SCTP의 멀티호밍 특성에 대한 성능 평가

송 정 화[†] · 이 미 정[††] · 고 석 주[†††]

요 약

SCTP는 TCP와 마찬가지로 연결 지향적이며 신뢰성 있는 데이터의 전송을 위한 전송 계층 프로토콜이다. SCTP는 오류 및 플로우 제어 등 많은 부분에 있어서 TCP의 방식을 그대로 따르며, 거기에 더하여 멀티스트리밍과 멀티호밍 특성을 가진다. 이 논문에서는 TCP와 다른 대표적인 특징들 중 멀티호밍이 성능에 미치는 영향을 살펴 보았다. 먼저 멀티호밍을 지원하는 SCTP와 그렇지 않은 경우의 SCTP나 TCP Reno, TCP SACK의 성능을 시뮬레이션을 통해 비교하였다. 또한, 멀티호밍을 지원하는 경우에 SCTP의 재전송 정책이 성능에 미치는 영향을 살펴보았다. 프로토콜간의 성능 비교를 통해서는 SCTP가 사용하는 몇 가지 혼잡제어 메커니즘 특징으로 인해 SCTP가 TCP Reno나 TCP SACK에 비해 향상된 성능을 보임을 확인할 수 있었으며, 특히 멀티호밍을 지원하는 경우의 SCTP가 가장 짧은 지연을 가짐을 확인하였다. 또한, 멀티호밍을 지원하는 경우 현재의 SCTP의 재전송 정책이 프라이머리 경로와 대체 경로간의 경로 특성의 따라 성능 저하를 가져올 수 있는 잠재적인 문제점을 가짐을 확인하였다. 따라서 재전송을 위한 경로 선택에 있어서 대체 경로의 상태 파악이 중요한 요소이며 이를 위한 방안이 필요할 것이다.

Performance Evaluation on SCTP multi-homing Feature

Jeonghwa Song[†] · Meejeong Lee[††] · Seokjoo Koh[†††]

ABSTRACT

Stream Control Transmission Protocol(SCTP) is a new connection-oriented, reliable delivery transport protocol operating on top of an unreliable connectionless packet service such as IP. It inherits many of the functions developed for TCP, including flow control and packet loss recovery functions. In addition, it also supports transport layer multihoming and multistreaming. In this paper, we study the impact of multi-homing on the performance of SCTP. We first compare performance of single-homed SCTP, multi-homed SCTP, TCP Reno and TCP SACK. We, then describe potential flaw in the current SCTP retransmission policy, when SCTP host is multihomed. Our Results show that SCTP performs better than TCP Reno and TCP SACK due to several changes from TCP in its congestion control mechanism. In particular, multi-homed SCTP shows the best result among the compared schemes. Through experimentation for multi-homed SCTP, we found that the current SCTP retransmission policy may deteriorate the perfomance when the retransmission path is worse than the original path. Therefore, the condition of retransmission path is a very important factor in SCTP performance and a proper mechanism would be required to measure the condition of the retransmission path.

키워드 : SCTP(SCTP), 멀티호밍(Multi-Homing), 재전송 정책(Retransmission Policy)

1. 서 론

SCTP는 IP 네트워크에서의 텔레포니 시그널링 메시지를 전송하기 위해 IETF SIGTRAN 워킹 그룹에서 제안되었으며, 범용 전송 프로토콜로 2000년 10월에 SCTP 표준 문서 RFC 2960으로 제정되었다. SCTP는 연결 지향적인 프로토콜로써 통신을 위해서 두 엔드 포인트는 커넥션을 설립해야 하며, SCTP에서는 이 커넥션을 어소시에이션 (association)이라 부른다. SCTP는 TCP와 마찬가지로 신뢰성 있는 프로토콜이며, 오류 및 플로우 제어는 몇 가지 사항을 제외하고 대부분 TCP의 방식을 그대로 따르고 있다.

TCP와 다른 SCTP의 대표적인 특징으로는 멀티스트리밍과 멀티호밍이 있다. 따라서 SCTP는 데이터를 여러 개의 다른 스트림으로 나누는 것이 가능하며, 각 스트림은 그 특성에 따라 배달될 수 있고, 다른 스트림들과 독자적으로 다루어 질 수 있다. 스트림은 TCP와 마찬가지로 엄격히 순서를 지켜서 전달되도록 정의할 수도 있으며, 부분적으로만 순서를 지키거나, 순서와 무관하게 수신측에 도착하자마자 응용 계층으로 배달되도록 정의할 수도 있다. 엄격히 순서를 지켜서 전달되도록 하는 경우에 일부 데이터가 손실되거나 순서에 맞지 않게 수신측에 도착함으로써 발생하는 헤드 오브 라인 블로킹(Head-of-Line Blocking) 문제도 각 스트림이 독립적으로 동작하므로 해당 스트림에만 영향을

† 준 회 원 : 이화여자대학교 대학원 컴퓨터학과
†† 정 회 원 : 이화여자대학교 컴퓨터학과 교수
††† 정 회 원 : 한국전자통신연구원 표준연구센터 선임연구원
논문접수 : 2003년 8월 26일, 심사완료 : 2004년 2월 23일

줄 뿐, 다른 스트림에는 영향을 주지 않는다는 장점을 가진다.

두 번째 차이점인 멀티호밍은 IP계층과의 상호작용에 관계된 것이다. TCP는 하나의 호스트가 단지 하나의 IP 주소만을 가지는 것을 가정하는 반면 SCTP는 한 호스트가 여러 개의 IP 주소를 가지는 것을 허용한다. 이에 따라 커넥션 구분을 위해 TCP가 송신자 주소, 송신자 포트 번호, 수신자 주소, 수신자 포트 번호의 조합을 이용하는 것과는 달리 SCTP는 송신자의 IP 주소들의 집합, 송신자 포트 번호, 수신자의 IP 주소 집합, 수신자 포트 번호로 어소시에이션을 구분하게 된다. 이러한 멀티호밍 지원은 네트워크 결함에 좀 더 탄력적으로 대처할 수 있도록 하며, IP계층에서 보다 더 높은 수준의 신뢰성을 제공할 수 있도록 한다.

이 논문에서는 TCP와 다른 대표적인 특징들 중 멀티호밍이 성능에 미치는 영향을 살펴 보고자 한다. 먼저 멀티호밍을 지원하는 SCTP와 그렇지 않은 경우의 SCTP나 TCP Reno, TCP SACK의 성능을 시뮬레이션을 통해 비교하고자 한다. 또한, 멀티호밍을 지원하는 경우에 SCTP의 재전송 정책이 성능에 미치는 영향을 살펴보고자 한다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어서 2장에서는 SCTP의 새로운 특성들 중에서 우리가 다루고자 하는 특성인 멀티호밍에 대해서 자세히 소개하고, 3장에서는 비교하는 프로토콜들의 성능에 영향을 미치는 TCP의 혼잡 제어 메커니즘과 SCTP의 혼잡 제어 메커니즘을 설명한다. 4장에서는 시뮬레이션을 통해서 TCP Reno와 TCP Sack, 멀티홈드 호스트(multihomed host)에서의 SCTP와 싱글홈드 호스트(singlehomed host)에서의 SCTP의 성능을 비교, 분석한다. 마지막으로 5장에서는 결론과 함께 앞으로의 연구 방향을 기술한다.

2. SCTP 멀티호밍

앞 절에서 언급한 것처럼 SCTP와 TCP의 큰 차이점은 SCTP가 여러 개의 IP 주소들로 어소시에이션을 구성할 수 있다는 점이다. TCP의 경우는 두 엔드 포인트 간의 멀티호밍을 지원하지 않는다. 따라서 엔드 포인트의 IP 주소가 더 이상 액세스 가능하지 않게 되면, TCP 커넥션은 타임 아웃되거나 중단되게 되고 이는 응용에 의해 복구되어야 한다. 이러한 복구를 위한 오버헤드와 지연은 응용에 따라서는 중요한 문제이다. 그러나, SCTP는 이러한 문제를 염두해 두고 물리적으로 다중의 인터페이스가 존재하는 경우 모든 인터페이스로의 경로에 문제가 생기지 않는 한 어소시에이션이 유지될 수 있도록 디자인되었다. SCTP는 TCP와 달리 [송신자 IP 주소들의 집합, 송신자 포트 번호, 수신자 IP 주소 집합, 수신자 포트 번호]에 의해 어소시에이션이 정의된다. 어소시에이션 설립단계에서 각 SCTP 엔드 포인트는 사용 가능한 자신의 IP 주소들을 상대에게 알린다. 이것을 통해 상대 엔드 포인트는 주소 리스트를 만들게 되고, 이 유효한 주소들을 가지는 패킷에 대해서는 해당 어소시에이션으로 받아들이게 된다. 만약 이들 IP 주소가 다른 경로를 통해 액세스되도록 라우팅 되었다면, 이러한 특성은 네트워크 계층의 중복성을 제공하게 된다. 이러한 네트워크 계층에서의 중복성은 IP 주소들 중 하나가 도달할 수 없게 되어도 대체 IP 주소들 중 하나로 패킷을 재라우팅함으로써 그 호스트를 액세스하는 것이 가능하도록 한다.

그런데, RFC 2960에 따르면 SCTP는 이러한 여러 개의 링크들을 부하 분산을 위해서는 사용하지 않으며, 주소 리스트 중에서 프라이머리 주소라 불리는 하나의 주소를 정하고 이 주소를 통해서 상대편과 통신을 한다. 그 밖의 주소에 대해서는 세컨더리 주소(secondary) 혹은 대체 주소라 부르며, 단지 프라이머리 주소로 보낸 데이터가 손실되는 경우에 재전송을 위한 경로로써 사용한다[1].

이러한 SCTP의 현재 재전송 정책은 모든 재전송을 대체 목적지 주소로 보냄으로써 재전송 패킷이 수신측에 제대로 도착할 수 있는 기회를 늘리자는 데에 목적을 둔다. 여기에는 기본적으로 패킷 손실의 원인이 해당 목적지 주소가 더 이상 도달 가능하지 않거나 해당 경로가 혼잡이라는 가정을 가진다. 그러나 경우에 따라서는 현재의 SCTP의 재전송 정책은 성능을 감소시킬 수도 있다. 이 논문에서는 시뮬레이션을 통해 이러한 현재의 SCTP의 재전송 정책의 잠재적인 문제점을 보여주고자 한다.

3. 혼잡 제어 메커니즘

멀티스트리밍이나 멀티호밍 외에도 두 프로토콜간의 혼잡 제어 메커니즘의 차이로 인해 두 프로토콜의 성능은 달라지게 된다. SCTP는 기본적으로 TCP의 혼잡 제어 메커니즘을 기반으로 하고 있으나, 부분적인 차이로 인해 두 프로토콜은 성능 면에서 다른 양상을 띄게 된다. 이를 살펴보기 위해 두 프로토콜의 혼잡 제어 메커니즘에 대해 간략히 설명하고자 한다.

3.1 TCP의 혼잡 제어 메커니즘

기본적인 TCP 혼잡 제어는 슬로우 스타트(Slow Start), 혼잡 회피(Congestion Avoidance) 알고리즘 기반으로 하며 빠른 재전송(Fast Retransmit)과 빠른 복구(Fast Recovery) 두 가지의 알고리즘이 추가로 권고된다[2]. 슬로우 스타트와 혼잡 회피는 TCP 송신자가 네트워크에 주입한 데이터의 양을 제어하기 위해 사용된다. 슬로우 스타트는 RTT마다 혼잡 윈도우(cwnd)를 초기값으로부터 지수적으로 증가시켜가는 과정을 말한다. 이 과정은 가능한 빨리 전송 채널을 채우는 것을 목적으로 하며 슬로우 스타트 임계치에 다다를때까지 진행된다. TCP 송신자는 전송 초기에 또는 패

킷의 손실을 감지한 후에 이러한 슬로우 스타트를 한다. 혼잡 윈도우가 슬로우 스타트 임계치에 도달하게 되면, 혼잡 회피 알고리즘이 시행된다. 이것은 혼잡 윈도우를 선형적으로 증가시키는 방법으로써 RTT당 혼잡 윈도우를 1MTU (Maximum Transmission Unit)씩 증가시키도록 한다. 패킷 손실이 일어나게 되면, 패킷은 재전송되고 슬로우 스타트 임계치는 반으로 감소하게 되며, 혼잡 윈도우는 1MTU로 감소하게 된다.

이러한 슬로우 스타트와 혼잡 회피로 이루어진 혼잡 제어는 다음 문제들을 가진다. 패킷의 손실은 재전송 타이머의 타임 아웃에 의해서만 감지되는데 이것은 데이터 전송에 있어서 긴 지연을 가져오게 된다. 일반적인 전송 환경 아래에서 TCP가 중복 ACK을 받는 것은 순서에 어긋나게 수신측에 데이터가 도착했음을 나타낸다. 그러나 네트워크에 의한 패킷 지연이나 손실로 중복된 ACK을 받을 수도 있다. 이러한 사실에 근거하여 빠른 재전송 알고리즘은 재전송 타이머가 타임아웃되기 전에 3개의 중복된 ACK을 받게 되면 패킷을 손실로 간주하고 재전송을 하도록 하여 긴 지연을 줄이도록 한다. 또 패킷 손실 감지 후에 혼잡 윈도우를 1MTU로 줄이므로 인해 전송 채널을 최대한 활용하지 못한다는 문제를 해결하기 위해 빠른 복구 알고리즘이 제안되었다. 빠른 복구 알고리즘은 빠른 재전송에 의해 감지된 패킷 손실의 경우에는 혼잡 윈도우의 값을 슬로우 스타트 임계치에 3MTU를 더한 값으로 정하고 수신받게 되는 모든 새로운 중복 ACK에 대해 1MTU씩 증가시키도록 한다.

TCP Tahoe는 앞서 설명한 슬로우 스타트, 혼잡 회피, 빠른 재전송 알고리즘을 포함하는 TCP로써, 이전 TCP에 빠른 재전송 알고리즘을 추가하였으며, 측정한 RTT를 재전송 타이머의 타임아웃 값을 정하는 데에 사용하도록 수정하였다[3].

TCP Reno는 슬로우 스타트, 혼잡 회피, 빠른 재전송, 빠른 복구의 네 개의 알고리즘으로 이루어져 있다. TCP Tahoe를 향상시킨 TCP로써, 빠른 복구를 사용함으로 인해 Tahoe에 비해 패킷 손실 후 빠르게 전송률을 회복하는 것이 가능하다. 즉, TCP Reno는 재전송 타이머의 타임아웃에 의해 패킷 손실이 발견된 경우에만 심각한 혼잡이라 가정하고 TCP Tahoe와 마찬가지로 혼잡 윈도우를 1MTU로 줄이고 슬로우 스타트를 한다. 중복 ACK의 전송으로 패킷 손실이 발견된 경우에는 손실 후에 적어도 3개의 패킷이 전송되었으므로 심각한 혼잡이 아니라고 판단하고 혼잡 윈도우를 반으로 줄이고 혼잡 회피를 한다.

TCP SACK(Selective ACKnowledgment)은 수신자가 받은 패킷에 대한 정확한 정보를 송신자에 제공함으로써 TCP Reno의 문제점인 하나의 윈도우 내에서의 다중의 패킷 손실을 효과적으로 다루도록 하기 위한 방안이다[4, 5]. TCP Reno는 한 윈도우 내에서 여러 개의 패킷이 손실되는 경우에 모든 손실된 패킷에 대해서 빠르게 재전송을 할 수 없고 재전송 타이머의 타임아웃을 기다려야 한다. SACK에서 이러한 문제를 해결하기 위해 TCP 수신자가 송신자에게 자신이 수신한 패킷들을 정확하게 알려서 송신자가 선택적인 재전송이 가능하도록 한다. 따라서 손실된 패킷만이 재전송되도록 한다. 이를 위해 SACK 포맷에는 수신되어 수신자의 버퍼에 큐 되어 있는 세그먼트를 나타내는 블록들의 리스트를 포함한다. SACK은 TCP 헤더의 옵션 필드가 40byte로 제한되어 있으므로 최대 4개까지의 SACK 블록들을 가질 수 있으며, TCP 송신자는 SACK 옵션을 포함하는 ACK을 받으면 그 정보를 이용하여 재전송할 패킷을 결정한다.

TCP SACK과 같이 여러 개의 패킷 손실이 있는 경우 효과적인 복구를 위한 TCP로는 TCP NewReno가 있다. 기본적으로 TCP 송신자는 ACK을 받을 때에만 전송한 패킷이 정확히 도착했다는 것을 알 수 있다. TCP 송신자는 손실된 패킷의 수나 재전송이 필요한 패킷들을 정확히 알 수 없고, 또한 RTT당 하나의 손실된 패킷만 파악하는 것이 가능하다[5]. 빠른 재전송이 일어난 후에 그 결과로 모든 전송한 패킷이 ack되거나 일부분의 패킷이 ack되게 된다. 만약 부분적인 ACK(partial ACK)을 받게 된다면, 이것은 하나 이상의 패킷이 손실되었음을 나타낸다[6]. TCP NewReno에서는 재전송을 할 때, 전송된 가장 높은 순서 번호(sequence number)를 recover라는 변수에 기록하여 ACK을 받을 때 이 ACK이 부분적인 ACK인지 전송한 모든 패킷에 대한 ACK인지를 판단하도록 한다. 만약 부분적인 ACK이라면 아직 ACK되지 않은 가장 낮은 순서 번호를 가진 패킷을 재전송하고, 혼잡 윈도우는 ack된 데이터의 양만큼 감소시킨다. 윈도우의 증가는 한 세그먼트 단위로 이루어지고 새로운 패킷의 전송은 혼잡 윈도우가 허용할 때만이 가능하다. TCP NewReno는 TCP SACK과 같이 한 윈도우 내의 여러 개의 패킷 손실에 효과적인 복구를 위한 방안으로 TCP SACK이 송신자와 수신자 모두의 수정을 필요로 하므로 SACK을 지원하지 않는 TCP에 권고된다. TCP NewReno는 RTT당 하나의 패킷만이 재전송 가능하다는 단점을 가지므로 여러 개의 손실이 발생하는 경우에는 TCP 송신자는 어느 정도의 지연을 겪은 후에 복구가 가능하다.

3.2 SCTP의 혼잡 제어 메커니즘

SCTP는 TCP처럼 슬로우 스타트, 혼잡 회피, 빠른 재전송으로 이루어지는 윈도우 기반 혼잡 제어 메커니즘을 사용하여 신뢰성 있는 전송을 보장하고 손실된 패킷, 순서에 어긋나게 도착한 패킷, 중복된 패킷을 감지한다. 기본적인 혼잡 제어 메커니즘은 TCP와 유사하며, 다음과 같은 차이점을 가진다.

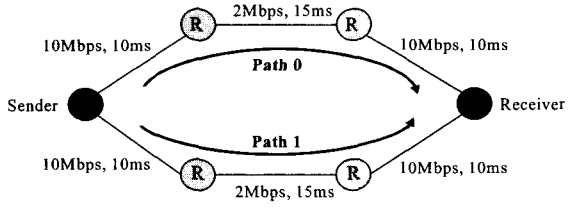
- TCP에서 SACK은 선택 가능한 옵션이지만, SCTP에서는 필수적인 사항이다.
- TCP의 경우는 cwnd의 초기값이 TCP의 경우는 일반적으로 1MTU이나, SCTP에서는 2×MTU이다. 이는 동일한 네트워크 상황에서 SCTP가 TCP에 비해 좀 더 빠른 시간 내에 많은 데이터를 전송할 수 있도록 한다.
- TCP는 cwnd가 ACK된 패킷의 수에 의해 증가되나, SCTP에서는 ACK된 바이트 수에 의해 증가한다.
- TCP는 cwnd가 슬로우 스타트 임계치보다 적은 경우에만 슬로우 스타트를 하고 cwnd가 슬로우 스타트와 같은 값이 되면 혼잡회피 과정으로 들어가는데 반해, SCTP에서는 cwnd가 슬로우 스타트 임계치와 같을 때까지 슬로우 스타트를 수행한다. 이는 작은 차이로 보이나 [7]에서의 성능 실험 결과 의미 있는 수준의 성능 차이를 보임을 알 수 있다.
- TCP는 세 개의 연속된 중복 ACK에 의해 빠른 재전송을 하는 반면, SCTP에서는 데이터의 손실을 알리는 네 개의 연속된 중복 ACK에 의해 빠른 재전송을 한다. 즉, 재전송하기까지 SCTP가 TCP보다 하나의 중복 ACK을 더 필요로 한다.
- TCP와는 달리 SCTP는 명시적인 빠른 복구 알고리즘을 사용하지 않는다. TCP나 SCTP의 혼잡 제어 알고리즘은 패킷을 전송한 후 ack 받지 못한 패킷들의 수보다 cwnd가 큰 경우에만 새로운 데이터의 전송이 가능하다 [1, 2]. TCP는 중복 ACK을 받을 때, 아직 ack받지 못한 패킷들의 크기를 나타내는 flight size를 줄이지 않기 때문에 cwnd를 증가시켜 주지 않으면 새로운 데이터를 전송하지 못한다. 따라서 명시적인 빠른 복구를 통해 빠른 재전송 후 중복 ACK을 받으면 cwnd를 1MTU씩 증가시켜 새로운 데이터의 전송을 가능하도록 한다. 그러나 SCTP는 중복 ACK을 받는 동안에도 SACK 사용으로 인해 새롭게 ack된 패킷을 파악할 수 있기 때문에 전송하고 ack받지 못한 패킷들의 크기를 나타내는 outstanding bytes가 감소하므로 cwnd의 증가 없이도 새로운 데이터의 전송이 가능하게 된다. 따라서 SCTP는 명시적인 빠른 복구 알고리즘을 사용하지 않는다.

4. 성능 평가

먼저 일반적인 데이터 전송시에 SCTP와 TCP의 성능 차이를 알아보기 위해 SCTP와 TCP에 대해 실험을 수행하였다. SCTP는 호스트가 멀티호밍을 지원하는 경우, 즉 목적지는 여러 개의 주소를 가지고 그 목적지에 도달하기 위해 여러 개의 경로를 가지는 경우에 이 경로를 모두 이용할 때와 호스트가 멀티호밍을 지원하지 않는 경우에 하나의 경로만을 이용할 때 각각에 대해서 실험을 수행하였다. TCP는 현재 신뢰성 있는 전송 프로토콜로써 일반적으로 가장 많이 사용되고 있는 프로토콜인 TCP Reno와 함께 TCP SACK을 실험하였다. SCTP가 기본적으로 SACK을 사용하므로 TCP SACK과의 비교를 통해 SACK 사용으로 인한 두 프로토콜 간의 성능차이를 배제하고, 멀티호밍과 함께 SACK이외의 SCTP 혼잡제어 방식으로 인해 발생하는 SCTP와 TCP의 성능 차이를 볼 수 있다. 또한, 멀티호밍을 지원하는 경우 프라이머리 경로와 대체 경로들간의 경로 특성에 따른 SCTP 성능을 살펴보기 위한 실험을 수행하였다. 시뮬레이션은 ns-2 네트워크 시뮬레이터에서 제공하는 TCP 모듈과 Delaware 대학교에서 ns-2 네트워크 시뮬레이터를 위해 구현한 SCTP 모듈을 이용하였으며, 리눅스 레드햇 7.3에서 수행되었다.

4.1 시뮬레이션 모델

(그림 1)은 시뮬레이션을 위해 사용된 토폴로지로써 두 라우터를 각각 연결하는 중간 링크는 2Mbps의 대역폭을 가지며 단방향 전파 지연은 15ms이다. 각 라우터는 다시 각각 하나의 에지 노드와 연결되어 있고, 이 링크들은 10Mbps의 대역폭을 가지며 단방향 전파 지연은 10ms이다. 종단간 단방향 전파 지연은 35ms로써 이는 미국 동부에서 서부 유럽까지의 대략적인 인터넷 지연이다. 결과적으로 두 에지 노드(송신원과 수신원)는 동일한 대역폭 및 지연을 가지는 두 개의 경로 path0과 path1을 통해 연결되어 있다. 각 에지 노드에는 TCP 또는 SCTP 에이전트가 올려지며, TCP 송신측은 path0을 통해 TCP 수신측에 데이터를 전송한다고 가정하였고, SCTP 송신측은 멀티호밍 호스트를 실험하는 경우 path0을 프라이머리 경로로 사용하고 path1을 대체 경로로써 재전송을 위해 사용한다고 가정하였다. 싱글호밍 호스트인 경우의 SCTP 성능 측정을 위해서는 TCP와 마찬가지로 path0을 이용하여 데이터를 전송한다고 가정하였다. TCP 혹은 SCTP 송신자는 일반적인 인터넷 MTU 크기와 같은 1500bytes로 패킷 크기를 고정하여 전송하며, SCTP인 경우에 어소시에이션이 하나의 스트림으로 구성되었다고 가정하였다.



(그림 1) 시뮬레이션 네트워크 토폴로지

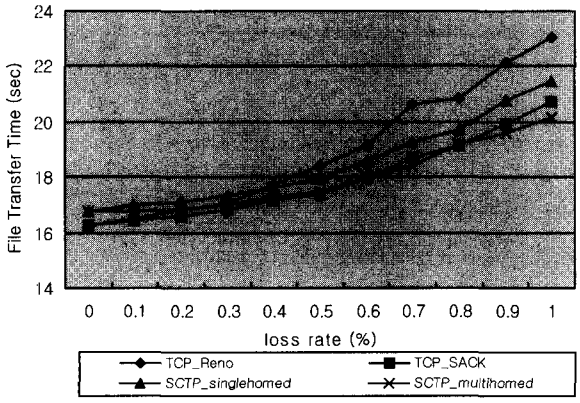
위의 기본적인 가정하에 두 경로의 패킷 손실율 및 대체 경로인 path1의 대역폭과 지연을 다양하게 변화시켜 보면서 4MB 파일을 전송하는 데에 소요되는 시간을 측정하였다. 파일 전송 시간은 송신측 SCTP에서 4MB 파일을 전송하기 시작하는 시간으로부터 이 파일이 수신측에 모두 도착하기까지의 시간을 나타낸다. 무작위성을 가정하는 실험

에 대해서는 모두 100번의 시뮬레이션을 수행하여 그 평균 값을 취한 결과를 보였다.

4.2 시뮬레이션 결과 및 분석

4.2.1 멀티호밍 특성에 따른 성능 평가

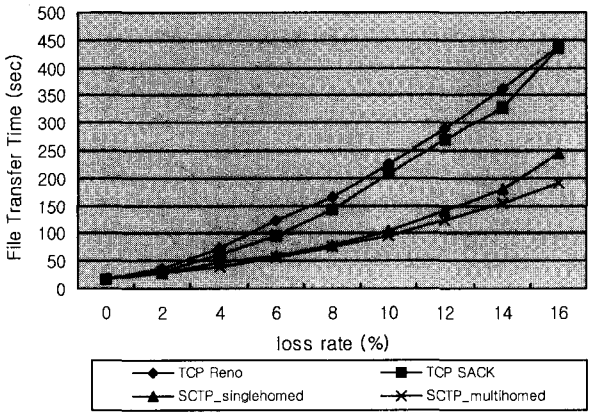
(그림 2)는 path0의 손실율을 0%~1%로 변화시켜 보면서 각 프로토콜의 파일 전송 시간을 측정한 결과를 보인 것이다. TCP Reno, TCP SACK, SCTP_singlehomed는 path0만을 이용하여 전송하며, SCTP_multihomed는 path0를 프라이머리 경로로 사용하고, path1을 재전송을 위해 사용하도록 한다. 이 때 기존의 대부분의 SCTP 성능 연구들에서 가정한 바와 같이[8,9] 재전송 경로인 path1은 패킷 손실이 전혀 없는 경로라고 가정하였다. 패킷 손실율은 어소시에이션이 유지되는 전체 시간동안의 패킷 손실에 대한 것이며, 패킷 손실은 균일 에러 모델(uniform error model)에 따라 발생하도록 하였다.



(그림 2) 4MB file 전송시 패킷 손실율에 따른 지연

path0의 손실율이 0.5%를 넘으면 SCTP_multihomed, TCP SACK, SCTP singlehomed, TCP Reno의 순으로 짧은 파일 전송 시간을 가지고, 또 프로토콜간의 전송 시간의 차이도 커지는 것을 알 수 있다. 이는 호스트가 멀티호밍을 지원하는 SCTP는 패킷 손실율이 높아짐에 따라 대체 경로로의 재전송 횟수가 많아지게 되는데, 재전송 경로를 프라이머리 경로와 동일한 대역폭 및 지연을 가지면서 패킷 손실이 발생하지 않는 경로로 가정하였으므로 하나의 경로만을 이용하는 다른 스킴들에 비해 손실에 대한 빠른 복구가 가능하기 때문이다. 호스트가 멀티호밍을 지원하지 않는 SCTP는 오히려 TCP SACK에 비해 긴 전송 시간을 가지는 것을 볼 수 있는데, 이는 SCTP가 TCP에 비해 상대적으로 긴 헤더 사이즈를 가지기 때문에 전송 오버헤드가 더 크기 때문이다[10]. 동일한 이유로 인해 패킷 손실율이 상대적으로 낮은 0%, 0.1%인 경우에서 TCP 스킴들이 SCTP 스킴들에 비해 짧은 전송 시간을 가진다.

(그림 3)은 path0의 패킷 손실율을 0~16%까지 좀 더 큰 스케일로 변화시켜 본 경우에 대한 결과를 보인 것이다. 패킷의 손실율이 높아질수록 멀티호밍을 사용하는지의 여부에 관계 없이 SCTP 스킴들이 TCP 스킴들에 비해 월등하게 좋은 성능을 보였다. TCP와 마찬가지로 path0만을 사용하는 SCTP도 TCP보다 좋은 성능을 보이므로 이것은 멀티호밍에 의한 성능 차이가 아닌 3장에서 설명한 SCTP와 TCP의 혼잡 제어 메커니즘의 차이에 따른 것이다. 또, 동일하게 SACK 메카니즘을 사용하는 TCP SACK과 싱글호밍 SCTP간의 차이는 멀티호밍이나 SACK 사용 여부에 따른 성능 차이가 아닌 그 밖의 혼잡 제어 방식의 차이에 의한 것으로 볼 수 있다. 손실율이 4%를 넘어서게 되면 호스트가 멀티호밍을 지원하는 경우의 SCTP가 path0만을 사용하는 SCTP보다 더 나은 성능을 보이는데, 이는 프라이머리 경로의 손실이 많은 경우에 재전송 경로를 사용하는 빈도수가 높아지게 되어 두 개의 경로를 사용하게 되는 시간이 길어짐에 따른 것이며, 또 이 실험에서 재전송 경로는 손실이 전혀 없는 경로이므로 빠른 손실 복구가 가능하기 때문이다. 이와 같은 실험 결과를 통해 종합적으로 볼 때, 패킷 손실에 대응하는 SCTP의 혼잡 제어 방식이 TCP보다 더 효율적이며, 완벽한 재전송 경로 가정 하에서는 멀티호밍을 사용하는 SCTP가 그렇지 않은 SCTP에 비해 더 높은 성능을 보임을 알 수 있다.



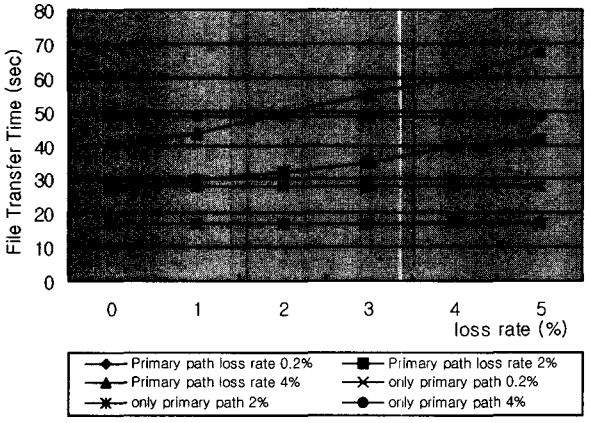
(그림 3) 4MB file 전송시 패킷 손실율에 따른 지연

4.2.2 멀티호밍 호스트의 대체 경로 특성에 따른 성능 평가

이 절에서는 재전송을 위한 대체 경로의 경로 특성을 다양하게 설정해 보면서 SCTP의 성능을 살펴 보고자 한다. SCTP는 멀티호밍 호스트인 경우 프라이머리 경로로 패킷을 전송하고 프라이머리 경로에서 손실된 패킷을 재전송하는 경우에만 대체 경로를 이용한다. 4.2.1절에서는 기존의 SCTP 성능연구에서 일반적으로 해온 가정에 따라 SCTP가 멀티호밍 특성을 이용하는 경우에 있어서 대체 경로는 대역폭, 링크 전파 지연은 프라이머리 경로와 같은 특성을 가지고 패킷 손실율은 0%라고 가정하였는데 이는 실제 인터넷 환경에서 적합하지 않다. 인터넷 환경에서 두 경로는 다른 특성을 지니고 있다고 보는 것이 일반적이며, 따라서 두 경로의 특성이 다른 경우 성능에 어떠한 영향을 주는지

살펴보고자 한다. 이를 위해 대체 경로로 재전송하는 현재의 SCTP 재전송 정책과 재전송을 위해 대체 경로를 이용하지 않고 데이터가 손실되었던 경로로 재전송하는 정책 두 가지를 서로 비교하였다.

먼저 두 경로의 대역폭, 링크 전파 지연은 같고 패킷 손실율이 다른 경우를 살펴보자. (그림 4)는 프라이머리 경로의 패킷 손실율이 각각 0.2%, 2%, 4%인 경우 대체 경로의 패킷 손실율을 0~5%로 변화시켜 보면서 파일 전송 시간을 측정한 결과를 보인 것이다. 프라이머리 경로의 패킷 손실율이 4%일 때를 살펴보면 대체 경로의 손실율이 낮은 경우에는 현재의 재전송 정책을 사용하는 것이 상대적으로 짧은 파일 전송 시간을 가지나 대체 경로의 패킷 손실율이 약 2%를 넘어서면서부터는 처음 전송과 재전송이 같은 경로로 이루어지는 방식이 더 나은 성능을 보인다. 그리고 대체 경로의 손실율이 높아질수록 그 차이가 더 커지게 된다. 프라이머리 경로의 패킷 손실율이 2%인 경우도 유사한 결과를 보이지만, 프라이머리 경로의 패킷 손실율이 4%인 경우에는 대체 경로의 패킷 손실율이 프라이머리 경로 패킷 손실율의 50% 가까이 될 때까지 대체 경로를 사용하여 재전송하는 것이 성능을 향상시키는데 도움이 되었던 것에 반해 프라이머리 경로의 패킷 손실율이 상대적으로 낮은 2%인 경우에는 대체 경로의 패킷 손실율이 프라이머리 경로 패킷 손실율의 25%보다 낮은 경우에만 약간의 성능 향상이 있음을 볼 수 있다.



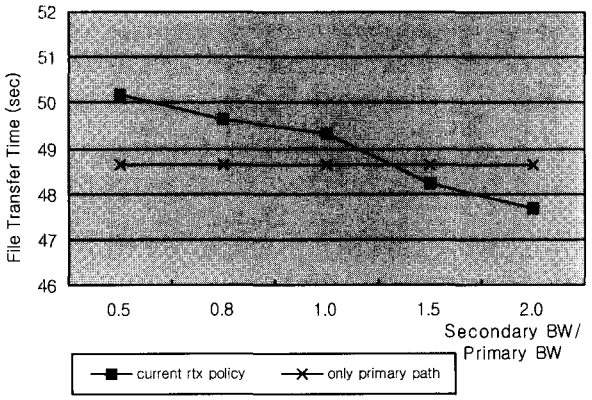
(그림 4) 대체 경로의 패킷 손실율에 따른 지연

이렇게 대체 경로의 패킷 손실율이 프라이머리 경로의 패킷 손실율보다 낮은 때에도 대체 경로를 사용하는 것보다 프라이머리 경로만을 사용하여 재전송하는 것이 짧은 파일 전송 시간을 가지는 것은 SCTP의 패킷 복구 알고리즘과 Karn의 알고리즘에 따른 RTT 측정 방식에 의한 것이다[11]. SCTP는 TCP와 마찬가지로 패킷 손실을 빠르게 복구하기 위해서 빠른 재전송 알고리즘을 사용한다. 그러나, 한번 빠른 재전송이 된 패킷에 대해서는 다시 빠른 재전송을 수행하지 않고, 타임아웃이 된 후에 재전송을 하게 된다[12]. 따라서, 한번 재전송된 패킷이 또다시 손실되는 경우, 이 패킷의 복구는 해당 경로의 RTO 시간에 크게 영향을 받게 된다. 각 경로의 RTO는 해당 경로에서의 RTT에 의해 결정되게 되며, Karn의 알고리즘에 의해 재전송 패킷에 의해 측정된 값은 RTT 측정에 반영되지 않는다. 이로 인해, 재전송을 위해 대체 경로를 사용할 때에 대체 경로에서의 손실로 인해 타임아웃이 발생하여 RTO가 증가하게 되면, 이 증가된 RTO는 오직 목적지가 도달가능한지를 파악하기 위한 하트비트(heartbeat) 패킷에 대한 ACK에 의해서만 업데이트가 가능하다. 그런데, 이 하트비트 패킷은 RFC 2960에서 권고하는 바에 따르면 매 30초마다 +/- 0~15초의 랜덤한 지터에 따라 드물게 보내어지기 때문에 많은 경우에 RTO는 실제 RTT에 비해 매우 큰 값을 가지게 된다. 즉, 타임아웃에 의해서만 재전송이 이루어지는 대체 경로에서의 손실은 길어진 RTO로 인해 복구에 오랜 시간이 걸리게 된다. 반면, 재전송을 위해 프라이머리 경로를 사용하는 경우에 빠른 재전송에 의한 패킷이 다시 손실되었을 때는, 타임아웃 발생으로 RTO가 증가하더라도 프라이머리 경로로 새로운 데이터의 전송이 계속 이루어지므로 새로운 데이터의 ACK에 의해 RTO는 감소하게 된다. 즉, 프라이머리 경로에서의 손실은 대체 경로에서의 손실에 비해 빠른 복구가 가능하다. 따라서, 프라이머리 경로의 패킷 손실율에 비해 대체 경로의 패킷 손실율이 성능 향상을 위해 더 까다로운 조건이 된다.

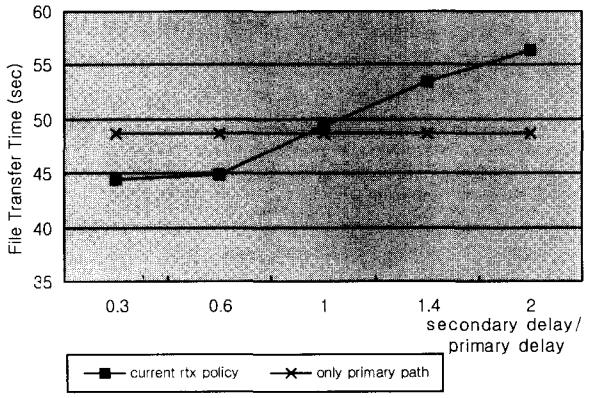
프라이머리 경로의 패킷 손실율이 0.2%인 경우는 프라이머리 경로에서의 패킷 손실율이 낮기 때문에 재전송 경로를 이용하는 빈도 자체가 적으므로 프라이머리 경로로만 재전송을 하는 방식과 전송 시간의 차이가 거의 없다. 위의 결과를 통하여 손실율 측면에서 볼 때 재전송을 위한 대체 경로가 데이터 전송을 위해 사용되고 있는 프라이머리 경로에 비해 신뢰성 있는 경로인 경우에만 두 경로를 동시에 사용하는 것이 성능 향상에 도움이 됨을 알 수 있다.

(그림 5)는 대체 경로의 대역폭을 변화시켜 보면서 파일 전송 시간을 측정한 결과를 보인 것이다. x축은 프라이머리 경로의 병목 링크의 대역폭(2Mbps)에 대한 대체 경로의 병목 링크의 대역폭 비를 표시한 것이다. (그림 4)에서 프라이머리 경로가 4%의 패킷 손실율을 가지는 경우, 대체 경로는 2%의 손실율이 가질 때 프라이머리 경로만을 사용하는 경우와 프라이머리와는 별도의 대체 경로를 이용해 재전송하는 경우의 파일 전송 시간이 거의 같음을 알 수 있다. 따라서 성능에 영향을 주는 파라미터인 손실율의 영향을 배제하기 위해 프라이머리 경로의 손실율 4%, 대체 경로의 손실율 2%인 경우에 대해 대체 경로의 대역폭을 변화시켜 보면서 전송 시간을 측정해 보았다. 손실율 변화에 따른 실험에서와 마찬가지로 대체 경로의 대역폭이 프라이머리 경로의 대역폭에 비해 좋은 경우에만 두 개의 경로를 사용하는 현재의 재전송 정책이 성능에 도움이 된다는 것

을 알 수 있다.



(그림 5) 프라이머리 경로는 4%의 패킷 손실율, 대체 경로는 2%의 패킷 손실율을 가지는 경우, 대체 경로의 대역폭 변화에 따른 지연



(그림 6) 프라이머리 경로는 4%의 패킷 손실율, 대체 경로는 2%의 패킷 손실율을 가지는 경우, 대체 경로의 전파 지연 변화에 따른 지연

(그림 6)은 대체 경로의 전파 지연 변화을 변화시켜 보면서 파일 전송 시간을 측정한 결과를 보인 것으로, x축은 프라이머리 경로의 전파 지연 시간(35ms)에 대한 대체 경로의 전파 지연 시간의 비를 표시한 것이다. 이 경우도 대체 경로의 전파 지연 시간이 프라이머리 경로의 전파 지연 시간에 비해 짧은 경우에만 현재의 재전송 정책을 사용하는 것이 효과적임을 알 수 있다. 대체 경로의 전파 지연에 의한 영향과 대역폭 변화에 따른 영향을 비교해 볼 때, 상대적으로 전파 지연 시간의 차이가 파일 전송 시간 차이에 크게 영향을 주는 요소로 보이는데, 이것은 해당 토폴로지에서 대역폭보다 전파 지연 시간이 RTT에서 차지하는 비중이 크기 때문이다. SCTP 계층에서는 대역폭이나 전파 지연 시간의 차이를 RTT를 통해서만 파악 가능한데, 결론적으로 손실율로 인한 성능 차이가 없다면, 대체 경로의 RTT가 프라이머리 경로의 RTT에 비해 짧은 경우에만 두 개의 경로를 사용하는 것이 효과적임을 알 수 있다. 따라서 SCTP 계층에서는 대체 경로를 재전송 경로로 사용하는 경우에 해당 경로의 RTT를 파악할 수 있다면, 성능 향상에 도움이 되리라 본다.

5. 결론 및 향후 연구 과제

SCTP는 TCP와 마찬가지로 신뢰성 있는 데이터의 전송을 위한 전송 계층 프로토콜이다. SCTP는 TCP의 대부분의 기능을 가지며 거기에 더하여 멀티스트리밍, 멀티호밍 기능을 가진다. 이 논문에서는 TCP에 비해 추가된 기능 중 멀티호밍에 대해 그 특징과 멀티호밍이 성능에 미치는 영향에 대해서 살펴보았다. 먼저 SCTP와 TCP 두 프로토콜의 혼잡 제어 메커니즘에 의한 성능 차이를 보기 위해 SCTP의 경우 싱글홈드 호스트와 TCP Reno 및 TCP SACK의 성능을 비교하였다. 그리고, 멀티홈드 호스트의 경우 프라이머리 경로와 대체 경로간의 경로 특성에 따라 대체 경로를 이용해 재전송을 수행하는 현재의 SCTP 재전송 정책이 SCTP의 성능에 어떠한 영향을 주는지 살펴보았다.

이를 통해 프라이머리 경로와 대체 경로의 손실율, RTT 등의 차이가 성능에 큰 영향을 미치며, 대체 경로가 대역폭이나 전파 지연, 손실율 등에서 프라이머리 경로에 비해 나쁜 특성을 가질 때는 물론 비슷한 특성을 지니는 경우에도 대체 경로를 재전송 경로로써 사용하는 것이 오히려 성능 저하를 가져옴을 알 수 있었다. 따라서 대체 경로를 재전송을 위해 이용할 때에는 대체 경로들의 상태 파악이 중요한 요소가 됨을 알 수 있다. 그러나 RFC 2960에 따르면 대체 경로에 대해서는 매 30초마다 +/- 0~15초 주기로 하트비트 패킷을 통해 경로의 상태를 파악하도록 제안되어 있으며, 이것은 해당 경로의 상태 정보를 얻기에는 부족하므로 이를 위한 다른 방안이 필요할 것으로 보인다. 특히 이러한 점은 프라이머리 경로의 페일로 인해 페일오버가 일어나는 경우 여러 개의 대체 경로들 중에서 데이터를 보낼 새로운 경로를 선택하는 과정에 있어서도 중요하다. 그러나 현재의 SCTP는 이 경우에 대해서도 구체적인 대체 경로 선택 기준이 없이 단지 목적지 리스트에 속하는 경로들에 최대한 다양하게 데이터를 전송하는 것을 목표로 한다. 따라서 대체 경로들의 상태를 파악하기 위한 방안과 파악한 정보를 바탕으로 적절한 재전송 경로를 선택하도록 하는 방안이 제안되어야 할 것이다.

참 고 문 헌

[1] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. "Stream Control Transmission Protocol," Proposed standard, RFC 2960, October, 2000.

[2] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," RFC 2581, April, 1999.

[3] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," ACM Computer Communication Review, July, 1996.

[4] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, October, 1996.

[5] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 2582, April, 1999.

[6] J. Hoe, "Improving the Startup Behavior of a Congestion Control Scheme for TCP," ACM SIGCOMM, August, 1996.

[7] A. Caro, K. Shah, J. Iyengar, P. Amer and R. Stewart, "SCTP and TCP Variants : Congestion Control Under Multiple Losses," Report TR2003-04 CIS Dept, U of Delaware, February, 2003.

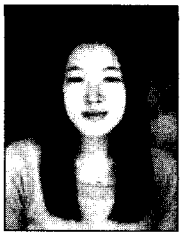
[8] T. Ravier, R. Brennan and T. Curran, "Experimental studies of SCTP multi-homing," First Joint IEI/IEE Symposium on Telecommunications Systems Research 2001, 2001.

[9] A. Jungmaier, E. Rathgeb, M. Schopp, M. Tuxen, "SCTP-A Multi-link End-to-end Protocol for IP-based networks," International Journal of Electronics and Communications, January, 2001.

[10] A. Jungmaier, M. schopp and M. Tuxen, "Performance Evaluation of the Stream Control Transmission Protocol," IEEE ATM Workshop 2000, June, 2000.

[11] A. Caro, P. Amer and R. Stewart, "Transport Layer Multihoming for Fault Tolerance in FCS Networks," CTA 2003, April, 2003.

[12] R. Stewart, L. Ong, I. Arias-Rodriguez, K. Poon, P. Conrad, A. Caro, and M. Tuexen, "Stream Control Transmission Protocol (SCTP) Implementer's Guide,"<draft-ietf-tsvwg-sctpimpguide-08.txt>, Internet Draft, Internet Engineering Task Force (IETF), March, 2003.



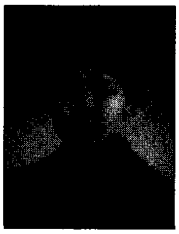
송 정 화

e-mail : jhsong@ewha.ac.kr

1998년~2002년 이화여자대학교 컴퓨터학과 학사

2002년~현재 이화여자대학교 과학기술대학원 컴퓨터학과 석사

관심분야 : SCTP multihoming, Load Sharing, mobile SCTP



이 미 정

e-mail : lmj@ewha.ac.kr

1983년~1987년 이화여자대학교 전자계산학 학사

1987년~1989년 University of North Carolina at Chapel Hill 컴퓨터학 석사

1990년~1994년 North Carolina State University 컴퓨터공학 박사

1994년~현재 이화여자대학교 공과대학 컴퓨터학과 부교수

관심분야 : 고속 통신 프로토콜 설계 및 성능 분석, 멀티미디어 전송을 위한 트래픽 제어, 인터넷에서의 QoS 지원, 무선 이동 네트워크, Ad-hoc 네트워크

고 석 주

e-mail : sjkoh@etri.re.kr

1992년 KAIST 경영과학과 공학사

1994년 KAIST 경영과학과 공학석사

1998년 KAIST 산업공학과 공학박사

1998년~현재 ETRI 표준연구센터 선임연구원

2000년~현재 ITU-T editor(SG17, SSG)

관심분야 : 인터넷 멀티캐스트, IP mobility, Mobile SCTP