중간노드의 계산량을 줄이기 위한 분산처리 방식의 Fair Queueing

A Distributed Method of Fair Queueing to Reduce the Computational Load of Intermediate Nodes

이 준 엽 (광운대학교, 석사과정)
이 승 형 (광운대학교, 교수)
양 훈 기 (광운대학교, 교수)

목 차

I. 서 론

패킷에 대한 스케쥴링(Scheduling)은 현재 Best-effort 방식으로 운용되는 인터넷에서 사용자들에게 서비스의 차별화 혹은 전송품질의 보장을 제공하기 위한 경우와 ITS(Intelligent Transportation System)와 같은 지능형 운송 시스템인 경우에 필요한 기능이다. ITS 분야는 MIT의 Wireless Sensor Node[1]와 Fleetnet[2] 등의 프로젝트에서 적용이 되고 있다. Wireless Sensor Network인 경우 정보 전달에 있어서 먼 거리에 있는 노드에 효율적으로 정보를 전달하기 위해서 분산된 프로세싱이 필요하며, Fleetnet 경우도 Multi-hop으로 이루어진 자동차간의 통신을 가능하게 하기 위해 ad hoc 네트워크를 구축하는데 이들 또한 분산된 환경에서의 정보 전달을 필요로 하게 된다. 지금까지 제안된 스케쥴링 알고리즘들은 간단한 구현에 의한 고속의 처리를 장점으로 하는 Round-Robin 방식[3,4]과 정확한 서비스를 장점으로 하는 Fair Queueing 방식[5,6,7]으로 나눌 수 있다. 계산량이 적으면서도 각 플로우에 차별화 된 서비스를 제공할 수 있는 DRR[3]은 현재 많은 상용제품에 응용되어 사용되고 있다[4]. 그러나 이러한 Round-Robin 방식의 스케쥴링 알고리즘은 모든 플로우를 정해진 순서대로 서비스하는 특성 때문에 정확한 서비스를 제공하는 데에는 한계가 있다. 한편, Fair Queueing 방식의 알고리즘들은 가장 이상적인 서비스 모델인 GPS (Generalized Processor Sharing)[5]의 동작을 패킷 단위의 서비스로 모방하기 위한 것으로써, Round- Robin 방식의 알고리즘 보다 훨씬 정확한 서비스를 플로우들에게 제공할 수 있다. 그러나, GPS에서의 서비스 순서를 기준으로 하기 때문에 스케쥴러는 모든 플로우의 모든 패킷에 대하여 도착시간 및 서비스완료 시간을 계산해야 하고, 이를 위해 시스템 차원의 가상적인 시간을 관리하여야 한다[5,6,7]. 이는 실제 구현 시에 스케쥴러의 계산량을 증가시켜서 시스템의 전송성능을 저하시키며 시스템의 확장성(Scalability)에 문제를 야기하는 원인이 된다. [8]에서 본 저자들이 제안한 DCFQ 알고리즘은 Fair Queueing 방식의 스케쥴러가 시스템 전체의 상태를 관리함으로 인한 계산량 및 구현상의 문제점을 없애는 동시에, 기존에 가장 정확한 성능을 낸다고 알려진 WF2Q[7]와 유사한 수준의 서비스를 제공한다. 본 논문에서는 DCFQ 알고리즘에 대해 TCP 트래픽을 적용하여 성능을 분석하고 다른 스케쥴링 방식과의 동작 특성 및 성능의 차이를 비교 분석한다. 본 논문의 구성은 다음과 같다. 2장에서는 DCFQ 알고리즘에 대한 간단한 설명과 3장에 시뮬레이션 결과를 제시한다. 4장에서 기존의 패킷 스케쥴링 방식들과 패킷 서비스 순서 및 계산량에 대한 비교를 한 후에 5장에서 DCFQ의 응용 및 향후 연구방향을 언급한다.

II. DCFQ : 분산처리 방식의 스케쥴링

대역폭 R의 출력 링크에 N개의 플로우 N={1,2,...N} 이 있다고 가정하고 플로우 i는 Ri의 서비스 율을 할당받아서 $\Sigma i R_i \leq R$이라고 가정한다. 플로우 i는 다른 플로우의 상태 혹은 전체 링크의 상태에 무관하게 독립적으로 자신의 상태를 관리하고, 패킷 스케쥴

러는 각 패킷 및 시스템에 대한 정보를 관리하지 않으며 각 플로우가 제공하는 정보에 따라 다음 서비스할 패킷을 선택한다. 플로우 i의 큐에 대한 정보는 증가 카운터 Ci와 레지스터 Di에 의해 관리된다. 스케줄러는 Di의 값을 비교하여 서비스의 순서를 정하게 되며, 각각의 큐에 대한 정보는 분산되어 있고 서로 독립적으로 관리된다. Pik를 플로우 i의 k번째 패킷이라고 하고 그 길이가 Lik라고 한다. Pik가 DCFQ에서 서비스가 시작되는 시간을 Sik라 하면, 카운터 Ci의 시간대 t에서의 증가율은 다음과 같이 정의된다.

$$\frac{d}{dt} C_i = \begin{cases} 0, \text{ if } S_i^k \le t \le S_i^k + \dfrac{L_i^k}{R} \\ R_i, \quad otherwise \end{cases} \qquad (1)$$

즉, Ci는 자신의 패킷이 서비스 받는 동안에는 증가하지 않으며, 그 외에는 Ri의 비율로 단조 증가한다. Li를 플로우 i의 맨 처음에 위치한 패킷의 크기라 할 때, 플로우 i는 Ci와 Di의 값을 계속 갱신한다. 이 Di는 스케줄러가 다음에 서비스할 패킷을 결정하는데 기준이 되며 다음과 같이 정의된다.

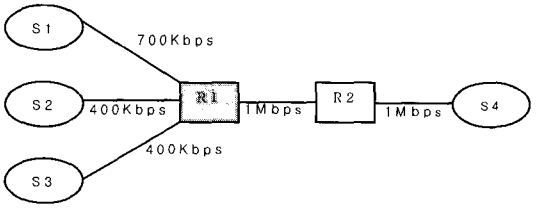
$$D_i = \begin{cases} \dfrac{C_i}{L_i}, flow\ i\ is\ backlogged \\ 0, \quad otherwise \end{cases} \qquad (2)$$

Di의 값은 큐가 비어있지 않은 플로우의 경우에 서비스를 받지 못하는 동안 계속 증가하며, 그 증가율은 Ri 및 Li의값에 따라 결정된다. P_i^k 가 선택되어 서비스를 받게 되는 경우에 Ci의 값은 다음과 같이 조정된다.

$$C_i = \max(C_i - L_i^k, 0) \qquad (3)$$

III. 시뮬레이션

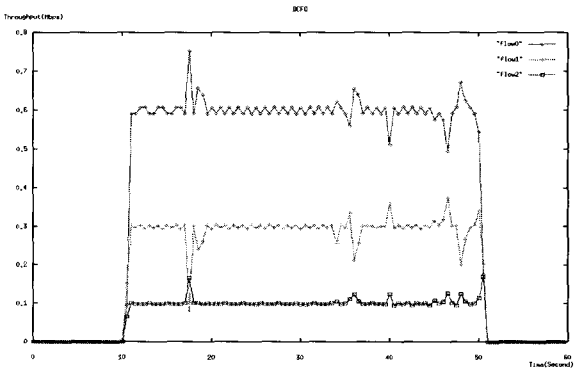
본 장에서는 [8]에서 제안된 DCFQ 알고리즘을 그림 1과 같은 IP 네트워크 구조를 대상으로 시뮬레이션한 결과를 제시한다. 시뮬레이션은 ns[9]를 사용하여 수행되었다. 그림 1에서 세 개의 소스 S1, S2, S3는 라우터 R1과 R2를 통해 싱크 S4로 packet을 전송하는 UDP 호스트들이며, R1에는 앞장에서 제안된 DCFQ 알고리즘을 적용하여 세 플로우에 대한 서비스를 하도록 한다.



〈그림 1〉 시뮬레이션 네트워크의 구성

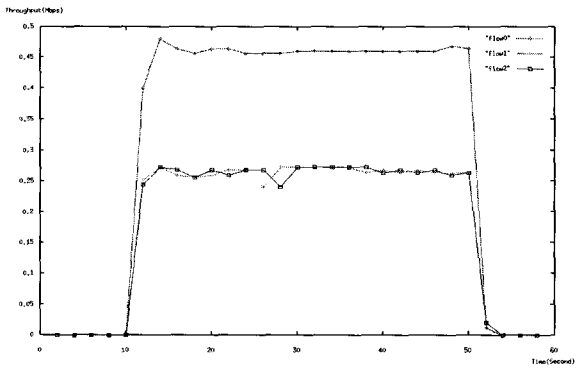
S1, S2 및 S3는 on-off 소스들로서 각각 700Kbps, 400Kbps 및 400Kbps의 Burst peak rate를 가지며, Burst time과 Idle time의 비율은 모두 2:0.1로 설정되었다. R1의 출력 링크 용량이 1Mbps이므로 R1-R2 링크에서 병목이 발생하며 DCFQ에 의한 대역폭 관리가 이루어진다. S1과 S2에서 보내는 패킷의 크기는 1Kbytes, S3의 패킷은 200Bytes이며, 각 플로우에 대한 서비스 웨이트(Weight)는 $\varphi 1=6$, $\varphi 2=3$, $\varphi 3=1$로 설정하였다. 이러한 네트워크 설정에 대하여 시뮬레이션 한 결과는 〈그림 2〉와 같다.

〈그림 2〉에서 보듯이, 각 플로우는 소스의 전송률에 상관없이 미리 할당된 대역폭만을 사용하고 있음을 알 수 있다. 패킷의 서비스 순서 결정을 위하여 시스템 차원의 가상 시간 정보를 사용하지 않고 분산된 클럭(카운터)을 사용하여 Fair Queueing을 수행할 수 있음을 보여주고 있다. 〈그림 2〉에서 플로우의 수율(Throughput)이 불규칙하게 급변하는 이유는 다른 소스가 Idle time일 때 출력 대역폭에 생기는 여유 분을 다른 플로우가 사용하기 때문이다.



〈그림 2〉 UDP를 적용한 시뮬레이션 결과

〈그림 3〉은 〈그림 1〉과 같은 네트워크 구성에 UDP 대신에 TCP 플로우를 2:1:1의 rate로 사용했을 경우에 싱크 노드인 S4에서 각각의 플로우에 대한 수율을 나타낸 것이다.

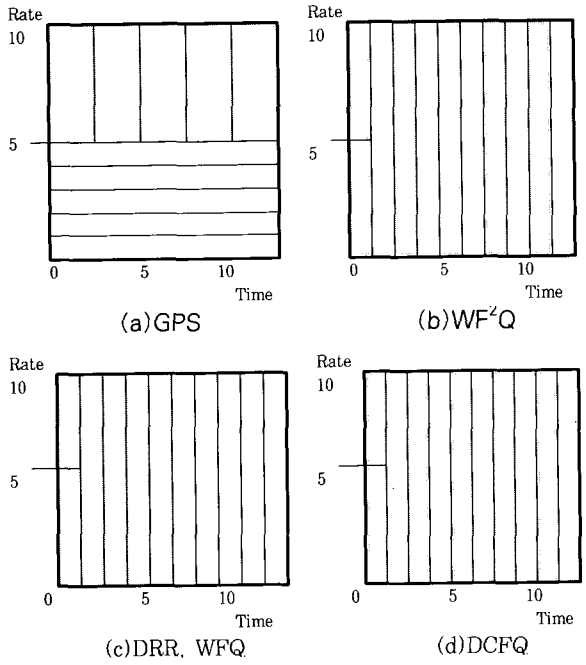


〈그림 3〉 TCP를 적용한 시뮬레이션 결과

Ⅳ. 다른 스케줄링 방식과의 비교

1. 패킷 서비스 방식

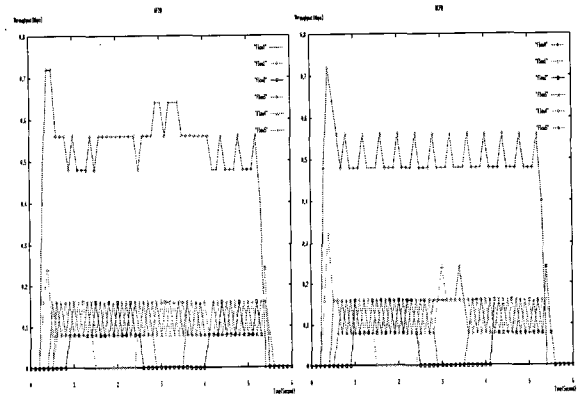
본 장에서는 DCFQ 알고리즘과 기존에 제안된 다른 패킷 스케줄링 방식과의 서비스 특성상의 차이에 대하여 비교한다. 〈그림 4〉는 DCFQ와 비교 대상인 GPS, WF2Q, DRR, WFQ의 패킷 서비스 특성을 나타낸 것이다. t=0에서 스케줄러가 서비스가 시작될 때 6개의 플로우가 많은 수의 패킷을 각자의 큐에 저장하고 있다고 가정한다. 이때 출력 링크의 속도는 10이고 모든 패킷의 길이도 10이며, 각 플로우의 서비스 웨이트(Weight)는 $\varphi 1=5$, $\varphi 2=\ldots=\varphi 6=1$으로 설정한다. 각 도표는 10의 출력율로 10초간 패킷들이 서비스 받는 과정을 나타내며, 모든 스케줄링 방식은 Work Conservation의 원칙을 따르기 때문에 모든 시간대에서 출력 링크의 서비스 율은 10이 된다. 또한 각 도표 내의 직사각형들은 크기가 10인 패킷이 서비스 받는 속도와 서비스 시간을 나타내고 있다. 음영으로 표시된 부분은 50%의 서비스를 할당받은 플로우 1의 패킷들을 나타낸다.



〈그림 4〉 여러 패킷 스케줄링 방법들의 서비스 순서

(a)의 이상적인 GPS의 경우에는 모든 플로우가 할당된 비율로 동시에 서비스를 받으므로, 다른 플로우의 패킷이 하나 처리되는 동안에 플로우 1의 패킷은 다섯 개가 전송된다. (b)는 WF2Q의 서비스 형태를 나타낸 것이다. 플로우 1의 패킷들과 다른 플로우의 패킷들이 하나씩 교대로 전송되어 할당된 서비스 웨이트를 만족함을 알 수 있다. (c)는 DRR 및 WFQ(혹은 PGPS)의 패킷 서비스 순서를 나타낸 것인데, 높은 웨이트를 갖는 플로우의 경우에는 GPS에서의 서비스 보다 훨씬 우선적인 전송이 이루어짐을 알 수 있다. 즉, 10의 시간 동안에 서비스 율은 GPS와 동일하지만, 웨이트가 높은 플로우의 패킷에 대해 우선적으로 전송이 된다. 이에 따라 플로우 1은 R1을 지나면서 Burst:Idle이 1:1인 On-Off 트래픽으로 변한다. 이는 인터넷의 경우처럼 Feedback 혼잡 제어를 사용하는 경우에 문제가 있으며[7], 음성이나 비디오 등의 실시간 트래픽의 전송에도 바람직하지 않다. 한편, DCFQ의 경우에는 WF2Q와 마찬가지로 웨이트가 높은 플로우에 편중되지 않는 서비스를 제공하여 플로우 1에게 일정한 전송률을 지원함을 알 수 있다.

〈그림 5〉는 〈그림 2〉의 네트워크에 6개의 플로우가 서비스 웨이트 $\varphi 1=5$, $\varphi 2=\ldots=\varphi 6=1$을 갖는 경우에 WF2Q와 DCFQ의 수율(Throughput) 특성을 비교한 결과이다. 여기서 모든 패킷의 크기는 1Kbytes로 하였으며, 플로우 1은 CBR 트래픽으로, 나머지 플로우들은 Burst:Idle = 2:0.4인 On-Off 트래픽을 가정한다.



〈그림 5〉 WF^2Q(왼쪽)와 DCFQ(오른쪽)의 수율 특성 비교

그림에서 보듯이 3sec와 3.5sec 사이에서 두 스케줄링 알고리즘의 서비스 특성을 비교할 수 있다. 그 시간에 6개 중 두 플로우가 Idle 상태에 있는데, 이로 인해 발생한 여유 대역폭을 WF^2Q의 경우에는 가장 큰 서비스 웨이트를 갖는 플로우 1이 대부분 사용하게 되나, DCFQ에서는 웨이트가 작은 플로우들이 주로 여유 대역폭을 사용하게 됨을 알 수 있다.

2. 스케줄러의 계산량

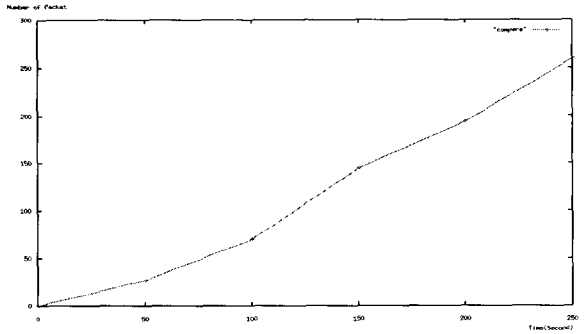
〈표 1〉은 주요 스케줄링 알고리즘에서 스케줄러의 계산량, 즉 Enqueue와 Dequeue 연산에서의 패킷당 Work Complexity[3]를 나타낸 것이다. 플로우의 개

수를 n이라 할 때, WFQ 및 WF^2Q는 모두 $O(\log n)$의 복잡도를 최적의 구현 시에 가지는데, 이는 Enqueue의 경우에는 가상 시간의 계산을 위해 모든 플로우에 대한 정보를 가져야 하며, 또한 Dequeue의 경우에는 모든 플로우를 비교하여 다음 서비스 패킷을 선정하여야 하기 때문이다. 이에 비해, SFQ는 Enqueue시에 $O(1)$, DRR은 언제나 $O(1)$의 복잡도를 갖는다.

〈표 1〉 주요 알고리즘들의 Work Complexity

Algorithm	Enqueue	Dequeue
WFQ[5], WF^2Q[7]	$O(\log n)$	$O(\log n)$
SFQ[6]	$O(1)$	$O(\log n)$
DRR[3]	$O(1)$	$O(1)$
DCFQ	$O(1)$	$O(\log n)$

Fair Queueing 알고리즘의 경우에는 Dequeue 연산은 $O(1)$에 수행될 수 없으며, DCFQ도 Dequeue의 경우는 복잡도가 $O(\log n)$이고, 패킷 당 Work Complexity 상으로는 SFQ와 동일한 성능을 갖는다. 그러나, SFQ의 경우에는 스케줄러가 n 플로우의 모든 패킷들에 대해 가상 시간을 부여하는데 비해 DCFQ에서는 도착하는 패킷들에 대해서 별도의 연산이 없고 각 플로우의 레지스터 관리는 각 플로우가 분산하여 담당하므로, 실제의 계산량은 DCFQ가 훨씬 적다. 따라서, 스케줄러를 고려할 때, DCFQ는 기존의 Fair Queueing 알고리즘들 중에서 가장 적은 계산량을 필요로 한다. 〈그림 6〉은 〈그림 1〉의 시뮬레이션 네트워크 상에서 S1, S2, S3이 1000bytes의 패킷을 전송 할 때 S4에서 수신된 패킷의 차이로 DCFQ 알고리즘을 사용했을 경우에 수신된 전체 패킷에서 WF2Q 알고리즘을 사용했을 경우에 수신된 전체 패킷의 차이를 나타낸다.



〈그림 6〉 DCFQ 알고리즘을 사용했을 경우 수신된 패킷과 WF^2Q 알고리즘을 사용했을 경우에 수신된 패킷의 차이

V. 결론 및 향후 연구 내용

DCFQ는 기존의 Fair Queueing 알고리즘들의 확장성 및 구현성 문제를 해결하기 위해서 패킷의 서비스 순서를 정하기 위한 가상 시간의 관리를 각 플로우에 분산시키는 방법을 사용하였다. 스케줄러는 각 플로우가 제공하는 값에 따라 다음 패킷을 선정하며, 패킷의 도착 시에 필요한 가상 서비스 완료 시간의 연산도 불필요하게 된다. 〈그림 6〉에서는 DCFQ와 WF^2Q간의 Enqueue의 Work Complexity 차이로 인한 수신된 패킷의 차이를 보였으며, 시간이 지날수록 그 양은 점점 증가함을 보였다.

분산하여 처리하는 특성으로 인하여, 여러 개의 Forwarding 엔진을 갖는 대규모의 라우터 설계에 응용될 수 있다. 즉, 라우팅을 마친 패킷들이 스위치를 통하여 큐잉 기능이 없는 인터페이스로 전달되는 경우에는 일반적인 스케줄링 알고리즘이 가정하는 출력 큐잉을 할 수 없으므로, 각 엔진에서 입력 큐잉을 구현할 필요가 있다. 또한 [1,2]와 같은 시스템에 적용 하기 위해서는 mobile node 혹은 sensor node 사이에 스케줄링을 위해 필요한 최소한의 정보 교환을 위한 효율적인 protocol의 개발이 요구된다.

참고문헌

1. A. Knaian "A Wireless Sensor Network for Smart Roadbeds and Intelligent Transportation Systems," Jun. 2000. Available at http://www.dedia.mit.edu/resenv/papers.html
2. H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch and D.Vollmer, "Position-Aware Ad Hoc Wireless Networks for Inter-Vehicle Communications : the Fleetnet Project", *Technical Program of MobiHoc '2001 : the ACM International Symposium on Mobile Ad Hoc Networking & Computing*, Otc. 2001.
3. M. Shreedhar, G. Varghese, "Efficient fair queueing using deficit round robin," *Proceedings of ACM SIGCOMM*, pp. 231-242, Sep. 1995.
4. Planning for Quality of Service, White paper, Cisco Systems, Available at http://www.cisco. com
5. A. Parekh, R. Gallager, "A generalized processor sharing approach to flow control : A single node case," *IEEE/ACM Transactions on Networking*, no. 3, pp.344-357, 1993.
6. S. Golestani, "A self-clocked fair queueing scheme for broadcast applications," *Proceedings of IEEE INFOCOM*, pp.636~646, May 1994.

7. J.C.R. Bennett, H. Zhang, "Worst-case Fair Weighted Fair Queueing," *Proceedings of INFOCOM*, pp. 120-128, 1996.
8. 이준엽 · 이승형, "시스템 가상 시간을 사용하지 않는 간단한 Fair Queueing," 한국통신학회 하계학술대회, pp.1039~1042, 2001
9. The Network Simulator: ns-2, Available at http://www.isi.edu/nsnam