Performance Modeling and Analysis of Distributed Access Network System Using DEVSim++ *

Kyou Ho Lee

Dept. of Broadband Comm. Network
ETRI
POB 106, Yusong-Gu
Taejon 305-600, Korea
kyou@winky.etri.re.kr

Tag Gon Kim

Dept. of Electrical Engineering
KAIST
373-1 Kusong-Dong, Yosong-Gu
Taejon 305-701, Korea
tkim@ee.kaist.ac.kr

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 a distributed access network system under development 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.

1 Introduction

ATM technology based B-ISDN has been expected as a next generation high speed communication. The technology will provide end users with a variety of public services which satisfy different service requirements, traffic characteristics, and geographical coverage. 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[2]. We have proposed a distributed access network architecture as an introductory phase of B-ISDN[5], which covers urban areas having various traffic characteristics and service requirements. The proposed system now is under development.

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][3].

Discrete event simulation has been widely used as a performance evaluation means in many areas of system design including communication networks. 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.

The DEVS formalism developed by Zeigler supports specification of discrete event systems in hierarchical, modular manner[6]. 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[7].

This paper describes performance modeling and simulation analysis for the distributed access network system under development. The modeling methodology is based on Zeigler's DEVS formalism to exploit compatibility between the hierarchical, modular model specification and the hierarchical distributed

*ISBN 0-8186-6440-1.

access network system architecture.

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 distributed access network system architecture. Development of a simulation model for the distributed access network system is given in Section 4 and simulation results in Section 5. We conclude this paper in Section 6.

2 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 [Zeg84]:

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

X : input events set;
S : sequential states set;
Y : output events set;
$\delta_{int} : S \rightarrow S$: internal transition function;
$\delta_{ext} : Q \times X \rightarrow S$: external transition function;
$\lambda : S \rightarrow Y$: output function;
$ta : S \rightarrow Real$: time advance function,

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

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 [Zeg84]:

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

X : input events set;
Y : output events set;
M : DEVS components set;
$EIC \subseteq CM.IN \times M.IN$:
external input coupling relation;
$EOC \subseteq M.OUT \times CM.OUT$:
external output coupling relation;
$IC \subseteq M.OUT \times M.IN$:
internal coupling relation;
$SELECT : 2^M - \emptyset \rightarrow M$: tie-breaking selector,

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

3 Characteristics of Distributed Access Network System

A distributed access network system is an interface system between the local exchange and subscribers. The system consists of a head node, a collection of rings, each consisting of a collection of ring nodes. Each ring node is connected to a number of subscribers.

The head node mainly performs a traffic switching among ring nodes and local exchanges. Each link of the head node is based on a STM-1 frame with 155 Mbps bandwidth. The ring node mainly functions multiplexing the traffic from subscribers to the head node as well as distributing the traffic from the head node to subscribers 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. For the transmission speed, we consider to transform the bandwidth into a number of cells. The STM-1 frame is recommended to have 260 x 9 Bytes without overheads. One ATM cell has 53 Bytes without overheads. Therefore 44 ATM cells are included in a STM-1 frame. The transmission mechanism in the distributed access network system is shown in Figure 1. A cell-by-cell mechanism is used for adding from and dropping into subscribers in the ring node and switching in the head node. But the transmission on the ring and the network is based on the STM-1 frame.

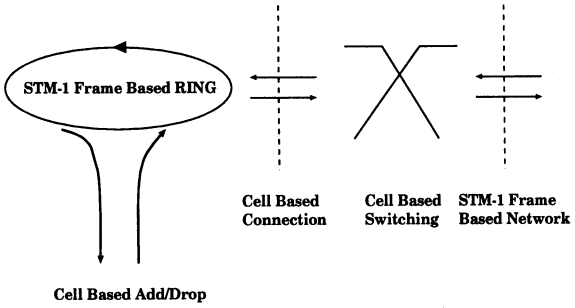

Figure 1: Transmission Mechanism

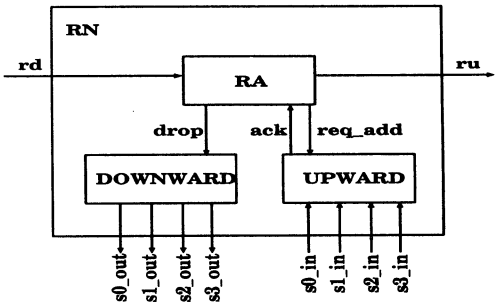

Figure 3: Coupled Model of RN

4 Models Development

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

4.1 Modeling Overview

The overall distributed access network system architecture is shown in Figure 2. At the top level, the distributed access network system consists of two subsystems, a HEAD and a RING. Having the (n x n) switching function for traffic, the HEAD can connect n RING's and communicate with n Local Exchange sites. Each RING comprise a set of identical Ring Nodes (RN's), each of which has 4 inputs and 4 outputs for communicating with subscribers.

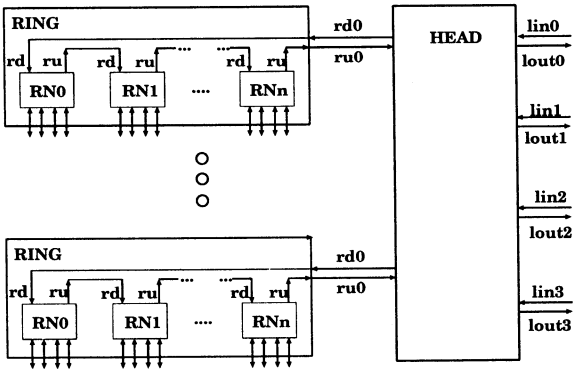

Figure 2: The System Architecture

A RN, 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 DOWNWARD for distributing cells dropped from RA into 4 subscribers.

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.

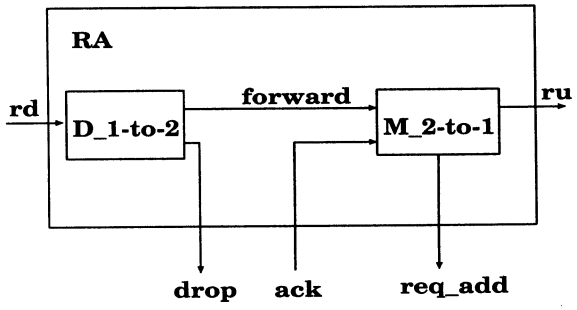

Figure 4: Coupled Model of RA

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

4.2 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++.

4.2.1 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 $\in$
{WAIT, ACTIVE, SEND, ADD}}

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

d_{int}: d_{int}(SEND) = WAIT
d_{int}(ACTIVE) = ADD
ta : ta(ADD) = infinity
ta(WAIT) = infinity
ta(ACTIVE) = active_time
ta(SEND) = sending_time

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

Figure 5 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".

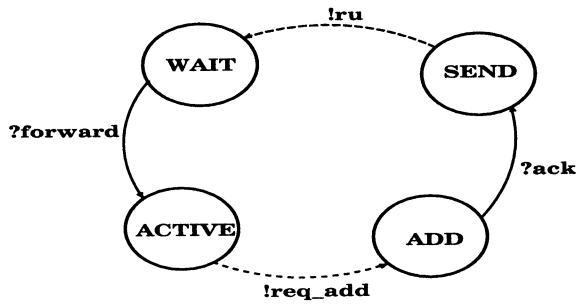

Figure 5: State Diagram for M_2-to-1

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);
}
```

4.2.2 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");
}
```

5 Simulation Experiments and Results

5.1 Simulation Experiments

Two goals for simulation experiments are as follows:

1. to foresee the maximum lengths of queues at: INBUF, CELLPOOL, RA and SWITCH. These give us important data for cell waiting status during transmission.

2. 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 RNs are connected to one RING and 4 subscribers are included in a RN, 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:

1. 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.

2. The traffic given at any port of the SWITCH are divided and routed to the rest ports of the SWITCH with equal probability.

3. Any RN has statistically the same portion of traffic sent from or added into a RING. If a RING includes 4 RNs, 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 a RN. 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 10^{12} cells should be generated. A couple of techniques, such as importance sampling[8] or the generalized extreme value theory[9], has been proposed to deal with such a problem.

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 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($125\mu s$) 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 $125*7/T_r$ μs. 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 ω corresponds to $44*7/155*\omega$ cells. For reducing simulation time complexity, we assume that a subscriber generates cells in a burst manner. Therefore, if a subscriber generates α cells at an instance, bandwidth of ω corresponds to $44*7/155*\omega/\alpha$ times of burst output generation frequency during T_r. Then, the intergeneration time ta is defined as:

$$ta = \frac{N}{\frac{7*44}{155} * \frac{\omega}{\alpha}} = \frac{155}{7*44} * \frac{N*\alpha}{\omega}.$$

We set that $T_r = 1$ and $\alpha = 44*7$. Consequently,

$$ta = \frac{155}{\omega}.$$

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

$$U[\frac{155}{\omega_{max}}, 2*\frac{155}{\omega_{avg}} - \frac{155}{\omega_{max}}]$$

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

5.2 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

ω_{avg}(Mbps)	inbuf	cellpool	head	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

ω_{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 for.

Table 2: Average Waiting Time

ω_{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

ω_{max}(Mbps) = 155.

6 Conclusion

Performance modeling and analysis for the distributed access network system under development has been discussed. 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 distributed access network system architectural characteristics, we employ Zeigler's DEVS formalism and develop model within DEVSim++ environment. As results of simulation experiments in DEVSim++, we analysis the length of queues located in connection points. Also we analysis cell waiting times with respect to QoS grade levels, which are for the cells waiting for to be added on a 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.

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

Acknowledgement

We give special thanks to Dr. Mun-Kee Choi and Mr. Tae-Soo Chung for their guidence. We are also very grateful of our colleagues for their helpful comments.

References

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

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

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

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

[5] 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.

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

[7] T. Kim and S. Park, "The DEVS Formalism: Hierarchical Modular Systems Specification in C++," *Proceedings of the 1992 European Simulation Multiconference*, June 1992.

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

[9] 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.