(int_p_loss4' = int_p_loss3)&
       (int_p_loss5' = int_p_loss4)&(int_p_loss6' = int_p_loss5)&
       (int_p_loss7' = int_p_loss6)&(int_p_loss8' = int_p_loss7)&
       (p_loss_flag' = true);

  // When the packet loss occurs burstly
  //       (do not update the history of packet loss intervals)
  [ENQMS] (q_len > MAXQLEN - ms_pnum) & (p_loss_flag)
    -> (q_len' = MAXQLEN) & (p_loss_flag' = (q_len = MAXQLEN));



  //The ENQUEUE operations for the three FTP sessions
  [ENQFTP1]  (q_len <= MAXQLEN - pnum_ftp1)
    -> 1 - P_LOSS_RATE : (q_len' = q_len + pnum_ftp1);
     + P_LOSS_RATE : true;

  [ENQFTP1]  (q_len > MAXQLEN - pnum_ftp1)
    -> (q_len' = MAXQLEN);

  [ENQFTP2] (q_len <= MAXQLEN - pnum_ftp2)
    -> 1 - P_LOSS_RATE : (q_len' = q_len + pnum_ftp2)
     + P_LOSS_RATE : true;

  [ENQFTP2]  (q_len > MAXQLEN - pnum_ftp2)
    -> (q_len' = MAXQLEN);

  [ENQFTP3]  (q_len <= MAXQLEN - pnum_ftp3)
    -> 1 - P_LOSS_RATE : (q_len' = q_len + pnum_ftp3)
     + P_LOSS_RATE : true;

  [ENQFTP3]  (q_len > MAXQLEN - pnum_ftp3)
    -> (q_len' = MAXQLEN);


  // DEQUEUE is executed together with the time elapsing event
  [TIMER] (q_len != 0)
    -> (q_len' = max(0, q_len - floor(min_lookahead *
                            (PACKETPERMSEC -PACKETPERMSEC2))));

  [TIMER] (q_len = 0) -> true;

endmodule
```

Figure 2: The Module of Router Described with PRISM Language

the TCP, we declare six variables and the behavior is implemented as about 30 lines of code.

In the transmission of RTP, a packet transmission rate is calculated from the throughput equation defined in [7]. To use the equation, we also have to model a packet loss rate and round trip time of the RTP session (see Sec.4.1.4 and 4.1.5). The module related to RTP contains six variables and is describe with about 40 lines of code.

In our model, the packet transmission behaviors of TCP and RTP are abstracted as a number of packets are transmitted simultaneously.

4.1.4 Round Trip Time

In our model, round trip time is obtained using physical delay and delay in the router. The delay in the router is calculated as the time to transmit all packets currently buffered in the router. Therefore, we obtain the delay in the router using current queue length and a packet transmission rate of the link.

4.1.5 Packet Loss Rate

In the TFRC specification[7], a packet loss rate is calculated using intervals of packet loss. To avoid the loss rate varying rapidly, a history of the packet loss intervals is used to calculate it. Our model declares eight integer variables to manage the history. In the calculation of the loss rate, recent intervals in the history are weighted heavily.

4.2 The Simplified Model

The detailed model described in Sec.4.1 is too complicated to verify its qualitative properties using model checking. Here, we create a simplified model based on simulation results on the detailed model to perform model checking. Using the simplified model, we verify the minimum and maximum throughput of the media server.

The simplified model is based on the throughput equation defined in [7]. In order to create the simplified model, we have to measure distributions of round trip time and packet loss rates on the detailed model. From the obtained distributions, we create discrete probability distributions of them. The simplified model decides its round trip time and a packet loss rate according to the probability distributions. Finally, by verifying occurrence probability of the throughput on the simplified model, we can obtain the minimum and maximum throughput.

The simplified model is implemented as 50 lines of code.

5 NS–2 Model

We use a simulation scenario shown in Fig.3. Values associated with each link represent transfer rates or physical delay of the link. TCP and UDP sessions in the scenario are derived from pre-defined protocol set in NS–2. In this example, we have to modify the pre-defined RTP protocol so that it can control network congestion because the pre-defined RTP protocol cannot control it. According to TFRC3448[7], we have modified the behaviors of RTP senders and receivers as follows; the receiver can return appropriate feedback packets to

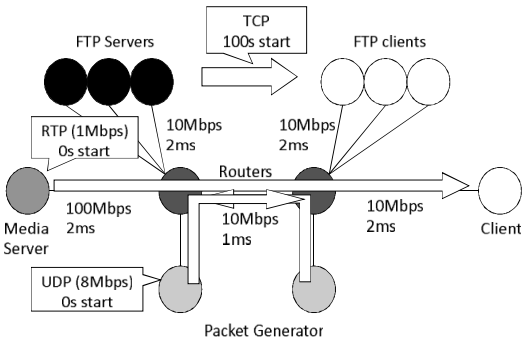

Figure 3: A Simulation Scenario Model of NS–2

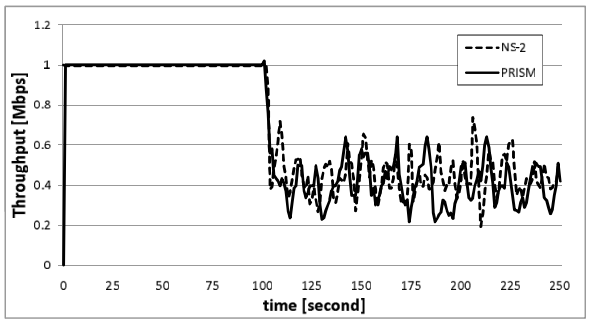

Figure 4: Comparison of the Througput Obtained with NS-2 and PRISM

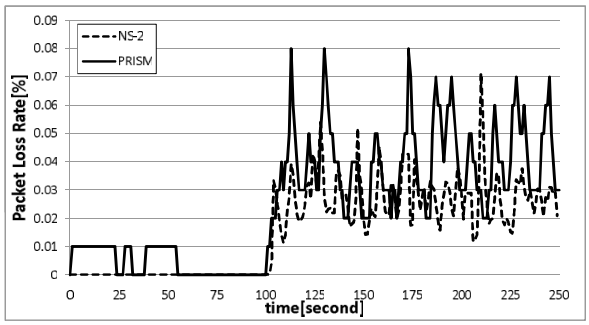

Figure 5: Comparison of Packet Loss Rates

the sender, which include information of a packet loss rate, network delay and so on, and the sender can control a transfer rate due to the feedback packets. In our implementation, we have added about 500 lines of code to eight files (twelve methods) of NS–2 sources.

6 Experiments

We have performed some experiments using our PRISM and NS–2 models described in Sec.4 and 5. The experiments were performed under an environment of Windows Vista 64 bit, Intel Core 2 Duo 2.33GHz, and 2.00GB of M.M.

In the experiments, we assume packet transmission parameters as follows: the packet size is 500 Byte, the number of packets acknowledged by a single TCP acknowledgment is one, and the TCP retransmission timeout value is $4 \times RTT$ second. Also we consider three scenarios with respect to the buffer size of routers. In the scenarios, the buffer sizes are 32, 64, and 128 KB respectively.

In the paper, we have performed two experiments. In the first experiment, we performed one trial run for PRISM and NS–2 simulation, respectively, and compared the simulation results. In the second experiment, we performed about 400 trial runs for PRISM simulation, and created a more simplified PRISM model based on the simulation results. Using the simplified model, we verify the minimum and maximum throughput of the RTP session. To make the simulation results more reliable, we chose about 400 thousands as the number of trial runs, which is default number trial runs for PRISM simulation.

6.1 Comparison of the Simulation Results

Here, we present the simulation results of the buffer size of 64KB in Fig.4 and 5. Throughput in the graph means the average throughput within one second, and a packet loss rate means a calculated value at the time as defined in [7]. In the scenario of the example, file transmission starts after 100 seconds from the start of RTP session, and this causes the network congestion. Consequently, the throughput of the RTP session goes down and the packet loss rate of it comes up. The simulation results of Fig.4 and Fig.5 show that our PRISM model and NS–2 model behave similarly even if the network congestion occurs.

To analyze the correctness of our PRISM model in detail, we compare the average, variance, minimum and maximum of throughput measured by PRISM and NS–2. Table 2 represents the analyzed data. Though the maximum throughput of PRISM and NS–2 are similar in all scenarios, the average and minimum throughput are a little bit different in the scenarios of 32KB and 128KB. We think one of the reasons is that we abstract a packet sending mechanism in the PRISM model, that is, when the packet transmission rate is high, our model transmits a number of packets at a time. We think this may cause the differences of behaviors between the PRISM model and NS–2 one. However, Table 2 shows that the behaviors of these models seem to be totally similar.

For one trial run of the PRISM simulation, it takes about two or three seconds, while it takes about three minutes in the simulation of NS–2.

6.2 Verification results for the simplified model

Here, we verify minimum and maximum throughput that the media server can provide in the worst case and best case. In the verification, we use a simplified model based on the

Table 2: Comparison of the Analyzed Data

Buffer size	32KB		64KB		128KB	
Model	NS-2	PRISM	NS-2	PRISM	NS-2	PRISM
Maximum	607	588	738	640	995	980
Minimum	203	48	193	216	33	144
Average	242	372	443	401	473	367
Variance	11	13	10	11	12	12

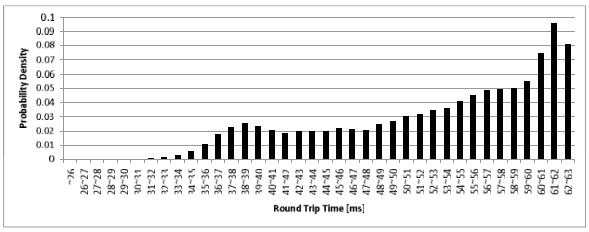

Figure 6: The Discrete Probability Distribution of Round Trip Time

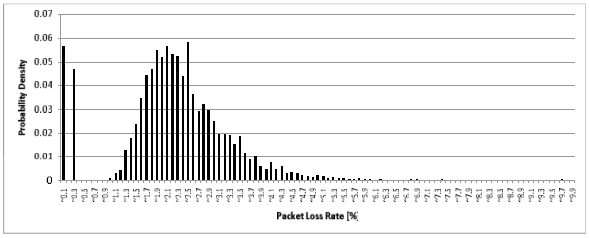

Figure 7: The Discrete Probability Distribution of Packet Loss Rates

simulation results on the detailed PRISM model. The simulation results should contain discrete probability distributions of the round trip time and the packet loss rates. To obtain reliable simulation results, we perform about 400 thousands of trial runs of PRISM simulation, which is default number of the PRISM simulation.

When the buffer size is 64KB, the discrete probability distributions of the round trip time and the packet loss rates are shown in Fig.6 and 7, respectively.

The verification property to be checked is given as follows; $P =?[Throughput < xUThroughput > x]$. This property means the probability of $Throughput$ being the value of x, where $Throughput$ is a variable to manage the throughput of the RTP session. The verification is performed with varying x from 0 to 1000. Figure 8 represents occurrence probability of the throughput. The graph represents the probability that the throughput becomes greater than or equals to value of x-axis.

In the Fig.8, when the throughput is less than 116Kbps, the probability keeps the value one, and until the throughput becomes 1000Kbps the probability is greater than zero. From the verification result, it is guaranteed that the throughput of the media server is greater than 116Kbps in the worst case. Also, the result shows that the media server can transmit packets with the rate of 1000Kbps in the best case. The simulation results of PRISM and NS–2 do not contradict to this verification result.

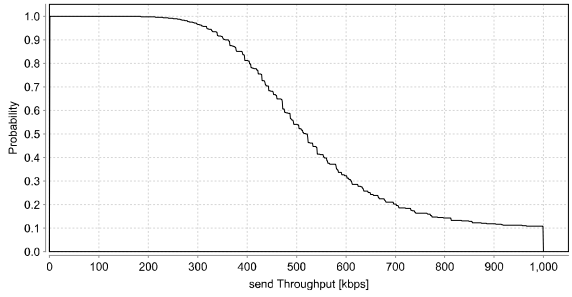

Figure 8: Occurrence Probability of Throughput

Although it takes about 10 hours to perform about 400 trial runs for PRISM simulation on the detailed model, in the verification on the simplified model, it takes about 40 seconds.

7 Discussion

In our case study, we have verified throughput of the media server using the detailed model and simplified model. The detailed model includes behaviors of the buffer control of the router and behaviors of the RTP, TCP and UDP protocols in detail. Although the detailed model is too complicated to verify qualitative properties of the system, we can measure some network properties using the simulation function of PRISM. On the other hand, the simplified model focuses on the verification of the minimum and maximum throughput, which is considered as one of important properties of the real-time systems. The simplified model is based on the throughput equation shown in Sec.2, where the dominant parameters such as a packet loss rate and round trip time are derived from the simulation results on the detailed model. The verification on the simplified model has been performed in a few seconds.

From the experiments, we can reconstruct our method as follows. The method will be useful to verify the minimum and maximum throughput that a RTSP connection can provide in an environment where some TCP and UDP connections exist simultaneously. In our method, first, we create a detailed model of the system which should have a buffer control mechanism of a router considered to be a bottleneck, as well as network congestion control mechanisms of RTSP and TCP. Next, using the simulation function of PRISM, we measure distributions of round trip time and a packet loss rate on the detailed model, and create discrete probability distributions of them. Based on the probability distributions and the throughput equation shown in Sec.2, we create a simplified model. The simplified model decides its round trip time and packet loss rate according to the probability distributions of them. Finally, the minimum and maximum throughput can be verified on the simplified model by probabilistic model checking with PRISM.

In our simplified model, we assume that packet loss rates and round trip time are decided independently. In fact, these parameters are not independent, i.e., when network congestion occurs, packet loss rates and round trip time tend to increase simultaneously. In the future work, we need to create a simplified model which represents their dependency.

Though simulation with NS–2 or PRISM can evaluate the system's performance, the simulation results depend on paths evaluated in the simulation. Therefore, from the simulation results, we cannot prove qualitative properties of the system. If we obtain a occurrence probability of throughput from the results of Fig.4, we cannot say it is reliable. In the paper, we created the simplified model and we can verify the qualitative properties of the system using model checking.

8 Conclusion

This paper presents a discrete Markov chain for a real-time distributed system, and evaluate its QoS properties using the simulation function of PRISM. To validate the correctness of our model, we also model it in a model for the well-known network simulator NS–2, and give the comparison of their simulation results. The comparison shows that the result of PRISM simulation is very similar to that of NS–2. It shows that the proposed method is useful to analyze the network performance. We think that such analysis is useful for other kind of network analysis.

The future works include automatic derivation of simplified model suitable for model checking analysis. Many abstraction techniques are proposed for model checking. We want to apply such techniques to the process.

Acknowledgments

This research is partially supported by the research grant of the Okawa Foundation, (ref. no.08–09) and Grant-in-Aid for Scientific Research (C)(21500036). Also, this work is being conducted as a part of Stage Project, the Development of Next Generation IT Infrastructure, supported by Ministry of Education, Culture, Sports, Science and Technology. We thank Dr. S. Hiromori for useful advices for NS–2.

REFERENCES

[1] A. Hinton and M. Kwiatkowska and G. Norman and D. Parker: "PRISM: A Tool for Automatic Verification of Probabilistic Systems," In Proc. of the 12th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06), vol.3920, pp.441–444, 2006.

[2] "The Network Simulator - ns–2," available at http://www.isi.edu/nsnam/ns/.

[3] E. M. Clarke, O. Grumberg, and D A. Peled: "Model Checking," MIT Press, 2000.

[4] E. J. Kim, K. H. Yum, and C. R. Das : "Calculation of Deadline Missing Probability in a QoS Capable Cluster Interconnect," IEEE International Symposium on Network Computing and Applications (NCA'01), pp.36, 2001.

[5] R. Rejaie, M. Handley, and D. Estrin : "RAP: An end-to-end ratebased congestion control mechanism for realtime streams in the internet," in Proc. of INFOCOM 1999, pp.1337–1345, 1999.

[6] I. Rhee, V. ozdemir, and Y. Yi : "TEAR: TCP emulation at recievers – flow control for multimedia streaming," NCSU Technical Report, 2000.

[7] M. Handly, S. Floyd, J. Padhye, and J. Widmer : "TCP friendly rate control (TFRC): protocol specification," Request for Comments (RFC)3448, 2003.

[8] M. Kwiatkowska, G. Norman, D. Parker and J. Sproston : "Performance Analysis of Probabilistic Timed Automata using Digital Clocks," In Proc. Formal Modeling and Analysis of Timed Systems (FORMATS'03), 2003.

[9] M. Kwiatkowska, G. Norman and J. Sproston : "Probabilistic Model Checking of the IEEE 802.11 Wireless Local Area Network Protocol," In Proc. PAPM/PROBMIV'02, volume 2399 of LNCS, pp.169–187, 2002.

[10] M. Kwiatkowska, G. Norman, J. Sproston and F. Wang : "Symbolic Model Checking for Probabilistic Timed Automata," In Proc. FORMATS/FTRTFT'04, volume 3253 of LNCS, pp.293–308, 2004.

[11] H. Hansson and B. Jonsson : "A logic for reasoning about time and probability," Formal Aspects of Computing, 6(5)pp.512–535, 1994.

[12] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton : "Verifying continuous time Markov chains," LNCS 1026, pp.499–513, 1996.

[13] R. Alur, T. A. Henzinger and M. Y. Vardi : "Parametric Real-time Reasoning," Proc. 25th ACM Annual Symp. on the Theory of Computing (STOC'93), 592–601,1993

[14] Y. Taniguchi, A. Ueoka, N. Wakamiya, M. Murata, and F. Noda, "Implementation and Evaluation of Proxy Caching System for MPEG-4 Video Streaming with Quality Adjustment Mechanism," in Proc. of The 5th AEARU Workshop on Web Technology, pp.27–34, 2003.

[15] E. Nagai, K. Okano, S. Kusumoto: "Performance verification considering the network delay for Real-time Distributed Systems with the probabilistic model checker PRISM (in Japanese)," IEICE Technical Report, vol.105, no.597, pp.19–24, 2006.

[16] D. Mahrenholz and S. Ivanov: "Real-Time Network Emulation with ns-2," In 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications, ser. 8. IEEE Computer Society, October 2004, pp. 29-36.

[17] M. Yajnik, S. Moon, J. Kurose, and D. Towsley: "Measurement and modeling of the temporal dependence in packet loss," In Proc. INFOCOMM'99, 1999.