□ 기술해설 □

이동 네트워킹 기술

서울대학교 임효준·김응도·김종권*

1. 서 론

이동 컴퓨팅은 최근 빠르게 성장하는 통신 시장이다. 고속의 무선 LAN 상품이 등장해 사용되고 있으며 WAN에서도 저속의 무선 서비스가 가능하다. 한편 다양한 형태의 망을 연결한 인터넷은 계속 팽창해 나가고 있으며 앞으로 이동하는 인터넷 호스트들이 증가할 전망이다. 인터넷에서 이러한 이동 컴퓨팅을 수용하기 위해서는 이동성에 의해 발생하는 문제들을 해결해야 한다. 이동 컴퓨팅 환경에서 새롭게 발생하는 문제들로 다음과 같은 것들을 들 수 있다.

먼저 이동 호스트들은 망에서의 접속 지점을 수시로 바꾸게 되므로 임의의 장소에 있는 이동 호스트를 찾아내고 통신 중에도 역동적으로 이동하는 호스트와의 연결을 유지하는 호스트 이동성을 지원하는 것이 필요하다. 호스트의 이동성 지원은 주로 망 계층에서 이루어지게 되며 이동하는 목적 호스트까지 메시지를 제대로 전달하는 데에 중점을 두게 된다.

이동 호스트는 컴퓨팅 능력이 고정 호스트보다 떨어진다. 이동 호스트에서 사용하는 메모리와 디스크는 소형, 고효율성을 요구하므로 일반적으로 고정 호스트의 그것보다 가격이 높을 것이며 충분한 전력을 공급하는 데에도 어려움을 겪게 된다. 따라서 이동성 지원을 위한 기능들은 이동 호스트에 큰 부담을 주지 않아야 한다.

이동 네트워킹에서 통신중 역동적 이동성을 지원하기 위해서 이동 호스트는 무선 라인을 통해 통신망과 연결될 것이다. 무선 환경은 유선보다 높은 에러율과 지연을 가지게 된다. 따라서 무선 라인에서의 높은 에러율과 지연으로 인한 성능 하락을 막는 새로운 방법이 필요하게 된다.

이동성을 지원하는 메커니즘은 링크 계층, 망 계층, 전송 계층 등에서 구현될 수 있다. 링크 계층에서의 이동성 지원은 주로 무선 라인의 높은 에러율과 지연을 보상하는 방법들이다. 대표적인 방법은 FEC(Forward Error Control)와 결합하여 링크 계층에서 재전송 프로토콜을 구축하는 방법이다[1]. 그러나 전송 계층과 데이타 링크 계층이 독자적으로 에러 제어 및 복구 기법을 실행할 경우 자칫 성능 저하를 초래할 수 있다[2]. 가령 데이타 링크 계층에서의 잦은 재전송으로 인한 지연이 TCP의 타임아웃 시간을 초과하여 TCP에서도 다시 재전송이 이루어진다면, 중복된 재전송으로 인해 심각한 성능 저하를 가져올 수 있다. 즉 데이타 링크 계층에서의 재전송은 전송 계층의 재전송 메커니즘과 밀접한 관계 속에서 이루어져야, 불필요한 중복 재전송을 줄일 수 있다.

망 계층에서의 방법은 데이타를 이동 목적지까지 제대로 전달하며 상위 프로토콜에 이동성을 숨기는 기능을 가지게 된다. 대표적인 방법으로 Mobile IP[3]와 IPv6에서의 이동성 지원 방법[4]을 들 수 있다.

전송 계층에서는 사용자의 핸드오버와 무선 라인에서의 높은 에러율에 따른 전송 프로토콜의 성능 하락을 막는 메커니즘들로 구성된다. 핸드오버 처리를 위한 방법으로 fast retransmission[5]을 들 수 있으며 무선 라인의 높

*종신회원

은 에러율로 인해 생기는 성능 하락을 막는 방법으로 indirect TCP[6]를 들 수 있다. Snoop [7]는 핸드오버와 높은 에러로 인해 생기는 문제를 함께 해결하기 위한 방법이다. 또 무선 상황에서 사용하기 위해 만들어진 방법은 아니지만 무선 라인에서 사용할 경우 좋은 성능을 보이는 것으로 알려진 방법으로 TCP Vegas [8], packet pair[9], SMART[10] 등을 들 수 있다.

본 논문에서는 링크 계층을 제외한 망 계층과 전송 계층에서 호스트 이동성을 지원하는 데 고려해야 할 문제점과 지금까지 제안된 기법들을 살펴보고자 한다.

2. 망 계층에서의 이동성 지원

2.1 Mobile IP[3, 11]

이동 호스트의 이동성을 네트워크 계층에서 지원하기 위해 고안된 것이 MobileIP이다. MobileIP를 지원하기 위해서 각 지역 네트워크 망에는 이동성을 지원하는 에이전트(agent)가 있고 이동 호스트의 현재 위치를 나타내는 care-of address를 이용해서 데이타 전송이 이루어진다.

MobileIP 설계의 기본 요구 조건은 투명성, 병립성, 그리고 보안성이다. 즉 이동 호스트는 계속 자신의 홈 주소를 사용하면서 어느 지점을 통해서라도 네트워크에 접속할 수 있어야 한다. 또한 기존의 보통 호스트와 라우터에는 아무 변화가 없으며 이동 호스트는 어떤 종류의 호스트와도 자유롭게 통신할 수 있어야 한다. 마지막으로 모든 등록 메시지는 인증을 받아야 한다.

MobileIP의 작동 원리는 그림 1, 2와 같다.

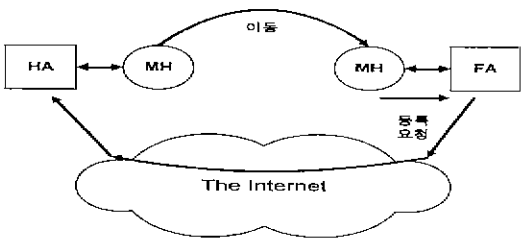


그림 1 Mobile IP의 이동, 등록 절차

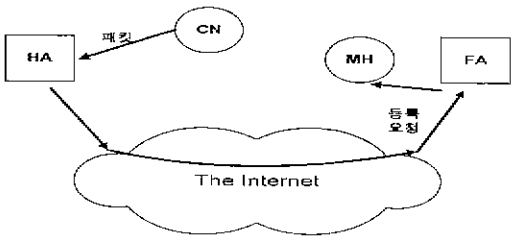


그림 2 Mobile IP의 데이타 전송

이동 호스트(MH)가 자신의 홈 에이전트(HA)가 아닌 다른 외부 에이전트(FA)가 관리하는 네트워크로 이동하였을 경우, 이동 호스트는 이 외부 에이전트에 등록을 하고 외부 에이전트는 이 사실을 홈 에이전트에게 알림으로써 이동 호스트의 현재 위치를 알려주게 된다. 이동 호스트로 전송되는 데이타는 우선 홈 에이전트로 보내진다. 홈 에이전트가 이 데이타를 터널링을 통해 외부 에이전트로 보내주면 외부 에이전트가 최종적으로 데이타를 이동 호스트로 전송해 주게 된다. 이때 이동 호스트의 현재 위치를 나타내는 care-of address는 외부 에이전트 자신의 IP 주소일 수도 있고, DHCP 등을 통해 할당받은 이동 호스트의 지역 주소일 수도 있다.

홈 에이전트가 이동 호스트로 데이타를 전송하는 처리 과정을 터널링(tunneling)[12, 13, 14]이라고 한다. 우선 본래의 데이타를 이동 호스트의 care-of address가 목적지로 설정된 다른 IP 패킷으로 감싼 다음에 데이타를 전송한다. care-of address에 도착한 패킷은 본래의 IP 패킷을 감싼 헤더를 제거하여 다시 본래의 IP 패킷으로 만들어진 후 이동 호스트로 전송된다.

2.2 IPv6의 이동성 지원[4]

IPv6는 IPv4의 주소 공간 고갈 문제를 해결하기 위해 만들어진 차세대 IP이다[15]. IPv6는 IPv4를 사용하는 현재의 인터넷 망에 큰 변화를 주지 않으면서 점진적으로 진행해 나갈 수 있도록 IPv4만을 지원하는 망과 함께 동작할 수 있다[16]. 또 IP의 변경은 자주 할 수 있는 일이 아니므로 IP를 바꾸면서 여러 추가적인 기능들을 포함시켰다.

이중의 일부 기능은 IPv4에서의 이동성 지

원보다 쉽게 이동성을 지원할 수 있게 해 주는데, 이러한 기능으로 주소 자동 설정(address autoconfiguration)[17, 18], 옵션 기능의 강화[15], 인접 호스트 발견(neighbor discovery)[19] 기능 등을 들 수 있다.

IPv6에서 이동 호스트는 동적으로 자신의 주소를 할당받을 수 있는데 이러한 기능을 주소 자동 설정 기능이라고 한다. 주소 자동 설정에는 상태를 유지하는(stateful) 방법[17]과 상태를 유지하지 않는(stateless) 방법[18]이 있다. 상태를 유지하는 경우는 주소 할당을 담당하는 서버가 있어 새로 이동해 온 호스트들에게 주소를 할당해 주게 된다. 이것은 IPv4에서의 DHCP(Dynamic Host Configuration Protocol)[20]을 확장한 것으로 DHCPv6라고 불린다. 상태를 유지하지 않는 경우는 각 이동 노드들이 분산적인 방법으로 자신의 주소를 결정하게 된다. 이때 새로 들어온 호스트의 주소가 기존에 있던 호스트의 주소와 같아지는 것을 방지하기 위해 새로 들어온 호스트는 임시로 주소를 만들어 지역망의 모든 호스트들에게 멀티캐스트하게 된다. 만약 어떤 호스트가 자신의 주소와 충돌이 생겼다는 메시지를 보내면 다른 주소를 만들어 같은 작업을 반복함으로써 충돌을 막게 된다. IPv6에서 이동 호스트는 바로 이러한 주소 자동 설정 기능을 이용해 자신의 care-of address를 할당받을 수 있다.

3. 전송 계층에서의 이동성 지원

3.1 TCP의 이동성 지원 문제

TCP[21]는 인터넷에서 사용되고 있는 전송 프로토콜이다. TCP는 동적으로 윈도우의 크기를 변화시키는 윈도우 기반의 혼잡(Congestion) 제어 방법을 사용한다. TCP는 기본적으로 패킷의 손실을 혼잡이 발생한 신호로 인식해 동작한다. 윈도우의 크기는 보낸 모든 패킷에 대한 ACK를 전송받은 경우 선형적으로 증가하며 일정 시간안에 ACK를 수신받지 못해 타임아웃이 걸리면 지수적으로 감소하게 된다. 즉, 윈도우의 크기를 조금씩 늘려 나가다가 혼잡이 발생한 경우 윈도우의 크기를 크게 줄이는 것이다. 이처럼 윈도우의 크기를 선형적으로 증가시키고 지수적으로 감소시키는 것은 혼잡제어 알고리즘을 안정적으로 수행하기 위한 필요 조건임이 알려져 있다.

TCP 연결이 맨처음 만들어진 경우는 적절한 윈도우의 크기를 알지 못하므로 윈도우의 크기는 1부터 시작하게 된다. 이때 윈도우를 적절한 크기로 증가시키는 데에 너무 많은 시간이 소요되므로 TCP의 시작 시기에는 slow start라는 메커니즘이 사용된다. 즉, 윈도우의 크기만큼의 ACK가 도착할 때마다 윈도우의 크기를 증가시키는 선형적 증가 방식을 쓰지 않고 하나의 ACK가 도착할 때마다 윈도우의 크기를 증가시키는 것이다.

Slow start는 연결이 처음 만들어진 시점뿐만 아니라 타임아웃이 걸린 경우에도 동작할 수 있다. 이 경우는 윈도우의 크기를 반으로 줄이는 대신 윈도우의 크기를 1로 줄인후 slow start 메커니즘을 동작시키게 된다. 윈도우의 크기가 타임아웃 발생 이전의 윈도우의 크기의 반까지 증가되면 다음부터는 선형적으로 윈도우의 크기가 증가된다.

그러나 타임아웃을 사용해 패킷의 손실을 알아내는 경우는 패킷의 손실을 감지하기까지 너무 많은 시간이 소요된다는 단점이 있다. 따라서 fast retransmission[22] 방법을 사용할 수 있다. 즉, ACK가 도착하지 않았을 때 타임아웃이 걸릴 때까지 기다리는 대신 동일한 패킷의 재전송을 요구하는 duplicate ACK가 3번 도착하면 즉시 재전송을 하는 것이다. 이러한 fast retransmission 방법을 사용해 패킷의 손실에 빠르게 대응할 수 있다.

TCP는 현재 여러 버전으로 구현되어 있는데, 대표적인 것은 Tahoe와 Reno 버전이다[22]. TCP Tahoe는 초기의 TCP 버전으로 타임아웃이 발생하거나 duplicate ACK가 도착한 경우에 모두 윈도우의 크기를 1로 줄이고 slow start를 시작한다. TCP Tahoe 이후의 버전인 TCP Reno는 타임아웃이 발생한 경우만 윈도우의 크기를 1로 줄이고 duplicate ACK가 도착한 경우는 윈도우의 크기를 반으로 줄이게 된다.

TCP는 다양한 대역폭에서 운용될 수 있으

며 오랫동안 인터넷에서 검증을 받은 프로토콜이다. 그러나 TCP가 이동 컴퓨팅 환경에서 사용될 경우는 새로운 문제가 발생하게 된다. 첫 번째 문제는 무선 라인의 높은 에러율로 인해 생기는 문제이다. TCP는 기본적으로 패킷의 손실을 혼잡의 척도로 생각하므로, 혼잡이 아닌 다른 이유로 생긴 에러에 대해서도 윈도우의 크기를 줄이므로 전체적인 성능을 저하시키게 된다. 또다른 문제는 핸드오버로 인한 문제이다. 즉 핸드오버가 진행되는 동안 이동 호스트에게 전달된 메시지는 전달되지 않을 것이므로 burst 에러가 발생하게 되어 TCP의 성능을 하락시키는 것이다. 이런 문제를 해결하기 위해 TCP를 개선한 여러 방법이 제안되어 왔다.

3.2 TCP Vegas[8]

TCP가 혼잡이 발생한 경우를 감지해 그에 적절하게 대응하는 혼잡 발견 메커니즘인데 반해, TCP Vegas는 혼잡을 미리 방지하는 혼잡 예방 메커니즘이다. TCP Vegas는 throughput의 변화를 감지해 현재 망에서 전송중인 데이타의 양을 짐작해, 전송중인 데이타의 양이 너무 많은 경우는 윈도우의 크기를 줄이고 전송중인 데이타의 양이 너무 적은 경우는 윈도우의 크기를 늘리게 된다.

TCP Vegas는 전송 중인 데이타의 양을 추정하는 척도로 RTT(Round Trip Time)를 사용한다. TCP Vegas는 패킷을 보낸 후 ACK가 도착할 때까지의 시간을 측정해 그 최소값을 BaseRTT로 저장해 혼잡이 발생하지 않았을 때의 RTT 값으로 생각한다. 그리고 윈도우의 크기를 BaseRTT로 나눈 값을 ExpectedRate로 계산하고 실제 보내는 rate와의 차이를 계산해 그 값이 임계값 α보다 작은 경우는 망 자원이 많이 남는 것으로 생각해 윈도우의 크기를 늘리고 임계값 β보다 큰 경우는 혼잡이 발생한 것으로 생각해 윈도우의 크기를 줄이게 된다.

TCP Vegas는 패킷의 손실을 혼잡의 척도로 생각하는 대신 RTT를 측정해 혼잡을 예방하기 때문에 에러가 많이 생기는 무선 라인 위에서도 잘 동작한다. 지금까지의 실험 결과는 TCP Vegas가 TCP Tahoe나 Reno보다 더 좋은 성능을 보이는 것으로 나타났다. 그러나 TCP Tahoe나 Reno가 TCP Vegas와 함께 사용될 때 TCP Vegas가 좋은 성능을 보일지는 앞으로의 연구가 필요하다.

3.3 Indirect TCP[6]

이동 호스트가 고정 호스트와 통신을 하는 경우, 이동 호스트의 이동성과 무선 링크의 불안정한 성질 때문에 성능 저하가 초래된다. I-TCP는 유선 링크와 무선 링크의 성질이 다르다는 점에 착안하여 End-to-end TCP 연결을 유선 링크에서의 연결과 무선 링크에서의 연결로 나누어서 통신을 하자는 방법이다.

이동 호스트가 고정 호스트와 통신하고자 하는 경우 이 연결은 이동 호스트에서 MSR까지의 무선 링크 연결과 MSR에서 고정 호스트까지의 유선 링크 연결로 나누어진다. 유선 링크와 무선 링크는 통신 채널의 상태가 매우 다르므로 각 채널의 상태에 맞게 두 연결은 서로 다른 트레픽 인자를 가지게 된다. 즉 유선 링크 연결에서는 보통의 TCP 연결을 사용하면 되지만 무선 링크 연결에서는 무선 링크의 상태에 적절히 맞추어진 트레픽 인자를 사용할 수 있다.

이렇게 연결을 무선 링크 연결과 유선 링크 연결로 나누는 이점은 이동성과 무선 링크의 불안정과 관련된 문제들이 무선 링크 내에서만 처리가 된다는 점이다. 따라서 고정 호스트와 이동 호스트가 하나의 연결을 통해서 통신할 경우 생길 수 있는 문제, 즉 핸드오프나 무선 링크 상에서의 비트 에러로 인한 불필요한 혼잡 제어 메커니즘의 호출 등의 문제를 제거할 수 있기 때문에 성능 향상을 얻을 수 있다. 또한 기지국이 이동 호스트를 대신해서 많은 통신 오버헤드를 처리해 주기 때문에 이동 호스트는 MSR과 통신하기 위한 간단한 무선 프로토콜만을 사용할 수 있다.

그러나 I-TCP의 치명적인 단점은 바로 이 연결 분리로 인해서 end-to-end 프로토콜 Semantics가 파괴된다는 것이다. 즉 TCP ACK가 end-to-end 의미를 가지지 않기 때문에 MSR의 고장이나 오랜 시간의 연결 단절시에는 중대한 문제가 발생하게 된다. 또한 이동

호스트의 핸드오프시에 MSR의 트랜스포트 계층간에 메시지들이 전송되어야 하는 오버헤드가 존재하며, I-TCP를 위한 새로운 소켓 인터페이스가 필요하게 된다.

3.4 Fast Retransmission[5]

이동 호스트의 핸드오프로 인한 지연과 패킷 손실을 TCP는 혼잡으로 인한 것으로 해석한다. 이러한 핸드오프로 인한 지연을 줄임으로써 TCP의 성능 향상을 높이는 방법으로 제시된 것이 fast retransmission이다.

일반적인 핸드오프 시나리오는 아래 그림 3과 같다. 기지국은 주기적으로 beacon 메시지를 보내고 이 beacon 메시지를 받는 이동 호스트는 자신이 어느 셀에 있는 지 알게 된다. 이때, 자신이 받고 있는 beacon메시지 보다 더 강한 신호의 beacon 메시지가 올 경우 핸드오프가 일어났음을 알 수 있다. 이 경우 이동 호스트는 greeting 메시지를 보냄으로써 새로운 기지국에게 자신의 존재를 알려주게 되고, 새로운 기지국은 이전의 기지국에게 핸드오프 사실을 알려준 후 라우팅 정보 등의 해당 정보들을 가져오게 된다.

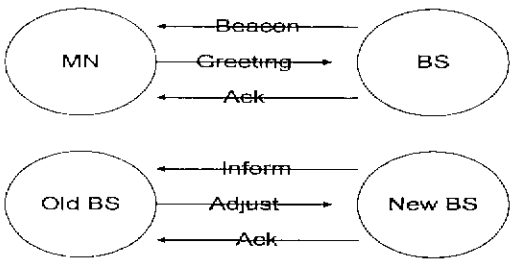


그림 3 핸드오프 메커니즘

최근 TCP 버전들은 세개의 동일 ACK(triplicate acknowledgement)를 받았을 경우, fast retransmission 프로시저를 호출하도록 구현되어 있다. 즉 세개의 동일 ACK를 받았을 경우 패킷 손실이 일어났다고 간주를 하고, 타임아웃을 기다리지 않고 바로 ACK를 받지 못한 패킷을 재전송하게 된다. 무선 환경에서 이동 호스트의 핸드오프를 다루는 fast retransmsission 역시 이와 비슷한 메커니즘을 따른다. fast retransmission 메커니즘은 이동 호스트와 고정 호스트의 TCP 코드를 약간 수정함으로써 만들 수 있다.

먼저 이동 호스트가 송신자일 경우를 살펴본다. 핸드오프가 일어났을 경우 이동 호스트는 Greeting 메시지를 새로운 기지국에 보내고 기지국으로부터 Greeting ACK 메시지를 받는다. 이 ACK를 받았다는 것은 핸드오프의 종결을 의미한다. 이때 이동 호스트의 MobileIP는 TCP에게 신호를 보내고, TCP는 타임아웃을 기다리지 않고 즉시 fast retransmission 프로시저를 호출하게 된다. fast retransmission 프로시저를 호출하게 되면 수신자로부터 ACK를 받지 못한 패킷부터 다시 재전송이 이루어진다.

고정 호스트가 송신자이고 이동 호스트가 수신자일 경우를 살펴보자. 핸드오프가 일어난 후(Greeting ACK 수신 후) 이동 호스트의 MobileIP는 TCP에게 핸드오프가 종결되었음을 알린다. 그러면 이동 호스트의 TCP는 고정 호스트의 TCP에게 핸드오프가 일어났다는 신호를 보내게 된다. 신호를 받은 고정 호스트의 TCP는 타임아웃을 기다리지 않고 바로 fast retransmission 프로시저를 호출하게 된다. 이때 이동 호스트가 고정 호스트에게 보내는 신호는 이동 호스트가 최종적으로 받은 패킷 번호를 적은 특별 ACK일 수도 있고, 세개의 동일한 보통 TCP ACK일 수도 있다. 보통 ACK를 사용하는 경우에는 고정 호스트의 TCP 코드를 수정하지 않아도 된다는 장점이 있다.

그러나 fast retransmission은 핸드오프에만 관심을 두었고, 비트 에러에 의한 패킷 손실 등 그 외의 요소에 대해서는 해결책을 제시하지 못하고 있다는 단점이 있다.

3.5 Snoop 모듈[7]

TCP는 패킷 손실의 주된 원인이 혼잡에 의한 것이라는 가정 하에서 만들어진 신뢰성 있는 트랜스포트 프로토콜이다. 그러나 무선 링크와 이동 호스트가 있는 환경에서는 혼잡 외에 비트 에러와 핸드오프에 의한 패킷 손실이 많이 발생한다. TCP는 이러한 무선 링크에서

의 비트 에러나 핸드오프로 인한 패킷 손실을 혼잡에 의한 패킷 손실로 가정하기 때문에, 불필요한 혼잡 제어 메커니즘을 작동하게 되고, 이는 곧 성능 저하로 이어진다. 이와 같은 문제점을 개선하기 위해 방법으로 기지국에서의 버퍼링과 재전송의 기능을 수행하는 snoop 모듈과, 핸드오프 시간을 줄이는 라우팅 프로토콜을 이용하는 방법이 제안되었다.

이 방법의 기본 목적은 기존의 TCP에 대한 수정 없이 성능 향상을 추구한다는 것이다. 또한 기지국에서의 재전송이 I-TCP(Indirect TCP)에서 나타나는 문제인 프로토콜 semantics를 어긋나지 않게 한다는 것이다.

Snoop 모듈은 무선 환경에서의 TCP 성능 향상을 위한 노력중에서, 비트 에러로 인한 패킷 손실을 줄이는 기능을 담당한다. 이는 다시 고정 호스트에서 이동 호스트로 데이타를 전송하는 부분과 이동 호스트에서 고정 호스트로 데이타를 전송하는 부분으로 나뉘게 된다. 고정 호스트에서 이동 호스트로 데이타를 전송하는 경우에는 기지국에서 데이타를 캐싱하고, 패킷 손실 시에는 이를 지역 재전송하게 된다. 이동 호스트에서 고정 호스트로 데이타를 전송하는 경우에는 기지국에서 손실된 패킷을 감지한 후 이동 호스트에게 NACK를 이용해서 이를 알려준다.

핸드오프로 인한 패킷 손실은 핸드오프로 인한 지연을 줄임으로써 그 영향을 최소화할 수 있다. 이를 위해서 이동 호스트가 현재 있는 셀을 관장하는 기지국 외에 근처의 핸드오프가 예상되는 기지국으로도 데이타를 전송하는 방법을 사용한다. 즉 현제 속한 셀과 그 주변 셀의 기지국들이 하나의 멀티캐스트 그룹에 가입하고 송신자는 데이타를 이 멀티캐스트 그룹으로 전송한다. 핸드오프가 일어난 경우, 새로 이동한 셀의 기지국에 있는 snoop 모듈은 캐쉬의 데이타를 전송하게 된다. 즉 기지국간에 핸드오프와 관련된 정보가 직접적으로 전달되지 않고, 이미 캐싱한 데이타를 이용해서 데이타 전송을 계속하므로 핸드오프 지연 시간을 줄일 수 있게 된다.

새로 이동해간 셀의 기지국이 캐쉬에 가지고 있는 데이타와 이동 호스트가 현재까지 수신한 데이타 사이에는 약간의 차이가 있을 수 있다. 그러나 snoop 모듈은 데이타를 캐싱하고 이를 지역 재전송함을 통해서 성능 향상을 얻는 방법이므로 I-TCP와는 달리 TCP ACK가 end-to-end 의미를 가지게 된다. 즉 프로토콜 semantics를 유지하므로 이러한 상태 정보의 차이가 치명적인 문제를 발생시키지 않는다.

3.6 Packet Pair[9]

Packet pair는 모든 라우터들이 round robin 방식의 스케쥴링을 한다고 가정하고 병목 현상이 발생한 라우터의 버퍼량을 추정하기 위해 두개의 연속된 패킷을 보낸다. 그림 4는 두 개의 패킷을 보냈을 때의 지연 양상을 보여 준다. 두 패킷의 간격은 병목 현상이 발생한 라우터에서 가장 커짐을 알 수 있다.

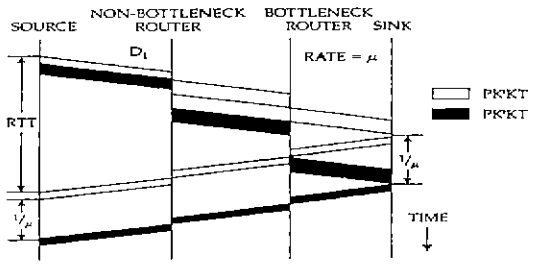


그림 4 쌍으로 보낸 패킷의 지연

Packet pair에서는 모든 패킷을 한 쌍의 단위로 전송하고 그 두 패킷의 ACK가 도착하는 간격으로부터 병목 라우터의 처리율을 추정한 후 지수적 평균 방법으로 처리율을 계산한다. 즉, k번째의 패킷에 대한 병목 라우터의 처리율이 $\mu(k)$라고 하고 새로 추정한 값이 $\hat{\mu}(k)$라고 할때 $\hat{\mu}(k+1)$은 다음과 같은 식으로 구한다.

$$\alpha\ \hat{\mu}(k+1)=\alpha\times\hat{\mu}(k)+(1-\alpha)\times\mu(k)$$

값은 상황에 따라 동적으로 변화된다. 전송했으나 아직 ACK를 받지 못한 패킷 수를 S라고 하고 RTT가 R일 때 병목 라우터에 있을 것으로 추정되는 패킷의 양 X는 다음과 같이 계산된다.

$$X=S-R\hat{\mu}(k+1)$$

이때 송신자는 이 값들을 이용해 송신율 $\lambda(k+1)$을 다음과 같이 결정한다.

$$\lambda(k+1)=\hat{\mu}(k+1)+(B-X)/R$$

여기서 B는 병목 라우터의 큐에 있는 패킷의 크기를 의미한다. 송신율을 이처럼 결정할 경우 혼잡 제어 방법이 안정적으로 동작함이 분석과 시뮬레이션을 통해 알려져 있다. 그러나 packet pair의 단점은 각 라우터가 WRR [23], WFQ[24], PGPS[25], HRR[26], DRR [27] 등 공평성있는 스케줄링 방법에 의해 동작해야 한다는 것인데, 현재 망에서는 그렇지 않은 라우터들이 많으므로 당장 사용하기에는 곤란하다. 또 에러가 발생한 경우에 병목 라우터의 처리율 추정값이 실제와 다를 수 있다. 이러한 문제는 SMART[10]를 함께 사용함으로써 해결할 수 있다.

3.7 SMART[10]

SMART(Simple Method to Aid Retransmission)는 기존의 ARQ 방법들을 개선한 새로운 재전송 기법이다. 기존에 사용되는 대표적인 ARQ 방법으로는 GBN(Go-Back N)과 SACK(Selective Acknowledgement)을 들 수 있다. GBN에서는 전송하였지만 아직 수신자로부터의 ACK를 받지 못한 패킷의 리스트를 유지하고 있어 타임아웃이 발생하면 전체 윈도우를 다시 재전송한다. 이러한 방법은 모든 에러를 burst한 것으로 간주하는 방법으로 융통성이 떨어지며 적절한 타임아웃의 크기를 결정하기가 힘들다. 이러한 문제를 해결하기 위해 TCP 등에서는 GBN을 개선해 혼잡 제어 알고리즘이 윈도우의 크기를 줄임으로써 윈도우의 앞에 있는 패킷만을 전송하게 한다. 그러나 이런 방법에서는 초기에 윈도우의 크기가 작아 throughput이 떨어지게 된다.

또 다른 ARQ 방법인 SACK는 ACK가 받은 패킷들의 리스트를 포함하고 있어 송신자가 일부의 패킷만을 재전송할 수 있게 하는 방법이다. 그러나 이 방법은 ACK의 크기가 너무 커지는 단점을 지니고 있다.

SMART는 이러한 기존 ARQ 방법의 단점을 보완하기 위한 새로운 재전송 방법이다. SMART는 ACK가 (C, S)의 두 정보를 포함하게 된다. C는 수신자가 지금까지 연속적으로 손실없이 받은 패킷의 마지막 일련 번호이며 S는 ACK를 발생시킨 패킷의 일련번호이다. 송신자가 ACK를 받으면 C와 S를 비교하게 된다. 만약 이 두 값이 같으면 손실 없이 패킷이 전송된 것이며 그렇지 않은 경우는 C와 S의 사이에 있는 패킷들이 손실된 것으로 간주해 재전송하게 된다. 이 방법은 ACK의 크기를 작게 하면서 GBN의 전체 윈도우 재전송 문제를 해결한 방법이다.

4. 결 론

본 논문에서는 인터넷에서의 이동성 지원을 위한 기존의 연구 결과들을 망 계층과 전송 계층을 중심으로 살펴보았다. MobileIP나 IPv6의 이동성 지원 방안은 상위 계층에 이동의 투명성을 제공할 수 있다. IP 계층에서의 해결은 이동 목적지로 패킷을 올바르게 전달할 수는 있지만 상위 전송 프로토콜의 성능을 떨어뜨릴 수 있으므로 위에서 소개한 TCP의 개선 방법들을 병행해 사용해야 한다. 이러한 TCP 개선 방법들은 여러 형태로 결합되어 사용될 수 있을 것으로 보인다.

지금까지 인터넷의 이동성 지원에 대한 연구는 주로 일대일 유니캐스트(Unicast) 연결의 경우에 대해 수행되어 왔으나 점차 멀티캐스트 연결에서 이동성을 지원하는 문제에 대한 연구가 활발히 진행되고 있다. 멀티캐스트와 이동성이 결합되는 경우는 유니캐스트에서 볼 수 없는 여러 다양한 문제가 발생하므로 앞으로 이에 대한 폭넓은 연구가 필요할 것이다.

참고문헌

[1] S. Paul, E. Ayanoglu, T. F. LaPorta, K. H. Chen, K. K. Sabnani and R. D. Gitlin, "An asymmetric link-layer protocol for digital cellular communications," Proc. Infocom '95(1995).

[2] A. DeSimone, M. C. Chuah and O. C. Yue, "Throughput performance of

transport-layer protocols over wireless LANs," Proc. Globecom '93(1993).

[3] C. Perkins, "IP Mobility Support," Internet RFC 2002.

[4] David B. Johnson, "Mobility Support in IPv6," Internet draft.

[5] Ramon Caceres, Liviu Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," IEEE Journal on Selected Areas in Communications, Vol. 13, No. 5 June 1995.

[6] A. Bakre and B. R. Badrinath, "Handoff and system support for indirect TCP/IP," Proc. Second Usenix Symp. on Mobile and Location-Independent Computing (1995).

[7] Hari Balakrishnam, Srinivasan Seshan and Randy H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," Wireless Networks I(1995) pp. 469~481.

[8] L. S. Brakmo and L. L. Peterson, "TCP Vegas : End to End Congestion Avoidance on a Global Internet," IEEE Journal on Selected Areas in Communications, Vol. 13, No. 8, October 1995, pp. 1465~1480.

[9] S. Keshav, "Packet-pair Flow Control," IEEE/ACM Transactions on Networking, to appear, 1997.

[10] S. Keshav and S. P. Morgan, "SMART : Performance with Overload and Random Losses," Proceedings of IEEE Infocom. 97, April 1997.

[11] C. Perkins, "IP Mobility Support version 2," Internet draft.

[12] C. Perkins, "IP Encapsulation within IP," RFC 2003.

[13] W. Simpson, "IP in IP Tunneling," RFC 1953.

[14] C. Perkins, "Minimal Encapsulation within IP," RFC 2004.

[15] S. Deering, R. Hinden, "Internet Protocol. Version 6 (IPv6) Specification," Internet draft.

[16] R. Gilligan, E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers," RFC 1933.

[17] J. Bound, "Dynamic Host Configuration Protocol for IPv6(DHCPv6)," Internet draft.

[18] S. Thompson, T. Narlin, "IPv6 Stateless Address Autoconfigurationm," RFC 1971.

[19] T. Narten, E. Nordmark, "Neighbor Discovery for IP Version 6 (IPv6)," RFC 1970.

[20] R. Droms, "Dynamic Host Configuration Protocol," RFC 2131.

[21] Information Sciences Institute, "Transmission Control Protocol", RFC 793.

[22] V. Jacobson, "Congestion Avoidance and Control," Proceedings of ACM SIGCOMM "88, Stanford, August 1988.

[23] A. Demers, S. Keshav, and S. Shenker, "Design and Analysis of a Fair Queieing Algorithm," Proceedings of ACM SIGCOMM 89, Austin, Sep. 1989.

[24] A. Demers, S. Keshav, and S. Shenker, "Design and Analysis of a Fair Queueing Algorithm," Proceedings of ACM SIGCOMM '89, Austin, September 1989.

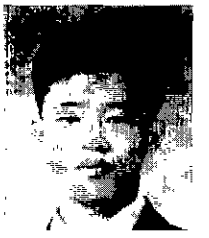
[25] Parekh, A. K., R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks : The Single-Node Case," IEEE/ACM Trans. on Networking, 1, 3 (June 1993).

[26] C. R. Kalmanek, H. Kanakia, and S. Keshav, "Rate Controlled Servers for Very High-Speed Networks," Proceedings of Globecom '90, San Diego, December 1990.

[27] M. Shreedhar, G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," Proceedings of ACM SIGCOMM '95, Boston, September 1995.

임 효 준

1996 서울대학교 전산과학과 학사
1996~현재 서울대학교 전산과학과 석사과정



김 응 도

1997 서울대학교 전산과학과 학사
1997~현재 서울대학교 전산과학과 석사과정

김 종 권

1981 서울대학교 산업공학과 학사
1987 미국 일리노이 대학 전산과학 박사
1987~1991 미국 벨코어 연구소
1991~현재 서울대학교 전산과학과 부교수 재직