1971226 김인우 네트워크 스터디 week4

3장 68 ~ 89 페이지

TCP

- Transmission Control Protocol
- 웹, 이메일, FTP 등 정확한 데이터 전달이 필요한 통신에 사용
- 점대점(point-to-point): 단일 송신 측, 단일 수신 측이 1대1로 통신한다.
- 신뢰적, 순서 유지된 바이트 스트림
  - 에러 발생시 재전송 요청으로 신뢰성을 보장한다.
- 파이프 라이닝: 정해진 윈도우 크기만큼만 전송하다.
  - TCP의 혼잡 제어 및 흐름 제어를 통해 윈도우 크기 설정
- 연결 지향(connection-oriented)
  - handshaking(제어 메시지의 교환)으로 송수신 측의 상태를 초기화한 후 데이터 교환
- 흐름 제어(flow control): 수신 측의 한계를 감안해 송신한다.

TCP segment

- 출발지 프로세스와 목적지 프로세스 포트 넘버가 각 16비트
- sequence number : 순서 정보 나타냄 - 세그먼트 단위가 아닌 데이터를 바이트 단위로 센다

  -> 1, 2, ...로 차례로 늘지 않고 10바이트 보내면 0이었다가 10이 된다.
- acknowledgement number: 다음에 받아야 할 데이터의 넘버를 준다. 0~9바이트를 잘 받았다면 다음 바이트는 10번부터이므로 10이 된다.
- 위의 두 number는 송신측, 수신측 둘다 사용할 수 있다.
- head len : 헤더 길이. 기본적으로 20바이트이다. 더 긴게 전송될 수 있다.
- URG : urgent data 이것이 중요한 데이터임을 표시한다. 지금 실제로 인터넷에서는 사용되지 않는다. 초기 군사용으로 인터넷이 설계될 때 생긴것..

- ACK : 액크놀리지 넘버를 포함하고 있는지 아닌지 표시한다.

- RST, SYN, FIN : TCP 연결 관련(설립, 해제 명령)

- receive window : ack 없이 한 번에 얼마나 많은 데이터를 전송할 수 있나 표시. 데이터 전송 속도 조절에 사용 / 수신 측이 받기 원하는 바이트 수(흐름제어)

- checksum : 데이터에 오류를 체크한다.

TCP 흐름 제어 1

- 수신 측이 감당하지 못하게 빠르게 데이터를 보내면 문제가 발생한다. -> 수신 측에 맞춰 적당한 속도로 보내야 한다.

- 송신 측이 보낸 데이터가 TCP socket reveiber buffers라는 곳에 쌓인다.

- 어플리케이션이 위 버퍼에서 데이터를 가져가는 속도, 버퍼에 데이터가 쌓이는 속도가 있어서 둘이 균형을 이뤄야 한다.

- 쌓이는 속도가 더 빠르면 점점 데이터가 쌓여서 버퍼가 가득 차게된다. -> 데이터 손실이 일어날 수 있음

- 수신 측이 버퍼에서 데이터를 가져가는 속도가 더 빠르면 버퍼가 비게되서 데이터를 기다려야 함.

- 흐름 제어는 버퍼에 데이터가 너무 빨리 쌓이지 않도록 넘치지 않게 조절하는 것이다. -> receiver window size로 조절함.

  - 송신자는 항상 수신자에 맞춰야 한다.

TCP 흐름 제어 2

- 수신 측이 보내는 TCP 헤더의 rwnd 값을 통해 비어있는 버퍼 공간을 알려줄 수 있다.

  - RcvBuffe 크기는 소켓 옵션으로 설정 (전형적인 default는 4096 byte이다. 점점 증가 중이다.)

  - 많은 OS가 RsvBuffer를 자동 조절한다.

- 송신 측은 ACK 안된 데이터의 양을 수신 측의 rwnd 값으로 제한한다.

  + Sliding window : TCP에서 윈도우를 통해 흐름제어 하는 프로토콜

- 수신 측 버퍼가 넘치지 않도록 보장한다.

TCP 혼잡 제어

- additive increase(AI) : 송신하는 데이터(cwnd)를 늘릴 때는 천천히 조심해서 늘린다.
  - 선형적으로 기존에 10이었다면 11, 12, 13...
  - congestion avoidance라고도 한다.
- multiplicative decrease(MD) : 송신하는 데이터(cwnd)를 줄일 때는 확 줄여야 함
  - 예: 10 -> 5 절반이나 더 급격히 감소시킨다.
  - fast recovery라고도 한다.
- cwnd(congestion window size) 와 rwnd(receiver window size) // 원활하면 <=, 혼잡하면 <(cwnd는 rwnd를 넘으면 안되므로 더 줄임.)
- cwnd는 시간의 지남에 따라 톱날 모양으로 그래프를 그린다.
- 혼잡 상황을 아는 방법 2가지 : timeout이 발생(더 심각함), 중복된 ack를 3번 받았을 경우

TCP Slow Start(SS)

- 처음에는 cwnd를 1로 설정해서 보냄(가장 작은 단위로)
- 잘 되면 2배로 늘려서 보냄 // 목욕탕 뜨거운 물에 손부터 넣어보고 점점 몸을 넣는 것처럼.
- 어느 정도 수준이 되면 선형적으로 바뀜 // 처음엔 SS로 시작해서 어느 정도 수준이 되면 AI로 한다.
  - 수준의 기준값은 상황에 따라 달라짐. 보통 목표로 하는 윈도우 사이즈의 절반 값으로 함

TCP : detecting(혼잡상황인지 알아냄), reacting to loss

- 1. 타임 아웃 - 심각한 상황
  - 윈도우 사이즈를 초기(1)로 줄인다. -> SS 상태로 되돌린다.
- 2. 타입 아웃은 발생하지 않았지만 ack가 3번 중복되었을 때
  - TCP RENO버전이면 절반으로 줄인다.
- TCP Tahoe버전은 무조건 혼잡 생기면 1로 되돌림

TCP 3-단계 핸드셰이킹

- 1. 클라이언트가 초기 순서 번호 X를 고르고 TCP SYN 메시지 송신 // SYNbit = 1, Seq = x

- 2. 서버는 초기 순서 번호 y를 고르고 SYN에 대한 응답으로 TCP SYNACK 메시지 송신

  // SYNbit = 1, Seq = y, ACKbit = 1, ACKnum = x + 1

- 3. SYNACK(x)는 서버가 살아있음을 의미함. SYNACK에 대한 ACK를 송신.

  - 클라이언트가 서버로 보내는 데이터를 이 segment에 전송 가능

  - ACKbit = 1, ACKnum = y + 1 -> ACK(y)는 클라이언트가 살아있음을 의미함

- 클라이언트와 서버 모두 처음엔 LISTEN(대기) 상태이고 설립되면 ESTAB상태가 된다

TCP 연결 닫기 과정

- 1. 클라이언트가 ClientSocket.close()로 FINbit = 1, seq = x보냄 -> 더 이상 송신은 불가하고 받을 수는 있음 // ESTAB -> FIN_WAIT_1 상태

- 2. 서버는 응답으로 ACK bit = 1, ACKnum = x + 1을 보냄 -> 아직 데이터 보낼 수 있는 상태. (남은 데이터 처리)

  - ESTAB -> CLOSE_WAIT 상태 / 클라이언트는 메시지 받고 FIN_WAIT_2가 됨.

- 3. 서버에서 더 이상 보낼 데이터 없으면 FINbit = 1, seq = y 보냄 -> 더 이상 송신 불가한 상태. // LAST_ACK -> CLOSED

- 4. 클라이언트가 응답. ACKbit = 1, ACKnum = y + 1 // TIMED_WAIT -> CLOSED

- 양 쪽 모두 ESTAB 단계에서 CLOSED 단계가 된다. 한 번에 되는게 아니라 중간에 위 과정을 처리하면서 중간 단계들을 거친다.

- TIMED_WAIT단계는 클라이언트에서 마지막으로 서버에게 보낸 응답이 손실될 수 있기 때문에 max segment lifetime의 2배의 시간을 기다렸다 닫는다.

UDP

- User Datagram Protocol

- UDP 세그먼트는 사라지거나 순서 바뀌어 전달될 수 있다.

- 오류 발생해도 수정하지 않는다.

- TCP에 비해 간단한 프로토콜이다.
  - 핸드셰이킹 없이 비연결형 통신
  - 헤더 크기가 작다.
  - 혼잡 제어 없다.
- Checksum: 데이터 손실 여부 확인
- 신뢰성보다 빠른 요청/응답, 실시간 스트리밍, 음성 서비스 등에 사용된다.

+ 요즘은 버퍼링을 이용해서 TCP를 사용해서 스트리밍 하기도 함. - 넷플릭스

- 브로드캐스트: 송신자가 전송한 데이터가 네트워크에 연결된 모든 호스트에 전송된다.
- 멀티캐스트: 한 번의 송신으로 데이터를 네트워크 상에 어떤 그룹에 속해있는 호스트들에게 전송한다.

UDP segment header

- application data(payload)
- source 포트 넘버, dest 포트넘버(각각 16비트)
- legnth : 헤더를 포함한 UDP 세그먼트의 바이트 단위 길이
- checksum

UDP checksum

- 목표 : 전송된 세그먼트에서 오류를 검출 : 예) 바뀐 비트
- 송신 측
  - 헤더를 포함해 세그먼트의 내용을 16비트(로 쪼갬)의 정수의 연속으로 취급
  - checksum : 세그먼트의 내용을(16비트로 쪼갠 것 다 더한 결과의 1의 보수를 취함
  - UDP checksum 필드에 checksum 값을 넣는다
- 수신 측
  - 전송받은 세그먼트의 checksum을 계산 - 송신 측의 결과와 일치하는지 확인.
  - 결과의 모든 bit가 1인가?

- NO : 오류 발견 / YES : 오류 미발견. 오류 있을 수 있음.

인터넷 checksum 예시

- 두 16비트 정수를 더한다

1 1 1 ... 0 0 0

0 0 1.... 1 1 0 // 두 줄을 더함

- 비트가 추가되면 첫번째 자리에 올림자리 1을 취함

- 덧셈 결과인 16비트를 checksum으로 하여 전송.

-> 송신 측의 checksum은 1의보수를 취하므로 수신측의 checksum을 더해서 모두 1이면 오류 미발견.

- TCP에서도 checksum 사용 가능. // UDP는 에러 발견해도 보고만 하고 수정안함. TCP는 재전송 요청함.

네트워크 상태 확인

- netstat -n명령으로 트랜스포트 계층의 프로토콜, 로컬 컴퓨터의 ip주소:포트번호, 통신 상대의 ip주소:포트번호, 접속 상태 표시

- 상태 값 의미

  - ESTABLISHED: TCP로 접속되어 통신되는 상태

  - LISTEN: 서버가 수신 대기 상태. -a 옵션으로 확인 가능하다.

  - TIME_WAIT: 접속 종료하려는 상태

IP 주소 체계

- IP 주소: 호스트, 라우터의 인터페이스를 위한 32-bit 식별값

- interface: 호스트/라우터와 물리 링크 간의 연결

  - 라우터는 보통 복수의 인터페이스 보유

  - 호스트는 한 개 또는 두 개의 인터페이스 보유 예) 유선 이더넷, 무선 802.11

- IP 주소는 각 인터페이스에 대응됨

- 질문: 노트북, pc 하나마다 한 개의 ip주소가 할당되나?

  -> X. ip주소는 호스트 하나 당 하나가 할당되는 것이 아니라 호스트에 달려있는 네트워크 인터페이스(랜카드) 한 개당 하나에 할당된다.

  - 노트북이 이더넷 랜카드와 와이파이 랜카드를 모두 가질 수 있음. -> ip는 2개

- 3개의 네트워크를 연결하는 라우터는 3개의 네트워크 인터페이스를 통해 각각 연결됨 -> 적어도 3개 ip가 필요

  - 보통 라우터는 복수의 ip주소를 갖게 된다.

- IPv4에서 ip는 크게 4부분으로 표시 - 각 8비트로 0~255 표현 가능.

- time to live(TTL) : 남아있는 최대 hop 수(각 라우터에서 감소)

IPv6

- 초기의 동기 : 32-bit 주소 공간은 곧 고갈될 것이다. -> IPv6에서는 128-bit로 늘어남

- 추가적인 동기

  - 처리/전달 속도를 높이도록 header 형식을 개선

  - QoS를 지원할 수 있도록 header를 변경

- IPv6 datagram 형식

  - 고정된 40byte 길이의 header

  - 단편화 허용 x

- hop limit(hop count limit): TTL과 동일 기능.

IPv4에서 IPv6로의 전환

- 모든 라우터를 동시에 업그레이드할 수 없다

  - "flag day"는 불가능 -> 어느 날 한 번에 전환하는 것은 불가능하다.

  - IPv4와 IPv6 라우터가 혼재할 때 네트워크는 어떻게 운영되어야 할 것인가?

- 터널링(tunneling)

- IPv4 라우터들 사이에서는 IPv6 datagram을 IPv4 datagram의 페이로드로서 전달함 // 비효율적이지만 어쩔 수 없다.

- IPv6 라우터 간의 터널처럼 사용함.

- IPv6가 나온지 20년 지났지만 여전히 전환이 이행중이다. 지금도 혼용중.

- IPv6는 나중에 나왔으므로 IPv4 지원이 되지만 IPv4는 IPv6를 지원 못함

- 특히, IPv4의 라우터가 어떻게 IPv6 데이터를 지원하는가. -> 터널링

IPv6 - 수용

- 구글: 8%의 클라이언트가 IPv6를 사용

- NIST: 미국 정부 도메인 중 1/3이 IPv6 사용 가능

- 보급과 사용에는 오랜 시간이 걸린다 // 20년째 진행중. 페이스북, 스트리밍, 스카이프 등 응용계층의 변화에 비하면 매우 느리다.

질문자: 김인우

1. 클라이언트가 서버로 200바이트를 보내고, 서버로부터 300바이트를 받았을 때 클라이언트 측의 일련번호는 301이다. O / X

2. 서버가 클라이언트에게 연속으로 100바이트 씩 3번 전송했을 때, 클라이언트가 서버에게

확인 응답번호로 각각 101, 101, 101을 보냈다면 실패로 간주한다. O / X

3. 송신 측에서 수신 측에게 탐색 패킷을 보냈을 때, 수신 측의 응답에서 보낸 윈도우 사이즈가 0이 아니면 전송을 재개한다. O / X

4. 동영상 스트리밍 서비스는 일부 데이터가 누락되거나 왜곡되어도 문제가 없기 때문에 TCP를 사용하는 것이 좋다. O / X

5. A회사의 사무실 네트워크와 B회사의 사무실 네트워크가 서로 다른 네트워크일 때 동일한 사설 IP주소를 사용할 수 있다. O / X

6. IPv4의 헤더에서 홉 리미트는 데이터그램의 수명을 나타낸다. O / X

7. IP 어드레스는 지역과 네트워크 단위로 할당되기 때문에 IP주소를 통해 네트워크의 위치를 어느정도 파악할 수 있다. O / X

답

1. X p.71

2. O p.73

3. O p.74

4. X p.76

5. O p.85

6. X p.87, 89

7. X p.82

질문자: 최지은

1. TCP의 데이터 분할 단위는 세그먼트이고, UDP의 데이터 분할 단위는 데이터그램이다.(o/x)

2. TCP통신에서 송신 측과 수신 측이 제시한 MSS크기가 다를 경우 큰 값을 따라간다.(o/x)

3. 재전송 방법 중 하나인 선택적 확인 응답은 송수신 측 모두가 이것을 지원해야 사용할 수 있다. (o/x)

4. TCP에는 브로드 캐스트와 멀티캐스트 기능이 있다.(o/x)

5. 생존 기간을 초과한 패킷이 네트워크상에서 발견되면 소멸시키도록 규정하고 있다.(o/x)

6. IPv4와 IPv6는 호환 가능하다.(o/x)

7. IPv6는 라우터에서 데이터를 분할하지 않는 방식으로 사양이 만들어져 있다. (o/x)

답

1. O

2. X

3. O

4. X

5. O

6. X

7. O