Packet vs. Circuit Switching

on demand packet processing — resources reserved per active "connection"

Virtual circuits emulate "circuit" w/ packets

Time + frequency division multiplexing

Circuit switching enables predictable performance and simpler/fast switching, but is complex to setup, inefficient when bursty, prone to error

Packet switching = efficient use of resources, simpler implementation, robust "route around trouble" but unpredictable, need congestion control

Types of Delay

Transmission – time to push all packet bits into a link, packet size/transmission rate

Propagation – time to move from one -> other, link length/propagation speed of link

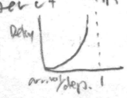
Queuing – sit in buffer until processed, depends on arrival rate @ queue, burstiness, transmission rate of outgoing link

$L = AW$ — L - avg # packets waiting in queue "length", A - average arrival rate, W - average wait time

Processing – how long to process a packet

Link bandwidth – # bits sent/received per time

Bandwidth-Delay Product – bandwidth x prop. delay, number of "in flight" bits at any time

Queuing  Approaches ∞ if arrivals too fast, finite buffer -> loss

Given bottlenecks, average throughput = $\min\{R, R'\} = R$

Internet Hierarchy and Layering

Application -> OSI has Presentation -> Session

Transport – reliable (or unreliable) transport

Network – Best-effort global packet delivery

Data Link – Best-Effort local packet delivery

Physical – Physical transfer of bits

Layers interact with surrounding, two layers only interact through shared layer

Layering helps w/ complexity and flexibility but makes things slow, encapsulation sometimes inconvenient

End-to-end Principle – Design principle which states that application requirements should be implemented in end-systems only (reliability, security)

- may be more complex in network, delay

"dumb" network, "smart" end systems

Don't keep state in routers, "fate sharing"

Network Layer

Forwarding in data plane, where to send packet

Routing system-wide, how to setup forwarding table

Global state refers to collection of forwarding tables in routers. vs. local routing state

"valid" state if forwarding decisions get packet to dest., must be no dead-ends and no loops

Pick a point, remove unused links -> minimum-spanning tree

Routing Algorithms

Least-cost paths yield no loops, destination-based and produce the minimum spanning tree

Link-state (OSPF uses!) – Dijkstra's algorithm

All routers flood link state to all other routers

Convergence delay depends on time to detect failure, flood link-state info and recompute forw. tables

May see lost packets, looping, out-of-order

Distance Vector (RIP)

Distributed algorithm, passing local link costs and least-cost path to neighbors, and beyond

Bellman-Ford Algorithm

$$d_x(Z) = \min_n \{cost(x,n) + d_n(Z)\}$$

for all possible neighbors n

Pick as next hop for dest z the neighbor that results in the least cost path to z

Convergence when no more improvement possible

O(NE) messages, O(N²) computation, O(network diam) convergence, O(N) entries

Count-to-infinity

Z through Y, Y through X, Y loses connection to X and routes through Z...

Poisoned-reverse: tell a node you route through that your distance to destination is ∞

O(n) update time, O(network diam) convergence, O(n) entries

Interdomain Routing

AS ("domains") – autonomous system is a network under a single administrative control (unique ID)

Scaling

Naive solution gives entry for each dest, not scalable!

Hierarchical address aggregation simplifies table size, must be planned ahead of time

IPV4 Addresses

Unique 32-bit number associated w/ host, dotted quad

Divided into prefix (network component) and suffix (host component)

Classful addressing -> 3 main classes

Ⓐ 7/24, Ⓑ 14/16, Ⓒ 21/8 net/host bits

CIDR (classless interdomain routing)

Flexible boundary by slash notation (applies bit mask when /n bits and all zeros)

Hierarchical address allocation only helps if allocation matches topological hierarchy

Multi-homing makes it less effective/more difficult to aggregate addresses

BGP (Border Gate Protocol)

ASes want policy, autonomy and privacy

Customer, provider and peer relationships help define routing relationships

Based on distance vector with four changes

① Not always picking shortest path routes, policy!

② Path-vector routing, loop avoidance is easy. Flexible policies based on knowledge of entire path

③ Selective route advertisement (not all connected graph reachable)

④ BGP may aggregate routes by prefixes

Route attributes

① ASPATH – vector lists all ASes traversed

② Local PREF – choose between different ASs paths

③ MED – "multi-exit discriminator", when ASes connected by 2 or more links

④ IGP cost – select closest path to next AS

eBGP > iBGP

Router ID

} in terms of precedence order

IP Layer / Data Plane

IP packet consists of payload + header, we only care about the contents of header

4-bit version – specify version number

4-bit header length – specify header length in bytes

8-bit type of service – allow packets to be treated differently based on needs

16-bit total length – total packet length

Fragmentation (if size > MTU)

Reassembly typically done at end system dest.

16-bit identification – to match packets

3 flags – Reserved (ignore), DF (don't fragment), MF – indication of more fragments

13-bit fragment offset – portion of original payload in 8-byte units, allows fragmentation of existing fragments

8-bit time-to-live (TTL) – avoid loops forever, decremented at each hop, dropped if -> zero

8-bit protocol – tells higher level protocol and used for demuxing at receiving host

16-bit checksum – protect against corruption, recomputed with TTL at each hop

32-bit Source IP Address

32-bit Destination IP Address, and additional options section (optional)

Possible Security Concerns

Can spoof addresses, victim receives traffic or is also wrongly blamed

Launch a denial-of-service attack

May use source route to choose path, firewalls may just drop packets with options

Use TOS to have higher priority, not used today

Fragmentation can evade network monitoring, split attack into multiple fragments, or send invalid overlapping fragments

TTL allows discovery of network topology (traceroute) (initial TTL may be distinct to OS)

(EE) IPv6 eliminates fragmentation and checksum, more address bits, eliminate header length, added flow label

BGP issues

reachability (not all are reachable by policy)

security

AS can blackhole by claiming to serve a prefix, make them "prove" they have a path

AS can forward down fictitious path

Convergence (only if Gao-Rexford)

Performance

Domains usually use hot potato routing

Policy paths usually will not be the shortest

AS path length may be misleading

BGP outages are a problem, most prefixes are stable

Hierarchy – BGP is bloated + underspecified

Route updates include both announcements and withdrawals <IP prefix: route attributes>

eBGP – BGP sessions between border routers in different ASes, share routes

iBGP – session between border router and internal routers in same AS

IGP – OSPF/RIP for AS

Gao-Rexford Rules

Destination advertised by:

customer -> everyone (peer, provider, customer)

peer/provider -> only customers

Routes are "valley free"

Gao-Rexford will guarantee convergence, other policies may not converge!

IP Routers

Router defined by N - # external "ports",
R "line rate" of each port, capacity = N × R
Consists of route/control "control plane" (pushes forwarding tables to line cards)
Linecards (input) - process packets on way in "data plane"
Interconnect (switching) fabric - transfer packets from input to output ports
Linecards (output) - Process packets before they leave

Input Linecards
- Receive incoming packets, update IP header (TTL, checksum, frag)
- Lookup output port for dest, queue packet at switch fabric
- Speed is a concern! Use specialized hardware
- Aggregation of addresses improves scalability (otherwise 4 bill. entries!)
  - Practice using longest prefix matching
  - Can optimize using prefix tree, can further optimize using heuristics, keep track of popular dest.

Output Linecards
- Packet classification: map each packet to a "flow"
- Buffer management: decide when + which packet to drop
- Scheduler: decide when and which packet to transmit
- Can implement policy like deny all email traffic (access control), policy (route IP telephony -- from X to Y via), QoS (ensure no more than 50 mbps from X)
- FIFO Router
  - No classification, drop tail buffer management: drop incoming packets when buffer full
  - FIFO packet scheduling as well
- Packet Classification
  - Classify IP packet based on number fields in pack header
    - Source/dest IP address, source/dest TCP port number
    - Type of service, type of protocol
- Scheduler
  - One queue per "flow"
  - Scheduler decides when and from which queue to send packet
  - Must be fast, policy-dependent (priority scheduler, fair round robin scheduler)
  - Head of line blocking can be fixed w/ virtual output queues
    - Schedule by where it is going, more efficient but complicated

Transport Layer Protocols

Demux packets between applications
Serves as additional service upon IP → app
Important because IP only performs best-effort (don't want to deal w/ in appl)
Provides services to end-to-end systems, like reliable, in-order data delivery and well-paced data delivery

Socket: software abstraction used by application interact w/ transport layer
UDP → SOCK_DGRAM, TCP → SOCK_STREAM
Port: transport layer identifier, packet has source/dest port # in tp header
- OS stores mapping between sockets and ports
- IP header has source/dest address, TP Layer header has port number

UDP maps local dest port and address → socket
TCP maps address pair + port pair to socket
Well known ports (0-1023) ssh:22, http: 80, (1024-65535) ephemeral

UDP

header: source port, dest port, checksum, length

Reliable Transport
- Checksums: corruption, ACK: received packet, NACK: did not receive packet, Seq #: ID packets, retransmissions: resend packets, timeouts: when to resend packet, forward error correction, network encoding

Mechanisms of Reliable Transport
- Sliding Window (efficiency) Throughput = min[n·DATA/RTT, Link Bandwidth]
  - window is the set of adjacent sequence numbers, size n
  - send up to n packets at once, window slides w/ successful ACK
  - Packets in flight, a way to view this
- Cumulative ACKs or selective ACKs w/ sliding windows
  - Go-back-N
    - Only take packets in order, sender sets timer for first outstanding ACK
  - Selective Repeat sends back specific ACKs for each individual packet (retransmit on timeout), efficient retransmit but need timer for each packet

TCP, delivers a reliable, in-order bytestream (abstraction)
- Header includes source/dest port, seq number, acknowledgment, header length, flags, advertised window, checksum, urgent point, options
- Sequence Numbers
  - TCP packet no bigger than MSS, IP not bigger than MTU
  - MSS = MTU - (IP header) - (TCP header)
  - Sequence number = 1st byte in segm. + initial sequence number
  - If all before X received, ACK X+8, else if highest-in-order is Y (Y+1 < X), ACK Y+1 sent back
- Receiver do not drop out-of-sequence packets, fast retransmit
- TCP uses timer on last sent and unacknowledged packet
  - RTT estimation → exponential averaging: EstRTT = α EstRTT + (1-α) SampleRTT
  - Karn/Partridge: α = 0.875, measure only original sampleRTT
  - Each time RTO expires, RTO = 2*RTO → 2×EstRTT if successful measurement
  - Jacobson/Karels: Deviation = |SampleRTT - EstRTT|, RTO = EstRTT + 4 × EstDeviation (exp. avg of dev)
- Connection Establishment
  - Host sends SYN w/ ISN K, recv. sends SYNACK w/ ISN K+1, host returns an ACK with possible data (set SYN + ACK w/ flag bits), K+2 ISN
  - FIN to close connection → ACK, FIN back and ACK
  - Can send RST to terminate immediately, B just accepts and finishes
- Advertised Window and Flow Control
  - Receiver indicates value of W in ACKs, sender limits packets in flight
  - Receiver advertises when window ends (sender will not exceed this)