IEICE TRANS. INF. & SYST., VOL.E94–D, NO.11 NOVEMBER 2011

PAPER

Modeling and Analysis for Universal Plug and Play Using PIPE2

Cheng-Min LIN[†], ***Member***, **Shyi-Shiou WU**[†a)], ***and*** **Tse-Yi CHEN**[†], ***Nonmembers***

SUMMARY Universal Plug and Play (UPnP) allows devices automatic discovery and control of services available in those devices connected to a Transmission Control Protocol/ Internet Protocol (TCP/IP) network. Although many products are designed using UPnP, little attention has been given to UPnP related to modeling and performance analysis. This paper uses a framework of Generalized Stochastic Petri Net (GSPN) to model and analyze the behavior of UPnP systems. The framework includes modeling UPnP, reachability decomposition, GSPN analysis, and reward assignment. Then, the Platform Independent Petri net Editor 2 (PIPE2) tool is used to model and evaluate the controllers in terms of power consumption, system utilization and network throughput. Through quantitative analysis, the steady states in the operation and notification stage dominate the system performance, and the control point is better than the device in power consumption but the device outperforms the control point in evaluating utilization. The framework and numerical results are useful to improve the quality of services provided in UPnP devices.

key words: *UPnP, peer-to-peer communication, Markov chains, embedded systems*

1. Introduction

UPnP provides efficient peer-to-peer network connectivity for personal computers and embedded systems [1]. It has been successfully applied to many products, such as Interactive TV [2], A/V [3], [4], media servers [5] and Internet radio. A UPnP system consists of devices and control points. A UPnP device is a server that advertises its services to control points. A control point can search for a specific service on the network. UPnP devices automatically announce their network address and supported device and services types, enabling clients that recognize those types to immediately begin using the device without user intervention. To provide plug and play, the UPnP system commits to five phases:

1. Discovery: control points search for UPnP devices.
2. Description: a control point requests the device description provided by the device, when the control point finds an interesting device.
3. Control: a control point controls one or more of services provided by a device.
4. Eventing: a control point will be notified when device's state is changed.
5. Presentation: each UPnP device provides a document written in standard hypertext markup language (HTML).

Although many studies have addressed UPnP implementation, little attention has been given to performance evaluation [6], [7]. A systematic way of modeling UPnP's behaviour and evaluating the system performance is still lacking. This paper presents a framework for evaluating UPnP's performance in power consumption, system utilization and network throughput using Markov chains technology. We first utilize the GSPN framework and the PIPE2 [8] tool to model the UPnP system, and then propose a formal framework to evaluate the system performance of the UPnP system. We also introduce a standard approach for using GSPNs as a high-level model for generating Continuous-Time Markov Chains (CTMCs) and then use Markov Reward Models (MRMs) to compute the performance for the UPnP system. These analyses can help us to understand the UPnP device's runtime behaviour and enhance the reliability of UPnP products, and it is useful to improve UPnP device's quality.

The rest of this paper is organized as follows. Section 2 presents the system model of CTMCs used in analyzing performance and reliability. Section 3 introduces the modeling interaction behaviour between a device and a control point for the UPnP protocol. Quantitative analysis is illustrated in Sect. 4. Concluding remarks are presented in Sect. 5.

2. System Model

Most systems use Markov chains introduced by A.A. Markov in 1907 to analyze performance and reliability, especially CTMCs. According to the definition presented by Kwiatkowska et al. [9], a labelled CTMC is a tuple, $\mathbf{C} = (S, \bar{s}, \mathbf{R}, L)$, consisting of four components as follows.

1. S is a finite set of states;
2. $\bar{s} \in S$ is the initial state;
3. $\mathbf{R} : S \times S \rightarrow \mathbb{R}_{\geq 0}$ is the transition rate matrix, where $\mathbb{R}_{\geq 0}$ denotes positive real number;
4. $L : S \rightarrow 2^{AP}$ is a labelling function which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions (AP) that are valid in the state.

For CTMCs, we should change the transition rate matrix $\mathbb{R}$ into the infinitesimal generator matrix $\mathbb{Q}$. Hence, we calculate the matrix $\mathbb{R}$ to obtain the matrix $\mathbb{Q}$. The calculation equation is given by

Manuscript received October 13, 2010.
Manuscript revised April 26, 2011.

[†]The authors are with Graduate Institute of Electrical Engineering, Nan Kai University of Technology, Tsao Tun, 542, Nan Tou County, Taiwan, R.O.C.
a) E-mail: sswu@nkut.edu.tw

DOI: 10.1587/transinf.E94.D.2184

$$\mathbb{Q}(s,s') = \begin{cases} \mathbb{R}(s,s') & if\, s \neq s' \\ -\sum_{s'' \neq s} \mathbb{R}(s,s'') & otherwise. \end{cases} \quad (1)$$

For steady-state solutions of CTMCs, the computation of steady-state probability vector π can be solved using the following linear equation and normalization condition,

$$\mathbf{0} = \pi\mathbb{Q}, \pi\mathbf{1} = \mathbf{I}, \quad (2)$$

where the unit vector $\mathbf{I} = [1, 1, \ldots, 1]^T$. In performance evaluation, we are concerned more with reliability, availability, and throughput than the state probability. Hence, Markov reward models (MRMs) provide a unifying framework for obtaining these performance parameter values using reward rate assignment [10]. A reward rate is assigned based on the system requirements. Let the reward rate r_i be assigned to the state $i \in S$. For steady-state of CTMCs, based on Eq. (2) the expected reward rate, $E[Z(t)]$, in the limit as $t \to \infty$ can be calculated using

$$\mathbf{E}[\mathbf{Z}(\infty)] = \mathbf{E}[\mathbf{Z}] = \sum_{i \in S} r_i \pi_i. \quad (3)$$

3. Modeling and Analysis

In this section, we will introduce the framework based on GSPN model, including modeling UPnP, reachability decomposition, semantic analysis, and reward assignment. PIPE2 [8] is an open source, platform-independent tool for creating and analyzing GSPN. PIPE2 provides an easy-to use graphical user interface, and it can ease the work of creation, saving, loading and analysis of Petri nets. The PIPE2 is used modeling UPnP first. Then, a reachability graph is generated by PIPE2. We decompose the graph into three stages, including initialization, understanding, and operation & notification. For operation & notification stage, the GSPN model is used to process semantic analysis. Lastly, a reward assignment is used in the CTMC model to calculate system performance.

3.1 Modeling UPnP Protocol

A Petri net (PN) is a bipartite graph denoted by $PN = \{P, T, D^-, D^+, m0\}$. It consists of two types of nodes called places, P, and transitions, T. Arcs used to connect between a places and a transition can be divided into two categories: input, D^+, and output, D^-. An initial marking $m0$, a member of marking set M, used to describe the number of tokens in each place in initial state. GSPNs generalize PNs to provide two types of transitions. The first one is called timed transitions which produces exponentially distributed firing time. The second type is called immediate transitions, which has priority over the timed transitions. A priority can be assigned immediately if the priority issue is required. The GSPN model also provides inhibitor arcs. An inhibitor arc is indicated by a small circle instead of an arrowhead, it will disable the transition from firing when connected places contain a token.

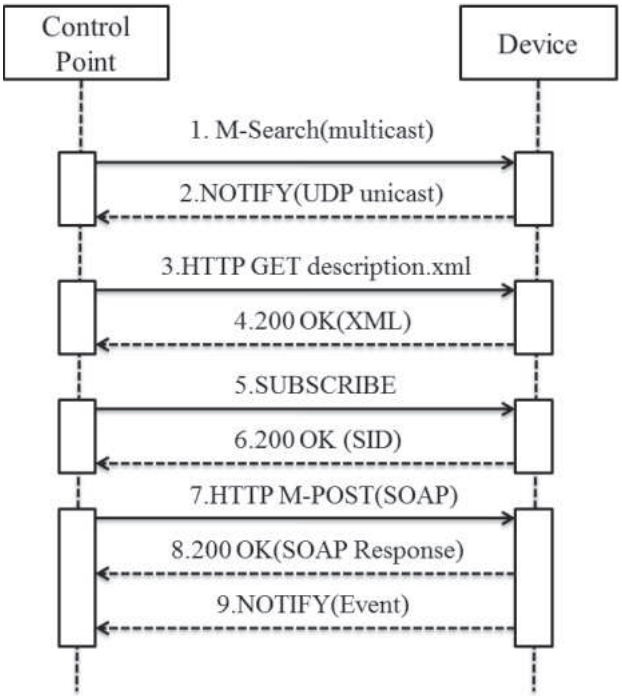

Fig. 1 The sequence diagram of the UPnP protocol [1].

UPnP products can be divided into two groups, namely control points and devices. A control point is a client system. As shown in Fig. 1, clients can use the control point to explore UPnP supported service, and then interact with the UPnP devices using their provided services. Figure 2 shows the GSPN model of the UPnP protocol. The model consists of 32 places and 23 transitions. P0 and P1 represent the entry point of the control point and the device, respectively. T0 and T1 are initial tasks. The task first calls the UPnPInit function.

As a control point finishes its initial task, it will call the UpnpRegisterClient function to register a callback function for providing a handle routine to process the users actions. In the other words, P4, P2, and P30 respectively obtain a token after T0 is fired. Next, the control point uses UpnpSearchAsync function to start searching for interesting device in a UPnP network when P4 holds the token. Now, the control point sends a multi-cast message of M-Search to all devices connected in the UPnP network. After that, T2 will be fired and then P3 will be given a token to represent the device has received the message.

For a UPnP device after executing the UPnPInit function, the UpnpRegisterRootDevice function is called to register a root device to provide services for processing control point requests. Now, P5 will hold a token after T1 fired. When the device has received a message of M-Search, P3 holds a token. Hence, the device will reply to the control point if the device is the interesting one searched by the control point. Now P6 has a token to represent transmitting a reply message. T4 is fired when transmitting the message is completed and then P7 will obtain a token. The scenarios of

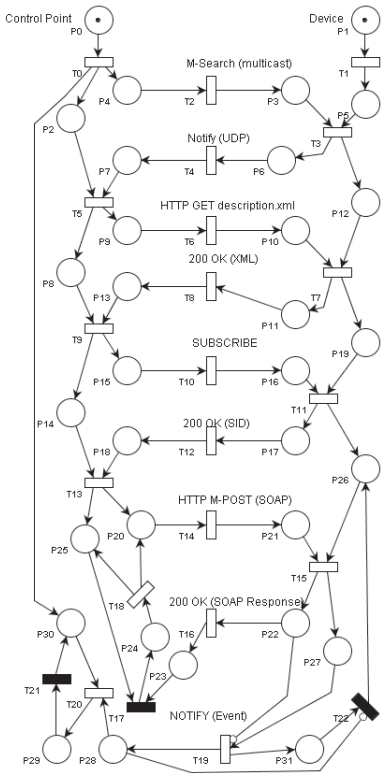

Fig. 2 Modeling the UPnP protocol.

transmitting other messages are similar. Hence, we omit the detailed description of the actions.

As discussed above, P2 to P19 represent serial interactions for establishing a connection between a control point and a device from the control point searching for a device to the control point subscribing events from the device. The sequences from P20 to P26 mean that a control point invokes an action, and the sequences from P27 to P31 signify handling an event for changing the devices state. We use a feedback mechanism for these actions to be kept alive, including invoking an action and handling an event. In Fig. 2, there are three feedbacks to let P24, P26, and P30 continue obtaining tokens. The feedback of P24 in the control point is to process receiving a SOAP response from the UPnP device. The feedback of P26 indicate that the device waits a SOAP action triggered by the control point. For the control point, the feedback of P30 is to monitor whether its described event is changed or not. Notice that all transitions are timed transitions excepting T17, T21, and T22 which are immediate transitions.

3.2 Reachability Decomposition

After using the Petri net to model UPnP protocol, Petri net state space analysis should be performed to confirm whether the system is bounded, safe, and no deadlock. Such analysis is a tedious and difficult task. Fortunately, the PIPE2 tool provides a function for state space analysis. When these results are correct, a reachability graph is generated by PIPE2.

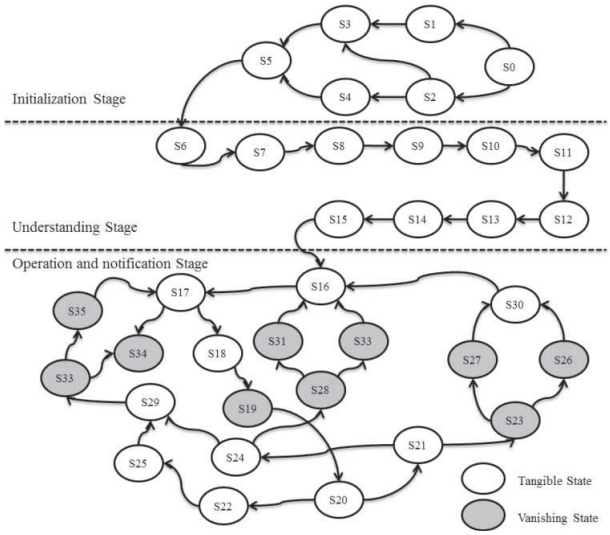

Fig. 3 Reachability graph for modeling UPnP protocol.

Figure 3 demonstrates the reachability graph for modeling the UPnP protocol shown in Fig. 2. It consists of 36 marking states. According to the UPnP behavior, we can divide the reachability graph into the following three stages.

1. Initialization: The states, S0 to S5, indicate doing initialization tasks for the control point and device. These tasks are performed as having a parallel mechanism.
2. Understanding: The states, S6 to S15, represent a sequential interaction for establishing a connection between a control point and a device. Obviously these states work in a sequence mechanism.
3. Operation and Notification: The states, S16 to S35, are running states and work in an interaction mechanism. These states reveal that a user operates a control point to send actions to a device for changing the devices state and the device then informs all control points having subscribed the state changed event.

As discussed above, the two stages of Initialization and Understanding are transient. As viewed from Fig. 3, each state in these two stages is executed just once. Hence, we can ignore them according to Markov chains. Next, we observe the operations at Operation and Notification stage, there are 20 states but ten states are vanishing states. These vanishing states will occur because immediate transitions are fired. Hence, we use a technology called the elimination of vanishing markings [10] to merge 20 states into 10 states. Table 1 illustrates how to eliminate 10 vanishing states forming 10 tangible marking based on the reachability graph as shown in Fig. 4. We called this technology reachability decomposition. For instance, S28, S31, and S32 are vanishing states and their destinations all are S16. Hence, we can eliminate the three states and use M16 state to represent their states.

Table 1 Elimination of vanishing states into tangible markings for modeling UPnP, the bold words indicates tangible states.

tangible/vanishing states	tangible markings
S16, S28, S31, S32	M16
S17, S33, S34, S35	M17
S18	M18
S19, **S20**	M19
S21	M20
S22	M21
S23, S26, S27, **S30**	M22
S24	M23
S25	M24
S29	M25

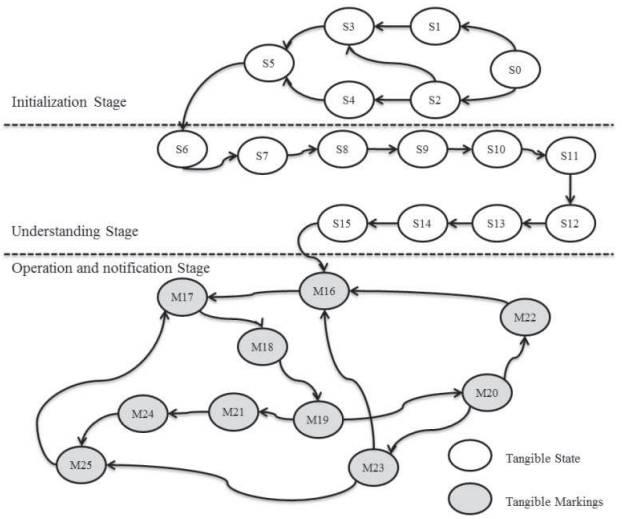

Fig. 4 The UPnP reachability graph after elimination of vanishing states into tangible markings.

3.3 Semantic Analysis

Through the discussion in the previous subsection, the system states can be reduced to 26 states from 36 for modeling UPnP. Although many states exist in the initialization and understanding stage, the probability for steady-state in both stages can be viewed as zero because these states are transient states in GSPN analysis. In the other words, the probability of these states will be zero when time becomes infinite.

Moreover, 10 states of the 20 states in the operation and notification stage are the vanishing states, hence we only focus on the other 10 tangible states. We categorize the timed transitions into three types: user-related, transmission-related and computing-related. T18 is the only one user-related transition. T2, T4, T6, T8, T10, T12, T14, T16, and T19 are transmission-related. The other timed transitions are computing-related. Table 2 summarizes the steady state distribution of tangible states. These states are divided into two groups: zero and nonzero. Zero represents the state to be temporary; otherwise steady-state. The marking M0 to M15 in Table 2 exists in states S0 to S15 in Fig. 3. Table 3 presents the marking distribution about P20 to P31

Table 2 Steady state distribution of tangible states for modeling UPnP.

Marking	M0-M15	M16-M25
Value	0	> 0

Table 3 Subset of tangible states for modeling UPnP.

	P20	P21	P22	P23	P24	P25	P26	P27	P28	P29	P30	P31
M16	1	0	0	0	0	1	1	0	0	0	1	0
M17	0	1	0	0	0	1	1	0	0	0	1	0
M18	0	0	1	0	0	1	0	1	0	0	1	0
M19	0	0	0	0	1	0	0	1	0	0	1	0
M20	0	0	0	0	1	0	0	0	1	0	1	1
M21	1	0	0	0	0	1	0	1	0	0	1	0
M22	0	0	0	0	1	0	1	0	0	0	1	0
M23	1	0	0	0	0	1	0	0	1	0	1	1
M24	0	1	0	0	0	1	0	1	0	0	1	0
M25	0	1	0	0	0	1	0	0	1	0	1	1

in ten steady states. The number 1 in Table 3 indicates that a token is existing in a place, and the number 0 represents no token in the place. For instance, the value of P20, P25, P26 and P30 is 1 in marking M16. In other words, four tokens will be put into places P20, P25, P26, and P30, respectively. As viewed in Fig. 2, we know that the control point will send an action message to the device. The other states can be inferred and so on. We call it semantic analysis that the meaning is specified in the state.

Using GSPN analysis, the ten tangible states impacting the system performance are interpreted as follows.

1. Sending Action State (M16): The control point prepares to send an action to a device.
2. Processing Action State (M17): The device has received an action from the control point.
3. Sending Response State (M18): The device has processed the action and is sending a response to the control point.
4. Waiting Action and Sending Eventing State (M19): The control point waits for a user to trigger a new action and the device prepares to send an eventing because of the previous action.
5. Waiting Action and Processing Eventing State (M20): The control point is waiting for a user to trigger a new action and is processing an eventing.
6. Sending Action and Sending Eventing State (M21): The control point prepares to send an action to a device and the device prepares to send an eventing because of the previous action.
7. Waiting Action State (M22): The control point is waiting for a user to trigger a new action.
8. Sending Action and Processing Eventing State (M23): The control point is sending an action and is processing the eventing changed by the previous action.
9. Sending Eventing State (M24): The device is sending the state changed eventing.
10. Processing Eventing State (M25): The control point is processing the eventing changed by the previous action.

3.4 Reward Assignment

Ten critical steady states for UPnP operations were introduced in the previous section. In this section, we show these states working on different devices. In other words, individual behavior in the control point and the device is useful for analyzing single device performance, such as power consumption, system utilization, and network throughput. We can use Eq. (3) to calculate the system performance if an appropriate reward rate is given to the equation. For examples, a real reward is used to obtain power consumption; a binary reward is assigned to evaluate system availability, utilization, and reliability; a count reward is utilized to calculate the system throughput. For power consumption analysis, we should consider the state probability. A state probability with index i is denoted by π_i. The probability of control points transmission state denoted $p_{c,t}$ can be calculated using

$$p_{c,t} = \pi_{16} + \pi_{21} + \pi_{23}. \tag{4}$$

The probability of control points computing state denoted $p_{c,c}$ can be calculated using

$$p_{c,c} = \pi_{20} + \pi_{23} + \pi_{25}. \tag{5}$$

We know that the sum of the probability of three states is equal to one; hence, The probability of control points idle state denoted $p_{c,i}$ can be given by

$$p_{c,i} = 1 - p_{c,t} - p_{c,c}. \tag{6}$$

As discussed above, the total power consumption is given by

$$P_c = p_{c,t} \times P_{c,t} + p_{c,c} \times P_{c,c} + p_{c,i} \times P_{c,i}, \tag{7}$$

where $P_{c,t}$, $P_{c,c}$, and $P_{c,i}$ are the power consumption of a control point working in transmission, computing, and idle states, respectively. In the same way, the power consumption of a device called P_d can be inferred by

$$p_{d,t} = \pi_{18} + \pi_{19} + \pi_{21} + \pi_{24}. \tag{8}$$

$$p_{d,c} = \pi_{17}. \tag{9}$$

$$p_{d,i} = 1 - p_{d,t} - p_{d,c}. \tag{10}$$

$$P_d = p_{d,t} \times P_{d,t} + p_{d,c} \times P_{d,c} + p_{d,i} \times P_{d,i}. \tag{11}$$

For system utilization, it can be easily calculated from busy probability. We know that the busy probability is $1 - p_{x,i}$, where x is c for a control point and d for a device. Hence, the system utilization for a control point and a device is

$$U_c = 1 - p_{c,i}, \tag{12}$$

$$U_d = 1 - p_{d,i}, \tag{13}$$

where U_c denotes control point utilization and U_d denotes device utilization. For network throughput in steady states, three arrival rates should be considered, including sending action rate, sending response rate, and sending eventing rate. These three rates are originally sending action command. Hence, we assume that the three rates are equal and are all denoted λ_a. The network throughput called η can be calculated by

$$\eta = (p_{c,t} + p_{d,t}) \times \lambda_a. \tag{14}$$

4. Experimental Results

In this section, we present the results of power consumption, system utilization and network throughput which are based on Eqs. (4) to (14) for analyze the performance of the UPnP protocol using the GSPN analysis results provided by the PIPE2 tool. Using GSPN analysis to evaluate system performance, specifying the rates of transitions do not come easy. According to the modeling process represented in [10], we should refer to a real-world system before using stochastic process or queuing network to evaluate system performance. In this paper, two methodologies are used to specify the rates of transitions or rewards. One is based on the nature of the programs, such as computing-based or communication-based programs, and the other is in accordance with the specifications of components, such as controller or network chip. We assume that the rate of computing-related and transmission-related transitions is set to 0.01 and 0.001, respectively, while the rate of user-related transitions is set to from 0.0001 to 0.01 for evaluating power consumption and system utilization, and 0.005 for evaluating throughput. According to S3C2440 [11] and DM9000E [12] specification, we calculated the power consumption under the conditions that the rate of a control point or a device working in transmission, computing, and idle states is 0.59, 0.31, and 0.213 W, respectively. Figure 5 illustrates that the control point is better than the device in power consumption but the device outperforms the control point in evaluating utilization as shown in Fig. 6. Throughput goes up considerably when the arrival rate is over 0.01 as shown in Fig. 7.

According to above discussion, the framework mentioned in this paper can be easily used to evaluate the UPnP system in terms of power consumption, system utilization and network throughput. Table 4 shows a comparison between our framework and the presented approaches. Jong et

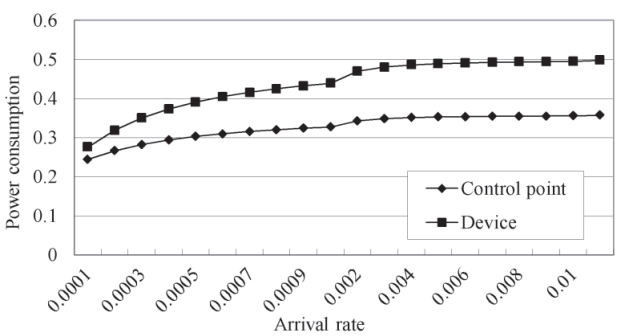

Fig. 5 Power consumption for UPnP control point and device.

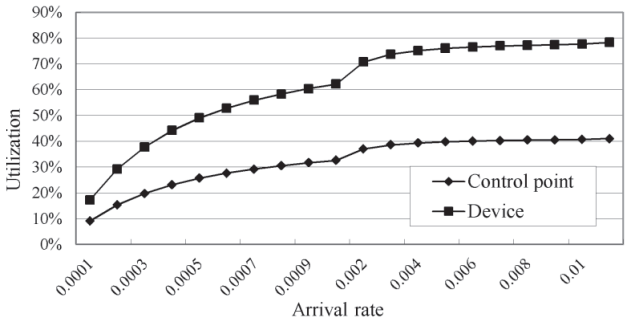

Fig. 6 System utilization for UPnP control point and device.

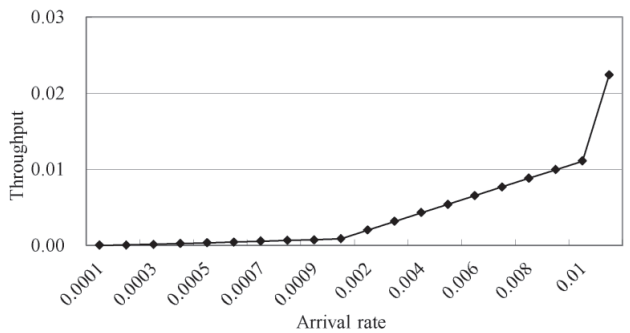

Fig. 7 Throughput for the UPnP network.

Table 4 The framework is compared with presented approaches.

	Power issue	Utilization	Throughput	Failure rate	authentication
[6] in 2008				◎	
[7] in 2009					○
Our scheme	◎	◎	◎		
○ mechanism is proposed. ◎ results are evaluated.					

al. proposed an efficient autonomous failure recovery mechanism to reduce the number of failures in 2008. Roached et al. in 2009 used coloured Petri net to model authentication.

5. Conclusions

This paper proposed a framework to analyze the run behavior of a UPnP system. We first used GSPN model to generate Markov chains steady-states for the interactions between the device and control point. A model is proposed and a reachability graph is drawn by the PIPE2. The interaction between control points and devices in UPnP system can be divided into three stages, including Initialization, Understanding and Operation & Notification. We found the probabilities in steady states are zero when the system is running in Initialization and Understanding stages. In Operation & Notification stage, all states in this stage can be divided into two states: tangible states and vanishing states. We use an elimination technology to delete these vanishing states caused by immediate transitions because that vanishing state is temporary. Next, we inferred the power consumption, system utilization and network throughput using the CTMC model and reward assignment skill. Through quantitative analysis, the steady states in the operation and notification stage dominate the system performance. However, the proposed framework is useful to evaluate the performance between control points and devices for UPnP system.

Our future work includes the following issues.

1. Continuously developing efficient software for embedded systems.
2. Designing proposed performance analysis models.
3. Modeling tools implementation.
4. Designing algorithms to improve performance for embedded systems.

Acknowledgements

The authors would like to thank the National Science Council of the Republic of China, Taiwan for financially supporting this research under Contract No. NSC 98-2221-E-252 -016. The excellent comments of the anonymous reviewers are greatly acknowledged as having helped a great deal in improving the quality and readability of this paper.

References

[1] http://www.upnp.org
[2] G. Hobling, T. Rabl, D. Coquil, and H. Kosch, "Interactive TV services on mobile devices," IEEE Multimedia, vol.15, no.2, pp.72–76, 2008.
[3] T. Hwang, H. Park, and J. Chung, "Personal mobile A/V control point for home-to-home media streaming," IEEE Trans. Consum. Electron., vol.54, no.1, pp.87–92, 2008.
[4] B. Lim, K. Park, and J. Kim, "UPnP A/V transport service using the add-in system," Proc. Third International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST '09), pp.52–57, 2009.
[5] F.M. Matsubara, T. Hanada, S. Imai, and S. Miura, "Managing a media server content directory in absence of reliable metadata," IEEE Trans. Consum. Electron., vol.55, no.2, pp.873–877, May 2009.
[6] Y.W. Jong, C.F. Liao, and L.C. Fu, "An efficient autonomous failure recovery mechanism for UPnP-based message-oriented pervasive services," IEEE International Conference on Systems, Man and Cybernetics, pp.1960–1965, 2008.
[7] T. Roached, K. Gorgonio, A. Perkusich, and H. Almedia, "Modeling the UPnP-UP protocol using coloured Petri nets," Proc. 17th International Conference on Software, Telecommunications & Computer Networks, pp.307–311, Sept. 2009.
[8] P. Bonet, C.M. Llado, R. Puijaner, and W.J. Knottenbelt, "PIPE v2.5: A Petri net tool for performance modeling," Proc. 23rd Latin American Conference on Informatics (CLEI 2007), San Jose, Costa Rica, Oct. 2007.
[9] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation, vol.4486 of Lect. Notes Comput. Sci. (Tutorial Volume), pp.220–270, 2007.
[10] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi, Queueing Networks and Markov Chains, pp.9–122, Wiley Interscience, 2006.
[11] http://html.alldatasheet.com/html-pdf/83787/SAMSUNG/S3C2440/246/1/S3C2440.html
[12] http://www.alldatasheet.com/datasheet-pdf/pdf/87572/ETC/DM9000E.html

Cheng-Min Lin was born in 1964. He received the B.S. and M.S. degrees in electronic engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, in 1989 and 1991, respectively, and the Ph.D. degree in Department of Information Engineering and Computer Science, Feng-Chia University, Taichung, Taiwan. Currently, he is a Professor in the Department of Computer and Communication Engineering, Graduate Institute of Electrical Engineering and Computer Science, Nan Kai University of Technology. His research interests include embedded systems, mobile computing, distributed systems, and telematics. Dr. Lin is a member of the IEEE Computer Society.

Shyi-Shiou Wu received his Ph.D. degree in Electronic Engineering from National Taiwan University of Science and Technology in 2005. He is an Associate Professor of Dept. of Electronic Engineering at Nan Kai University of Technology (NKUT), and he is also the dean of the Library & Information Service at NKUT. His research interests include wireless sensor networks, game science, and service-oriented architecture. Dr. Wu is a member of the IEEE Computer Society & Education Society.

Tse-Yi Chen received his B.A. degree in Electronic Engineering from Southern Taiwan University in 2008. He received his M.S. degree in Graduate Institute of Electrical Engineering and Computer Science at Nan Kai University of Technology (NKUT), His research currently interests include embedded systems, wireless sensor networks and telematics.