CS118 Discussion 1C, Week 3

Zhehui Zhang

Outline

- Review
- Application Layer Protocol:
  - SMTP, DNS, P2P
- Transport Layer Protocol:
  - UDP, principles of reliable transport protocol
  - Principles of reliable data transfer

Review: HTTP

- HTTP: a stateless protocol on top of TCP
  - HTTP is based on pull model
  - Persistent HTTP V.S. Non-persistent HTTP
  - Method Types: GET, HEAD, POST, PUT, DELETE, Conditional GET
  - What if we want stateful service? Cookie
  - Web Caches (proxy server)

Question for review

Consider a short, 10-meter link, over which a sender can transmit at a rate of 150 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or handshaking) are 200 bits long. Assume that *N* parallel connections each get *1/N* of the link bandwidth. Assuming propagation speed of light 3*10^8

Now consider the HTTP protocol, and suppose that each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer.

Question for review

Consider a short, 10-meter link, over which a sender can transmit at a rate of 150 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or handshaking) are 200 bits long. Assume that *N* parallel connections each get *1/N* of the link bandwidth. Assuming propagation speed of light 3*10^8

Now consider the HTTP protocol, and suppose that each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer.

Ignoring transmission delay.

How many RTTs:

1) With non-persistent HTTP?
2) With non-persistent HTTP and 10 parallel TCP connections?
3) With persistent HTTP?
4) With persistent HTTP and pipeline?

Question for review

Consider a short, 10-meter link, over which a sender can transmit at a rate of 150 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or handshaking) are 200 bits long. Assume that *N* parallel connections each get *1/N* of the link bandwidth. Assuming propagation speed of light 3*10^8

Now consider the HTTP protocol, and suppose that each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer.

Ignoring transmission delay.

How many RTTs:

1) With non-persistent HTTP? 2+2*10=22
2) With non-persistent HTTP and 10 parallel TCP connections? 4
3) With persistent HTTP? 2+10=12
4) With persistent HTTP and pipeline? 3

Question for review

Consider a short, 10-meter link, over which a sender can transmit at a rate of 150 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or handshaking) are 200 bits long. Assume that *N* parallel connections each get *1/N* of the link bandwidth. Assuming propagation speed of light 3*10^8

Now consider the HTTP protocol, and suppose that each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer.

What is the propagation delay?
Tp=100/(3*10^8)
What is the transmission delay for one object (10Kbits)?
Tx(data)=100*10^3/150
What is the transmission delay for one control packet (200bits)?
Tx(control)=200/(150)

Answer

Note that each downloaded object can be completely put into one data packet. Let Tp denote the one-way propagation delay between the client and the server.

First consider parallel downloads via non-persistent connections. Parallel download would allow 10 connections share the 150 bits/sec bandwidth, thus each gets just 15 bits/sec. Thus, the total time needed to receive all objects is given by:

First 2RTT to get initial object

(200/150+Tp + 200/150 +Tp + 200/150+Tp + 100,000/150+ Tp)
+ (200/(150/10)+Tp + 200/(150/10) +Tp + 200/(150/10)+Tp + 100,000/(150/10)+ Tp)
= 7377 + 8*Tp (seconds)

2RTT to get referenced objects

Link rate is 150/10 since parallel connections share the link

Then consider persistent HTTP connection. The total time needed is give by:

(200/150+Tp + 200/150 +Tp + 200/150+Tp + 100,000/150+ Tp) First 2RTT to get initial object
+ 10*(200/150+Tp + 100,000/150+ Tp)
=7351 + 24*Tp (seconds) 10RTT to get referenced objects

Assume the speed of light is $300*10^6$ m/sec, then Tp=10/($300*10^6$)=0.03 microsec. Tp is negligible compared with transmission delay.

Review: SMTP

- Can it directly send a binary file?
  - What if the binary file contains CRLF.CRLF?
    - Connection closed will be closed since the CRLF.CRLF marks the end of the message body

- How can it transmit binary file with CRLF.CRLF
  - It is necessary to convert the binary attachment into a stream of text characters (ASCII) before it is put into the part of the mail message that will be reserved for the attachment.
  - Base64 encoding converts a sequence of binary data into a sequence of characters. It expands the attachment by a factor of 4/3. So a 3 kB binary file would convert into a 4 kB text attachment.

DNS

- What is the transport layer protocol?
- How the scalability is achieved?
- Who will use iterative/recursive query?
- Why is DNS resolver needed?

Suppose that on April 19, 2019 at 15:35:21, you have issued "`dig google.com A`" to get an IPv4 address for `google.com` domain from your caching resolver and got the following result:

```
; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17779
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             239     IN      A       172.217.4.142

;; AUTHORITY SECTION:
google.com.             12412   IN      NS      ns4.google.com.
google.com.             12412   IN      NS      ns2.google.com.
google.com.             12412   IN      NS      ns1.google.com.
google.com.             12412   IN      NS      ns3.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         13462   IN      A       216.239.32.10
ns2.google.com.         13462   IN      A       216.239.34.10
ns3.google.com.         13462   IN      A       216.239.36.10
ns4.google.com.         13462   IN      A       216.239.38.10

;; Query time: 81 msec
;; SERVER: 128.97.128.1#53(128.97.128.1)
;; WHEN: Wed Apr 19 15:35:21 2019
;; MSG SIZE rcvd: 180
```

(a) What is the discovered IPv4 address of google.com domain?

(b) If you issue the same command 1 minute later, how would ANSWER SECTION" look like?

Suppose that on April 19, 2019 at 15:35:21, you have issued "`dig google.com A`" to get an IPv4 address for `google.com` domain from your caching resolver and got the following result:

```
; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17779
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             239     IN      A       172.217.4.142

;; AUTHORITY SECTION:
google.com.             12412   IN      NS      ns4.google.com.
google.com.             12412   IN      NS      ns2.google.com.
google.com.             12412   IN      NS      ns1.google.com.
google.com.             12412   IN      NS      ns3.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         13462   IN      A       216.239.32.10
ns2.google.com.         13462   IN      A       216.239.34.10
ns3.google.com.         13462   IN      A       216.239.36.10
ns4.google.com.         13462   IN      A       216.239.38.10

;; Query time: 81 msec
;; SERVER: 128.97.128.1#53(128.97.128.1)
;; WHEN: Wed Apr 19 15:35:21 2019
;; MSG SIZE rcvd: 180
```

(c) If the client keeps issuing dig google.com A every second, when would be the earliest (absolute) time the local DNS server would contact one of the google.com name servers again?

(d) If the client keeps issuing dig google.com A every second, when would be the earliest (absolute) time the local DNS server would contact one of the .com name servers?

Suppose that on April 19, 2019 at 15:35:21, you have issued "`dig google.com A`" to get an IPv4 address for `google.com` domain from your caching resolver and got the following result:

```
; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17779
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             239     IN      A       172.217.4.142

;; AUTHORITY SECTION:
google.com.             12412   IN      NS      ns4.google.com.
google.com.             12412   IN      NS      ns2.google.com.
google.com.             12412   IN      NS      ns1.google.com.
google.com.             12412   IN      NS      ns3.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         13462   IN      A       216.239.32.10
ns2.google.com.         13462   IN      A       216.239.34.10
ns3.google.com.         13462   IN      A       216.239.36.10
ns4.google.com.         13462   IN      A       216.239.38.10

;; Query time: 81 msec
;; SERVER: 128.97.128.1#53(128.97.128.1)
;; WHEN: Wed Apr 19 15:35:21 2019
;; MSG SIZE rcvd: 180
```

(a) 172.217.4.142
(b) google.com. 179 IN A 172.217.4.142
(c) (Wed Apr 17 15:35:21 2019 + 239 sec)
(d) (Wed Apr 17 15:35:21 2019 + 12412 sec)

Question

Suppose that you walked into Boelter Hall and get connected to CSD WiFi network, which automatically gave you IP address of the local DNS server. Suppose the local DNS server has just rebooted and its cache is completely empty; RTT between your computer and the local DNS serverr is 10ms and RTT between the caching resolver and any authoritative name server is 100ms; all responses have TTL 12 hours. Suppose the DNS lookup follows iterative searching.

(a) If you try to go to `ucla.edu`, what would be minimum amount of time you will need to wait before your web browser will be able to initiate connect to the UCLAs web server? Suppose the location of UCLAs web server is delegated to ucla.edu name server.

(b) What would be the time, if a minute later you will decide to go to `ccle.ucla.edu`? Suppose `ccle.ucla.edu` happen to be delegated to ucla.edu name server.

Answer

Suppose that you walked into Boelter Hall and get connected to CSD WiFi network, which automatically gave you IP address of the local DNS server. Suppose the local DNS server has just rebooted and its cache is completely empty; RTT between your computer and the local DNS serverr is 10ms and RTT between the caching resolver and any authoritative name server is 100ms; all responses have TTL 12 hours. Suppose the DNS lookup follows iterative searching.

(a) If you try to go to `ucla.edu`, what would be minimum amount of time you will need to wait before your web browser will be able to initiate connect to the UCLAs web server? Suppose the location of UCLAs web server is delegated to ucla.edu name server.

(b) What would be the time, if a minute later you will decide to go to `ccle.ucla.edu`? Suppose `ccle.ucla.edu` happen to be delegated to ucla.edu name server.

a) 310ms

b) 110ms

Application Layer: CDN

- CDN: Content Distribution Network
  - Globally distributed network of web servers
  - Stores and replicates images, videos and other files

- https://eclass.uoa.gr/modules/document/file.php/D245/2015/cdn.02f.ppt

Application Layer: protocols

- P2P: no always-on server, peers are intermittently connected
- Calculate minimum of content distribution time

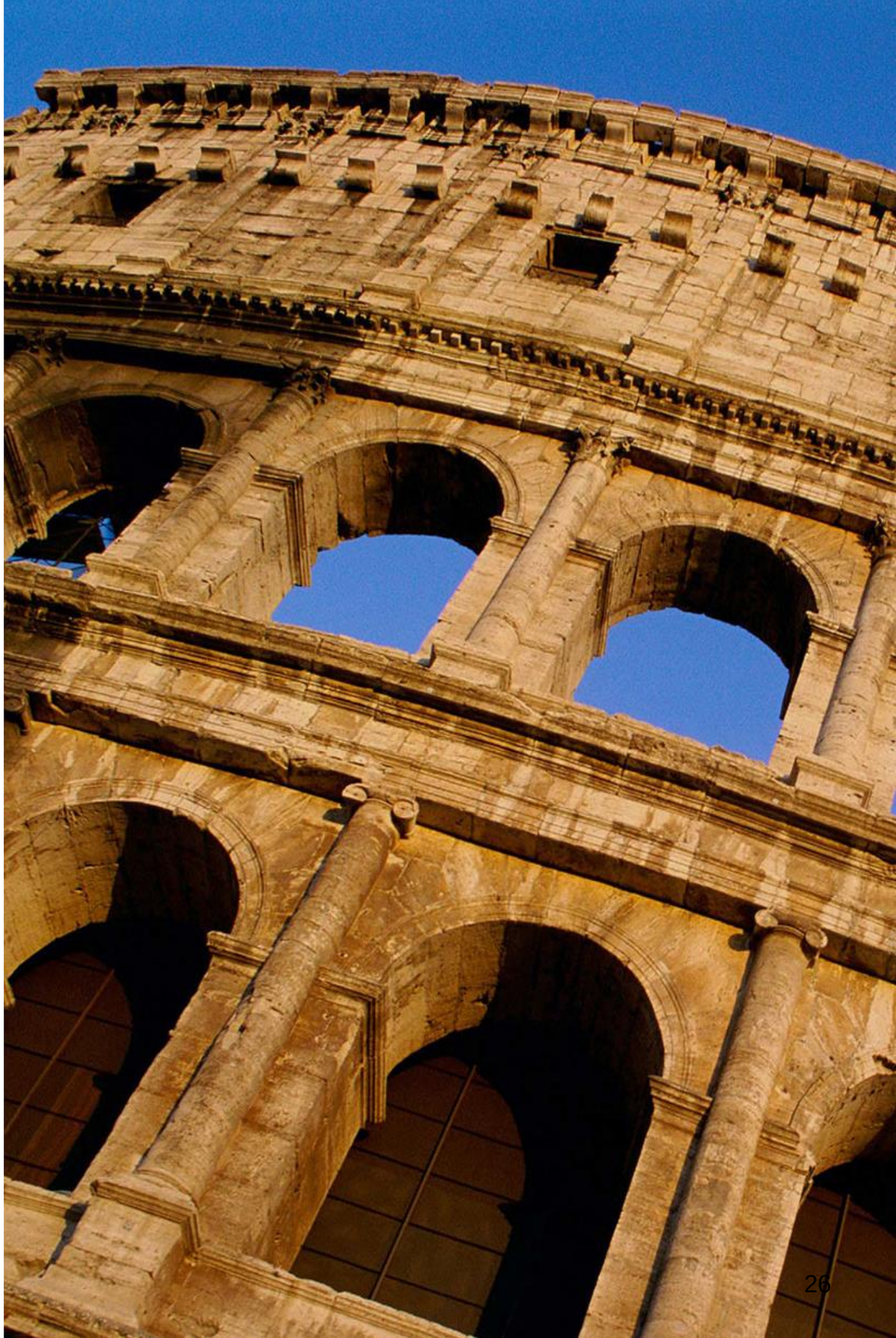
$$D_{cs} = \max(\frac{NF}{\mu_s}, \frac{F}{d_{min}})$$

$$D_{p2p} = \max(\frac{F}{\mu_s}, \frac{F}{d_{min}}, \frac{NF}{\mu_s + \sum_i \mu_i})$$

Internet video

- Multimedia streaming: perspectives
  - Rate control
  - Error control
- Streaming protocols:
  - Stream description: SDP , SMIL …
  - Stream control: RTSP — Remote control the session
  - Media transport: RTP — Error control and flow control
  - Resource reservation (if any!): DiffServ, RSVP — provide QoS
- HTTP-based: DASH (Dynamic Adaptive Streaming over HTTP)

Transport Layer



Transport Layer V.S. Network Layer

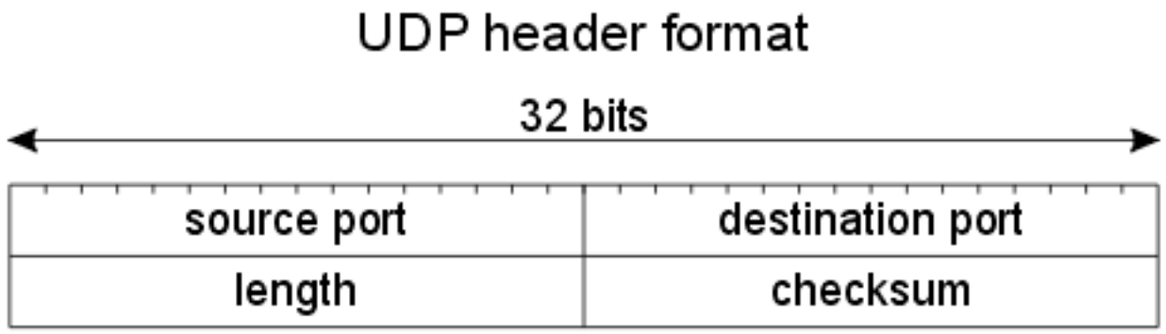
- Network layer: logical communication between **hosts**
  - **IP address** is used for identifying a host
- Transport layer: logical communication between **processes**
  - **IP address and port number** are used for identifying a process

Multiplexing and De-multiplexing

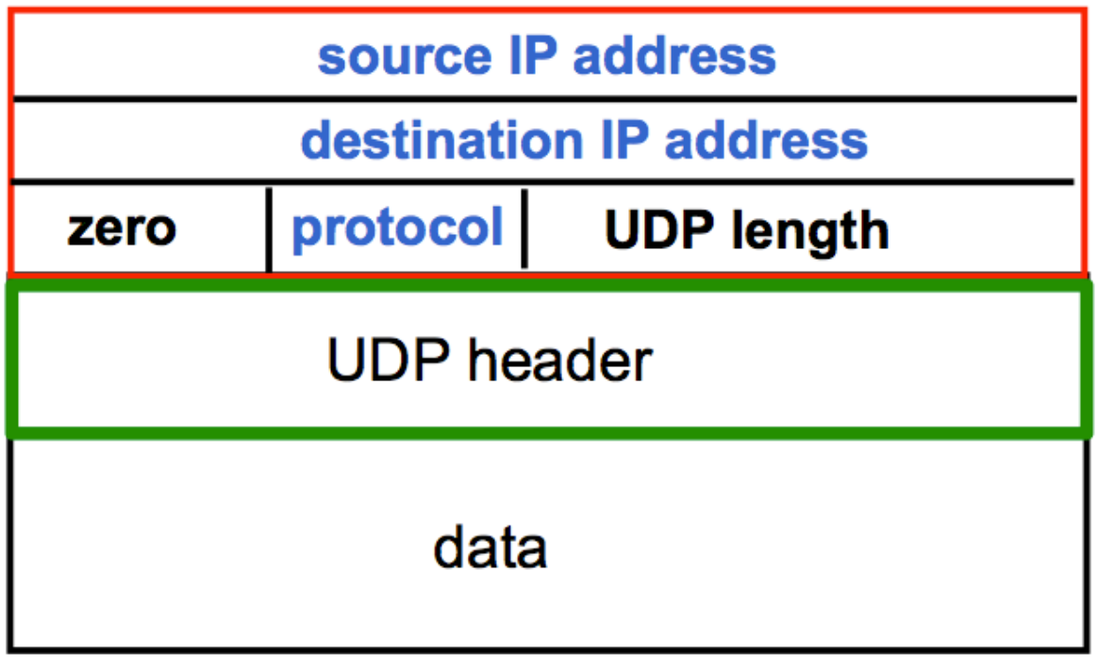
- Multiplexing at send host: gather data from multiple sockets
- De-multiplexing at receiving host: deliver received segments to the right socket
- **Five tuples** (src_ip, src_port, dst_ip, dst_port, protocol) are used for multiplexing/demultiplexing
  - How to identify a TCP/UDP socket? **lsof -i**
  - Can TCP and UDP share the same port numbers? **Yes! e.g. DNS**

UDP

- No connection establishment
- No connection state
- Small packet overhead (8 byte)



- How to calculate checksum?
  - **Pseudo header** + **UDP header** + data
  - Also applicable to TCP
  - Why pseudo header?



Principles of Reliable Data Transfer

- How to deal with bit errors?
  - Error detection (e.g. checksum)
  - Receiver feedback
  - Retransmission
    - Why not error correction?
- How to deal with duplicate packets due to retransmission? **Sequence number**
- How can the sender detect that ACK or data is lost? **Timer**

More details in next week …