Performance Modeling and Analysis of ATM-based Network System Using DEVS Methodology

Kyon Ho Lee*, Tag Gon Kim**, Joon Won Lee*** *Regular Members*

ABSTRACT

DEVSim++ is a C++ based, object-oriented modeling/simulation environment which realizes the hierarchical, modular DEVS formalism for discrete event systems specification. This paper describes a methodology for performance modeling and analysis of an ATM-based network system within the DEVSim++ environment. The methodology develops performance models for the system using the DEVS framework and implement the models in C++. Performance indices measured are the length of queues located at connection points of the system and cell waiting times with respect to QoS grades for a network bandwidth of 155 Mbps.

I. Introduction

Performance modeling and simulation analysis are essential to optimizing system parameters for new design as well as existing ones. Especially, as complexity of systems is increased, simulation modeling may be the only means to evaluate performance of such systems. ATM networks are an example of such complex systems[1][2][3].

A number of network systems have been developing for ATM technology based B-ISDN as a next generation high speed communication. The technology can provide end users with a variety of public services which satisfy different service requirements, traffic characteristics, and geographical coverage[4]. Owing to requirements in such satisfaction, an interface technique between end users and ATM local exchanges is one of major issues for the ATM network. The reference model defined by ITU-T SG13 consists of three area networks of B-ISDN UNI, namely, Customer Premises Network, Access Network, and Transport Network[5].

ATM-based network is mostly constructed in a star topology. However, a star topology can not be the best in all cases of networking domain. Especially, a subscriber access domain interfacing customer premises network to ATM transport network requires more deliberated considerations[6~10]. Sohn proposed a hybrid structure of subscriber access network for the introductory phase broadband communication network[11]. Lee discussed an optimized configuration of different ATM access network systems which were developed to cover various subscriber access domains depending on service requirements, traffic characteristics, and geographical coverage[12].

This paper describes performance modeling and simulation analysis for ATM-based network systems. We consider a star-ring-star topology for establishing an ATM subscriber access network. This paper also provides a modeling and simulation analysis methodology for ATM-based network systems. The methodology is based on Zeigler's DEVS (Discrete Event System Specification) paradigm to exploit compatibility between the hierarchical, modular model specification and the hierarchical distributed access network system architecture. The DEVS formalism and DEVSim++ environment are basis of modeling and simulation.

The DEVS formalism developed by Zeigler

* ETRI 교환.전송기술연구소 라우터기술연구부(kyou@etri.re.kr)
** 한국과학기술원 전기및전자공학과(tkim@ee.kaist.ac.kr)
*** 안동대학교 전자정보산업학부(leejw@anu.andong.ac.kr)
논문번호 : 99122-0328, 접수일자 : 1999년 3월 28일

supports specification of discrete event systems in hierarchical, modular manner[13][14]. Discrete event simulation has been widely used as a performance evaluation means in many areas of system design including communication networks[15]. In such performance study, simulation models are much more reliable and accurate than analytical ones, which may omit some aspects of the behavior of systems under design. In particular, when temporal issues of systems are significant, discrete event modeling and simulation can be considered the best solution. DEVSim++ is a realization of the DEVS formalism in C++, which provides modelers with facilities for modeling systems within DEVS semantics and simulating DEVS models in hierarchical fashion[16][17].

We organize this paper as follows. Section 2 presents a brief review of the DEVS formalism and DEVSim++ modeling and simulation environment. Section 3 describes characteristics of the ATM-based network system architecture under consideration. Development of a simulation model for the system is given in Section 4 and simulation results in Section 5. We conclude this paper in Section 6.

Ⅱ. DEVS Formalism: A brief review

A set-theoretic formalism, the DEVS formalism, specifies discrete event models in a hierarchical, modular form. Within the formalism, one must specify 1) the basic models from which larger ones are built, and 2) how these models are connected together in hierarchical fashion. A basic model, called an atomic model (or atomic DEVS), has specification for dynamics of the model. An atomic model AM is specified by a 7-tuple:

$$AM = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a >$$

X : input events set;
S : sequential states set;
Y : output events set;
δ_{int} : S→S : internal transition function;
δ_{ext} : Q×X→S : external transition function;
λ : S→Y : output function;
t_a : S→Real : time advance function,

where $Q=\{(s,e) \mid s \in S, 0 \le e \le ta(s)\}$: total state of M.

We call the four functions in the tuple, namely, internal transition function, external transition function, time advance function and output function, the DEVS characteristic functions. Note that the formalism specifies a discrete event system in the system-theoretic viewpoint. That is, the first three elements in the 7-tuple are system's input, system's state set, and system's output, respectively, and the next four elements give constraints among the three. Thus, when developing an atomic DEVS model, modelers can specify the characteristic functions independently. Such separated specification exploits reusability of the characteristic functions through inheritance in our object-oriented simulation environment of DEVSim++.

The second form of the model, called a coupled model (or coupled DEVS), tells how to couple (connect) several component models together to form a new model. This latter model can itself be employed as a component in a larger coupled model thus giving rise to construction of complex models in hierarchical fashion. A coupled model CM is defined as:

CM = <X, Y, M, EIC, EOC, IC, SELECT>

X : input events set;
Y : output events set;
M : DEVS components set;
EIC⊆CM.IN×M.IN : external input coupling relation;
EOC⊆M.OUT×CM.OUT : external output coupling relation;
IC⊆M.OUT×M.IN : internal coupling relation;
SELECT : 2M-ϕ →M : tie-breaking selector,

where the extensions .IN and .OUT represent the input ports set and output ports set of

respective DEVS models.

The three coupling relations, namely, EIC, EOC, and IC, specifies input/output connections between component models and the coupled model. The coupling scheme (CS) of a coupled model is defined as a collection of the three relations, (EIC, EOC, IC). The coupling scheme plays the main role in hierarchical construction of DEVS models.

DEVSim++ is a realization of the DEVS formalism in C++. The DEVSim++ environment supports modelers to develop discrete event models using the hierarchical composition methodology in an object-oriented framework. The environment is a result of the combination of two powerful frameworks for systems development: the DEVS formalism and the object-oriented paradigm.

Ⅲ. System Characteristics

A system under consideration is an interface system between the local exchange and subscribers, which constitutes subscriber access network in a star-ring-star topology. The system consists of a switch node, a collection of rings, each consisting of a collection of edge nodes. Each edge node is connected to a number of subscribers.

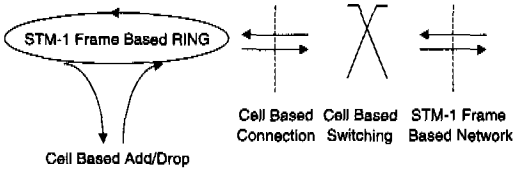


Fig. 1. Transmission Mechanism

The switch node mainly performs a traffic switching among edge nodes and local exchanges. Each link of the switch node is based on an STM-1 frame with 155 Mbps bandwidth. The edge node mainly concentrates and distributes traffic between subscribers and the switch node through the ring. The traffic from subscribers is based on ATM cells with the speed of DS-1, DS-3 or STM-1 depending on applications. The transmission behavior in the system is shown in Figure 1.

A simple cell-based Add/Drop or Switching behavior is used for adding from and dropping into subscribers in the edge node or switching in the switch node.

Ⅳ. Development of Simulation Models

This section describes modelling system architecture and shows development of a distributed access network system simulation model in DEVSim++.

1. Modeling Overview

The overall system architecture under consideration is shown in Figure 2. At the top level, the system consists of two subsystems, a SWITCH and a number of RINGs. Having the (n × n) switching function for traffic, the SWITCH can connect n RING's and communicate with n Local Exchange sites. Each RING comprise a set of identical Edge Nodes (EN's), each of which has 4 inputs and 4 outputs for communicating with subscribers.

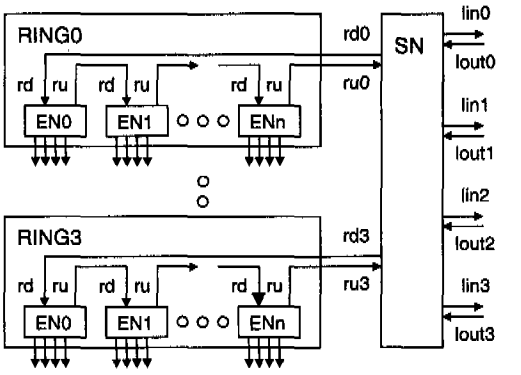


Fig. 2. System Architecture

An EN, as shown in Figure 3, consists of a ring access (RA) which accesses the ring to add or drop cells, UPWARD for concentrating cells sent from 4 subscribers into RA and DOWN-

WARD for distributing cells dropped from RA into 4 subscribers.

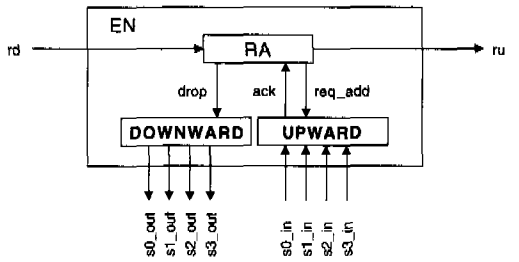


Fig. 3. Coupled Model of EN

RA consists of two atomic models, D_1-to-2 and M_2-to-1, as shown in Figure 4. D_1-to-2 forwards traffic to the ring if there is no cell dropped into local subscribers. Otherwise, D_1-to-2 drops the cell to local subscribers. Likewise, M_2-to-1 forwards traffic, arrived from D_1-to-2, into the ring if there is no cell to add on. Being ready to add on the ring, M_2-to-1 inserts a cell being ready into the empty slot on the frame. If there is no empty slot on the frame, M_2-to-1 forwards with no adding.

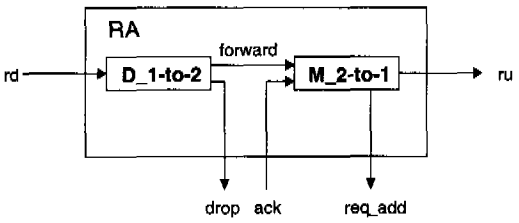


Fig. 4. Coupled Model of RA

SWITCH also includes RA which accesses the ring RA in SWITCH has input and output channels to receive and transmit the cell stream from and to SWITCH. On the other hand, RA in EN has I/O channels to add and drop a cell from and to subscriber.

2. Modeling Considerations

1) Model Reusability:

From modeling, the following observation can be made. First, an identical model is reused as a component of a number of different locations. This is because the system should provide a number of subscribers and/or equipment with reasonable connections. For example, assume that one system has four RING's, each of which has four RN's. Then, totally 20(=4×4+4) M_2-to-1 atomic models are used for one distributed access network system. Second, there are many models which perform similar functions.

Being implemented in an object-oriented environment of C++ the DEVSim++ is efficient to develop high reusable models. It makes possible to extend, to add, or to change models so easily. Therefore, it requires less efforts to simulate and analyze a system under consideration.

2) Transformation of Time into Number of Cells:

The transmission both on the ring and on network(from Local Exchange) is based on an STM-1 frame with 155Mbps bandwidth. As shown in Figure 1, adding from or dropping into subscribers in EN are based on the cell-based transmission behavior. The cell-based transmission behavior is also employed to transmit or receive into/from SWITCH, and to switch cells inside of SWITCH.

Concerning transmission speed in the development of a system behavioral model, bandwidth can be considered as a number of cells. The STM-1 frame is recommended to have 260×9 Bytes without overheads. One ATM cell has 53 Bytes. Therefore 44 ATM cells are included in an STM-1 frame. By transformation of transmission time into numbers of cells, we easily model the transmission operation.

3) Separation of Models:

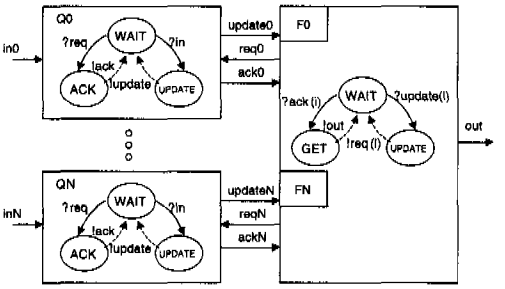


Fig. 5. Model of an N-input Multiplexer

We use "Flag" to separate a model with others.

Figure 5 shows an example model of a N-input multiplexer used for switching. If an input switching to other channels is enqueued into Qi, Qi sets the corroresponding "flag" located in PROC by means of the "update" assertion, and then wait. PROC asserts "req" to Qi which has set the "flag" and wait for "ack" from Qi. When Qi receives "req", it gives "ack" to PROC.

3. Models Development in DEVSim++

Regarding models development in DEVSim++, we describe development of an atomic model, M_2-to-1, and a coupled model, RA, in DEVSim++.

3.1 Atomic Models

The atomic model M_2-to-1 can be represented in DEVS semantics as follows:

X = {?forward, ?ack}
Y = {!ru, !req_add}
S = {phase | phase ∈ {WAIT, ACTIVE, SEND, ADD}}

δ_{ext}: δ_{ext} ((WAIT), ?forward) = ACTIVE
δ_{ext}(ADD, ?ack) = SEND

δ_{int} : δ_{int} (SEND) = WAIT
δ_{int} (ACTIVE) = ADD

t_a : t_a (ADD) = infinity
t_a (WAIT) = infinity
t_a (ACTIVE) = active_time
t_a (SEND) = sending_time

O : O(ACTIVE) = !req_add
O(SEND) = !ru

Figure 6 shows the state transition diagram of M_2-to-1. M_2-to-1 has two inputs, forward and ack, and two outputs, ru and req_add. When an input arrives at the port "?forward", M_2-to-1 transits into the phase ACTIVE and stays there for active_time units. Then it outputs on the port "!req_add" and then transits to the phase ADD. At the phase ADD, it waits for an input "?ack" to be arrived. On receiving the input "?ack", M_2-to-1 transits to the phase SEND and stays there for sending_time units. After then it returns to the phase WAIT after generating an output on the port "!ru".

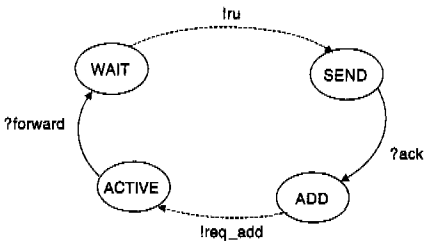


Fig. 6. State Diagram for M_2-to-1

The followings are codes for implementation of M_2-to-1 within DEVSim++.

```
const int   M21_ATV_TIME = 0;
const int   M21_SND_TIME = 0;
enum {WAIT, ACTIVE, ADD, SEND};

// external transition function
void m21_ext_transfn(State_vars& s,
        const timeType&, const Messages& x)
{
 if (*x.get_port() == "forward") {
    if (s.get_value("phase") == WAIT) {
       s.set_value("phase", ACTIVE);
       s.set_value("size", x.get_value());
            } else
              exit(1);
    } else if (*x.get_port() == "ack") {
      if (s.get_value("phase") == ADD) {
                    int global, local;
         global = s.get_value("size");
         local = x.get_value();
         s.set_value("phase", SEND);
         s.set_value("size", global + local);
    } else
       exit(1);
    } else
       exit(1);
}
```

```
// internal transition function
void m21_int_transfn(State_vars& s)
{
   if (s.get_value("phase"") == ACTIVE)
      s.set_value("phase", ADD);
      else if (s.get_value("phase") == SEND)
      s.set_value("phase", WAIT);
      else
         exit(4);
}

// output function
void m21_outputfn(const State_vars& s,
         Messages& message)
{
    int total;
    if (s.get_value("phase") == ACTIVE) {
       total = s.get_value("size");
       message.set("req_add",
                     MAXCELLS - total);
    } else if (s.get_value("phase") == SEND) {
            total = s.get_value("size");
            message.set("rout", total);
    }
}

// time advance function
timeType  m21_time_advancefn(const  State_vars&
s)
{
    if(s.get_value("phase") == ACTIVE)
      return M21_ATV_TIME;
    if(s.get_value("phase") == SEND)
      return M21_SND_TIME;
    else
      return infinity;
}

// routine for creating the model
void create_m21(Atomic_models& m21)
{
    String*          name = m21.get_name();

    m21.set_sigma(infinity);

m21.set_state_var(3,"phase", "name", "size");
    m21.set_state_value("phase", WAIT);
    m21.set_state_value("name", name);
    m21.set_state_value("size", 0);

    m21.set_ext_transfn(m21_ext_transfn);
    m21.set_int_transfn(m21_int_transfn);
    m21.set_outputfn(m21_outputfn);

m21.set_time_advancefn(m21_time_advancefn);
}
```

3.2. Coupled Models

The coupled model RA, shown in Figure 4, consists of three atomic models. The coupled model RA can be represented in DEVS semantics as follows:

```
DN = < X, Y, M, EIC, EOC, IC, SELECT >
X = {?nq, ?rd}
Y = {!ru, !drop}
M = {D_1-to-2, M_2-to-1}
EIC = {(RA.rd, D_1-to-2.rd), (RA.ack,
M_2-to-1.ack)}
EOC = {(D_1-to-2.drop, RA.drop),
(M_2-to-1.req_add, RA.req_add), (M_2-to-1.ru,
RA.ru)}
IC = {(D_1-to-2.forward, M_2-to-1.forward)}
```

The following codes show DEVSim++ implementations for the coupled model RA.

```
void create_D12(Atomic_models& D12);
void create_M21(Atomic_models& M21);
void create_GEN(Atomic_models& GEN);

void make_RA(Coupled_models& ra)
{
    Atomic_models&    d12    =    *(new
Atomic_models("D12"));
    Atomic_models&    m21    =    *(new
Atomic_models("M21"));
    create_D12(d12);
    create_M21(m21);

    ra.add_inports(2, "rd", "nq");
    ra.add_outports(2, "ru", "drop");
```

```
    ra.add_children(2, &d12, &m21);
    ra.add_coupling(&ra, "rd", &d12, "rd");
    ra.add_coupling(&ra, "ack", &m21, "ack");
    ra.add_coupling(&m21, "ru", &ra, "ru");
    ra.add_coupling(&d12, "forward", &m21,
"forward");
    ra.add_coupling(&d12, "drop", &ra, "drop");
    ra.add_coupling(&m21, "req_add", & ra,
"req_add");
}
```

V. Simulation Experiments and Results

1. Simulation Experiments

Two goals for simulation experiments are as follows:

- To foresee the maximum lengths of queues at: INBUF, CELLPOOL, RA and SWITCH. These give us important data for cell waiting status during transmission.
- To estimate average waiting times of cells with respect to QoS grade levels, which are waiting in CELLPOOL.

For the experimentations, several cases of subscribers having different average bandwidths are applied. Since the transmission speed through a RING or a SWITCH is upto 155Mbps, if 4 ENs are connected to one RING and 4 subscribers are included in a EN, about 10 Mbps in average can be given to one subscriber. Maximum queue lengths and average waiting time are measured for various subscribers' bandwidths ranging from 5Mbps to 100Mbps.

A summary of assumptions for simulation modeling is as follows:

- 90% of the traffic from a RING is routing to the network through the SWITCH. And the rest(10%) is forwarding back into the same RING, which is destined to the subscribers connecting to the same RING.
- The traffic given at any port of the SWITCH are divided and routed to the rest ports of the SWITCH with equal probability.
- Any EN has statistically the same portion of traffic sent from or added into a RING. If a RING includes 4 ENs, 25% of the traffic sent from a RING are dropped to be routed into destined subscriber. The rest are forwarded into the next stage of an EN. During forwarding, a new traffic from subscribers is added on, which has the same probability as dropping.

For simulating cell loss rate of 10-12, more than 1012 cells should be generated. This is a typical method for simulation analysis of ATM-based network. Such two techniques as importance sampling[18] and the generalized extreme value theory[19] have been proposed to deal with such a problem.

We can approach differently to concern system behavior at the discrete-event level. One way is that the value for numbers of cells, instead of cell by cell, are generated and distributed with given probability density functions. It is an easier way to handle event messages as well as to implement a simulator. Instead of counting how many events("cells") waiting in queues, we just consider the integer value calculated in queues.

We employed a token passing based simulation scheme. In the scheme, only one token traverses each RING. Each token consists of a number of slots. Indeed, a slot means a message. When a model receives a token, it can remove/insert messages from/into the token. But, the total number of slots in a token cannot exceed a bound. We have already known that 44 slots exist in a frame (125μ s) of an 155Mbps RING. It is natural that a token is responded by n*44 slots. For simplicity, we set n to 7. Consequently, a token is composed of 44*7 slots.

The relationship between physical and virtual times can be acquired easily. Let the RING turnaround time in virtual time be T_r. Thus, one unit in virtual time corresponds to $125*7/T_r$ μ s. Now, we should discuss about how we can design subscriber models with given average and maximum bandwidths. Consider that in a RING only one subscriber is active and others are inactive. Since 155Mbps corresponds to 44*7 cells

during T_r, bandwidth of corresponds to 44*7/155* ω cells. For reducing simulation time complexity, we assume that a subscriber generates cells in a burst manner. Therefore, if a subscriber generates cells at an instance, bandwidth of corresponds to 44*7/155*ω /α times of burst output generation frequency during T_r. Then, the intergeneration time t_a is defined as:

$$t_a = \frac{N}{\frac{7*44}{155} * \frac{w}{a}} = \frac{155}{7*44} * \frac{N*a}{w} \quad (1)$$

We set that T_r=1 and α =44*7. Consequently,

$$t_a = \frac{155}{w} \quad (2)$$

Assume that the request rate of a subscriber has a uniform distribution and the maximum and average bandwidths of the subscriber is max and avg, respectively. Then the subscriber can be modeled statistically as:

$$U[\frac{155}{w_{max}}, (2*\frac{155}{w_{avg}} - \frac{155}{w_{max}})] \quad (3)$$

where U[a,b] denotes a uniform random number generator in [a,b].

2. Simulation Results

Table 1 shows maximum queue lengths for the given subscriber's average bandwidths. Each number means how many cells are waiting in the queue. In other words, it gives the queue length which should be implemented to avoid cell loss.

Table 1. Maximum Queue Length

avg(Mbps)	inbuf	cellpool	switch	switch
5	308	655	30	280
10	308	3552	171	343
30	770	36334	282	347
50	7392	45584	263	345
100	27643	45815	319	340

max (Mbps) = 155.

The simulation results for average waiting times with respect to QoS grade levels are shown in Table 2. Note that the traffic with lower QoS grades can rarely be served.

Table 2. Average Waiting Time

avg(Mbps)	QoS0	QoS1	QoS2	QoS3
5	1.96	1.72	1.66	1.61
10	20.79	7.05	4.56	2.97
30			64.60	7.47
50				33.64
100				68.17

max (Mbps) = 155.

VI. Conclusion

Performance modeling and analysis for an ATM-based network system has been discussed. We consider a star-ring-star topology for establishing an ATM subscriber access network. The objectives of modeling are not only to analyze dynamic traffics in a transient state but also to make decisions of architectural parameters such as queue lengths. By consideration of the system architectural characteristics, we employ Zeigler's DEVS formalism and develop models within DEVSim++ environment. As results of simulation experiments in DEVSim++, we analyze the length of queues located in connection points. Also we analyze cell waiting times with respect to QoS grade levels, which are for the cells waiting for to be added on the network.

Such results help us to decide the maximum lengths of queues to avoid cell loss. We can observe that a queue in the SWITCH is rarely dependent of the subscriber's bandwidth. But queues at the other locations in the RING is much dependent of each subscriber's bandwidth. We also observe that the traffic with lower QoS grade can rarely be served if a subscriber's bandwidth is more than 30 Mbps.

A modeling and simulation analysis methodology for the ATM-based network system within Zeigler's DEVS paradigm has also been provided. The methodology can exploit compatibility between the hierarchical, modular model specification and the hierarchical distributed access

network system architecture.

For future work, we should collect more data for various situations. From this we can optimize the design parameters for the system under development.

References

[1] Chai and S. Ghosh, "Modeling and Distributed Simulation of a Broadband-ISDN Network," *IEEE Computer*, September 1993.

[2] G. Pujolle and D. Gaiti, "Performance Management Issues in ATM Networks," *Proceedings of Information Networks and Data Communications*, April 1994.

[3] V. S. Frost and B. Melamed, "Traffic Modeling For Telecommunications Networks," *IEEE Communications Magazine*, March 1994.

[4] W. Stalling, *ISDN and Broadband ISDN*, Macmillan Publishing Company, 1992.

[5] The ATM Forum, *ATM User-Network Interface (UNI) Specifications : Version 3.1.*, PRT Prentice Hall, Englewood Cliffs, NJ, 1995.

[6] Y. Cha, I. Kim, J. Jun, K. Lee, K. Han, Proposal and Implementation of Multiaccess ATM Network with Single Ring Topology, *Journal of Korea Information Science Society*, Vol. 23, No. 6, pp. 241-254, June 1998.

[7] DAVIC, *DAVIC 1.2 Specification, Part 04*, 1996

[8] The ATM Forum, *RBB(Residential Broadband) WG Baseline Text*, July 1997.

[9] Z. Budrikis, G. Mercankosk, M. Blasikiewwicz, Y. Yao, and P. Potter, A Generic Flow Control for B-ISDN, *Proceeding of IEEE INFOCOM92*, pp. 895-903, 1992.

[10] W. Denzel, A. Engbersen, A. Herkersdrof, and E. Port, Shared-Medium-Based Subscriber Ring Access to ATM Networks, *Proceeding of ISS95*, pp. 452-356, April 1995.

[11] E. Son, S. Hong, and K. Kim, "Network Architecture for the Introductory Phase Broadband Subscriber Access Network," *Proceedings of Information Networks and Data Communications*, April 1994.

[12] P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Academic Press, 1984.

[13] K. Lee, K. Ki, S. Ro, J. Choe, and J. Kim, Performance Analysis of the B-NT System Using Simulator, *Journal of Korea Institute of Communication Science*, Vol. 23, No. 6, pp. 1503-1514, June 1998.

[14] T. Kim, "The DEVS Formalism: Reusable Model Specification in an Object-oriented Framework," *Int. Journal in Computer Simulation*, vol 5, no. 4, pp. 397-415, 1995.

[15] J. Nam, C. Son and T. Kim, "Performance Evaluation of Congestion Control on Interworking Conventional LANs with B-ISDN," *Proceedings of SCSC'94*, pp. 175-180, July, 1994, San Diego, CA.

[16] T. Kim and S. Park, "The DEVS Formalism: Hierarchical Modular Systems Specification in C++," *Proceedings of the 1992 European Simulation Multiconference (ESM92)*, pp. 152-156, June, 1992.

[17] T. Kim, *DEVSim++ Users Manual, Ver 1.0*, CORE Lab., KAIST, Korea. 1994.

[18] Q. Wang and V. S. Frost, "Efficient Estimation of Cell Blocking Probability for ATM Systems," *IEEE Trans. on Networking*, April 1993.

[19] F. B. Bernabel, "ATM System Buffer Design Under Very Low Cell Loss Probability Constraints," *Proceedings of IEEE Conf. on Computer Communication, INFOCOM'91*, April 1991.

이 규 호(Kyou Ho Lee) 정회원

1980년 : 경북대학교 전자공학과 공학사

1982년 : 경북대학교 대학원 전자공학과 공학석사

1998년 : Univ. of Gent 컴퓨터 공학과 공학박사

1986년~1988년 : 미국 AIT Inc,

연구원

1983년~현재 : 한국전자통신연구원(ETRI) 책임연구원

<주관심 분야> ATM네트워크, 병렬처리를 이용한 고속프로토콜처리, 고속 컴퓨터 통신 시스템

김 탁 곤(Tag Gon Kim) 정회원

1975년 : 부산대학교 전자공학과 공학사

1980년 : 경북대학교 대학원 전자공학과 공학석사

1988년 : Univ of Arizona, 전기/컴퓨터공학과 공학박사

1980년~1983년 : 부산수산대학교(현: 부경대학교), 통신공학과, 전임강사

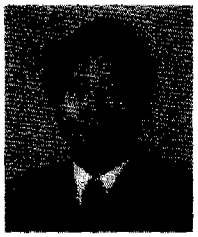
1987년~1989년 : Environmental Research Lab(미), 연구엔지니어

1989년~1991년 : Univ of Kansas, 전기/컴퓨터공학과, 조교수

1991년~1993년 : 한국과학기술원 전기/전자공학과 교수

<주관심 분야> Discrete Event Modeling/Simulation, Computer/Communication Systems Analysis

이 준 원(Joon Won lee) 정회원



1976년 : 서울대학교 전자공학과 공학사

1992년 : 충북대학교 전산학과 이학석사

1997년 : 충북대학교 전산학과 이학박사

1977년~1979년 : 삼성전기 근무

1980년~1998년 : 한국전자통신연구원(ETRI) 책임연구원

1998년~현재 : 안동대학교 전자정보산업학부

<주관심 분야> 초고속통신망, 정보통신표준화, 방재통신망, 통신프로토콜