실시간 통신에서 가변 지연을 만족하기 위한 Multiple Rotating Priority Queue Scheduler

허 권† · 김 명 준††

요 약

실시간 스케줄러는 대역폭, 필요 버퍼량 등과 같은 네트워크 자원을 효율적으로 이용하면서 한정된 통신 지연(bounded delay)을 제공해야 한다. 이러한 제한 조건을 만족시키기 위해서 많은 스케줄링 방법론이 제시되었다. 그중 EDF 스케줄링 방법론이 최적의 성능을 갖는 것으로 알려져 있다. 그러나 EDF 스케줄링 방법론은 "sort"나 "search"와 같은 연산 작업을 수행함으로서, 과다한 오버헤드를 발생시킨다. Rotating Priority Queues(RPQ) 스케줄러는 EDF 연산 작업 없이 EDF 스케줄러에 근접한 성능을 갖는 스케줄러이다. 그러나 RPQ 스케줄러는 과다한 버퍼량을 필요로 한다. 본 논문에서는 이러한 문제점을 해결하기 위해서 Multiple Rotating Priority Queues(MRPQ) 스케줄러를 제시한다. MRPQ 스케줄러는 "block queue"라는 새로운 개념을 이용하여 회전 우선 순위 queue를 다중 계층으로 구성한다. 이렇게 구성된 MRPQ 스케줄러는 RPQ 스케줄러에서 필요한 버퍼량의 반 정도의 버퍼량만을 사용하여, RPQ 스케줄러와 동일한 동작을 수행한다. 또한 MRPQ 스케줄러는 RPQ 스케줄러와 동일한 최대 지연시간을 제공한다.

Multiple Rotating Priority Queue Scheduler to Meet Variable Delay Requirment in Real-Time Communication

Kwon Hur† · MyungJun Kim††

ABSTRACT

Packet schedulers for real-time communication must provide bounded delay and efficient use of network resources such as bandwidth, buffers and so on. In order to satisfy them, a large number of packet scheduling methods have been proposed. Among packet scheduling methods, an EDF (Earliest Deadline First) scheduling is the optimal one for a bounded delay service. A disadvantage of EDF scheduling is that queued packets must be sorted according to their deadlines, requiring a search operation whenever a new packet arrives at the scheduler. Although an RPQ (Rotating Priority Queue) scheduler, requiring large size of buffers, does not use such operation, it can closely approximate the schedulability of an EDF scheduler. To overcome the buffer size problem of an RPQ scheduler, this paper proposes a new scheduler named MRPQ (Multiple Rotating Priority Queue). In a MRPQ scheduler, there are several layers with a set of queues. In a layer, queues are configured by using a new strategy named block queue. A MRPQ scheduler needs nearly half of buffer size required in an RPQ scheduler and produces schedulability as good as an RPQ scheduler.

1. 서 론

최근 컴퓨터 네트워크를 기반으로 한, 화상회의 원격회의 VOD 등과 같은 많은 응용프로그램들이 실시간 통신을 필요로 한다. 이와 같은 응용프로그램들이 실시간 통신을 하기 위해서는, 근원지(source)에서 목적지(destination)까지의 경로(rou-te)를 결정하고, 그 경로에 포함된 교환기(switch)와 링크(link)의 자원을

† 준 회 원 : 신텔 정보통신
†† 정 회 원 : 충북대학교 컴퓨터과학과 교수
논문접수 : 1999년 8월 2일, 심사완료 : 1999년 9월 17일

해당 패킷들을 대표하여 할당받는 연결(connection)을 설정해야 한다. 이렇게 설정된 연결을 통해, 각 응용프로그램들은 한정된 통신 지연(bounded delay)을 보장하면서 통신을 해야된다[1, 2].

통신 지연은 교환 지연(switching delay), 스케줄링 지연(scheduling delay), 대기 지연(queueing delay), 전달 지연(propagation delay)으로 구성된다. 이러한 통신 지연 요소들은 고정 지연과 가변 지연으로 구분 할 수 있다. 고정 지연은 물리적으로 고정된 통신 지연을 갖는 것으로 교환 지연과 전달 지연이 있다. 가변 지연은 특정 스케줄링 정책에 따라 통신 지연 정도가 달라지는 것을 말한다. 가변 지연으로는 스케줄링 지연과 대기 지연이 있다[2, 3].

모든 연결의 주어진 지연한계(delay bound)를 만족하기 위해서는 스케줄링 지연을 최소화시키고, 각 연결의 주어진 지연한계를 고려하여 대기 지연을 조절해야 한다. 이러한 작업은 패킷들 간의 전송 경쟁을 제어하는 스케줄러에 의해서 이루어지며, 대표적인 스케줄링 방법론으로는 Fair que-ueing[10], Virtual Clock[11], Stop-and-Go[12], Hierarchical Round Robin[13], 정적 스케줄링(Static Scheduling)[1], EDF(Earliest Deadline First) 스케줄링[4, 5] 등이 있다. 이 중에서 EDF 스케줄링 방법론은 링크의 대역폭(bandwidth)을 최대로 활용할 수 있는 최적의 실시간 스케줄링 방법론으로 알려져 있다[5]. 그러나 EDF 스케줄러는 패킷이 입력될 때마다 검색 연산이나 정렬 연산을 수행하게 되어 스케줄링 지연이 가중되는 단점이 있다.

최근 간단한 연산 작업만으로 EDF 스케줄러와 유사한 동작과 성능을 갖는 RPQ(Rotating Priority Queue) 스케줄링 방법론이 개발되었다. RPQ 스케줄러는 우선순위 큐들의 우선 순위를 나타내는 색인 값을 회전 간격(rotating interval)이라는 일정시간 마다 변화시킴으로서 EDF 스케줄링과 비슷한 동작을 하게 된다[5]. 그러나 RPQ 스케줄링은 과다한 버퍼 량을 필요로 하는 문제점이 있다. 본 논문에서는 이러한 문제점을 해결하면서 RPQ 스케줄링 방법론과 동일한 최대 지연시간을 제공하는 MRPQ(Multiple Rotating Priority Queue) 스케줄링 방법론을 제시한다.

2장에서는 본 논문에서 다루고 있는 실시간 네트워크의 모델을 규정하고, 3장에서는 실시간 통신을 위한 RPQ 스케줄링 방법론에 대해서 자세히 설명한다. 그리고 4장에서는 RPQ 스케줄링의 문제점을 해결하기 위한 MRPQ 스케줄링 방법론을 제시한다. 5장에서는 RPQ 스케줄링 방법론과 비교하여 MRPQ 스케줄링 방법론의 성능을 알아보고 6장에서는 결론을 맺는다.

2. 네트워크 모델

네트워크 상에 존재하는 특정 교환기는 'W_k', 교환기 W_k에 존재하는 출력포트는 '$L_{k,z}$', 출력포트 $L_{k,z}$에 존재하는 스케줄러는 '$S_{k,z}$'라고 하고, 특정 연결은 'M_i'라고 한다. 스케줄러 $S_{k,z}$에서 제공하는 정적 우선 순위는 '$P_{k,z,j}$'라고 한다. 단, 낮은 값일수록 높은 우선 순위를 나타내고, 한 우선 순위를 한 개의 독립된 객체로 여긴다. 정적 우선 순위의 개수는 '$|P_{k,z}|$'라고 하고, 우선 순위가 '0'부터 시작되는 $|P_{k,z}|$개의 우선 순위 $P_{k,z,j}$로 구성된 집합을 우선 순위 집합 '$H_{k,z}$'이라고 한다.

$$H_{k,z} = \{P_{k,z,j} | P_{k,z,0} = 0, P_{k,z,1} = 1, \ldots, P_{k,z,|P_{k,z}|-1} = |P_{k,z}| - 1\}$$

우선 순위 집합 $H_{k,z}$중에 연결 M_i에 할당된 우선 순위 $P_{k,z,j}$는 '$p_{k,z,i}$'라고 한다. 그리고 $P_{k,z,j}$를 할당받은 모든 연결들의 집합을 $C_{P_{k,z,j}}$라고 한다. 연결 M_i가 경유하는 교환기 별로 미리 주어지면서 반드이 지켜져야하는 연결 M_i의 지연한계를 '$d_{k,z,i}$'라고 하고, 스케줄러 $S_{k,z}$가 보장할 수 있는 지연시간을 '$D_{k,z,i}$'라고 한다(즉, "$d_{k,z,i} \geq D_{k,z,i}$"를 항상 만족 시켜야 한다). 그리고 스케줄러 $S_{k,z}$에 입력된 연결들 중에 가장 긴 $d_{k,z,i}$를 갖는 연결의 $d_{k,z,i}$를 '$MAX_{k,z}$'라고 한다. 스케줄러 $S_{k,z}$에서 필요한 버퍼 량을 '$Q_{k,z}$'라고 하고, 링크의 대역폭은 'B'로 표기한다. 단 네트워크 상에 존재하는 모든 링크의 대역폭은 동일하다고 가정한다. 임의 시간 t 동안 전송할 수 있는 최대 데이터 량을 'Γt'라고 한다.

정적 스케줄러나 RPQ 스케줄러에서는 연결에 정적 우선 순위를 할당하기 위한 기준으로 '우선 순위 할당 범위' '$E_{k,z,j}$'라는 것을 결정한다. 예를 들어 연결의 지연한계가 (0msec, 3msec] 사이에 존재하는 연결에는 우선 순위 '0'을 할당하고, (3msec, 6msec] 사이에 존재하는 연결에는 우선 순위 '1'을 할당할 경우, (0msec, 3msec]는 우선 순위 '$P_{k,z,0}$'에 대한 우선 순위 할당 범위 '$E_{k,z,0}$'라고 하고, (3msec, 6msec]는 우선 순위 '$P_{k,z,1}$'에 대한 우선 순위 할당 범위 '$E_{k,z,1}$'라고 한다. 이때 우선 순위 할당 범위 간격은 3msec가 되고, 우선 순위

개수는 '2'가 된다.

3. Rotating Priority Queue 스케줄링 방법론

RPQ(Rotating Priority Queue) 스케줄러는 각 연결의 지연한계를 보장하기 위해서, (과정 1)과 같은 방법으로 각 연결에 정적 우선 순위를 할당한다. 그리고 (과정 1)에서 결정된 우선 순위의 개수가 $|P_{k,z}|$개일 경우, 스케줄러는 우선 순위가 '0'부터 시작되는 $|P_{k,z}|$개의 FIFO 큐를 보유한다. 각각의 FIFO 큐들은 자신들의 우선 순위를 나타내는 우선 순위 색인과 연결된다.

(과정 1) 연결 M_i에 대한 우선 순위 $p_{k,z,i}$ 할당 방법

㉮ 우선 순위의 개수 $|P_{k,z}|$와 우선 순위 집합 $H_{k,z}$을 결정한다.

$$|P_{k,z}| = \lceil \frac{MAX_{k,z}}{\Delta} \rceil$$
$$H_{k,z} = \{P_{k,z,j} | P_{k,z,0}=0, P_{k,z,1}=1, \ldots, P_{k,z,|P_{k,z}|-1}=|P_{k,z}|-1\}$$

㉯ 우선 순위 할당 범위 간격과 우선 순위의 개수를 이용하여 우선 순위 할당 범위 $E_{k,z,j}$를 결정한다. 단, 우선 순위 할당 범위의 간격은 회전 간격 Δ와 동일하다.

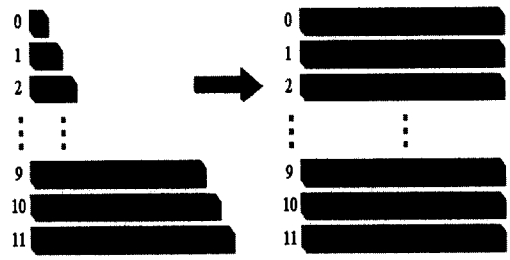
㉰ 특정 우선 순위 할당 범위 $E_{k,z,j}$에 포함되는 지연한계 $d_{k,z,i}$를 요구하는 연결 M_i에 해당 우선 순위 $P_{k,z,j}$를 부여한다.

$$p_{k,z,i} = \{P_{k,z,j} | P_{k,z,j} * \Delta \le d_{k,z,i} \le P_{k,z,j} * \Delta + \Delta\}$$

단, 모든 연결들 중에 가장 짧은 지연한계를 요구하는 연결의 지연한계는 회전 간격 Δ보다 커야 한다.

RPQ 스케줄러에 입력되었을 때, 각각의 패킷들은 해당 연결의 우선 순위에 해당하는 FIFO 큐에 저장된다. 그리고 RPQ 스케줄러는 우선 순위 순으로 FIFO 큐를 선택하고, 그 큐에 저장된 패킷들을 차례대로 전송한다. 단, 회전 간격마다 FIFO 큐들을 회전하여 큐의 우선 순위를 높여주게 된다. 이러한 결과, 큐에 저장된 패킷들의 우선 순위 또한 회전 간격마다 높아지게 된다. 회전 동작은 우선 순위 $P_{k,z,j}$를 갖는 큐에 입력된 패킷을 우선 순위 $P_{k,z,j}-1$을 갖는 큐로 복사하는 작업으로 이루어지는 것이 아니라, 단순히 큐와 연결된 색인 값을 정수 '1'씩 감소시킴으로서 회전 효과를 얻게된다. 단, 우선 순위 색인 값의 순환을 고려하여 가장 높은 우선 순위 색인 값을 갖는 우선 순위 색인은 가장 낮은 우선 순위 색인 값을 갖게 된다.

RPQ 스케줄링 방법론은 정적 스케줄링 방법론과 같이 각 연결에 정적 우선 순위를 부여한다. 그러나 RPQ 스케줄링 방법론이 정적 스케줄링 방법론과 다른 점은 큐를 회전시킴으로서, 큐에 저장된 패킷의 우선 순위가 회전 간격마다 높아진다는 것이다. 이러한 특성은 서비스를 받지 못한 시간을 고려하여 우선 순위를 할당하는 EDF 스케줄링 방법론과 유사하다. RPQ 스케줄러에서는 회전 간격을 좁게 하면 할수록 우선 순위를 결정하는 우선 순위 할당 범위 간격이 좁아지게 되므로 데드라인이 가장 가까운 패킷을 선택하게 될 가능성이 높아지게 된다. 그러므로 RPQ 스케줄러의 스케줄링 가능도(schedulability)를 EDF 스케줄러의 스케줄링 가능도에 근접하게 하기 위해서는 회전 간격을 줄이면 된다. 만일 한 패킷을 전송하는데 걸리는 시간과 회전 간격을 같게 한다면, 그 동작은 EDF 스케줄러와 같게 된다.



(그림 1) RPQ 스케줄러의 필요 버퍼 량

RPQ 스케줄러에서 우선 순위 $p_{k,z,i}$를 갖는 연결 M_i의 최대 지연시간은 $(p_{k,z,i}+1)*\Delta$이다[16]. 그리고 각 우선 순위 큐에서 요구되는 버퍼 량은 해당 큐에 입력된 패킷의 최대 지연시간 동안 최대로 전송할 수 있는 데이터 량과 동일하다. 그러나 큐를 회전함으로서, (그림 1)과 같이 스케줄러의 모든 우선 순위 큐에서는 가장 낮은 우선 순위 큐에서 요구되는 버퍼 량과 동일한 크기의 버퍼 량을 필요로 한다. 이러한 특성으로 인해, RPQ 스케줄러의 스케줄링 가능도를 높이기 위해서 회전 간격을 줄일 경우, 보다 많은 우선 순위를 가져야 되는 RPQ 스케줄러는 보다 많은 버퍼 량을 필요로 하

게 된다. 필요 버퍼 량과 회전 간격과의 관계는 (정리 1)과 같이 유도할 수 있다.

(정리 1) RPQ 스케줄러에서 필요한 버퍼 량은 식 (1)과 같다

$$Q_{k,z} = \Gamma\Delta\,(|P_{k,z}|)^2 \qquad (1)$$

$\Gamma\Delta$: 회전 간격 동안 최대로 전송할 수 있는 량
($\Gamma\Delta = B*MAX_{k,z} / |P_{k,z}|$)

4. Multiple Rotating Priority Queue 스케줄러

4.1 Multiple Rotating Priority Queue 스케줄링 방법론

RPQ 스케줄러에서 발생되는 과다한 필요 버퍼 량은 우선 순위 큐들이 회전함으로서 모든 우선 순위 큐들이 동일한 크기의 버퍼 량을 필요로 하기 때문에 발생된다. 본 논문에서는 회전 간격을 좁게 하면서 필요 버퍼 량을 줄이기 위한 방안으로, RPQ 스케줄러와 같이 회전 우선 순위 큐들로 구성되면서 동일한 방법으로 회전하는 "우선 순위 큐 계층"이라는 것을 하나 이상 구성한다. 한 계층의 구성 요소인 우선 순위 큐의 개수를 '$r_{k,z}$'라고 하고 계층의 개수를 '$R_{k,z}$'이라고 한다. 만일 회전 간격이 RPQ 스케줄러와 동일하다면, 전체 우선 순위 큐의 개수 ($R_{k,z} * r_{k,z}$)는 RPQ 스케줄러에 존재하는 우선 순위 큐의 개수와 동일하다. 이와 같이 여러 계층으로 RPQ 스케줄러를 구성한 방법론이 본 논문에서 제시하는 MRPQ(Multiple Rotating Priority Queue) 스케줄러이다. MRPQ 스케줄러의 자세한 구성 방법은 (과정 2)와 같다.

(과정 2) Multiple Rotating Priority Queue 스케줄러 구성 방법

㉮ (과정 1)로 결정된 $|P_{k,z}|$ 만큼의 우선 순위 색인을 만든다.

㉯ 우선 순위 색인들은 우선 순위 집합 $H_{k,z}$ 중 한 개의 우선 순위를 선택하여 우선 순위 색인 값으로 한다.

㉰ 연속된 우선 순위 값을 갖는 $r_{k,z}$개의 우선 순위 색인들을 선택하여 단일 우선 순위 색인 집합 $G_{k,z,n}$을 구성한다.

(조건 1) $r_{k,z}$와 $R_{k,z}$는 스케줄러 설계자에 의해서 임의적으로 결정할 수 있으나, "$|P_{k,z}| = R_{k,z} * r_{k,z}$"을 만족해야 한다.

㉱ ㉰를 $R_{k,z}$번 반복하여 $R_{k,z}$개만큼의 $G_{k,z,n}$을 구성한다.

(조건 1) 한번 선택된 우선 순위 색인은 다시 선택될 수 없다.

㉲ $G_{k,z,n}$에 포함된 $r_{k,z}$개의 우선 순위 색인과 $r_{k,z}$개의 큐를 이용하여 한 개의 회전 우선 순위 큐 계층을 구성한다.

(조건 1) 우선 순위 색인과 큐는 일대일로 연결된다.

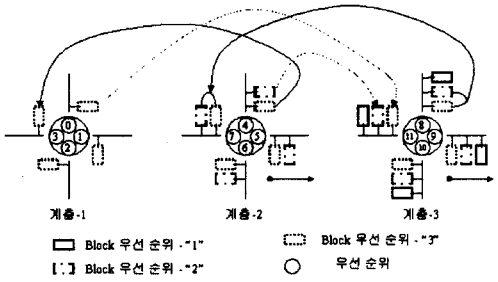
㉳ ㉲를 $R_{k,z}$번 반복하여 $R_{k,z}$개의 회전 우선 순위 큐 계층을 만든다.

한 계층의 구성 요소인 큐는 RPQ 스케줄러의 $r_{k,z}$-1번째 우선 순위 큐에서 요구되는 버퍼 량과 동일한 크기의 LIFO(Last In First Out) 큐인 블록 큐(block queue)의 조합으로 구현된다. 그리고 한 개의 큐를 구성하는 블록 큐들간의 관계는 LIFO이고 각 블록 큐는 전송되는 순서로 블록 우선 순위를 할당받는다. 예를 들어 하나의 큐가 두 개의 블록 큐 A, B의 조합으로 구성되고 블록 큐 B에 존재하는 패킷을 블록 큐 A에 존재하는 패킷보다 먼저 전송할 경우, 블록 큐 B의 블록 우선 순위가 블록 큐 A의 블록 우선 순위보다 높다. 단, 최하위 블록 우선 순위는 $R_{k,z}$와 같다. 각 계층별로 큐를 구성하는 블록 큐의 수는 다르다. '0'부터 $r_{k,z}$-1까지의 값을 갖는 우선 순위 색인들로 구성된 첫 번째 계층에서는 각각의 큐들이 한 개의 블록 큐로 구현되고, $r_{k,z}$부터 $2 * r_{k,z}$-1까지의 값을 갖는 우선 순위 색인들로 구성된 두 번째 계층에서는 두 개의 블록 큐로 구현된다.

그리고 $(L-1) * r_{k,z}$부터 $L * r_{k,z} - 1$까지의 값을 갖는 우선 순위 색인들로 구성된 L번째 계층에서는 L개의 블록 큐로 구현된다. 본 논문에서는 낮은 우선 순위 색인 값을 갖는 계층일수록 상위 계층이라고 한다. 그리고 $P_{k,z,j}$를 값으로 하는 우선 순위 색인과 연결된 큐를 "$P_{k,z,j}$ 큐"라고 한다.

이렇게 구성된 MRPQ 스케줄러는 회전 동작을 수행할 때마다 서로 다른 계층에 존재하면서 근접한 우선 순위를 갖는 큐들간에 패킷 이동이 발생된다. 예를 들어 (그림 2)와 같이 MRPQ 스케줄러를 구성하였을 경우, 두 번째 계층에 존재하면서 우선 순위 '4'인 큐에 저장된 패킷을 첫 번째 계층에 존재하는 우선 순위 '3'

인 큐로 이동한다. 그러나 큐의 구성을 LIFO 관계로 연결된 블록 큐의 조합으로 구현하였기 때문에 이동하고자 하는 패킷은 항상 일정한 블록 큐에 존재하게 된다. 그러므로 스케줄러는 계층간을 연결하기 위해서 미리 정해진 블록 큐들만을 이동시키면 된다. 각 계층별로 이동하게 되는 블록 큐는 다음과 같다. L번째 계층에 포함된 $r_{k,z}*(L-1)$ 우선 순위 큐에서 L-1 번째 계층에 포함된 $r_{k,z}*(L-1)-1$ 우선 순위 큐로 패킷을 전달하기 위해서는 L-1개의 낮은 블록 우선 순위를 갖는 블록 큐를 이동하면 된다. 그리고 최하위 계층을 제외한 모든 계층에서는 회전 간격마다, 상위 계층으로 이동하지 않은 블록 큐를 최하위 계층에 존재하는 최하위 우선 순위 큐로 반환한다.



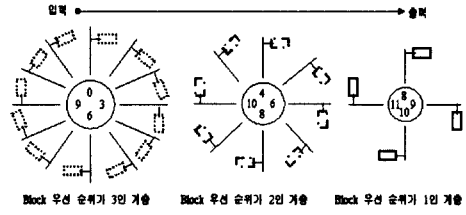
(그림 2) MRPQ 스케줄러의 구성

4.2 MRPQ 스케줄러 구현

앞에서 제시한 MRPQ 스케줄러 구성 방법론에서는 각 계층간에 패킷을 전달하기 위해서 블록 큐를 이동하는 작업을 수행하게된다. 이러한 작업은 각 계층간에 자료 복사 연산을 유발시키므로 스케줄링 지연을 가중시킨다. 본 논문에서는 계층간에 이동하게 되는 블록 큐들이 일정하다는 것에 착안하여 자료 복사 연산을 수행하지 않고 스케줄링을 하는 방법을 다음과 같이 제시한다.

동일 블록 우선 순위를 갖는 블록 큐들로 구성된 새로운 계층을 구성한다. 각 계층을 구성하는 블록 큐들은 우선 순위 색인과 일대일로 연결된다. 이때 우선 순위 색인 값은 블록 큐들이 소속되었던 큐의 우선 순위 색인 값과 동일하다.

각 계층에서 우선 순위 색인들의 색인 값은 다음과 같다. 블록 우선 순위가 '1'인 $r_{k,z}$개의 블록 큐들로 구성된 계층에서는 우선 순위 색인들이 $|P_{k,z}|-r_{k,z}$에서 $|P_{k,z}|-1$까지의 우선 순위 색인 값을 갖으며, 블록 우선 순위가 '2'인 $2*r_{k,z}$개의 블록 큐들로 구성된 계층에서는 우선 순위 색인들이 $|P_{k,z}|-r_{k,z}*2$에서 $|P_{k,z}|-1$까지의 우선 순위 색인 값을 갖는다. 그리고 블록 우선 순위가 m인 $m*r_{k,z}$개의 블록 큐들로 구성된 계층에서는 우선 순위 색인들이 $|P_{k,z}|-r_{k,z}*m$에서 $|P_{k,z}|-1$까지의 우선 순위 색인 값을 갖는다. 블록 우선 순위가 높은 블록 큐들로 구성된 계층을 상위 계층이라 하고, 블록 우선 순위가 낮은 블록 큐들로 구성된 계층을 하위 계층이라 한다. 이렇게 구성된 계층의 수는 이전에 존재했던 계층의 수와 동일하다. (그림 3)은 (그림 2)를 위와 같은 방법으로 구현한 것이다.



(그림 3) MRPQ 스케줄러 구현

4.3 MRPQ 스케줄링 과정과 예제

MRPQ 스케줄러의 스케줄링 과정은 다음과 같다. 실시간 통신을 원하는 연결들은 (과정 1)과 동일한 방법으로 우선 순위를 할당받는다. 그리고 연결에 소속된 패킷이 스케줄러에 입력될 경우, 그 패킷은 연결과 동일한 우선 순위를 갖는 블록 큐에 LIFO 순서로 입력된다. 단, 해당 블록 큐가 여러 계층에 존재한다면 하위계층에 입력된다. 그리고 MRPQ 스케줄러는 회전 간격마다 모든 우선 순위 색인 값을 정수 '1'씩 감소시킨다. 패킷을 전송하고자 할 때에는 가장 높은 우선 순위 색인 값을 가지면서 동시에 가장 높은 블록 우선 순위를 갖는 블록 큐를 선택한 후, 그 블록 큐에 저장된 패킷을 LIFO 순서대로 전송한다.

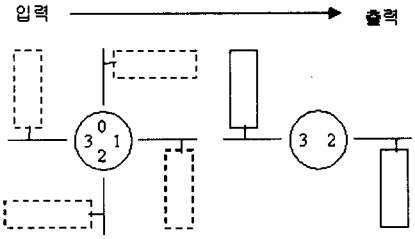
MRPQ 스케줄러에 입력된 패킷의 주어진 지연한계 $d_{k,z,i}$를 보장하기 위해서는 우선 순위 $p_{k,z,i}$를 갖는 패킷의 최대 지연시간 $D_{k,z,i}$이 $d_{k,z,i}$보다 작거나 같아야 한다. $D_{k,z,i}$는 (정리 2)와 같이 예견할 수 있으며, 이는 RPQ 스케줄러와 동일하다. (정리 2)는 다음과 같이 증명할 수 있다. (과정 1)에서와 같이 우선 순위 $p_{k,z,i}$에 대한 우선 순위 할당 범위는 $p_{k,z,i}*\Delta+\Delta$보다는 작거나 같고, 가장 높은 우선 순위 0에 대한 우선 순위 할당 범위는 0보다 크거나 같다. 그리고 우선 순위 할당

범위의 간격이 회전 간격 Δ와 같으므로 우선 순위 $p_{k,z,i}$를 갖는 패킷이 가장 높은 우선 순위를 갖는 큐를 지나 가장 낮은 우선 순위를 갖는 큐에 저장되기 직전까지 걸리는 시간은 최대 $p_{k,z,i}*\Delta+\Delta-0$가 된다. 그러므로 MRPQ 스케줄러 $S_{k,z}$가 보장할 수 있는 최대 지연 시간은 (정리 2)와 같다.

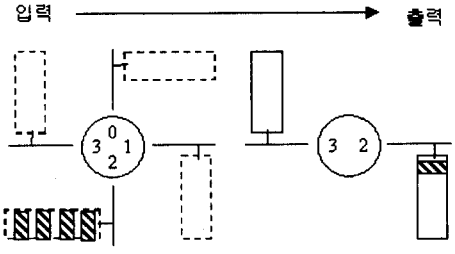
(정리 2) 패킷의 최대 지연시간은 식 (2)와 같다

$$D_{k,z,i}=(p_{k,z,i}+1)*\Delta \qquad (2)$$

(그림 4)는 MRPQ 스케줄러를 다음과 같은 가정에서 수행한 것이다. 스케줄러에 입력된 모든 패킷의 크기는 일정하고, 회전 간격 Δ동안 최대로 전송할 수 있는 데이터 량은 두 패킷의 크기와 같다. (그림 4)에서 (상태 1)은 스케줄러의 초기 상태를 나타낸다. (상태 2)는 초기 상태부터 Δ시각까지 우선 순위가 '2'인 다섯 개의 패킷이 입력된 상태이고, (상태 3)은 초기 상태부터 Δ시각까지 우선 순위가 2인 두개의 패킷이 LIFO 순서로 전송된 상태이다. 그리고 (상태 4)는 Δ



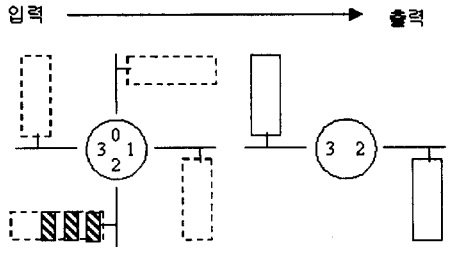
(상태 1) 초기상태



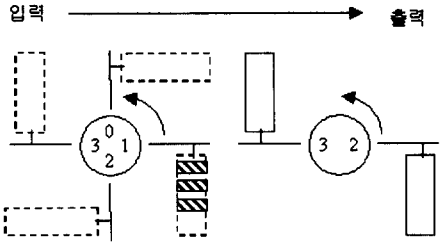
(상태 2) 0부터 △사이에 입력된 상태

새로 입력된 패킷

(그림 4.1) MRPQ 스케줄링 과정

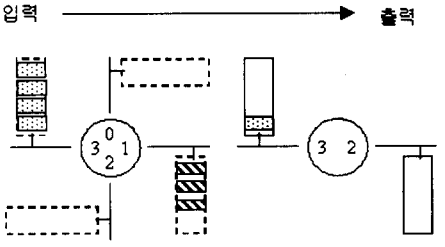


(상태 3) 0부터 △사이에 출력된 상태



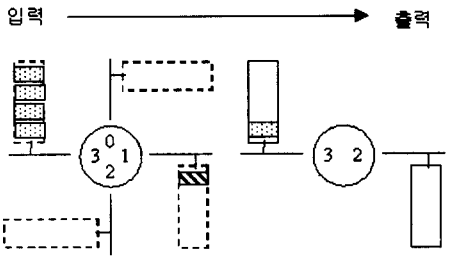
(상태 4) △시각에 회전한 상태

(그림 4.2) MRPQ 스케줄링 과정



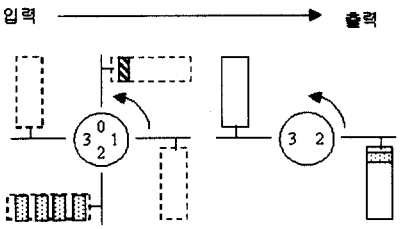
(상태 5) △부터 2△사이에 입력된 상태

새로 입력된 패킷

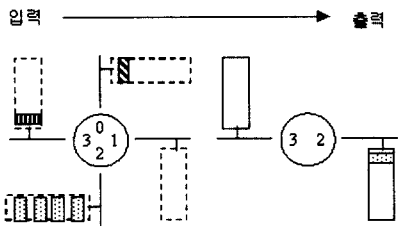


(상태 6) △부터 2△사이에 출력된 상태

(그림 4.3) MRPQ 스케줄링 과정



(상태 7) 2Δ시각에 회전한 상태



(상태 8) 2Δ부터 3Δ사이에 입력된 상태

새로 입력된 패킷

(그림 4.4) MRPQ 스케줄링 과정

시각에 모든 계층을 회전한 것이다. 이때 회전 동작은 모든 우선 순위 색인의 값을 감소시킴으로서 이루어진다. (상태 5)는 Δ시각부터 2Δ시각까지 우선 순위가 '3'인 5개의 패킷이 입력된 상태를 나타내고, (상태 6)는 Δ시각부터 2Δ시각까지 우선 순위가 '1'인 2개의 패킷이 전송된 상태를 나타낸다. (상태 7)은 (상태 4)와 같이 모든 계층을 회전한 것이다. (상태 8)은 2Δ시각부터 3Δ시각까지 우선 순위가 '3'인 한 개의 패킷이 입력된 상태이다.

5. MRPQ 스케줄러의 스케줄링 가능도

응용프로그램들이 실시간 통신을 수행하기 위해서 새로운 연결을 설정하고자할 때, 해당 연결이 미리 설정된 다른 연결들에게 영향을 주지 않으면서 실시간 제약사항을 만족시킬 수 있는지를 검증하고 해당 연결에 네트워크 자원을 할당해야한다. 이러한 작업은 CAC (Connection AdmissionControl) 정책에 의해서 수행된다. 네트워크 자원을 효율적으로 사용하면서 보다 많은 연결들을 설정하기 위해서, CAC 정책은 정확한 스케줄링 가능도를 이용하여 모든 연결들의 실시간 제약사항 만족여부를 검증해야 된다. 본 장에서는 J. Liebeherr, D. E. Wrege, 그리고 D. Ferrari가 제시한 스케줄링 가능도의 필요충분 조건 유도 방식[5]을 이용하여 MRPQ 스케줄링 가능도의 충분 조건을 유도한다.

5.1 트래픽의 특성화(Traffic Characterization)

본 절에서는 MRPQ 스케줄링 가능도의 충분조건을 유도하기 위해서 트래픽을 특성화한다. t시각에 스케줄러 $S_{k,z}$에 도착하는 연결 M_i에서의 트래픽을 $a_{k,z,i}(t)$라고 한다. 연결 M_i에서의 트래픽은 최대 전송시간 $s^{max}_{k,z,i}$와 최소 전송시간 $s^{min}_{k,z,i}$을 갖는 패킷들로 구성된다. 그리고 식 (3)을 사용하여 일정 시간 간격 $[t, t+\tau]$동안 연결 Mi로부터 도착한 트래픽을 나타낸다. 연결 Mi로부터 스케줄러에 도착하는 최대 트래픽은 [2, 3, 5]에서와 같이 트래픽 제약 함수 $A^*_{k,z,i}$로 한정된다고 가정한다.

$$A_{k,z,i}[t, t+\tau] = \int_t^{t+\tau} a_{k,z,i}(t)dt \quad (3)$$

$$A_{k,z,i}[t, t+\tau] \le A^*_{k,z,i}(\tau) \quad (4)$$

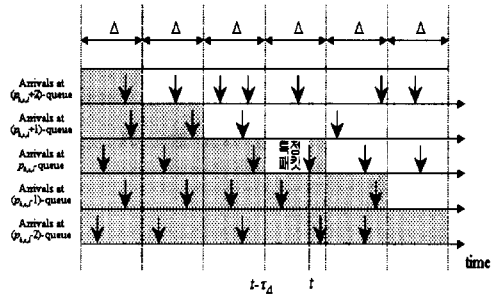
where $A^*_{k,z,i} = 0$ for all $t < 0$ and

$A^*_{k,z,i}(t) \ge 0$ for $t \ge 0$

본 논문에서는 트래픽 제약 함수 $A^*_{k,z,i}$를 보장하기 해서 traffic policer 메커니즘(mechnim)과 rate controller 메커니즘이 사용되는 것으로 가정한다. traffic policer 메커니즘은 연결 M_i의 트래픽이 $A^*_{k,z,i}$를 따르지 않을 경우, 그 트래픽을 거부(reject)하는 메커니즘이고, rate controller 메커니즘은 연결 M_i의 트래픽이 $A^*_{k,z,i}$를 따르게 하기 위해서, 연결 M_i에서 발생되는 패킷들을 임시적으로 버퍼에 저장하는 메커니즘이다[2, 6, 7]. 트래픽 제약 함수는 최악(worst case) 트래픽을 특성화하는 (σ, ρ)-모델[2, 3]과 (x_{min}, x_{ave}, I, s)-모델[4] 그리고 전송속도(rate)를 조절하는 leaky bucket 메커니즘[8, 9] 등으로 유도할 수 있다.

본 논문에서는 MRPQ 스케줄링 가능도의 충분 조건을 유도하기 위해서, 특정 시각 t에 스케줄러에 저장된 패킷들의 량을 계산한다. 그리고 이 것을 스케줄러의 작업부하 'W(t)'라고 하고, 연결 M_i에서 발생되어 t 시각에 스케줄러 $S_{k,z}$에 남아 있는 패킷들의 량을 '$W_{k,z,i}(t)$'라고 한다. (그림 9)는 $6*\Delta$ 시간동안 우선 순

위 $p_{k,z,i}+2$, $p_{k,z,i}+1$, $p_{k,z,i}$, $p_{k,z,i}-1$, $p_{k,z,i}-2$를 갖는 연결들에서 발생된 패킷들이 시각별로 각 우선 순위와 연결된 LIFO 큐에 입력된 상태를 나타낸다. 만일 특정 패킷이 t 시각에 스케줄러에 입력되었을 경우, t 시각 전에 수행한 회전 작업 중에 가장 최근에 발생한 회전 작업의 수행 시각은 $t-\tau\Delta$이라고 한다. 그리고 (그림 5)에서 빗금으로 표시된 부분은 우선 순위가 $p_{k,z,i}$인 연결 M_i에서 발생한 패킷들 중에 t 시각에 MRPQ 스케줄러에 입력된 특정 패킷과 그 패킷보다 우선적으로 전송되는 패킷들을 나타낸다.



(그림 5) MRPQ 스케줄러의 작업부하

특정 패킷보다 우선적으로 전송되는 패킷들이 스케줄러에 존재하지 않는 시각을 $t-\lambda$이라고 할 때, λ는 식 (7)과 같이 유도할 수 있다. 식 (7)에 대한 증명은 다음과 같다. t-y 시각 이전에 스케줄러에 입력되어 특정 패킷보다 우선적으로 전송되는 패킷들이 있다고 가정할 경우, 그 패킷들은 두 부류로 나누어 질 수 있다. 한 경우는 특정 패킷보다 높거나 같은 우선 순위(q)를 갖으면서 [0, t-y] 사이에 스케줄러에 입력되어 특정 패킷들보다 먼저 전송되는 패킷들과 특정 패킷보다 낮은 우선 순위(q)를 갖으면서 $[0, \min\{t-y, (t-\tau_\Delta)+(p_{k,z,i}-q)\Delta\}]$ 사이에 스케줄러에 입력되어 특정 패킷들보다 먼저 전송되는 패킷들이다. 하지만 두 경우를 모두 고려한 최악의 경우는 $[0, \min\{t-y, (t-\tau_\Delta)+(p_{k,z,i}-q)\Delta\}]$ 사이에 스케줄러에 입력된 패킷들이 된다. 그러므로 λ는 $[0, \min\{t-y, (t-\tau_\Delta)+(p_{k,z,i}-q)\Delta\}]$ 사이에 입력된 모든 패킷들의 작업부하가 '0'이 되는 y가 된다.

$$\lambda = \min\left\{ y \mid y \geq 0,\ \sum_{q=1}^{|P_{k,z}|} \sum_{M_u \in C_q} W_{k,z,u} \left(\min\{t-y, (t-\tau_\Delta)+(p_{k,z,i}-q)\Delta\}\right) = 0 \right\} \quad (7)$$

특정 시각 t에 우선 순위 $p_{k,z,i}$를 가지고 스케줄러에 도착한 특정 패킷의 전송완료 시각을 '$t+\delta$'라고 할 때, $t+\tau\ (0 \leq \tau \leq \delta)$ 시각 이전에 특정 패킷보다 우선적으로 전송될 수 있는 패킷들은 아래와 같이 유도할 수 있다.

1) 특정 패킷이 MRPQ 스케줄러에 입력될 때 저장되는 큐의 우선 순위 $p_{k,z,i}$보다 낮은 우선 순위 ($p_{k,z,i} < q$)를 갖는 큐에 $t-\lambda$ 시각부터 $(t-\tau\Delta)-(q-p_{k,z,i})*\Delta$ 시각 전에 임의의 패킷들이 입력될 경우, Δ시간마다 발생되는 큐들의 회전효과에 의해서 $t-\tau\Delta$ 시각 이후에는 임의의 패킷들이 저장된 큐의 우선 순위가 특정 패킷이 저장된 큐의 우선 순위보다 항상 높아지기 때문에 그 큐에 저장된 패킷들은 특정 패킷보다 우선적으로 전송된다. 식 (8)은 이와 같은 이유로 우선 순위 $p_{k,z,i}$를 갖는 특정 패킷보다 낮은 우선 순위를 갖으면서 우선적으로 전송되는 패킷의 량을 나타낸다.

$$\sum_{q=p_{k,z,i}+1}^{|P_{k,z}|} \sum_{M_u \in C_q} A_{k,z,u}[t-\lambda, (t-\tau_\Delta)+(p_{k,z,i}-q)*\Delta] \quad (8)$$

2) 특정 패킷과 같이 우선 순위 $p_{k,z,i}$를 갖는 큐에 입력되는 임의의 패킷이 $t-\lambda$ 시각부터 $t-\tau\Delta$ 시각 전에 입력될 경우, Δ시간마다 발생되는 큐들의 회전효과에 의해서 $t-\tau\Delta$ 시각이후에는 임의의 패킷들이 저장된 큐의 우선 순위가 특정 패킷이 저장된 큐의 우선 순위보다 높아지게 되므로 그 패킷들은 특정 패킷보다 우선적으로 전송되게 된다. 그리고 t 시각부터 $t-\tau\Delta+\Delta$ 시각 전까지 특정 패킷과 동일한 큐에 입력된 패킷은 큐가 LIFO 관계로 구성되었기 때문에 특정 패킷보다 우선적으로 전송된다. 그러므로 특정 패킷과 동일한 우선 순위를 갖는 큐에 입력되지만 특정 패킷보다 우선적으로 전송되는 패킷의 량은 식 (9)와 같다. 단, 식 (10)은 특정 패킷의 데이터 량을 포함한다.

$$\sum_{M_u \in C_{p_{k,z,i}}} \{A_{k,z,u}[t-\lambda, t-\tau_\Delta] + A_{k,z,u}[t, t-\tau_\Delta+\Delta]\} \quad (9)$$

3) 특정 패킷이 입력되는 큐 보다 높은 우선 순위 ($p_{k,z,i} > q$)를 갖는 큐에 입력되는 임의의 패킷들이 $t-\lambda$ 시각부터 $t-\tau\Delta+\Delta*(p_{k,z,i}-q)$ 시각 전에 입력될 경우, 임의의 패킷들이 저장된 큐의 우선 순위가 특정 패킷이 저장된 큐의 우선 순위보다 항상 높으므로 그 큐에 저장된 패킷들은 특정 패킷보다 먼저 전송된다. 그리고 $t-\tau\Delta+\Delta*(p_{k,z,i}-q)$ 시각부터 $t-\tau\Delta+(p_{k,z,i}$

$-q+1)*\Delta$ 시각 전에 입력된 경우, 특정 패킷과 동일한 큐에 저장되지만 특정 패킷보다 나중에 입력되어 우선적으로 전송된다. 그러므로 특정 패킷이 입력되는 큐보다 높은 우선 순위를 갖는 큐에 입력되면서 특정 패킷보다 우선적으로 전송되는 패킷 량은 식 (10)과 같다.

$$\sum_{q=1}^{p_{k,z,i}-1}\sum_{M_u\in C_q} A_{k,z,u}[t-\lambda, \min\{t+\tau, (t-\tau_\Delta)+(p_{k,z,i}-q+1)*\Delta\}] \quad (10)$$

4) $t-\lambda$ 시각 이전에 스케줄러에 입력된 후, 특정패킷이 저장되는 큐보다 같거나 낮은 우선 순위를 갖는 큐에 $t-\tau\Delta$ 시각 이 후 저장되는 패킷 ($d \geq \lambda + p_{k,z,i} * \Delta$), 즉 식 (8), 식 (9), 식 (10)에서 고려하지 않은 패킷이 $t-\lambda$ 시각에 전송 중에 있다면 비선점(non-preemptive) 스케줄러인 MRPQ 스케줄러는 특정 패킷을 포함하여 식 (8), 식 (9), 식 (10)에 포함되는 패킷들의 전송을 그 패킷의 전송이 완료될 때까지 대기 시켜야 한다. 이와 같이 스케줄러의 비선점 특성에 의해서 특정 패킷보다 우선적으로 전송되는 패킷의 량 $R(t-\lambda)$은 식 (11)과 같다.

$$R(t-\lambda) \leq \max_{M_u, d_u \geq \lambda + p_{k,z,i}*\Delta} s_{M_u}^{\max} \quad (11)$$

$s_{M_u}^{\max}$: 연결 Mu에 속한 패킷들 중에 전송시간이 가장 긴 패킷의 전송시간

그러므로 $t+\tau$ 시각에 MRPQ 스케줄러에 저장되어 있는 패킷들 중에 우선 순위 $p_{k,z,i}$를 가지고 t 시각에 스케줄러에 입력된 특정 패킷보다 우선적으로 전송되는 패킷들의 량 $W^{p_{k,z,i},t}(t+\tau)$은 식 (8), 식 (9), 식 (10), 식 (11)을 이용하여 식 (12)와 같이 유도할 수 있다.

$$\begin{aligned} &W^{p_{k,z,i},t}(t+\tau) = \\ &\sum_{q=p_{k,z,i}+1}^{|P_{k,z}|}\sum_{M_u\in C_q} A_{k,z,u}[t-\lambda, (t-\tau_\Delta)+(p_{k,z,i}-q)*\Delta] \\ &+ \sum_{M_u\in C_{p_{k,z,i}}} \{A_{k,z,u}[t-\lambda, t-\tau_\Delta] + A_{k,z,u}[t, t-\tau_\Delta+\Delta]\} \\ &+ \sum_{q=1}^{p_{k,z,i}-1}\sum_{M_u\in C_q} A_{k,z,u}[t-\lambda, \min\{t+\tau, (t-\tau_\Delta) \\ &+ (p_{k,z,i}-q+1)*\Delta\}] + R(t-\lambda) - (\lambda+\tau) \end{aligned} \quad (12)$$

특정 패킷을 전송하는데 걸리는 시간이 's'이라면, 특정 패킷을 전송하기 시작한 시각은 $t+\delta-s$ 시각이 된다. 그러므로 δ는 식 (13)과 같이 결정할 수 있다.

$$\begin{aligned} \delta &= s + \min\{z \mid W^{p_{k,z,i},t}(t+z) = s, z \geq 0\} \\ &\leq \min\{z \mid W^{p_{k,z,i},t}(t+z) = 0, z \geq 0\} \end{aligned} \quad (13)$$

s : 특정 패킷의 전송경과 시간 ($s \leq s_{M_i}^{\max}$)

5.3 스케줄링 가능도의 충분 조건

특정 패킷이 주어진 지연한계를 지키기 위해서는 "$W^{p_{k,z,i},t}(t+\delta)=0$"을 만족시키는 δ가 "$\delta \leq p_{k,z,i}*\Delta$"에 존재해야 된다. 그러므로 "$W^{p_{k,z,i},t}(t+p_{k,z,i}*\Delta) \leq 0$"이 만족된다면 특정 패킷은 항상 주어진 지연한계를 만족한다. 식 (12)와 트래픽 제약 함수를 이용하여 $W^{p_{k,z,i},t}(t+p_{k,z,i}*\Delta)$의 upper bound를 유도하면 식 (15)와 같다.

$$\begin{aligned} &W^{p_{k,z,i},t}(t+p_{k,z,i}*\Delta) \leq \\ &\sum_{M_u\in C_1} A^*_{k,z,u}(\lambda+(p_{k,z,i}-1+1)*\Delta) \\ &+ \sum_{q=2}^{|P_{k,z}|}\sum_{M_u\in C_q} A^*_{k,z,u}(\lambda+(p_{k,z,i}-q+1)*\Delta) \\ &+ R(t-\lambda) - (\lambda+p_{k,z,i}*\Delta) \end{aligned} \quad (14)$$

$$\begin{aligned} \leq\ &\sum_{M_u\in C_1} A^*_{k,z,u}(\lambda+p_{k,z,i}*\Delta) \\ &+ \sum_{q=2}^{|P_{k,z}|}\sum_{M_u\in C_q} A^*_{k,z,u}(\lambda+(p_{k,z,i}-q+1)*\Delta) \\ &+ \max_{M_u, d_u \geq \lambda+p_{k,z,i}*\Delta} S_{M_u}^{\max} - (\lambda+p_{k,z,i}*\Delta) \end{aligned} \quad (15)$$

또한 식 (15)보다 $W^{p_{k,z,i},t}(t+p_{k,z,i}*\Delta)$는 작거나 같다. 그러므로 식 (16)과 같이 유도할 수 있다. 그리고 특정 패킷뿐만 아니라 모든 패킷을 적용시키기 위해서, $\lambda + p_{k,z,i} * \Delta$를 t로 치환하면 식 (16)은 식 (17)과 같이 변환할 수 있다. 그러므로 MRPQ 스케줄러에 입력되는 패킷 량이 식 (18)을 만족시킨다면 모든 패킷들은 그들의 주어진 지연한계를 만족시킨다.

$$\begin{aligned} &\sum_{M_u\in C_1} A^*_{k,z,u}(\lambda+p_{k,z,i}*\Delta) \\ &+ \sum_{q=2}^{|P_{k,z}|}\sum_{M_u\in C_q} A^*_{k,z,u}(\lambda+(p_{k,z,i}-q+1)*\Delta) \\ &+ \max_{M_u, d_u \geq \lambda+p_{k,z,i}*\Delta} S_{M_u}^{\max} - (\lambda+p_{k,z,i}*\Delta) \\ &\sum_{M_u\in C_1} A^*_{k,z,u}(t) \\ &+ \sum_{q=2}^{|P_{k,z}|}\sum_{M_u\in C_q} A^*_{k,z,u}(t-q*\Delta+\Delta) \\ &+ \max_{M_u, d_u \geq t} S_{M_u}^{\max} - t \leq 0 \end{aligned} \quad (17)$$

(정리 3) 식 (18)이 만족된다면 스케줄러 $S_{k,z}$에 입력

된 모든 패킷들은 그들의 주어진 지연한계를 항상 만족시킨다.

$$t \le \sum_{M_u \in C_1} A^*_{k,z,u}(t) + \sum_{q=2}^{|P_{k,z}|} \sum_{M_u \in C_q} A^*_{k,z,u}(t - q*\Delta + \Delta) + \max_{M_u,\, d_u \ge t} S^{\max}_{M_u} \quad (18)$$

6. MRPQ 스케줄러의 성능 평가

우선 순위 $p_{k,z,i}$를 갖는 패킷의 최대 지연시간 $D_{k,z,i}$ 동안 최대로 전송할 수 있는 데이터 량의 크기와 $p_{k,z,i}$와 동일한 $P_{k,z,j}$를 갖는 블록 큐들의 전체 크기가 동일하므로 스케줄러에서 필요한 전체 버퍼 량은 각 계층에 존재하는 블록 큐들의 전체 크기를 계층별로 더하여 구할 수 있다. 각 계층에 존재하는 블록 큐들의 전체 크기는 각 계층의 우선 순위 수에 블록 큐의 크기를 곱한 만큼의 버퍼 량이며, 각 계층의 우선 순위 수는 최상위 계층에서의 우선 순위 개수 '$v_{k,z}$'에 블록 우선 순위를 곱한 수와 같다. (정리 4)는 위와 같은 방법으로 일반화된 필요 버퍼 량 계산 방법을 유도한 것이다.

(정리 4) MRPQ 스케줄러에서 필요한 버퍼 량은 식 (19)와 같다.

한 블록 큐의 크기 : $\Delta * v_{k,z}$ 동안 최대로 전송할 수 있는 량 = $\Gamma_\Delta * v_{k,z}$

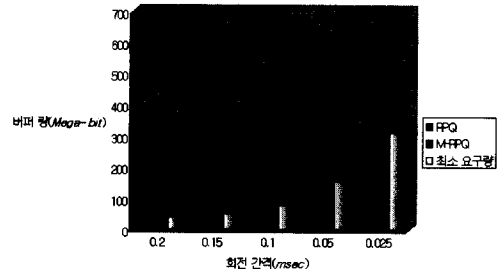
필요 버퍼 량 = 한 블록 큐의 크기 * 블록 큐의 전체 개수

$$Q_{k,z} = \Gamma_\Delta * R_{k,z} * (v_{k,z})^2 (R_{k,z}+1)/2$$
$$v_{k,z} = r_{k,z} \quad (19)$$
$$(R_{k,z} = \lceil |P_{k,z}|/v_{k,z} \rceil)$$

RPQ 스케줄러에서는 입력된 모든 패킷들의 주어진 지연한계를 보장하기 위해서, 우선 순위가 가장 높은 큐에 저장된 패킷들을 회전 시각 직전까지 모두 전송해야 되는 반면에 MRPQ 스케줄러에서는 각 계층에서 우선 순위가 가장 높은 블록 큐에 저장된 패킷들을 회전 시각 직전까지 모두 전송해야 한다. 그러나 각 계층에서 가장 높은 우선 순위를 갖는 패킷들이 다른 계층에 존재하는 보다 높은 우선 순위를 갖는 패킷들에 의해 전송하지 못한다면, RPQ 스케줄러도 위와 같은 제약 조건을 만족시키지 못한다. 이와 같은 특성은 동일 우선 순위를 갖는 블록 큐들이 서로 LIFO 관계로 입·출력되고, 우선 순위 $P_{k,z,j}$를 갖는 모든 블록 큐들이 요구하는 전체 버퍼 량이 RPQ 스케줄러에서 우선 순위 $P_{k,z,j}$를 갖는 큐에서 요구하는 버퍼 량과 동일하기 때문이다. 이때 두 스케줄러에서 요구하는 버퍼 량은 $(P_{k,z,j}+1)*\Delta$ 동안 최대로 전송할 수 있는 데이터 량이다.

(그림 6)은 $v_{k,z}$ 값을 '2'로 하였고 링크의 대역폭 B를 155Mbps, 최대 지연한계 $MAX_{k,z}$를 10msec, 회전 간격 Δ을 0.2msec에서 0.025msec까지 줄인 조건하에서 MRPQ 스케줄러에서의 필요 버퍼 량과 RPQ 스케줄러에서의 필요 버퍼 량 그리고 RPQ 스케줄러에서 큐의 회전을 고려하지 않았을 때, 요구되는 버퍼 량 즉, 정적 우선 순위 스케줄러에서 요구하는 버퍼 량을 비교한 것이다.

(그림 6)에서와 같이 MRPQ 스케줄러에서 필요한 버퍼 량은 RPQ 스케줄러에서 필요로 하는 버퍼 량의 절반에 가까운 량에 불과하다. 또한 이 크기는 우선 순위 큐의 회전을 고려하지 않은 요구 버퍼 량과 유사하다.



(그림 6) RPQ 스케줄러와 MRPQ 스케줄러의 필요 버퍼 량 비교

이러한 특성은 MRPQ 스케줄러가 RPQ 스케줄러에서 필요로 하는 버퍼 량만으로 회전 간격을 절반정도로 줄일 수 있다는 것을 나타낸다. 만일 MRPQ 스케줄러와 RPQ 스케줄러의 회전 간격이 동일하다면, MRPQ 스케줄러에 입력된 패킷들과 RPQ 스케줄러에 입력된 패킷들은 동일한 최대 지연시간을 보장받는다. 이러한 특성은 MRPQ 스케줄러와 RPQ 스케줄러의 우선 순위 할당 기준과 전체 우선 순위의 개수가 동일하고 두 스케줄러가 회전 간격마다 큐의 우선 순위를 높

이게 되므로 해당 큐가 최하위 우선 순위를 갖게 되는 시간이 동일하기 때문이다.

7. 결론 및 차후 연구 과제

MRPQ 스케줄러는 RPQ 스케줄러와 같이 우선 순위 색인 값만을 감소시킴으로서 EDF 스케줄러에서 수행되는 정렬이나 검색 연산 작업과 유사한 동작을 수행하게 된다. 그러나 RPQ 스케줄러에서는 큐와 연결된 우선 순위 색인 값을 변화시킴으로서 최하위 우선 순위 큐에서 필요로 하는 버퍼 량을 모든 큐에서도 필요로 하게 되는 반면, MRPQ 스케줄러에서는 우선 순위 큐를 여러 계층으로 분할된 블록 큐의 조합으로 구성함으로서 필요 버퍼 량을 최대로 줄일 수 있다. 이러한 결과, MRPQ 스케줄러는 RPQ 스케줄러에서 필요로 하는 버퍼 량의 절반에 가까운 버퍼 량만으로 동일한 회전 효과를 얻을 수 있다.

연결을 기반으로 실시간 통신을 하고자 할 때에는 CAC(Connection Admission Control) 알고리즘을 통하여 최대 지연 한계와 같은 실시간 제약 사항의 만족여부를 검증 받은 후, 해당 연결에 자원을 할당하여 연결을 설정하게 된다. 이러한 CAC 알고리즘에서 실시간 제약 사항의 검증은 특정 스케줄러의 스케줄링 가능도를 기반으로 수행된다. 본 논문에서는 MRPQ 스케줄러의 스케줄링 가능도를 필요조건 식으로 유도하였다. 그러나 실시간 통신을 요구하는 보다 많은 연결을 설정하고 링크의 대역폭과 같은 네트워크 자원을 보다 효율적으로 사용하기 위해서는 스케줄링 가능도의 필요충분조건이 요구된다.

참 고 문 헌

[1] Chengzhi Li, Riccardo Bettati, Wei Zhao. "Static Priority Scheduling for ATM Networks," In Proc. of the Real-Time Systems Symposium, San Francisco, CA, December 1997.

[2] R. L. Cruz. "A calculus for network delay, part I, part II," IEEE Trans. on Information Theory, January 1991.

[3] Chengzhi Li, A. Raha and Wei Zhao. "Stability in ATM Networks," In Proc. of the IEEE INFOCOM 1997.

[4] D. Ferrari and D. Verma. "A Scheme for real-time channel establishment in wide-area networks," IEEE Journal on Selected Areas in Communications, Vol.8, pp.368-379, April 1990.

[5] J. Liebeherr, D.E. Wrege, and D. Ferrari. "Exact Admission Control in Networks with Bounded Delay Sevices," IEEE/ACM Transactions on Networking, Vol.4, No.6, pp.885-901, December 1996.

[6] H. Zhang. "Providing End-to-End Performance Guarantees Using Non-Work-Conserving Disciplines," Computer Communications Journal : Special Issue on System Support for Multimedia Computing, Vol.18, No.10, October 1995.

[7] Q. Zheng, Y. Nemoto. "Connection Admission Control for Hard Real-Time Communication in ATM Networks," The 17th International Conference on Distributed Computer Systems, pp.28-30 May 1997.

[8] J. S. Turner. "New Directions in communications(or Which Way to the Information Age ?)," IEEE Communication Magazine, Vol.25, No.8, pp.8-15, October 1986.

[9] The ATM Forum Technical Committee, "Traffic Management Specification Version 4.0," af-tm-0056.000, April 1996.

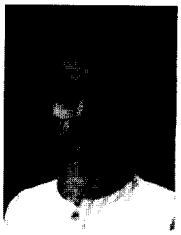
[10] Alan Demers, Srinivasan Keshav, and Scott Shenker, "Analysis and simulation of fair queueing algorithm," In Journal of Internet-working Research and Experience, pp.3-26, October 1990. Also in Proc. ACM SIGCOMM, pp.3-12, 1989.

[11] Lixia Zhang. "A New Architecture for Packet Switched Network Protocols," PhD dissertation, Massachusetts Institute of Technology, July 1989.

[12] S. Jamaloddin Golestani. "A stop-and-go queueing framework for congestion management," In SIGCOMM Symposium, Communications Architecture & Protocols, pp.8-18, Philadelphia Pennsylvania, September 1990, ACM SIGCOMM.

[13] Charles R. Kalmanek, Hemant Kanakia, and Srinivasan Keshav. "Rate controlled servers for very high-speed networks," In IEEE Global Tele-

communications Conference, San Diego, California, December 1990.



허 권

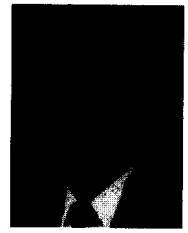
e-mail : kwon@kisco.co.kr

1997년 서원대학교 전자계산학과 졸업(학사)

1999년 충북대학교 전자계산학과 졸업(석사)

1999년~현재 신텔 정보통신 근무

관심분야 : 실시간 시스템, 실시간 스케줄링, 실시간 네트워크



김 명 준

e-mail : mjkim@cbucc.chungbuk.ac.kr

1979년 서강대학교 수학과 이학사 (부전공 : 전자계산학)

1984년 플로리다 공과대학 전자계산학 공학석사

1992년 텍사스 A&M 대학교 전자계산학 공학박사

1992년~1993년 한국전기통신공사 초빙연구원

1996년~1997년 한국전자통신연구소 초빙연구원

1993년~현재 컴퓨터과학과 부교수

관심분야 : 실시간시스템, 운영체제, 데이터베이스