Internet Programming & Protocols Lecture 22

Slow links and Compression

Asymmetric networks

Concept Collection

- ACK/NAK cumulative ACK
- ACK clocking
- AIMD
- Auto-tuning
- Bandwidth-delay product
- Best effort
- Bit error rate
- Checksums
- Client/server/concurrent/iterative
- Compressed ACK
- Congestion control/avoid
- Conservation of packets
- CIDR
- CSMA/CD
- cwnd/sstrhesh
- Datagram vs reliable stream
- Delay-based congestion
- Discrete event simulation
- Dup threshold
- ECN
- Exponential backoff
- Flow control
- Forward ACK
- Fragmentation
- Inverse sqrt p
- Layers/encapsulation
- Maximum segment lifetime(MSL)
- MTU MSS/MTU discovery
- Network mask
- Packet switching vs circuit-based
- Partial ACK
- promiscuous
- Routing
- RTT and RTT estimation
- Selective ACK (SACK)
- Self-clocking
- Sliding window
- Slow-start
- Subnets/supernets
- Switch vs hub
- TTL

Our tool set

- ping/traceroute
- ifconfig/netstat
- strace
- lsof
- dig
- ethereal tcpdump/tcptrace/xplot
- ttcp/iperf/netperf
- ns

Plan of attack

- Network overview ✓
- BSD sockets and UDP ✓
- TCP ✓
  - Socket programming
  - Reliable streams
  - Header and states
  - Flow control and bandwidth-delay
  - Measuring performance
  - Historical evolution (Tahoe …SACK)
  - Congestion control
- Network simulation (ns) ✓
- TCP accelerants ✓
- TCP over wireless, satellite, …
- TCP implementations

LECTURES

14	Models and measurement
15	emulation and simulation
16	ns
17	S-TCP, HSTCP BI-TCP
18	Bandwidth estimation
19	Vegas, fast, westwood
20	AQM, RED, ECN. XCP
21	Parallel and UDP
22	Slow speed, asymmetric
23	satellites
24	wireless
25	Kernel implemenation, web100
26	Cluster TCP, zero copy
27	review

TCP for various networks

- We have a rich collection of TCP flavors and tuning options
- Our emphasis has been getting TCP to perform well over long delay, high speed networks
- In the next few sessions, we will look at which flavors and options are needed to make TCP perform well over various other networks
  - Slow links
  - Asymmetric networks
  - Satellite (long-delay) networks
  - Wireless/mobile/adhoc networks

TCP for slow links: compression

- Reducing number of bytes you have to send
- Application layer
  - bzip/compress/zip
  - MPEG/jpg/MP3
- Network
  - Application layer (PGP, bbcp)
  - Presentation layer (SSL)
  - Transport layer
    - TCP header compression
  - Link layer (modems)

Link layer compression (modem)

- For slow speed links, compressing data so you put fewer bits on the wire is a big win.
- Compression employed in hardware modems for 56k dialup and ISDN

Table 6-6 Maximum Transmission Speeds Specified by Modem Standards

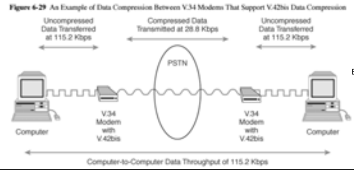
Business Data Communications (6e)

Modem Standard	Maximum Transmission Rate	Baud Rate(s)
V.90	56,000 bps (downstream)	3,000, 3,200 for downstream
	33,600 bps (upstream)	2,400, 2,743, 3,429, 3,800 for upstream
V.34	28,000 bps/33,600 bps	3,000, 3,200 for 28.8 kbps 2,400, 2,743, 2,800, 3,429 for 33.6 kbps
V.32ter	19,200 bps	2,400
V.32bis	14,400 bps	2,400
V.32	9,600 bps	2,400
V.22bis	2,400 bps	600
Bell 212A	1,200 bps	600
Bell 103	300 bps	300

Data Compression

- Modem *data compression* capabilities enable modems to have data throughput rates greater than their maximum bit rates
- This is accomplished by substituting large strings of repeating characters or bits with shorter codes
- Widely supported standards for data compression include
  - *V.42bis* --- up to 4:1 compression using the Lempel Ziv algorithm
  - *MNP Class 5* --- supports 1.3:1 and 2:1 ratios (via Huffman encoding and run-length encoding)
  - *MNP Class 7* – up to 3:1 compression
  - *V.44* --- capable of 20% to 100% improvements over V.42bis

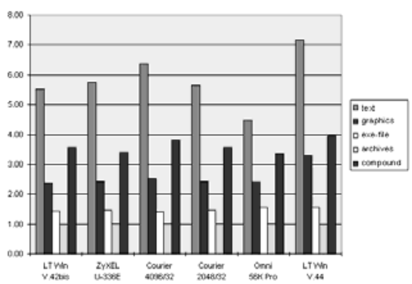
Figure 6-29 An Example of Data Compression Between V.34 Modems That Support V.42bis Data Compression



Business Data Communications (6e)

v.44 vs v.42

- Compression reduces payload by a factor of 3 or more ➔ improves throughput by a factor of 3 or more

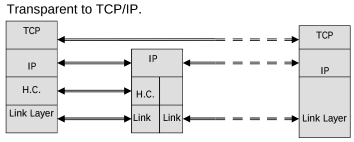


TCP header compression: Motivation

- TCP/IP header size is (at least) 40 bytes.
- Significant overhead for small packets
- Example: Using telnet over slow modem connection.
  - In many cases the data size is one byte.
  - 40 bytes of header, then return ACK is 40 bytes (52 bytes with timestamp)
- Solution: Compress TCP/IP headers
  - Improve TCP/IP performance over low speed serial links.
  - Defined in RFC 1144

TCP/IP Header Compression

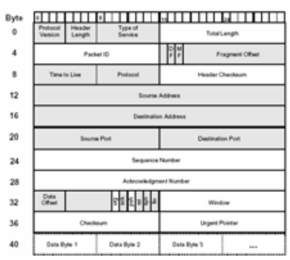
- This is not an end to end compression.
  - Compression is done in the entry point of the (slow) serial link.
  - Decompression is done in egress point of the serial link.
  - Compression is done between the network and the link layers.
    - In the SLIP driver
    - Transparent to TCP/IP.



Basic Idea

- The sender and receiver keep track of active connections
- The receiver keeps a copy of the header from the last packet from each connection.
- Differential coding: The delta between the current and the previous packet is sent.
- Constant fields
  - In a TCP connection many fields are likely to remain constant.
  - A connection number is sent instead of these fields.

Constant Fields



•Some fields are unnecessary

IP checksum (IP header is not transmitted).

Total length (redundancy with layer 2 protocols).

Changeable Fields

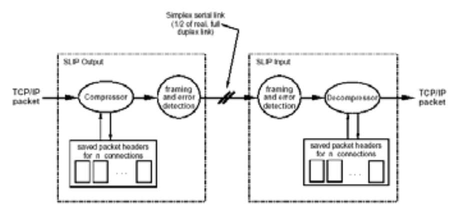
- Other fields can be changed.
- Nevertheless, they do not all change at the same time.
  - e.g. in an ACK packet the sequence number may remain constant.
- The sender sends only the fields that are changed.
  - It uses the copy of the last packet that was sent for each connection.
  - A bit mask that indicates which fields were sent
- How the fields change?
  - The difference between current and previous packet ID is small (usually < 256, i.e. one byte).
  - The difference between current and previous sequence number is less than 2^{16} (i.e. 2 bytes).
- The differences in the changing fields are sent rather than the fields themselves.

Packets Types

- The sender can send 3 types of packets.
- The packet type is stored in the header of the link layer protocol.

1. TYPE_IP packets are regular uncompressed IP packets.
   Non-TCP packets.
   Uncompressible TCP headers.
2. UNCOMPRESSED_TCP packets are identical to the original packets except the IP protocol field is replaced with a connection number.
   Use to (re-)synchronizes the receiver.
   Use to send a TCP packet of new connection.
3. COMPRESSED_TCP.

The Compression System



The Compression System

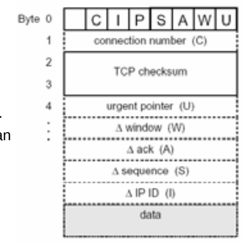
- IP packets goes through the compressor.
- Non-TCP packets and uncompressible TCP packets are marked as TYPE_IP and passed to the framer.
- Compressible TCP packets are looked up in an array of packets headers.
  - If a matching connection is found:
    - The incoming packet is compressed.
    - The uncompressed header is copied into the array.
    - A packet of type COMPRESSED_TCP is sent to the framer.
  - If no match is found:
    - The header is copied into an array of packet headers.
    - A packet of type UNCOMPRESSED_TCP is sent to the framer.

The Compression System

- The decompressor does switch on the type of incoming packets.
- TYPE_IP packets simply pass through.
- UNCOMPRESSED_TCP packets
  - The connection number is extracted and used as an index into the array of saved headers.
  - The header is copied into the array
  - IPPROTO_TCP is restored in the protocol field in the IP header.
- COMPRESSED_TCP packets
  - The last packet from that connection is extracted from the array of saved header using the connection number.
  - The compressed header is used to restore a new TCP/IP header and construct a new TCP/IP packet.
  - The new header is stored in the array.

Compressed Packet Format

- The first byte is a bit mask that identifies which of the fields are actually changed.
- TCP Checksum of the original packet is located in the compressed header.
  - An end-to-end integrity check is still valid.
  - Used for error detecting and resynchronize.
- The delta's of fields are usually smaller than 255.
  - One or two bytes are used to encode the difference.



Uncompressible TCP Headers

- TYPE_IP
  - Fragmented IP packets.
  - If any of the TCP control bits (SYN, FIN, RST) are set or if the ACK bit is CLEAR.
    - Only when the connection is established or terminated.
- UNCOMPRESSED_TCP
  - The difference between fields cannot be encoded (i.e. more than 2^{16}-1).
  - In case of negative sequence number or ack.

Notes

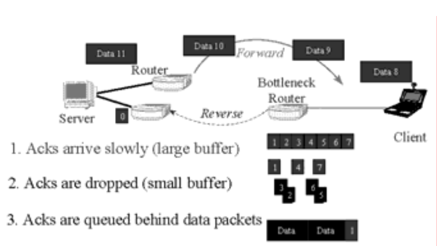
- The compression is a differential coding, thus the framer must not re-order packets.
- The framer must provide good error detection.
- If connection numbers are compressed, the framer must provide an error indication.
- The average compressed header size is ~3 bytes.
- Other header compression schemes
  - IP (RFC 2507)
  - RTP
- Also see RFC 3150 end-to-end performance for slow links

> Watch out when doing bandwidth tests (iperf or ttcp) over compressed links – get absurdly high througput.
>
> With ttcp you can input a compressed file to defeat compression
>
> ttcp –t host < file.tgz

TCP over asymmetric networks

- types of asymmetric networks
  - Bandwidth/capacity asymmetry
    - Forward and reverse path have different data rates
      - ADSL
      - HFC
      - broadband (6 mbs/ 360kbs)
      - Satellite/dialup (400 kbs downlink, dialup uplink 14k to 56k)
      - StarBand satellite (500 kbs down, 50 kbs up)
    - Similar effect can happen with cross-traffic on the reverse path
  - Media-access asymmetry
    - Wireless base-station has quicker MAC access than mobile nodes
      - Base station "owns" down-link
      - End nodes compete/collide for up-link channel (hub & spoke model)
    - Packet radio network
      - Half duplex – reverse and forward path traffic compete!
      - Wildly varying RTTs and ACK queuing
  - Loss rate asymmetry
    - Different link layer loss characteristics (satellite vs landline)
    - Or congestion on forward or reverse path inducing congestive loss
- See RFC 3449 (TCP on asymmetric paths)

Bandwidth/capacity asymmetry and TCP



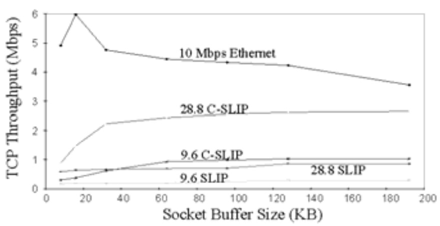
Asymmetric bandwidths

- Capacity of reverse path can limit performance of forward path
- Normalized bandwidth ratio, k
  - Forward path 10 mbs, reverse 100 kbs, raw bw ratio is 100
  - With 1000-byte packets in the forward, and 40-byte ACK in reverse, ratio of packet sizes is 25
  - k = 100/25 = 4
    - If there is more than one ACK for every 4 data packets then the reverse channel will get saturated before the forward channel
- In ns you can experiment with asymmetric paths
  - Use simplex-link in place of duplex-link

```
$ns simplex-link $n0  $n1 $linkbw   $linkdelay   DropTail
$ns simplex-link $n1  $n0   28k     $linkdelay   DropTail
```

Asymmetric bandwidths

- Example 10 mbs forward path with 28.8 kbs reverse path
  - ACK's with timestamp option 52 bytes, data packet size 1000 bytes
  - Use compressed TCP header (C-SLIP) reduce ACK size to 18 bytes
  - Delayed ACKs help too (half as many ACK packets)



Reverse path loss

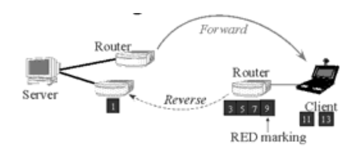
- Reverse path (ACK) will also have finite router buffer space
- ACK's can be dropped
  - Cumulative ACK lets TCP proceed
  - But sender may become bursty, which may cause forward path losses
  - Sender TCP algorithms based on ACK counting
    - Slower slow-start
    - Slow linear recovery
    - Disrupt fast retransmit
    - Sender pause waiting for ACKs
- Traffic on the reverse path further reduces available bandwidth
  - e.g. your doing web surfing (outbound) while downloading file (inbound)
  - Outbound ACKs for the file transfer may get dropped, stalling transfer
- Some of our atou experiments from home experienced ACK-limited throughput (at first, mistakenly sent full (null) SACK blocks)

Solutions for the reverse (ACK) path

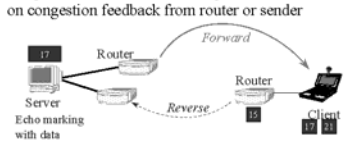
- Reduce frequency of ACKs
  - ACK congestion control (ACC)
  - ACK filtering (AF)
- Handling infrequent ACKs
  - Sender adaptation (SA)
  - ACK reconstruction (AR)
- Scheduling mechanisms
  - ACK-first scheduling (AFS)
- None of the above are elegant – work arounds

ACK congestion control (ACC)

- Use RED to throttle sender, but mark on ACK path

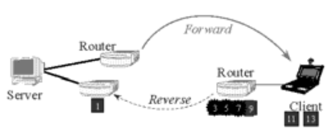


Adaptive extension of TCP delayed acks based on congestion feedback from router or sender



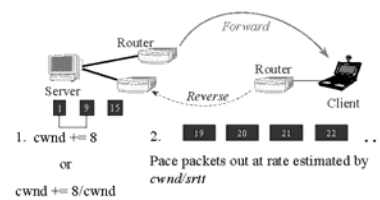
ACK filtering (AF)

- Purge all redundant, cumulative ACKs from constrained reverse queue
- Deterministic or random purging
- Use in conjunction with sender adaptation or ACK reconstruction



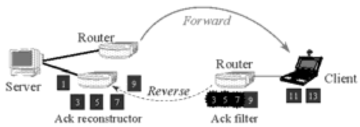
Sender adaptation (SA)

- With infrequent ACKs
  - TCP window growth is slower (slow-start and linear recovery)
    - Could do "byte counting" instead of ACK counting
      - e.g., a delayed ACK counts as two ACKs
  - Sender tends to be bursty (cumulative ACK release)
    - Pace data packets instead of burst



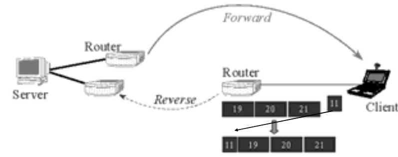
ACK reconstruction (AR)

- Reconstructor regenerates ACKs at other end of constrained channel
- Shields sender from large gaps in ACK sequence, reduces burstiness
- ACKs are regenerated at rate depending on
  - Input rate from constrained channel
  - Number of required ACKs (target ACK spacing)



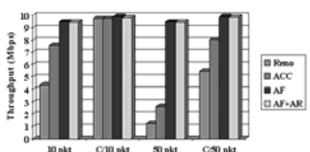
ACKs-first scheduling (AF)

- Bi-directional traffic
  - Both data and ACKs sharing reverse channel
- Move ACKs to the front



Uni-directional performance

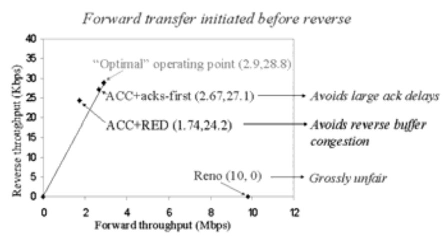
- TCP transfers in the forward direction alone
- Maximum window size set to 100 KB; no losses on forward path



- Header compression helps
- Large reverse channel buffer hurts for Reno and ACC
- Fairness greatly improves using AF and ACC for multiple transfers

Bi-directional transfers

- Inbound and outbound data transfers
- ACC + ACKs-first close to optimal
  - Optimal: max forward throughput when reverse throughput is largest



Bandwidth asymmetry summary

- Good solution has several components
- Header compression reduces problem
- Reduce the frequency of ACKs over reverse channel (ACC + AF)
- Handle infrequent ACKs (SA + AR)
- Move ACKs to front (AF)

- I don't think any of these are widely deployed except in proxies (later)

Next time …

- Satellite nets and inter-gallactic TCP

assignment 9 and 10