Week6 Review Notes

ELEC0099: Introduce to Internet Protocol Networks 21/22

RUFENG DING

Contents

1 TCP

Two types of transport over IP:

1. TCP
   - Reliable Transport
   - Controlled rate of transmission
   - complex
2. UDP
   - Unreliable Transport
   - uncontrolled rate of transmission
   - very simple

TCP applications using TCP use usually the **client/server** paradigm
Clients connect to servers, ask for information, get the information and close the connection.

1.1 TCP functions

1. **Demultiplexing**. several flows get into the same destination and we have to deliver right packets to right process.
2. **Reliability**. some flows require no packet loss. TCP should manage retransmission.
3. **Congestion Control**. how to manage traffic go without congestion.

1.2 TCP complexity

1. in Linux TCP consists of 11443 lines of code while UDP is 1522 lines.
2. TCP is a protocol but people will also refer to the algorithm that congestion control.

1.3 TCP-3 way handshake

1. STEP1: client host send TCP SYN segment to server
   - specifies initial seq #
   - no data
2. STEP2: server host receives SYN, replies with SYNACK segment
   - server allocates buffers
   - specified server initial seq #
3. STEP3: client receives SYNACK, time replies with ACK segment, which may contain data

1.4 TCP packet

table:

<table>
<tr><td colspan="8">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="10">Sequence Number</td></tr>
<tr><td colspan="10">Acknowledgment Number</td></tr>
<tr><td>Data offset</td><td>Reserved</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>windows</td><td>windows</td></tr>
<tr><td colspan="8">checksum</td><td colspan="2">Urgent Pointer</td></tr>
<tr><td colspan="9">Options</td><td>Padding</td></tr>
<tr><td colspan="10">Data</td></tr>
</table>

1.5 TCP connection establishment

- during connection establishment the initial sequence number for both directions is chosen
- options like MSS(maximum segment size) may be negotiated
- REMEMBER: TCP connections are full-duplex with different counters and parameters in both directions
- The same packet can send data and acknowledge on the opposite direction

1.6 TCP multiplexing

1. each process ”listen” and sends to and from a port. These two ports are included in the TCP packet
2. destination ports are standardized, source ports are chosen dynamically by the operating system.

NAT: network address translation : Ports
NAT touter need to use layer form :TCP to know the ports.

1.7 TCP reliability

- each TCP packet has to be acknowledged: this is an ACK packet
- if data is bidirectional these ACK packets can contain the data in the reverse direction
- computer does not need to receive ACK of the n^{th} packet to send the $n + 1^{th}$ packet. Each time there are x number of packets unacknowledged. This is the **congestion window**

1.8 Nagle Algorithm

1. a computer only send data when user write more than MSS : avoid packet with one byte

2. this can be switched off
3. the PUSH flag is used for a sender to signal a receiver that the data should be given to the application straight away

1.9 Congestion Control

TCP sliding Window the congestion window ”contains” the packets in flight, those not acknowledged.

- every time an acknowledgment arrives the windows slides to the right.
- The size of the window will change with time.

roughly:

$$rate = \frac{Cwnd}{RTT} bytes/s \tag{1}$$

where $Cwnd$ is the size of the congestion Window and RTT is the Round Trip Time.

Slow Start .

Every TCP connection start with a congestion window of 1 and the sender then applied the **Slow Start** algorithm: every time an ACK arrives the Congestion window is increased by 1 until a value of **SSH thresh** is reached.

Congestion Avoidance

- in the second part of TCP algorithm, the window is increased by 1/cwnd each time a ACK is received
- every time a packet is lost, one reduces the cwnd by half. sshthresh is also reduced by half of the congestion windows at the time of loss

How do we detect loss?

- Nobody tells the end system that a packet was loss!!!
- The end system got two ways
  - ACK *Timesout* , that is a specific amount of time passes without havin received the correspondent ACK
  - We recive *3 dupacks* - 3 packets acknowledging packets sent after the one which loss we are detecting

Timeout Calculaion to estimate if a packet is lost we need to calculate RTO (Retransmission Timeout Value) First one has to estimate the RTT

$$R = aR + (1 - a)M \tag{2}$$

where M is the measured RTT, a is recommended to be 0.9

$$RTO = R \times b \tag{3}$$

where b is recommended to be 2
some implementations complicate this a bit by taking into account the mean deviation
because when we receive a ACK of a retransmitted packet we don't know for sure which packet is being acknowledged. These measurements are never taken into account for RTO calculation.(Karn's algorithm)

TCP with aprropriate byte counting RFC 3465 proposed that TCP counts bytes instead of packets : this prevents artificially increasing congestion windows with very small packets(only require changes in the server)

Acknowledgments in practice in reality TCP acknowledgement sends the next byte to be expected without any ”holes”

2 Advanced TCP

2.1 Sharing congestion state

- normal operation each connection keeps its own information(state)
- if the server got several connections to the same client this information can be shared.
  - Temporal sharing : sharing with CLOSED connections
  - Ensemble sharing : sharing with ACTIVE connections

2.2 TCP options

TCP can be extended by defining options: define an option number, length and specific data.eg:

- MSS
- Window Scale Option
- Timestamp option
- PAWS-Protection against wrapped Segment numbers
- TCP extension for Transactions

2.3 TCP URGENT flag

app using TCP can signal to the other side that some data is urgent and should be delivered as soon as possible
This use two fields on the TCP packet

- URG bit is sent
- URGENT POINTER is set to the offset where the urgent data begins. This pointer should be added to the sequence number to determine where is the first byte of the urgent data

2.4 TCP connection Close

to close a connection 4 packets are needed. Both side should send a FIN and receive a ACK. This way data is delivered to the app on both side.

1. FIN(A)
2. ACK of FIN(B)
3. FIN(B)
4. ACK of FIN(A)

2.5 TCP Selective Acknowledgements Option

- TCP option allows receivers to transmit more details about packets not received.
- Receiver informs the data sender of non-contiguous blocks of data that have been received and queued.

2.6 delayed acknowledgements

often a receiver delays sending an ACK to save bandwidth:

- receives more data which is then acknowledged
- has do send some data back and piggybacks the ACK in the data packet

200ms in Windows by default

2.7 TCP Keepalive

- TCP connection can be alive for months without any data being exchanged
- TCP provides a *Keepalive* option but its use is controversial(keep telling other side: I am alve)

2.8 TCP Cubic

optimized for high bandwidth, high delay networks
two operating regions:

- first is a concave portion where the window quickly ramps up to the window size before the last congestion event.
- Next is the convex growth where CUBIC probes for more bandwidth, slowly at first then very rapidly

$Cwnd$ function is:

$$W(t) = C(t-K)^3 + Wmax \tag{4}$$

where C is a constant (default 0.4), t is elapsed seconds and K is the time period the function takes to increase W to $Wmax$.

2.9 TCP BBR

old idea in the Internet (see TCP Vegas) is to detect congestion before loss and to react less drastically

2.10 Buffer Bloat

Ironically, sometimes more memory in routers can cause worse performance. This happens particularly in home routers. 1-Application delay may increase. 2-More memory implies a bigger delay in congestion signal.

2.11 Explicit Congestion Notification(ECN)

Routers can tell the end-systems that the network is getting congested and therefore they should reduce their Congestion windows. This technique is very efficient but demands that the *routers play the game*. Uses two bits[DSCP][CU] of Type of service field in the IP packet.

3 UDP

User Datagram Protocol

- Provides multiplexing/demultiplexing through UDP ports.
- no reliability (no ACKs)
- no congestion control
- app just send data to IP layer at the rare they want/can:
  no connection establishment and teardown $->$ connectionless service.

3.1 UDP header

$|<------32bits------>|$

Source Port	Destination Port
Length	UDP checksum
data	

input to checksum:

- UDP header
- UDP data
- IP source and destination address
- in IPv4 is optional(all zeros)
- compulsory in IPv6

UDP reliability some applications do not require 100% reliability. If you want to use UDP you have to manage reliability yourself.

UDP Congestion control UDP is the best solution for now in : some app do not cope well with the saw-tooth behavior of TCP (not tolerant to changes in the transfer rate) but the widespread use of UDP could take the Internet to congestion collapse.

3.2 Who uses UDP?

mainly two types of applications:

1. applications with very small data transfers(eg. DNS) where establishing the connection would be a big overhead.

2. Real time applications where congestion control would be damaging:
   - voice over OP
   - video-conferencing
   - Telemetry

   Remember: Multicast has to use UDP.9

4 Applications

Operating System Kernel(From low level to high level)

- Ethernet, Wi-Fi, FDDI, SDH
- IP, ICMP, IGMP
- TCP, UDP

all upper : socket interface.
Processes or Threads running in user space:
Email, WWW, p2p, Voice, etc.

4.1 Elastic application

Elastic:

- Interactive eg. Telnet, X-windows
- Interactive bulk eg.FTP, HTTP
- Asynchronous eg.Email, Voice-mail

.
examples:

Email

- asynchronous
- message is not real-time
- delivery in several minutes is acceptable

File transfer

- interactive service
- require ”quick” transfer
- ”slow” transfer acceptable

Network file service

- interactive service
- similar to file transfer
- fast response required
- (usually over LAN)

WWW

- interactive
- file access mechanism(!)
- fast response required
- QoS sensitive content on WWW pages

4.2 Inelastic(real-time)applications

Inelastic(real-time)

- Tolerant
  - Adaptive
    - Delay adaptive
    - rate adaptive
    - ——————upper is newer real-time app——————-
    - —————lower is traditional real-time app————-
  - Non-adptive
- In-tolerant
  - Rate Adaptive
  - Non-adaptive

example:
.

Streaming voice and video

- not interactive
- end-to-end delay not important
- end-to-end jitter not important
- data rate and loss very important

Real-time voice and video

- person-to-person
- interactive(Virtual and Augmented Reality)
- Important to control:
  - end-to-end delay
  - end-to-end jitter
  - end-to-end loss
  - end-to-end data rate

4.3 Email

oldest internet application:example of an Elastic Asynchronous Application
different protocols to send and to receive due to user not always connected to server.

PC −− SMTP − > server −− SMTP − > server −− POP − > SMTP

- Originally designed as a simple ASCII text replacement for the office memo.
- now can transport any kind of file using Multipurpose Internet Mail Extensions (MIME)
  - allows files of different types to be sent as ’Attachments’
  - these files are encoded so that they are not corrupted in transit
- has led to an explosion in the bandwidth required by email
- has also led to an explosion in storage requirements.

4.3.1 SMTP

Simple Mail Transfer Protocol.

- only guarantee to transfer 7bits ASCII characters (hence the need to encode other file types)
- Protocol allows for a sender to identify themselves, specify a recipient and transfer the message
- SMTP ensures that the email is not deleted from the sending system until the receiver has saved the message to non-volatile storage.

Protocol Features

- consists of simple commands: MAIL,VRFY, EXPN, RSET, DATA...
- Every command received by the server side must be replied to with a result code.
  - 2XX action taken, Result OK
  - 3XX action pending
  - 4XX Non fatal error, transaction can be tried again
  - 5XX Fatal error, transaction should be aborted

4.3.2 Typical Mail setup

- unusual to allow desktop system to directly send email to any host on the Internet.
- Generally organizations will setup mail gateway which ensure:
  - headers are correct for the organization
  - virus screening and possible other security measures are taken
- same applies to receive mail: there are more than one servers to receive incoming mail and hold it for users.

4.3.3 Protocols for Accessing Mail

- client systems generally pull received mail form server
- use POP(Post Office Protocol) or IMAP
  - both will allow user to read the email received on the server
  - POP expects user to download email to Mail User Agent(MNA), so mail is stored and organized in client systems.
  - IMAP retains the email on the server, only download the head of email. Messages are downloaded as they are read.

  - POP system are vulnerable to loss of the client system and do not allow a coherent view of email from multiple clients.
  - IMAP requires an active net connection while reading messages.
  - some clients are now smart enough to synchronize email between client and server so that you get the best of both worlds.

4.4 WWW

30 years old and one of the most successful applications in the Internet.
We need to distnguish 3 **independent** entities:

- HTTP - the protocol to transfer data
- HTML - the language to describe the data
- URL's - The method of naming and identifying specific data(pages, images, etc.)

4.4.1 HTTP

the protocol allow HyperText Markup language pages to delivered over IP network.

- Client use browser to access the web.
- Services is provided by a Web server.
- Well know port number 80.
- Uses TCP: Potentially multiple connections per page.

Other characters:

- **Length encoding and Hearders.** These are in the beginning of a reply content.
- **Negotiation.** A client can negotiate which media it can accept
- **Conditional Requests.** A client can request a page **only** if it has been modified
- **Max forward.** a server can minimize the number of proxies on the path

Persistent HTTP .

Non-persistent HTTP issues:

- requires 2 RTTs per object

- OS must work and allocate host resources for each TCP connection
- but browsers often open parallel TCP connections to fetch referenced objects.

Persistent HTTP:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server are sent over connection

Persitent without pipelining:

- client issues new request only when previous response has been received.
- one RTT for each referenced object.

Persitent with pipelining:

- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects
- requests contain more than one object
- Objects are sent one after the other

HTTP Other interactions

- POST method
- Chunk response

Maintaining State in HTTP for some services the web server must maintain information about the user's session and it is done with **Cookies**

1. server include in the reply with set-cookie
2. browser sees and stores the cookie for each site
3. when the browser accesses the same site again it just send the cookie
4. THIS FEATURE can be switched off
5. cookies are vital for : e-commerce

4.4.2 Current Trend:HTTP 2.0

- Reduce Delay in Web pages
- data compressions of HTTP headers
- Server Push technologies
- Fixing the head-of-line blocking problem in HTTP1.1
- Making persistent pipelining work in practice
- do not use text based request
- good adaption by several browsers

HTTP2 multiplexing:

- order of request and reply can be different
- allows for prioritization

HTTP2 Server Push:

- if server thinks client need objects, it will send them straight away
- reduce latency

HTTP2 Header compression:

headers in HTTP2 provide information about the request of response. Header take around 800 bytes of bandwidth and sometimes few KB if it carries cookies. Therefore compressing headers can reduce the bandwidth latency.
Header compression is not like request/respond body gzip rather it is like **not sending the same headers again.**

4.4.3 HTML

- A Markup Language provides hinds to the client on how to display the page.
- final decision about how the page looks like are taken by the browser
- Images can be embedded in text.
- Links can be inserted to allow the download of any sort of file type.
- Similar to MIME, in that the browser can invoke the required application to display the file.
- Cause of the bandwidth explosion on the Internet
- New version: HTML5

4.4.4 URLs

Uniform Resource Locator

- originally just web addresses, but is generalized to include any network service.
- eg. http://www.ee.ucl.ac.uk:8080/-tom/index.html
  - Protocol: HTTP (Could also be ftp, ldap)
  - //www.ee.uck.ac.uk : this is the host address
  - :8080 : when the port isn't the normal 80
  - /-tom/index.html : the filename to be retrieved

4.4.5 Web Proxing and Caching

sometimes administrators do not want browsers to access directly to the Internet

1. contact a proxy server
2. proxy server gets the documents from the server
3. file is returned to the proxy
4. the proxy returns the file to the browser

optionally, the server caches the page, so that other users access it faster.

4.4.6 Middleboxes

TCP packets form residential or mobile user often get modified by middleboxes inside ISPs. Several fields may get changed, including sequence numbers. Sometimes, proactive ACKing is used. The box acknowledges packets that did not arrive to speed up the congestion control mechanism

4.4.7 Peer2Peer File Sharing

- represents a paradigm shift on the Internet: the application is simultaneously **client and server**
- user's program serves files to other "clients" and gets files from other "servers". These "clients" and "servers" are bundled in the same applications.
- The p2p programs form an **Overlay network** where each node is always connected to a small set of 'neighbours'

some problems:

- Querying for the desired file. It could centralized in one server or by sending queries to other peer2peer nodes which will transmit it to the necessary one.
- Bootstrapping: how to find neighbours
- Finding good ”neighbours”
- choosing good neighbours to optimize QoS
- Creating incentives for cooperation

5 Multimedia

5.1 RTP

real-time protocol. It specifies a packet stucture for packets carrying audio and video data.

- RTP packets provide:
  - Payload type identification
  - packet sequence
  - numbering
  - timestamping
- RTP runs in the end system
- RTP packets are encapsulated in UDP segments
- Interoperability : if two runs RTP and they can work together.

RTP runs on the top of UDP RTP library provide a transport layer of interface that extend UDP:

- payload type identification
- packet sequence numbering
- time-stamping

RTP Header

- **Payload Type 7 bits** indicates type of encoding current used. If sender changes encoding in middle of conference, it will inform the receiver through this Payload Type field.
- **Sequence Number 16 bits** increments by one for each RTP packet sent, and may be used to detect packet loss and to restore packet sequence

RTCP Real-time control Protocol

- work in conjunction with RTP
- each participant in RTP session periodically send RTCP control packet to all other participants
- each RTCP packets contains sender and / or receiver reports: report statistics useful to application(number of packet sent/number of packets loss/inter-arrival jitter/etc.)
- feedback can be used to control performance : senders may modify its transmission based on the feedback.

5.2 RTSP

Real-Time Streaming Protocol
Defined in RFC 2326 and it is a n out-of-band protocol running on port 544. It can run over TCP or UDP.

- several app consist of several streams which need to be coordinated
- services like playback, fast-forwarding, pausing are very useful.

RTSP Operation .
Web browser $-----$ HTTP GET $->$ Web server
Web browser $<-$ presentations etc. $--$ Web server

media player(client)(show the process between client and server)
$SETUP \rightarrow$
$\leftarrow$
$PLAY \rightarrow$
$\leftarrow$

$\leftarrow mediastream$

$PAUSE \rightarrow$
$\leftarrow$

$TEARDOWN \rightarrow$
$\leftarrow$
media server(server)

TCP video streaming (youtube etc)

- use TCP
- they all do buffering
- 3 strategies:
  - short ON/OFF periods at the application layer
  - Long ON/OFF periods
  - No ON/OFF periods. app downloads everything in one go(need storage in the device and produce useless transmission)

5.3 DASH

dynamic adaptive streaming over HTTP

1. server send to client a manifest : containing all available encodings and speed in XML / URL for each video audio subtitles etc.
2. Client chooses depending on local conditions

5.4 CDNs

content distribution networks

- most content is replicated in CDNs
- Servers all around the world
- two big advantages:
  - save bandwidth
  - better user experience
- two main techniques : DNS / HTTP redirect