Tech Topic #4

February 1, 2010

Today's Objectives

- Leveraging one-to-many delivery
- Wednesday is project updates
  - 15 min meetings on Wednesday (between 7am and 11am)
  - Informal assignment is to work on paper outline
  - Send me email with slot preference (first come, first served)
- Schedule is updated
  - Monday is gaming
  - Tech topics on multimedia for mobile devices and multimedia for wireless devices

Where the Replication Happens

- With multicast, (native, hierarchical or ALM), the challenge is to get people interested in receiving the same content
  - For streaming, hard to do on-demand service
  - For content delivery, reliability and congestion control are hard
- For one-to-many distribution today, the key is (number_of_receivers *x* stream_bandwidth)
  - If the product is large (either component), one-to-many makes sense

Streaming Applications

- Not much demand for actual streaming
  - Except maybe video conferencing and gaming
  - There, solutions are more typically an exploder/replicator/aggregator
- “Streaming” now is more of a progressive download
  - Even buffering is of limited necessity
  - Key is greater bandwidth than playout rate
    - Then only small buffer is necessary
- With cheap memory and disk space, prefetching is an easier solution than coordinating receivers

IMJ Novel Aspects

- Tradeoff between number of channels and on-demand like service
  - The more channels, the less time a user would have to wait
- Distributed server architecture
  - Multicast source could be located anywhere
- Practical issues
  - Content capture was packet trace and replay
  - Service involving real copyrighted content

Reliability

- Pretty intuitive that any kind of retransmission scheme for wide-scale file delivery is hard
  - Kills scalability
  - All manner of solutions have been proposed to offload the repair work from the primary distribution server
- Another problem is synchronizing receivers to the transmission
  - The Starburst Case Study
- In the late 1990s, work was moving towards some kind of parity or forward error correction (FEC) approach

Network Coding

- Lots of different schemes for how to mix original data with redundant data
  - Factor is the characteristics of the loss and how bursty it is
  - Another is how large the original block of data is
- Even some hybrid solutions
  - Mix of NACK and FEC (the predecessor to Sigcomm ‘98)
  - Data carousel combined with FEC
- Proposal was for a “digital fountain”

Applicable Network Coding

- Reed Solomon
  - Spread *k* packets worth of data over *n* transmitted packets
  - Receive any *k* packets, to re-construct file
  - If loss is high enough, may not receive n of k packets
  - Can cycle through *n* packets again
  - Basic idea: basic mix of data elements from file
- Tornado codes
  - Faster
  - But have to receive >n packets
  - Basic idea: more like a logic problem of XORed bits
- Basic tradeoff is processing v. overhead/bandwidth
  - Good observation in analyses that tradeoff has changed

Tornado Codes into a Protocol

- Some very nice features
  - Receivers can just join the group, no need to inform source
  - Though source probably wants to know for tracking
  - Receiver can stay as long as necessary
  - For certain kinds of applications (software updates), stream of packets can be sent continually
  - If no receivers, multicast tree is pruned back to the source

Additional Comments

- Not sure why the evaluation had to be so opaque
- Large files are easy to deal with
  - Break them into smaller files: pyramid broadcasting
- Handled congestion via layered multicast
  - As it turns out, not a particularly effective solution
  - Attempt to mimic TCP backoff through layering
  - Congestion control for multicast is hard
    - Hard to mimic TCP AIMD (and therefore, hard to be fair)
  - Lots of “leave latency” associated with multicast

Next Challenge

- Great theoretical technique, hard to apply in practice
  - No multicast
  - Would be an interesting study to compare multicast-based layered DF approach to unicast
  - Could also compare to other distribution techniques
    - What would the metrics be?
- To overcome the problem of no native multicast, develop a unicast-based approach
  - Instead of a traditional server delivering content, think of a specialized piece of hardware that is super efficient at supporting lots of TCP connections

Scalable TCP

- Use the Tornado codes on unicast data
- Instead of buffering data on the server side, draw from an endless supply of Tornado code-generated data
  - Even for retransmissions, a different packet can be sent
- How is this better than UDP?
  - Congestion control, reliability, used existing protocol, no firewall filtering, no multicast required
  - Any lack of one of these features would have made the solution a non-starter
- Final determinant of deployability becomes simplicity