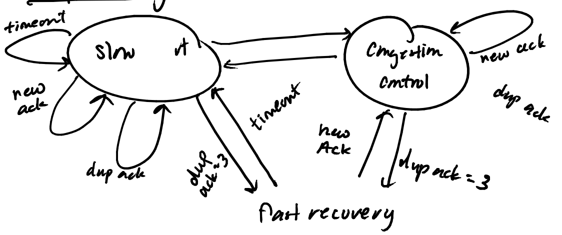
TCP Congestion Control



Detecting Cong.

Packet delays

Packet loss (fail-safe)

Duplicate loss : isolated l

Timeout : much m (adjust rate)

Issues w/ TCP

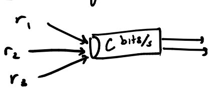
$\text{tput} : \sqrt{\frac{3}{2}} \frac{1}{RTT\sqrt{p}}$

Implications?

- TCP unfair if heterogeneous RTTs.
- High speed TCP numbers unreasonable
  - └ adapt by Router assisted approaches? or increase CWND fn f past threshold speed

Router Assisted Ctrl. Ensure flows into flows, "share"; routers classify packets FIFO queue in router

Max-Min fairness

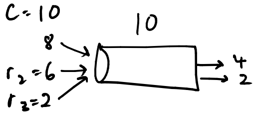


given a set of bandwidth demands r_i and total bandw. C, max-min allocations are:

$a_i = \min(f, r_i)$ allocation

where f = unique val such that $\text{sum}(a_i) = C$

: C = 10



① $\frac{C}{3} = 3.33$, but r_3 needs only 2

· can service r_3, remove from accounting $C = C - r = 8$; N = 2

② $\frac{C}{2} = 4$ can't service all, so give fair s

FQ vs. FIFO

(+) Isolation : cheating flows don't benefit

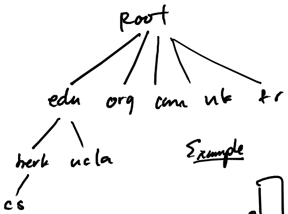
(+) bandwidth share doesn't depend on RTT

(+) flows pick rate adjustment scheme

(-) more complex than FIFO : per flow queue / state additional per-packet bookkeeping

DNS Goals (Application Layer)

· fast lookups, hierarchical, unique

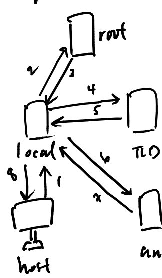


Root

TLD

Authoritative

Example



· query from host to local recursive, the rest iterative

records

- servers store RRs.
  - name, value, type, TTL)

Type (→ Address)

: hostname

value : IP address

type: NS

name: domain

val : name of dns server for domain

DNS Caching

- reduces load at all levels, reduces delay for client
- effective b/c Top level servers rarely change
- popular sites visited often → local DNS server often has info. cached

HTTP

(web's app layer protocol, uses TCP as underlying transport protocol

Non persistent

each request/response pair over a separate TCP connection

brand new connection for each object

↳ each suffers a delivery delay of 2 RTTs: 1 to establish connection, 1 to retrieve object

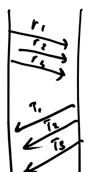
Concurrent Requests & responses

use multiple connections in parallel network ⚠

Persistent Connections

Maintain TCP connection across multiple requests.

Pipelined Requests / responses



- batch requests & responses to reduce # of packets.
- multiple requests in 1 TCP segment

Evaluate! Getting n small objects? ← time dominated by latency

1. one at a time ⟶ ~ 2nRTT
2. M concurrent ⟶ ~ 2⌈n/m⌉RTT
3. Persistent ⟶ ~ (n+1)RTT
4. pipelined ⟶ ~ 2RTT
5. pipelined/persistent ⟶ ~ 2RTT the first time, RTT after

Getting N Large Objects

← the only thing that helps is getting more Bandwidth

1. 1 at a time : ~ n F/B
2. M concurrent : ⌈n/m⌉ F/B
3. Pipelined and/or pers : ~ n × F/B

Ethernet (old) (link layer)

Random access protocols

CSMA

- carrier sense : check if already sending. If so, wait
- collision detection : listen while transmitting
  - └ if collision, abort, send jam signal
- random access : binary exponential backoff wait random time before trying again

Ethernet frames

how does link layer determine where frames end/begin?

└ count bytes! → need frames.

Frames : Sentinel bit framing :

01111110	╳	01111111

bit stuffing !!!

- Sender inserts a 0 after 5 1's
- receiver always removes 0 after 5 1's

HTTP caching

Where?

1. Clients: forward proxies ⇓ reduce traffic, decrease latency
2. Server ⇓ decrease server load network load

HTTP Ex: A retrieving files F & G from site B. RTT b/t A & B is 10msec = (.01sec)

Bandwidth b/t A & B is 10Mbps.

- Time to retreive both files?

① Sequential, non persistent?

RTT (2 SYN/ACK + 2DATA/ACK)

+ 2 (Transmission time)

$= 4 \times .01 + 2\left(\frac{1mb}{1mbps}\right) =$

$.04 + .2 = .24$

② Concurrent, Nonpersistent

2RTT (1 SYN/ACK + 1 DATA/ACK)

$+ \frac{1 \text{ Transmission time}}{2}$ ← b/c dedicated link!

Switched Ethernet : concurrent communication

Mac Addresses (link Layer)

- associated w/ network adapter
- Flat name space of 48 bits in HEX.
- social security #!
- portable, stays the same ....
- ✗ used to get packet b/t interfaces on the same network

vs.... IP Addresses

- configured, learned dynamically, postal addy
- ✗ used to get a packet to dest. subnet.

"Routing" w/ broadcast Ethernet

- doesn't use LS/DV b/c not scalable (MAC addresses cannot aggregate like IP); plug & play!

How it works : Sender transmits frame to broadcast link, frame contains MAC address.

└ if dest matches receiver's MAC addy, or broadcast (FF:FF:FF:FF:FF:FF)

→ PASS FRAME TO NETWORK LAYER!

SPANNING TREE APPROACH

✗ Messages: (Y, d, X)

propose Y as root, distance d, and from node X

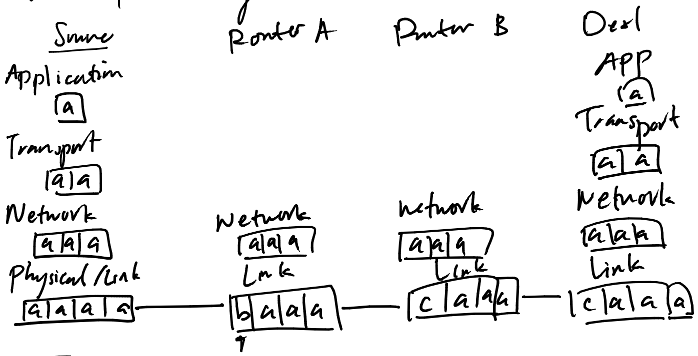
· switches elect node w/ smallest identifier as root

How? Switched Ether

① build spanning tree for loop free flooding

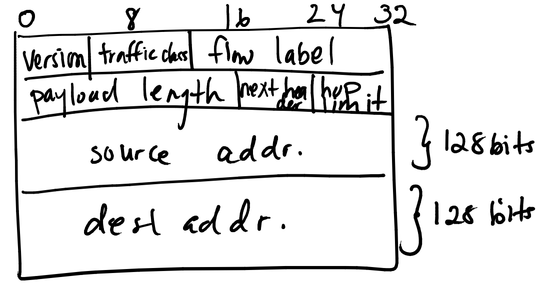
② "self learning" switches .... optimizes ...... switches can reach dest w/o flooding

View of all layers



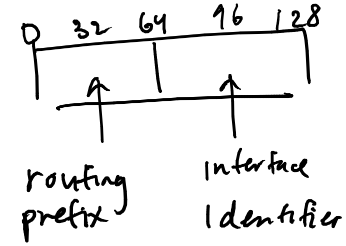
IPV6
why? exhausted space.
Differences?
- addresses 128 bits, not 32 bits
- headers different
- address management

Same?
- routing protocols
- longest prefix / shortest path



hop limit: TTL
flow label: new, help network identify flow

Addresses



64 bits dedicated to addressable interface

Naming

- Application layer: URLs, domain names
- Network: IP addresses: host's network location
- Link layer: MAC addresses: host identifier

all 3 for E2E

Layer	ex	Structure	Config	resolution
APP	berk.com	organizational	manual	↕ DNS
Network	123.45.6.78	topological	DHCP	↕ ARP
Link	45-CC-42...	flat	hard coded	

Datacenters

- scalability baseline req.
- more emphasis on performance
- less on heterogeneity & interoperability

Solutions
- extend DV/LS?
  - (-) scales poorly.... N destinations, O(N) routing entries/msgs

ARP & DHCP Link Layer Discovery Protocols used by Network Layer

2 functions:
1. Discovery of local end hosts, for communication b/t hosts on same LAN
2. Bootstrap communication w/ remote hosts
   - what's my IP address?
   - who/where is my DNS server?
   - who/where is my first hop router?

Ideas:
broadcast
soft state
caching

DHCP: used by host to scover:
- netmask, IP ad P addresses)
for DNS server(s) 1 days for f.h
router

Discovery Mechanisms
ARP/DHCP: broadcast
- flooding doesn't scale!
- zero configuration
- no centralized point of failure

DNS:
- scalable (no floods)
- needs config: (local, root servers, etc)

Arp table: IP address → MAC addr.
- maintained by every host
- consult table when sending packet, broadcast IP address, receiver responds w/ MAC address to IP addy.

To reach destinations

1) need own IP: DHCP
need local DNS: DHCP

Send Packet
- Same subnet: ARP (use MAC addy of dest)
- Other subnet: ARP+DHCP (use MAC address of first hop router)

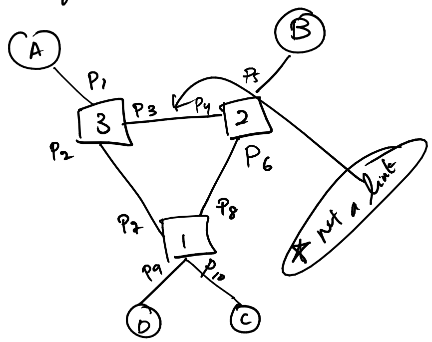
} use netmask to tell if host on same or diff subnet, or DHCP

Recap

MAC?
own: hard coded
others: ARP (given IP addy)

IP.
own: DHCP
others: DNS (given domain name)

Spanning Tree, Self Learning



Switch A wants to talk to B (cache empty) ..... State of forwarding table after ARP request complete?

S1

dest	port
A	p7

S2

d	p
A	P6

S3

d	p
A	p1

After B responds?

S1

A	p7
B	p8

S2

A	P6
B	p5

S3

A	P1
B	p2

C sends to A

S1

B	p8
C	p9

S2

B	p5
C	p6

S3

B	p2
C	p2

(all newest entries move up, replaces oldest b/c max 2 entries in this problem)

HTTP continued

ⓒ sequential, 1 persistent connection

3RTT (1 syn/ACK + 2 Data/Ack)

+ 2 transmission time = .03 + .02 = .23 sec

ⓓ pipelined, persistent

2RTT(1 syn/ACK + 1 data/ack) + 2 transmission time

= .02 + .2 = .22

E2E

① self discovery (DHCP)

IP	UDP	DHCP Disc.

IP: source: 0.0.0.0

Dest: 255.255.255.255 (broadcast!)

add a link layer frame

LL	IP	UDP	DHCP disc

Mac addr: FF:FF....

② Machine running DHCP server prepares offer w/ IP, DNS IP, Default Gateway IP, subnet mask

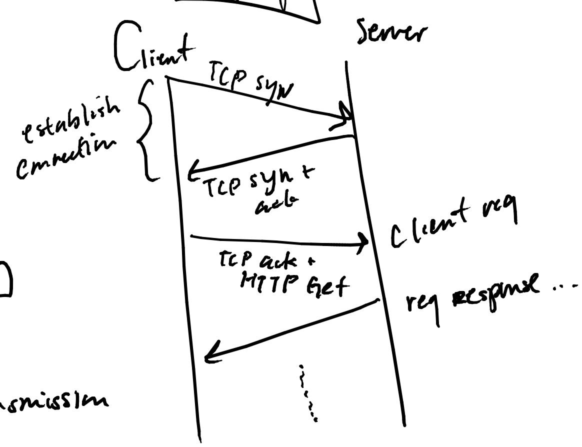
...... Client accepts offer by broadcasting 'request' msg, server ACKs.

③ connect to google?

- need (MAC) address of router

└ broadcast ARP req, Gateway Router responds

LL	Arp



④ get dest ip address! (DNS)

DNS response w/ Google's IP address. (unless cache)

⑤ use HTTP to communicate

TCP is transport layer protocol used

LL	IP	TCP	HTTP

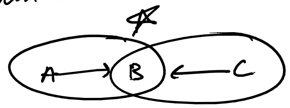
ethernet	IP	TCP	HTTP

ethernet ↑ Router's mac

IP → source: me, dst: google IP

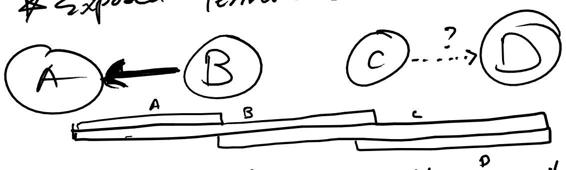
Wireless

* Hidden terminals



- cannot use carrier sense!!
  (needs to sense at receiver

* Exposed Terminals



- if B talks to A, C doesn't transmit to D b/c of carrier sense..... it would have worked!

MACA : multiple access w/ collision avoidance

- if no CTS, assume collision
- if you hear a CTS, wait for ACK.
- if hear RTS, no CTS, send

Gao Rexford Rules for BGP: