CS 838 Building Fast Networks

Chapter 3 – Fifteen implementation principles

Algorithmics example

- Forensics problem
  - Outside sources talking to more than 5 internal destinations are suspicious
  - Keep a forensic log with all of their packets
  - Even those before they contacted 5 destinations
- Solution idea
  - Go back in time at the router
  - Buffer all packets for say 10 seconds
  - When source turns bad, pluck its pkts from buffer

Algorithmics example (contd.)

- How do we find the packets in the buffer?
- Indexing by source is expensive
- Solution:
  - Build the forensic log as you clean up the buffer
  - Might introduce delay of less than 10 seconds

15 implementation principles

- Meant to stimulate creativity, not replace it
- Abstract principles – “overlaps” possible
- Structure of the chapter
  - P1 to P5 systems thinking
  - P6 to P10 modularity with efficiency
  - P11 to P15 speeding up key routines
- Questions to ask before you start work

Systems thinking

- P1 – Avoid obvious waste
- P2 – Shift computation in time
  - P2a – Precompute
  - P2b – Evaluate lazily
  - P2c – Share expenses, batch
- P3 – Relax system requirements
  - P3a – Trade certainty for time
  - P3b – Trade accuracy for time
  - P3c – Shift computation in space

Systems thinking (contd.)

- P4 – Leverage off system components
  - P4a – Exploit locality
  - P4b – Trade memory for speed
  - P4c – Exploit existing hardware
- P5 – Add hardware
  - P5a – Use memory interleaving and pipelining
  - P5b – Use wide word parallelism
  - P5c – Combine DRAM and SRAM effectively
Modularity with efficiency

- P6 – Create efficient specialized routines
- P7 – Avoid unnecessary modularity
- P8 – Don’t be tied to reference implementation
- P9 – Pass hints in layer interfaces
- P10 – Pass hints in protocol headers

Speeding up key routines

- P11 – Optimize the expected case
  - P11a – Use caches
- P12 – Add state for speed
  - P12a – Compute incrementally
- P13 – Exploit existing degrees of freedom
- P14 – Use bucket sorting, bitmaps
- P15 – Create efficient data structures

Don’t rush in

- Is it worth improving performance?
- Is this really a bottleneck?
- What impact will this have on rest of system?
- Does initial analysis allow big improvement?
- Is it worth adding custom hardware?
- Can protocol changes be avoided?
- Do prototypes confirm initial promise?
- Will gains be lost if environment changes?