WRR 알고리즘 지원 시스톨릭 구조 가상 출력 큐

조 용 권, 이 문 기, *이 정 희, *이 범 철
연세대학교 전기전자공학과, *한국전자통신연구원
전화 : 02-2134-4731

Systolic Architecture Vitrual Output Queue with Weighted Round Robin Algorithm

Yongkwon Cho, Moonkey Lee, *Junghee Lee, *Bumchul Lee
Dept. of EE, Yonsei University
*ETRI
E-mail : ykcho@yonsei.ac.kr

In the input buffer switch system, VOQ(Virtual Output Queue) archives 100% throughput. The VOQ with the systolic architecture maintains an uniform performance regardless of a number of packet class and output port, so that it doesn't have a limitation of scalability. In spite of these advantages, the systolic architecture VOQ is difficult to change sorting order. In this paper, we proposed a systolic architecture VOQ which support weighted round robin(WRR) algorithm to provide with flow control service .

I 서론

입력 버퍼 구조의 스위치에서 처리율(Throughput) 100%를 위한 가상 출력 큐(Virtual Output Queue:VOQ)의 구현 방안으로는 N개의 FIFO를 이용하는 구조[1], 링크드 리스트를 이용한 FIFO 구조[2][3], 쉬프트 레지스터 구조[4], 선형 시스톨릭 큐[5]를 이용하는 방법이 있다. FIFO 구조와 쉬프트 레지스터 구조는 패킷 클래스의 수와 출력포트 수의 증가에 따라서, 시스템 성능이 떨어지는 단점이 있다.

선형 시스톨릭 큐 구조는 FIFO 구조와 쉬프트 레지스터 구조와 달리 클래스의 수와 출력포트 수에 무관하게 일정한 성능을 서비스를 제공하는 장점을 가지고 있다[5][6].

본 논문에서는 선형 시스톨릭 큐 구조에서 QoS를 위한 스케줄링 방식인 WRR[7]을 지원할 수 있는 WRR 시스톨릭 큐의 알고리즘 및 구조를 제안한다. WRR 시스톨릭 큐는 클래스별 패킷의 출력 비율을 중앙에서 관리하지 않고, 버퍼 블록들이 능동적으로 제어할 수 있는 구조를 갖는다. 버퍼들에 의한 분산 제어는 WRR 방식의 스케줄링이 가능하도록 해주고 클래스 수와 출력포트 수에 시스템의 성능이 영향을 받지 않아 뛰어난 확장성을 가지도록 한다. 2장에서는 선형 시스톨릭 큐의 원리 및 구조에 대해서 알아보고, 3장에서는 WRR 시스톨릭 큐의 알고리즘을 제안한다. 4장에서는 WRR 시스톨릭 큐를 구현하기 위한 구조를 제안하고, 5장에서 결론을 맺는다.

II 선형 시스톨릭 큐 구조

시스톨릭 큐는 내부에 동일한 기능의 버퍼 블록이 연속해서 여러 개 있는 구조로써 패킷의 읽기 쓰기 동작을 일정한 시간 안에 수행 할 수 있는 특징을 가지고 있다. 선형 시스톨릭 큐는 가장 기본적인 구조로 시스톨릭 내부에 패킷 저장 및 정렬을 위한 버퍼 블록들이 전 후에 있는 블록들 사이에서만 제어신호와 저장된 패킷을 전달한다.

기본적인 선형 시스톨릭 큐는 입력되는 패킷을 우선순위에 따라서 정렬하는 기능을 가지고 있는데, Hashemi는 'Single Queue'를 통해서 다중 출력을 지원할 수 있도록 만들었다[6].

시스톨릭 큐 구조를 VOQ에 적용하기 위해서는 큐에 입력되는 패킷에 정렬을 위해 사용될 정보를 덧붙여줘야 한다. 정렬 기준이 되는 정보를 '태그(Tag)'라 하고, 입

력된 패킷에 태그를 붙여주는 블록을 '태깅 유닛(Tagging Unit)'이라 한다. [그림 1]은 선형 시스톨릭 큐 블록을 보여준다. 입력된 패킷은 태깅 유닛을 통해 정렬 기준 정보인 태그가 덧붙여진다. 태그를 가지고 있는 패킷은 정렬을 위해서 시스톨릭 버퍼 블록 시퀀스에 저장이 된다. 시스톨릭 버퍼 블록 시퀀스는 각각의 버퍼 블록 사이에 포워드 패스와 백워드 패스를 통해서 패킷이나 제어 정보를 주고받는다. [그림 1]에서 첫 번째 버퍼 블록은 태깅 유닛에서 저장 할 패킷을 입력받거나 패브릭 스위치로 패킷을 출력하는 역할을 한다. 블록킹은 외부로부터 플로우 제어를 위해 받는 신호다. 블록킹 신호에 따라서 시스톨릭 큐는 패킷 출력을 제어할 수 있다. 블록킹 유닛은 외부 제어 신호에 따라서 시스톨릭 큐의 출력을 블록킹 하는 내부 제어 신호를 발생해주는 역할을 한다.

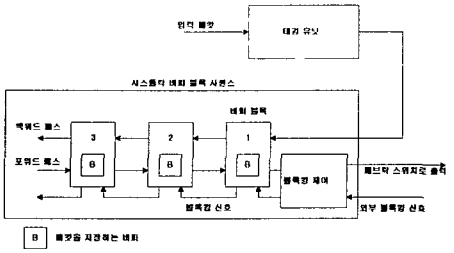

그림 3. 선형 시스톨릭 큐 블록

선형 시스톨릭 큐는 병렬 시스톨릭 큐나 하이브리드 시스톨릭 큐 구조에 비해서 간단한 하드웨어 구조를 가지고 있고, 두개의 패킷이 가지고 있는 태그의 우선 순위를 비교하여 정렬하는 단순한 동작 때문에 버퍼 블록들의 제어도 단순해진다. 제어 회로가 단순해지면 다양한 정렬 방식을 적용하기가 어려워진다. Hashemi는 태깅 유닛의 태그 생성 방법을 이용하여 해결하는 방안을 제시했다. 그러나, WRR을 지원하기 위해서는 태깅 유닛에서 출력포트 별로 어떤 클래스 패킷이 몇 번 출력 될 것인지를 각각 분리해서 관리를 해야 한다. 이 경우 출력포트의 수와 클래스의 수가 증가하면 태깅 유닛의 하드웨어 복잡도가 증가해지는 문제점을 가지고 있다. 따라서, 본 논문에서는 시스톨릭 큐 정렬을 위한 WRR 시스톨릭 큐 알고리즘 및 구조를 제안한다.

Ⅲ WRR 시스톨릭 큐 알고리즘

본 논문에서는 선형 시스톨릭 큐에 WRR 지원을 위한 알고리즘을 제안한다. WRR을 지원하기 위해서 각각의 버퍼 블록들은 태그에 저장되어 있는 가중치와 가중치 증가량 따라서 정렬 할 뿐만 아니라, 가중치 증가량을 변경하는 기능도 수행한다. [그림 2]는 WRR 시스톨릭 큐의 태그를 보여준다. WRR 시스톨릭 큐의 태그는 N 비트의 출력포트 주소(Output ID), W 비트의 패킷 가중치(Weight), I 비트의 가중치 증가량(Increment), F 비트의 패킷 제어 정보(Flag), A 비트의 패킷 저장 메모리 주소(DATA Address)로 구성되었다. 출력포트 주소는 패킷의 목적지 출력포트 주소인데 N 비트를 가지므로 2N 만큼의 출력포트를 지원 할 수 있다. 패킷 가중치와 패킷 가중치 증가량은 패킷이 속해있는 클래스의 대역폭에 따라 해당 패킷의 출력을 스케줄링 하는데 이용된다. 패킷 가중치는 해당 클래스의 대역폭이고, 패킷 가중치 증가량은 버퍼 블록에서 변화되면서 WRR에 따른 스케줄링이 가능하도록 해준다. 패킷 가중치는 W 비트를 가지므로 2W 만큼 다른 종류의 가중치를 패킷에 부여 할 수 있도록 해준다. 패킷 제어 정보는 버퍼가 비어있음을 표시하거나, 그룹 알고리즘에서 새로운 그룹을 형성하기 위해 사용된다.

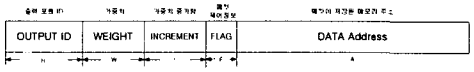

그림 4. WRR 시스톨릭 큐의 태그

선형 시스톨릭 큐에서 WRR을 지원하기 위해서는 각 클래스의 패킷이 몇 개나 큐 안에 저장되어 있는지에 관한 정보가 필요하다. 기존 방식은 태깅 유닛에서 출력포트와 클래스별로 각각의 정보를 관리하여 태그를 생성시킬 때 이 정보를 반영한다. 그런데, 출력포트의 수와 클래스의 수가 증가하면 각각의 정보를 따로 관리해 주는 것이 힘들게 된다. WRR 시스톨릭 큐에서는 패킷 개수의 정보를 시스톨릭 버퍼 블록들이 스스로 관리하도록 만들어 출력포트와 클래스 수 증가에 영향을 받지 않도록 만들어 주었다. [그림 3]은 WRR 시스톨릭 큐에서 가중치 증가량이 변화되는 모습과 WRR 그룹이 생성되는 모습을 보여준다.

새로운 패킷이 들어오면 태깅 유닛은 패킷이 속해있는 클래스에 해당하는 가중치와 가중치 증가량을 이용해 태그를 만든다. 가중치 값은 해당 클래스 패킷의 대역폭에 따라 결정되고 가중치 증가량은 가중치와 동일한 값으로 초기화된다. 시스톨릭 버퍼 블록은 태그에 있는 가중치와 가중치 증가량을 이용해 패킷을 정렬하는데, 저장된 패킷과 전의 버퍼 블록에서 넘어온 패킷의 태그를 비교해서 가중치와 가중치 증가량이 동일할 경우 가중치 증가량을 한 개 올려준다. [그림 4]은 가중치가 '0'인 패킷을 연속해서 시스톨릭 큐에 입력했을 때, 가중치 증가량이 변화되면서 WRR 그룹을 형성하고 패킷이 정렬되는 모습을 가중치와 가중치 증가량을 이용해 보여준다. [그림 3(a)]는 버퍼 블록 시퀀스에 한 개의 패킷이 이미 저장되어 있는 상태에서 다시 새로운 패킷을 입력한 경우인데, 저장된 패킷과 입력된 패킷의 가중치와 가중치 증가량이 같기 때문에 가중치 증가량을 하나 올려준다. [그림 3(b)]는 가중치 증가량이 증가된 패킷은 두 번째 버퍼 블록에 저장되고 다시 새로운 입력이 들어온 모습이다. [그림 3(c)]는 세 번째로 입력된 패킷이 두 번째 버퍼 블록에 저장된 패킷과 다시 경쟁한 후 가중치 증가량이 다시 변하는 모습이다.

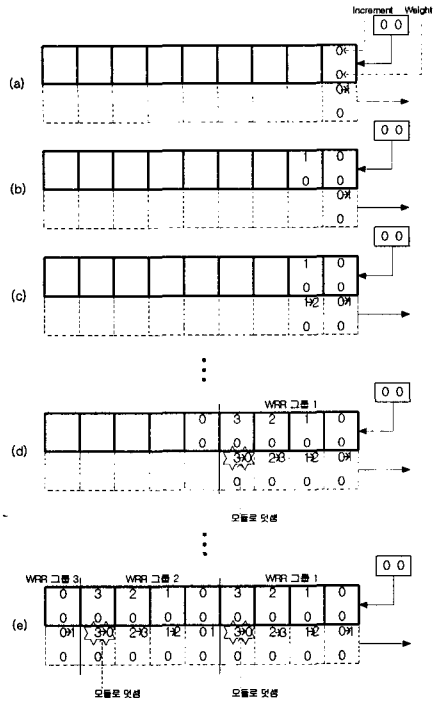

그림 5. WRR 시스톨릭 큐에서 가중치 증가량의 변화와 WRR 그룹 생성

WRR 시스톨릭 큐에서 패킷 정렬을 위해 가중치 증가량을 계속해서 올려주는 것은 비효율적이기 때문에 가중치 증가량의 값의 올려주는 것은 모듈로 덧셈 연산을 하도록 만들었다. [그림 3(d)]는 모듈로 덧셈을 이용해 가중치 증가량을 증가시키는 모습이다. [그림 3]은 가중치 증가량 증가는 모듈로-4 덧셈을 이용하는 것을 예로 들었기 때문에 가중치 증가량이 '3'이후에 증가되면 모듈로 덧셈에 의해서 '0'이 된다. 가중치가 '0'이 아닌 클래스의 패킷은 모듈로 덧셈에 의해 가중치 증가량이 처음 값이 될 때는 '0' 대신 가중치와 동일한 값으로 초기화 시켜준다. 가중치 증가량을 태그 생성 때와 동일하게 '0'이 아닌 가중치로 초기화를 시켜주는 이유는 새로운 WRR 그룹에서 생성을 위해서다.

WRR 시스톨릭 큐에서는 모듈로 연산에 의해 가중치 증가량이 초기 값으로 변화된 패킷이 저장되기 전까지를 WRR 그룹으로 정의한다. [그림 3(d)]에서는 모듈로 덧셈에 의해 가중치 증가량이 '0'이 된 패킷이 저장되기 전까지인 4번째 버퍼 블록까지가 한 개의 WRR 그룹을 형성한다. [그림 3(e)]는 모듈로 덧셈에 의해 WRR 그룹들이 생성된 모습이다.

가중치 증가량을 버퍼 블록에서 증가 시켜주는 동작은 가중치가 서로 다른 여러 클래스의 패킷을 서로 섞어 놓았을 때 WRR 규칙으로 서로 정렬 되도록 해준다.

여러 클래스의 패킷들이 서로 섞여 있을 때는 모듈로 덧셈에 의해 새로운 WRR 그룹을 생성하는데 문제점이 있다. 모듈로 덧셈에 의해 가중치 증가량이 초기 값으로 변해도 현재 머물러 있는 WRR 그룹을 벗어나 새로운 WRR 그룹으로 들어갈 때까지는 초기화 된 가중치 증가량으로 패킷을 정렬하면 안 된다. 이를 위해서 태그의 제어 플래그를 이용한다. 모듈로 덧셈에 의해 가중치 증가량이 초기화되면 플래그에 LOCK 정보를 표시했다가 새로운 WRR 그룹으로 들어가면 LOCK 정보를 풀어준다. 플래그에 LOCK 정보를 가지고 있는 패킷은 정렬되지 않고 뒤의 버퍼 블록으로 전달되기만 한다. 패킷이 새로운 WRR 그룹으로 들어간 것은 비어있는 버퍼를 만나거나, 가중치 증가량이 변하거나, WRR 그룹의 마지막에 해당되는 패킷을 지난 경우를 확인하면 알 수 있다.

WRR 그룹은 가중치에 따라서 패킷 출력 수를 조정해 대역폭을 제어하는데 이용된다. 패킷 출력 수 제어 기준을 가중치가 가장 높은 클래스로 정의하면 한 개의 WRR 그룹 내에는 기준이 되는 클래스의 패킷이 태그의 가중치 증가량 비트 수에 I에 따라 2I 개만큼 저장된다. 가중치가 '0' 인 클래스 패킷은 가중치 증가량이 '0'에서 시작해서 2I-1까지 증가 될 수 있기 때문이다. 가중치가 WEIGHT인 클래스에 속한 패킷은 한 개의 WRR 그룹 내에 2I-WEIGHT개 저장된다. 따라서, 가중치 '0'인 클래스가 2I개 출력될 때 가중치 WEIGHT인 클래스는 2I-WEIGHT개 출력된다.

지금 까지 WRR 시스톨릭 큐에서 패킷을 정렬하는 것은 패킷을 출력되지 않는 상황을 가정한 것이다. 그런데, 큐에서 패킷이 출력되면 WRR 그룹의 일부분만이 남아있게 된다. 이 때, 새로 입력되는 패킷의 가중치 증가량을 초기 값으로 설정하면 WRR 시스톨릭 큐는 일부분 만 남아있는 WRR 그룹의 부족한 패킷을 다시 채워 업어진 부분에 대해서 다시 그룹을 다시 형성한다. 이것을 방지하기 위해서 태깅 유닛에서는 입력할 패킷의 가중치 증가량을 가중치를 이용해 초기화하지 않고, 출력 대기 중인 패킷보다 입력되는 패킷이 낮은 정렬 순서를 갖도록 만든다. 입력될 패킷의 가중치가 출력 대기중인 패킷의 가중치보다 작으면(WEIGHT 값이 큰 경우) 입력될 패킷의 가중치 증가량을 출력 대기 패킷과 같게 만들고, 같거나 크면(WEIGHT 값이 같거나 작은 경우) 가중치 증가량이 출력 대기 패킷보다 하나 크도록 만든다. 이러한 방법으로 새로 들어오는 패킷에 의해 출력되고 있는 WRR 그룹이 재생되는 것을 방지한다.

Ⅳ WRR 시스톨릭 큐 구조

WRR 시스톨릭 큐는 일반적인 선형 시스톨릭 큐의 기본 구조와 동일하다. 선형 시스톨릭 큐는 버퍼 블록들이 연속해서 연결되어있는 단순한 구조로 되어있기 때문에 제어가 간단하고 확장성이 뛰어난 장점을 가지고 있다. WRR 시스톨릭 큐는 선형 시스톨릭 큐의 버퍼 블록에 WRR지원을 위해 가중치 증가량을 처리하기 위한 모듈로 덧셈기를 추가한 구조이기 때문에, 제어가 단순하고 뛰어난 확장성을 유지한다. 시스톨릭 큐는 버퍼 블록은 레지스터로 구성되어 있기 때문에 범용 메모리를 기반으로 하는 시스템에 비해서 하드웨어가 커지는 문제점이 있다. 또, 동일한 버퍼 블록이 반복되는 구조를 가지고 있기 때문에 IP 기반의 가변 길이 패킷을 처리하기가 어려워진다. 이 문제는 실제 패킷을 저장하는 곳은 범용 메모리를 사용하고 VOQ를 위한 큐 관리 블록은 시스톨릭 큐로 분리하여 해결한다. [그림 4]는 패킷 저장을 위한 범용 메모리 블록과 VOQ를 위한 WRR 시스톨릭 큐의 구조를 보여준다.

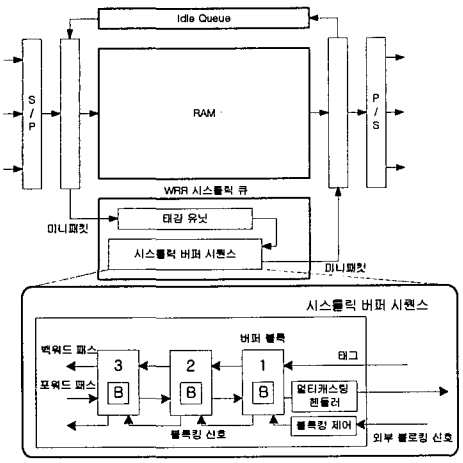

그림 6. WRR 시스톨릭 큐 기반 VOQ의 구조

VOQ에 입력되는 패킷은 범용 메모리에 저장되고 패킷의 클래스 정보와 메모리 주소로 구성된 미니 패킷을 만들어 WRR 시스톨릭 큐에 저장한다. VOQ의 출력은 WRR 시스톨릭 큐에서 미니 패킷이 출력되면 메모리 주소를 이용해 메모리에 저장된 패킷을 내보낸다.

블록킹 제어 블록은 VOQ의 플로우 제어(Flow Control)를 위한 것이고, 멀티캐스팅 헨들러(Multicasting Handler)는 멀티캐스팅 알고리즘에 따라서 큐의 출력 패킷을 제어하기 위한 블록이다.

WRR 시스톨릭 큐의 버퍼 블록에서 레지스터는 미니 패킷 저장을 위한 저장 레지스터와 앞의 버퍼 블록에서 전달된 미니 블록을 저장하기 위한 임시 레지스터로 구성 되어있다. 두 개의 미니 패킷의 태그를 비교하여 저장 순위가 낮은 미니 패킷은 임시 레지스터에 저장되었다가 백워드 패스를 통해 다음 버퍼 블록으로 전달되고, 저장 순위가 높은 미니 패킷은 저장 레지스터에 있다가 포워드 패스를 통해 앞의 버퍼 블록으로 전달된다. WRR 시스톨릭 큐의 버퍼 블록의 특징은 WRR 따른 정렬 방식을 지원하기 위해 내부에 모듈로 덧셈을 사용한다는 것이다. 모듈로 덧셈은 미니 패킷의 태그에서 가중치 증가량을 알고리즘에 따라 증가시키기 위한 블록이다.

V 결론

본 논문에서는 선형 시스톨릭 큐 방식에 WRR(Weighted Round Robin)을 지원하는 WRR 시스톨릭 큐의 알고리즘과 구조를 제안한다. WRR 시스톨릭 큐는 선형 시스톨릭 큐 방식의 장점인 간단한 구조와 뛰어난 확장성을 가지고 있으면서 WRR을 지원할 수 있는 특징을 가지고 있다. 일반적인 선형 시스톨릭 큐에서 WRR을 지원하기 위해서는 각각의 출력포트와 클래스별로 현재 저장되어 있는 패킷의 수를 관리하는 방식의 복잡한 태그생성 블록을 사용해야 하지만, WRR 시스톨릭 큐는 저장 패킷 수를 따로 관리하지 않고 내부 시스톨릭 버퍼 블록에서 자동으로 관리되기 때문에 복잡도를 줄이고, 출력포트의 수와 클래스의 종류의 증가에 시스템 성능이 영향을 받지 않는 확장성을 갖는다.

참고문헌

[1] J. Chao, "A Novel Architecture for Queue Management in the ATM Network," IEEE JSAC," vol. 9, no. 7, pp. 1110-1118, Sep. 1991.

[2] G. J. Jeong, J. H. Lee, B. C. Lee, ,"Design of pipelined routing engine for input-queued ATM switches," Electronics Letters , Vol. 37 Iss. 2, Jan. 2001, pp.137 -138

[3] G. J. Jeong and M. K. Lee, "A Scalable Pipelined Memory Architecture for Fast ATM Packet Switching," IEICE Trans. Fundamentals on Electronics, Commuications, and Computer Sciences, vol. E82-A, no. 9, pp. 1937-1944, Sep. 1999.

[4] J. Chao and N. Uzun, "A VLSI Sequencer Chip for ATM Traffic Shaper and Queue Management," IEEE J. Solid-State Circuits, vol. 27, no. 11, pp. 1,634-1,643, Nov. 1992.

[5] S. W. Moon, J. Rexford, and K. G. Shin, " Scalable Hardware Priority Queue Architectures for High-Speed Packet Switches" , IEEE Trans. on Comp., VOL. 49, NO. 11, pp. 1215-1227, Nov. 2000

[6] Hashemi, M.R.; Leon-Garcia, A.; "The single-queue switch: a building block for switches with programmable scheduling,", IEEE JSAC , Vol. 15 Iss. 5, June 1997, pp.785 -794

[7] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip," IEEE JSAC, Vol. 9, No. 8. pp. 1265-1279, Oct. 1991.