Date: ...../...../.....

CSE421 – Computer Networks – SFQ

Lecture 16 – IPv4 Functions

Network layer – Transport segment from sending to receiving host.
Sender encapsulates segments into packets.
Network layer protocols in every host+router. Router examines header fields in all IP packets passing through it. On receiving side, delivers segments to transport layer.

Functions of Network layer – (i) Forwarding – move packets from router's I/P to router's appropriate O/P

(ii) Routing – determine route taken by packets from S to D using routing algorithms. Each router has entries in a routing table which is used to determine the route. (Decision making)

Packet-Switching → Virtual circuits + Datagram Network

* Connection + Connectionless Service.

VC Network ↔ Datagram Network

- similar to transport layer services
- service → host to host
- implementation → network core
- has no choice → network provides one or the other

* Datagram Networks ↑ UDP is similar to this. (connectionless)
  - No call setup at network layer
  - Routers: no state about end-to-end connections
    ↳ no network level connection
  - Packets forwarded using Dest^n IP.
    ↳ packets betⁿ same S-D pair may take different paths.

  Advantage – Faster
  Disadvantage – Reliability ↓ Security ↓

* VC Networks (connected)

Date: ...../...../.....

Signalling Protocol

CP
- used to setup and maintain teardown VC
- initiates call + fixed connection route/path is predetermined & then data is sent
- used in ATM, frame-relay, X.25
- not used in today's internet

* Functions of a router — (I) forwarding — datagram from inc. to out Links
Datagram rate > forwarding rate — Queuing (II) Decentralized Switching — routers do not have connections — func. on their own

* Internet Network layer —

Routing Protocols	IP Protocols	ICMP Protocol
- path selection	- addressing	- error reporting
	- datagram format	- router signalling
	- packet handling	

* IPv4 Datagram Format —
  * minimum size → 20 bytes (no data)
  * maximum size → 65535 bytes

1st row - 32 bits

0 to 4 bits → Version → 0100 (4)
4 to 8 bits → IHL (Internet Header Length) → 20 to 60 bytes — how much of a packet is just header.
8 to 16 bits → TOS → Variable length
(Flag values) [0/1] — (I) Precedence → can assign priority to a specific datagram
(II) Delay
rate of message delivery over a communication
(III) Throughput → 1 (packets will increase)
(IV) Reliability → 1 (req. a line that is reliable)
(V) Reserved
16 to 32 bits → TL (Total length) TL = IHL + Data

1st five rows till Destn. Address → $\frac{32 \times 5}{8}$ = 20 bytes

3rd row - 32 bits

0 to 8 bits → TTL (time to live) — a timer; suppose a packet's destn IP address Ded/corrupted/IP does not exist → it will keep roaming around in the network from R to R so a timer is used. When the time is up, the packet is discarded. Per router TTL value drops by 1.
without the timer it will keep moving from R to R (infinite loop)
↳ bandwidth waste ↑

Date: ...../...../.....

Per hop, TTL value is reduced by 1. Packet remains alive till TTL = 0.
Packet always checks the header of the packet for TTL information.
TTL = 0 → packet dropped.

3rd row

8~16 bits → Protocol (TCP/UDP in transport)
16~32 bits → Header checksum (only checks H as data already checked in transport layer
4th row (32 bits) → Source Address
5th row (" ") → Destn Address

0~32 bits → Options field (Extra 40 bytes)
0~32 bits → Data

* IP Fragmentation and Reassembly —
MTU — Max Transfer Unit
- upper limit of bytes in a link betn R-R.
- different link; different MTUs
Suppose one datagram is very large. → 4500 bytes
MTU of one link → one datagram
" " another " → three " → 1500×3
↳ dividing datagram due to MTU → Fragmentation.
* Reassembled only at final destn
* IP Header used to identify and order the fragments
* Fragmentation is done from router-to-router.

* Math — next page

Date: ...../...../.....

④ Fragmentation Math –

Packet/Datagram/Total length → 2000 bytes

Header → 20 bytes (if not given) ↓

Datagram = Header + Data

Identifier of original IP Datagram → packet's fragmented datagram will also have same identifier → shows fragment is of which main datagram → reassembles

2nd row → 0~16 bits Identification

16~19 bits – Reserved

(Flags) Dont Fragment (DF) → 1 → cannot frag

More Fragment (MF) → 1 → further frag is needed; not last frag.

19~32 bits – Fragment Offset

↳ order of fragments for reassembly → 0 → last frag. arrange the packets.

TCP had seq number for reassembling data segments. Fragmentation uses offset for reassembly.

Why is offset 8 bytes? – slide 14

Total length (TL) = 5140

= 20(H) + 5120(D)

MTU = 1500

TL form ↙ = 20(H) + 1480(D)

why only D and not H? → same H is used; copied from main packet so not included in calculation

Data { Initial Data = 5120
MTU Data Limit = 1480
1st Fragment = 5120 − 1480 = 3640

Size = D + 20 = 1500 { Size of 1st Fragment = 1480 + 20 = 1500
2nd fragment = 3640 − 1480 = 2160
3rd fragment = 2160 − 1480 = 680
4th fragment = 680 + 20 = 700 ↳ No more fragments

0~1479 0/8 = 0
1480~2959 1480/8 = 185
2960~4439 370
4440~5119 555

TL → 16 bits
Fragment Offset → 13 bits
Starting Byte/8

} Divided by 2^3 → 3 shifts

Date: ...../...../.....

Packet size = 4500 bytes
= 20(H) + 4480(D)

MTU = 1500 bytes
= 20(H) + 1480(D)

Start initial byte = 8 bytes
↳ always divisible by 8.

8/8 = 1

1480(D) + 20(H) = 1500
1480(D) + 20(H) = 1500
1480(D) + 20(H) = 1500
40(D) + 20(H) = 60

4480(D)

4480 − 1480 = 3000
3000 − 1480 = 1520
1520 − 1480 = 40

	offset	
0 ~ 1479	0 + 1	If start byte = 24 +3
1480 ~ 2959	185 + 1	24/8 = 3 +3
2960 ~ 4439	370 + 1	+3
4440 ~ 5919	555 + 1	+3

Packet size = 20101234
= 20101194(D) + 40(H)

Header = 40

MTU = 16032
= 15992(D) + 40(H)

(I) No. of packets.
(II) Last packet size
(III) offset value of 100th packet

(I) No. of packets = $\frac{20101194}{15992}$ = 1256.953 = 1257 ↳ remaining fragment

(II) Last packet size = 20101194 − 1256(15992)
= 15242 bytes

Last packet size = 15242 + Header = 15242 + 40
= 15282 bytes

nth packet size (data) except last packet → same size

(III) 0th packet → 0/8 = 0

nth packet's offset = Data × (n−1)

2nd packet ← 1st u → 15992/8 = 1999
100th u → 1999 × 100 = ~~199900~~ 197901
99

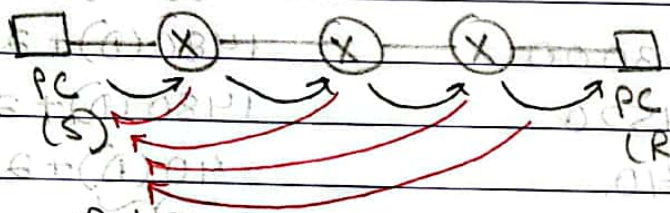
Date: ...../...../.....

ICMP (Internet Control Message Protocol) —

↳ error handling; network congestion; availibility of remote hosts

– carries no app^{n} data but status information of network

Main ICMP — (1) Ping
(1) Traceroute



Test connectivity w/ Receiver PC.

* Ping IP of Receiver → Yes/No Response but no other internal information. No scope to pinpoint where the connection failed.
* Traceroute. IP of Receiver → Returns information per hop. You can pinpoint is where the connection failed + find the error and troubleshoot.

– to detect error.

Ping — test reachability of a host
* sends ICMP echo request packets to the target host & waits for an ICMP response (4)
* in the process it measures the time from transmission to reception (RTT) and records any packet loss.
* shows individual information of each packet. Yes/No depends on the packet response. → byte size + RTT + TTL
* if initial TTL = 120 and TTL = 114 then we have crossed 6 hops. ↳ Remaining
* shows ping statistics → % loss
* shows average RTT
* at least one response (out of 4) → ping successful

Next Page

Date: ...../...../.....

ICMP Messages—

↳ type, code and first 8 bytes of IP datagram causing error

1st row { 0–7 bits type
8–15 bits code
16–31 checksum

2nd row { 0–31 message

Type	Code	Message	
0	0	echo reply (ping)	R → S
8	0	" req (")	S → R
11	0	TTL expired	
3	1	Dest^n host unreachable	
3	3	" port "	

(some errors =)

Traceroute — tool used to trace path from S → D host.

* shows IP address + domain name of each router/hop — returned to the client.
* if there is an error/no response —
  * * * Request timed out.
  packet could not finish sending/receiving
* Source sends series of UDP segments to dest^n.
* nth datagram arrives to nth router —
  (I) router discards datagram
  (II) sends ICMP message to source
    ↳ includes name of router + IP address
    ↳ on arrival, source calculates RTT
    ↳ three times
* Stop → UDP segment eventually reaches dest^n host;
  Dest^n returns ICMP port unreachable packet
    ↳ Source gets this message + stops.

—x—