Delays and IPC

CS144 Review Session 2
April 11, 2008
Ben Nham
Announcements

• Except for things that pertain only to you, use the newsgroup, not the staff list
Delays

• We want to send a message between two hosts, hooked up directly via a link
  – Message is of length $L$ bits
  – Link’s transmission rate is $R$ bits per second
  – Hosts are distance $d$ meters apart

• What are the relevant delays?
  – Propagation delay
  – Transmission delay
Propagation Delay

• The time it takes to send one bit from A to B
• Bounded by:
  – Length $d$ of link
  – Propagation speed $s$ of link
    • Bounded by speed of light
    • Usually about $2 \times 10^8$ m/s
• Prop delay = $d/s$
Transmission Delay

- Time it takes for all bits of the message to be pushed onto the link
- Bounded by:
  - Length L of message
  - Transmission rate R of link
    - E.g. fully utilized Fast Ethernet = 100 Mbps
    - ADSL = 1.5 Mbps (varies)
- Transmission delay = L/R
Total Delay

• For one link:
  – Total Delay = Prop delay + Trans delay
    = \( \frac{d}{s} + \frac{L}{R} \)
  – Suppose:
    • \( d = 1 \text{ km}, \ s = 2 \times 10^8 \text{ m/s} \rightarrow t_{\text{prop}} = 5 \text{ μs} \)
    • \( L = 500 \text{ bytes}, \ R = 100 \text{ Mbps} \rightarrow t_{\text{trans}} = 40 \text{ μs} \)
    • Total = \( t_{\text{prop}} + t_{\text{trans}} = 45 \text{ μs} \)
• What happens when we compose links, placing routers in between?
  – Have to add a queuing delay component
Worst-Case Bisection Bandwidth

- Take a network topology, and partition the nodes in two equal parts
- Bisection bandwidth = bandwidth of links that travel across partition
- The worst possible bandwidth between all sets of equal partitions is the worst-case bisection bandwidth
- Note: people commonly use “bisection BW” to mean “worst-case bisection BW”
Worst-Case Bisection Bandwidth

• Take a network topology, and partition the nodes in two equal parts
• Bisection bandwidth = bandwidth of links that travel across partition
• The worst possible bandwidth between all sets of equal partitions is the worst-case bisection bandwidth
• Note: people commonly use “bisection BW” to mean “worst-case bisection BW”
Client-Server File Transfer

• Suppose Blizzard wants to distribute a patch for WoW
  – Patch size is 450 MB = 3600 Mbits
  – Need to distribute patch to 5 million players over 5 hours
  – How many 100 MBps servers are needed?
• 3600 Mbits * 5 million / 5 hours = 1*10^{12} bps
• 1*10^{12} bps / 100 Mbps = 10000 servers
Peer to Peer File Transfer

• Use **one** 100 Mbps server, and suppose each client has a 1Mbps full-duplex link
• Split the patch into 450 chunks, 1MB each
• Then:
  – Seed each chunk to one client each
  – Wait for every client to get at least one chunk
  – Wait for every client to acquire the other 449 chunks
Seeding

- Seed each chunk to one client each
- $450 \times 1 \text{ MB} / 100 \text{ Mbps} = 1 \text{ min to seed}$
Exponential Growth

- Wait for every client to get at least one chunk
- We have 450 seed clients with 1 Mbps links
  - 5 million clients / 450 seeds = 11112 clients/seed
  - This distribution takes place exponentially:
    - ceil(log₂ 11112) = 14 time steps
    - Each time step is 1 MB / 1 Mbps = 8 s
    - Total time: 14 * 8 s = 2 minutes
Exponential Growth

• Wait for every client to get at least one chunk
• We have 450 seed clients with 1 Mbps links
  – 5 million clients / 450 seeds = 11112 clients/seed
  – This distribution takes place exponentially:
    • ceil(log_2 11112) = 14 time steps
    • Each time step is 1 MB / 1 Mbps = 8 s
    • Total time: 14 * 8 s = 2 minutes
Exponential Growth

• Wait for every client to get at least one chunk
• We have 450 seed clients with 1 Mbps links
  – 5 million clients / 450 seeds = 11112 clients/seed
  – This distribution takes place exponentially:
    • ceil(\log_2 11112) = 14 time steps
    • Each time step is 1 MB / 1 Mbps = 8 s
    • Total time: 14 * 8 s = 2 minutes
Peak Transfer

• Now each client has a single chunk
• Everyone can utilize their full 1Mbps connection
• For any client, takes $449 \times 8s = 1$ hour to download the rest of the chunks in the patch
• Adding up:
  – Seeding takes 1 minute
  – Exponential growth until everyone has a chunk takes 2 min
  – Finishing transfer takes 1 hour
  – $1 \text{ min} + 2 \text{ min} + 1 \text{ hour} \ll 5 \text{ hours}$
  – $1$ server $\ll 10000$ servers
Lab 2: Fingerd

• Finger service isn’t enabled locally on cluster, so use “finger user@whois.stanford.edu” instead

• You will need to understand:
  – Writing servers (last week)
  – New IPC syscalls:
    • fork()
    • exec*(…)
    • pipe()
    • dup2(src, target)
Forking

- Makes a copy of the current process (inherits everything: memory space, file descriptors, signal handlers, ...)
- The child will return with 0
- The parent will return with the pid of the child
- Parent must either:
  - Explicitly wait on the child
  - Install a signal handler that catches the SIGCHLD from child and waits in the signal handler
  - Otherwise, zombie process created!
- Can use exec* to replace ourselves with another executable

Figure 2-1, The Design and Implementation of 4.4BSD
Example: Forking Daytime Server

• See daytime2.c
Pipes

• A pipe is like a one-way socket
• Useful for communicating from parent to child, or vice versa
• Bundles two file descriptors together:
  – 0\textsuperscript{th} can be written to (analogous to stdin)
  – 1\textsuperscript{st} can be read from (analogous to stdout)
• Usual usage for parent to child comm (switch for child to parent comm)
  – Create pipe
  – Fork
  – Parent closes read side of pipe, writes to write side of pipe
  – Child closes write side of pipe, reads from read side of pipe
Duplicating Descriptors

• int dup2(int old_desc, int new_desc)

• After execution, new_desc is a copy of old_desc

• Example: Sending stdout to stderr
  – 2 is fd for stderr by default
  – 1 is fd for stdout by default
  – dup2(2, 1)
Example

• Executing the equivalent of "cat /usr/share/dict/words | wc -l"

• See numwords.c
Resources

• IPC tutorial on assignment webpage
• Man pages
• Tutorials: