SCTP 멀티호밍 특성을 활용한 부하 분산 기법
(A Load-Sharing Scheme using SCTP Multi-homing)

송 정 화 † 이 미 정 ††
(Jeonghwa Song) (Meejeong Lee)

요 약 현재 네트워크에서는 호스트가 다중의 액세스 포인트를 통해 인터넷에 접속될 수 있는 경우를 빈번하게 발견할 수 있다. 본 논문에서는 하나의 사용자 플로우가 사용할 수 있는 가용대역폭을 높이기 위해 이들 다중 인터페이스로의 경로에 효율적으로 부하를 분산하는 방안을 제안하였다. 이를 위해 멀티호밍을 지원하는 새로운 연결기반 전송 계층 프로토콜 표준인 SCTP(Stream Control Transmission Protocol)가 다중 인터페이스로 부하를 분산하도록 확장하였고, 이를 LS(Load Sharing) 모드 서비스라 명명하였다. LS 모드 서비스는 흐름 제어와 혼잡 제어를 분리하였으며, 혼잡 윈도우에 비례하여 각 인터페이스 경로에 데이타를 분배한다. 또한, 특정 경로에서의 손실이 다른 경로에 미치는 영향을 최소화하기 위해 중복적인 패킷 재전송을 하도록 하였으며, SACK이 순서대로 도착하지 않는 경우에도 수신자 윈도우를 제대로 파악할 수 있는 방안을 제안하였다. 이로 인해 LS 모드 서비스는 다중 인터페이스를 사용함으로 인해 발생하는 부작용을 최소화하는 동시에 가용대역폭 향상을 위한 효율적인 부하 분산을 한다. 시뮬레이션을 통해 제안하는 방안이 경로 간 대역폭 차이에 관계없이 두 경로에서 가용한 대역폭의 합에 가까운 작업량을 달성함을 볼 수 있었다. 또한 지연이 두 배까지 되는 경로를 사용할 때에도 단일 경로 사용에 비해 20%의 성능향상을 가져올 수 있음을 보았다.

키워드 : SCTP, 멀티호밍, 부하 분산

Abstract Networks often evolve to provide a host with multiple access points to the Internet. In this paper, we propose a transport layer load distribution mechanism utilizing the multiple network interfaces simultaneously. We specifically propose an extension of Stream Control Transmission Protocol (SCTP) to have load sharing over multiple network interfaces. We named the particular service provided by the proposed load sharing mechanism to be LS (Load Sharing) mode service. LS mode service is based on the following four key elements: (i) the separation of flow control and congestion control, (ii) congestion window based striping, (iii) redundant packet retransmission for fast packet loss recovery, (iv) a novel mechanism to keep track of the receiver window size with the SACKs even if they arrive out-of-order. Through simulations, it is shown that the proposed LS mode service can aggregate the bandwidth of multiple paths almost ideally despite of the disparity in their bandwidth. When a path with a delay of 100% greater is utilized as the second path, the throughput is enhanced about 20%.

Key words : SCTP, Multi-homing, Load Sharing

1. 서 론

현재 네트워크에서는 호스트가 다중의 액세스 포인트를 통해 인터넷에 접속하는 경우를 빈번하게 발견할 수 있다. 고정 호스트의 경우 무선 랜과 유선 랜을 동시에 사용하여 인터넷에 접근할 수 있으며, 이동 호스트의 경우도 다중의 무선 액세스 네트워크를 통해 인터넷에 접근 할 수 있다. 이처럼 다중의 액세스 포인트를 가지는 호스트를 멀티홈드 호스트라고 한다.

멀티홈드 호스트가 네트워크 계층의 중복성을 이용하는 방법은 두 가지로 나눌 수 있다. 첫째는 데이타 스트림 별로 하나의 최적 네트워크 인터페이스를 선택하여 부하를 분산하는 것이고, 다른 방법은 하나의 데이타 스트림의 전송을 위해 여러 개의 네트워크 인터페이스들을 동시에 사용하는 것인데, 본 논문에서는 이 두 번째

· 본 연구는 대학 IT연구센터 육성·지원 사업의 연구결과로 수행되었음
† 비 회 원 : 이화여자대학교 컴퓨터학과
jhsong@ewha.ac.kr
†† 정 회 원 : 이화여자대학교 컴퓨터학과 교수
lmj@mm.ewha.ac.kr
논문접수 : 2004년 1월 7일
심사완료 : 2004년 8월 3일

방법에 관하여 연구하였다. 이 방법은 사용자 플로우가 사용할 수 있는 대역폭을 증대시킬 수 있으며, 특히 하나의 액세스 네트워크로는 응용이 원하는 충분한 대역폭을 제공할 수 없는 때에 유용하다. 또한 특정 인터페이스에 문제가 있거나, 하나의 액세스 네트워크에서 혼잡이 발생하더라도 종단간의 연결이 끊어지지 않도록 해 준다.

실제로, 물리적 네트워크의 중복성을 이용하는 부하 분산은 프로토콜의 여러 계층에서 구현될 수 있다. 그러나 응용 계층에서의 부하 분산은 각 응용이 데이타 스트림을 분산 및 통합할 수 있어야 한다는 제약점을 가진다. 또한 네트워크 계층에서의 부하 분산은 전송 경로 특성에 관한 정보를 유지하지 않기 때문에 효율적인 부하 분산은 할 수 없고, 대부분의 전송 프로토콜이 하나의 연결에 속하는 데이타 스트림은 하나의 경로를 통해 전송하는 것을 가정하므로 오히려 성능의 저하를 가져올 수 있다. 반면, 전송 계층은 일반적으로 혼잡 제어를 위해 전송 경로 특성에 관한 정보를 유지하기 때문에 이들 정보를 사용할 수 있다는 점에서 부하 분산에 적합하다.

이에, 본 논문에서는 TCP, UDP와 함께 인터넷 표준 범용 전송 계층 프로토콜로 최근 채택된 연결 기반 전송 계층 프로토콜인 SCTP(Stream Control Transmission Protocol)를 부하 분산을 위해 확장하고 이를 LS(Load Sharing) 모드 서비스라 명명하였다. SCTP는 멀티호밍을 지원하기 때문에 하나의 사용자 플로우를 다중의 액세스 네트워크로 분산하는 방안을 구현하기에 매우 적합한 프로토콜 플랫폼을 가지고 있다. 다만 현재 SCTP 명세에서는 호스트가 다중 인터페이스를 가지고 있는 경우에 이를 경로 결함 보완의 목적으로 사용하는 방법에 대해서만 기술하고 있으므로[1], 본 논문에서는 LS 모드 서비스를 위해 현재의 SCTP를 확장하고자 하며, 이를 통해 경로 결함 허용뿐 아니라 처리율 측면의 성능 향상도 도모하고자 한다.

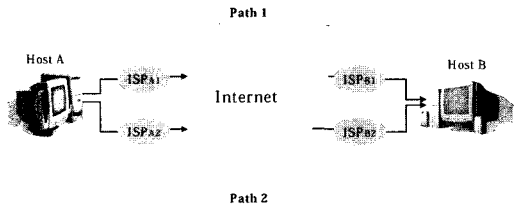

그림 1 다중의 네트워크 연결을 가진 호스트들 간의 통신

그림 1은 다중의 네트워크 연결을 가진 호스트들 간의 통신을 보여주는 것으로 SCTP는 TCP와는 달리 멀티홈드 호스트 간에 설정된 다중의 경로를 모두 어소시에이션[1]에 매핑시킬 수 있다. 현재 SCTP는 이들 경로 가운데 하나만을 프라이머리 경로로 선택하여 그 경로로만 데이타를 전송하고 나머지 경로는 프라이머리 경로에서 손실된 패킷의 재전송이나 프라이머리 경로의 문제 발생시 이를 대체하는 백업용으로만 사용한다[1].

본 논문에서는 하나의 사용자 데이타 스트림을 효율적으로 여러 경로에 나누어 보내는 방안을 제안하며, 각 경로의 대역폭과 지연의 차로 인해 발생할 수 있는 문제를 최소화하여 대역폭 집성 효과를 높여줄 수 있는 방안을 제안한다. 이를 위하여 혼잡 및 오류 제어를 경로 별로 수행하고, 경로 별 혼잡 윈도우에 비례하여 데이타를 분배하는 방안을 제안한다. 또한 손실된 패킷은 중복적으로 재전송함으로써 빠르게 오류를 복구하고 한 경로로의 재전송이 연속적으로 실패하여도 어소시에이션이 계속 지속될 수 있도록 하였으며, 다중 경로 사용으로 인해 ACK 패킷이 순서에 맞지 않게 도착하는 것을 감안하여 송신원에서 수신원의 가장 최근 윈도우 크기를 계산하는 방안을 제안한다. SCTP는 사용자 플로우가 여러 개의 데이타 스트림으로 구성된 경우 이를 지원하는 멀티스트리밍의 특성도 가지는데[1], 본 논문에서 제안하는 방안은 하나의 사용자 플로우가 하나의 데이타 스트림으로 구성된 경우만을 다룬다.

오류 및 혼잡제어에 있어서는 기존의 TCP 오류 및 혼잡제어 방안에 대한 많은 연구들이 있으나 이들은 단일 경로 전송 시 성능 향상을 목적으로 한데 반해[2-8], 본 논문에서 제안하는 혼잡 및 오류제어 방안은 다중 경로를 이용한 전송으로 인해 발생하는 문제 해결에 초점을 둔다는 점에서 차별화 된다.

시뮬레이션을 통해서 제안하는 LS 모드 서비스가 단일 경로 사용이나 네트워크 계층에서의 부하 분산 방안에 비하여 우수한 성능을 보임을 확인할 수 있었다. LS 모드 서비스는 경로 간 대역폭 차에 관계없이 두 경로에서 가용한 대역폭의 합에 가까운 작업량을 달성하였다. 또한, 전파 지연이 두 배까지 차이가 나는 경로를 사용할 때에도 단일 경로 사용에 비해 20%의 성능향상을 가져 올 수 있었으며, 단일경로를 사용하는 경우보다 작업량이 더 적어지는 최악의 경우는 발생하지 않았다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 다중의 인터페이스를 동시에 이용하여 대역폭을 향상시키는 위해 기존에 연구되어온 방안들을 계층별로 살펴보고, 3장에서는 제안하는 SCTP 멀티호밍을 이용한 대역폭 향상 방안을 자세히 설명한다. 4장에서는

1) 연결기반의 전송 프로토콜인 SCTP는 두 지점간의 연결을 어소시에이션(association)이라 한다.

시뮬레이션을 통하여 제안하는 방안과 타 계층 방안들의 성능을 비교·분석한다. 마지막으로 5장에서는 본 논문의 결론을 기술한다.

2. 관련 연구

다양한 계층에서 다중의 네트워크 인터페이스를 동시에 이용하여 사용자에게 높은 대역폭을 제공하는 방안이 기존에 많이 연구되어 왔다. 이 절에서는 제안되어 온 방안들을 계층별로 간략하게 살펴보고 그 제약점 및 제안하는 방안의 차별성을 설명한다.

• 응용 계층에서의 대역폭 향상 방안

응용 계층에서의 가용 대역폭 증대 방안으로는 다중의 TCP 커넥션을 병렬적으로 사용하는 방안이 많이 연구되어 왔다. 즉, 사용자의 응용 계층에서 네트워크에서 가용한 인터페이스에 대해 각각 소켓을 열어 이 인터페이스에 데이타를 분산하고 수신자 응용 계층에서 이를 재순서화한다. [9]에서는 여러 개의 TCP 소켓을 생성하여 데이타를 분배하는 데에 사용하는 PSockets library를 제안하여 더 높은 네트워크 대역폭을 얻고자 하였다. 그러나 응용 계층에서는 이용 가능한 대역폭을 정확하게 측정하는 데에 어려움이 있으며, 사용자 응용에서 데이타를 분산하고 재순서화하는 오버헤드와 함께 각 경로의 이용 가능한 대역폭에 차이가 있을 때 수신된 데이타를 재순서화하기 위한 커다란 수신 버퍼를 가져야 한다는 단점을 가진다. [10]에서는 동일한 경로에 여러 개의 TCP 커넥션을 설립함으로써 발생하는 네트워크 자원 사용의 불공정 문제를 최소화하면서 송수신자간의 전체 성능은 최대화하는 방안을 제안하였다. 그러나 이 방안 또한 [9]에서와 마찬가지로 커다란 수신 버퍼를 가져야 하며, 효율적인 데이타 분배와 재순서화 등을 위해서는 전송 계층과 중복된 많은 기능들이 각 응용에 요구된다.

• 전송 계층에서의 대역폭 향상 방안

R-MTP(Reliable Multiplexing Transport Protocol)는 무선 환경에서 이동 호스트가 좀 더 높은 대역폭을 얻을 수 있도록 디자인된 율 기반(rate-based) 전송 계층 프로토콜이다[11]. R MTP는 여러 개의 채널을 동시에 사용하는 방안으로, 전송 계층이 이용 가능한 채널들과 각 채널의 이용 가능한 대역폭에 대한 정보 등 링크 계층의 정보를 가지도록 한다. 이를 위해 R-MTP는 주기적으로 프로빙 패킷을 보내어 이용 가능한 대역폭을 측정하고 이 명시적인 대역폭에 기반해서 데이타를 나눈다. 따라서 R MTP는 대역폭 측정의 정확도가 매우 중요한 요소가 된다. 만약 대역폭을 프로빙하는 주기보다 짧은 시간 안에 대역폭이 크게 변동하게 되면, R MTP는 성능 저하를 경험하게 된다. 그러나 이를 방지하기 위해 짧은 주기로 프로빙을 하게 되면 프로빙 패킷으로 인해 네트워크 자원이 소비되며, 그밖에도 R-MTP는 무선 환경에 적합하게 디자인되어 유선 환경에는 적합하지 않다는 문제점을 지닌다.

[12]에서는 하나의 사용자 플로우를 위하여 전송계층에서 여러 개의 TCP 커넥션을 생성하고 각각의 커넥션은 하나의 TCP 커넥션처럼 행동하도록 하며, 이 커넥션들에 데이타를 나누고 이들을 전송 계층에서 관리하도록 하는 구조를 가진 pTCP라는 프로토콜을 제안했다. pTCP는 경로 별 혼잡 윈도우 크기에 따라 데이타를 분배하며, 어떤 경로에 혼잡이 발생할 때에는 그 경로에 이미 할당한 데이타를 다른 경로에 재배치 할 수 있도록 하였다. 그리고 하나의 경로에서 발생한 손실에 의해 전체 성능이 저하되지 않도록 빠르게 손실을 복구하기 위해 SACK 메커니즘을 사용한다. 이 방안은 제안하는 방안과 유사하나 동시에 여러 개의 경로로의 전송이 가능할 때에 순차적으로 경로들을 돌아가며 사용하되 순서가 된 경로의 윈도우가 허용하는 만큼을 모두 그 경로를 통해 전송하기 때문에, 제안하는 방안에 비하여 모든 경로들을 공정하게 동시에 사용하지 못한다. 또한 혼잡 시 이미 분배한 데이타를 재분배함으로 인해 불필요한 재전송이 발생할 수 있으며, 이는 성능 저하를 가져 올 수 있다. 그리고 제안하는 방안에서는 빠른 재전송을 위해 SACK과 더불어 중복 전송을 사용하는데 반해, pTCP에서는 SACK에만 의존한다. 또한, pTCP가 동작하기 위해서는 각 패킷마다 추가적인 헤더가 필요하다.

• 네트워크 계층에서의 대역폭 향상 방안

[13]에서는 특정 TCP 커넥션에 속하는 데이타 스트림과 같은 하나의 사용자 데이타 스트림을 네트워크 계층에서 다중의 경로에 분배하기 위하여 여러 개의 인터페이스로 패킷을 전송하도록 라우팅 엔터티를 조작하는 방안을 제안하였다. TCP 커넥션의 경우에는 처음 커넥션을 맺은 인터페이스들 간의 통신만이 가능하므로, 네트워크 계층에서 이와 다른 제 2 혹은 제 3의 인터페이스 식별자를 송신자, 혹은 수신자 주소로 사용하기 위해서는 먼저 처음 커넥션을 맺은 인터페이스의 주소로 TCP 세그먼트를 인캡슐레이션하고, 제 2 혹은 제 3의 새로운 인터페이스 식별자로 이를 터널링한다. 또한 송수신자가 서로의 IP 주소들을 파악하도록 하기 위해 IP 헤더 혹은 TCP 헤더, 그리고 IP/TCP 헤더의 프로세싱 방식 등을 수정하는 방안을 제안하였다.

그런데 네트워크 계층에서의 분배 방안은 TCP와 같은 상위 계층의 동작을 고려하지 않기 때문에, 대역폭, 전파지연 등의 경로 특성이 다른 경우에 타임아웃이나 빠른 재전송이 발생할 수 있고, 이런 경우에는 하나의

경로를 사용하는 경우보다 오히려 성능 면에서 나쁜 결과를 가져오게 한다. [13]에서는 경로들 간 대역폭 차가 있는 경우 이로 인한 타임아웃 발생 문제를 완화시키기 위해 대역폭 차이 정도에 따라 재전송 타이머 계산에 이용되는 TCP 파라미터를 조정하는 방안을 제안하였고, 각 경로로 내보내는 패킷 크기를 대역폭 비에 따라 조절하는 것이 도움이 됨을 보였다. 또한 송신측에서 경로들의 대역폭 비에 따라서 out-of-order로 패킷을 전송함으로써 경로들 간 대역폭 차에 의한 잘못된 빠른 재전송 문제를 완화시킬 수 있음을 보였다. 그러나 이들 방안을 실제적으로 적용하기 위해서는 네트워크 계층에서 각 경로들의 대역폭을 파악할 수 있어야 하며, 대역폭 비에 따라 각 경로로 내보내는 패킷 크기를 조절하기 위한 방안 및 out-of-order 전송을 위한 구체적인 방안이 요구되는데, [13]에서는 이와 같은 방안들을 제시하지 않고 있다.

3. LS 모드 서비스를 위한 SCTP 수정 방안

이 절에서는 먼저 제안하는 LS 모드 서비스의 주된 디자인 요소들을 설명하고, LS 모드 서비스 추가를 위한 SCTP 확장에 대하여 자세히 설명한다.

3.1 주요 요소

LS 모드 서비스는 다음 네 가지 주요 정책에 기반해 디자인되었다.

- **경로별 오류 제어:** 현재 SCTP의 혼잡 제어는 경로 별로 수행하고, 오류 및 흐름 제어는 어소시에이션 별로 수행한다. 그런데 LS 모드 서비스에서는 기존의 SCTP와는 달리 여러 경로를 동시에 사용하여 전송하기 때문에 한 경로에서의 오류 제어가 다른 경로의 혼잡 제어에 영향을 미치지 않도록 하기 위해 혼잡 제어와 함께 오류 제어도 경로 별로 수행한다. 오류 제어를 경로 별로 수행하지 않는다면 A라는 경로에 할당된 패킷이 손실되었을 때 이에 대한 재전송이 B라는 경로를 통해 성공적으로 수행되어도 경로 A에서 그 패킷에 대한 재전송이 성공적으로 수행되었다고 간주하게 되어 경로 A의 혼잡 제어가 경로 A의 '혼잡' 상태를 제대로 파악할 수 없게 되기 때문이다. 따라서 LS 모드 서비스에서는 각 경로에서 그 경로에 할당된 패킷들의 집합에 대하여 패킷이 할당된 순서에 따라 순차적으로 성공적인 배달이 이루어지도록 경로 별로 오류 제어를 수행하고 이와 병행하여 경로 별 혼잡 제어를 수행한다.
- **혼잡 윈도우에 비례한 경로 별 데이타 분배:** LS 모드 서비스에서는 SCTP의 구조를 어소시에이션 전체를 제어하는 어소시에이션 관리 모듈(Association Management Module)과 각 경로들을 제어하는 경로 관리 모듈(Path Management Module)로 분리한다. 어소시에이션 관리 모듈은 경로 관리 모듈로부터 경로 정보 업데이트 메시지를 받을 때마다 경로 정보 업데이트를 수행하고, 경로 별 혼잡 윈도우에 근거한 데이타 분배 알고리즘을 수행하여 사용자 응용의 데이타를 다중 경로에 할당한다. 이는 혼잡 윈도우가 경로의 대역폭과 지연 프로덕트를 반영한다고 가정한 것이다.
- **중복적인 패킷 재전송:** LS 모드 서비스의 수신측에서는 다중의 경로를 통해 배달되는 패킷을 하나의 수신 버퍼에 저장하여 정렬하고 이를 사용자 응용에 배달한다. 따라서 손실된 패킷의 복구가 느려지게 되면 수신자 윈도우는 손실이 없는 다른 경로들의 전송에 의해 채워지게 되어 더 이상 어소시에이션의 어떤 경로도 패킷을 전송할 수 없게 되는 상황이 발생하게 된다. 이를 피하기 위해 LS 모드 서비스에서는 손실된 패킷이 신속히 복구되도록 다중의 경로에 중복적으로 전송한다.
- **수신자 윈도우 크기 계산:** 데이타 패킷과 마찬가지로 데이타 수신자로부터 전송되는 ACK 패킷 역시 서로 다른 경로를 통해 전달되므로 데이타 수신자가 보낸 순서와 다른 순서로 데이타 송신자에게 도착하는 경우가 빈번히 발생한다. 따라서 가장 최근에 수신한 ACK 패킷이 데이타 수신측에서 가장 나중에 발생한 ACK 패킷이 아닐 수도 있다. 즉, 가장 최근에 받은 ACK에 실려 있는 수신자 윈도우 크기 정보를 이용하여 흐름 제어를 할 수 없으므로 LS 모드 서비스에서는 수신자가 보낸 ACK이 가장 최근 업데이트된 내용임을 확신할 수 있을 때에만 수신자로부터 받은 ACK 정보를 이용하여 수신자 윈도우 크기를 업데이트하도록 한다.

3.2 SCTP 프로토콜 개요 및 패킷 헤더 수정

LS 모드 서비스에서는 각 경로의 상태에 따라 모든 이용 가능한 경로에 사용자 데이타 스트림을 전송한다. 이를 위해 SCTP 프로토콜 패킷 헤더의 수정이 요구된다. 자세한 설명에 앞서 먼저 본 논문에서 사용되는 용어들을 정의한다. 다음 정의들은 SCTP 명세인 RFC-2960에 따른다.

- **청크(Chunk):** SCTP 패킷 내에 들어가는 정보의 단위로써 제어를 위한 청크와 사용자 데이타를 포함하는 청크로 분류됨; 하나의 SCTP 패킷은 여러 개의 청크를 포함할 수 있음
- **Rwnd(Receiver Window):** 수신자의 수신 가능한 버퍼의 크기를 나타내는 변수로 전체 어소시에이션에 대해 하나의 변수 유지
- **Cwnd(Congestion Window):** 전송되는 데이타의 양을 제한하기 위한 혼잡 윈도우 변수로 목적지 주소 각각에 대해 유지
- **TSN(Transmission Sequence Number):** SCTP에서 사용자 데이타 청크에 대해서 어소시에이션

레벨에서 순차적으로 부여하는 번호

- **SID(Stream IDentifier)**: SCTP 어소시에이션 내에서 사용자 데이타가 속하는 스트림을 구분하기 위한 식별자
- **SSN(Stream Sequence Number)**: SCTP 특정 스트림 내에서 사용자 데이타에 순차적으로 부여하는 번호

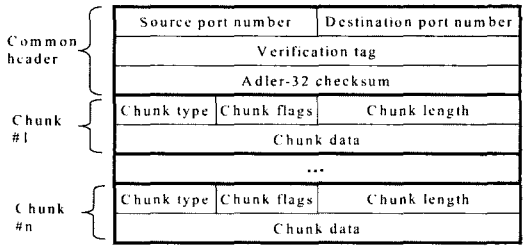

그림 2 SCTP 패킷

일반적으로 SCTP 패킷은 그림 2와 같이 공통 헤더와 하나 이상의 청크들로 이루어지며, 하나의 SCTP 패킷은 제어 청크와 서로 다른 사용자 스트림에 속하는 데이타 청크들로 구성되는 것이 가능하다. LS 모드 서비스에서는 하나의 사용자 스트림을 전송하므로 SCTP 패킷은 하나의 청크로 이루어지는 것을 가정하며, 패킷의 크기는 각 경로들의 MTU 값 중 가장 작은 값으로 정하여 다른 경로로 재전송 할 때 패킷이 분할되지 않도록 한다.

또한, 데이타 전송을 위한 청크 헤더 필드인 Stream ID, TSN, SSN를 이용하는 방식에 있어서도 수정이 요구된다. SCTP 명세에서 Stream ID는 데이타 청크가 속하는 사용자 스트림을 나타낸다. 그리고 TSN과 SSN은 각 데이타 청크에 대한 번호로써, TSN은 어소시에이션의 신뢰성과 프라이머리 경로의 혼잡 제어를 위해 사용하며, SSN은 수신측에서 사용자 스트림 내에서의 순서에 맞게 응용에 전달하기 위해 사용된다.

이에 반하여, LS 모드 서비스에서의 Stream ID는 데이타 청크 혹은 패킷이 전송된 경로를 나타내는데, 같은 경로로 배정되는 패킷들을 LS 서비스 모드 스트림이라 부르기로 한다. 이것은 어소시에이션을 이루는 사용자 레벨의 스트림과는 구별되는 것으로, 이 후 본 논문에서 스트림이라 하면 특별한 부연 설명이 없는 한 LS 서비스 모드 스트림을 의미하는 것이다. LS 모드 서비스에서 사용자 스트림은 어소시에이션 관리 모듈에 의해 다중의 LS 서비스 모드 스트림으로 나누어지고, 각 LS 서비스 모드 스트림은 다른 경로를 통해 전송된다. TSN은 사용자 스트림의 순서를 맞추기 위해 사용되며,

Type = 3	Chunk Flags	Chunk Length
Cumulative TSN Ack		
Advertised Receiver Window Credit (a_rwnd)		
Number of Gap Ack Blocks = N		Number of Duplicate TSNs = X
Gap Ack Block #1 Start		Gap Ack Block #1 End
...		
Gap Ack Block #N Start		Gap Ack Block #N Start
Duplicate TSN 1		
...		
Duplicate TSN X		

그림 3 현재 SCTP SACK format

Type = 3	Chunk Flags	Chunk Length
Cumulative TSN Ack		
Cumulative SSN Ack		
Advertised Receiver Window Credit (a_rwnd)		
Number of Gap Ack Blocks = N		Number of Duplicate SSNs = X
Gap Ack Block #1 Start		Gap Ack Block #1 End
...		
Gap Ack Block #N Start		Gap Ack Block #N Start
Duplicate SSN 1		
...		
Duplicate SSN X		

그림 4 LS 모드 서비스를 위한 SACK format

SSN은 LS 모드 서비스 스트림에 대한 신뢰성과 혼잡 제어를 위해 사용된다.

SACK 포맷은 그림 4와 같이 수정되었으며, 먼저 현재의 SACK 포맷인 그림 3에서 각 필드의 역할을 살펴보면 다음과 같다. cumulative TSN Ack 필드는 갭 없이 연속적으로 수신받은 청크들 중 가장 높은 TSN번호를 나타내며, Advertised Receiver Window Credit (a_rwnd) 필드는 송수신자간의 어소시에이션에 대해 하나만을 유지하는 수신자 버퍼의 이용 가능한 공간의 크기를 나타낸다. Number of Gap Ack Blocks 와 Number of Duplicate TSNs 필드는 각각 현재 SACK 내에 포함되어 있는 Gap Ack Block의 수와 중복 TSN의 수를 나타낸다. Gap Ack block # Start/End 필드는 갭이 있는 경우 수신된 청크들의 블록을 나타내며, Cumulative TSN Ack으로부터의 오프셋으로 표시한다. Duplicate TSN 필드는 중복적으로 받은 TSN의 번호를 기록하는 필드이나 현재 SCTP에서는 이를 이용하지 않는다.

LS 모드 서비스를 위해서 그림 3의 기존 SCTP SACK 필드 가운데 TSN 번호를 사용하는 필드들을 모두 SSN 번호를 사용하도록 수정한다. 이것은 혼잡 및 오류 제어를 경로별로 수행하기 위해서이다. 단,

cumulative TSN Ack 필드는 흐름 제어를 위해 필요하므로 그대로 사용하고, Cumulative SSN Ack 필드를 추가하였다. 또한 Gap Ack Block #Start/End는 Cumulative SSN Ack으로부터의 오프셋으로 표시된다. 즉, 수신자는 특정 SID의 SSN에서의 홀이 있을 때에만 SACK 메시지의 Gap Ack Block 필드를 통해서 송신자에 손실을 알리도록 한다. 다시 말해, TSN 번호에 홀이 있더라도 SSN번호에서 홀이 없다면 이것은 손실이 아니라 서로 다른 특성을 가진 경로를 동시에 사용하는 것에 기인한 것으로 간주한다.

한편 수신자가 전송 받은 패킷들을 상위 계층으로 올려 보낼 때는 SSN이 아니라 TSN 번호 순서로 올려 보낸다. 이는 패킷들이 서로 다른 스트림에 속하지만 어소시에이션 레벨에서 볼 때, 각 패킷은 모두 하나의 사용자 스트림에 속하므로 수신자 측에서 원래 송신자의 응용 계층으로부터 내려온 순서에 따라 상위 계층으로 데이타를 올려 보내도록 하기 위함이다.

3.3 구조 및 동작과정

그림 5는 LS 모드 서비스 구현을 위한 SCTP 전체 구조를 나타낸 것이다. LS 모드 서비스에서는 SCTP 구조를 어소시에이션 전체를 제어하는 어소시에이션 관리 모듈(Association Management Module)과 각 경로들을 제어하는 경로 관리 모듈(Path Management Module)로 분리하였으며, 흐름 제어는 어소시에이션 별로 두는 어소시에이션 관리 모듈에서 수행하고, 오류 및 혼잡 제어는 각 경로 별로 독자적으로 두는 경로 관리 모듈에서 수행한다. 송신측 어소시에이션 관리 모듈은 사용자 응용의 데이타를 다중의 경로에 할당한다. 수신측 어소시에이션 관리 모듈은 데이타 패킷의 전송 경로에 관계없이 하나의 버퍼에 수신하여 순차적으로 정렬

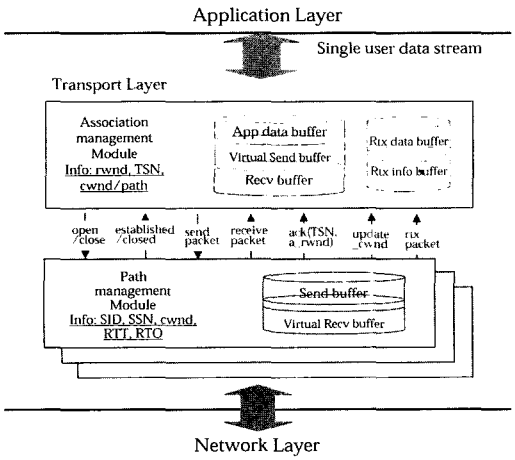

그림 5 LS 모드 서비스를 구현을 위한 SCTP 구조

하고 사용자 응용에게 전달한다. 송신측 어소시에이션 관리 모듈은 수신측 어소시에이션 관리 모듈의 버퍼가 허용하는 한도 내에서 전송할 데이타양을 결정한다. 반면, 각 경로 별 경로 관리 모듈은 서로 독립적으로 동작하며 기존의 SCTP와 마찬가지로 통신하고 있는 상대방 경로 관리 모듈과 데이타 패킷, ACK 패킷을 주고받으면서 해당 경로에 대한 혼잡 제어와 오류 제어를 수행한다.

SCTP 사용자 인터페이스는 사용자 응용이 SCTP 어소시에이션 설립을 요청할 때 기존의 SCTP 서비스와 LS 모드 서비스 중 어느 것을 이용할 것인지 선택할 수 있도록 해 주어야 하며, 통신하는 두 SCTP 엔터티의 어소시에이션 관리 모듈은 어소시에이션 설립 시에 이를 서로 협상하여야 한다. 이를 위해 SCTP 어소시에이션 설정을 위한 Init Chunk와 Init ACK Chunk의 Chunk Flags 필드를 이용하도록 한다. 이 필드는 현재 SCTP에서 사용되지 않고 있는 필드로써, 이 필드에 값을 부여하고 송수신자가 이를 프로세싱 하도록 수정함으로써 송수신자는 LS 모드 서비스를 수행할 것을 협상할 수 있다. 만약 통신하는 두 SCTP 엔터티 중 하나가 LS 모드 서비스를 지원하지 않는다면 이들 간의 통신은 현재 SCTP 방식으로만 이루어질 수 있다. 어소시에이션의 설립 후, 어소시에이션 관리 모듈은 이용 가능한 송수신자간의 네트워크 인터페이스 쌍 각각에 대해 경로 관리 모듈을 만든다.

먼저 송신측의 동작과정을 살펴보면, 어소시에이션 관리 모듈은 사용자 응용의 데이타가 저장되어 있는 App data buffer와 재전송 데이타가 저장되어 있는 Rtx data buffer를 확인하여 전송할 데이타가 있는 경우에 각 경로로 패킷을 분배한다. 패킷 분배는 어소시에이션 관리 모듈이 유지하고 있는 rwnd 정보와 경로 관리 모듈로부터 전달받은 cwnd 정보를 이용하여 이루어진다. 만약 App data buffer와 Rtx data buffer 모두 데이타가 없다면, 더 이상 전송할 데이타가 없는 경우로 어소시에이션 관리 모듈은 사용자 응용으로부터 새로운 데이타를 받을 때까지 또는 재전송이 필요한 데이타가 발생할 때까지 패킷 분배 작업은 중지하고 경로 관리 모듈의 지시에 따라 cwnd 및 rwnd의 업데이트만을 수행한다.

어소시에이션 관리 모듈은 우선적으로 Rtx data buffer에 있는 데이타를 전송하고 난후 App Data buffer에 있는 데이타를 전송한다. 이는 재전송이 우선적으로 이루어지도록 하기 위해서이다. Rtx data buffer에 있는 저장되어 있는 패킷들을 전송할 경로는 이들 패킷에 대한 이전 전송 경로 히스토리 정보를 가지고 있는 Rtx info buffer를 이용하여 결정하는데, 이에 대

해서는 3.6절의 오류 제어에서 자세히 다루고자 한다. App data buffer로부터 가져온 데이타는 경로 관리 모듈로 보내기 전에 SCTP 패킷을 만들기 위해 공통 헤더와 데이타 청크 헤더를 붙인다. 이 때, 데이타 청크 헤더 중 SID, SSN은 값을 배정하지 않은 채로 보내며, 이 값은 후에 경로 관리 모듈에 의해서 부여된다. 그리고 어소시에이션 관리 모듈은 흐름 제어를 위하여 전송 경로에 할당한 패킷의 TSN을 Virtual Send buffer에 저장한다.

경로 관리 모듈은 통신하고 있는 상대방과 데이타 패킷 및 ACK 패킷을 주고받으면서, 해당 경로에 대해 RTT(Round Trip Time), RTO(Retransmission Time-Out), cwnd와 같은 정보를 유지하고 혼잡 제어 및 에러 복구를 수행한다. 경로 관리 모듈은 어소시에이션 관리 모듈로부터 받은 데이타 패킷의 청크 헤더에 해당 경로의 SID, SSN 값을 부여하고 send buffer에 복사본을 저장한 후 해당 인터페이스로 패킷을 내보낸다. 또한 SACK 패킷을 수신하면 프로세싱을 통해 혼잡 제어, 에러 제어, send buffer로부터 ACK된 패킷 삭제 등의 일을 수행한다. 그리고 변화된 cwnd 정보, cumulative TSN, ACK된 패킷의 TSN 및 현재 수신측이 수신 받을 수 있는 버퍼 크기를 나타내는 a_rwnd 정보를 어소시에이션 관리 모듈에 알리며, 빠른 재전송이나 타임아웃에 의해 재전송이 필요한 경우에는 경로 관리 모듈의 send buffer에서 해당 패킷에 재전송이 필요함을 표시하고, 표시된 패킷의 복사본을 어소시에이션 관리 모듈로 보내어 Rtx data buffer에 저장하도록 한다.

수신측의 경로 관리 모듈이 패킷을 수신하면, 패킷의 헤더를 분리한 다음 헤더는 virtual recv buffer에 저장하고, 데이타는 TSN과 함께 어소시에이션 관리 모듈로 보낸다. 그 후, 경로 관리 모듈은 virtual recv buffer의 정보를 이용하여 SACK 패킷을 생성해 송신자에게 전송한다. 동시에 어소시에이션 관리 모듈은 여러 경로 관리 모듈로부터 전송 받은 데이타를 TSN 순으로 recv buffer에 저장하고, TSN에서의 첫 번째 홀 직전까지 사용자 응용으로 올려 보낸다.

이러한 어소시에이션 관리 모듈과 경로 관리 모듈 간의 상호작용을 위한 인터페이스는 그림 5에서 보이는 것과 같이 다음 여덟 가지 함수로 이루어진다: open(), close(), established(), closed(), send(), receive(), rtx(), ack(TSN, a_rwnd), update_cwnd(). 어소시에이션 관리 모듈은 open()과 close() 함수를 호출하여 해당 경로 관리 모듈을 생성 및 해제하며, 경로 관리 모듈은 established()와 closed() 함수를 호출하여 경로 관리 모듈이 생성 및 해제가 이루어졌음을 어소시에이션 관리 모듈에게 알린다. 또한 어소시에이션 관리 모듈은 send() 함수를 호출하여 데이타를 경로 관리 모듈로 전달하며, receive() 함수를 통해 경로 관리 모듈로부터 수신된 데이타를 전달 받는다. 송신자의 경로 관리 모듈은 ACK을 수신하면 ack() 함수를 통해 a_rwnd 정보와 TSN을 어소시에이션 관리 모듈에 전달하며, update_cwnd() 함수를 호출하여 변화된 혼잡 윈도우 값을 알린다. 만약 재전송이 필요하다면, rtx() 함수를 호출하여 재전송이 필요한 패킷을 어소시에이션 관리 모듈로 전달한다.

3.4 데이타 스트림 분배 알고리즘

다중 경로를 동시에 이용하여 대역폭 및 처리율을 향상시키고자 하는 LS 모드 서비스를 위해서는 이들 다중 경로에 대해 사용자 데이타 스트림을 효율적으로 분배하는 알고리즘이 요구된다. 이를 위해 LS 모드 서비스에서는 혼잡 윈도우 기반의 분배방안을 제안한다.

그림 6은 데이타 스트림 분배 알고리즘을 pseudo

$cwnd_i$: 경로 i의 현재 혼잡 윈도우 사이즈를 패킷수로 나타낸 값
rwnd : 어소시에이션 전체의 현재 수신자 윈도우 사이즈를 패킷수로 나타낸 값
UA: 전송 계층이 받은 사용자 응용의 데이타 양
a_rwnd (available receiver window size): 이용 가능한 수신자 윈도우 사이즈
HTSN : 전송한 TSN 중 가장 높은 값
CTSN : Cumulative Acked TSN
[a]: a를 반올림한 정수 값

$$rwnd = \left\lfloor \{\min(a\_rwnd, UA) - (HTSN - CTSN)\} / chunksize \right\rfloor$$

```
if (rwnd > 0) {
  sum_cwnd = Σ_pathi cwnd_i;
  if (rwnd >= sum_cwnd)
    for (all path i) send_i = cwnd_i;
  else
    for (all path i) send_i = [cwnd_i / sum_cwnd * rwnd];}
```

그림 6 데이타 스트림 분배 알고리즘

code로 보인 것이다. 데이타 스트림의 분배는 수신자 윈도우(a_rwnd)와 혼잡 윈도우(cwndi) 정보를 이용하여 이루어진다. 수신자 윈도우는 수신자가 수신 받을 수 있는 버퍼 크기인 수신자 윈도우와 전송 계층이 유지하고 있는 사용자 응용의 데이타 크기 중 작은 값으로 정해진다. 새로운 패킷의 전송은 이렇게 계산된 수신자 윈도우 내에서 이루어지며, 각 경로의 혼잡 윈도우 크기에 비례하여 각 경로들에 분산된다. 수신자 윈도우가 모든 경로의 혼잡 윈도우의 합 이상으로 존재한다면 각 경로에 혼잡 윈도우만큼의 패킷을 전송하도록 하고, 수신자 윈도우가 충분치 못한 경우에는 수신자 윈도우가 허용하는 만큼의 패킷을 각 경로의 혼잡 윈도우 비에 따라 전송한다. 이것은 수신자 윈도우의 제약으로 인해, 즉, 수신자 버퍼나 사용자 응용의 데이타가 충분치 않아서 각각의 경로에 혼잡 윈도우만큼의 패킷을 전송할 수 없을 때에도 데이타가 특정 경로에만 집중되는 것을 피할 수 있도록 한다.

3.5 흐름 제어 알고리즘

현재 SCTP 흐름 제어 알고리즘에서는 송신원측에서 수신자가 보낸 SACK의 a_rwnd 필드와 Cumulative TSN Ack 필드를 이용하여 다음과 같이 내보낼 수 있는 데이타의 양을 결정한다.

$$Receiver\ Window = available\ receiver\ window - amout\ of\ outstanding\ packets$$

그런데 LS 모드 서비스에서는 경로 별 혼잡제어를 수행하기 위하여 수신측에서 SACK 패킷을 보낼 때, 해당 데이타 패킷이 수신된 인터페이스로 이를 내보내므로 SACK 패킷 역시 데이타 패킷과 마찬가지로 서로 다른 경로를 통해 전달된다. 따라서 SACK 패킷이 데이타의 수신자가 보낸 순서와 다르게 송신자 측에 도착하는 현상이 빈번하게 발생하기 때문에, 현재의 흐름 제어 방식을 그대로 사용한다면 송신자가 수신자 버퍼 크기 이상으로 전송을 허용하여 심각한 경우 어소시에이션이 끊어지는 현상이 발생한다. 이 문제를 해결하기 위해서 LS 모드 서비스에서는 수신자가 보낸 SACK이 가장 최근 업데이트된 내용임을 확신할 수 있을 때에만 SACK내의 a_rwnd 정보를 이용하도록 한다.

어소시에이션 관리 모듈은 패킷을 전송하기 위해 경로 관리 모듈로 전달할 때 Virtual Send buffer에 패킷의 TSN을 저장하며, 해당 TSN에 대한 ACK을 받았을 때 버퍼로부터 삭제한다. 경로 관리 모듈은 SACK 패킷을 받게 되면 ack된 패킷을 경로 관리 모듈의 Send buffer로부터 삭제하는데, Send buffer내의 패킷은 TSN, SID, SSN 정보를 모두 가지므로 ack된 패킷의 SID와 SSN에 해당하는 TSN을 알 수 있게 된다. 경로 관리 모듈은 이렇게 파악한 ack된 패킷의 TSN과 SACK내의 필드인 cumulative TSN, a_rwnd 정보를 어소시에이션 관리 모듈에게 알린다. 어소시에이션은 자신의 Virtual Send buffer를 통해 현재 유지하는 cumulative TSN보다 새로운 SACK의 cumulative TSN 값이 더 크거나, cumulative TSN은 이전과 같으나 추가적으로 ack된 패킷이 존재하는 경우에만 수신한 SACK의 a_rwnd 정보를 이용한다. 만약 이 조건을 만족하지 않으면 수신자 윈도우 크기를 업데이트하지 않는다. 이를 통해 수신자 윈도우를 계산하면 다음과 같다.

$$Receiver\ Window = \frac{available\ receiver\ window}{chunk\ size} - (Highest\ TSN\ that\ has\ been\ ever\ sent - Highest\ Cumulative\ TSN\ acknowledged)$$

3.6 오류 제어 및 재전송 정책

현재의 SCTP에서는 재전송 패킷이 수신자에 도착할 확률을 높이기 위해 패킷 전송이 실패하면 처음 전송한 경로와 다른 경로로 재전송하도록 한다[14]. 그런데 현재의 SCTP가 데이타 전송을 위해서 하나의 경로만을 사용하는 것과 달리 LS 모드 서비스에서는 다중의 경로를 동시에 사용하고, 전송되는 경로에 따라 다른 SID, SSN을 가지며 서로 다른 경로 관리 모듈에 의해 오류 제어 및 혼잡 제어가 이루어진다. 따라서 원래 전송과 재전송이 다른 경로로 이루어지면 각 경로 관리 모듈은 올바른 혼잡 제어를 수행할 수 없다. 한편, 처음 전송 경로와 같은 경로로만 재전송이 이루어지도록 한다면, 특정 경로에서 손실된 패킷의 복구가 느릴 때 수신자의 수신 버퍼에 패킷이 점점 쌓이게 되어 수신자 윈도우가 고갈되므로 손실이 일어나지 않은 경로의 전송률까지도 저하시키게 된다. 따라서 LS 모드 서비스에서는 이러한 문제를 해결하기 위해 처음 전송 경로와 같은 경로로 재전송을 하는 동시에 빠른 손실 복구를 위해 다른 경로로도 재전송을 하도록 한다. 즉, 원래 경로로 재전송을 함으로써 각 경로 별로 정상적인 혼잡 및 오류 제어가 이루어지도록 하는 동시에 새로운 경로로도 재전송을 함으로써 빠르게 손실을 복구하여 수신자 윈도우 부족으로 인한 전체 성능의 저하를 막을 수 있도록 한다.

이를 위해 어소시에이션 관리 모듈은 Rtx data buffer와 Rtx info buffer를 가진다. Rtx info buffer에는 재전송된 패킷이 전송된 경로에 관한 정보가 들어있으며, 이 정보를 이용하여 특정 TSN을 가지는 패킷이 같은 경로로 중복 재전송되는 일이 없도록 한다. 그림 7은 Rtx Info buffer의 구조를 보여준 것이다. Send_bit는 전송된 경로를 나타내는 필드이고, Ack_bit는 그 경로로부터 ACK이 수신되었는지를 나타내는 필드이다. Rtx Info buffer의 정보는 패킷이 재전송될 때마다 추가되거나 수정되며, 특정 TSN에 해당하는 엔트리는 그

TSN	Path_1 Send_bit	Path_1 Ack_bit	Path_2 Send_bit	Path_2 Ack_bit	...	Path_n Send_bit	Path_n Ack_bit
10	T	T	T	F	...	F	F
19	F	F	T	T	...	F	F
			⋮				

그림 7 Rtx info buffer의 포맷과 예

TSN에 해당하는 패킷이 전송된 모든 경로로부터 ACK이 들어올 때 버퍼로부터 삭제된다.

어소시에이션 관리 모듈은 새로운 ACK이 수신될 때마다 다음과 같은 재전송 과정을 수행한다.

(1) Rtx data buffer로부터 재전송할 패킷을 복사해온다.
(2) 복사해 온 패킷의 TSN을 확인하여 Rtx info buffer에서 기존에 전송되었던 경로 정보를 찾는다.
(3) Rtx info buffer에 해당 TSN에 관한 정보가 없다면 새로운 엔트리를 추가하고, 패킷의 SID에 해당하는 경로의 Send_bit를 TRUE로 바꾼다.
(4) Send_bit가 FALSE인 경로들 중 혼잡 윈도우가 허용하는 경로를 찾아 해당 경로의 Send_bit를 TRUE로 바꾸고 패킷을 해당 경로의 경로 관리 모듈로 보내며, Rtx data buffer로부터 패킷을 삭제한다. Send_bit가 FALSE이면서 혼잡 윈도우가 열려 있는 경로가 없다면 다음 패킷을 처리한다.
(5) 패킷을 받은 경로 관리 모듈은 자신의 경로의 SID와 SSN으로 패킷의 SID/SSN 필드를 수정하여 패킷을 전송한다.

즉, 재전송은 이전에 전송되었던 경로들을 제외한 경로 중 혼잡 윈도우가 허용하는 경로로만 전송된다. 이러한 재전송 방안은 특정 패킷의 전송을 위해 사용되었던 경로가 연속적인 타임아웃 발생으로 프로토콜이 정의한 임계치를 초과하여 경로 손실로 판단되어도 어소시에이션의 연속성을 유지할 수 있도록 한다. 결과적으로 특정 경로가 손실되어도 그 경로에 배정되었던 ACK되지 않은 패킷들은 다른 경로로 재배정되며, 손실로 판단된 경로는 혼잡 윈도우가 1로 감소하게 되어 손실된 패킷에 대한 재전송만을 수행하고 새로운 패킷은 전송하지 않기 때문에 어소시에이션은 손실된 경로로 인한 피해를 입지 않게 된다.

4. 시뮬레이션 및 결과 분석

시뮬레이션은 ns-2 네트워크 시뮬레이터[15]에서 제공하는 TCP 모듈과 Delaware 대학교에서 ns-2 네트워크 시뮬레이터를 위해 구현한 SCTP 모듈[16]을 이용하였으며, 리눅스 레드햇 7.3에서 수행되었다. 그림 8은 시뮬레이션을 위해 사용된 네트워크 모델이며, 경로들의 대역폭, 지연 및 손실률 등의 경로 특성이 다른 경우에 대해 실험하였다. 각 라우터는 각각 하나의 에지 노드와 연결되어 있고, 각 에지 노드에는 TCP 또는 SCTP 에이전트가 올려지며, TCP 송신측은 path0이나 path1 중 하나의 경로로 TCP 수신측에 데이타를 전송하고, SCTP 송신측은 path0, path1 두 개의 경로를 모두 데이타 전송에 이용한다.

TCP 혹은 SCTP 송신자는 일반적인 인터넷 MTU 크기인 1500bytes 크기의 패킷을 전송하며, 하나의 스트림을 가정한다. 또한 전송 계층에서 가지는 사용자 응용의 데이타는 항상 충분하다고 가정한다.

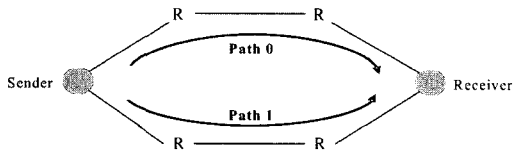

그림 8 시뮬레이션 네트워크 토폴로지

전송 계층에서는 손실을 배제할 경우 두 경로 간 특성차이는 RTT를 통해서만 파악이 가능하다. RTT는 전송 지연과 전파 지연, 그리고 그 밖의 지연들로 이루어지며, 이 중 전송 지연과 전파 지연을 제외한 나머지 지연들을 무시할 만한 수준으로 가정한다면, RTT에 영향을 주는 요소는 전송 지연과 전파 지연이다. 이 중 경로간의 RTT 차이의 원인이 대역폭의 차이로 인한 경우와 전파 지연의 차이로 인한 경우, 각각은 전체 성능에 다른 영향을 주므로 두 가지에 대해 제안한 LS 모드 서비스 방안이 어떠한 성능을 보이는지 살펴보고자 한다. 또한 혼잡에 의한 손실 발생 시 각 프로토콜별 성능 비교 및 제안하는 LS 모드 서비스의 재전송 방안에 대한 평가를 수행하였다.

4.1 경로들의 특성 차가 대역폭 차이로 인한 경우

그림 9는 두 경로의 RTT 차이가 대역폭 차이로 인한 경우의 실험 결과로, path0은 10Mbps로 대역폭을 고정시키고 path1의 대역폭을 10Mbps부터 2Mbps까지 2Mbps 단위로 변화시켜가면서 성능을 측정한 것이다.

"Ideal"은 두 경로의 이용 가능한 대역폭을 합한 값이

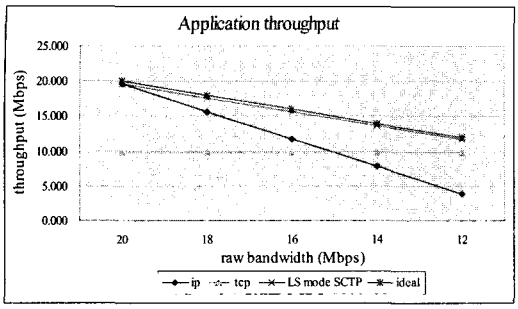

그림 9 경로 간 대역폭 차에 따른 응용 계층 처리율

다. LS mode SCTP는 패킷 전송을 위해 다중 경로를 사용하는 LS 모드 서비스의 SCTP이다. LS 모드 SCTP는 두 경로의 대역폭 차이와 무관하게 항상 거의 ideal에 가까운 처리율을 보인다. 이것은 두 경로를 독자적으로 사용하기 때문이다. LS 모드 SCTP는 그림 9에서 보는 바와 같이 두 경로의 대역폭 차이에 관계 없이 "Ideal"에 가까운 성능을 보인다. 즉, 두 경로의 가용 대역폭 합에 해당하는 작업량을 달성한다.

IP 계층에서의 부하 분산 방식은 가용한 모든 경로로 라운드 로빈 방식에 의해 데이타를 전송한다. 이 방식의 경우 두 경로의 대역폭 차이가 커질수록 성능은 크게 감소하며, 두 경로의 대역폭 합이 14Mbps로 대역폭의 비율이 2.5:1인 시점부터는 하나의 경로를 사용하는 TCP보다도 낮은 처리율을 보인다. 이는 두 경로의 대역폭 차이에 무관하게 라운드 로빈으로 두 경로에 같은 양의 데이타를 전송함으로 인해 대역폭이 낮은 경로로의 전송이 대역폭이 높은 경로의 전송률까지 떨어뜨리기 때문이다. 이는 표 1을 통해 확인할 수 있다. 표 1은 프로토콜별로 각 경로의 활용률을 보여주는 것으로, 각 경로의 활용률은 경로에서의 병목 링크의 활용률로 측정하였다. LS mode SCTP에서는 두 경로의 대역폭 차이에 관계없이 두 경로 모두 100%에 가까운 경로 활용률을 보여주는 반면, IP 방식은 대역폭이 낮은 경로인 path1의 전송률로 대역폭이 높은 path0의 전송률을 낮춤으로써 경로 간 대역폭 차이가 커짐에 따라 path0의 경로 활용률이 크게 떨어지는 것을 볼 수 있다.

TCP는 10Mbps의 대역폭을 가진 path0만을 사용하여 데이타를 전송하는 경우로 10Mbps에 근접한 처리율을 보이나 하나의 경로만을 사용하므로 제안한 LS mode SCTP보다는 낮은 처리율을 가진다.

4.2 경로들의 특성 차가 전파 지연으로 인한 경우

이 실험은 두 경로의 RTT 차이가 전파 지연 차이로 인한 경우로 path0은 6ms로 전파 지연을 고정시키고 path1의 전파 지연을 6ms~12ms로 1ms단위로 전파 지연을 변화시키면서 각 프로토콜에 대해 성능을 측정한 것이다. 각 그림의 x축은 path1의 전파 지연 시간을 나타낸다. TCP는 path0을 사용하는 경우와 path1을 사용하는 경우를 모두 실험하였다.

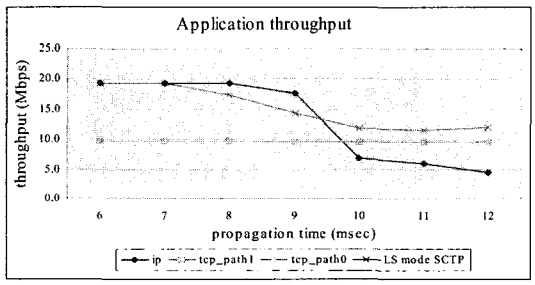

그림 10 경로 간 전파 지연차에 따른 처리율

그림 10은 두 경로의 전파 지연 차이에 따른 처리율을 보인 것이다. 두 경로의 특성이 비슷한 경우(6ms~9ms)에는 IP, LS mode SCTP, TCP 순으로 높은 처리율을 보이지만 경로간의 특성 차이가 커짐에 따라 LS mode SCTP나 IP방식은 처리율이 점점 감소한다. IP 방식의 경우 앞서 언급한 대로 두 경로의 전파 지연의 차이가 크지 않은 경우에는 가용한 대역폭인 20Mbps에 근접한 처리율을 보이지만 두 경로의 전파 지연차가 커짐에 따라 급격하게 성능이 저하하여 가용한 대역폭의 20%수준으로까지 떨어지게 된다. 이는 순서에 맞지 않게 수신자 측에 도착하는 패킷들로 인해 송신자는 3개 이상의 중복 ACK을 받게 되어 잘못된 빠른 재전송(fast retransmit)이 빈번히 일어나기 때문이다.

이를 확인하기 위하여 표 2에서 IP 방식에서 경로 전

표 1 대역폭 차이에 따른 프로토콜별 경로 활용률

Path utilization	TCP	IP		LS mode SCTP	
		path0	path1	path0	path1
10Mb:10Mb	99.96%	99.94%	99.94%	99.96%	99.96%
10Mb:8Mb	99.96%	79.94%	99.93%	99.86%	99.96%
10Mb:6Mb	99.96%	59.95%	99.92%	99.91%	99.96%
10Mb:4Mb	99.96%	39.96%	99.90%	99.72%	99.96%
10Mb:2Mb	99.96%	19.97%	99.84%	99.62%	99.96%

표 2 IP 방식의 잘못된 재전송 횟수

IP	잘못된 빠른 재전송 횟수
6ms:6ms	0
6ms:7ms	0
6ms:8ms	0
6ms:9ms	0.3
6ms:10ms	4.8
6ms:11ms	4.9
6ms:12ms	6.1

파 지연 차이에 의해 발생하는 잘못된 빠른 재전송 횟수를 측정한 결과를 보였다. 경로 간 전파 지연의 차이가 크지 않은 6ms:6ms~8ms 경우에는 잘못된 빠른 재전송이 일어나지 않으나, 경로 간 전파 지연 차이가 커짐에 따라 잘못된 재전송이 증가하는 것을 볼 수 있다. 이러한 잘못된 빠른 재전송은 네트워크가 혼잡이 아님에도 불구하고 혼잡이라 가정하고 TCP의 혼잡 제어 메커니즘에 따라 혼잡 윈도우를 반으로 줄이기 때문에 결과적으로 전송률을 낮추게 되어 성능 저하를 가져오게 한다.

LS mode SCTP는 두 경로의 전파 지연의 차이가 거의 없는 6ms, 7ms 의 경우에는 가용한 대역폭인 20Mbps 수준의 처리율을 보이나 전파 지연 차이가 증가함에 따라 점차적으로 처리율이 감소한다. 그 원인은 표 3에서 알 수 있듯이 전파 지연 차이가 증가함에 따라 전파 지연이 긴 경로를 최대한 활용하지 못하기 때문이다. 경로를 최대한 활용하기 위해서는 해당 경로를 채울 수 있을 만큼의 데이타를 전송하여 ACK 패킷이 연속해서 송신자 측에 도착할 수 있어야 한다. 그런데, 실험에서 사용한 수신자 윈도우 크기가 충분치 않아 긴 경로를 충분히 활용하지 못했다. 이 문제를 해결하기 위해서는 수신자 윈도우를 크게 잡아서 최대한 두 경로를 독자적으로 사용할 수 있도록 해야 한다. 그런데 이는 종단 간 평균 패킷 지연 및 수신자 전송 계층에서의 패킷 지연 시간 등을 가중시킬 수 있다는 문제점을 가진다. 그러나 LS mode SCTP 경우 IP와 같이 잘못된 혼잡제어로 인해 단일 경로를 사용하는 경우보다 성능이 더 감소하는 일은 발생하지 않았다.

4.3 패킷 손실 발생에 따른 성능 비교

LS 모드 서비스는 패킷이 전송되는 경로에 따라 다른 스트림 ID를 가지므로 손실이 발생하면 해당 패킷은 처음 전송되었던 경로로 재전송되어야 한다. 그러나 3.6절에서 언급한 대로 특정 경로에서의 손실은 전체 어소시에이션의 성능 저하를 가져올 수 있으므로 손실 복구는 빨리 이루어져야 한다. 이를 위해 LS 모드 서비스에서는 손실된 패킷에 대해서는 전송 가능한 여러 경로로 중복적인 재전송을 하도록 한다.

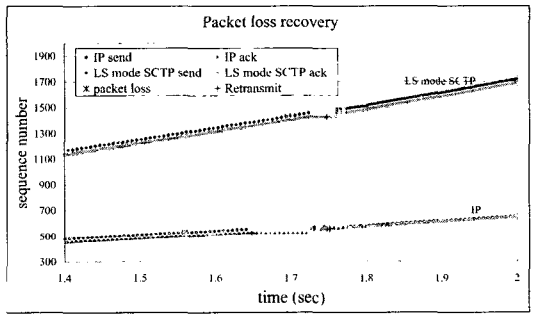

그림 11 프로토콜에 따른 패킷 손실 복구

그림 11은 두 경로의 가용 대역폭이 각각 2Mb, 10Mb로 차이가 있을 때, 대역폭이 적은 경로에서 손실이 생기는 경우 IP 방식과 LS mode SCTP가 각각 이를 복구하는 과정을 보여준 것이다. LS mode SCTP는 IP 방식의 40%의 시간으로 손실을 복구하는 것을 볼 수 있는데, 이는 LS mode SCTP가 두 경로에 중복적으로 재전송을 함으로써 대역폭이 낮은 경로에서의 손실을 대역폭이 높은 경로를 통해 빠르게 복구하기 때문이다. 이에 반해 IP 방식에서는 손실의 복구가 느릴 뿐만 아니라 손실로 인해 잘못된 재전송까지 발생하는 경우가 있다.

그림 12는 대역폭의 80%에 해당하는 외부 트래픽을

표 3 전파 지연 차이에 따른 프로토콜별 경로 활용률

Path utilization	TCP	IP		LS mode SCTP	
		path0	path1	path0	path1
6ms:6ms	99.14%	98.90%	98.90%	98.90%	98.90%
6ms:7ms	98.98%	98.59%	98.59%	98.90%	98.76%
6ms:8ms	98.78%	98.33%	98.33%	98.90%	78.36%
6ms:9ms	98.59%	90.65%	90.60%	98.90%	48.05%
6ms:10ms	98.38%	35.21%	34.63%	98.90%	21.58%
6ms:11ms	98.16%	30.96%	30.36%	98.90%	19.13%
6ms:12ms	97.94%	23.33%	22.61%	98.90%	22.73%

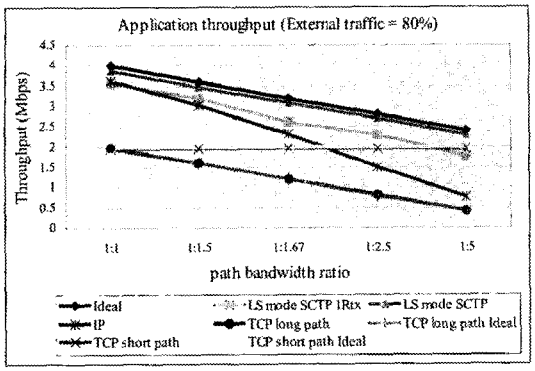

그림 12 프로토콜별 응용 계층 처리율

주입하여 혼잡을 발생시켰을 때 두 경로의 대역폭 비율이 1:1~1:5로 변화함에 따라 처리율이 어떻게 변화하는지를 보인 것이다. 이 실험에서는 LS 모드 서비스의 중복 재전송의 효과를 파악하기 위해, 원래 전송되었던 경로로만 재전송하는 LS 모드 서비스의 성능도 함께 보였다.

그림 12에서 두 개의 경로 중 RTT가 상대적으로 긴 경로로 전송한 TCP의 결과를 TCP long path라고 하였고, RTT가 상대적으로 짧은 경로로 전송한 TCP 결과를 TCP short path라고 하였다. TCP는 RTT가 긴 경로를 사용하든 짧은 경로를 사용하든 관계없이 각각의 토폴로지에서 Ideal한 처리율을 보인다. 그러나 단일 경로만을 사용하므로 전체 처리율에 있어서는 제안하는 LS mode SCTP에 비해 낮은 처리율을 보인다. IP 방식의 경우, 대역폭 차가 커짐에 따라 처리율이 급격하게 저하되어 대역폭 차가 1:1.67을 넘어서면부터는 하나의 경로를 사용하는 TCP보다도 낮은 처리율을 보여준다. 이는 그림 9와 유사한 결과로 이미 대역폭 차에 의해 각 경로를 최대한 활용하지 못하게 되는 데에다가 손실 복구를 위한 재전송 이외에 불필요한 재전송까지 발생하기 때문에 송신자가 송신률을 필요 이상 낮추어 전체 처리율이 급격히 저하된다. LS mode SCTP 1Rtx라 표시된 것은 원래 전송되었던 경로로만 재전송을 하는 LS mode SCTP에 대한 결과인데, 이 방안은 대역폭 차가 작은 경우에는 비교적 높은 처리율을 가지나, 대역폭 차가 큰 1:5의 경우 단일 경로를 사용하는 TCP보다도 약간 낮은 처리율을 보인다. 이것은 RTT가 긴 경로에서 발생한 손실에 대한 복구가 길어져서 RTT가 짧은 경로로의 전송까지도 저하시키기 때문이다. 반면 제안하는 중복 재전송 방안은 1:1~1:5의 모든 실험에 대해서 ideal에 근접한 처리율을 보인다.

그러나 이러한 중복 재전송은 동일한 TSN을 가진 여러 개의 패킷을 생성하게 되므로 네트워크의 자원을 중복으로 이용하여 이에 따른 오버헤드가 발생하게 된다.

표 4 중복 재전송에 의한 오버헤드 측정

Path Bandwidth Ratio	Overhead(%)	
	Redundant	Original
1:5	0.149710	0.065232
1:2.5	0.073272	0.086037
1:1.67	0.139370	0.092495
1:1.25	0.214004	0.115640
1:1	0.142074	0.054957

표 4는 중복 재전송 방안에 의한 오버헤드를 측정한 것으로, 오버헤드율은 다음과 같이 계산하였다.

$$\text{오버헤드율}(\%) = \frac{\text{송신측 SCTP가 내보낸 총 패킷 수} - \text{수신측 SCTP가 받은 총 패킷 수}}{\text{송신측 SCTP가 내보낸 총 패킷 수}} \times 100$$

표 4에 따르면 중복 재전송 방안이 하나의 경로로만 재전송을 하는 경우에 비해 대체적으로 오버헤드율이 높으나 절대적인 수치로 볼 때는 중복 재전송에 의한 오버헤드는 크지 않음을 확인할 수 있다.

5. 결 론

본 논문에서는 호스트가 다중의 액세스 포인트를 통해 인터넷에 접속될 수 있을 때, 하나의 사용자 플로우가 사용할 수 있는 가용대역폭을 높이기 위해 이들 인터페이스들을 동시에 이용하는 전송 계층 데이타 분산 방안을 제안하였다. 이를 위해 멀티호밍을 지원하는 전송 계층 프로토콜인 SCTP를 다중 인터페이스로 부하를 분산하도록 확장하였으며, 시뮬레이션을 통해 제안한 방안이 기존의 TCP나 네트워크 계층에서의 데이타 분산 방식에 비해 높은 대역폭을 얻음을 확인하였다. 각 인터페이스와 연결된 경로의 특성이 서로 다른 경우에 네트워크 계층에서의 데이타 분산 방안은 타임아웃이나 잘못된 재전송의 발생으로 성능 저하를 가져올 수 있는 반면, 제안한 전송 계층에서의 방안은 그러한 문제점 없이 높은 대역폭을 얻을 수 있다. 특히 제안한 방안은 두 경로간의 특성 차이가 대역폭 차이로 인한 경우 최적의 성능을 보이며, 전파 지연의 차이로 인한 경우에도 네트워크 계층에서의 데이타 분산 방안에 비해 효율적임을 알 수 있었다. 또한 제안하는 LS 모드 서비스의 중복 재전송 방안에 대해서도 시뮬레이션 결과를 통해 그 효율성을 확인할 수 있었다.

참 고 문 헌

[1] R. Stewart et al., "Stream Control Transmissio Protocol," IETF RFC 2960, Oct. 1997.

[2] I. F. Akyildiz, G. Morabito, and S. Plalz "TCP-Peach: A new congestion control sche

for satellite IP networks," IEEE/ACM Trans. Networking, Jun. 2001.
[3] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanov, "TCP selective acknowledgement options," RFC2018, Oct. 1996.
[4] L. S. Barkmo, S. O. Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in Proc. ACM SIGCOMM, Oct. 1994.
[5] M. Mathis and J. Mahdavi, "Forward acknowledgement: Refining TCP congestion control," in Proc. ACM SIGCOMM, Aug. 1996.
[6] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP westwood: Bandwidth estimation for enhanced transport over wireless links," in Proc. ACM MOBICOM 2001.
[7] S. Floyd, T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 2582, Apr. 1999.
[8] Aleksandar Kuzmanovic and Edward W. Knightly, "TCP-LP: A Distributed Algorithm for Low Priority Data Transfer," In Proc. IEEE INFOCOM, Mar. 2003.
[9] H. Sivakumar, S. Bailey, and R. Grossman. "PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks," In Proc. IEEE Supercomputing (SC), Nov. 2000.
[10] T. Hacker and B. Athey, "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network," In Proc. IEEE IPDPS, Apr. 2002.
[11] L. Magalhaes and R. Kravets, "Transport level mechanisms for bandwidth aggregation on mobile hosts," In Proc. IEEE ICNP, Nov. 2001.
[12] HY Hsieh and R. Sivakumar, "A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts," In Proc. ACM MOBICOM, Sep. 2002.
[13] D. Phatak, T. Goff, "A Novel Mechanism for Data Streaming Across Multiple IP Links for Improving Throughput and Reliability in Mobile Environments," In Proc. IEEE INFOCOM, Jun. 2002.
[14] A. Caro, P. Amer, J. Iyengar and R. Stewart, "Retransmission Policies with Transport Layer Multihoming," In Proc. IEEE ICON, Sep. 2003.
[15] The Network Simulator, ns-2, http://www.isi.edu/nsnam/ns.
[16] ns-2 SCTP module, http://pel.cis.udel.edu

송 정 화
2002년 이화여자대학교 컴퓨터학과 졸업(학사). 2004년 이화여자대학교 컴퓨터학과 졸업(석사). 2004년~현재 삼성전자 기술총괄 기술전략실 연구원. 관심분야는 SCTP multihoming, Load Sharing, mobile SCTP

이 미 정
정보과학회논문지 : 정보통신
제 31 권 제 2 호 참조