

CS228A: Introduction, Outline, and Need for Principles

George Varghese

January 7, 2002

Course goal: how to efficiently implement a protocol?

- Outline a set of 15 principles/heuristics that have been useful for speeding up implementations. Comes together with statutory warnings.
- Describe a set of basic tasks (e.g., timers, lookups) that occur in most protocols. Describe efficient techniques for implementing these tasks, and showing how these techniques use the basic principles.
- Give you a chance to apply the principles to a set of 16 problems that we will work out during the course.
- To make the tutorial self-contained, provide background on some important tasks (e.g., fair queuing, switching, load balancing) that you may be less familiar with.

Many practical examples based on actual products and implementations.

Why Principles: A Coin Weighing Puzzle

Consider a coin weighing puzzle:

- Thirteen coins, one heavy or light.
- Given a balance to weigh coins against each other but not absolute weights.
- Can you find the anomalous coin in 3 weighings.
How about 364 coins and 6 weighings?

*Instead of computing, I had to think about
the problem, a formula for success that I
recommend highly.*

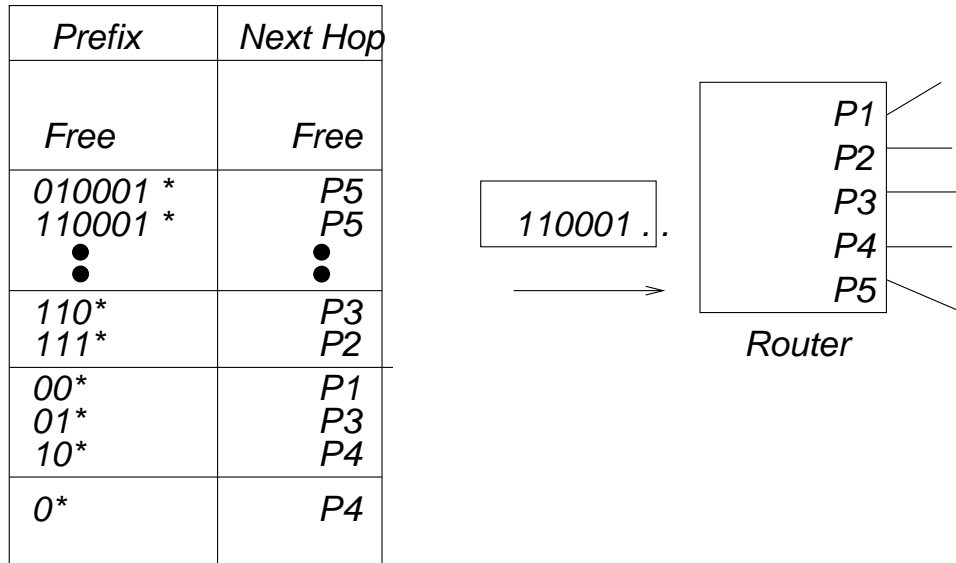
— Ivan Sutherland

How do you think about the problem?

Possible principles to exploit

- **Exploit degrees of freedom:** 728 outcomes needed, 3 outcomes of each weighing. 6 weighings implies 3^6 outcomes. Tight. Every weighing must provide 3 outcomes.
- **Use algorithmic ideas:** Recursion. 2 Subproblems.

Why Principles: A Ternary CAM Puzzle

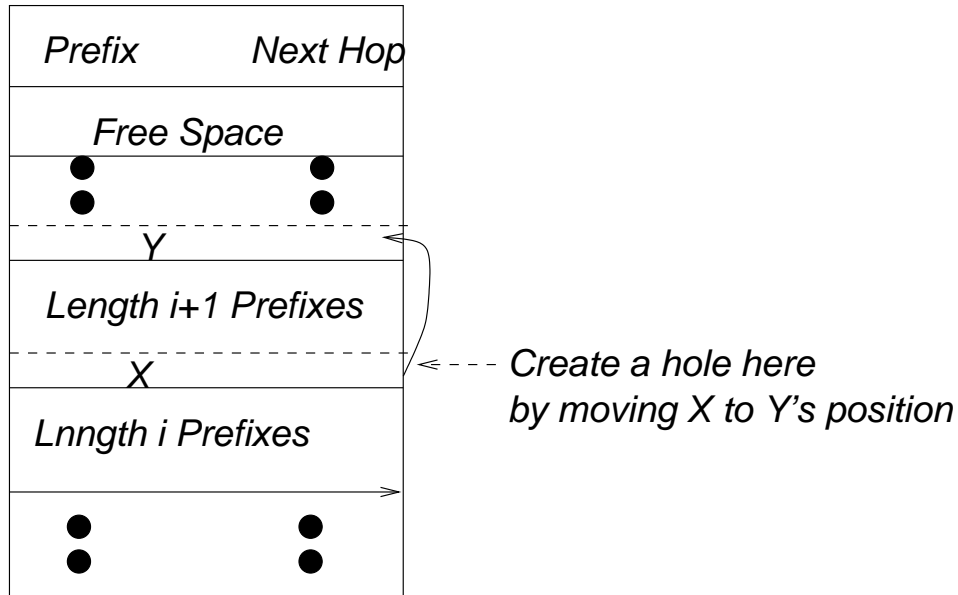


- Longest matching prefix problem for packet forwarding by a router. Match 32 bit IP address to longest match.
- Ternary CAM searches all locations in parallel to output (in one cycle) the lowest memory location that matches, considering don't cares.

Update Problem

- Ternary CAM requires sorting prefixes so that a prefix like 1000* must come before a matching prefix like 100*. Sort of like sorting.
- Thus if we have 100,000 prefixes (backbone today), does adding a new prefix requiring 100,000 memory accesses to create a single hole?

Thinking instead of computing



- **Understand and Exploit Degrees of Freedom:** Only require ordering between prefixes of *different* lengths, not of same length.
- **Use algorithmic techniques:** Recursion again, reduce to a smaller problem. Here “smaller” is closer to free space at top.

Can do even better. Can you squeeze some more juice out of this lemon. Exploit more degrees of freedom.

And so . .

- We will proceed to study a number of protocol implementation issues. Before we do so, we need to study the rules of the game. What is hard, what is easy in say hardware design (routers), OSs (endnodes).
- Then we will study the principles in detail. Then work our way through 16 mini-examples. Then we can start the course! Routers first and then endnodes. See you next time.