# EECS 122, Lecture 12

Kevin Fall

kfall@cs.berkeley.edu

## IP Forwarding

- Hosts and routers may need to perform indirect delivery (through a router)
- Often hosts have a single router to use ("default router"), but in general both routers and hosts can make use of multiple routers
- Question: Which router to use?

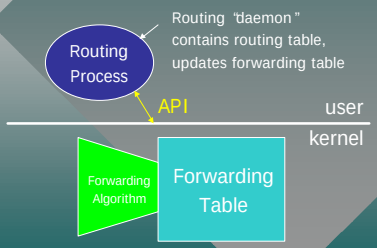
## Which Router to Use?

- Really two questions:
  - what algorithm and data structures are needed at a router to properly forward an IP datagram to its next hop
  - how do the routers become aware of the optimal paths (next hops) to use
- Focus on the first question for now…

## Routing Tables

- Hosts and routers maintain tables containing routing information
- More precisely, *routing tables* are used to compute optimal routes and *forwarding tables* are used to dictate immediate next-hops for each destination
- Forwarding tables are updated by routing protocols based on the routing tables

## Typical Configuration




## Forwarding Tables (FIBs)

- Also called "forwarding information base," (FIB) generally contains the following:

| Mask | Destination | Next Hop |
|---|---|---|
| M1 | D1 | H1 |
| M2 | D2 | H2 |
| M3 | D3 | H3 |
| M4 | D4 | H4 |

## Using Forwarding Tables

- Table entries (routes) are selected based on destination address of packet to route

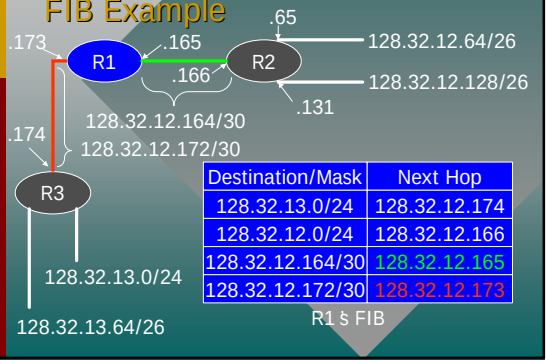
| Mask | Destination | Next Hop |
|---|---|---|
| M1 | D1 | H1 |

- So, to route to destination D, we compute (D & M1) and compare with D1. If they match, H1 is a possible next hop.

## Using Forwarding Tables

- By comparing (D & $M_i$) for each route entry *i*, we get a set of matching route entries
- The entry with the longest prefix (most number of '1' bits in the mask field) is the single winner
- Algorithm called "longest" or "best" prefix match, and is used by all IP routers

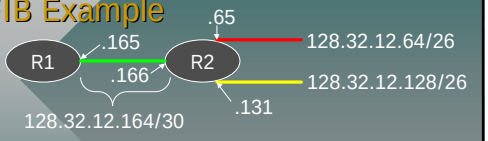
## FIB Example




| Destination/Mask | Next Hop |
|---|---|
| 128.32.13.0/24 | 128.32.12.174 |
| 128.32.12.0/24 | 128.32.12.166 |
| 128.32.12.164/30 | 128.32.12.165 |
| 128.32.12.172/30 | 128.32.12.173 |

R1's FIB

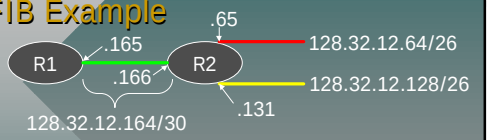
## FIB Example




| Destination/Mask | Next Hop |
|---|---|
| 0.0.0.0/0 | 128.32.12.165 |
| 128.32.12.164/30 | 128.32.12.166 |
| 128.32.12.64/26 | 128.32.12.65 |
| 128.32.12.128/26 | 128.32.12.131 |

R2's FIB

## FIB Example




| Destination/Mask | Next Hop |
|---|---|
| 0.0.0.0/0 | 128.32.12.165 |
| 128.32.12.164/30 | 128.32.12.166 |
| 128.32.12.64/26 | 128.32.12.65 |
| 128.32.12.128/26 | 128.32.12.131 |

R2's FIB

Packet destined for 128.32.12.187 matches the 1st and 4th routes, but the 4th is longest, so the packet is locally delivered on the yellow link.

## Implementation

- Longest Prefix Match Algorithm
  - must be executed for each packet
  - must work with the entire Internet routing table (2/25/99: 42869 entries at MAE-E, 16 and 24 bit prefixes dominate [~28k])
- Two Common Software Approaches
  - Hash-based lookup
  - Tree-based lookup

## Tree-Based Lookups

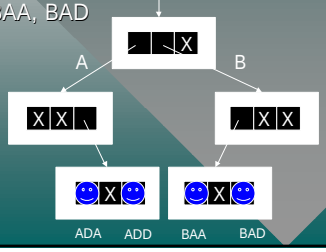
- Tree search appears useful, but should not be based on direct key compares (we are using bit prefixes, not full keys)
- Rather, observe each key (destination address) is composed of a series of bits
- Each tree depth will map one or more bits in the search key

## Search Tries (from reTRIEval)

- M-ary tree, where each node is an M-place vector with components corresponding to characters or digits
- Each node on level *l* represents all valid keys beginning with a certain sequence of *l* characters
- Each node specifies an M-way branch, depending on the (M+1)st character/digit

## Simple Trie Example

- Alphabet: {A,B,D}, words: ADA, ADD, BAA, BAD

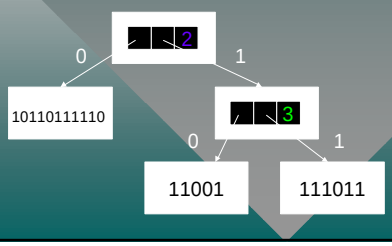



## Patricia Trees

- Keys are represented by binary strings
- To avoid 1-way branching that would happen in a simple binary search trie, include a *skip* value, which indicates how many bits to skip before examining the next bit at each node
- Sometimes called "path compression"

## Simple Patricia Example

- Keys: 11001, 111011, 10110111110

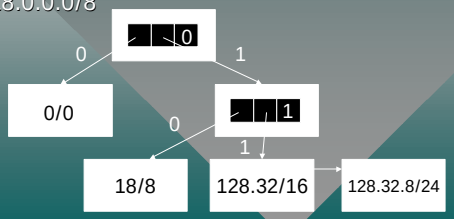



## Patricia Tree Properties

- N-1 internal nodes, N external nodes
- search performance insensitive to insertion order, but very sensitive to distribution of digits
- assuming random distribution, approx # of digits/bits inspected is about $\log_M N$, for M-ary trie with N nodes [see Knuth vol3 for the (nontrivial) analysis]

## Simple P-Tree for IP Routing

- (from BSD, circa 1990)
- Keys: 0/0, 128.32.0.0/16, 128.32.8.0/24, 18.0.0.0/8




## Observations

- This (simple) scheme places common prefixes together in the same leaf (using a linked list)
- Parent pointers in nodes are used to back up in the tree in case of a mis-match (default router or non-contiguous mask)
- Various new approaches have been developed: DP-Tries, LC-Tries, etc.

## Hashing-Based Approaches

- Original BSD routing support included hash on network routed and host routes, but this was before LPM was required
- Newer hash based scheme provides 32 different hash tables, and looks for exact match
- Simplest: perform linear search for the right hash table, starting w/longest

## Reaction to LPM

- Three main approaches to problem of IP lookups:
  - faster software algorithms
  - hardware acceleration (e.g. CAMs)
  - protocol modifications
- Faster algorithms have called into question the protocol approaches

## Hardware Approaches

- Content Addressable Memory (CAM)
  - exact-matching CAMs can be used 1 per prefix length
  - 32 CAMs for IPv4, 128 for IPv6 (expensive)
  - Max size about 8k right now

## Protocol Approaches

- IP and Tag/Layer 3 "Switching"
  - similar to notion of VCID in circuit-switched networks (and may use them!)
  - requires separate protocol to provide address/TAG mappings
- Basically uses IP packets and IP routing as "signaling" for circuit set-up

## L3 Switching

- Normal forwarding requires LPM lookup on destination address at each hop
- In L3 Switching, assign a short "tag" to each packet as it enters the network (same domain)
- Tags are rewritten at each hop, and might be carried in L2 (e.g. ATM VC's)

## L3 Switching Claimed Benefits

- Label provides fast lookup (1 cycle)
- Can incorporate additional policy or characteristics at ingress router, so that even pkts w/same dest go along different paths
- Provides some functions similar to source routing without the routing cost

## L3 Switching Complications

- Requires adoption of "label distribution" protocol, which may not span admin. domains
- Probably not worth it on a per-flow basis (web flows)
- Does not completely avoid LPM search
- Time to Live (TTL) question…

## The TTL Question

- In regular IP forwarding, TTL zero implies discard (good if loops arise)
- No TTL decrementing per hop here…
  - so, decrement it by the amount it should have been decremented along the complete path
  - requires egress LS router to know path length

## Why L3 Routing Still Needed

- So, can we use L3 switching for everything?
  - No, hard to support these things:
    - world-wide label assignment
    - packet filtering/firewalls
    - non L3-switching hosts
    - non L3-switching routers

## Multi-Protocol Label Switching (MPLS)

- IETF Internet Drafts addressing L3 switching
- Latest architecture document 2/99
  - take advantage of ATM switching
  - be compatible with RSVP/Int Serv, existing routing protocols
  - support aggregation, multicast, good OA&M

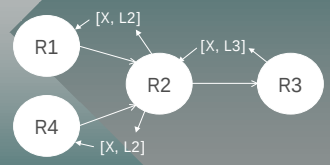
## MPLS Architecture Features

- Downstream assignment of labels for unicast traffic, driven by either control or data traffic, multiple label dist. protocols
- LIFO "stack" of labels for hierarchy
- Both "normal" and source-routing
- Variable label granularity, label merging
- TTL "fixup"

## Label Merging

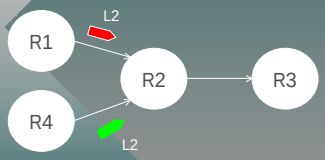
- Suppose you have >1 packets destined for the same IP address but that arrive with different labels
- The ability to assign the same outgoing label for both incoming packets is called "label merging" (some ATM switches)
- Useful to avoid maintenance of a large number of VCs between switches

## Label Merging Example



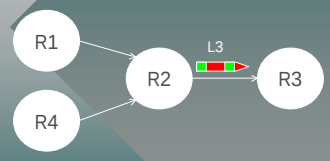

•[IP prefix, label] indicate tag mapping
•R2 is merge point for L2->L3

## Label Merging Example




•Packets P1,P2 take paths {R1,R4},R2,R3

## Label Merging Example




•Interleaved cells using label L3

## Merging on ATM

- Problem: cells within ATM/AAL5 contain VPI/VCI (VC) numbers. Hardware either carries each VPI/VCI as a label, or each VPI as a label.
- VC Merging: have to reconstruct AAL frames to avoid interleaving cells
- VP Merging: use common VPI with different VCIs to demultiplex