링크를 효율적으로 사용하기 위한 QoS 기법

ISP로부터 제공 받아 사용하는 대부분의 WAN 구간은 LAN 구간에 비해 상대적으로 고가이기 때문에 적은 대역폭으로 구성되는 것이 일반적이다. 이때 WAN 구간의 대역폭보다 큰 용량의 트래픽을 전송해야 하는 상황이 된다면 외부로 전송되지 못한 패킷들이 큐에 적체되게 돼 지연, 지터, 드롭이 발생하게 될 것이다. 만일 이 트래픽이 비디오나 음성과 같이 일정시간 내에 전달돼야만 하는 패킷이라면 아마도 정상적인 서비스를 제공하지 못하게 될 것이다. 이번호에서는 링크를 효율적(Link Efficiency)으로 사용하기 위한 QoS 기법에 대해 알아보자.

김화식
zion@onsetel.co.kr 온세통신 시설운영팀 CCIE
s12840@freechal.com NRC Network+ 팀 마스터

난 이 도 上 中 下

지난호에는 QoS의 세부적인 튜닝 부분으로 들어가서 혼잡 회피에 대해 알아봤다. TCP의 작동 메커니즘을 확인하고 혼잡을 예방하기 위한 방법인 RED와 WRED를 소개했다. 이번 시간에는 QoS의 세부적인 튜닝 방법 중 하나로, 링크를 효율적으로 사용하기 위한 기술인 압축(compression) 기법과 LFI(Link Fragmentation and Interleaving) 기법에 대해 알아볼 것이다. 압축 기법은 WAN 구간으로 패킷을 전송하기 전에 압축을 해서 전송하는 기술로, 패킷의 페이로드(payload) 부분을 압축하는 방법과 헤더(hader) 부분을 압축하는 방법으로 구분된다.

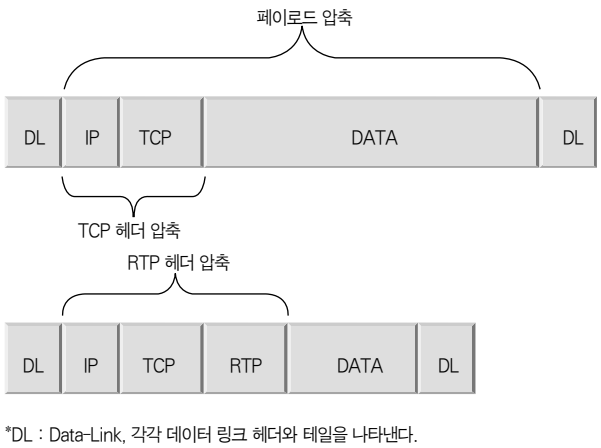
LFI 기법은 연속 지연(serialization delay)에 직접적으로 영향을 주는 기술로, 크기가 작은 패킷이 큰 패킷 뒤에 있는 경우, 큰 패킷이 큰 패킷의 길이만큼 더 지연이 되는 현상을 줄이기 위한 것이다. 이는 큰 패킷을 여러 작은 크기로 나눈(fragmentation) 후 각 개체들이 서비스 되어지는 사이사이에 작은 패킷을 끼어넣어(interleaving) 주는 방법으로, 보통 작은 패킷들이 지연에 민감한 애플리케이션에서 사용되는 경우가 많다.

압축 기법의 종류와 특징

압축 기법은 원본 패킷을 수학 알고리즘을 이용해 작게 압축해 전송한다. 반대편에서는 수신된 압축패킷을 다시 수학 알고리즘을 이용해 압축을 푸는 과정을 진행한다. 많은 수학자와 컴퓨터 과학자들이 다양한 압축 알고리즘을 개발하는 과정에서 압축하는 비율에 따라 효율이 좋은 경우와 나쁜 경우가 발생함을 발견했다. 또 같은 비율의 압축 방법이라도 다른 압축 알고리즘을 이용하면 각각 다르게 CPU와 메모리를 사용한다는 것도 확인했다. 따라서 트래픽 패턴과 장비의 성능을 분석하고 적절한 압축 기법을 적용해야 한다. 앞서 말했듯이 어떤 부분을 압축하느냐에 따라 페이로드 압축과 헤더 압축으로 구분되는데, 페이로드 압축은 패킷의 헤더 부분과 데이터 부분을 압축하는 방법으로, 패킷의 크기가 큰 경우에는 헤더 압축보다 좋은 압축 효율을 제공하는 장점이 있다. 반면, 헤더 압축은 헤더 부분만을 압축하는 방법으로, 패킷의 크기가 작을 때 좋은 압축 효율을 제공한다.

(그림 1)은 페이로드 압축시 압축 필드와 헤더 압축시 압축 필드를

그림 1 | 페이로드 압축과 헤더 압축시 압축 필드 비교

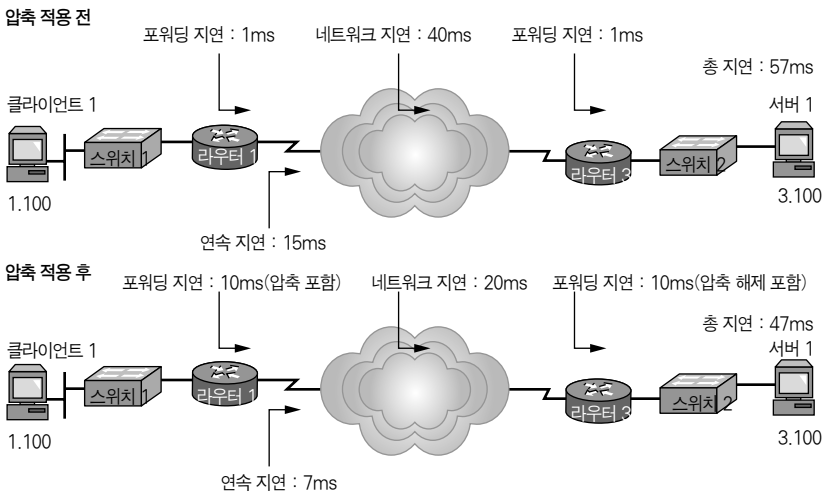


*DL : Data-Link, 각각 데이터 링크 헤더와 테일을 나타낸다.

표시한 그림이다. 두가지 타입의 압축방법 모두 CPU와 메모리 부하를 발생시킨다. 어느 압축 알고리즘이나 계산하는데 걸리는 시간만큼 패킷의 지연은 늘어나기 마련이다. 하지만 압축 과정에서 작아진 패킷들은 적은 대역폭에서도 높은 성능을 발휘할 수 있다.

(그림 2)는 압축에 의한 지연과 대역폭의 변화에 대해 나타낸 그림이다. 압축 알고리즘에 의한 계산시간 증가로 포워딩 지연은 증가하지만, 압축에 의해 작아진 패킷 덕에 연속 지연이나 큐잉 지연은 줄어든다. 또한 프레임 릴레이와 ATM 네트워크 환경이라면 네트워크 지연도 줄어든다(네트워크 지연도 패킷의 크기에 비례하기 때문).

그림 2 | 압축 적용 전 · 후 지연과 대역폭 변화 비교



(그림 2)에서 1500바이트 패킷이 클라이언트에서 서버까지 전송하는 상황을 보면, 먼저 압축하지 않은 상황에서의 총 지연시간은 57(1+15+40+1)ms인 것을 알 수 있다. 압축 비율을 2:1로 적용했을 때 47(10+7+20+10)ms로 10ms 정도 줄어든 것을 알 수 있다. 실제적으로는 10ms의 지연시간이 줄어든 효과보다는 WAN 구간에서 전송할 수 있는 효율이 두배로 늘어났다는 점이 더 중요하다고 볼 수 있다. 물론, 전체적으로 두배의 전송 효율이 늘어난 것은 아니지만 WAN 구간에 의해 전체적인 성능이 결정되는 상황을 감안하면, 압축 기법은 실제로 전체적인 성능 향상에 크게 도움이 될 수 있다. 최근에는 압축 알고리즘을 하드웨어에서 지원해 CPU나 메모리의 부하를 줄임과 동시에, 압축으로 인해 생기는 지연도 크게 줄어들었다.

페이로드 압축

페이로드 압축은 패킷의 3계층 헤더를 포함한 데이터 필드 부분을 압축하는 기법으로, 패킷의 크기가 클수록 압축 효율이 좋아진다. 라우터에서 지원하는 페이로드 압축 기법은 스택커(stacker), MPPC(Microsoft Point-to-Point Compression), 프레딕터(predictor) 등 3가지가 있다. 프레딕터는 일종의 코드북을 메모리에 저장하고 있다가 압축하고자 하는 문자열을 코드북의 내용과 비교해 같은 내용이 있으면 코드북의 자리값을 표기하는 방식으로 압축을 한다. 프레딕터는 메모리를 많이 사용하기 때문에 메모리 크기에 의존적이라고 할 수 있다. 스택커와 MPPC는 압축하고자 하는 문자열을 CPU가 Lempel-Ziv라는 압축 알고리즘을 이용해 압축한다. CPU를 많이 사용해 계산하기 때문에 CPU에 의존적이라고 할 수 있다. 페이로드 압축 기법은 실제 적용시 유의할 사항이 몇 가지 있는데 다음과 같다.

- WAN 구간의 지원 여부에 따른 압축 기법 선택
- 라우터의 CPU와 메모리에 따른 압축 기법 선택
- 적용시 양쪽 장비에 동일 압축 기법 적용

표 1 | 포인트 투 포인트 페이로드 압축 기법 비교

구분	스택커	MPPC	프레딕터
Lempel-Ziv 압축 알고리즘 사용	○	○	X
코드사전을 이용한 압축 기법 사용	X	X	○
HDLC 지원	○	X	X
X.25 지원	○	X	X
LAPB 지원	○	X	○
프레임 릴레이 지원	○	X	X
PPP(Point-to-Point) 지원	○	○	○
ATM 지원	○	○	○

* HDLC : High-Level Data Link Control
* LAPB : Link Access Procedure Balanced

헤더 압축

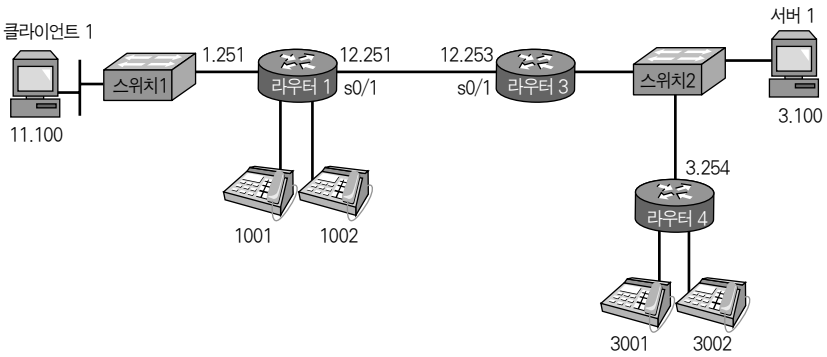
헤더 압축은 패킷의 헤더(3, 4계층) 부분을 압축하는 기법으로, 2가지 종류로 분류돼 사용된다. 하나는 TCP 헤더 압축으로 IP와 TCP 헤더부분 40바이트를 3~5바이트로 압축하며, 또 다른 하나는 RTP 헤더 압축으로 IP, UDP, RTP 헤더 40바이트를 2~4바이트로 압축한다. TCP 헤더 압축의 경우 압축의 효율을 최대한으로 높이기 위해서는 패킷의 크기가 작아야 한다. 예를 들어 64바이트의 패킷과 1500바이트의 패킷에 각각 TCP 헤더 압축을 적용해 보자. 64바이트의 패킷 중 40바이트의 헤더 부분을 압축하면 27~29바이트로 줄어들어, 압축전과 비교하면 약 2.37배(64/27)의 압축 효과가 생긴다. 그러나, 1500바이트의 패킷은 40바이트의 헤더 부분을 압축해도 1463바이트 정도로 줄어들어 압축전과 비교하면 약 1.02배(1500/1463)의 압축

효과밖에 생기지 않는다.

RTP 헤더 압축은 VoIP 트래픽에 적용할 때 가장 탁월한 효과를 발휘한다. VoIP 트래픽은 대부분 작은 패킷을 사용하기 때문이다. 예를 들면, 라우터에서 G.729 코덱을 사용하면 데이터의 크기는 20바이트 밖에 되지 않는다. 20바이트의 데이터에 IP, UDP, RTP 헤더가 40바이트 붙어 60바이트의 VoIP 패킷이 되기 때문에 헤더 부분을 4바이트 정도로 압축을 하면 VoIP 패킷의 크기는 60바이트에서 24바이트로 줄어들게 된다. (표 2)는 VoIP 데이터에서 헤더 압축 적용 전과 적용후의 총 필요 대역폭의 비교를 나타낸 것이다.

그림 3 | LFI의 적용

그림 4 | LE 메커니즘 초급 설정 연습 문제



연속 지연 줄이는 'LFI'

압축기법 사용의 가장 큰 이유는 적은 대역폭을 조금이라고 넓게 사용하고자 하는 목적으로, 성능을 향상시키는 측면에서 접근한 것이다. 이에 반해 LFI(Link Fragmentation and Interleaving)는 직접적으로 연속 지연을 줄이기 위한 목적으로 만들어진 것이다.

먼저, 이해를 돕기 위해 연속 지연에 대해 간단하게 설명하도록 하겠다. 연속 지연은 프레임을 물리적인 링크에 실어보내는 과정에서 걸리는 시간을 말한다. 만일 물리적인 대역폭이 xbps라고 하면 1/x초 동안에 1비트를 전송할 수 있으며, y비트의 프레임을 전송한다면 y/x 초가 걸리게 된다. 실제로 예를 들면 56Kbps의 링크에서 1500바이트 프레임(1만 2000비트)을 전송한다고 가정하면, 연속 지연은 12,000/56,000초가 걸리므로 214ms가 된다. 여기에서 문제가 발생할 수 있다. 1500바이트의 패킷 뒤에 60바이트의 VoIP 패킷이 있게 되면, 이 VoIP 패킷은 자기자신의 연속 지연은 약, 8.5ms 밖에 안되지만 실제로는 앞의 패킷이 전송될 때까지 기다려야 하기 때문에 연속 지연은 222.5ms가 된다. 이 수치는 이미 VoIP 트래픽으로서는 사용할 수 없다는 것을 뜻한다(VoIP 트래픽은 총지연의 합이 300ms 이내로 전송돼야 정상적으로 서비스를 할 수 있다).

위의 예처럼 작은 패킷들이 큰 패킷 뒤에서 서비스 되는 과정에서 생기는 문제점을 해결하기 위해 만든 기술이 LFI인 것이다. LFI는 작은 패킷의 앞에 있는 큰 패킷을 여러 조각으로 분해(Fragment)한 후 작아진 조각들 사이에 작은 패킷을 끼워넣어 먼저 서비스되게 만드는 기술이다.

(그림 3)을 보면 앞에 있는 1500바이트의 패킷을 300바이트×5개로 분해한 후 첫번째 조각을 서비스한 후 두번째 조각의 자리에 60바이트의 VoIP 패킷을 끼워넣은 모습을 볼 수 있다. 이 LFI 기술을 적용한 후 VoIP 패킷의 연속 지연은 52.5ms(44+8.5)로 적용하기 전 222.5ms에 비하면 170ms의 지연시간을 줄여주기 때문에 실제로 양질의 VoIP 패킷으로 서비스할 수 있게 된 것이다(나눠진 300바이트의 패킷에 8바이트의 플래그먼트 헤더와 2바이트의 플래그먼트 테일러가 붙어서 실제 나눠진 패킷의 크기는 311바이트가 된다).

표 2 | VoIP패킷의 RTP 헤더 압축 적용 비교표

코덱	VoIP 원본 필요 대역폭	IP/UDP/RTP 헤더 사이즈	2계층 헤더 타입	2계층 헤더 사이즈	전체 필요 대역폭
G.711	64Kbps	40bytes	이더넷	14byte	85.6Kbps
G.711	64Kbps	40bytes	MLPPP/FR	6byte	82.4Kbps
G.711	64Kbps	2byte(cRTP)	MLPPP/FR	6byte	67.2Kbps
G.729	8Kbps	40bytes	이더넷	14byte	29.6Kbps
G.729	8Kbps	40 bytes	MLPPP/FR	6byte	26.4Kbps
G.729	8Kbps	2byte(cRTP)	MLPPP/FR	6byte	11.2Kbps

실전! 연습문제

그럼 이제부터는 연습문제를 풀어보면서 LE(Link Efficiency) 메커니즘 초급 설정에 대해 알아보자.

☞ 연습문제 1

(그림 4)를 보고 다음의 조건을 만족하도록 라우터 3에 설정하라.

① 라우터 3는 포인트 투 포인트 프로토콜 128Kbps로 연결돼 있다.
② 페이로드 압축 기법을 이용해 설정하라.
③ 메모리의 여유를 활용할 수 있게 설정하라

☞ 연습문제 1에 대한 설명

압축 적용시 유의사항을 확인하며 적용한다.

① WAN 구간의 지원 여부에 따른 압축 기법 선택
→ PPP에서는 3가지 압축기법이 모두 지원된다.
② 라우터의 CPU와 메모리에 따른 압축기법 선택
→ 페이로드 압축기법 중 메모리를 주로 사용하는 기법은 프레딕터이다.
③ 적용시 양쪽 장비에 동일 압축기법 적용

☞ 연습문제 1 설정하기

```
!
interface Serial 0/1
  bandwidth 128                    →대역폭 설정
  ip address 192.168.12.253 255.255.255.0
encapsulation ppp                  →포인트 투 포인트 환경설정
  compress predictor               →페이로드 압축기법(프레딕터) 설정함
  clockrate 128000
!
```

☞ 연습문제 1 설정 확인하기

```
r3# show compress
 Serial0/1
   Software compression enabled
   uncompressed bytes xmt/rcv 323994/5494
   compressed     bytes xmt/rcv 0/0
   1    min avg ratio xmt/rcv 1.023/1.422
   5    min avg ratio xmt/rcv 1.023/1.422
   10   min avg ratio xmt/rcv 1.023/1.422
   no bufs xmt 0 no bufs rcv 0
   resyncs 2
```

☞ 연습문제 2

(그림 4)를 보고 다음의 조건을 만족하도록 라우터 3에 설정하라.

① 라우터 3는 포인트 투 포인트 프로토콜 128Kbps로 연결돼 있다.
② TCP 헤더 압축 기법을 이용해 설정하라.

☞ 연습문제 2 설정하기

```
!
interface Serial 0/1
  bandwidth 128                    →대역폭 설정
  ip address 192.168.12.253 255.255.255.0
encapsulation ppp                  →포인트 투 포인트 환경설정
  ip tcp header compression        →TCP 헤더압축기법설정
  clockrate 128000
!
```

☞ 연습문제 2 설정 확인하기

```
r3# show ip tcp header-compress
TCP/IP header compression statistics:
   Interface Serial0/1:
     Rcvd:       252 total,  246 compressed,  0 errors
                 0 dropped,  0 buffer copies,  0 uffer failures
     Sent:        371 total,  365 compressed,
                 12995 bytes saved,  425880 byte sent
                 1.3 efficiency improvement factor
     Connect:    16 rx slots,  16 tx slots,
             218 long searches,  6 misses 0 collisions,  0 negative cache hits
                 98% hit ratio,  five minute miss rate 0 misses/sec,  1 max
```

생각보다 쉽다는 느낌이 늘 것이다. 이번에는 조금 복잡한 환경에서 LE(Link Efficiency) 메커니즘 고급 설정 연습을 해보자.

☞ 연습문제 3

(그림 5)를 보고 다음의 조건을 만족하도록 R3에 설정하여라.

① R3과 R1은 Frame-Relay 128 kbps로 연결되어있다.
② 연속 지연이 10ms가 되도록 분해(Fragment)하여라.
③ 모든 트래픽은 64 kbps 로 쉐이핑 하여라. (FRTS사용)

④ Tc 는 10ms로 설정하여라.

⑤ Be 는 사용하지 말아라.

⑥ 서브인터페이스를 이용해 설정하여라.

☞ 연습문제 3에 대한 설명

① 128Kbps 회선에서 연속 지연이 10ms가 되도록 하기 위해서는

→ x/128000 = 0.01 → x는 1280비트 → 바이트로 바꾸면 160바이트임.

② Tc를 10ms로 설정하기 위해서는

→ Bc/CIR(64000bps) = Tc(10ms) → Bc는 640비트이면 됨.

☞ 연습문제 3 설정하기

```
!
interface Serial0/0
  no ip address                    → 서브인터페이스 사용하기 위해IP를 부여하지않음
  encapsulation frame-relay        → 프레임 릴레이 설정을 선언함
  load-interval 30
  clockrate 128000                 → 클럭레이트를128000으로설정함
  bandwidth 128                    → 대역폭을 128Kbps로 설정함
  frame-relay traffic-shaping      → 프레임 릴레이 트래픽 쉐이핑을 선언함
!
interface Serial0/0.1 point-to-point      → 서브 인터페이스 선언
  ip address 192.168.2.253 255.255.255.0  → 소스 포트가 TCP 80인 패킷을 구분
  frame-relay class shape-all-64          → shape-all-64라는 클래스 맵을 실행시킴
  frame-relay interface-dlci 102          → 프레임 릴레이 dlci 값을설정함
!
map-class frame-relay shape-all-64   → shape-all-64라는 클래스 맵을 선언
  frame-relay traffic-rate 64000 640 → FRTS를 실행함;Tc를 10ms로 설정하기위해
                                       Bc값을CIR의 1/100으로 설정함
  no frame-relay adaptive-shaping
  frame-relay fair-queue
  frame-relay framement 160        → 64Kbps의 대역폭에서 연속 지연을
                                     10ms로하기 위해 160바이트로 분할함
```

☞ 연습문제 3 설정 확인하기

```
R3# show frame-relay fragment interface s 0/0.1 102
 fragment size 160     fragment type end-to-end
 in framgmented pkts 52          out framgmented pkts 9367
 in framgmented byte 5268        out framgmented byte 1511866
 in un-fragmented pkts 5552      out un-fragmented pkts 6320
 in un-fragmented byte 341743    out un-fragmented byte 405268
 in assembled pkts 5577          out per-fragmented pkts 7387
 in assembled bytes 346749       out per-fragmented byte 1862784
 in dropped reassembling pkts 0  out dropped fragmenting pkts 0
 in timeouts 0
 in out-of-sequence fragments 0
 in fragments with unexpected B bit set 0
 in fragments with skipped sequence number 0
out interleaved packets 0

R3# show frame-relay fragment
interface       dlci       frag-type     frag-size   in-frag   out-frag
dropped-frag
Serial0/0.1   101       end-to-end          160         54       9700
0
```

조금 복잡해 보이지만 지금까지 배워왔던 설정들을 생각해보면 별로 어렵지 않을 것이다. 이번호를 끝으로 이론적인 부분에 대한 설명은 마무리한다. 중요한 것은 어느 한 기술을 아는 것이 아니고 전체적인 흐름을 파악하는 것이다. 가장 먼저 클래스를 구분하고 마킹을 한 후 컨디셔너를 거쳐 쉐이핑 또는 폴리싱을 적용하고, 큐잉을 이용해 차별화된 서비스를 제공한 후, 필요시에는 RED나 WRED를 적용하며 WAN 구간의 대역폭이 적으면 압축이나 LFI를 적용하는 것처럼, 엔드 투 엔드로 물 흐르듯이 막힘없이 흘러가게 하는 것이 QoS에서는 중요하다. 다음시간에는 QoS를 공부하고 적용하고 싶어하는 사람들을 위해 가상의 시나리오를 만든 후 엔드 투 엔드 QoS를 적용하는 것에 대해 살펴볼 것이다. NET

그림 5 | 연습 문제 3의 예제