Vortragsfolien zu: ACM Symposium on Principles of Distributed Computing (PODC '98) June 28 - July 2 1998, Puerto Vallarta, Mexico

Scalable Prefix Matching for Internet Packet Forwarding

Marcel Waldvogel

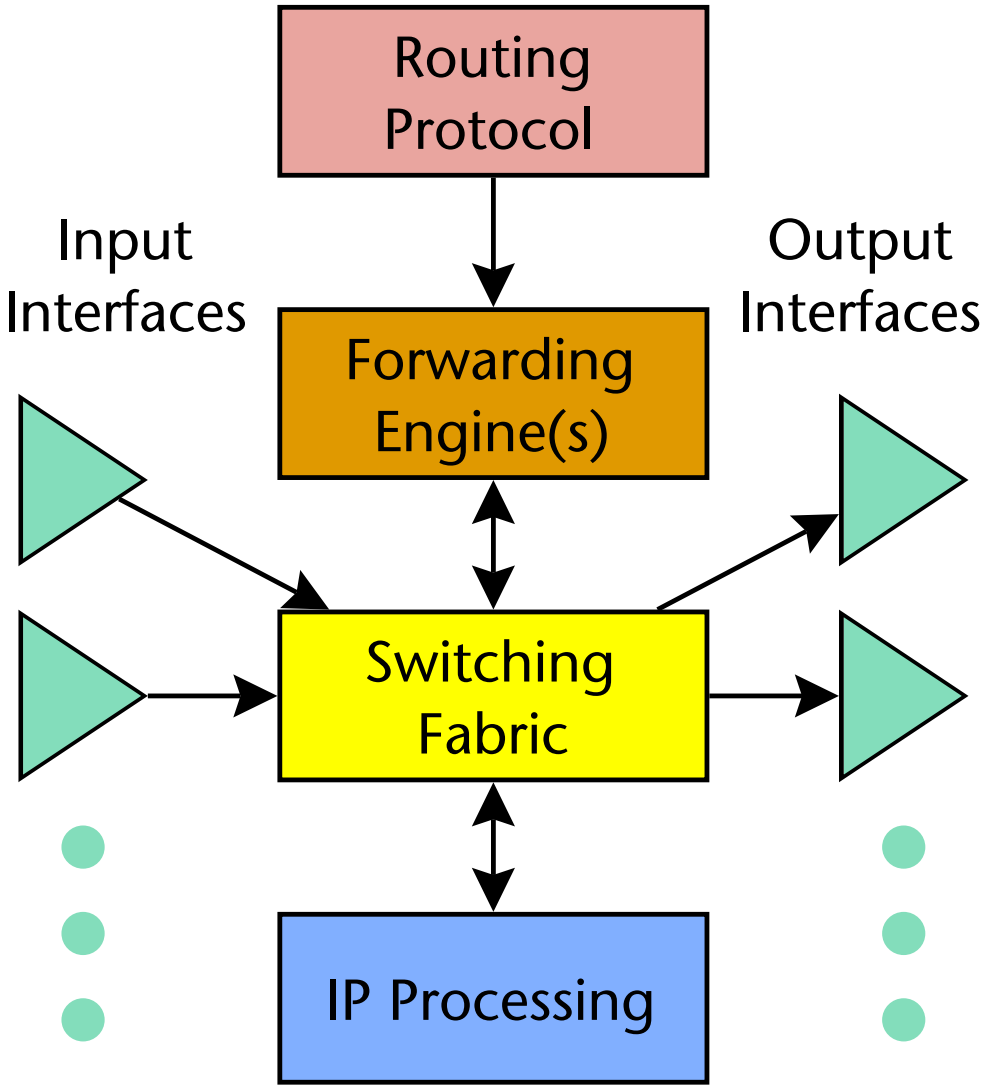
Computer Engineering and Networks Laboratory
Institut für Technische Informatik und Kommunikationsnetze

Background

- Internet growth
  - Bandwidth
  - Size
  - Complexity
- Classful $\rightarrow$ Classless Inter-Domain Routing (CIDR)
- IP version 6
- Demand for QoS

Routers

- Hop-by-hop
- TTL, checksum update
- Forwarding decision
- (Fair) queueing



Motivation

- Higher Link Speeds ✔
- Higher Data Throughput ✔
- Fair Queueing ✔
- Faster Forwarding Decision ?
- Packet Classification for QoS ?

Overview

- Current Routing Techniques
  - Routing Database
  - Patricia Tries
  - Faster Forwarding
- Binary Search on Prefix Lengths
- Build and Update
- Fast Hashing
- Analysis
- Conclusions

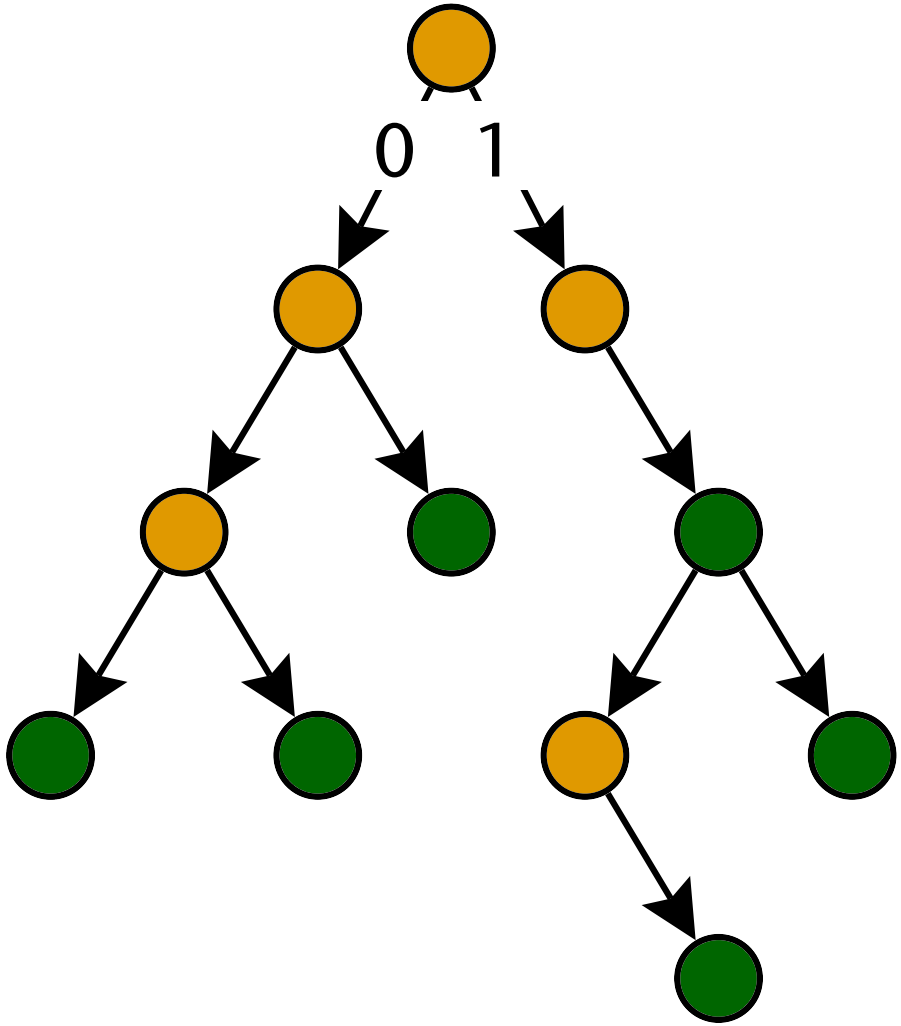
Routing Database

- Information spread through Routing Protocols
  - Per-network or per default
- Old (pre-CIDR): Hash tables
  - 3 prefix lengths (class A, B, C: 8, 16, 24 bits)
  - Length determined from address
- CIDR
  - Arbitrary prefix length
  - Best matching prefix (BMP)
  - Also for IPv6

128.252.0.0/16 = 1000000 11111100 *
129.132.66.64/26 = 1000001 10000100 01000010 01*

Patricia Tries

- Binary trie
- Entries vs. plain nodes
- Example: 110011



Faster Forwarding

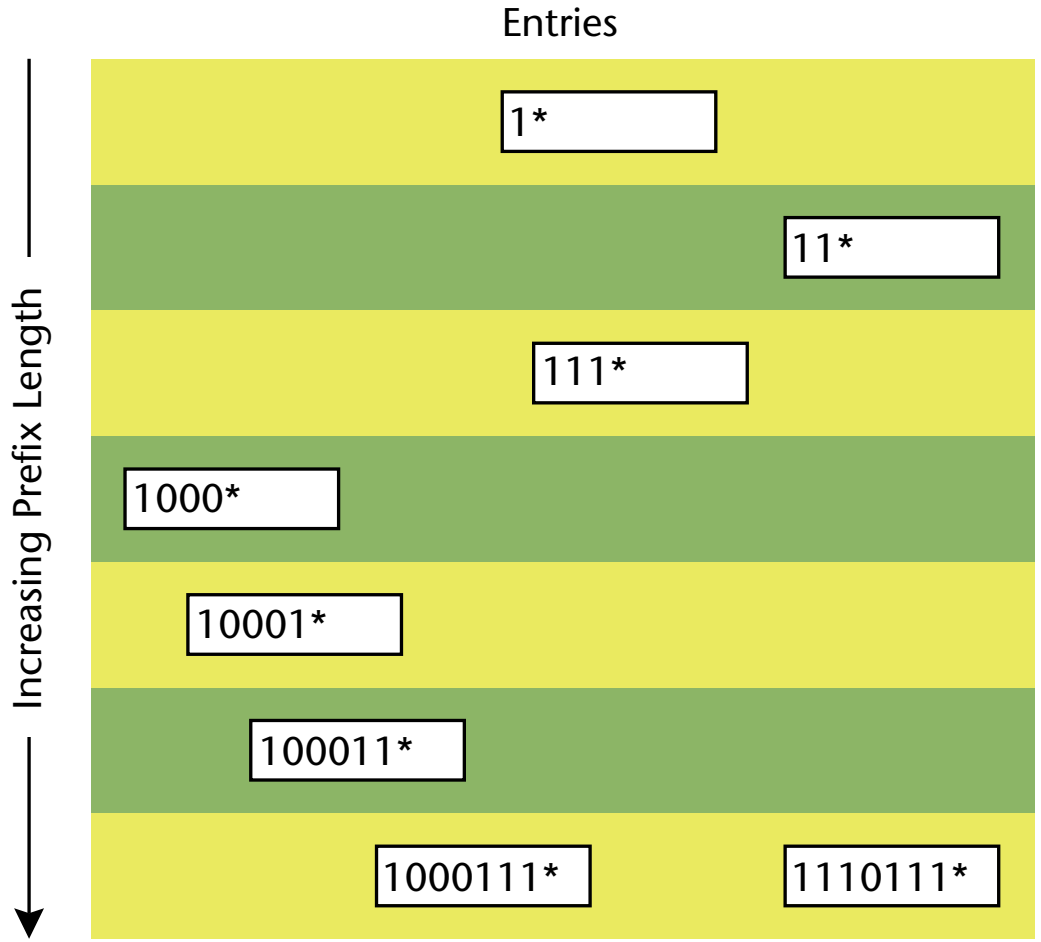
- Alternatives to Patricia
  - Multi-level tries
  - Binary search on prefixes
- Hardware
  - Content Addressable Memories (CAMs)
  - Hardware Patricia search
- Protocol solutions
  - Label Switching
  - ATM
- Caching

Overview

- Current Routing Techniques
- Binary Search on Prefix Lengths
  - Basic Scheme
  - Refinements
- Build and Update
- Fast Hashing
- Analysis
- Conclusions

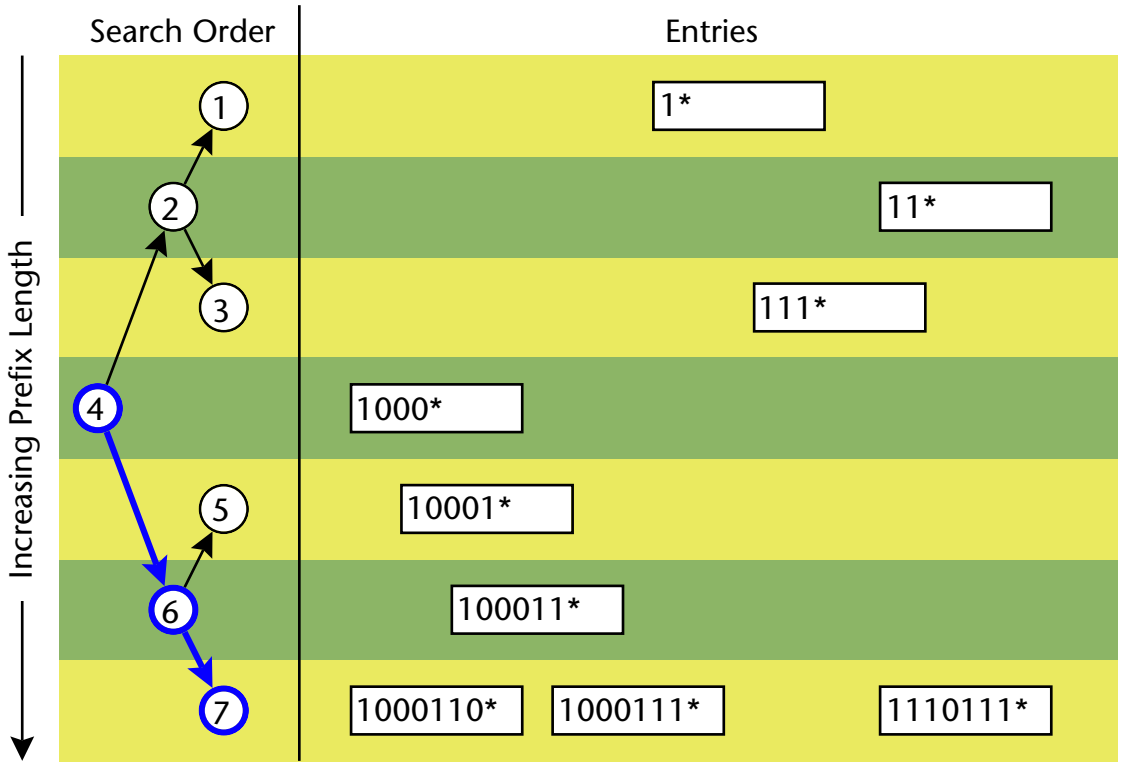
Fast Searching: Basic Idea

- One hash table per prefix length
- Result: Linear search of hash tables



Binary Search on Hash Tables

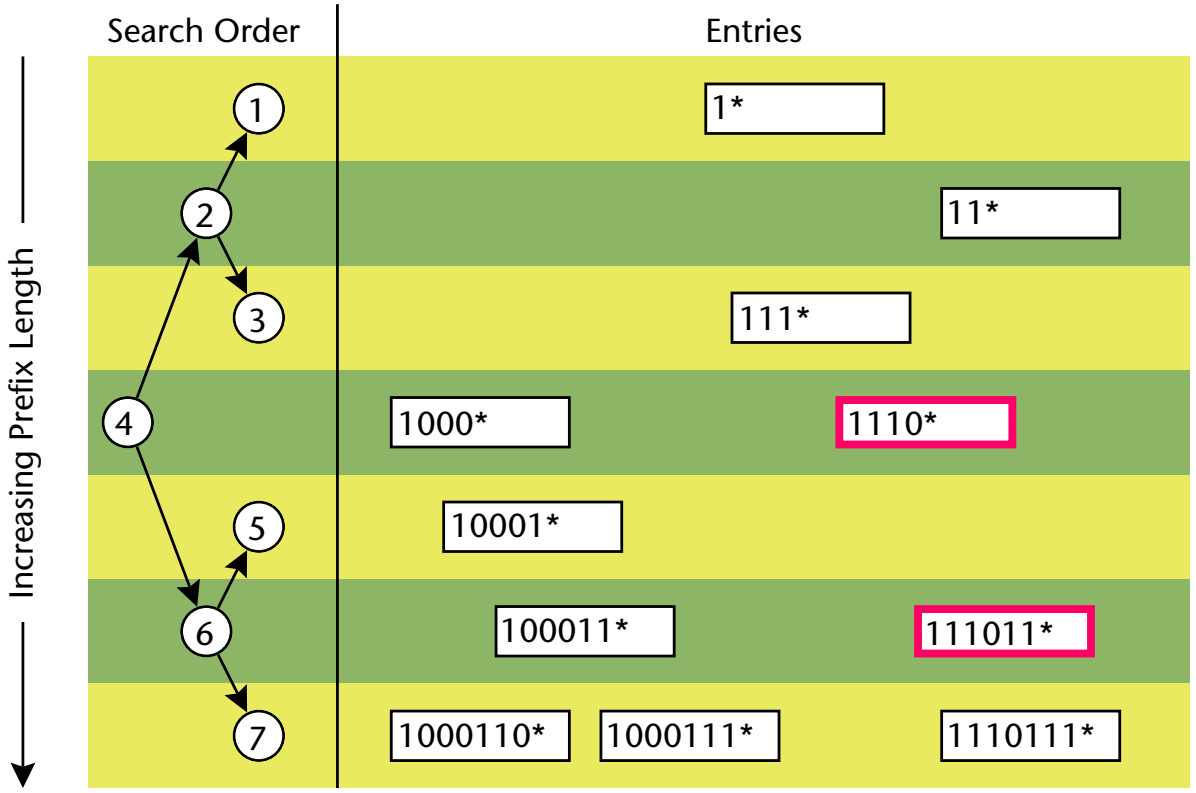
- Binary search needs *less than/greater than* comparison
- Example: 1110111



- Result: More information needed

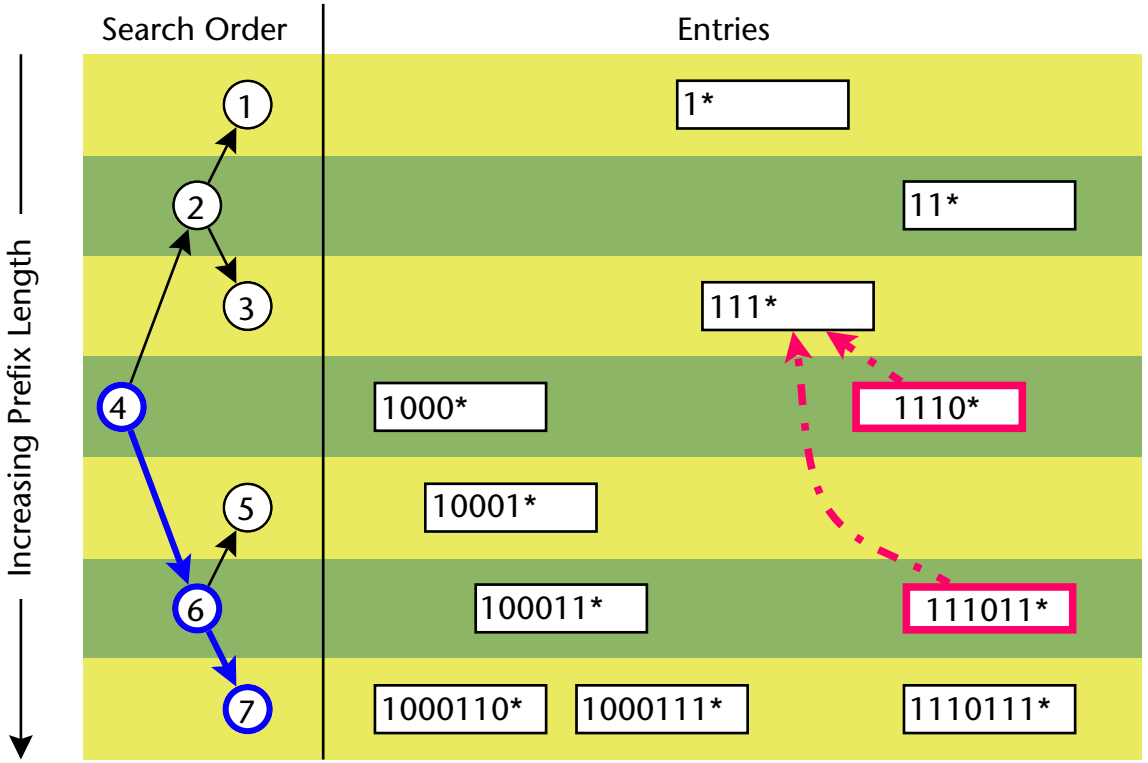
Marker Placement

- Simple approach: At each level above
- Better approach: Only at levels that will be traversed
- Result: Less than O($\log_2$ AddressBits) markers per prefix
- Reality: Much less
- Prefix != entry



Misleading Markers

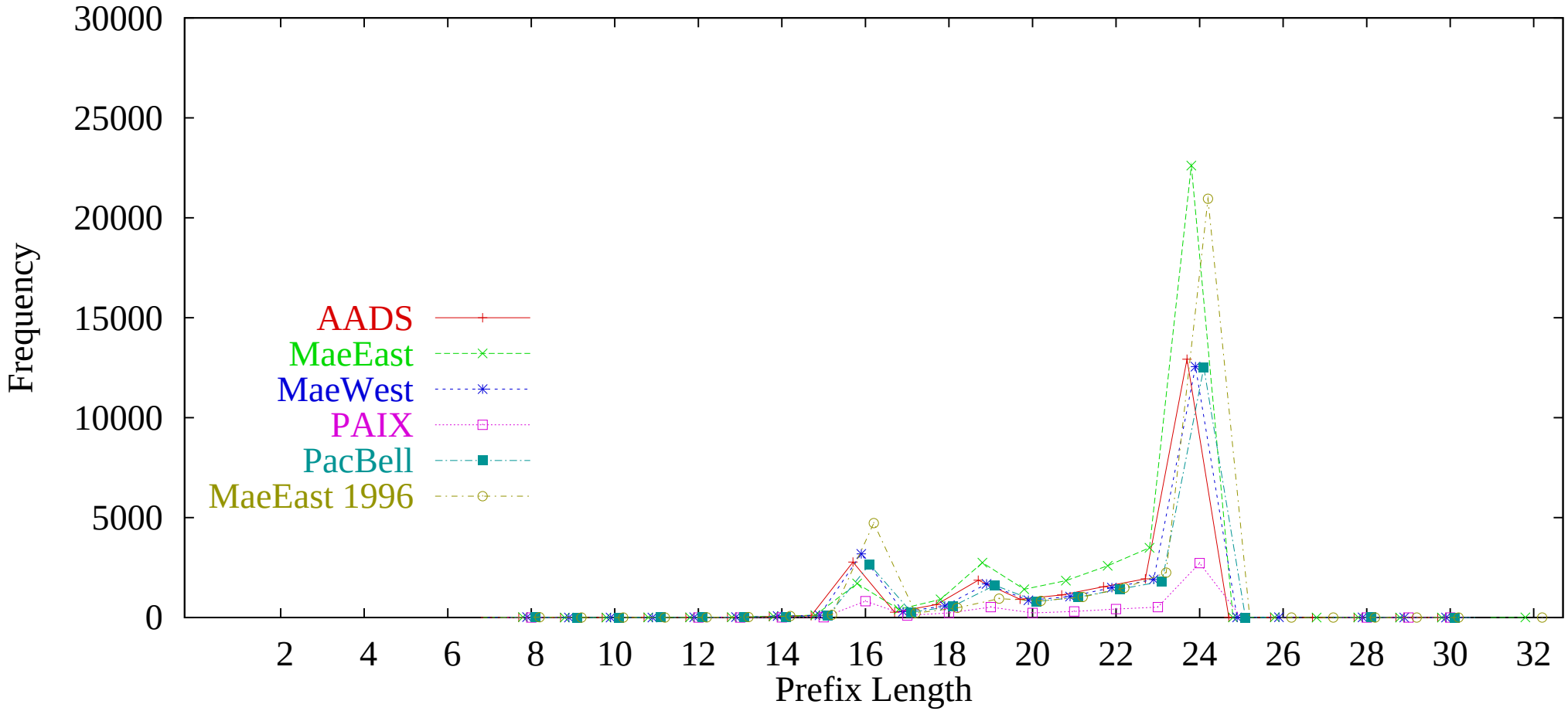
- Markers may require backtracking
- Example: 1110110



- Fix: Precomputation, store BMP in markers

Asymmetric Binary Search

- Backbone routers: Non-uniform prefix length distribution
- Improve average search time

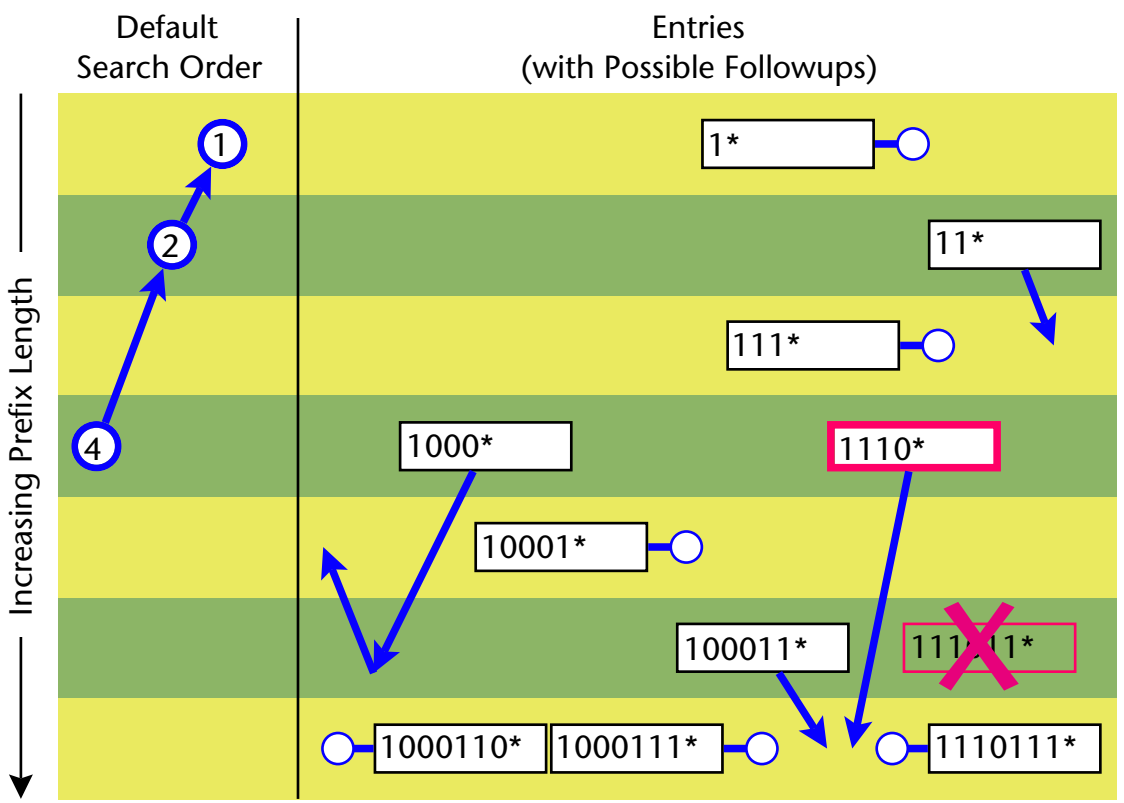


Specializing Further

- Prefix lengths region-dependent?
- Improve after each successful step
- Compactly encode remaining lengths
- Bitmap vs. search tree vs. rope

Mutating Binary Search

- Improve search tree after each match
- Rope Search
  - Only keep track of the skeleton of the tree
  - Reduces search time
  - Reduces markers
  - Ropes point at prefix lengths, not at entries!

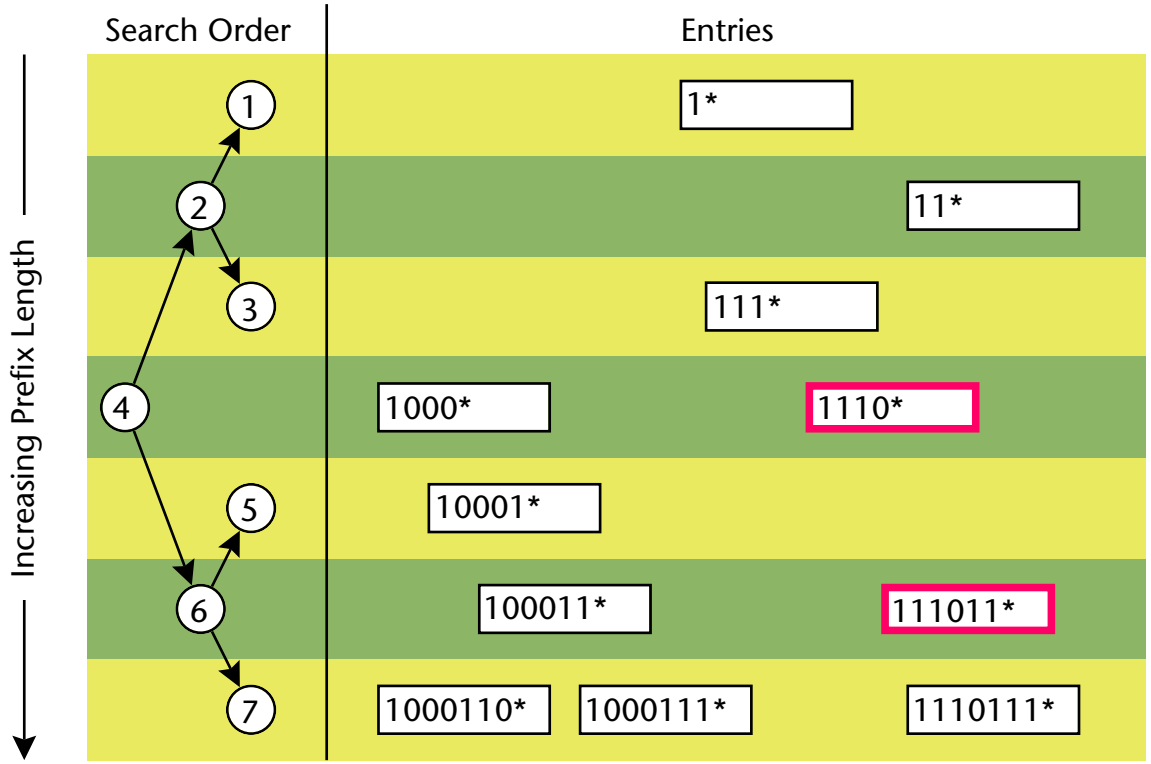


Overview

- Current Routing Techniques
- Binary Search on Prefix Lengths
- Build and Update
  - Build for Binary Search
  - Build for Rope Search
  - Updating Markers' BMP entry
  - Search Tree Restructuring
  - Hash Collisions
- Fast Hashing
- Analysis
- Conclusions

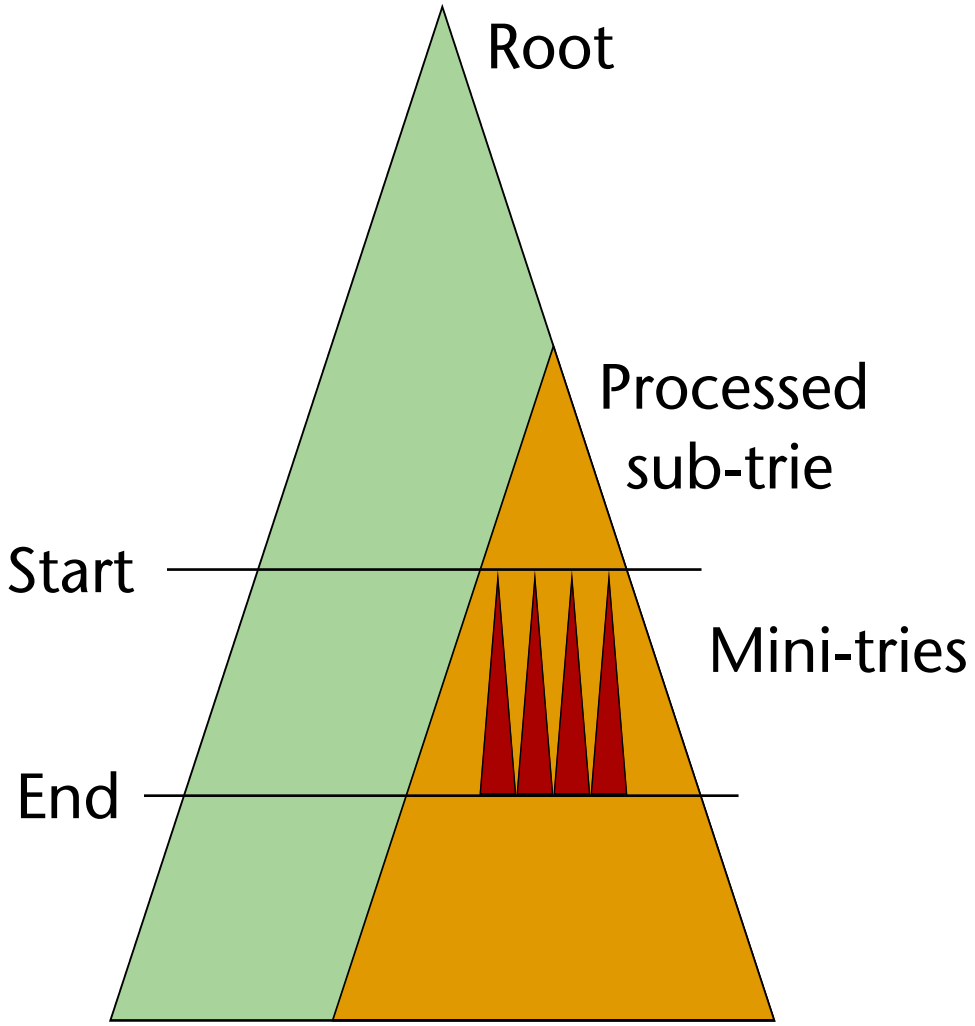
Build for Binary Search

- Insert prefix into appropriate hash table
- Walk binary search tree backwards placing markers



Build for Rope Search

- Bottom-up merging
- Aggregate mini-tries
- Individual mini-tries

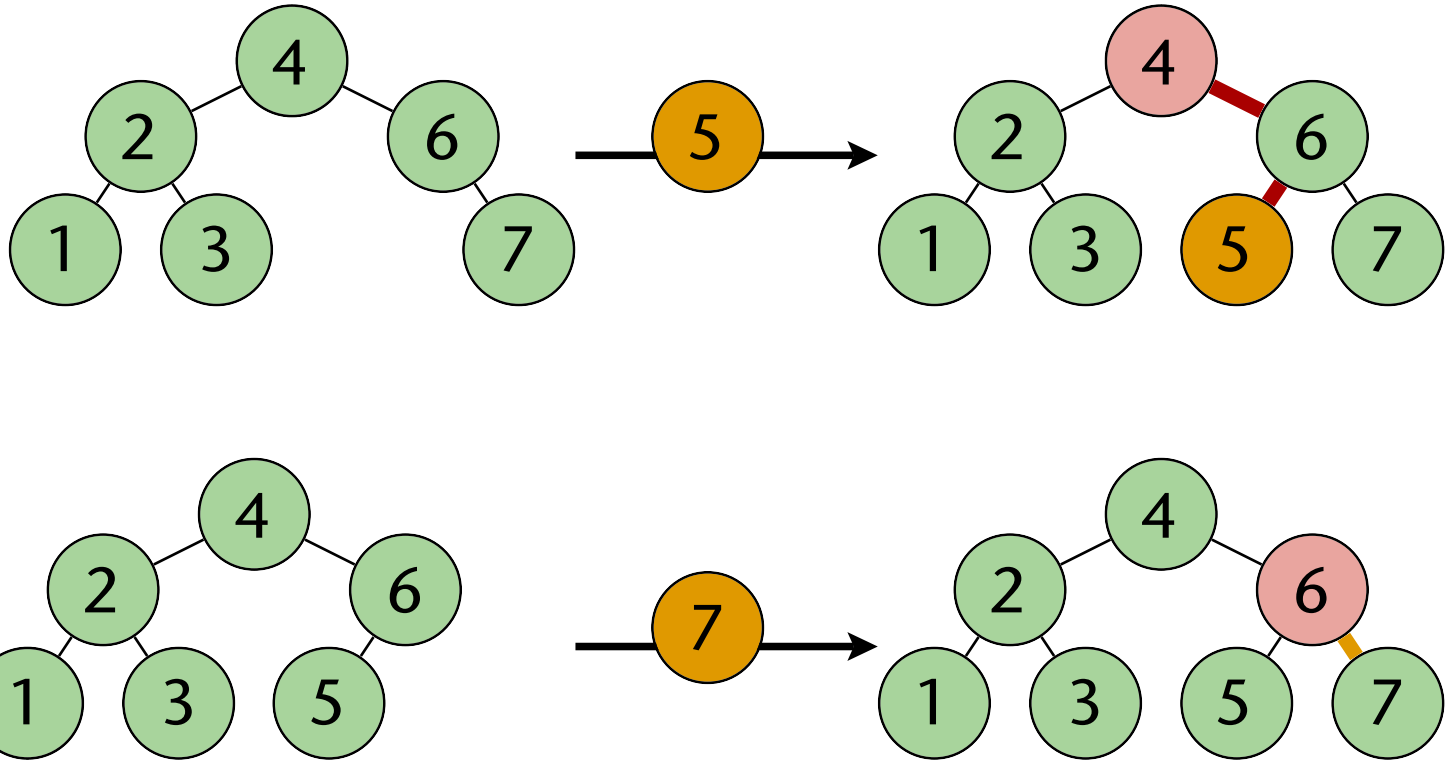


Updating Markers' BMP entry

- Updating can be O(N)
- Solution: Group into $\sqrt{N}$ partitions
  - Significantly improve update times (40,000 $\rightarrow$ 200)
  - Search cost: One memory access
- Generalize to higher roots

Search Tree Restructuring

- Adding a prefix with *new* length
- Only a single rope change

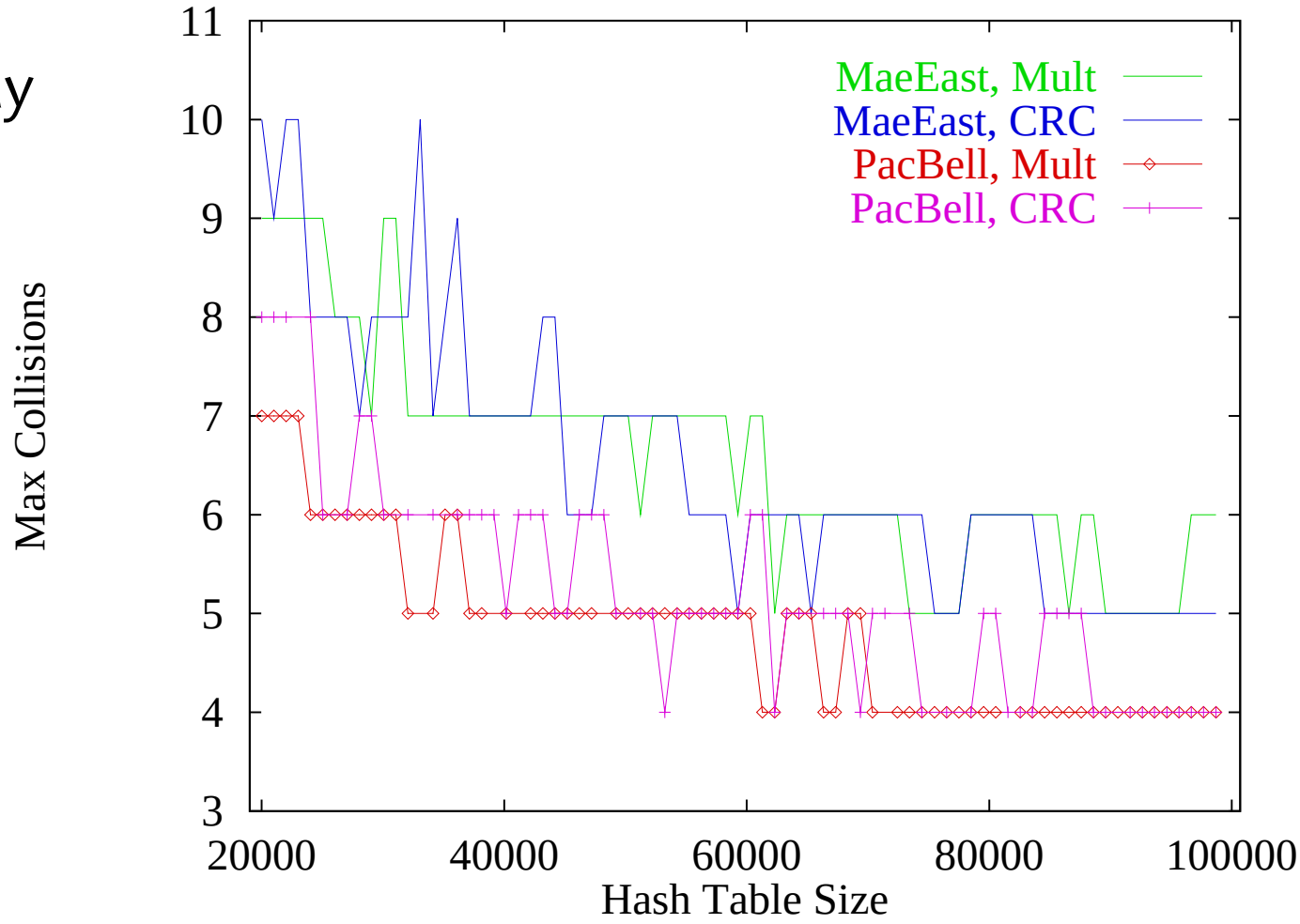


Overview

- Current Routing Techniques
- Binary Search on Prefix Lengths
- Build and Update
- Fast Hashing
  - (Dynamic) Perfect Hashing
    Too expensive for lookup
  - Limiting Collisions
  - Causal Collision Resolution
- Analysis
- Conclusions

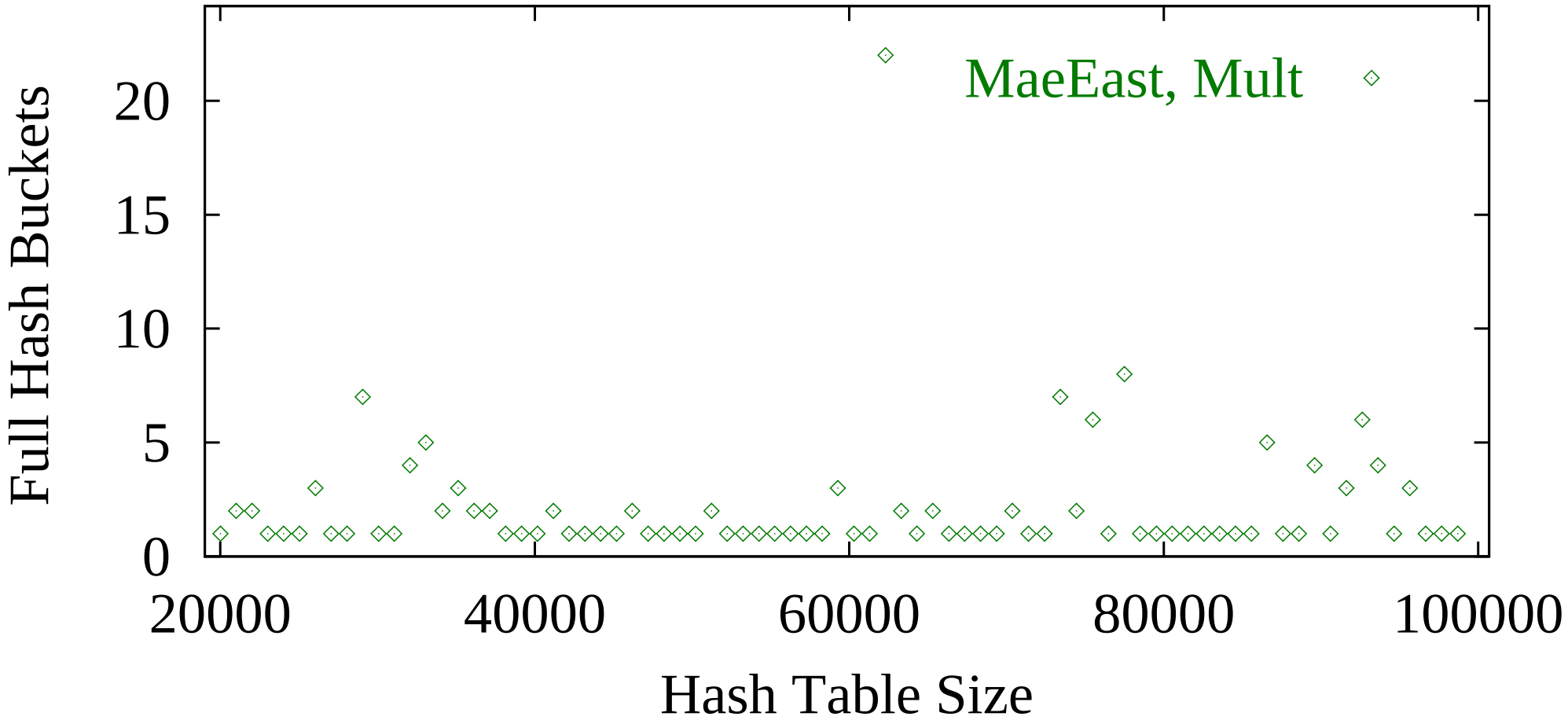
Limiting Collisions

- Hash into buckets (>1 entry)
- Bucket size: Up to cache line size
- Sparse array



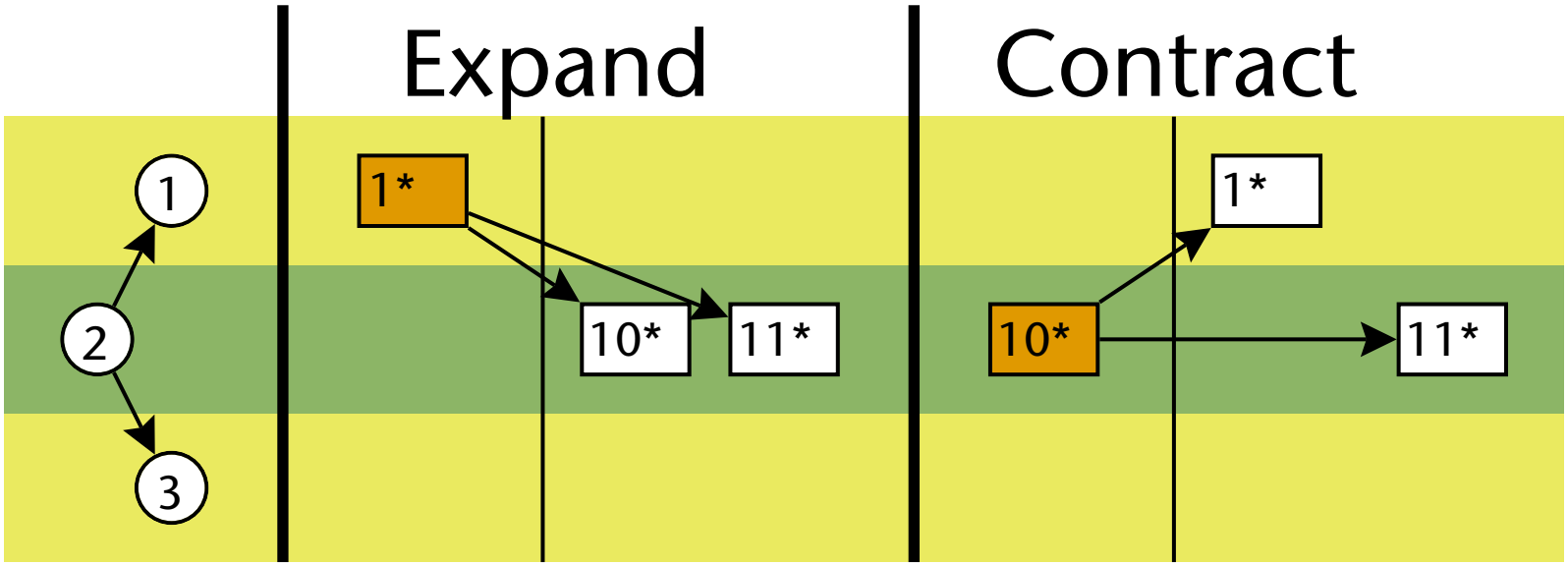
Full Bucket Count

- Observation: Very few buckets require worst case size



Causal Collision Resolution

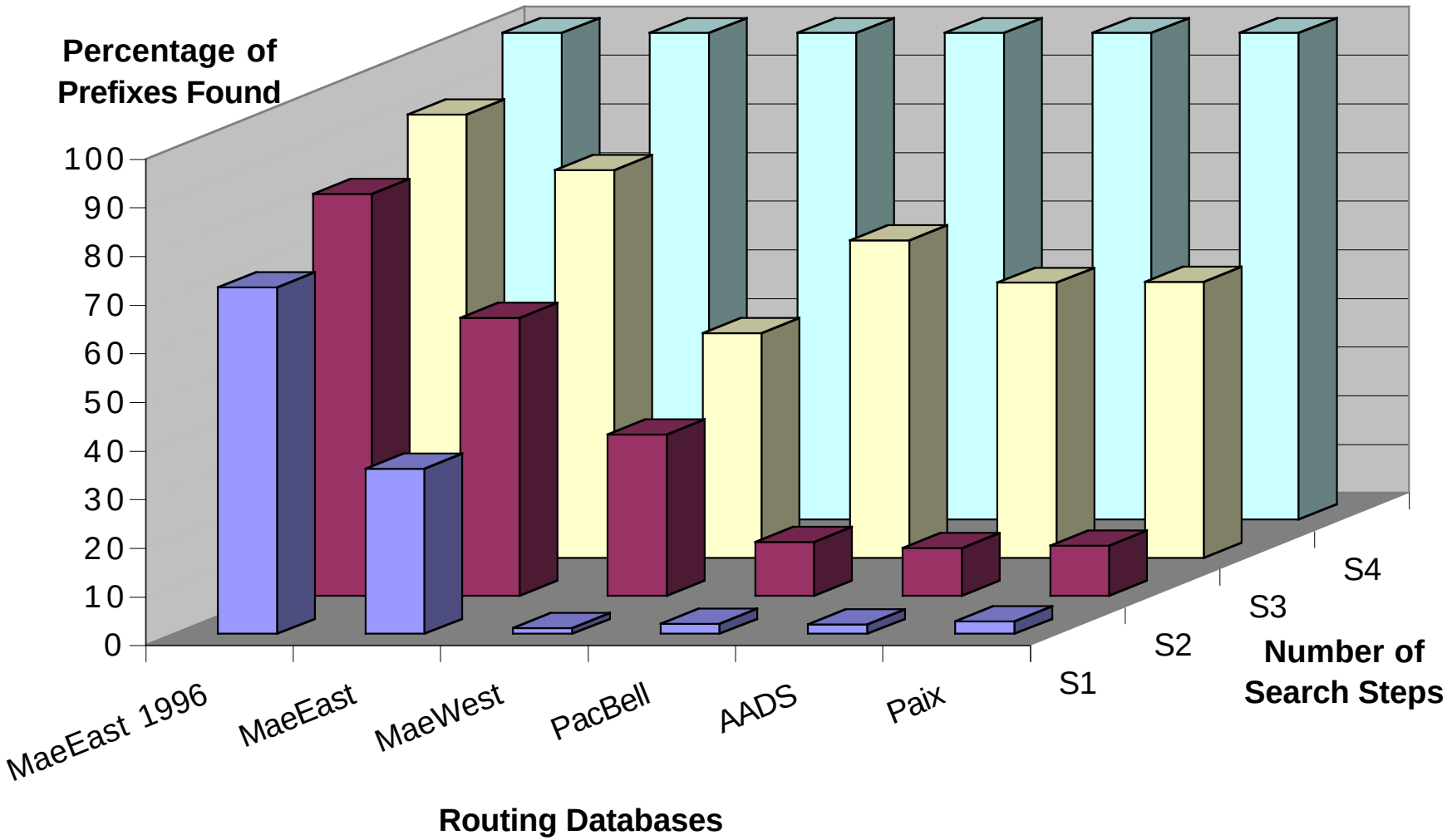
- Goal: Move entries into hash bucket with space
- How: Split it into two entries



Overview

- Current Routing Techniques
- Binary Search on Prefix Lengths
- Build and Update
- Fast Hashing
- Analysis
  - Lookup Speed for IPv4
  - Projections for IPv6
  - 2-D Packet Classification
- Conclusions

Lookup Speed for IPv4

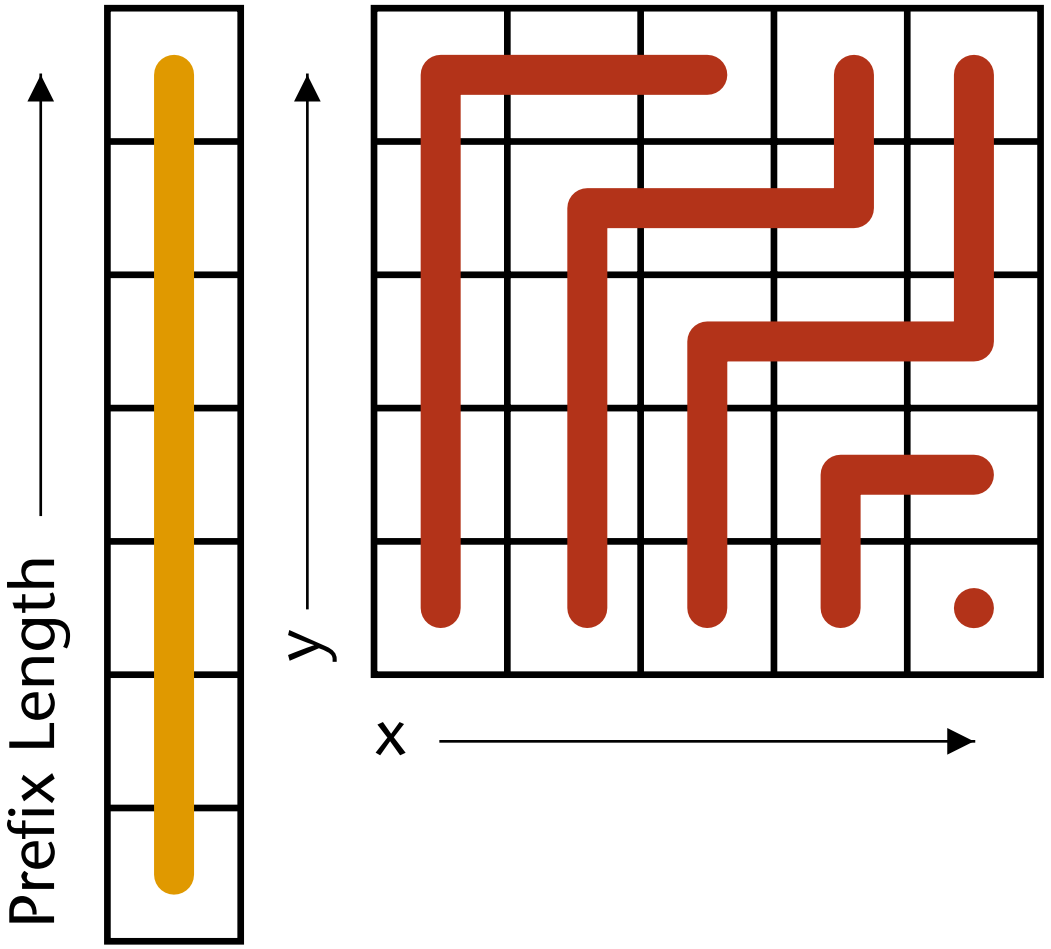


Projections for IPv6

- 4x longer addresses
- More networks and nodes
- Hope that backbone routers will be able to use small routing tables $\rightarrow$ suboptimal routing
- Hierarchy boundaries at more prefix lengths
- Policy routing will still force ISPs to have bigger routing tables
- For our approach, only 2 memory lookups more

2-D Packet Classification

- Source and destination prefixes
- ‘‘Winding’’ paths of increasing specificity
- O(W log W)
- Sparse Matrizes



Conclusions

- Fast, space efficient, and scalable lookup algorithm
- New class of search algorithms
- Fast update
- No need for hardware, yet cheap hardware possible
- No need to proliferate protocol changes

Extensions

- Extend to two-/multi-dimensional packet classification
  - Preliminary results available
- Other uses
  - Flexible memory management
  - Access control lists
  - Substring searching (databases)

Future Work

- Light-weight protocols for secure group communication
- Secure distributed storage
- Distributed key storage
- Protocols for bandwidth fairness enforcement