실시간 주기적 메시지 스케줄링을 위한 여유시간 분할방법

정회원 유 해 영*, 심 재 홍**, 최 경 희**, 정 기 현***, 박 승 규**, 최 덕 규**

Laxity Decomposition Method for Scheduling Real-time Periodic Messages

Hae-Young Yoo*, Jae-Hong Shim**, Kyung-Hee Choi**, Gi-Hyun Jung***, Seung-Kyu Park**, Dug-Kyoo Choi** *Regular Members*

요 약

본 논문에서는 블록킹되지 않는 스위치를 통해 경로가 설정되는 실시간 메시지를 위한 효과적이고 분석적인 스케줄링 방법을 제시하고자 한다. 스케줄링 가능한 메시지들의 여유시간을 분할하여 이를 여유테이블에 배치하여 관리한다. 이 테이블을 이용하여 다음에 스케줄링될 패킷을 대각선열에 배치한 트래픽 행렬을 작성한다. 수정된 MLF-SDR 알고리즘을 이 트래픽 행렬에 적용하여 스케줄링 한다. 이 알고리즘의 적용 예를 간단히 보이고, 알고리즘의 성능을 시뮬레이션하여 보았다. 실험 결과 I/O 포트 수가 적은 스위치에 대해 스케줄링 성공률이 매우 높다는 것을 확인하였다.

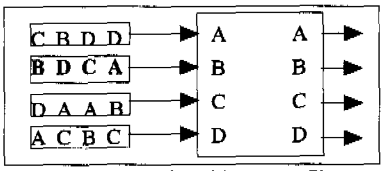
ABSTRACT

This paper presents a very unique and analytic method for scheduling real time messages routed through a non-blocking switch. The laxities of schedulable messages are decomposed and the laxity table is rearranged so that the packets to be scheduled in the next time instance are placed in the diagonal of the traffic matrix. And the modified MLF-SDR algorithm is performed on the table. We present some examples and simulation results which show that the success rate is very high, even when the size of switch is large.

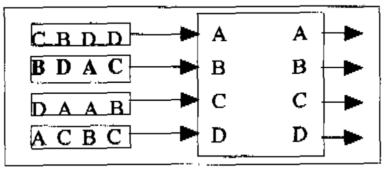
Ⅰ. Introduction

Non-blocking switch such as crossbar switch is utilized to connect input links with their distinct output link counterparts. As an application, the satellite-switched time-division multiple access (SS/TDMA) method is used to link a number of earth stations. Under the technique, a packet is transmitted during each time slot, and the switch can receive at most one packet from each input port and transmit at most one packet to each output port. If more than one packet from different input port are transmitted to an output port at the same time, the output port cannot handle both of packets in their deadlines. This conflict may force undelivered packets to be

* 단국대학교 전산통계학과(yoo@dankook.ac.kr)
** 아주대학교 정보통신대학 정보 및 컴퓨터공학부(khchoi@madang.ajou,ac,kr)
*** 아주대학교 공과대학 전기전자공학부(khchung@madang,ajou.ac.kr)
논문번호 : 98515-1127, 접수일자 : 1998년11월27일
※ 본 연구는 정보통신 진흥연구원과 아주대학교의 지원에 의하여 이루어진 것임.



(a) An example with no conflict



(b) An example with conflict

Fig. 1 Examples of switch schedule

discarded and reduce the performance of switch.

Figure 1 (a) and (b) show an example of a 4 × 4 switch and messages with four packets. The packets on each input port are labeled with the letters for destination output ports. In fig. 1 (a), during each time slot, only one packet arrives at each input port, and only one packet is transmitted to one of four distinct output ports. There are no conflicts in (a) and therefore all packets are successfully transmitted to their destinations in their deadlines. But in (b), the packets labeled C in the first time slot and A in the second time slot are conflicted and thus can not be transmitted in their deadlines.

The previous works [1]-[3] on SS/TDMA have concentrated on finding scheduling algorithms for data with no deadlines or the same periodicity. They are not suitable to real-time applications that contains randomly arriving tasks. The targets of time slot assignment (TSA) in the SS/TDMA are to find a less (or no if possible) conflict packet assignment technique and to reduce the number of switching nodes. The first target helps to minimize the time to schedule while maximizing switch utilization. The second target is to simplify the complexity of switch.

Paper [4] proposed *system of distinct representatives* (*SDR*) algorithm. The paper showed that, for an N N switch, if all messages have the same period and the utilization of both input and output ports are equal to 1.0, then the SDR algorithm can schedule the messages without any conflicts.

In paper [5] the authors introduced a periodic message model. In the model, the periods of messages are different and their relative deadlines of messages vary. They assume the maximum allowable time for transmitting a message is equal to the period of message. They handled the multiple-period time-slot assignment (MP-TSA) problem. To see the performance, their heuristic algorithms based on the minimum laxity first (MLF), the earliest deadline first (EDF), and the SDR were simulated. The simulation results showed that the success rate (defined as the percentage of time that the algorithm can find a conflict-free schedule) was not that satisfactory. For some cases with the size of switch greater than six or seven, the rate approaches zero and the switch becomes useless. To improve the performance they introduced several costly techniques such as adding extra buffers, extending the deadlines of messages and heuristically swapping the messages.

In this paper, we propose a very analytic way to avoid message conflicts and to improve the success rate. The proposed algorithm is based on the MLF but much different. In the algorithm the laxities of all messages are decomposed so that the decomposed laxities become prime numbers. And according to the decomposed laxities, the modified MLF-SDR algorithm is performed in a very unique way. Some simulations show that the performance of the proposed algorithm is better than those of the previous works and the success rate increases dramatically. Even when the number of switch I/O ports is large enough for actual applications, the success rate is high enough to be utilized.

The rest of this paper is organized as follows. In section two, we describe briefly the model of switch system and introduce some notations. The suggested scheduling algorithm is presented in section three. Section four demonstrates how the algorithm works through an example. The results of simulation and analysis are described in section five. Conclusion is followed in section six.

Ⅱ. System Model and Notation

In the system model a message *Mi*s are generated periodically where $0 \quad i \quad L$. The message is characterized by four parameters and presented as (*srci*, *dsti*, *ci*, *pi*) where *srci* is the source port, *dsti* the destination port, *ci* the length of message (a multiple of packets) and *pi* its period (measured by the number of packets). In every *pi* period a message *Mi* containing *ci* packets is transmitted from *srci* to *dsti*. The *j*th instance of message *Mi* is presented as *Mij*.

We assume that the relative deadline of *Mi,j* is equal to the period of message. This implies that no preparation time is required between two consecutive transmissions. By the relative deadline the transmission of *Mi,j* must be completed. Any message not to be completed until its deadline is discarded.

The laxity of a message *Mi,j* at a time instant *t* is defined as *di,j* - *t* - *ci,j*, where *di,j* is the relative deadline of *Mi,j* and *ci,j* is the number of messages remaining to be transmitted at *t*. A non-blocking *N* *N* switch is used for transmitting messages. The *srci* is one of the *N* input ports, and *dsti* is one of the *N* output ports. The switch has no buffers. Thus if more than one packet arrive at the same destination at the same time slot, a conflict occurs. The utilization of a periodic message *Mi*, *Ui* is defined as *Ui* = *ci*/*pi*. The utilization of an input link *k*, *U*(*k*) is equal to *i* *U*(*i*) where *srci* = *k*. And the definition of utilization for an output link is similar. In paper [6], authors proved that when the utilization is less than or equal to 1.0, the EDF algorithm can schedule all schedulable messages in their deadlines. Mok [7] also showed that the MLF algorithm is optimal.

In paper [4], the authors used the system of distinct representatives (SDR) algorithm. The SDR algorithm is based on the information provided by system traffic matrix. A traffic matrix *T* is an *N* *N* table. At time *t*, the *j*, *k*th element of *T*, *T*[*j*,*k*] contains a descriptor for each message *Mi* with *srci* = *j*, and *dsti* = *k*. The *Mi* has packets ready to be transmitted. The descriptor includes an identifier, the number of packets remaining to be transmitted for the current message instance, and the deadline of message instance (equal to the period). The SDR algorithm chooses elements from matrix *T* which is derived from the traffic matrix *T*. The construction of *T* will differ depending on the algorithm being used. The SDR algorithm chooses as many elements from *T* as possible with the constraint that among the chosen elements, there is at most one element from each line of *T*. (A line is either a row or a column of a matrix.) This is because at most one packet on each input port and output port is scheduled during each slot. The set of chosen elements is called system of distinct representatives of packets in *T*.

In paper [5], the authors introduced the minimum laxity first using systems of distinct representatives (MLF-SDR) algorithm. The MLF-SDR algorithm combines the SDR algorithm and the MLF algorithm. The algorithm uses the traffic matrix *T* as input. During each time slot *t*, the MLF-SDR algorithm does the following. Let *l1* < *l2* < · < *lm* be a list of all distinct laxities of all message instances contained in *T*. The MLF-SDR algorithm first constructs matrix *T* such that if *T*[*j*, *k*] contains a message instance with minimum laxity *l1*, then the message instance is added to *T*[*j*, *k*]. The SDR algorithm is then run on *T*. Once the system of distinct representatives is chosen from *T*, a packet from each chosen *T*[*j*,*k*] is scheduled at *t* on the corresponding input and output ports, and the number of packets in *T*[*j*,*k*] for the message instance is decreased by one. After as many packets with laxity *l1* as possible are scheduled, a new matrix *T* is constructed from all message instances in *T* with laxity *l2* such that *T*[*j*, *k*] is not in a row or column that was already chosen in *T* from the earlier step. Algorithm SDR is then run on the new *T*, and the packets which correspond to the chosen representatives are scheduled. This process is repeated for each of *li* (*1* *i* *m*).

Ⅲ. Laxity Decomposition Algorithm

In this section we suggest the laxity decomposition algorithm. The algorithm consists of two major parts. In the first part, the laxity is decomposed by making the laxity several small prime numbers and a new decomposed traffic matrix is constructed. The second part applies the modified SDR algorithm to the new decomposed traffic matrix that was made in the first part.

The descriptor for each message in the traffic matrix *T* is presented as *Mi*(*ci* / *pi*) where *Mi* is the message identifier, *ci* is the number of packets remaining to be transmitted for the current message instance, and *pi* is the period (relative deadline) of message instance. In the first part of decomposition procedure, a fractional number is generated with *ci* as its numerator and *pi* as denominator. If *pi* is a prime number, the laxity decomposition procedure is not applied and the message *Mi* is scheduled as in the normal MLF-SDR algorithm. In the case when *pi* is not a prime number, the fractional number *ci* / *pi* is decomposed into several small numbers.

In the case that *pi* is not a prime number, there exists a set of divisors of *pi*, say { *s1, s2,* ... , *sk* } where *sk* is greater than 1 and less than *pi*, and *sj* is less than *sj+1*. Meanwhile *ci* can be expressed as a polynomial, *ci* = *rksk* + ...+ *r2s2* + *r1s1*, where *sj*s are the divisors of *pi* and *rj*s are constants. For example, Mi(7/20) is decomposed as Mi(5/20 + 2/20), since 5 and 2 belong to the set of divisors of 20, {2, 4, 5, 10} and 7 can be expressed as 1*5 + 1*2. Through the decomposition, the message *Mi*(7/20) is decomposed and two messages with different laxities with the same period, *Mi1*(5/20) and *Mi2*(2/20) are made. The messages *Mi1*(5/20) and *Mi2*(2/20) are treated as *Mi1*(1/4) and *Mi2*(1/10). This implies that the message *Mi* that has 7 packets which must be transmitted in its relative deadline 20 can be treated as two different messages, one message *Mi1* with a packet to be transmitted in every 4 time slots, another one *Mi2* with a packet in every 10 time slots.

To decompose a message with more than one packet into several messages with small number of packets, we use the prime number decomposition rule. Instead of scheduling a message with many packets in a short period, it is easier to schedule and transmit several decomposed messages with small number of packets in a relatively longer period. By the decomposition, the packets contained in a message could be scheduled evenly before its deadline. This improves the flexibility of scheduling and thus increases the success rate significantly.

Now let a decomposed traffic matrix be *T**. The *T** is modified in every time instance and suppose that *T*t* is the modified *T** at time *t*. The second part of the proposed algorithm is applied to *T*t* so that *N* messages to be transmitted are selected and scheduled at time instance *t*. The second part is constructed as three steps: obtaining *T*t* from *T*t-1*, applying the algorithm to select *N* messages from *T*t*, and updating *T*t* with the new numbers of remaining packets and the new relative deadlines.

In the first step, the procedure to extract *T*t* from *T*t-1* is done by *exchanging* the rows and columns appropriately. The packets to be possibly transmitted are chosen so that they will be repositioned in the diagonal of *T*t* from *T*t-1* and the laxity of *T*t* [*j,j*] must be greater than or equal to *T*t* [*j+1,j+1*]. The procedure starts from choosing messages for *T*t* [*1,1*] from *T*t-1*. The first message *T*t* [*1,1*] becomes the one with the minimum laxity among messages to be transmitted in *T*t-1*. If the number of messages with the minimum laxity is only one, the content of *T*t* [*1,1*] is obviously determined. But if several messages have the same minimum laxity, then choose a message in the row which includes the least number of messages with the minimum laxity and in the column which includes the most number of messages with the minimum laxity. In an extreme case, the number of such messages may be greater than one. In the case, any message can be chosen heuristically. Once *T*t-1*

$[l,m]$ is chosen as $T^*t\ [1,1]$, then row l and column m in $T^*t\text{-}1$ are moved to the first row and column in T^*t, respectively. So $T^*t\text{-}1\ [l,m]$ becomes $T^*t\ [1,1]$. $T^*t\ [2,2]$ is chosen next. Other contents of $T^*t\ [j,j]$, where $3\ \ j\ \ N$, are chosen similarly. According to the choice of messages, the orders of row entries (input port order) and column entries (output port order) change. That is, row entries and column entries of T^*t are different from those of $T^*t\text{-}1$.

After constructing T^*t from $T^*t\text{-}1$, the second step is to select N messages from the table. If all contents in the diagonal have at least one packet to be transmitted (ci is greater than zero), they are all chosen and scheduled to be transmitted. But there are some undesirable cases not to be able to select diagonal messages. The first case is that the content of any diagonal entry is empty (no messages). The second case is that the packets of message have been starved, that is, $ci = 0$. Lastly there are some cases that if a packet with the smallest laxity is chosen, it would be impossible to select all N packets. Now let such a message be $T^*t[j,j]$, where $1\ \ j\ \ N$. In any above case, instead of selecting $T^*t[j,j]$, choose another message, say $T^*t[j,k]$, where $k\ \ j$, with the minimum laxity in the same row that the starvation has occurred. The message is obviously not in the diagonal of matrix T^*t. Since a message not in the diagonal is selected, a conflict may occur between $T^*t[j,k]$ and $T^*t\ [k,k]$ in the same column. Therefore, $T^*t\ [k,k]$ can not be selected. Another message $T^*t\ [k,m]$, where $m\ \ k$, is chosen among the messages in the kth row by considering the minimum laxity. Again $T^*t\ [k,m]$ may produce another conflict with a diagonal message in $T^*t\ [m,m]$. so another message $T^*t\ [m,n]$, where $n\ \ k$ and $n\ \ m$ has to be chosen in the mth row. The procedure of the second step is to avoid possible message starvation and to improve the utilization.

As the final step of the proposed algorithm, T^*t is modified and prepared to make T^*t+1. All relative deadlines are decreased by one and the numbers of packets of the selected messages in time t are also decreased by one. If the relative deadlines become zero, the deadline of message should be set to pi, the number of packets to be transmitted to ci. If all packets in the traffic matrix T^*t had been scheduled before their deadlines but some packets in the original message have still remained waiting for being scheduled, the empty entries are replenished in advance even though their new decomposed periods do not start yet. For example, consider a message whose period is 10 and the number of packets is 3. As described above, the message is decomposed into two messages, say MA and MB with pA=5, pB=10, cA=1, and cB =1, respectively. If all packets of MA and MB have been already scheduled by the third time slot, then at the fourth time slot the number of packets remaining to be transmitted and the relative deadline of message MA are increased by cA and pA respectively. However, if the relative deadline of the decomposed message is equal to the relative deadline of its original message, the replenishment procedure should not be done. (In above case, MB cannot be replenished.) Since this replenishment reduces the chance of starvation in traffic matrix T^*t, the procedure contributes to increasing the success rate.

Ⅳ. Examples

In this section, the detail process of the proposed algorithm is described through an example. Let us consider a message set depicted in table 1. The deadlines are not shown since we have assumed that they are equal to their periods. The MLF-SDR algorithm described in [5] could not schedule this message set, but the following example shows that the message set becomes schedulable by the proposed algorithm.

The MLF-SDR algorithm fails to find three packets without missing the deadlines (note that a 3 ×3 switch is used). At the first time slot, M1, M5 and M7 are selected since they have the minimum laxity 1. At the second time slot, M4, M2, and M7 are selected. At the third time slot, *M1*, *M5* and *M7* are selected.

Table 1. Original message set (U = 1.0)

Message Id	Source	Destination	Number of Packets	Period
M1	*A*	*A*	1	2
M2	*A*	*B*	1	5
M3	*A*	*C*	3	10
M4	*B*	*A*	1	2
M5	*B*	*B*	1	2
M6	*C*	*B*	3	10
M7	*C*	*C*	7	10

Table 2. Decomposed message set

Message Id		Source	Destination	Number of Packets	Period
M1		*A*	*A*	1	2
M2		*A*	*B*	1	5
M3	*M31*	*A*	*C*	1	5
	M32			1	10
M4		*B*	*A*	1	2
M5		*B*	*B*	1	2
M6	*M61*	*C*	*B*	1	5
	M62			1	10
M7	*M71*	*C*	*C*	1	2
	M72			1	5

Table 3. Initial traffic matrix

Destination / Source	*A*	*B*	*C*
A	1/2	1/5	1/5, 1/10
B	1/2	1/2	
C		1/5, 1/10	1/2, 1/5

Table 4. Schedule produced by the proposed algorithm

Time slot / Source link	1	2	3	4	5	6	7	8	9	10
A	*M1*	*M2*	*M1*	*M3*	*M1*	*M2*	*M1*	*M3*	*M3*	*M1*
B	*M5*	*M4*	*M5*	*M4*	*M5*	*M4*	*M5*	*M4*	*M4*	*M5*
C	*M7*	*M7*	*M7*	*M6*	*M7*	*M7*	*M7*	*M6*	*M6*	*M7*

At the fourth time slot, the laxity of *M4* for output link *A* (the first column in traffic matrix) becomes zero, and the laxity of *M7* for output link *C* becomes three. And the packets from message instances, *M4* and *M7* with the smallest laxity are scheduled. However *M2* and *M5* in the second column and the first or second rows (note that the rows represent the input links) of the traffic matrix have no more packets remaining to be transmitted. And thus it is impossible to select packets for output link *B*.

But the proposed algorithm can schedule the message set as following. According to the decomposition technique described in previous section, the message set is decomposed as shown in table 2.

Now lets see how the proposed algorithm schedules successfully the message set. The initial traffic matrix is shown in table 3.

Table 4 shows the packets scheduled by the proposed algorithm until time slot ten, which is the least common multiple of three periods: two, five and ten. At the first time slot, *M1*, *M5*, and *M7* are scheduled, since they have the minimum laxity 1. At the second time slot, *M4*, *M2*, and *M7* are scheduled since the laxity of *M4* is zero, and *M2* and *M7* are selected among four message instances *M2*, *M3*, *M6*, and *M7* with the same laxity four. In a similar way, *M1*, *M5* and *M7* are scheduled at the third time slot. At the beginning of the fourth time slot, the number of packets and the relative deadline of *M7* are replenished, because its number of packets has been zero and the original deadline is equal to ten. Whether the number of packets is replenished or not, the proposed algorithm schedules *M4* with the laxity of zero, *M3* and *M6* with the laxity of one. Table 5 depicts the traffic matrix at the fourth time.

Table 5. Traffic matrix replenished at the fourth time slot

Destination / Source	*A*	*B*	*C*
A	0/1	0/2	1/2, 1/7
B	1/1	0/1	
C		1/2, 1/7	1/3, 1/7

V. Simulation Results

We randomly generate *N*2 periodic messages with a fixed schedule length, *L*. The input and output ports for each periodic message are chosen arbitrarily from the *N* input and *N* output links. The period *pi* of each message is initially set to the schedule length, *L*. The number of packets, *ci*, generated in each message is drawn from a uniform distribution in the range [1, 0.5**pi*]. The generated message is regarded as a valid message if the utilization of its input and output links does not exceed 1.0. Otherwise, the message is discarded. With the procedure, the average link utilization *U* never reaches to 0.95. Therefore, to make the utilization 0.95, we add new messages or modify the generated messages. Since the generated messages has the same period, we make the number of packets and period relatively prime numbers while making the periods of messages different. In the simulations, 1000 randomly generated message sets are used.

We will compare our proposed algorithm with the MLF-SDR algorithm in [5]. The authors in [5] also introduced several heuristic algorithms such as MLF-SDR, EDF-SDR, MLF-RR, and EDF-RR. But among the algorithms, the MLF-SDR shows the best success rate.

Figure 2 shows the success rate for different schedule lengths by the proposed algorithm (dec-mlf-sdr) and the algorithm (mlf-sdr) in [5]. Schedule length (*L*) varies from 12 to 420. For average link utilization (*U*) and switch size (*N*), *U* = 0.95 and *N* = 4 are used to compare with the result shown in [5]. For all schedule lengths, the proposed algorithm in this paper shows higher success rates. Especially, even when schedule length is very high greater than 60), the success rates remains almost constant. This is due to the fact that decomposed packets are scheduled relatively evenly in a long period and they get a better chance to be scheduled. This is not the case for the algorithm shown in [5].

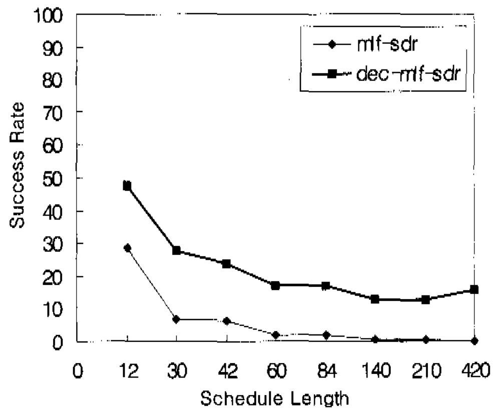


Fig. 2 Success rates for various schedule lengths(N = 4, U = 0.95)

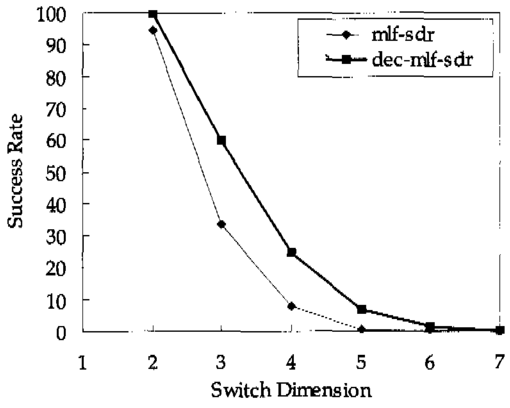


Fig. 3 Success rates for various switch sizes (L = 30, U = 0.95)

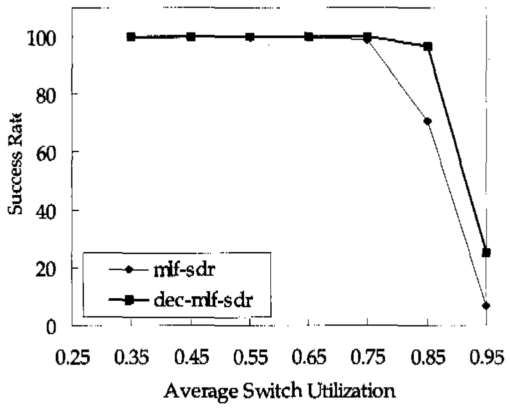


Fig. 4 Success rates for various switch utilization factors(N = 4, L = 30)

Figure 3 shows the success rate for different switch sizes. Switch size varies from 2 to 7. If N is greater than 6, the success rates drop nearly to zero in both algorithms. For switch utilization factor (U) and schedule size (L), $U = 0.95$ and $L = 30$ are used. Our algorithm shows higher success rates for all cases.

Figure 4 shows the success rates for different switch utilization factors. Switch utilization factor U varies from 0.35 to 0.95. When utilization factor is smaller than 0.75, both algorithms show nearly 100 percent success rate. But if utilization factor is greater than 0.75, our algorithm shows its superiority.

Ⅵ. Conclusion

We proposed an efficient packet switching algorithm. One message with a large laxity is divided into small messages with smaller laxities. The decomposition of laxities increases the chance for messages to be scheduled appropriately and improves success rate. To reduce the starvation of messages in traffic matrix, the proposed algorithm carefully rearranges the contents of traffic matrix and reduces the possibility of starvation. The simulation results showed that the proposed algorithm outperformed the algorithm shown in the previous studies.

References

[1] M. A. Bouucelli, I. Gopal, and C. Wong, "Incremental Time-Slot Assignment in SS/TDMA Satellite Systems", *IEEE Trans. on Comm.*, Vol. 39, No. 7, pp.1147-1156, July 1991.

[2] S. Chalasani and A. Varma, "Efficient Time-Slot Assignment Algorithms for SS/TDMA Systems with Variable Bandwidth Beams", *IEEE Trans. on Comm.*, Vol. 24, No. 2/3/4, pp.1359-1370, Feb./Mar./Apr. 1994.

[3] W. T. Chen, P. Sheu, and J. Yu, "Time Slot Assignment in TDM Multicast Switching Systems", *IEEE Trans. on Comm.*, Vol. 42, No. 1, pp.149-164 Jan. 1994.

[4] T. Inukai, "An Efficient SS/TDMA Time Slot Assignment Algorithm", *IEEE Trans. on Comm.*, Vol. 27, No. 10, pp.1449-1455, Oct. 1979.

[5] I. Philp and J. Liu, "A Switch Scheduling Problem for Real-Time Periodic Messages", *submitted to Telecommunications Systems Journal*, http : //pertsserver.cs.uiuc.edu / papers / abstract-index.html.

[6] C. Liu and J. Layland, "Scheduling Algorithms for Multi- programming in a Hard-Real-Time Environment", *Journal of ACM*, Vol. 20, No. 1, pp.46-61, Jan. 1973.

[7] A. K. Mok, "Fundamental Design Problems of Distributed Systems for the Hard Real Time Environment", Ph.D Thesis, M.I.T., 1983.

유 해 영(Hae-Young Yoo) 정회원
1979년: 단국대학교 수학과(이학사).
1981년: 단국대학교 수학과(이학석사).
1994년: 아주대학교 컴퓨터공학과 박사과정 수료.
1983년~현재: 단국대학교 전산통계학과 교수.
<주관심 분야> 소프트웨어 공학, 시스템 프로그램.

심 재 홍(Jae-Hong Shim) 정회원
1987년 : 서울대학교 자연과학대학계산통계학과 (이학사).
1989년 : 아주대학교 공과대학 컴퓨터공학과 (공학석사).
1998년 : 아주대학교 정보통신대학 컴퓨터공학과 박사과정 수료. 현 아주대학교 시스템 s/w 연구실 연구원.
<주관심 분야> 운영체제, 분산처리 시스템.

최 경 희(Kyung-Hee Choi) 정회원
1976년: 서울대학교 사범대학 수학교육과(학사).
1979년:그랑데꼴 ENSEEIHT (엔지니어, 공학석사).
1982년:Paul Sabatier대 (공학박사).

1982년~현재 아주대학교 정보 및 컴퓨터공학부 교수.
<주관심 분야> 운영체제, 분산시스템, 실시간시스템, 프로그래밍 방법론.

정 기 현(Gi-Hyun Jung) 정회원
1984년 : 서강대학교 전자공학과 졸업(공학사).
1988년 : University of Illinois, EECS 공학석사.
1990년 : Perdue Univ(공학박사),
1992년~현재 아주대학교 전기전자공학부 부교수.
<주관심 분야> 멀티미디어, VLSI, 실시간 시스템.

박 승 규(Seung-Kyu Park) 정회원
1974년 : 서울대학교 공과대학 응용수학과 (공학사).
1976년 : 한국과학기술원 전산과 (이학석사).
1982년 : 프랑스 INPG ENSIMAG 전산학 박사.
1976년~1977년 : 한국과학 기술연구소 연구원.
1977년~1992년 : 한국전자통신연구소 부장.
1984년~1985년 : IBM 왓슨연구소 연구원.
1992년~현재 : 아주대학교 정보 및 컴퓨터공학부 교수.
<주관심 분야> 멀티미디어 구조, 이동컴퓨팅 구조, 다중프로세서 구조, 차세대 컴퓨터 구조

최 덕 규(Dug-Kyoo Choi) 정회원
1966년 서울대학교 공과대학 원자력공학과(공학사).
1984년 : Wright State University 전산과(석사).
1989년 : University of Massachusetts 전산학 박사.
1993년~1996년 : 국방과학연구소 책임연구원.
1999년~현재 : 아주대학교 정보 및 컴퓨터공학부 교수.
<주관심 분야> High speed local area networks, ATM and WATM protocols, Mobl-ie communication networks.