Common Protocols

An Engineering Approach to Computer Networking
The grand finale

- Previous chapters presented principles, but not protocol details
  - these change with time
  - real protocols draw many things together
- Overview of real protocols
  - standards documents are the final resort
- Three sets of protocols
  - telephone
  - Internet
  - ATM
## Telephone network protocols

<table>
<thead>
<tr>
<th></th>
<th>Data Plane</th>
<th>Control Plane (SS7)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>App</strong></td>
<td>Voice/Fax</td>
<td>ASE/ISDN-UP TCAP</td>
</tr>
<tr>
<td><strong>Session</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Transport</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td></td>
<td>SCCP/MTP-3</td>
</tr>
<tr>
<td><strong>Datalink</strong></td>
<td>Sonet/PDH</td>
<td>MTP-2</td>
</tr>
<tr>
<td><strong>Physical</strong></td>
<td>Many</td>
<td>MTP-1</td>
</tr>
</tbody>
</table>
Traditional digital transmission

- Long distance trunks carry multiplexed calls
- Standard multiplexing levels
- Digital transmission hierarchy

<table>
<thead>
<tr>
<th>Multiplexing level</th>
<th>Name</th>
<th># calls</th>
<th>Rate (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DS1</td>
<td>24</td>
<td>1.544</td>
</tr>
<tr>
<td>2</td>
<td>DS2</td>
<td>96</td>
<td>6.312</td>
</tr>
<tr>
<td>3</td>
<td>DS3</td>
<td>672</td>
<td>44.736</td>
</tr>
<tr>
<td>4</td>
<td>DS4</td>
<td>4032</td>
<td>274.176</td>
</tr>
</tbody>
</table>
Plesiochronous hierarchy

- Plesiochronous = nearly synchronous
- Tight control on deviation from synchrony
- What if stream runs a little faster or slower?
- Need justification
Justification

- Output runs a bit faster always
- Overhead identifies bits from a particular stream
- If a stream runs faster, use overhead to identify it
- Overhead used everywhere except at first level (DS1)
Problems with plesiochrony

- Incompatible hierarchies around the world
- Data is spread out! Hard to extract a single call
- Cannot switch bundles of calls
Synchronous Digital Hierarchy

- All levels are synchronous
- Justification uses pointers

<table>
<thead>
<tr>
<th>Data Rate (Mbps)</th>
<th>US Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 51.84</td>
<td>OC-1</td>
</tr>
<tr>
<td>2 155.52</td>
<td>OC-3</td>
</tr>
<tr>
<td>3 466.56</td>
<td>OC-9</td>
</tr>
<tr>
<td>4 622.08</td>
<td>OC-12</td>
</tr>
<tr>
<td>5 933.12</td>
<td>OC-18</td>
</tr>
<tr>
<td>6 1244.16</td>
<td>OC-24</td>
</tr>
<tr>
<td>8 1866.24</td>
<td>OC-36</td>
</tr>
<tr>
<td>9 2488.32</td>
<td>OC-48</td>
</tr>
<tr>
<td>9953.28</td>
<td>OC-192</td>
</tr>
</tbody>
</table>
SDH (SONET) frame

* ID = IDENTIFIES THE OC-1 NUMBER (1 . . N) IN AN OC-N FRAME

** MULTI = INDICATES THAT PAYLOAD SPANS MULTIPLE PAYLOAD ENVELOPES
SDH

- 9 rows, 90 columns
- Each payload container (SPE) served in 125 microseconds
- One byte = 1 call
- All overhead is in the headers
- Pointers for justification
  - if sending too fast, use a byte in the overhead, increasing sending rate
  - if sending too slow, skip a byte and move the pointer
  - can always locate a payload envelope, and thus a call within it => cheaper add drop mux
SDH justification
## Signaling System 7 (SS7)

<table>
<thead>
<tr>
<th>OSI layer name</th>
<th>SS7 layer name</th>
<th>Functionality</th>
<th>Internet example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Application Service Element</td>
<td>Application</td>
<td>FTP</td>
</tr>
<tr>
<td></td>
<td>Transaction Capabilities Application part</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RPC</td>
<td></td>
<td>RPC</td>
</tr>
<tr>
<td>Transport</td>
<td>Signaling Connection Control Part</td>
<td>Connections, sequence numbers, segmentation and reassembly, flow control</td>
<td>TCP</td>
</tr>
<tr>
<td>Network</td>
<td>Message Transfer Part 3 (MTP-3)</td>
<td>Routing</td>
<td>IP</td>
</tr>
<tr>
<td>Datalink</td>
<td>MTP-2</td>
<td>Framing, link-level error detection and retransmission</td>
<td>Ethernet</td>
</tr>
<tr>
<td>Physical</td>
<td>MTP-1</td>
<td>Physical bit transfer</td>
<td>Ethernet</td>
</tr>
</tbody>
</table>
SS7 example

- Call forwarding

- To register
  - call special number
  - connects to ASE
  - authenticates user, stores forwarding number in database

- On call arrival
  - call setup protocol checks database for forwarding number
  - if number present, reroutes call

- SS7 provides all the services necessary for communication and coordination between registry ASE, database, and call setup entity
MTP Header
# Internet stack

<table>
<thead>
<tr>
<th></th>
<th>Data Plane</th>
<th>Control Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>App</strong></td>
<td>HTTP</td>
<td>RSVP/OSPF</td>
</tr>
<tr>
<td><strong>Session</strong></td>
<td>Sockets/Streams</td>
<td></td>
</tr>
<tr>
<td><strong>Transport</strong></td>
<td>TCP/UDP</td>
<td></td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>IP</td>
<td>IP/ICMP</td>
</tr>
<tr>
<td><strong>Datalink</strong></td>
<td>Many</td>
<td>Many</td>
</tr>
<tr>
<td><strong>Physical</strong></td>
<td>Many</td>
<td>Many</td>
</tr>
</tbody>
</table>
IP

- Unreliable
- Best effort
- End-to-end
- IP on everything- interconnect the world
IP
Fragmentation

- IP can fragment, reassemble at receiver
- Fragment offset field
- More fragments flag and Don’t fragment flag
- Reassembly lockup
  - decrement timer and drop when it reaches 0
- Fragmentation is harmful
  - extra work
  - lockup
  - error multiplication
- Path MTU discovery
  - send large pkt with Don’t fragment set
  - if error, try smaller
IP fields

- **TTL**
  - decremented on each hop
  - decremented every 500 ms at endpt
  - terminates routing loops

- **Traceroute**
  - if router decrements to 0, send ICMP error packet
  - source sends packets with increasing TTL and waits for errors

- **Options**
  - record route
  - timestamp
  - loose source routing
ICMP

- Destination unreachable
- Source quench
- Redirect
- Router advertisement
- Time exceeded (TTL)
- Fragmentation needed, but Dont frag flag set
TCP

- Multiplexed
- Duplex
- Connection-oriented
- Reliable
- Flow-controlled
- Byte-stream
Fields

- Port numbers
- Sequence and ack number
- Header length
- Window size
  - 16 bits => 64 Kbytes (more with scaling)
  - receiver controls the window size
  - if zero, need sender persistence
  - silly window syndrome
- Checksum
- Urgent pointer
- Options
  - max segment size
HTTP

- Request response
- Protocol is simple, browser is complex
- Address space encapsulation
- Request types
  - GET
  - HEAD
  - POST
- Response
  - status
  - headers
  - body
**ATM stack**

<table>
<thead>
<tr>
<th>Application</th>
<th>Data Plane</th>
<th>Control Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>UNI/PNNI</td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td>Q.2931</td>
</tr>
<tr>
<td>Session</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td></td>
<td>SSCOP</td>
</tr>
<tr>
<td>Network AAL1-5</td>
<td></td>
<td>S-AAL (AAL5)</td>
</tr>
<tr>
<td>Data Link ATM</td>
<td></td>
<td>ATM</td>
</tr>
<tr>
<td>Physical Many</td>
<td></td>
<td>Many</td>
</tr>
</tbody>
</table>
ATM

- Connection-oriented
- In-sequence
- Unreliable
- Quality of service assured
Virtual paths

- High order bits of VCI
- All VCIs in a VP share path and resource reservation
- Saves table space in switches
  - faster lookup
- Avoids signaling
- May waste resources
- Dynamic renegotiation of VP capacity may help
- Set of virtual paths defines a virtual private network
AAL

- Was supposed to provide “rest of stack”
- Scaled back
- 4 versions: 1, 2, 3/4, 5
- Only 1, 3/4 and 5 important in practice
AAL 1

- For synchronous apps
  - provides timestamps and clocking
  - sequencing
  - always CBR
  - FEC in data bytes
**AAL 3/4**

- For data traffic (from a telco perspective!)
- First create an encapsulated protocol data unit EPDU
  - (common part convergence sublayer-protocol data unit CPCS-PDU)
- Then fragment it and add ATM headers
AAL 3/4

- Error detection, segmentation, reassembly
- Header and trailer per EPDU \textit{and} per-cell header!
AAL 5

- Violates layering, but efficient
- Bit in header marks end of frame
**AAL5 frame format**

<table>
<thead>
<tr>
<th></th>
<th>DATA</th>
<th>PAD</th>
<th>UU</th>
<th>CPI</th>
<th>LENGTH</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 64 KBYTES</td>
<td></td>
<td></td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>0–47 BYTES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

AAL5 (ATM Adaptation Layer 5) is a protocol for carrying ATM cell streams over point-to-point physical layers. The frame format includes fields such as DATA, PAD, UU, CPI, LENGTH, and CRC.
SSCOP

- Reliable transport for signaling messages
- Functionality similar to TCP
  - error control (described below)
  - flow control (static window)
- Four packet types
  - sequenced data / poll / stat / ustat
- No acks!
- Sender polls, receiver sends status
  - includes cumulative ack and window size
- If out of order, sends unsolicited status (ustat)
- Key variable is poll interval
IP-over-ATM

- Key idea: treat ATM as a link-level technology
  - ignore routing and QoS aspects
- Key problems
  - ATM is connection-oriented and IP is not
  - different addressing schemes
  - ATM LAN is point-to-point while IP assumes broadcast
- Basic technologies
  - IP encapsulation in ATM
  - Resolving IP addresses to ATM addresses
  - Creating an ATM-based IP subnet
  - Mapping multicast groups to ATM
IP encapsulation in ATM

- Put data portion of IP packets in AAL5 frame
  - works only if endpoints understand AAL5
- Instead, place entire IP packet with AAL5 frame
- General solution allows multiprotocol encapsulation
Resolving IP addresses to ATM addresses

- Need something like ARP, but can’t use broadcast
- Designate one of the ATM hosts as an ARP server

- Inverse ARP automatically creates database
Creating an ATM-based IP subnet

- IP assumes free availability of bandwidth within a subnet
- If all hosts on ATM are on same IP subnet, broadcast reaches all => congestion
- Partition into *logical IP subnets*
  - at the cost of longer paths between ATM-attached hosts
Next-hop routing

- Avoids long paths
- Next-hop server stores IP-to-ATM translations independent of subnet boundaries
  - like DNS
Resolving multicast addresses

- ARP server cannot resolve multicast addresses (why?)
- Actively maintain set of endpoints that correspond to a particular Class D address
- Multicast Address Resolution Server provides and updates this translation
LAN emulation

- If destination is on same LAN, can use ATM underneath datalink layer
- Need to translate from MAC address to ATM address
- Also need to emulate broadcast for Ethernet/FDDI
Cells in Frame (CIF)

- Solutions so far require expensive ATM host-adapter card
- Can we reuse Ethernet card?
- Encapsulate AAL5 frame in Ethernet header on point-to-point Ethernet link
- CIF-Attachment Device at other end decapsulates and injects the frame into an ATM network
- Software on end-system thinks that it has a local host adapter
- *Shim* between ATM stack and Ethernet driver inserts CIF header with VCI and ATM cell header
  - may need to fragment AAL5 frame
  - can also forward partial frames
- Cheaper
  - also gives endpoints QoS guarantees, unlike LANE
Holding time problem

- After resolution, open an ATM connection, and send IP packet
- When to close it?
- Locality
  - more packets likely
  - hold the connection for a while to avoid next call setup
  - but pay per-second holding time cost
- Optimal solution depends on pricing policy and packet arrival characteristics
- Measurement-based heuristic works nearly optimally
  - create the inter-arrival time histogram
  - expect future arrivals to conform to measured distribution
  - close connection if expected cost exceeds expected benefit