전연결 멀티미디어 서비스를 지원하는 위성통신 시스템의 성능분석

준회원 탱 유*, 종신회원 김 덕 년**

Performance Analysis of Satellite Communication System for Multimedia Services with Full Connectivity

Yue Teng* *Associate Members*, Doug Nyum Kim** *Lifelong Members*

요 약

본 논문은 On Board Processing(OBP) 위성통신 중계기시스템에 대한 채널할당방식과 그 성능에 대하여 분석하였으며 채널할당의 효율을 개선하기 위한 CRA와 DRA에 대한 개선방식을 제시하였다. 수학적 분석과 시뮬레이션을 통해서 시스템의 성능을 평가한다. 실시간과 비실시간 서비스를 구분하였으며 음성, 화상과 같은 실시간 서비스는 지연민감성을 고려하여 우선순위를 높게 고려하였다. 탑재교환기는 CRA와 DRA의 두가지 할당방식을 역동적으로 활용한 알고리즘을 사용하여 채널 효율을 높이는 우수한 기법을 소개한다.

Key Words : Full connectivity, onboard switch, priority, Continuous Rate Assignment (CRA), Dynamic Rate Assignment (DRA).

ABSTRACT

This paper analyzes the channel assignment techniques and their performance in the On Board Processing (OBP) satellite communication system. It suggests the new Continuous Rate Assignment (CRA) and Dynamic Rare Assignment (DRA) for improving the efficiency of channel assignment at the OBP switch. Mathematical analysis and simulation are given to evaluate the system performance. Aggregate real-time and non-real-time services are considered as different classes. Higher priority is given to voice and video real-time services to avoid delay variation. Onboard scheduler uses CRA and DRA ways to arrange the capacity allocation dynamically. An improved algorithm is given to make the channel more efficient by doing some evaluation of the switching matrix.

Ⅰ. INTRODUCTION

In recent years, there has been great interest in providing multimedia services using geostationary earth orbit (GEO) satellite communication system. Wide-area coverage and broadcast nature of the satellite medium allow long-distance communication which enables direct transmission via satellite instead of trespassing many routers in terrestrial network. Since multibeam technology allows frequency reuse between beams, system capacity is increased based on given satellite bandwidth.

Traditional satellite services provide one-way connection, from ISP to users, such as TV broadcast. The connection from user to ISP is not considered or that function is done by terrestrial network. Until last

* 명지대학교 통신공학과 (dukim@mju.ac.kr)
논문번호 : KICS2005-05-224, 접수일자 : 2005년 5월 31일

few years, some kinds of this return channel have not been developed for traffic from user to ISP such as the digital video broadcasting-return channel via satellite (DVB-RCS)[1] supported by the European Telecommunications Standards Institute (ETSI). However, within the development of Internet, requirement of providing full connectivity becomes more and more urgent. That is, the satellite acts as a provider of dynamic links for any pair of UTs when they want to build a connection between them.

Quality of service (QoS) is the ability of a network element (e.g. an application or host) to have some level of assurance that its traffic and service requirements can be satisfied. QoS of IP-based satellite network have become very popular for multimedia applications. The Internet Engineering Task Force (IETF) has proposed QoS architectures to provide guaranteed service level to different applications over terrestrial networks including integrated services (IntServ), differentiated services (DiffServ)[2] and multi-protocol label switching (MPLS)[3]. It is an urgent need for developing QoS architecture and doing some analysis of performance for satellite network[4]. For example, the time delay in transferring large files may not be a problem in data transmission, but, it is not the case in voice transmission[5].

To achieve efficient utilization of given limited bandwidth while maintaining the user QoS requirement, a suitable dynamic capacity allocation (DCA) is necessary for the onboard scheduler[6]. This paper emphasizes on the DCA algorithms for different types of services by using a switching matrix[7].

The paper is organized as follows. Section II describes the system architectures and discusses the transmission process for different types of services both from the UTs' view and from the onboard scheduler's view. Section III provides a mathematical model of the system under some assumptions. By creating some Markov Chains, throughput and delay are calculated for individual type of service. Section IV suggests an improved algorithm of DCA by evaluating the switching matrix. Delay versus throughput performance for different sets of parameter values is then examined in Section V and phenomena for different types of services are discussed under heavy traffic. Section VI provides concluding remarks.

Ⅱ. SYSTEM DESCRIPTION

There are totally N nonoverlapping zones. Each zone is covered by an uplink beam and downlink beam. The UT (User Terminal) acts as a router in terrestrial network. It may connect to an entire LAN, an ISP, an apartment, or just a single PC.

Each uplink (downlink) beam is composed of a control channel and an information channel. The former is used to send control information such as the connection requests generated from UTs or assignment broadcast from the satellite. And the latter is used to transmit packet after connection has been established.

Frame is the basic allocation unit with T_f time duration and T_f is equally divided into M slots with T_s time duration each. Slot is the elementary transmission unit that contains one packet.

Multimedia services are supported by the system with different priorities. Real-time services such as voice and video conferencing are delay sensitive. Non-real-time services like data, on the other hand, are more tolerant to delay.

Two categories of assignments are described as below:

- Continuous Rate Assignment (CRA): Using this category, the scheduler assigns a fixed number of slots in each frame to the UT for the entire duration of a message transmission. This is suitable for real-time service transmission.
- Dynamic Rate Assignment (DRA): Using this category, the scheduler assigns dynamic number of slots to the UT frame by frame. In another word, when system becomes busier, fewer slots will be assigned to the non-real-time services. The upper bound and lower bound of the assigned number of slots in each frame are determined by the UT's request.

In this way, real-time services using CRA have a

higher priority when compared with non-real-time services using DRA. However, DRA is suitable to transmit data such as big files for this type of service is tolerant to delay.

Completely separate buffers are designed onboard to store unassigned requests for every connection pair. For each connection pair, there are two kinds of buffers to store the two different types of requests individually. Requests in one buffer, same connection pair and same service type, are regarded to have the equal chances of getting the new assignment. For real-time requests, when a request is assigned, that request will be deleted from the real-time buffer. On the other hand, non-real-time request will not be deleted from the buffer until the transmission has completely terminated. This policy is necessary for CRA has invariable information while DRA is changeable according to the system traffic.

When UT has received a message, it analyzes the message and sends a corresponding request in the control channel. Meanwhile, it leaves the free state (FS) mode and enters the channel access (CA) mode. Since request only includes very simple information such as source zone, destination zone and service type, the request length is very small and the control channel is supposed to be large enough to transmit all the requests to the satellite successfully. No collisions happen during the request transmission. The request is only rejected when its corresponding buffer onboard is full. In that case, UT will retransmit that request after a round trip time delay (RTD).

UT keeps staying in CA mode until it has received some information, which indicates that the request has successfully be accepted by the satellite and stored in buffer. Then UT enters waiting transmission (WT) mode. In this system, we suppose UT keeps in the WT mode until it has got some assignment. Actually, in the practical case, UT may give up transmitting the message if the waiting time is too long.

When UT receives some assignment, it enters the message transmission (MT) mode. For real-time service, UT keeps in MT mode until the message has been completely transmitted. But for non-real-time service, when system traffic is heavy ,it may return to WT mode and wait for assignment again. After message has been completely sent, UT returns to FS mode and waits for new coming requests.

It is possible that UT can be in different modes if it has multi-process. Each process works for one message transmission simultaneously and independently. This always happens when traffic is heavy for UT, especially UT works as a router. In the following description, we suppose one UT only has one process to transmit message and can only be in one mode at the same time. For multi-process UT, we regard it as several different single process UTs. This is reasonable and simple for analyzing and description.

A switching matrix is designed onboard to record all the information for assigned requests. The capacity of the switching matrix is the total number of slots that can be assigned to requests during a frame period. The maximum value of the capacity should be $N \times M$, because each line (row or column) of the switching matrix can accommodate at most M slots assignments per frame.

In lots of papers, when a slot has been assigned to a message, it keeps being used by that message in the future frames until released. However, in this system, for real-time services, scheduler can assign different slots for a message in different frames while the total number of slots assigned for that message in each frame remains fixed. For non-real-time services, scheduler controls the total number of slots assigned to a message in each frame dynamically. In another word, it can increase or decrease the number of slots assigned for that message due to system traffic. Assignment information is broadcast to UTs frame by frame. Therefore, the channel throughput can be increased while the system becomes somewhat complicated.

At the beginning of each frame, onboard scheduler first collects real-time-requests in buffers, and assigns them to the switching matrix one by one. Requests, not assigned, are kept in buffer and wait for assignment again in next frame. Assigned requests are deleted from buffer and their information is kept in the switching matrix in future frames until they terminate. After CRA process for real-time requests has been finished, onboard scheduler does DRA process for non-real-time requests if there are still some free slots

left in the switching matrix. The only difference is that every non-real-time request is kept in buffer until the according message has been completely transmitted meanwhile each assigned non-real-time request occupies the switching matrix for one frame. The order that requests are assigned both in CRA and DRA can be randomly or be determined by some evaluation of the switching matrix. The former is analyzed in Section Ⅲ and the later is suggested in Section Ⅳ.

After CRA and DRA have been finished, onboard scheduler assigns the requests in switching matrix to individual slots. This is a proved way and all the requests in switching can be assigned without any conflicts[7].

Onboard scheduler broadcasts the information of assignment to UTs through the control channel. Real-time requests in switching matrix or non-real-time requests in buffers are released if their corresponding transmissions terminate.

Ⅲ. THEORETICAL ANALYSIS

To simplify analyzing and formulating, in the following models, we suppose that the fixed number of assigned slot in CRA is 1 for a real-time message. And one slot or nothing is assigned for a non-real-time message each frame in DRA. More generalized case will be simulated for illustration purpose in Section Ⅴ.

3.1 Request In Process

New coming requests are stored in buffers before the beginning of next frame. If buffer of some connection pair is full, the corresponding coming requests are blocked and retransmitted by UTs after one RTD. The variable R_r is defined as the number of coming new requests of a connection pair for real-time service while R_n is for non-real-time services. The number of coming new requests is supposed to obey the Possion distribution with parameter λ that can be divided into λ_r and λ_n for different types of services individually.

$$\Pr[R_r = k] = \frac{\lambda_r^k e^{-\lambda_r}}{k!} \tag{1}$$

$$\Pr[R_n = k] = \frac{\lambda_n^k e^{-\lambda_n}}{k!} \tag{2}$$

We define variables B_{r0} and B_{n0} to indicate the number of requests in buffer of a connection pair at the beginning of the frame (after the assignment process) while B_{r1} and B_{n1} are defined for the same meaning but at the end of the frame. So

$$\Pr[B_{r1} = k] = \begin{cases} \sum_{i=0}^{k} \Pr[B_{r0} = i]\Pr[R_r = k-i] & 0 \le k \le b_r \\ \sum_{i=0}^{k} \sum_{j=k-i}^{\infty} \Pr[B_{r0} = i]\Pr[R_r = j] & k = b_r \end{cases} \tag{3}$$

$$\Pr[B_{n1} = k] = \begin{cases} \sum_{i=0}^{k} \Pr[B_{n0} = i]\Pr[R_n = k-i] & 0 \le k \le b_n \\ \sum_{i=0}^{k} \sum_{j=k-i}^{\infty} \Pr[B_{n0} = i]\Pr[R_n = j] & k = b_n \end{cases} \tag{4}$$

Here, b_r is the number of buffer rooms for real-time requests of one connection pair and b_n is for non-real-time requests. We say one buffer room accommodates one request.

The request block rate due to full buffer can be calculated as the average number of rejected requests divided by the average coming new requests.

Real-time request block rate is given by

$$P_{br} = \frac{\sum_{i=0}^{b_r} \sum_{j=b_r-i+1}^{\infty} \Pr[B_r = i]\Pr[R_r = j](i + j - b_r)}{\lambda_r} \tag{5}$$

Non-real-time request block rate is given by

$$P_{bn} = \frac{\sum_{i=0}^{b_n} \sum_{j=b_n-i+1}^{\infty} \Pr[B_n = i]\Pr[R_n = j](i + j - b_n)}{\lambda_n} \tag{6}$$

3.2 CRA Process

The total number of buffer rooms onboard for real-time requests is $v = b_r \times N \times N$. In CRA process, we assign requests in rooms one by one at random. Here, the order of room selection to be assigned is random and each of them is tried once per frame. So there are totally v steps in CRA. For each room, if

there is a request in it, try to assign the request. Otherwise, do nothing and go on next step for anther room. The probability that a room is occupied by a request can be calculated as follows

$$\sigma = \sum_{i=0}^{b_r} \frac{i}{b_r} \cdot \Pr[B_{r1} = i] \tag{7}$$

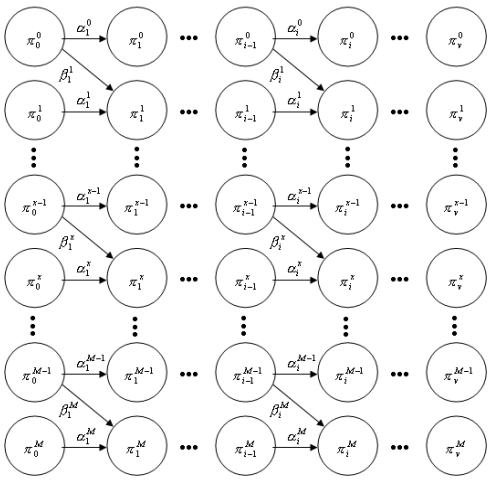


Fig. 1. π state transition diagram

We choose a row as an observed line of the switching matrix. (The same case is for choosing a column.) Define L_0 as the number of assigned requests in observed row before CRA and L_1 as the same meaning after CRA. The changing process from L_0 to L_1 by v steps can be described as in Fig.1 where π_i^x is the probability that after *ith* step there are x requests assigned per line in switching matrix.

During the *ith* step, when a request falls in the observed row, it can be assigned if both the corresponding row and column have less than M assigned request. The probability that a request is fall in the observed row is σ/N. Here, α_i^x is the probability that the request cannot be assigned when there are x assigned requests in observed row. And β_i^x is the probability that the request can be assigned when there are $x-1$ requests in observed row. Then

$$\alpha_i^x = \begin{cases} (1-\frac{\sigma}{N}) + \frac{\sigma \pi_{i-1}^M}{N} & 0 \le x \le M-1 \\ 1 & x = M \end{cases} \tag{8}$$

$$\beta_i^x = \frac{\sigma}{N} \sum_{y=0}^{M-1} \pi_{i-1}^y \quad 1 \le x \le M \tag{9}$$

Thus π_i^x can be described as follows

$$\pi_i^x = \begin{cases} \alpha_i^0 \pi_{i-1}^0 & x = 0 \\ \alpha_i^x \pi_{i-1}^x + \beta_i^x \pi_{i-1}^{x-1} & 1 \le x \le M \end{cases} \tag{10}$$

Within the initial case that

$$\pi_0^x = \Pr[L_0 = x] \tag{11}$$

And after CRA, we have

$$\Pr[L_1 = x] = \pi_v^x \tag{12}$$

A real-time service request is removed from the buffer if it is successfully assigned. Otherwise, it is kept in the buffer adding a tried and unassigned tag. This kind of tag is deleted after all the assignment in the current frame is finished. We define τ_i^l as the probability that after *ith* step, there is totally *l* tried and unassigned real-time requests kept in buffer.

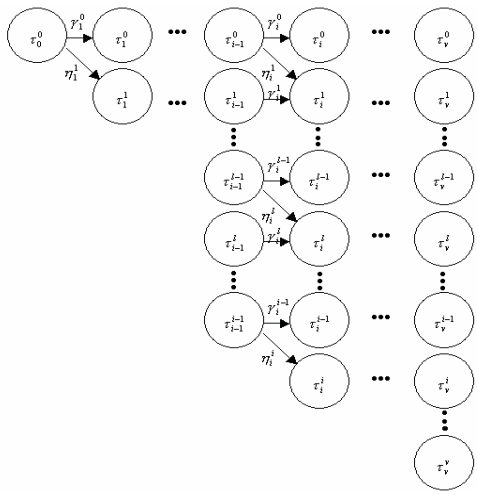


Fig. 2. τ state transition diagram

As described in Fig.2, γ_i^l is the probability that there is no tried and unassigned request to be kept in buffer during the *ith* step, based on *l* tried and unassigned requests in buffer previously. And η_i^l is the probability that there is a tried and unassigned

request kept in buffer during the ith step, based on *l-1* tried and unassigned requests in buffer previously. Then

$$\gamma_i^l = (1-\sigma) + \sigma \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} \pi_{i-1}^x \pi_{i-1}^y \quad 0 \le l \le i-1 \tag{13}$$

$$\eta_i^l = \sigma\left(2\pi_{i-1}^M - \pi_{i-1}^M \pi_{i-1}^M\right) \quad 1 \le l \le i \tag{14}$$

Thus, τ_i^l can be described as follows

$$\tau_i^l = \begin{cases} \gamma_i^0 \tau_{i-1}^0 & l = 0 \\ \gamma_i^l \tau_{i-1}^l + \eta_i^l \tau_{i-1}^{l-1} & 1 \le l \le i-1 \\ \eta_i^l \tau_{i-1}^{l-1} & l = i \end{cases} \tag{15}$$

Within the initial case that

$$\tau_0^0 = 1 \tag{16}$$

The probability of buffer room occupancy for request after CRA is given by

$$\varepsilon = \sum_{i=0}^{v} \frac{i}{v} \cdot \tau_v^i \tag{17}$$

For each connection pair, the number of real-time requests kept in buffer after CRA can be described as

$$\Pr[B_{r2} = k] = \binom{b_r}{k} \cdot \varepsilon^k (1-\varepsilon)^{b_r - k} \tag{18}$$

3.3 DRA Process

The process of DRA is similar with that of CRA, as shown in part B. The only difference is that a request is removed from the buffer only when that message is completely transmitted. Meanwhile some parameters using for real-time services should be modified to those for non-real-time services. In the following part, we will show the necessary modified equations. Those, same with part B, are omitted.

The total number of buffer rooms onboard for real-time requests is $v = b_n \times N \times N$ and the probability of buffer room occupancy before DRA is given by

$$\sigma = \sum_{i=0}^{b_n} \frac{i}{b_n} \cdot \Pr[B_{n1} = i] \tag{19}$$

It is obvious that there are L_1 requests (including only real-time requests) assigned in observed row before DRA, which we have got in part B. Define L_2 as the number of assigned requests (including real-time requests and non-real-time requests) in observed row after DRA. The initial case as in equation (11) is modified to

$$\pi_0^x = \Pr[L_1 = x] \tag{20}$$

And after DRA, we have

$$\Pr[L_2 = x] = \pi_v^x \tag{21}$$

A non-real-time service request will not be removed from buffer until that message terminates. So for a successful assignment, a non-real-time request has a probability of p_n to be removed from buffer. Here, p_n is the parameter of Geometric Distribution that a non-real-time message terminates in a frame. Equation (13) and (14) should be changed as follows

$$\gamma_i^l = (1-\sigma) + \sigma p_n \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} \pi_{i-1}^x \pi_{i-1}^y \quad 0 \le l \le i-1 \tag{22}$$

$$\eta_i^l = \sigma \cdot \left(1 - p_n \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} \pi_{i-1}^x \pi_{i-1}^y\right) \quad 1 \le l \le i \tag{23}$$

For each connection pair, the number of non-real-time requests kept in buffer after DRA can be described as

$$\Pr[B_{n2} = k] = \binom{b_n}{k} \cdot \varepsilon^k (1-\varepsilon)^{b_n - k} \tag{24}$$

3.4 Assignment Release Process

After the assignment information has been broadcast to the UTs, all the assigned slots for non-real-time services are released. For real-time service, some assignment slots are released if the services terminate. The probability that a real-time service terminates obeys the Geometric Distribution with the parameter p_r.

Since we choose the equilibrium point, following equations can be got.

$$\Pr[L_0 = i] = \sum_{j=i}^{M} \binom{j}{j-i} p_r^{\,j-i} (1-p_r)^i \Pr[L_1 = j] \quad 0 \le i \le M \tag{25}$$

$$B_{r0} = B_{r2} \tag{26}$$

$$B_{n0} = B_{n2} \tag{27}$$

3.5 Throughput and Delay Calculation

Throughput of the system, the quotient of current capacity divided by the maximum capacity of the switching matrix, can be calculated as

$$S_t = \sum_{i=0}^{M} \frac{i}{M} \cdot \Pr[L_2 = i] \tag{28}$$

Throughput that occupied by real-time services is given by

$$S_r = \sum_{i=0}^{M} \frac{i}{M} \cdot \Pr[L_1 = i] \tag{29}$$

And throughput that occupied by non-real-time services is given by

$$S_n = S_t - S_r \tag{30}$$

Delay is defined as the time period from the beginning that the source sends a request to the end that the message completely reaches the destination. From UT's view, for real-time service, delay can be divided into three parts. The first part is the period from the time that UT sends first request to the time UT receives the information that the request is successfully transmitted. The second part is the period from the end of first part to the time that UT gets first assignment. The third part is the period from the end of second part to the time that the message completely reaches the destination. Using Litter's result to calculate the second part by equation (5), we can get the average delay for real-time service that is

$$D_r = \left(\frac{RTD}{1-P_{br}} + \frac{T_f}{2}\right) + \frac{\sum_{k=0}^{b_r} k \cdot \Pr[B_{r2} = k]}{\lambda_r \cdot (1-P_{br})} + \left(\frac{T_f}{p_r} - \frac{T_f}{2} + RTD\right)$$

$$= \frac{(2-P_{br})RTD}{1-P_{br}} + \frac{\sum_{k=0}^{b_r} k \cdot \Pr[B_{r2} = k]}{\lambda_r \cdot (1-P_{br})} + \frac{T_f}{p_r} \tag{31}$$

For non-real-time service, since requests are removed from buffer only if that message terminates, the second part and the third part above are overlapped. Thus, using Litter's result to calculate the second part by equation (6) while omitting the overlapped period, we can get the average delay for non-real-time services that is

$$D_n = \left(\frac{RTD}{1-P_{bn}} + \frac{T_f}{2}\right) + \frac{\sum_{k=0}^{b_n} k \cdot \Pr[B_{n1} = k]}{\lambda_n \cdot (1-P_{bn})} + \left(RTD - \frac{T_f}{2}\right)$$

$$= \frac{(2-P_{bn})RTD}{1-P_{bn}} + \frac{\sum_{k=0}^{b_n} k \cdot \Pr[B_{n1} = k]}{\lambda_n \cdot (1-P_{bn})} \tag{32}$$

Take care that in equation (31), we used B_{r2}, which is the buffer state after CRA process. While in equation (32), we used B_{n1} that is the buffer state before DRA process. The reason is that in equation (32), the second part includes the third part when calculating delay for non-real-time service.

Ⅳ. AN IMPROVED ALGORITHM

In this section, we introduce an improved algorithm for CRA or DRA process. In Section Ⅲ, the order of selecting requests for assignment is random, while in the improved algorithm we select requests by doing some evaluation of the switching matrix. The purpose of the algorithm is to assign requests as many as possible under any given cases.

In the following description, we say the value of a position means the number of requests at a cross point of a row and a column. The value of a line means the number of requests in a row (or column) of the switching matrix. In order to compare with the random selection algorithm in Section III, we suppose a request is assigned one slot per frame. We define three states for requests. A request of assigned state means it has been assigned in previous frame. A request of new state means it is kept in buffer and to be assigned in current frame. The third one is called semi-assigned state whose definition is given in the algorithm. Steps below describe this algorithm.

1. Fill all the requests in switching matrix.
2. If there is totally *M* assigned requests in a line, that line is said to be in assigned state. And all the new requests are deleted from the switching matrix.
3. If a line has a value less or equal to *M*, that line is said to be in semi-assigned state. And the entire new requests in that line are changed to be in semi-assigned state.
4. Except for all the semi-assigned and assigned lines, check the remaining lines one by one using a round robin rule. Find the line, which has only one position containing new requests. Delete the new requests one by one until the line is changed to be the case in step.2 or there are no new requests left. This step ends until every unassigned line has more than two positions containing new requests.
5. Calculate the sum of the row value and column value for all the position containing new requests. Choose a position that has the maximum sum and delete a new request. Repeat this step until there appears a line that has only one position containing new requests. Then go to Step 4.
6. Check all the semi-assigned lines by round robin rule. If a line has a value larger *M*, delete the semi-assigned requests in that line until the line value equals *M*. After all the semi-assigned lines have been checked in this step, the whole algorithm ends.

Simulation and comparison are given in next section.

V. NUMERICAL RESULTS

In following description, RTD is supposed to be 250ms and each frame period is 50ms divided by M=5 slots . Symmetrical traffics are used for real-time service and non-real-time service with ($\lambda_r = \lambda_n = \lambda 2$) and ($p_r = p_n = 0.5$). Here, unit of request rate λ is the requests per frame of a connection pair and unit of delay is second.

Table 1. Comparison of analysis and simulation Results for Switching Matrix Size 4×4

REQUEST RATE λ	SYSTEM THROUGHPUT		DELAY FOR REAL-TIME SERVICE		DELAY FOR NON-REAL-TIME SERVICE	
	SIM.	ANAL.	SIM.	ANAL.	SIM.	ANAL.
0.2	0.3194	0.3200	0.6000	0.6000	0.6019	0.6000
0.4	0.6387	0.6400	0.6011	0.6007	0.6454	0.6478
0.6	0.9039	0.9052	0.6063	0.6057	0.9412	0.8948
0.8	0.9818	0.9809	0.6229	0.6272	1.6334	1.6712
1.0	0.9943	0.9905	0.6668	0.6711	3.0079	3.2122

Table 2. Comparison of analysis and simulation Results for Switching Matrix Size 10×10

REQUEST RATE λ	SYSTEM THROUGHPUT		DELAY FOR REAL-TIME SERVICE		DELAY FOR NON-REAL-TIME SERVICE	
	SIM.	ANAL.	SIM.	ANAL.	SIM.	ANAL.
0.05	0.2010	0.2000	0.6000	0.6000	0.6010	0.6000
0.10	0.4013	0.4000	0.6002	0.6000	0.6059	0.6027
0.15	0.5971	0.5998	0.6010	0.6004	0.6339	0.6387
0.20	0.7978	0.7982	0.6030	0.6021	0.7680	0.7804
0.25	0.9548	0.9543	0.6086	0.6060	1.4581	1.4629
0.30	0.9902	0.9908	0.6180	0.6153	2.4373	2.5135

Numerical examples of the system performance measures are studied using analysis and simulation. Since several approximate assumptions are made in the analysis of throughput and delay, comparisons of analysis and simulation are made in Table I and II for different sizes of switching matrix individually.

In table I, it is observed that the analytic results are very close to the simulation except for $\lambda = 1.0$. Even in this case, the discrepancy between simulation and analysis is less than 0.5 percent for throughput and 10 percent for delay. In Table II, under the same assumption as in Table I, we only change the number of zones from 4 to 10 and the switching matrix size is changed to 10×10. By using some proper traffic rate, we get the throughput values symmetrically distributed between (0,1). It is observed that the analytic results are much closer to the simulation compared with the discrepancy in Table I, especially for the case when throughput is near 1.

In the following simulated cases, we use symmetrical traffics for real-time service and non-real-time service with switching matrix size of 4×4.

In Fig. 3, throughput versus traffic rate is examined. When traffic is not heavy ($\lambda < 0.6$), system deals with the both types with same efficiency. But when traffic becomes heavier, more system capacity is assigned to real-time service while sacrificing non-

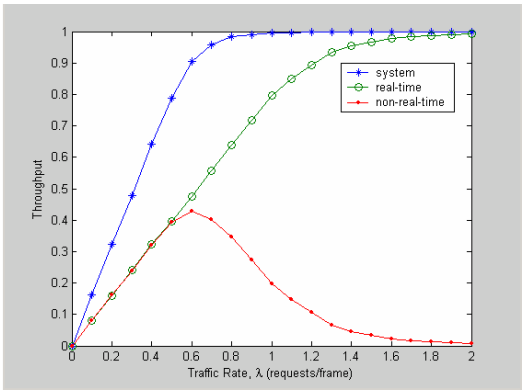


Fig. 3. $Throughput-\lambda$ curve $(p_r=p_n=0.5)$

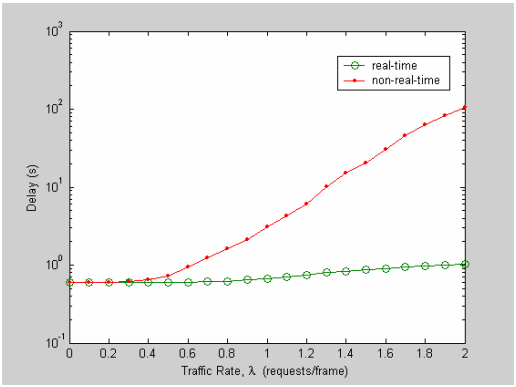


Fig. 4. $Delay-\lambda$ curve $(p_r=p_n=0.5)$

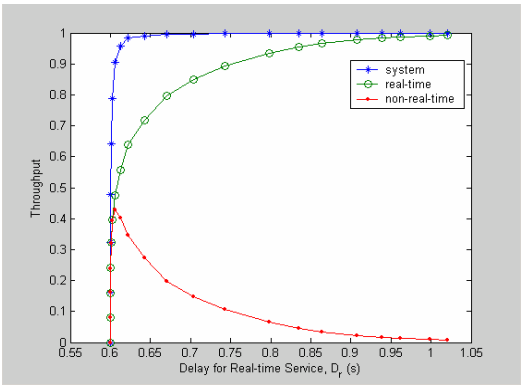


Fig. 5. $Throughput-D_r$ curve $(p_r=p_n=0.5)$

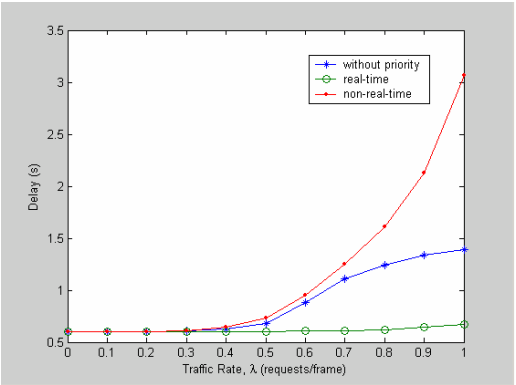


Fig. 6. $Delay-\lambda$ curve of priority comparison

real-time service. As a Result in Fig 4, delay for real-time service doesn't increase much while it increased exponentially for non-real-time service when traffic becomes heavier. Since non-real-time service is tolerant to delay, we may say that delay for real-time service is the critical requirement to be satisfied. As shown in Fig. 5, throughput versus delay of real-time service curve tells that high system throughput can be got with a short real-time service delay. And if that delay times, most part of the system capacity can be assigned for real-time service.

In this system, we give a higher priority to real-time service. We compare it with the case without priority for the symmetrical traffic given above. In Fig.6, with priority, delay for real-time service is decreased while for non-real-time service it is increased compared with the case without priority. However, the increased part doesn't matter much because non-real-time service is not sensitive to delay. In Fig. 7, the curve shows that it

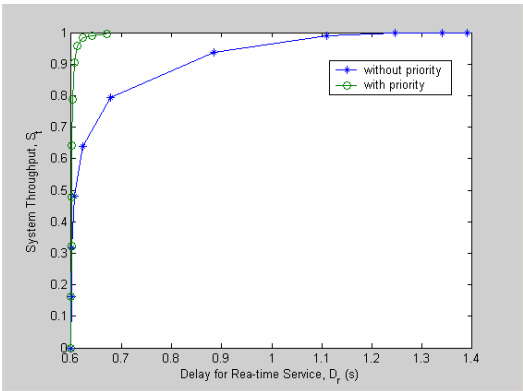


Fig. 7. S_t-D_r curve of priority comparison

takes shorter delay for real-time service to get a high system throughput if priority is supported. The advantage is obvious if system uses a priority policy.

Fig. 8 shows system throughput versus D_r for different parameters of p with symmetrical load traffic for both types of services. Here, we define $p=p_r=p_n$.

We can observe that the curve with a smaller p starts at a higher delay value when throughput is near zero, because smaller p indicates a longer message length and its necessary transmission time is inversely proportional to p. It is also observed that the curve with a smaller p runs to a high throughput (>99.5%) by a longer delay increment. The reason is that message with a smaller p should wait a longer time onboard for assignment (see in Fig.9) because other assigned messages are not easy to be released.

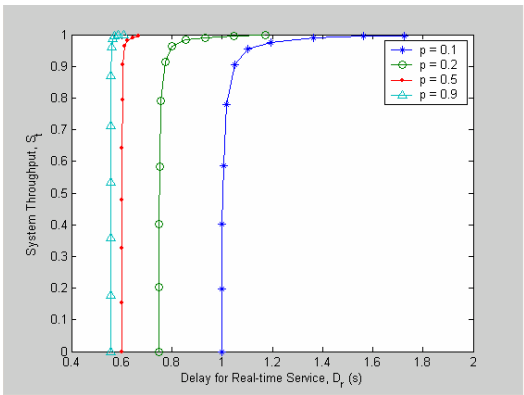


Fig. 8. $S_t - D_r$ curve with different p

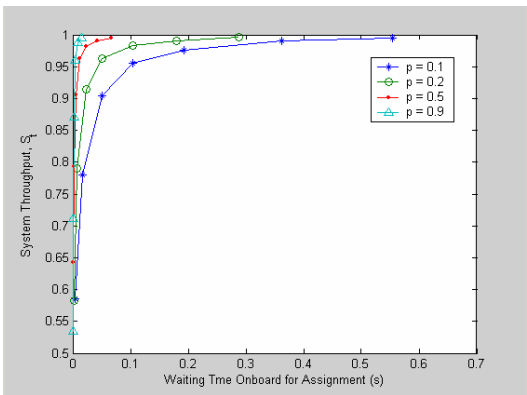


Fig. 9. $S_t - Waiting\ time$ curve with different p

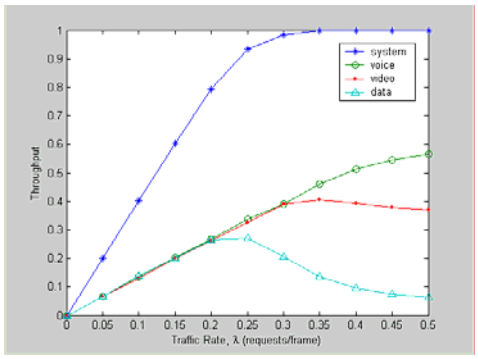
As we discussed above, system using a random selection assignment can get short delays for both types of traffic (See in Fig. 5) if $\lambda < 0.6$. In another word, a few collisions happen during the assignment when traffic is not heavy. So the comparison for random selection and the improved algorithm we suggested in Section Ⅳ under heavy traffic ($\lambda \geq 0.6$). In table III, it is shown that the improved algorithm gets a higher system throughput, which is much closer to 1. Meanwhile, delay for both types of services are reduced compared with the random selection assignment.

Table 3. Comparison fo Random selection assigment and improved Algorithm

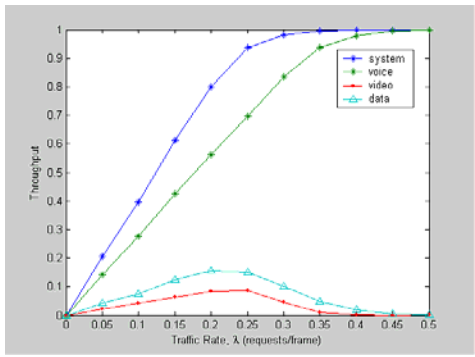
REQUEST RATE λ	SYSTEM THROUGHPUT		DELAY FOR REAL-TIME SERVICE		DELAY FOR NON-REAL-TIME SERVICE	
	RAN.	IMP.	RAN.	IMP.	RAN.	IMP.
0.6	0.9039	0.9043	0.6063	0.6053	0.9412	0.8802
0.7	0.9581	0.9705	0.6225	0.6124	1.2486	1.1668
0.8	0.9818	0.9897	0.6229	0.6200	1.6334	1.5435
0.9	0.9887	0.9957	0.6429	0.6390	2.1271	2.0850
1.0	0.9943	0.9973	0.6668	0.6541	3.0079	2.7960

In the following step, we discuss some more general cases that traffic consists of voice, video and data with different percentage of composition. Voice and video are regarded as real-time service while data is regarded as non-real-time service. Video traffic requires 6 slots to be assigned per frame, which is 6 times of the voice's requirement. M is modified from 5 to 10 in order to accommodate video traffic. In order to compare these three traffics, we use $p = 0.6$ for video message and $p = 0.1$ for voice and data message. Thus the total number of assigned slots is same for all the messages.

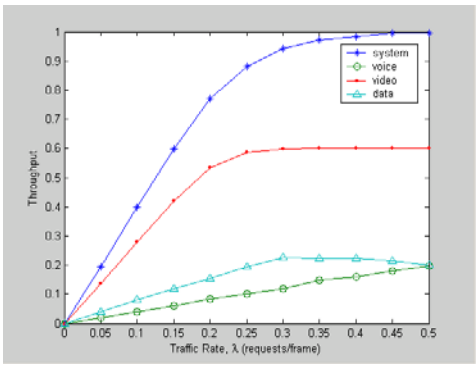
Fig. 10 (a) uses an equal load of traffic (1/3 voice, 1/3 video and 1/3 data). It is shown that when traffic is less than 0.25, all the messages are equally treated and the system throughput is three times of each traffic throughput. After traffic becomes heavier, data throughput decreases rapidly because of low priority. We can also observe that under a heavy traffic, voice has a higher throughput than video. The reason is that voice can get assignment even when there is fewer free slots in the switching matrix while video can only get assignment if there are at lease 6 free slots of its according row and column. Fig. 10(b) uses a voice dominant traffic (70% voice, 10% video and 20% data). It is observed that when traffic is heavy, almost all the system capacity is occupied by voice traffic. Fig. 10 (c) uses a video dominant traffic (10% voice, 70% video and 20% data), we can find that the maximum throughput for video is 60%. The reason is among the 10 slots in each frame, only 6 slots can be assigned to a video message and 4 left slots is not



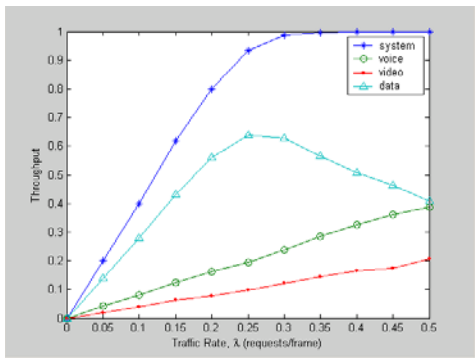
(a) equal load



(b) voice dominant



(c) video dominant



(d) data dominant

Fig. 10. $Throughput - \lambda$ curve of generalized case

enough to accommodate another video message. Fig. 10 (d) uses a data dominant traffic (20% voice, 10% video, 70% data). It is observed that data service can get a high throughput if the traffic is not heavy. When traffic becomes heavier, data throughput decreases while voice and video occupy more system capacity because of their higher priority.

Ⅵ. CONCLUSION

In this paper, we have investigated the structure and performance of a satellite communication system providing multimedia services. Different priorities are supported for real-time and non-real-time services using CRA and DRA individually. Mathematical model is created and theoretical analysis is compared with the simulation results. We also suggested an improved algorithm by doing some evaluation of the switching matrix. Performance using general traffic, including voice, video and data is analyzed by simulation. The following concluding remarks can be extracted from our results.

1) System with priorities can reduce the delay for real-time service while make the delay for non-real-time service longer.
2) High system throughput (more than 98 %) can be got with a short delay for real-time service.
3) By doing some evaluation of the switching matrix can make the system throughput higher and delay shorter especially under heavy traffic.
4) For general cases, when traffic is not heavy, system treats voice, video and data with same efficiency. When traffic becomes heavier, more system capacity is assigned to voice and video traffic by sacrificing data traffic. Voice traffic has a higher chance to be assigned than video traffic because of its fewer slots required per frame.

REFERENCES

[1] ETSI EN 301 790 v1.3.1, Mar. 2003.

[2] S. Blake et al., An architecture for differentiated services, RFC 2475, Dec. 1998.

[3] T. Ors, C. Rosenberg, "Providing IP QoS over GEO satellite systems using MPLS," *Int. J. Satell. Commun.* Vol. 19, pp.443-461, 2001.

[4] S. Kota and M. Marches, "Quality of service for satellite IP networks: a survey," *Int. J. Satell. Commun. Network.* Vol. 21, pp.303-349, 2003.

[5] T. Yum and M. Chen, "Dynamic channel assignment in integrated services cable networks," *IEEE Trans. Commun.*, vol. 42, no. 2/3/4, pp.2023-2027, 1994.

[6] T. Le-Ngoc, "Switching for IP-Based Multimedia Satellite Communications," IEEE J. Select. Areas Commun., vol. 22, no. 3, pp. 462-471, Apr. 2004.

[7] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," IEEE Trans. Commun., vol. com-27, no. 10, pp. 1449-1455, Oct. 1979.

Yue Teng ***Associate Member***

1999~2003 received the B.S. degree in computer science and engineering from Northeastern University, Shenyang, China;

2003~present pursuing for the M.S. degree in Communication Enginerring at Myongji University, Korea.

<Research interest> IP-based satellite network, Onboard switch, TCP over satellite.

Doug Nyun Kim ***Lifelong Member***

1971~1975 received the B.S. degree in Electrical Engineering from Seoul University, Korea;

1980-1981 received the M.S. degree in Electrical Engineering from SUNY at Stony Brook, USA

1985~1988 received the Ph.D. degree in Electrical Engineering from Auburn University, USA;

1988~1995 Research fellow with the Electronics and Telecommunications Research Institute (ETRI)

1995~present Professor with the Department of Communication Engineering, Myongji University.

<Research interests> Wireless Communication, High Speed Satellite Communication, Access Protocol.