SCTP에서 대체 경로의 RTT 정확도 향상

준회원 김 예 나*, 박 우 람*, 김 종 혁*, 종신회원 박 태 근**

Accuracy Improvement of RTT Measurement on the Alternate Path in SCTP

Ye-Na Kim*, Wooram Park*, Jonghyuk Kim* *Associate Members*
Taekeun Park** *Lifelong Member*

요 약

SCTP(Stream Control Transmission Protocol)는 새로운 전송계층 프로토콜로 다양한 기능들을 제공한다. 그 중에서 멀티호밍(multihoming)은 두 단말 사이의 어소시에이션(SCTP에서의 연결)이 여러 개의 경로를 사용할 수 있게 해 주는데, 여러 경로 중, 주 경로(Primary Path)는 처음 전송되는 데이터를 송수신하기 위하여 사용되고 대체 경로(Alternate Path)는 재전송되는 데이터를 송수신하기 위하여 사용된다. 그러나 SCTP의 현재 재전송 정책은 대체 경로로 데이터를 재전송함으로써 데이터 도착 성공률을 높여주는 반면 사실상 많은 상황에서 성능 저하의 주요인이 됨이 확인되고 있다. 이는 칸 알고리즘에 의한 것으로 대체 경로로 재전송된 데이터를 대체 경로의 RTT(Round-Trip Time)를 업데이트 하는데 사용할 수 없도록 하고 있기 때문이다. 본 논문에서는 이러한 성능 저하를 피하기 위해 새로운 기법을 제안한다. 제안하는 기법은 DATA chunk와 SACK chunk에서 사용되지 않는 2비트를 사용하여 첫 번째 전송과 재전송을 명확하게 구별한 뒤 RTT를 업데이트함으로써 RTO(Retransmission Time-Out) 값을 보다 정확하게 유지 할 수 있도록 하였다.

Key Words : SCTP, RTT, Multihoming, Transport Layer.

ABSTRACT

The Stream Control Transmission Protocol(SCTP) is a reliable transport layer protocol that provides several features. Multihoming is the one of the features and allows an association(SCTP's term for a connection) between two endpoints to use multiple paths. One of the paths, called a primary path, is used for initial data transmission and in the case of retransmission an alternate path is used. SCTP's current retransmission policy attempts to improve the chance of success by sending all retransmissions to an alternate destination address. However, SCTP's current retransmission policy has been shown to actually degrade performance in many circumstances. It is because that, due to Karn's algorithm, successful retransmissions on the alternate path cannot be used to update RTT(Round-Trip Time) estimation for the alternate path. In this paper we propose a scheme to avoid such performance degradation. We utilize 2bits which is not used in the flag field of DATA and SACK chunks to disambiguate original transmissions from retransmissions and to keep RTT and RTO(Retransmission Time-Out) values more accurate.

※ 이 논문은 2008년 단국대학교 대학연구비의 지원으로 연구되었음.
* 단국대학교 컴퓨터학과 컴퓨터과학({yenakim, wrpark, hyuk0104}@dankook.ac.kr)
** 단국대학교 컴퓨터학부 멀티미디어공학전공 (tkpark@dku.edu)
논문번호 : KICS2009-01-027, 접수일자 : 2009년 1월 21일, 최종논문접수일자 : 2009년 3월 31일

Ⅰ. 서 론

최근 새로운 전송 계층 프로토콜인 SCTP(Stream Control Transmission Protocol)[1]는 차세대 인터넷 전송 프로토콜로 많은 주목을 받고 있다[2,3,4]. SCTP는 UDP의 메시지 지향(message-oriented) 특성과 TCP의 혼잡 제어 및 흐름 제어 메커니즘과 같은 특성을 모두 포함하는 등 UDP와 TCP의 장점을 살리도록 설계되었다. 또한 SCTP는 기존의 전송 프로토콜에서 제공하지 않았던 기능인 멀티스트리밍(multistreaming)과 멀티호밍(multihoming)도 제공한다. 이 중에서도 멀티호밍은 SCTP에서 많은 관심을 받고 있는 기술 중 하나이다[2,5,6,7].

멀티호밍은 하나의 SCTP 세션이 한 개 이상의 IP 주소를 사용할 수 있도록 해준다. SCTP에서는 여러 개의 주소들 가운데 하나의 주소만 선택하여 데이터를 주고받는데 사용하며 이 주소에 해당하는 전송 경로를 주 경로(primary path)라고 한다. 그 외의 경로들은 대체 경로(alternate path)라고 하며 데이터 재전송과 백업의 목적으로 사용된다. 그림 1은 호스트 A와 호스트 B가 멀티호밍으로 연결된 토폴로지를 보여준다. 경로 (A_1, B_1)가 주 경로이고 경로 (A_2, B_2)를 대체 경로라고 할 때 주 경로의 목적지인 B_1이 도달 불가능(unreachable) 상태가 된다면 멀티호밍은 호스트 A가 데이터를 대체 경로의 목적지인 B_2로 전달함으로써 SCTP 연결을 유지할 수 있게 해준다.

현재 SCTP의 재전송 정책은 호스트가 멀티호밍일 경우 반드시 데이터를 이전에 전송했던 경로가 아닌 다른 경로로 재전송 해야만 한다고 RFC4960[1]에 명시되어 있다. SCTP의 이 같은 재전송 정책은 대체 경로로 재전송함으로써 동일한 데이터가 또다시 손실되는 것을 피하게 해준다. 그러나 SCTP는 대체 경로로 재전송된 데이터를 대체 경로의 RTT (Round-Trip Time)를 업데이트 하는데 사용하지 않아 사실상 이러한 SCTP의 재전송 정책은 많은 상황에서 성능이 저하됨이 확인되었다[5,8].

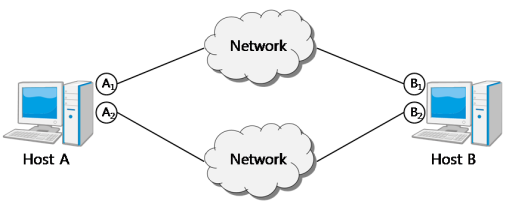


그림 1. 멀티호밍 토폴로지 예

이러한 성능 저하 문제점을 해결하기 위해 전송시간이 기록되는 TIMESTAMP chunk를 각 패킷에 추가하여 재전송의 모호성(ambiguity)을 제거하고 성능 향상을 추구한 연구가 있었다[9]. 이러한 timestamp 기법은 대체 경로로 재전송된 데이터도 RTT를 업데이트 하는데 사용할 수 있기 때문에 RTO(Retransmission Time-Out) 값을 보다 정확하게 유지할 수 있게 해준다. 그러나 TIMESTAMP chunk를 각 패킷에 추가함으로써 12바이트의 오버헤드가 추가적으로 발생한다는 단점이 있다.

본 논문에서는 이러한 단점을 해결하고 RTT의 정확한 측정을 위한 기법을 제안하였다. 제안한 기법은 TIMESTAMP chunk를 사용하지 않고 DATA chunk와 SACK chunk에서 사용되지 않는 flag 필드의 2비트를 사용하기 때문에 12바이트의 오버헤드가 추가적으로 발생하지 않는다. 시뮬레이션을 통한 성능 시험에서 timestamp 기법을 사용했을 때와 같은 상당한 수준의 RTT 정확도를 보장함은 물론 12바이트의 오버헤드도 발생하지 않음을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 SCTP 재전송 정책의 문제점을 해결하기 위하여 제안된 기존의 연구에 대해서 기술하고, 3장에서는 제안하는 기법에 대하여 서술한다. 4장에서는 성능 평가를 위한 실험 환경과 실험 결과를 보인다. 마지막으로 5장에서는 결론을 맺는다.

Ⅱ. 관련연구

현재 SCTP의 재전송 정책은 주 경로에서 손실된 데이터를 대체 경로로 재전송함으로써 데이터 도착 성공률을 높여준다는 장점이 있다. 그러나 모든 SACK을 RTT 업데이트 하는데 사용하지 않아 사실상 SCTP의 재전송 정책은 SCTP의 성능 저하를 일으킨다는 단점이 있다[8]. 이는 칸 알고리즘(Karn's algorithm)에 의한 것으로 대체 경로로 재전송이 성공적으로 이루어져도 재전송된 데이터는 대체 경로의 RTT를 업데이트하는 데는 사용되지 않기 때문이다. 대체 경로의 RTT를 업데이트 할 수 있는 트래픽은 목적지까지의 도달 가능성(reachability)을 확인하는 HEARTBEAT chunk 뿐이다[8]. RFC4960[1]에 의하면 이러한 HEARTBEAT chunk는 15초에서 45초 사이의 임의의 값으로 전송되기 때문에 대체 경로의 RTT를 정확하게 계산하는데 어려움이 있다.

SCTP의 성능 향상을 위해 [9] 연구에서는 첫 번째 전송과 재전송을 명확하게 구별할 수 있도록 데

Type	Flags	Length
Timestamp		
Timestamp echo		

그림 2. TIMESTAMP chunk

이터 전송 시간을 기록 할 수 있는 TIMESTAMP chunk를 각 패킷에 추가하는 방법을 제안하였다. 재전송의 모호성(ambiguity)을 없애기 위해 칸 알고리즘 사용을 중지하고 대체 경로로 재전송된 데이터도 RTT를 업데이트 하는데 사용한다. 이처럼 timestamp 기법은 대체 경로의 RTT를 업데이트 하는데 보다 많은 정보를 제공해준다.

SCTP의 각 패킷에 추가되는 TIMESTAMP chunk는 12바이트로 그림 2와 같다. 송신자가 DATA chunk 전송 시 *timestamp* 필드에 전송시간을 저장하고 수신자는 송신자로부터 받은 전송시간 값을 *timestamp echo* 필드에 복사해서 송신자에게 전송한다[9].

TIMESTAMP chunk를 사용하면 대체 경로의 RTT 측정을 위한 정보를 보다 많이 제공해준다는 장점이 있는 반면 TIMESTAMP chunk로 인해 12바이트의 오버헤드가 추가적으로 발생한다는 단점이 있다.

Ⅲ. 제안하는 기법

본 논문에서는 칸 알고리즘을 사용하지 않으면서 timestamp 기법의 단점이었던 오버헤드도 해결하는 기법을 제안한다. 먼저 SCTP의 패킷 형식은 그림 3과 같다. SCTP 패킷은 여러 개의 chunk를 담아 전송할 수 있고 각 chunk의 공통된 형식은 그림 3의 우측에 나와 있다. 본 논문에서 제안하는 기법은 DATA chunk와 SACK chunk의 flag 필드에서 사용되지 않는 2비트를 사용한다.

제안하는 기법을 설명하기에 앞서 칸 알고리즘을 적용한 original SCTP와 본 논문에서 제안하는 기법을 적용한 SCTP를 비교하면 그림 4와 같다. Host A가 Host B에게 전송한 데이터가 손실되어 Host A는 RTO 시간 후에 대체 경로로 손실된 데이터를 재전송한다. 대체 경로를 통해 재전송된 데이터를 수신한 Host B는 이에 대한 SACK을 Host A에게 전송한다. 이때 SACK을 수신한 Host A의 행동은 그림 4(a), (b)와 같이 다르다.

그림 4(a)의 Host A는 도착한 SACK이 첫 번째 전송한 데이터에 대한 확인 응답인지 재전송한 데이터에 대한 확인 응답인지 구별 할 수가 없어서 도착한 SACK을 RTT 업데이트 하는데 사용하지 않는다. 반면 그림 4(b)의 Host A는 그림 4(a)와 같은 상황임에도 불구하고 2비트 정보를 활용하여 도착한 SACK이 대체 경로로 재전송된 데이터에 대한 확인 응답임을 알고 이를 대체 경로의 RTT를 업데이트 하는데 사용한다.

그림 4(b)에서 사용되는 2비트의 의미와 RTT 측정 대상은 표 1과 같다. 송신자가 데이터를 첫 번째로 전송하는 경우에는 '01' 비트를 입력하고 '01' 비트가 붙은 SACK을 수신하였을 경우에는 주 경로의 RTT를 업데이트한다. 반면 데이터를 두 번째로 전송하는 재전송일 경우에는 '10' 비트를 입력하고 '10' 비트가 붙은 SACK을 수신하였을 때는 대체 경로의 RTT를 업데이트 한다. 데이터를 세 번째 이상 전송하는 경우에는 '11' 비트를 입력해주고 '11' 비트가 붙은 SACK을 수신하였을 때는 RTT를 업데이트를 하지 않는다. 이는 제안한 기법이 2비트를 사용하여 전송 횟수를 표시하기 때문에 세 번

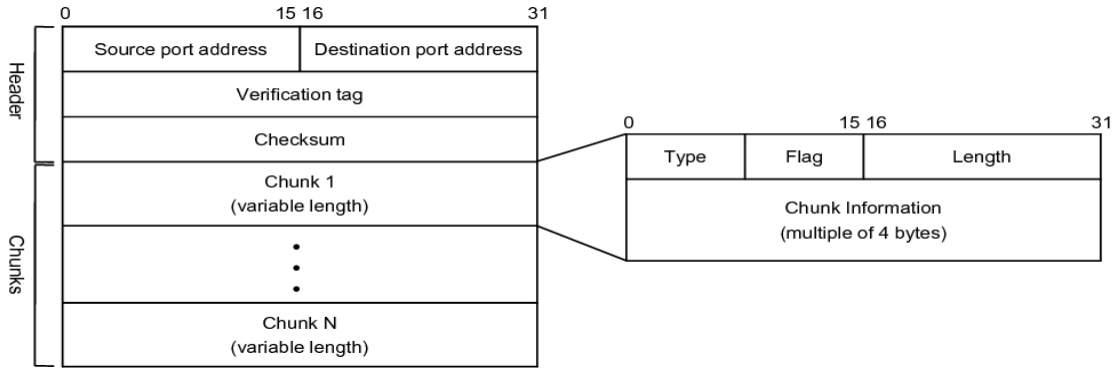
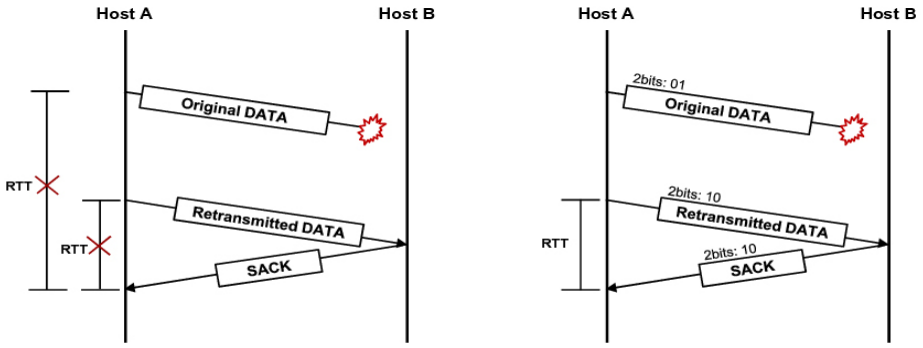


그림 3. SCTP 패킷과 공통된 chunk의 형식



(a) Original SCTP (b) 제안한 기법을 적용한 SCTP

그림 4. Original SCTP와 제안한 기법을 적용한 SCTP의 비교

이상 데이터를 전송한 경우, 수신한 SACK이 몇 번째 전송된 데이터에 대한 응답으로 수신한 것인지 송신자가 판단할 수 없게 되어져서 original SCTP에서와 같이 RTT를 업데이트하지 않는 것이다. 마지막으로 표 1에 없는 '00' 비트는 original SCTP와의 호환성을 위해서 사용하지 않는다. 다시 말하면 제안하는 기법을 적용한 SCTP를 사용하는 송신자가 상대방으로부터 '00' 비트를 수신할 경우 '00' 비트 송신자가 original SCTP 사용중인 것을 인지하고 original SCTP와 동일하게 작동할 수 있게 하기 위함이다.

DATA chunk의 2비트를 사용하여 몇 번째로 전송하는 데이터인지를 표시하고 수신한 SACK chunk의 2비트가 나타내는 경로의 RTT를 업데이트하는 송신자 측의 규칙은 그림 5와 같다. 그림 5의 Rule 1은 DATA chunk 생성 시 송신자의 규칙을 보여준다. 송신자는 S라고 하는 집합을 관리하는데, 집합 S는 전송한 DATA chunk들 중 수신 확인 응답을 받지 못한 DATA chunk들에 대한 (TSN, F, T)들의 집합을 의미한다. 송신자는 DATA chunk 생성 시 몇 번째 전송하는 데이터인지에 따라 2비트 값을 flag F에 입력하고 이를 데이터의 전송 순서 번호인 TSN, 전송 시간 T와 함께 집합 S에 저장을 해둔다. 송신자가 데이터를 보내고 그 데이터에 대한 SACK을 수신하였을 때의 규칙은 그림 5의 Rule 2와 같다. 송신자는 i번 째 수신한 $SACK_i$의 Cumulative TSN ACK 필드값과 Gap ACK Block 필드값을 이용하여 도착 확인 받은 TSN들을 집합 S_{SACKi}에 저장한다. 그리고 집합 S_{SACKi}에는 포함되어 있으나 이 전에 수신한 집합 $S_{SACKi-1}$에는 포함되지 않은 TSN들은 S_d라고 하는 집합에 저장한다. 집합 S_d를 생성한 후 집합 $S_{SACKi-1}$은 지우고 다음 번 계산을 위해 집합 S_{SACKi}를 저장한다. 그림 5의 Rule 3은 RTT 업데이트 대상을 찾는 규칙을 보여준다. 송신자는 집합 S_d에 속하는 TSN들에 해당하는 (TSN, F, T)들을 집합 S에서 검색하여 이를 집합 S_T에 저장한다. 그리고 집합 S_T의 (TSN, F, T)들 중에서 $SACK_i$의 F와 동일한 값을 가지는 가장 큰 TSN을 결과값으로 반환하는 함수인 $f_{MAX}(S_T)$을 통해 TSN을 알아낸 후 그 TSN에 해당하는 데이터가 전송된 시간과 $SACK_i$를 수신한 시간으로 RTT를 계산한다. 이때 RTT 업데이트 대상 경로는 수신한 $SACK_i$의 F값에 의해 결정된다. 송신자 측의 마지막 규칙은 그림 5의 Rule 4와 같다. 송신자는 해당 경로의 RTT를 업데이트 한 후에 집합 S_T에 포함된 (TSN, F, T)들을 집합 S에서 삭제하고 다음 사용을 위해 집합 S_T는 비운다.

반면 SACK chunk의 2비트를 사용하여 어느 경로의 RTT를 업데이트해야 하는지 송신자에게 알려주기 위한 수신자 측의 규칙은 그림 6과 같다. 그림 6의 Rule 1은 수신자가 DATA chunk를 수신했을 때의 규칙을 보여준다. 수신자는 송신자와 마찬가지로 R이라고 하는 집합을 관리하며 이는 수신한 DATA chunk들에 대한 (TSN, F)들의 집합이다. 수신자는 DATA chunk를 수신하면 DATA chunk의 TSN과 F를 집합 R에 저장한다. 그림 6의 Rule 2는

표 1. 2비트의 의미와 RTT 측정 대상

	의미	RTT 측정 상
01	First Transmission	주 경로
10	Second Transmission	대체 경로
11	nth Transmission (n≥3)	-

Rule 1: DATA chunk 생성
- i) 데이터의 전송 횟수에 따라 flag F 설정
  - if ($NumTx$ = 1) then F = 01
  - if ($NumTx$ = 2) then F = 10
  - if ($NumTx$ >= 3) then F = 11
- ii) DATA chunk의 (TSN, F, T)를 집합 S에 저장
  - S = {x|x=(TSN, F, T)}

Rule 2: $SACK_i$ 수신
- i) 집합 S_{SACKi}에 i번 째 도착한 SACK($SACK_i$)로부터 도착 확인 받은 TSN들을 저장
  - $S_{SACKi} = S_{CumACKi} \cup S_{GapACKi}$
- ii) 집합 S_d에 $SACK_{i-1}$로부터 도착 확인 받지 못했으나 $SACK_i$로부터 도착 확인 받은 TSN들을 저장
  - S_d = {TSN|$TSN \in S_{SACKi}$ and $TSN \notin S_{SACKi-1}$}
- iii) 집합 S_d 계산 후 집합 $SACK_{i-1}$을 지우고 집합 $SACK_i$을 저장

Rule 3: RTT Update
- i) $SACK_i$가 가장 최근에 받은 SACK일 때 집합 S_d에 속하는 TSN들에 대하여 집합 S로부터 (TSN, F, T)들을 찾아내어 이를 집합 S_T에 삽입
  - S_T = {x|x=(TSN, F, T) and $x \in S$ and $TSN \in S_d$}
- ii) 함수 $f_{MAX}(S_T)$ 결과값인 TSN의 전송시간 T와 $SACK_i$를 받은 시간으로 RTT 계산
  - $f_{MAX}(S_T)$: 집합 S_T의 (TSN, F, T)들 중에서 $SACK_i$의 F와 동일한 값을 가지는 가장 큰 TSN을 반환하는 함수
- iii) RTT 업데이트 대상 선정
  - if (F = 01) then 주 경로의 RTT 업데이트
  - if (F = 10) then 대체 경로의 RTT 업데이트
  - else ignore

Rule 4: 사용된 (TSN, F, T) 삭제
- i) 집합 S_T에 포함된 (TSN, F, T)들을 집합 S에서 삭제하고 집합 S_T를 공집합으로 만듦

그림 5. 송신자 측 규칙

SACK chunk를 생성할 때의 규칙을 보여준다. SACK chunk를 생성할 때까지 저장된 집합 R의 모든 F값들이 동일하면 그 동일한 F값을 SACK chunk의 flag 필드에 입력해서 전송한다. 집합 R의 F값들이 동일하지 않으면 저장된 F값 중 우선순위가 높은 F값을 입력해서 전송한다. 우선순위를 사용하는 이유는 데이터가 주로 주 경로를 통해 전송되어 대체 경로로 전송되는 데이터의 수가 상대적으로 작기 때문에 RTT 업데이트 하는데 있어서 대체 경로에 우선순위를 주기 위함이다. 또한 '11' 비트의 우선순위가 가장 높은 것은 몇 번째 전송인지

Rule 1: DATA chunk 수신
- i) 수신한 (TSN, F)을 집합 R에 저장
  - R = {x|x=(TSN, F)}

Rule 2: SACK chunk 생성
- i) Flag setting (우선순위 : 11>10>01)
  - if (집합 R의 모든 F 값이 동일) then 동일한 F를 SACK에 사용
  - if (집합 R의 모든 F 값이 동일하지 않으면) then 높은 값의 F를 SACK에 사용

Rule 3: 사용된 (TSN, F) 삭제
- i) SACK chunk 전송 후, SACK 생성에 사용된 (TSN, F)를 집합 R에서 삭제

그림 6. 수신자 측 규칙

모르는 데이터가 하나라도 섞여 있는 경우 RTT를 계산하지 않도록 하기 위함이다.

Ⅳ. 성능 평가

본 논문에서는 제안한 기법을 original SCTP, timestamp 기법과 RTT 업데이트 횟수, 오버헤드, 대체 경로의 RTO 측정 정확도를 비교하여 그 효율성을 어느 정도 증가시켰는지를 보이고자 한다. 네트워크 시뮬레이션은 ns-2[10]를 사용하였으며 성능 실험에 사용한 토폴로지는 그림 7과 같다. 종단간 단방향 전파 지연은 250ms이고 대역폭은 10Mbps이며 성능 변화를 위해 주 경로의 패킷 손실률은 0~16%, 대체 경로의 패킷 손실률은 0~10%로 변화시키며 성능을 측정하였다.

4.1 RTT 업데이트 횟수 및 오버헤드 비교

Original SCTP, timestamp 기법과 제안한 기법의 RTT 업데이트 횟수 및 오버헤드 비교를 위해 20번의 실험을 수행하였으며 실험 결과의 평균은 표 2와 같다. 먼저 original SCTP의 경우 RTT 업데이트 횟수가 수신한 SACK에 비해 현저히 작다. 이는 RFC4960[1]에서 매 왕복(round trip)마다 한

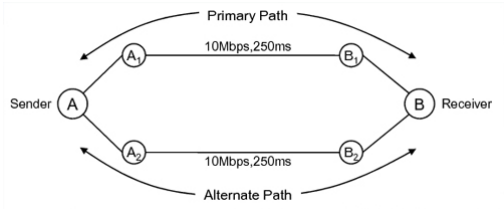


그림 7. 시뮬레이션 네트워크 토폴로지

표 2.RTT 업데이트 횟수 및 오버헤드 비교표

	수신 SACK 개수	RTT 업데이트 횟수			오버헤드(%) (TIMESTAMP chunk bytes /Transmitted bytes × 100)
		주 경로	대체 경로	합계	
Original SCTP	2595.2	337.2	5.2	342.4	0
SCTP with timestamp	2631.6	3603.1	93.65	3696.75	1.22%(88407/7249707)
SCTP with 2bits	2632.3	2537.2	92.7	2629.9	0

번의 RTT 측정을 권고하고 있기 때문이다. 즉 original SCTP는 SCTP 패킷 전송 시 여러 개의 DATA chunk 중 한 개의 DATA chunk에 대해서만 타이머를 구동하고 확인 응답을 기다린다. 또한 original SCTP는 칸 알고리즘을 사용하기 때문에 대체 경로로 재전송된 데이터를 대체 경로의 RTT를 업데이트 하는데 사용하지 않아 수신한 SACK에 비해 RTT 업데이트 횟수가 작은 것이다. 표 2에 나와 있는 original SCTP의 대체 경로 RTT 업데이트 횟수는 모두 HEARTBEAT chunk에 의한 것이다.

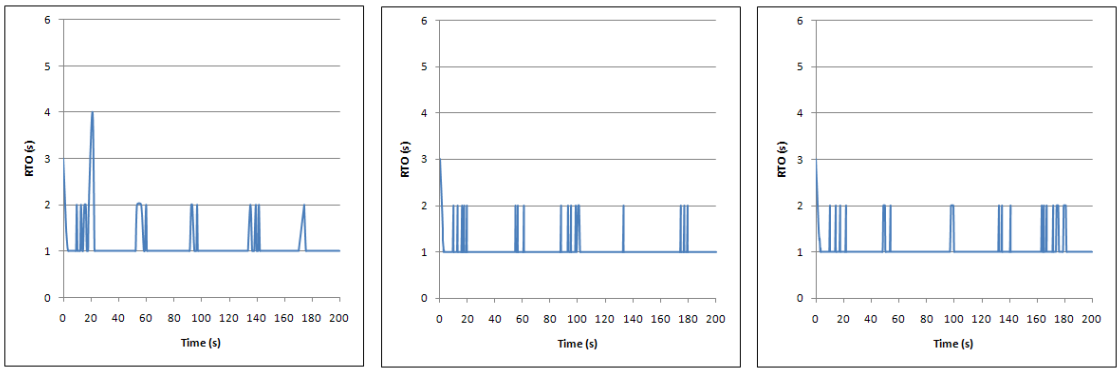
반면 timestamp 기법을 적용한 SCTP는 칸 알고리즘 사용을 중지하고 전송되는 SCTP 패킷의 모든 DATA chunk에 대하여 RTT를 계산한다. 따라서 하나의 SACK을 수신할 때에도 한 번 이상 RTT 업데이트를 하기 때문에 수신한 SACK의 개수 보다 RTT 업데이트 횟수가 많은 것이다. Timestamp 기법은 original SCTP 보다 RTT 업데이트를 자주 함으로써 RTO 값을 정확하게 유지할 수 있는 반면 TIMESTAMP chunk로 인해 전체 전송된 데이터량 대비 평균 1.22% 정도의 오버헤드가 발생한다는 단점이 있다.

마지막으로 본 논문에서 제안한 기법은 수신한 SACK의 개수와 RTT 업데이트 횟수가 거의 동일하다. 이는 timestamp 기법과 마찬가지로 칸 알고리즘 사용을 중지하였기 때문에 대체 경로로 재전송된 데이터에 대해서도 RTT 업데이트를 할 수 있기 때문이다. 그러나 timestamp 기법과 달리 하나의 SACK 수신 시 flag 필드값에 의해 한 번만 RTT 업데이트를 하기 때문에 수신한 SACK의 개수와 RTT 업데이트 횟수가 거의 일치하게 되었다. 이처럼 본 논문에서 제안한 기법은 original SCTP에 비해 대체 경로의 RTT 업데이트를 보다 많이 하면서도 TIMESTAMP chunk를 사용하지 않아 추가적인 오버헤드 없이도 RTT 업데이트하는 것을 알 수 있다.

4.2 대체 경로의 RTO 측정 정확도 비교

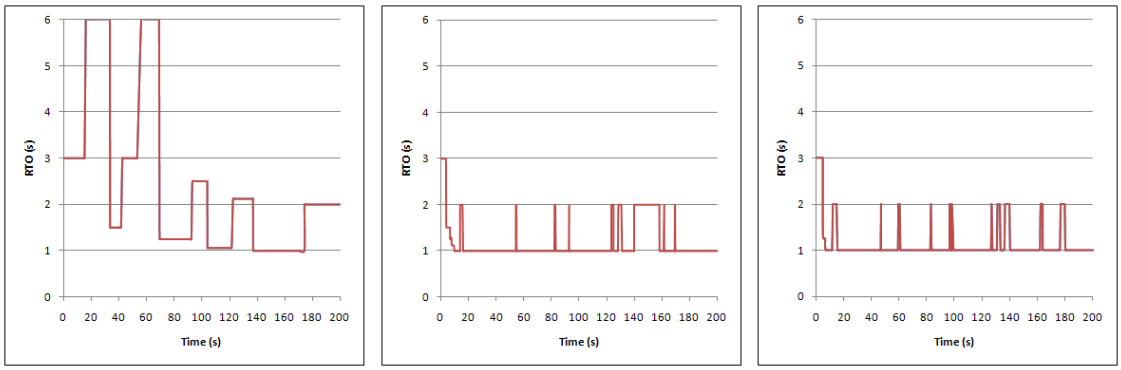
그림 8은 original SCTP, timestamp 기법과 제안한 기법을 사용했을 때 패킷 손실률에 따른 주 경로의 RTO 변화를 보여주고 있다. 그림에서 확인할 수 있듯이 세 기법 모두 패킷 손실로 인해 타임아웃이 발생해도 빈번한 RTT 측정으로 RTO 값의 빠른 복구가 가능함을 보여주고 있다.

그림 9는 original SCTP, timestamp 기법과 제안한 기법을 사용했을 때 패킷 손실률에 따른 대체 경로의 RTO 변화를 보여주고 있다. 그림 9(a)의 original SCTP는 칸 알고리즘 사용으로 인해 RTT 업데이트가 이루어지지 않아 결과적으로 RTO 값의



(a) Original SCTP (b) SCTP with timestamp (c) SCTP with 2bits

그림 8. Primary path의 RTO 비교



(a) Original SCTP (b) SCTP with timestamp (c) SCTP with 2bits

그림 9. Alternate path의 RTO 비교

빠른 갱신이 이루어지지 않음을 보여주고 있다. 그림 9(a)에서 대체 경로는 총 5번 RTT 및 RTO 업데이트가 이루어지는데 이는 모두 HEARTBEAT chunk에 의한 업데이트이다. 따라서 original SCTP 대체 경로의 RTO 값은 HEARTBEAT chunk에 대한 응답이 수신되기 전까지 부정확한 RTO 값을 유지하게 되고 결과적으로 네트워크의 성능 저하를 불러일으키게 된다. 반면 timestamp 기법을 적용한 그림 9(b)의 경우 original SCTP와는 달리 HEARTBEAT chunk에 대한 응답이 오기 전에 RTO 업데이트 하는 것을 볼 수 있다. 이는 timestamp 기법이 칸 알고리즘 사용을 중지했기 때문에 대체 경로로 재전송된 데이터에 대한 확인 응답도 RTO 업데이트 하는데 사용하기 때문이다. 따라서 그림 9(a)와 그림 9(b)의 대체 경로 RTO 변화 그래프를 비교했을 때 timestamp 기법을 사용한 SCTP가 original SCTP 보다 정확하게 RTO 업데이트를 하고 있음을 알 수 있다. 마지막으로 본 논문에서 제안한 기법을 적용한 SCTP에서의 대체 경로의 RTO 변화는 그림 9(c)와 같다. 그림 9(c)의 경우 TIMESTAMP chunk를 적용하지 않았음에도 불구하고 timestamp 기법과 같이 빈번한 RTO 업데이트로 정확한 값을 유지하고 있다. 이는 제안한 기법이 DATA chunk와 SACK chunk의 flag 필드에서 사용되지 않는 2비트를 활용하여 첫 번째 전송과 재전송을 명확하게 구별하고 재전송된 데이터에 대한 확인 응답도 RTO 업데이트 하는데 사용하기 때문이다. 이처럼 본 논문에서 제안한 기법은 TIMESTAMP chunk로 인한 추가적인 오버헤드 없이도 그와 상당한 성능을 보일 수 있다. 예를 들어, 주 경로의 패킷 손실률이 1%로 고정되어있고, 대체 경로의 패킷 손실률이 0%부터 10%까지 변화하는 환경에서 4MB의 파일을 다운로드 하는 경우, 파일 전송 완료 시간 측면에서 timestamp 기법은 original SCTP에 비하여 최소 2.5%, 최대 24.5%의 성능 향상을 가져온다[11]. 제안한 기법 또한 RTO 업데이트를 정확히 함으로써 이상과 동일한 수준의 성능 향상을 획득하였다.

V. 결 론

현재 Original SCTP의 재전송 정책은 칸 알고리즘 사용으로 인해 대체 경로의 RTT 및 RTO를 정확하게 측정하지 못하여 네트워크의 성능을 저하시키는 문제가 있다. 이러한 문제점을 해결하기 위하여 기존에 제안된 기법으로 timestamp 기법이 있다. Timestamp 기법은 칸 알고리즘 사용을 중지함으로서 대체 경로의 RTT 업데이트를 위한 정보를 보다 많이 제공해 주었으나 TIMESTAMP chunk로 인해 추가적인 오버헤드가 발생한다는 단점이 있다. 이러한 문제점을 해결하고자 본 논문에서는 DATA chunk와 SACK chunk의 flag 필드에서 사용되지 않는 2비트를 활용하는 기법을 제안하였다.

제안된 기법의 성능 평가를 위해 성능 실험을 수행하였고, 그 결과 original SCTP 보다 정확하게 대체 경로의 RTO 값을 측정하였다. 또한, 이와 더불어 timestamp 기법의 단점이었던 추가적인 오버헤드도 해결하여 네트워크 성능 저하를 피할 수 있음을 증명하였다.

향후 과제로는 제안한 기법을 주 경로와 대체 경로가 다른 특성을 가지는 환경과 MANET(Mobile Ad-hoc Network) 환경에 적용하여 그 성능을 측정할 예정이다.

참 고 문 헌

〔1〕 R. Stewart et al., "Stream Control Transmission Protocol," IETF RFC 4960, Sept. 2007; www.ietf.org/rfc/rfc4960.txt.

〔2〕 Randall Stewart and Chris Metz, "SCTP: New Transport Protocol for TCP/IP," *IEEE Internet Computing*, Vol. 5, No. 6, pp. 64-69, Nov. 2001.

〔3〕 Li Ma and F. Richard Yu, "Performance Improvements of Mobile SCTP in Integrated Heterogeneous Wireless Networks," *IEEE Transactions on Wireless Communications*, Vol. 6, No. 10, Oct. 2007.

〔4〕 Sang Tae Kim, Seok Joo Koh, Yong Jin Kim, and Modacom Incorporation, "Performance of SCTP for IPTV Applications," *ICACT2007*, Feb. 2007.

〔5〕 Dongkyun Kim, Jeomki Song, Joungsik Kim, Hongseok Yoo, Jungsoo Park, and Juan-Carlos Cano, "The Applicability of SCTP to Mobile Ad Hoc Networks," *ICACT2006*, Feb. 2006.

〔6〕 Ki-Il Kim, "Establishing Multiple Pahts for Multihomning on SCTP in MANET," *IEICE TRANS. COMMUN.*, Vol. E91-B, No. 4, April 2008.

〔7〕 Janardhan R. Iyengar, Paul D. Amer, and Randall Stewart, "Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths," *IEEE/ACM Transactions on Networking*, Vol. 14, No. 5, Oct. 2006.

〔8〕 Armando L. Caro Jr., Paul D. Amer, and Randall R. Stewart, "Transport Layer Multihoming for Fault Tolerance in FCS Networks," *MILCOM 2003*, Boston, MA, Oct. 2003.

〔9〕 Armando L. Caro Jr., Paul D. Amer, Janardhan R. Iyengar, and Randall R. Stewart, "Retransmission Policies With Transport Layer Multihoming," *ICON 2003*, Sydney, Australia, Sept. 2003.

〔10〕 UC Berkeley, LBL, USC/ISI, and Xerox Parc. ns-2 documentation and software, Version 2.31, 2001. http://www.isi.edu/nsnam/ns.

〔11〕 Armando L. Caro Jr., Paul D. Amer, Janardhan R. Iyengar, and Randall R. Stewart, "Retransmission Policies With Transport Layer Multihoming," Tech Report TR2003-05, CIS Dept, University of Delaware, March 2003.

김 예 나 (Ye-Na Kim) 준회원

2008년 2월 단국대학교 멀티미디어학과 졸업

2008년 3월~현재 단국대학교 컴퓨터학과 석사과정

<관심분야> SCTP, 이동 통신

박 우 람 (Wooram Park) 준회원

2008년 2월 단국대학교 멀티 멀티미디어학과 졸업

2008년 3월~현재 단국대학교 컴퓨터학과 석사과정

<관심분야> Wireless Network, IPTV

김 종 혁 (Jonghyuk Kim) 준회원

2008년 2월 단국대학교 멀티미디어학과 졸업

2008년 3월~현재 단국대학교 컴퓨터학과 석사과정

<관심분야> Service Discovery, MANET

박 태 근 (Taekeun Park) 종신회원

1991년 포항공과대학교 컴퓨터공학과 학사

1994년 포항공과대학교 컴퓨터공학과 석사

2004년 포항공과대학교 컴퓨터공학과 박사

1996년~2000년 SK Telecom 중앙연구원 선임 연구원

2000년~2001년 3Com Korea 과장

2001년~2002년 Ericsson Korea 차장

2004년~현재 단국대학교 컴퓨터학부 조교수

<관심분야> 이동 통신, QoS, 센서 네트워크, 멀티미디어 통신망