실시간 통신을 위한 Multiple Rotating Priority Queues 스케줄러

허권, 박윤석, 신규철, 김명준
충북대학교 전자계산학과

Multiple Rotating Priority Queues Scheduler for Real-Time Communication

Kwon Hur, Yun Seok Park, Kiu Cheol Shin, Myung Jun Kim
Department of Computer Science , Chungbuk National University

요 약

실시간 스케줄러는 대역폭, 필요 버퍼량 등과 같은 네트워크 자원을 효율적으로 이용하면서 한정된 통신 지연(bounded delay)을 제공해야 한다. 최근 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.

1. 서 론

최근 컴퓨터 네트워크를 기반으로 한, 화상회의 원격회의 VOD 등과 같은 많은 응용프로그램들이 실시간 통신을 요구하고 있다. 이와 같은 응용프로그램들이 실시간 통신을 하기 위해서는, 근원지(source)에서 목적지(target)까지의 경로(route)를 결정하고, 그 경로에 포함된 교환기(switch)와 링크(link)의 자원을 해당 패킷들을 대표하여 할당받는 connection을 설정해야 한다. 이렇게 설정된 connection을 통해, 각 응용프로그램들은 한정된 통신지연(bounded delay)을 보장하면서 통신을 해야된다[1][4][5].

통신지연은 물리적으로 고정된 통신지연을 갖는 고정 지연과 특정 스케줄링 정책에 따라 통신지연 정도가 달라지는 가변지연으로 구분된다. 고정 지연으로는 switching delay와 propagation delay가 있고, 가변 지연으로는 scheduling delay와 queuing delay가 있다[2][3]. 모든 connection의 주어진 지연한계(delay bound)를 만족하기 위해서는 scheduling delay를 최소화시키고, 각 connection의 주어진 지연한계를 고려하여 queueing delay를 조절해야 한다. 이러한 작업은 패킷들 간의 전송 경쟁을 제어하는 스케줄러에 의해서 이루어지며, 대표적인 스케줄링 방법론으로는 정적 스케줄링(static scheduling) [1][4][5][6], EDF (Earliest Deadline First) [4][5][7][8] 스케줄링, RPQ (Rotating Priority Queues) 스케줄링[8] 등이 있다.

정적 스케줄러는 고정된 우선 순위를 제공하고, 각 connection들은 connection들의 주어진 지연한계에 따라 우선 순위를 할당받는다. 이러한 정적 스케줄러는 우선 순위 개수만큼의 우선 순위 FIFO queue를 갖는다. 그리고 각 패킷은 스케줄러 입력 시, 해당 connection이 할당받은 우선 순위에 해당하는 우선 순위 FIFO queue에 저장된다. 정적 스케줄러는 우선 순위 순으로 FIFO queue를 선택하고 그 queue에 저장된 packet들을 차례대로 전송하게 된다. EDF 스케줄러는 새로운 패킷이 스케줄러에 들어 올 때마다 가장 짧은 deadline을 갖는 패킷을 선택하여 전송하게 된다. 이 때 deadline은 패킷이 입력된 시간에 주어진 지연한계를 더한 시간이 된다. EDF 스케줄링 방법론은 링크의 대역폭(bandwidth)을 최대로 활용할 수 있는 최적의 실시간 스케줄링 방법론으로 알려져 있다[7][8]. 그러나 정적 스케줄링에서는 우선 순위 할당이 connection 설정 과정에서만 이루어지는데 반해, EDF 스케줄러는 입력된 모든 패킷들 중에서 가장 짧은 deadline을 갖는 패킷을 선택하기 위해서 새로운 패킷이 입력될 때마다 "search" 연산이나 "sort" 연산을 수행하게 된다. 이러한 연산 작업은 scheduling delay를 가중시킨다.

최근 간단한 연산 작업과 정적 우선 순위만으로 EDF 스케줄러와 유사한 동작을 수행하는 RPQ (Rotating-Priority-Queues) 스케줄링 방법론이 개발되었다. RPQ 스케줄러는 우선 순위 queue들의 우선 순위를 나타내는 색인 값을 회전 간격(rotating interval)이라는 일정시간 마다 변화시킴으로서 EDF 스케줄링과 유사한 동작을 하게 되고, EDF 스케줄링에 근접한 스케줄링 가능도(schedulability)를 갖는다.[8]. 그러나 RPQ 스케줄링은 과다한 버퍼량을 필요로 하는 문제점이 있다. RPQ 스케줄링을 하기 위해서 필요한 버퍼량은 2장에서 보다 자세히 알아본다. 본 논문에서는 이러한 문제점을 해결하면서 RPQ 스케줄링 방법론과 동일한 최대 지연시간을 제공하는 MRPQ (Multiple-Rotating- Priority-Queues) 스케줄링 방법론을 제시한다.

네트워크 상에 존재하는 특정 교환기는 'W_i', 교환기 W_k에 존재하는 출력 링크는 '$L_{k,z}$', 출력 링크 $L_{k,z}$에 존재하는 스케줄러는 '$S_{k,z}$'라고 하고, 특정 connection은 '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, connection M_i에 할당된 우선 순위는 '$p_{k,z,i}$'라고 한다. 그리고 우선 순위가 '0'부터 시작되는 $|P_{k,z}|$개의 $P_{k,z,j}$로 구성된 집합을 우선 순위 집합 '$H_{k,z}$'이라고 한다 [$H_{k,z}$ = {$P_{k,z,j}$ | $P_{k,z,j}$는 '0'을 포함하고 $|P_{k,z}|$ 보다 작은 자연수}] connection M_i가 경유하는 교환기 별로 미리 주어진 connection M_i의 지연한계는 '$d_{k,z,i}$'라고 하고, 특정 우선 순위를 갖는 connection M_i에 포함된 패킷의 최대 지연시간을 '$D_{k,z,i}$'라고 한다 그리고 스케줄러 $S_{k,z}$에 입력된 connection들 중에 가장 긴 $d_{k,z,i}$를 갖는 connection의 지연한계를 '$MAX_{k,z}$'라고 한다. 스케줄러 $S_{k,z}$에서 필요한 버퍼량을 '$Q_{k,z}$'라고 하고, 링크의 대역폭은 'B'로 표기한다. 단 네트워크 상에 존재하는 모든 링크의 대역폭은 동일하다고 가정한다. 임의 시간 t 동안 전송할 수 있는 최대 데이터량을 'Γ_t'라고 한다.

정적 스케줄러나 RPQ 스케줄러에서는 connection에 정적 우선 순위를 할당하기 위한 기준으로 '우선 순위 할당 범위' '$E_{k,z,i}$'라는 것을 결정한다. 예를 들어 connection의 지연한계가 (0msec, 3msec] 사이에 존재하는 connection에는 우선 순위 '0'을 할당하고, (3msec, 6msec] 사이에 존재하는 connection에는 우선 순위 '1'을 할당할 경우, (0msec, 3msec]는 우선 순위

'0'에 대한 우선 순위 할당 범위라고 하고, (3msec, 6msec]는 우선 순위 '1'에 대한 우선 순위 할당 범위라고 한다. 이때 우선 순위 할당 범위 간격은 3msec가 되고, 우선 순위 개수는 '2'가 된다.

2장에서는 RPQ 스케줄링 방법론에 대해서 자세히 설명한다. 그리고 3장에서는 RPQ 스케줄링의 문제점을 해결하기 위한 MRPQ 스케줄링에 대해서 제시한다. 4장에서는 MRPQ 스케줄링 방법론의 특성을 알아보고 5장에서는 결론을 맺도록 한다

2. Rotating-Priority-Queues 스케줄러

RPQ (Rotating Priority Queues) 스케줄러는 각 connection의 지연한계를 보장하기 위해서, (과정-1)과 같은 방법으로 각 connection에 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 queue를 보유한다. 각각의 *FIFO queue*들은 자신들의 우선 순위를 나타내는 우선 순위 색인과 연결된다.

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

㉮ $|P_{k,z}|$와 $H_{k,z}$ 그리고 $E_{k,z,i}$을 결정한다.

$$|P_{k,z}| = \lceil \frac{MAX_{k,z}}{\Delta} \rceil$$

단, 모든 $E_{k,z,i}$의 간격은 회전 간격(Δ)과 동일하다

㉯ 특정 $E_{k,z,i}$에 포함되는 $d_{k,z,i}$를 요구하는 M_i에 해당 $P_{k,z,i}$를 부여한다.

단, 모든 connection들 중에 가장 짧은 지연한계를 요구하는 connection의 지연한계는 회전 간격(Δ)보다 커야 한다

RPQ 스케줄러는 회전 간격마다 FIFO queue들을 회전하게 된다. 이러한 회전 동작은 queue의 우선 순위를 높여주고 queue에 저장된 패킷들의 우선 순위도 높아지게 된다. 회전 동작은 queue안에 입력된 패킷을 다른 패킷으로 복사하는 작업으로 이루어지는 것이 아니라, 단순히 queue와 연결된 색인 값을 감소시킴으로서 회전 효과를 얻게된다. 단, 우선 순위 색인 값의 순환을 고려하여 가장 높은 우선 순위를 나타내는 우선 순위 색인은 가장 낮은 우선 순위를 나타내는 우선 순위 색인 값을 갖게 된다.

RPQ 스케줄러에서 우선 순위 $p_{k,z,i}$를 갖는 connection Mi의 최대 지연시간은 '$(p_{k,z,i}+1)*\Delta$'이다[8]. 그리고 각 우선 순위 queue에서 요구되는 버퍼량은 해당 우선 순위를 갖는 패킷의 최대 지연시간 동안 최대로 전송할 수 있는 데이터량과 동일하다 그러나 queue를 회전함으로서, RPQ 스케줄러의 모든 우선 순위 queue에서는 가장 낮은 우선 순위 queue에서 요구되는 버퍼량과 동일한 크기의 버퍼량을 필요로 한다 이러한 특성으로 인해, RPQ 스케줄러의 스케줄링 가능도를 EDF 스케줄러의 스케줄링 가능도에 근접하게 하기 위해서 회전 간격을 줄일 경우, 보다 많은 우선 순위를 가져야 되는 RPQ 스케줄러는 보다 많은 버퍼량을 필요로 하게 된다. 필요 버퍼량(Q)과 회전 간격(Δ)과의 관계는 (정리-1)과 같다.

(정리-1) RPQ 스케줄러에서 필요한 버퍼량은 (1)과 같다.

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

$$\Gamma_{\Delta} = B*MAX_{k,z} / |P_{k,z}|$$

3. MRPQ 스케줄러

본 논문에서는 회전 간격을 좁게 하면서 필요 버퍼량을 줄이기 위한 방안으로, RPQ 스케줄러와 같이 회전 우선 순위 queue들로 구성되면서 동일한 방법으로 회전하는 "계층"이라는 것을 하나 이상 구성한다. 한 계층의 구성 요소인 우선 순위 queue의 개수를 '$r_{k,z}$'라고 하고 계층의 개수를 '$R_{k,z}$'이라고 한다. 만일 회전 간격이 RPQ 스케줄러와 동일하다면, 전체 우선 순위 queue의 개수는 RPQ 스케줄러에 존재하는 우선 순위 queue의 개수와 동일하다. 이와 같이 여러 계층으로 RPQ 스케줄러를 구성한 방법론이 본 논문에서 제시하는 MRPQ (Multiple Rotating Priority Queues) 스케줄러이다. MRPQ 스케줄러의 자세한 구성 방법은 (과정-2)와 같다.

(과정-2) Multiple Rotating Priority Queues 스케줄러 구성 방법

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

㉯ $|P_{k,z}|$와 동이한 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 $I_{k,z,n}$을 구성한다.

조건-1) '$r_{k,z}$'과 '$R_{k,z}$'은 스케줄러 설계자에 의해서 임의적으로 변할 수 있으나, '$|P_{k,z}| = R_{k,z} * r_{k,z}$'을 만족해야 한다

㉱ ㉰를 '$R_{k,z}$'번 반복하여 '$R_{k,z}$'개의 $I_{k,z,n}$을 구성함.

조건-1) 한번 선택된 우선 순위 색인은 다시 선택될 수 없다

㉲ $I_{k,z,n}$에 포함된 '$r_{k,z}$'개의 우선 순위 색인과 '$r_{k,z}$'개의 queue를 이용하여 한 개의 회전 우선 순위 queue 계층을 구성한다

조건-1) 우선 순위 색인과 queue는 일대일로 연결된다

㉳ ㉱를 '$R_{k,z}$'번 반복하여 '$R_{k,z}$'개의 회전 우선 순위 queue 계층을 만든다.

한 계층의 구성 요소인 queue는 RPQ 스케줄러의 '$r_{k,z}$-1'번째 우선 순위 queue에서 요구되는 버퍼량과 동일한 크기의 LIFO(Last In First Out) queue인 "block queue"의 조합으로 구현된다 그리고 한 개의 queue를 구성하는 "block queue"들간의 관계는 LIFO이고 각 "block queue"는 출력되는 순서로 "block 우선 순위"를 할당받는다. 단, 각 계층에서 가장 낮은 우선 순위 색인값은 '$R_{k,z}$'와 같다. 각 계층별로 queue를 구성하는 "block queue"의 수는 다르며 '0'부터 '$r_{k,z}$-1'까지의 우선 순위 색인 값을 갖는 우선 순위 색인들로 구성된 첫 번째 계층에서는 각각의 queue들이 한 개의 "block queue"로 구현되고, '$r_{k,z}$'부터 '$2*r_{k,z}-1$'까지의 우선 순위 색인 값을 갖는 우선 순위 색인들로 구성된 두 번째 계층에서는 두 개의 "block queue"로 구현된다 그리고 '$R_{k,z}$'번째 계층에서는 '$R_{k,z}$'개의 "block queue"로 구현된다. 이와 같은 특성은 queue들이 각 계층별로 우선 순위가 가장 낮은 *queue*에서 요구하는 버퍼량과 동일한 크기의 버퍼량을 필요로 하고, 각 계층에서 우선 순위가 가장 낮은 queue들의 우선 순위 차가 일정하기 때문이다. 본 논문에서는 낮은 우선 순위 색인 값을 갖는 계층일수록 상위 계층이라고 한다.

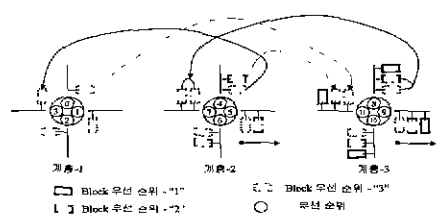

그림-1

이렇게 구성된 MRPQ 스케줄러는 회전 동작을 수행할 때마다 서로 다른 계층에 존재하면서 근접한 우선 순위를 갖는 *queue*들간에 패킷 이동이 발생된다. 예를 들어 (그림-1)와 같이 MRPQ 스케줄러를 구성하였을 경우, 두 번째 계층에 존재하면서 우선 순위 '3'인 queue에 저장된 패킷을 첫 번째 계층에 존재하는 우선 순위 '2'인 queue로 이동한다 그러나 *queue*의 구성은 LIFO 관계로 연결된 "block queue"의 조합으로 구현하였기 때문에 이동하고자 하는 패킷은 항상 일정한 "block queue"에 존재하게 된다. 그러므로 스케줄러는 계층간을 연결하기 위해서 미리 정해진 "block queue"들만을 이동시키면 된다. 각 계층별로 이동하게 되는 "block queue"는 다음과 같다. '$R_{k,z}$' 번째 계층에 포함된 '$r_{k,z}*(R_{k,z}-1)$' 우선 순위 queue에서 '$R_{k,z}$-1' 번째 계층에 포함된 '$r_{k,z}*(R_{k,z}-1)$-1' 우선 순위 queue로 패킷을 전달하기 위해서는 '$R_{k,z}$-1'개의 낮은 "block 우선 순위"를 갖는 "block queue"를 이동하면 된다 그리고 최하위 계층을 제외한 모든 계층에서는 회전 간격마다, 상위 계층으로 이동하지 않은 "block queue"를 최하위 계층에 존재하는 최하위 우선 순위 queue로 반환한다 그러나 이러한 작업은 scheduling delay를 가중시킨다. 본 논문에서는 계층간에 이동하게 되는 "block queue"들이 일정하다는 것에 착안하여 이동 작업을 수행하지 않고 스케줄링을 하는 방법을 다음과 같이 제시한다. 동일 "block 우선 순위"를 갖는 "block queue"들로 구성된 새로운 계층을 구성한다. 각 계층을 구성하는 "block *queue*"들은 우선 순위 색인과 일대일로 연결된다. 이때 우선 순위 색인 값은 "block queue"들이 소속되었던 queue의 우선 순위 색인 값과 동일하다. 이렇게 구성된 계층의 수는 이진

에 존재했던 계층의 수와 동일하다.

각 계층에서의 우선 순위 색인들의 색인 값은 다음과 같다. "block 우선 순위"가 '1'로 구성된 계층에서는 '$|P_{k,z}|-r_{k,z}$'에서 '$|P_{k,z}|-1$'까지의 우선 순위 색인 값을 갖는 우선 순위 색인들로 구성되고, "block 우선 순위"가 '2'로 구성된 계층에서는 '$|P_{k,z}|-r_{k,z}*2$'에서 '$|P_{k,z}|-1$'까지의 우선 순위 색인 값을 갖는 우선 순위 색인들로 구성된다. 그리고 $R_{k,z}$번째 "block 우선 순위"로 구성된 계층에서는 '$|P_{k,z}|-r_{k,z}*R_{k,z}$'에서 '$|P_{k,z}|-1$'까지의 우선 순위 색인 값을 갖는 우선 순위 색인들로 구성된다. "block 우선 순위"가 높은 "block queue"들로 구성된 계층을 상위 계층이라 하고, "block 우선 순위"가 낮은 "block queue"들로 구성된 계층을 하위 계층이라 한다. (그림-2)는 (그림-1)를 위와 같은 방법으로 구현한 것이다.

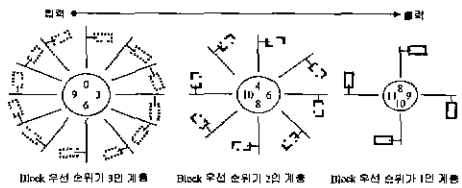

그림-2

MRPQ 스케줄러의 스케줄링 과정은 다음과 같다. 실시간 통신을 원하는 connection들은 (과정-1)과 동일한 방법으로 우선 순위를 할당받는다. 그리고 connection에 소속된 패킷이 스케줄러에 입력될 경우, 그 패킷은 connection과 동일한 우선 순위를 갖는 "block queue"에 LIFO 순서로 입력된다. 단, 해당 "block queue"가 여러 계층에 존재한다면 하위계층에 입력된다 그리고 MRPQ 스케줄러는 회전 간격마다 모든 우선 순위 색인 값을 자연수 '1'씩 감소시킨다 패킷을 전송하고자 할 때에는 가장 높은 우선 순위 색인 값을 가지면서 동시에 가장 높은 "block 우선 순위"를 갖는 "block queue"를 선택한 후, 그 "block queue"에 저장된 패킷을 LIFO 순서대로 전송한다. MRPQ 스케줄러에 입력된 패킷의 최대 지연시간은 패킷이 입력된 시간부터 해당 "block queue"의 우선 순위 색인 값이 최하위 우선 순위 색인 값을 갖게되는 바로 직전시간까지의 시간이 된다. 그러므로 특정 우선 순위를 갖는 패킷의 최대 지연시간은 $(D_{k,z,i}=(p_{k,z,i}+1)*\Delta)$이 된다. 이는 RPQ 스케줄러와 동일하다. 만일 특정 패킷이 입력된 "block queue"의 우선 순위 색인 값이 최하위 우선 순위 색인 값을 갖게되는 바로 직전까지 해당 패킷을 전송하지 못한다면 그 패킷은 주어진 지연한계를 지키지 못하게 된다.

4. MRPQ 스케줄러의 특성

최대 지연시간 $D_{k,z,i}$ 동안 전송할 수 있는 데이터량의 크기와 $p_{k,z,i}$와 동일한 $P_{k,z,j}$를 갖는 "block queue"들의 전체 크기가 동일하므로 스케줄러에서 필요한 전체 버퍼량은 각 계층에 존재하는 "block queue"들의 전체 크기를 계층별로 더하여 구할 수 있다. 각 계층에 존재하는 "block queue"들의 전체 크기는 각 계층의 우선 순위 수에 "block queue"의 크기를 곱한 만큼의 버퍼량이며, 각 계층의 우선 순위 수는 최상위 계층에서의 우선 순위 수 '$v_{k,z}$'에 "block 우선 순위"를 곱한 수와 같다 (정리-2)는 위와 같은 방법으로 일반화된 필요 버퍼량 계산 방법을 유도한 것이다.

(정리-2) MRPQ 스케줄러에서 필요한 버퍼량은 (3)와 같다.

$$Q_{k,z}= \Gamma_{\Delta}* R_{k,z}* v_{k,z}{}^{2}(R_{k,z}+1)/2 \qquad (3)$$
$$\Gamma_{\Delta} = B*MAX_{k,z} / |P_{k,z}| ,$$
$$R_{k,z}= \lceil |P_{k,z}|/ v_{k,z} \rceil , \; v_{k,z}= r_{k,z}$$

RPQ 스케줄러에서는 입력된 모든 패킷들의 주어진 지연한계를 보장하기 위해서, 우선 순위가 가장 높은 queue에 저장된 패킷들을 회전 시간 직전까지 모두 전송해야 되는 반면에 MRPQ 스케줄러에서는 각 계층에서 우선 순위가 가장 높은 "block queue"에 저장된 패킷들을 회전 시간 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 "block queue"들이 서로 LIFO 관계로 입·출력하고, 우선 순위 $P_{k,z,j}$를 갖는 모든 "block queue"에서 요구되는 전체 버퍼량이 RPQ 스케줄러에서 우선 순위 $P_{k,z,j}$를 갖는 queue에서 요구하는 버퍼량과 동일하기 때문이다. 이때 두 스케줄러에서 요구하는 버퍼량은 $(P_{k,z,j}+1)*\Delta$ 동안 최대로 전송할 수 있는 양이다.

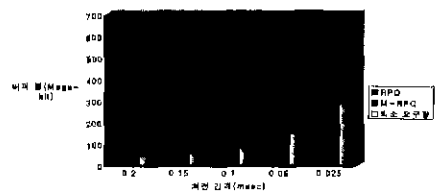

그림-3

(그림-3)은 아래와 같은 조건하에서 MRPQ 스케줄러에서의 필요 버퍼량과 RPQ 스케줄러에서의 필요 버퍼량 그리고 RPQ 스케줄러에서 queue의 회전을 고려하지 않았을 때, 요구되는 버퍼량을 비교한 것이다. MRPQ의 $v_{k,z}$ 변수를 "2"로 하였고, 링크의 대역폭(B)을 155 Mbps, 최대 지연한계(MAX)를 10 msec, 회전 간격을 0.2 msec에서 0.025 msec까지 좁혔다. (그림-3)에서와 같이 MRPQ 스케줄러에서 필요한 버퍼량은 RPQ 스케줄러에서 필요로 하는 버퍼량의 절반에 가까운 양에 불과하다. 또한 이 크기는 우선 순위 queue의 회전을 고려하지 않은 요구 버퍼량과 유사하다. 이러한 특성은 MRPQ 스케줄러가 RPQ 스케줄러에서 필요로 하는 버퍼량만으로 회전 간격을 절반에 가깝게 줄일 수 있다는 것을 나타낸다.

만일 MRPQ 스케줄러와 RPQ 스케줄러의 회전 간격이 동일하다면, MRPQ 스케줄러에 입력된 패킷들과 RPQ 스케줄러에 입력된 패킷들은 동일한 최대 지연시간을 갖는다 이러한 특성은 MRPQ 스케줄러와 RPQ 스케줄러의 우선 순위할당 기준과 전체 우선 순위의 수가 동일하고 두 스케줄러가 회전 간격마다 패킷의 우선 순위를 높이게 되므로 최하위 우선 순위를 갖는 queue에 도달하는 시간이 동일하기 때문이다

5. 결론 및 차후 연구 과제

MRPQ 스케줄러는 RPQ 스케줄러와 같이 우선 순위 색인 값만을 감소시킴으로서 EDF 스케줄러에서 수행되는 "sort"나 "search" 연산 작업과 유사한 동작을 수행하게 된다. 그러나 RPQ 스케줄러에서는 queue와 연결된 우선 순위 색인 값을 변화시킴으로서 최하위 우선 순위 queue에서 필요로 하는 버퍼량을 모든 queue에서도 필요로 하게 되는 반면, MRPQ 스케줄러에서는 우선 순위 queue를 여러 계층으로 분할된 "block queue"의 조합으로 구성함으로서 필요 버퍼량을 최대로 줄일 수 있다. 이러한 결과, MRPQ 스케줄러는 RPQ 스케줄러에서 필요로 하는 버퍼량의 절반에 가까운 버퍼량만으로 동일한 회전 효과를 얻을 수 있다

MRPQ 스케줄러에 입력된 모든 패킷들이 그들의 주어진 지연한계를 만족하기 위한 필요조건은 해당 우선 순위가 제공하는 최대 지연시간보다 주어진 지연한계가 같거나 작아야 한다 그러나 MRPQ 스케줄러를 보다 효율적으로 실시간 네트워크에 적용하기 위해서는 보다 정확한 필요 충분조건이 요구된다.

참고 문헌

[1] Chengzhi Li, Riccardo Bettati, Wei Zhao "Static Priority Scheduling for ATM Networks" In Proc of the Real-Time Systems Symposium, San Francisco, CA, Dec 1997
[2] A Raha, S Kamat, and W Zhao, "Admission control for hard real-time connections in ATM LANs" In Proc of the IEEE INFOCOM, Mar 1996
[3] Q Zheng and K. G Shin "Fault-Tolerant Real-Time Communication in Distributed Computing Systems" In IEEE Trans on Parallel and Distributed systems, May 1993 pp 470-480.
[4] C Aras, J Kurose, D Reeves, H Schulzrinne "Real-Time Communication in Packet-Switched Networks" In Proc IEEE, Vol 82, No 1, Jan, 1994, pp. 129-139
[5] Dilip D Kandlur, Kang G Shin, Domenico Ferrari "Real-time Communication in Multi-hop Networks" In 11th International Conference on Distributed Computing Systems (DCS), pp. 300-307, Arlington, TX, 1991
[6] Hui Zhang and Domenico Ferrari "Rate-controlled static-priority queueing" In Proc. of IEEE INFOCOM, volume 1, pp. 227-236, San Francisco, CA, 1993
[7] Victor Firoiu, Jim Kurose, Don Towsley "Efficient Admission Control for EDF Schedulers" In Proc of the IEEE INFOCOM '97 1997
[8] 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