고성능 Queue 관리를 위한 NDRR 알고리즘

김지훈 민경주 권택근
충남대학교
{jiphung, first, tgkwon}@cnu.ac.kr

NDRR Algorithm for High Performance Queue Management

Ji-Hoon Kim, Kyoungju Min, Taeck-Geun Kwon
Chungnam Nat'l University

요 약

라우터는 여러 곳에서 들어오는 패킷들을 빠르게 전달하는 기능을 담당하는 네트워크 장비로서, 들어오는 패킷들이 공평하게 서비스 받을 수 있도록 큐 관리 알고리즘을 사용한다. 그런데 대부분의 라우터들은 HOL 블록킹 문제 때문에 버퍼를 입력 포트 쪽이 아닌 가상적으로 출력 포트 쪽에 정의하는 VOQ로 구현을 하였고, 패킷들이 공평하게 서비스 받기 위해 DRR 알고리즘으로 구현하는 경향이 있다. 이 논문에서는 기존의 DRR 알고리즘에서 패킷 서비스를 위한 경직된 조건에 유연성을 주어 기존의 DRR 알고리즘의 복잡도와 공평성을 유지하는 한편 패킷 서비스 성능을 높여주는 NDRR 알고리즘을 제안한다.

1. 서론

현재 인터넷은 전 세계에 복잡하게 연결이 되어 있고 그 연결로 어떤 곳이든 정보를 주고받을 수가 있다. 이 연결들의 상위에는 라우터라는 네트워크와 네트워크를 연결하는 장비가 존재하는데, 이 라우터가 전 세계를 연결하는데 큰 역할을 하게 된다. 이러한 라우터가 연결을 빠르고 공평하게 서비스할 수 있도록 연구가 되었고 현재도 많은 연구가 진행 중이다. 라우터는 빠르고 공평한 서비스 제공을 위해 입력/출력 큐를 관리하는 여러 알고리즘들이 사용되는데 가장 대표적인 알고리즘이 DRR(Deficit Round Robin) 알고리즘이다. 이런 DRR 알고리즘은 각 입력포트에 대해서 할당량을 가지고 그 할당량만큼만 서비스를 하기 때문에 라우터에서 공평성을 제공할 수 있다. 그러나 이러한 DRR 알고리즘에서 할당량이라는 것이 특수한 환경에서 공평성을 지켜주기 위해 성능에 악영향을 줄 수 있다. 그래서 이 논문은 DRR 알고리즘의 문제점을 살펴보고, 이를 개선 할 수 있는 알고리즘을 제안하고, 기존 DRR 알고리즘의 성능을 비교한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 라우터의 큐 관리 알고리즘에 대해 기술하고, 3장에서는 보다 나은 성능 향상을 위한 새로운 알고리즘을 제안한다. 성능 평가 및 결과를 4장에서 기술하며 마지막으로 5장에서는 결론을 정리한다.

2. 기존 라우터의 큐 관리 알고리즘

2.1 VOQ(Virtual Output Queues)

라우터에는 일정한 곳으로 서비스하기 위한 정책이 존재한다. 그리고 패킷이 라우터로 들어오게 되면 정책을 적용해야 하는데 정책을 적용하기 위해 일정 시간동안 라우터 안에서 저장해야 한다. 그것이 바로 버퍼(Buffer)이다. 또한 라우터에는 패킷이 들어오는 입력포트와 라우터 안에서 일정한 정책을 적용시켜서 외부로 서비스가 되는 출력포트가 존재하게 된다. 그런데 이 버퍼는 한정된 자원이기 때문에 버퍼를 지정할 곳에 대해서 정의할 필요가 있다. 일단 라우터에 들어오게 될 패킷에 정책을 적용시키기 위해 버퍼의 위치를 입력포트에 버퍼를 정의하는 것을 생각할 수 있다. 그런데 이 방법에는 가장 큰 단점은 2개 이상의 여러 개의 입력포트에서 하나의 출력포트만 서비스를 하려고 할 때 여러 입력포트 중에 하나의 입력포트만 라우터 밖으로 서비스를 받게 된다. 이 상태에서 라우터 밖으로 서비스 되는 입력포트를 제외한 다른 여러 입력포트들은 기다리게 된다. 다른 여러 입력포트 측면에서 생각해보면 각 입력포트에서 하나의 출력포트로 서비스를 받으려는 패킷 때문에 뒤에 있는 패킷 또한 기다리게 된다.

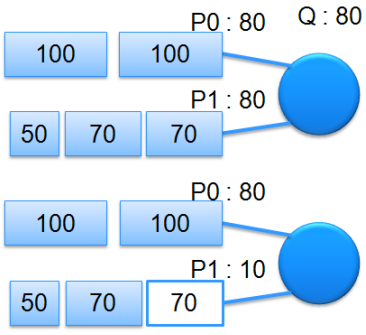
이러한 문제를 HOL(Head of Line) 블록킹이라 하고 라우터의 전체적인 성능의 60% 밖에 못 내는 악영향을 준다[1]. 그래서 이 문제점을 해결하기 위해 VOQ라는 출력포트 쪽에 가상의 큐를 정의하고 PIM[2]과 iSLIP[3]와 같은 셀 단위의 스케줄링 알고리즘을 이용해

기존에 비해 향상된 성능을 끌어낼 수 있게 되었다.[4].

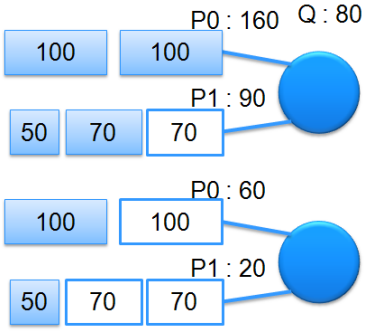
2.2 DRR 알고리즘[5]

VOQ와 셀 단위 스케줄링으로 아주 높은 성능의 라우팅 알고리즘 구현을 구현하게 되었다. 그렇지만 각 포트 별로 공평하게 서비스를 받지(fairness) 못하는 문제가 발생하였다. 가장 간단한 선입선출(FIFO : First In First Out) 방식과 라운드 로빈 방식으로 공평성을 구현 할 수 있지만 그 공평성은 상당히 낮은 수준의 공평성이다. 이것을 해결하기 위해 DRR 알고리즘이 고안이 되었다. 각 입력포트에 적립액(Deficit)을 정하고 그 적립액만큼 패킷을 출력포트로 보내도록 해서 공평성이 높은 라우터를 구현할 수 있다.

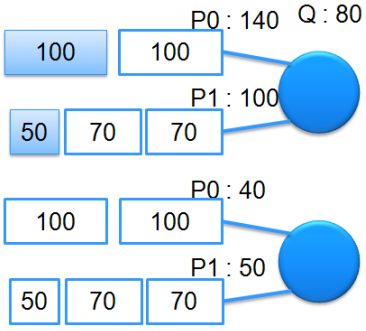
그림1에서는 2개의 포트에 대해서 DRR 알고리즘으로 패킷 스케줄링을 수행이 될 것이며 첫 번째 포트(P0)에는 100바이트 패킷이 두 개가 버퍼에 존재하고 두 번째 포트(P1)에는 70바이트 패킷 두 개와 50바이트 패킷 하나가 버퍼에 존재한다. 그리고 적립되는 할당량은 80바이트로 정해진 예시이며 각 라운드마다 80바이트의 할당량을 받게 된다. DRR 알고리즘에서는 기본적으로 라운드 로빈 방식으로 라운드가 존재하며 라운드 안에서는 순서대로 각 포트가 선택이 된다. 그림1 (a)에서 순서대로 P0와 P1으로 선택이 되는데 P0에서는 가장 앞에 존재 하는 패킷의 크기가 100바이트가 되고 할당량의 크기가 80바이트로 할당량의 크기가 더 작기 때문에 패킷이 서비스가 되지 않고 다른 포트로 넘어가게 된다. 그리고 P1에서는 패킷의 크기가 70바이트로 할당량이 더 크기 때문에 패킷이 서비스가 되고 할당량은 70바이트가 서비스가 되었기 때문에 할당량과 패킷의 차인 10바이트(80-70바이트)가 되며 모든 포트에 대해서 수행이 되었기 때문에 다음 라운드인 그림1 (b)로 넘어가게 된다. 다시 P0으로 선택이 되는데 새로운 라운드이기 때문에 할당량을 받아서 160바이트(80+80바이트)가 된다. 그리고 패킷의 크기가 100바이트로 할당량의 크기가 더 크기 때문에 패킷이 서비스가 되고 할당량은 60바이트(160-100바이트)가 되며 P1으로 넘어가게 된다. P1에서도 새로운 라운드이기 때문에 할당량을 받아서 90바이트(10+80바이트)가 되며 패킷의 크기가 70바이트로 할당량의 크기가 더 크기 때문에 패킷이 서비스가 되며 할당량은 20바이트(90-70바이트)가 된다. 그리고 모든 포트에 대해서 수행이 되었기 때문에 다시 새로운 라운드인 그림1 (c)로 넘어가게 된다. 또 다시 순서대로 포트가 선택되기 때문에 P0가 먼저 선택이 되며 새로운 라운드이기 때문에 할당량을 다시 받게 되어서 140바이트(60+80바이트)가 된다. 그리고 패킷의 크기가 100바이트로 할당량의 크기가 더 크기 때문에 패킷이 서비스가 된다. 그리고 다음 포트인 P1으로 넘어가게 되며 P1에서 또한 새로운 라운드이기 때문에 할당량을 받아 100바이트(20+80바이트)가 되며 패킷의 크기가 50바이트로 할당량의 크기가 더 크기 때문에 패킷이 서비스가 된다.



(a) 첫 번째 라운드



(b) 두 번째 라운드



(c) 세 번째 라운드

그림1. DDR 알고리즘의 예시

이와 같은 방식으로 DRR 알고리즘에서는 각 포트의 공평성을 지켜주면서 패킷 스케줄링이 수행이 되며 각 포트마다 한 번의 계산(할당량과 패킷의 크기를 비교)을 해서 패킷을 서비스하기 때문에 복잡도는 O(1)을 기대할 수 있다.

3. NDRR(Negative Deficit Round Robin) 알고리즘의 제안

3.1 기존 DRR 알고리즘의 문제

라우터에서는 가변적인 패킷에 대해서 서비스를 한다. 그렇기 때문에 이런 할당량이 정해져있는 DRR 알고리즘에서는 패킷의 크기가 할당량의 크기보다 큰 패킷이 많이 흘러 다니는 라우터에서는 문제가 발생할 수 있다. 기존의 DRR 알고리즘에서는 미리 정해진 할당량 보다 크기가 큰 패킷들이 입력포트에 들어오게 되면 패킷보다 많은 할당량을 받기 위해 다음 라운드로 가게 되고 많은 할당량을 받기 위해 소모되는 라운드는 전체적인 라우터의 성능저하를 가져 올 것이다. 만약에 모든 입력포트에 할당량보다 큰 패킷들이 들어오게 된다면 모든 입력포트의 패킷이 서비스되지 않고 많은 할당량을 받기 위해 라운드가 소모 될 것이며 짧은 시간지만 일정 시간에 라우터가 패킷을 내보지 못하는 정지된 상태(idle)가 되어 버린다. 이렇게 되어 버리면 전체적인 라우터의 성능에서는 큰 악영향을 미치게 될 것이다.

3.2 제안하는 알고리즘

위 문제의 원인은 패킷을 할당량에 따라서 서비스 되는 정책의 경직성을 들 수가 있다. 기존의 DRR 알고리즘에서는 패킷을 서비스하는 조건은 할당량보다 작은 패킷을 가지고 있는 경우로 제한되어있다. 그렇기 때문에 입력포트에 할당량보다 큰 패킷이 연속으로 들어 올 경우에는 첫 번째 할당량보다 큰 패킷이 할당량을 쓰게 되고 그 값이 0에 가까운 값으로 가게 될 것이며 다음에 들어온 패킷을 서비스하기 위해 할당량을 채우게 될 것인데 할당량보다 큰 패킷이기 때문에 조건 검사만 수행하고 실제 패킷 서비스를 하지 않고 라운드가 소비된다.

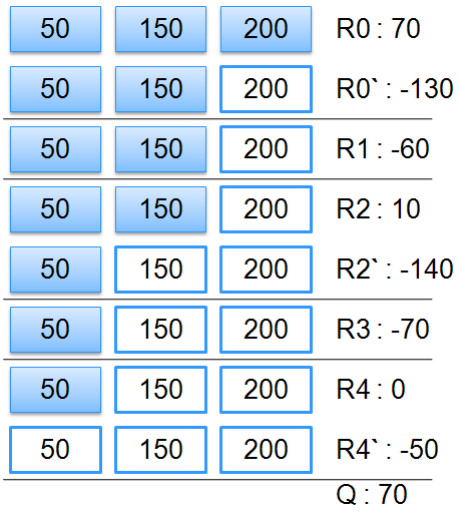
이런 점을 보완하기 위해 기존의 DRR 알고리즘에서 패킷을 서비스하는 정책을 더 유연하게 만들었다. 일단 패킷을 서비스하는 조건인 “할당량이 패킷크기보다 크거나 같다” 라는 제한된 조건을, 할당량이 패킷크기에 의존적이지 않고 “할당량이 0바이트보다 크거나 같다” 로 조건을 완화하였다. 단 서비스를 받으면 할당량을 패킷의 크기를 빼기 때문에 그에 따라서 할당량이 0바이트보다 큰 값을 가지는 것을 0바이트보다 작을 수 있는 음수도 갈 수 있도록 하였다. 이렇게 함으로써 연속되는 할당량보다 큰 패킷이 들어 올 경우에도 기존의 아무것도 하지 않는 라운드를 줄일 수가 있어서 이에 따라서 발생하는 성능저하를 막을 수 있다.

3.3 DRR 알고리즘과 NDRR 알고리즘의 비교

그림5의 (a),(b)는 DRR 알고리즘과 NDRR 알고리즘의 서로의 차이점을 보여주고 있다. DRR 알고리즘과 NDRR 알고리즘에는 큐에 200바이트, 150바이트, 50바이트의 패킷이 입력된 상태에서 각 라운드 Rn 별 패킷의 상태에 대해서 비교를 하였다.

기본적으로 DRR 알고리즘과 NDRR 알고리즘은 같은 방식으로 할당량이 받게 되며 라운드 로빈 방식으로 운영이 된다. 할당량은 70바이트로 결정을 하였다.

(a) DRR 알고리즘



(b) NDRR 알고리즘

그림2. DRR 알고리즘과 NDRR 알고리즘의 비교

DRR 알고리즘에서는 큐에 들어 있는 패킷이 200바이트로 할당량이 패킷 크기보다 작기 때문에 서비스가 되지 않으며 다음 라운드로 넘어 가게 된다. 그리고 다음 라운드인 R1에서 새로운 라운드이기 때문에 할당량이 140바이트(70+70바이트)로 누적이 되나 패킷의 크기가 200바이트이기 때문에 패킷이 서비스가 되지 않는다. 그

리고 다음 라운드인 R2에서는 할당량의 크기가 210바이트(140+70바이트)가 되며 패킷이 크기가 200바이트이기 때문에 R2'에서 패킷이 서비스가 된다. 다음 라운드인 R3로 가게 되면 할당량이 80바이트(10+70바이트)가 되나 큐에 들어 있는 패킷이 150바이트이기 때문에 패킷은 서비스 되지 않으며 다음 라운드로 넘어 가게 된다. R4에서는 할당량이 150바이트(80+70바이트)가 되며 패킷의 크기와 같기 때문에 서비스가 되게 된다. 그리고 R5에서 할당량의 크기가 70바이트(0+70바이트)이며 패킷의 크기가 50바이트로 더 작기 때문에 패킷이 서비스가 된다.

그러나 NDRR 알고리즘에서는 약간 다른 현상을 가지게 된다. 먼저 발견할 수 있는 현상은 패킷이 나가는 시점이다. DRR 알고리즘에서는 R2와 R4 그리고 R5에서 패킷이 서비스가 되지만 NDRR 알고리즘에서는 R0, R2, R4에서 패킷이 서비스가 된다. R0에서는 할당량의 크기가 패킷 크기보다 작지만 0바이트보다 크거나 같은 값을 가지기 때문에 패킷을 서비스하게 된다. 또한 R2와 R4에서도 모두 할당량의 크기가 패킷의 크기보다 작지만 0바이트보다 크거나 같은 값을 가지기 때문에 패킷이 서비스 된다. 이렇게 할당량이 0바이트보다 크거나 같은 값을 가질 때 서비스가 되기 때문에 DRR 알고리즘에서는 할당량이 언제나 0바이트보다 크거나 같은 0바이트과 양수를 가지지만 NDRR 알고리즘에서는 0바이트과 양수 그리고 음수까지 모두 가질 수가 있다. 이렇게 동작함으로써 DRR 알고리즘은 5 라운드 만에 모든 패킷을 서비스를 하지만 NDRR 알고리즘에서는 4 라운드 만에 패킷을 모두 서비스를 할 수 있어서 같은 패킷 수를 처리를 하지만 다 짧은 라운드에 서비스를 할 수 있다.

4. 성능 평가 및 결과

DRR 알고리즘과 NDRR 알고리즘의 성능 평가를 위해서 C언어를 이용해서 가상의 라우터를 구현을 하였다. 가상의 라우터를 구현하기 위해서 인터넷 트래픽의 특성이 필요하였다. 인터넷 트래픽의 특성 중에 패킷의 크기가 DRR 알고리즘과 NDRR의 필요한 변수로 쓰이게 된다. 인터넷 트래픽에서 패킷의 크기는 40바이트가 전체의 약 40%를 차지하고 576바이트가 전체의 약 30%, 1500바이트가 전체의 약 20%를 차지하고 있다고 한다[6]. 이렇게 정의함으로써 현실의 라우터와 최대한 비슷한 환경이 되도록 구성을 하였다. 또한 가상의 라우터에서는 입력포트가 10개부터 100개까지 가지는 라우터로 정의를 했으며 큐에 대기 중인 패킷의 수도 10개부터 100까지 정의를 하였다. 또한 DRR 알고리즘과 NDDR 알고리즘의 변수인 할당량은 64바이트를 할당량으로 정하였다. 그리고 두 알고리즘의 성능 비교는 평균 패킷의 처리 시간과 각 라운드에서 각 포트의 큐 길이들의 총합을 비교 하였다. 평균 패킷의 처리 시간이 짧을수록 해당하는 알고리즘이 같은 양의 패킷을 더 짧은 시간에 서비스한다는 것을 의미하며 각 라운드에서 각 라운드에서 각 포트의 평균 큐 길이는 그 총합이 작을수록 큐 안에서 대기하는 시간이 짧다는 것을 의미하기 때문에 두 알고리즘을 비교하였다. 그리고 성능의 비교는 DRR 알고리즘을 1로 가정했을 경우의 NDRR 알고리즘의 성능을 비교하였다.

표1. DRR 알고리즘과 NDRR 알고리즘에서 평균 패킷의 처리 시간의 비교

포트의 수	큐에 대기 중인 패킷 수	DRR	NDRR
10	10	1	0.931
	20	1	0.953
	50	1	0.958
	100	1	0.997
20	10	1	0.978
	20	1	0.948
	50	1	0.954
	100	1	0.974
50	10	1	0.926
	20	1	0.938
	50	1	0.971
	100	1	0.990
100	10	1	0.862
	20	1	0.916
	50	1	0.971
	100	1	0.985

표1에서는 두 알고리즘의 평균 패킷 처리 시간은 DRR 알고리즘을 1로 가정 했을 때 NDRR이 최대 0.862에서 최소 0.997의 성능을 가지게 되는 것을 보여주며 큐에 대기 중인 패킷 수가 작을수록 평균 패킷 처리 시간에서 더 좋은 성능을 가지게 되는 것을 알 수 있다. 그리고 표2에서는 각 라운드에서 각 포트의 평균 큐 길이의 비교는 DRR 알고리즘의 성능을 1로 가정했을 때 NDRR 알고리즘은 최대 0.652에서 최소 0.960의 성능을 가지게

되는 것을 알 수 있으며 또한 큐에 대기 중인 패킷의 수가 작을수록 NDRR 알고리즘이 DRR 알고리즘에 비해서 더 좋은 성능을 내는 것을 알 수 있다.

표1과 표2에서 전반적으로 NDRR이 DRR보다 포트 수와 큐에 대기 중인 패킷의 수에 따라 전반적으로 성능이 더 좋은 것을 알 수 있다.

표2. DRR 알고리즘과 NDRR 알고리즘에서 각 라운드에서 각 포트의 평균 큐 길이의 비교

포트의 수	큐에 대기 중인 패킷 수	DRR	NDRR
10	10	1	0.652
	20	1	0.809
	50	1	0.917
	100	1	0.959
20	10	1	0.662
	20	1	0.815
	50	1	0.922
	100	1	0.960
50	10	1	0.685
	20	1	0.819
	50	1	0.923
	100	1	0.960
100	10	1	0.672
	20	1	0.814
	50	1	0.922
	100	1	0.960

5. 결론

기존의 DRR 알고리즘에서는 패킷을 서비스하는 조건이 할당량이 패킷크기보다 크거나 같을 경우에만 패킷을 서비스를 하지만 NDRR 알고리즘에서는 패킷의 크기와는 상관없이 할당량이 0바이트보다 크거나 같으면 서비스 한다는 DRR 알고리즘보다 완화된 조건을 이용해서 기존의 DRR 알고리즘의 성능이 라우터의 포트의 수와 큐에 대기 중인 패킷의 수에 상관없이 좋은 성능을 가지는 것으로 알 수 있다.

그러나 인터넷 트래픽 특성을 최대한 반영한 가상의 라우터를 이용해서 제안된 알고리즘을 구현하고 성능 평가를 하였기 때문에 어느 정도 한계를 가지게 된다. 그렇기 때문에 향후 연구과제로 이렇게 제안된 NDRR 알고리즘을 C언어상의 가상 라우터가 아닌 실제 사용되는 라우터에 반영을 해서 성능 평가 과정이 필요하다. 또한 DRR 알고리즘은 오래된 알고리즘이기 때문에 오늘날의 인터넷 환경과는 어느 정도 맞지 않는 부분이 존재 한다. 그래서 이런 DRR 알고리즘이 맞지 않지만 NDRR 알고리즘이 적합한 곳을 찾아내고 그 환경에 최적화된 알고리즘이 되도록 구현하는 과정이 필요하다.

6. 참고문헌

[1] V. Puente, J.A. Gregorio, C. Izu and R. Beivide, “Impact of the Head-of-Line Blocking on Parallel Computer Networks: Hardware to Applications,“ EURO-PAR’99 Parallel Processing. Toulouse, France. August 1999 pp. 1222-1230

[2] Hyoung-Il Lee and Seung-Woo Seo, “Analysis model of multiple input-queued switches with PIM scheduling algorithm,“ IEEE Communication Letters, Vol.5 No.7, pp.316 -318, July 2001

[3] Nick McKeown, “The iSLIP scheduling algorithm for. input-queued switches,“ IEEE Trans. on Networking,. 7(1999)2, 188-201.

[4] S.H.Moon and D.K.Sung, “Variable Length Packet Scheduling Algorithm for IP Traffic,“ JCCI 2001, April 2001.

[5] M. Shreedhar and G. Varghese, “Efficient fair queueing using deficit round-robin,” IEEE/ACM Trans. Networking, vol. 4, no. 3, pp. 375-385, Jun. 1996.

[6] Sprint Applied Research Group “http://ipmon.sprint.com/packstat/packetoverview.php“