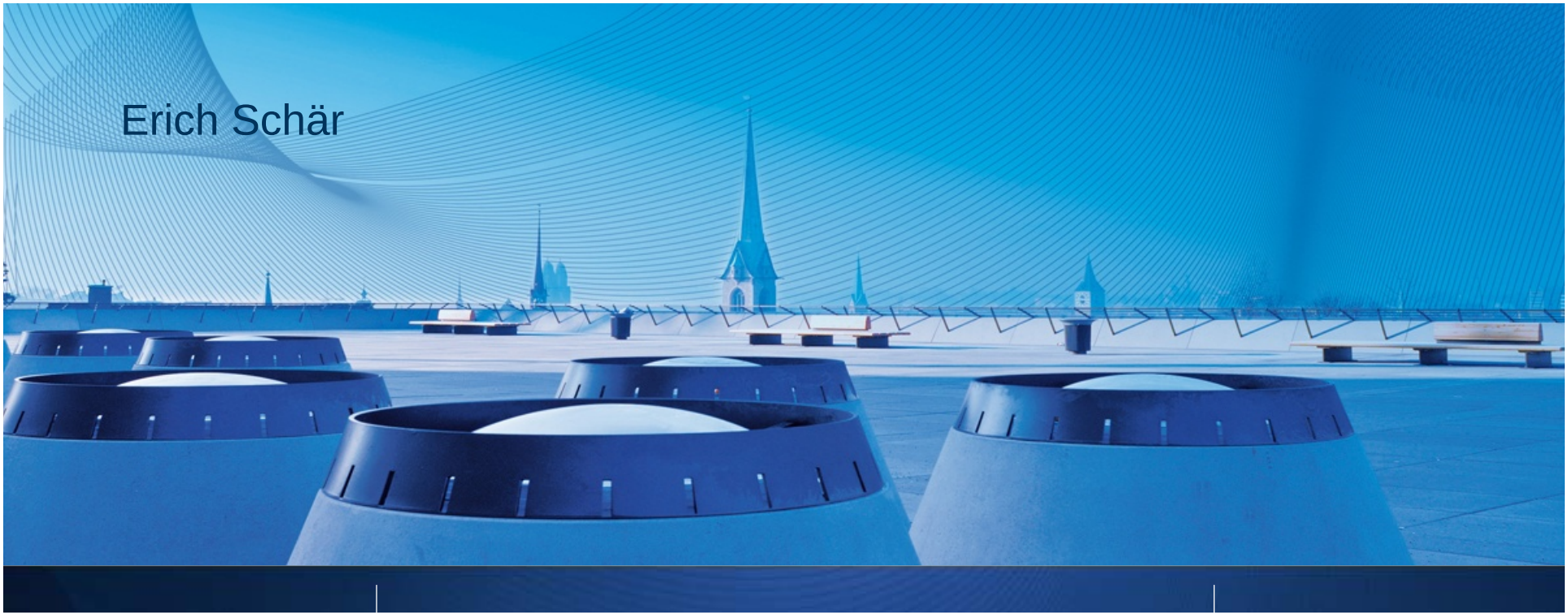
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group

Internet Routing

Seminar in Distributed Computing
with papers chosen by Prof. Bernhard Plattner



Erich Schär

Challenges for Internet Routing

- Internet initially designed as a small network for researchers

- More and more users
  - Cheap computers
  - Mobile devices

Increasing Internet Traffic

- 160 TB per second today
  - Growth rates per year 50% to 100%
- Drivers
  - Video applications like youtube
  - P2P filesharing
  - Webradio
  - Voice applications like Skype / TeamSpeak
- More traffic from more endpoints
  - Throughput hard to handle efficiently

Packet Scheduling

- Idea has been around for a long time

Possible applications:

- Give high priority to important packages and delay less important packages

IPv6

- IPv4 address space limited
- IPv6 also offers nice features
  - Protocol optimized for routers
  - IPsec
  - Multicast
- Proposed in 1995
- Still not widely used in practice

Overview

1. Scalable High Speed IP Routing Lookups
2. A Software Architecture for Routers

Scalable High Speed IP Routing Lookups

Marcel Waldvogel, George Varghese,
Jon Turner, Bernhard Plattner

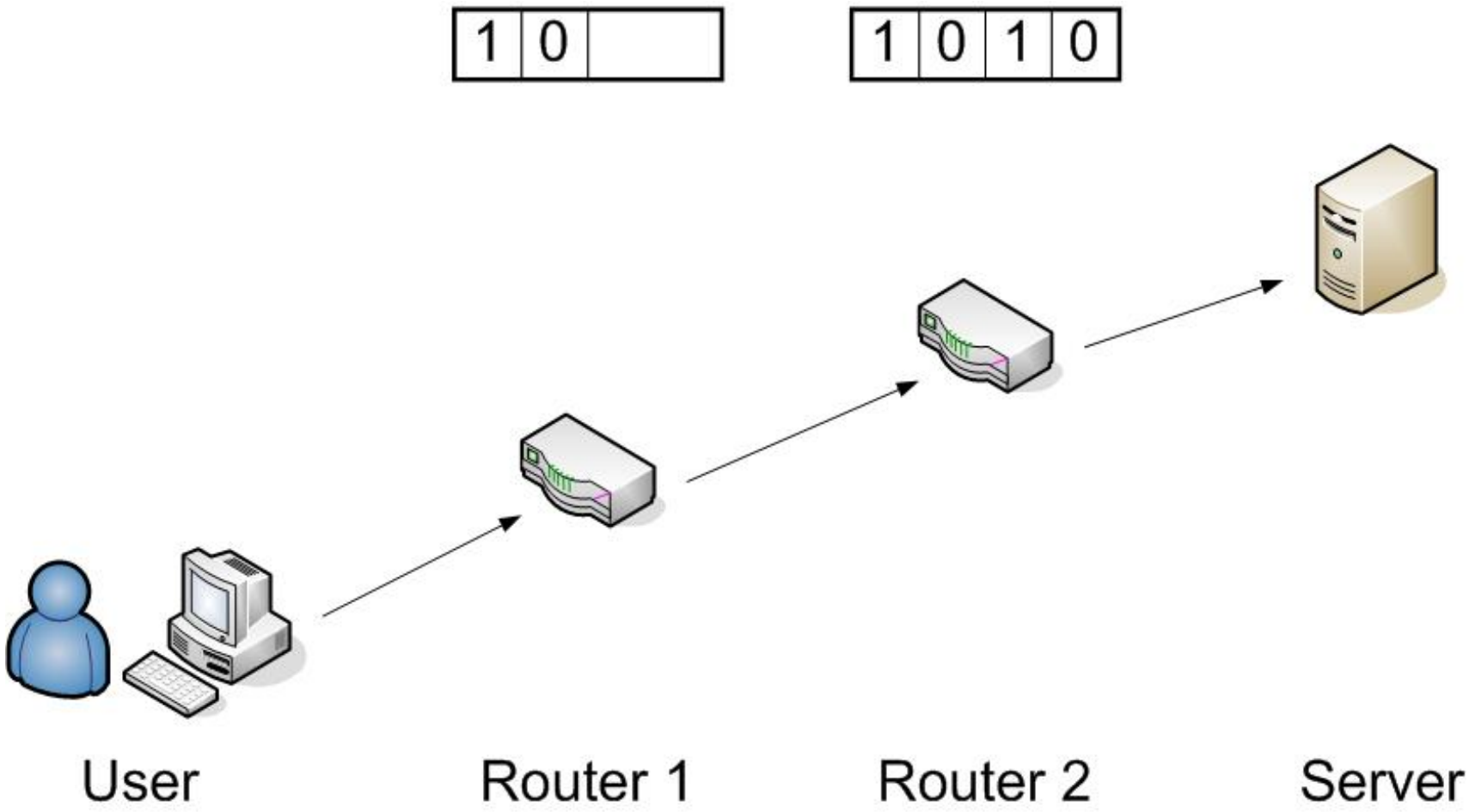
In ACM SIGCOMM Proceedings, 1997

Motivation

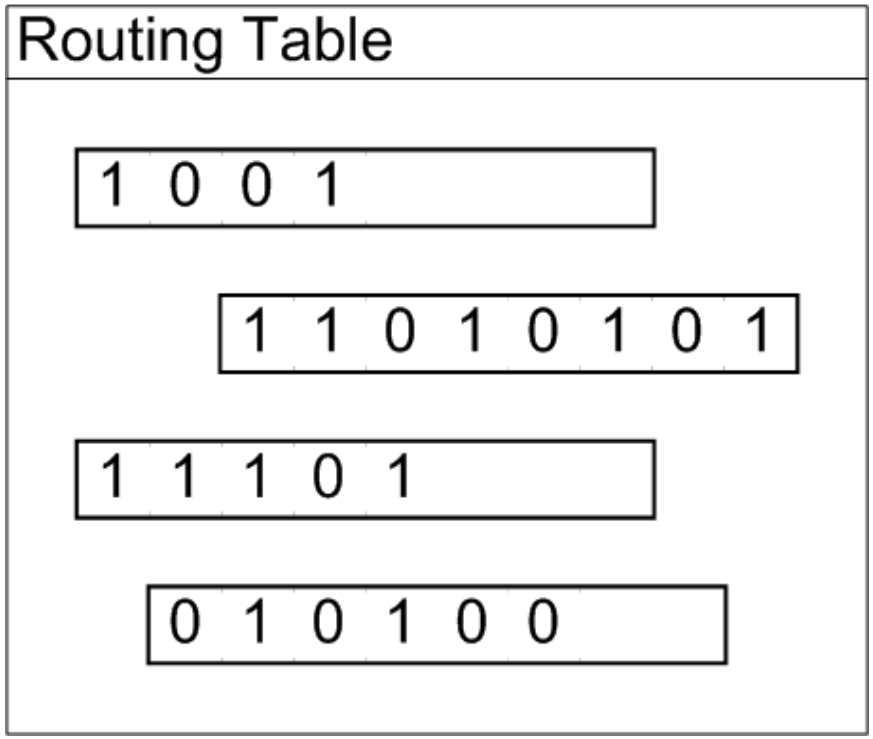
Three limiting Factors:

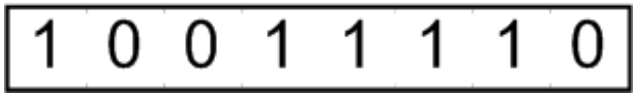
- Link Speed
- Router Data Throughput
- Packet Forwarding Rates

Refresher: Routing

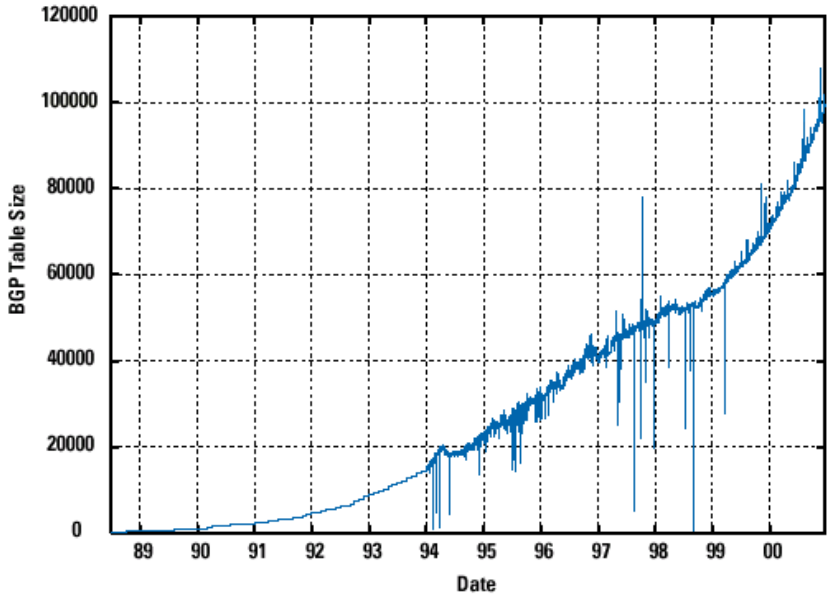


Refresher: Longest Prefix Matching





Increasing Routing Table Size

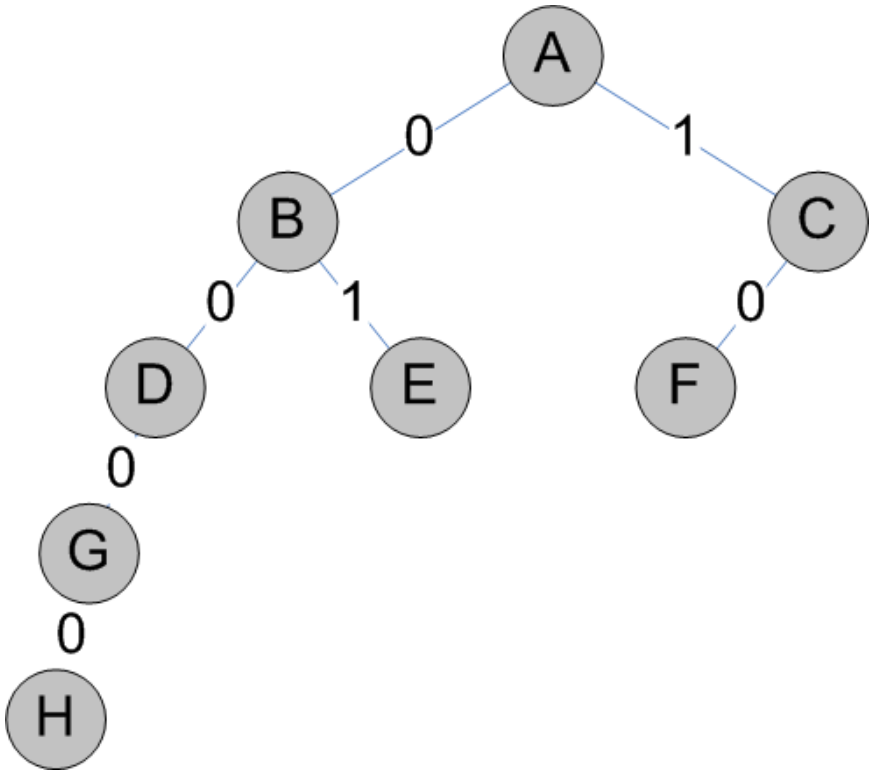


- 140 000 to 160 000 entries in the routing table today
- Increasing

Existing Approaches

- Trie based schemes
- Hardware solutions
- Protocol based solutions
- Caching

Trie



Example trie for longest prefixes:

- 0 0 0 0
- 0 1
- 1 0

Example:

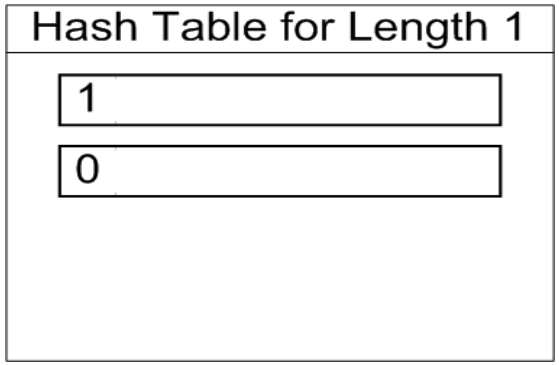
- Search longest prefix for IP 01001111

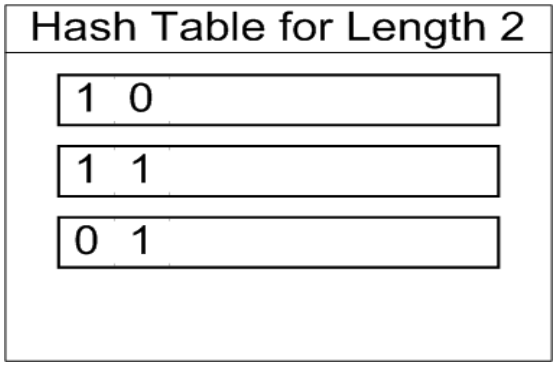
Runtime:

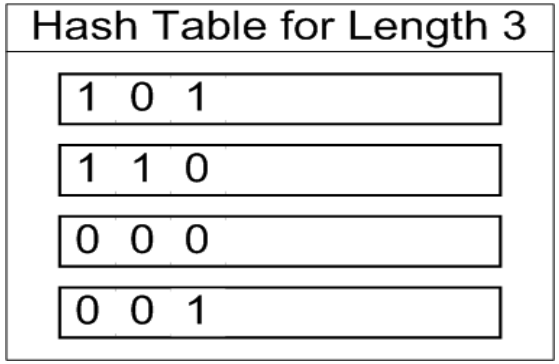
- Worst case runtime linear in length of address size
  O(w)

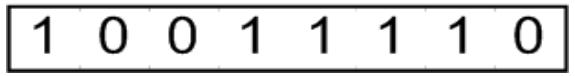
Algorithm: Hashing

- Needs a *precise* Key









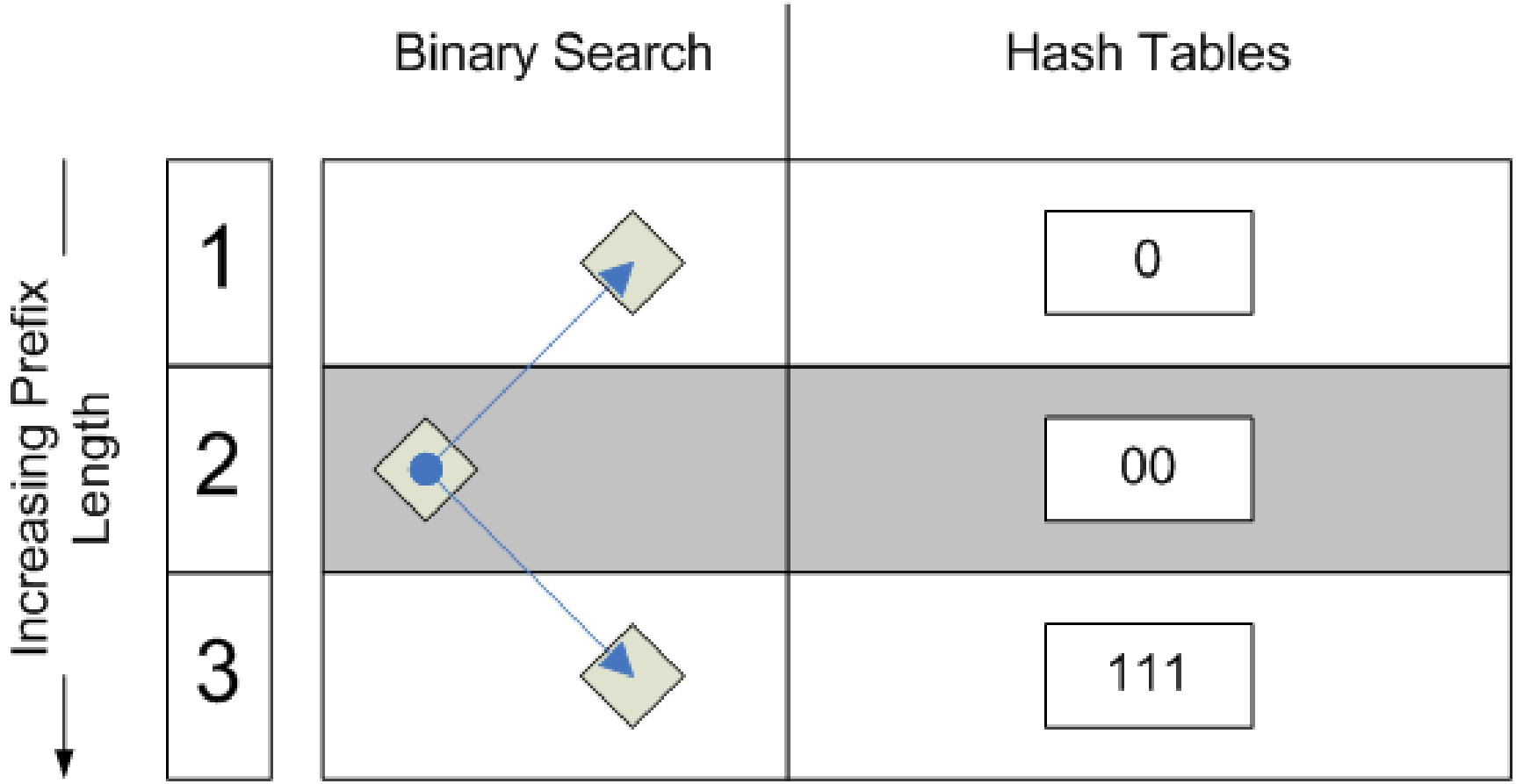
Algorithm: Hashing

- Search every hash table in constant time
- Worst case: have to search every hash table
- → O(w)

- Can we do better?

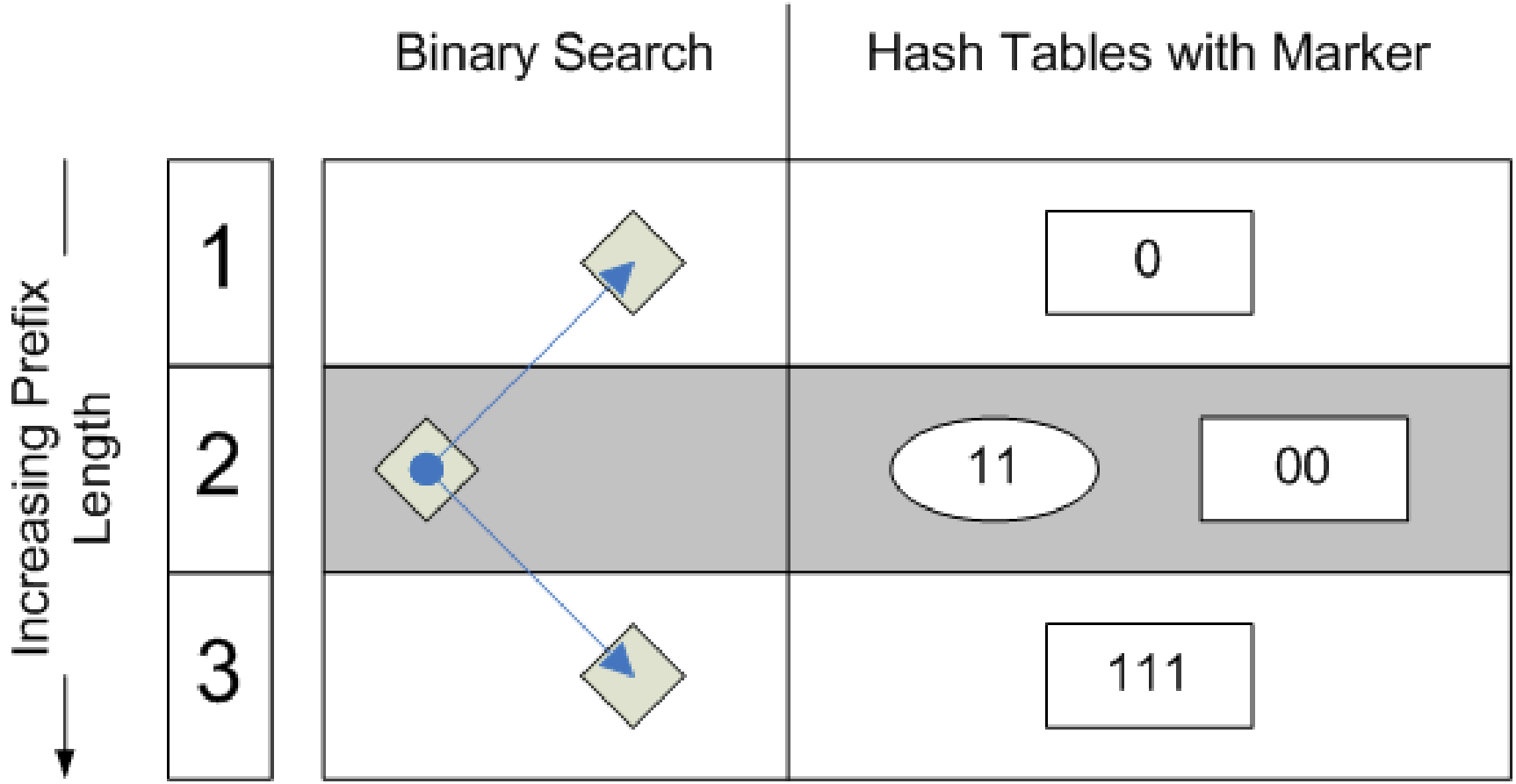
Binary Search

1 1 1 1 0 0 1 0 ?



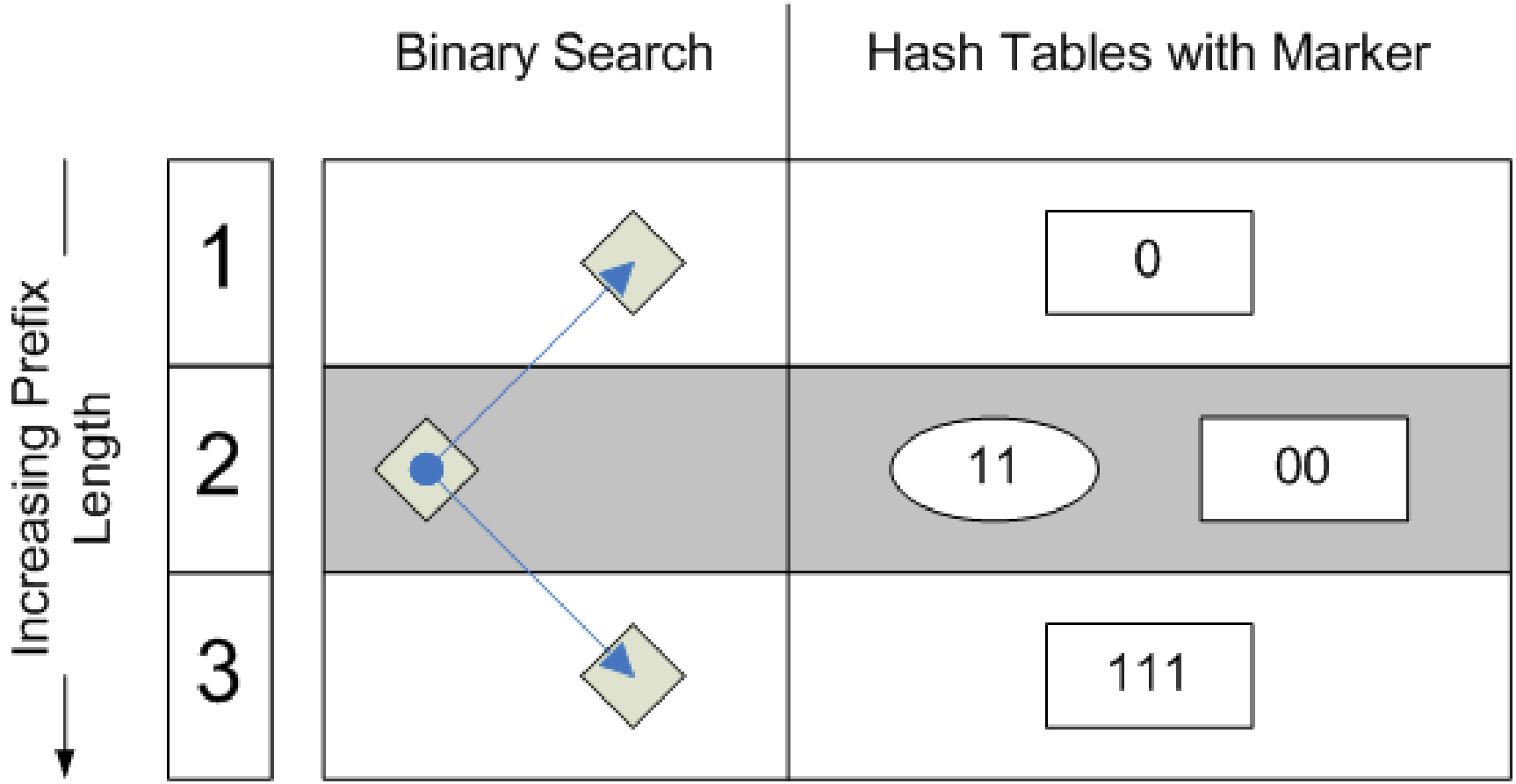
Binary Search

1 1 1 1 0 0 1 0 ?



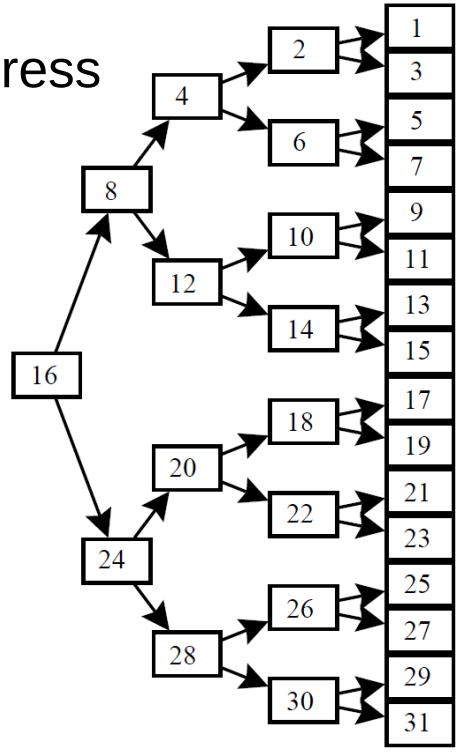
Binary Search

1 1 0 1 0 0 1 0 ?



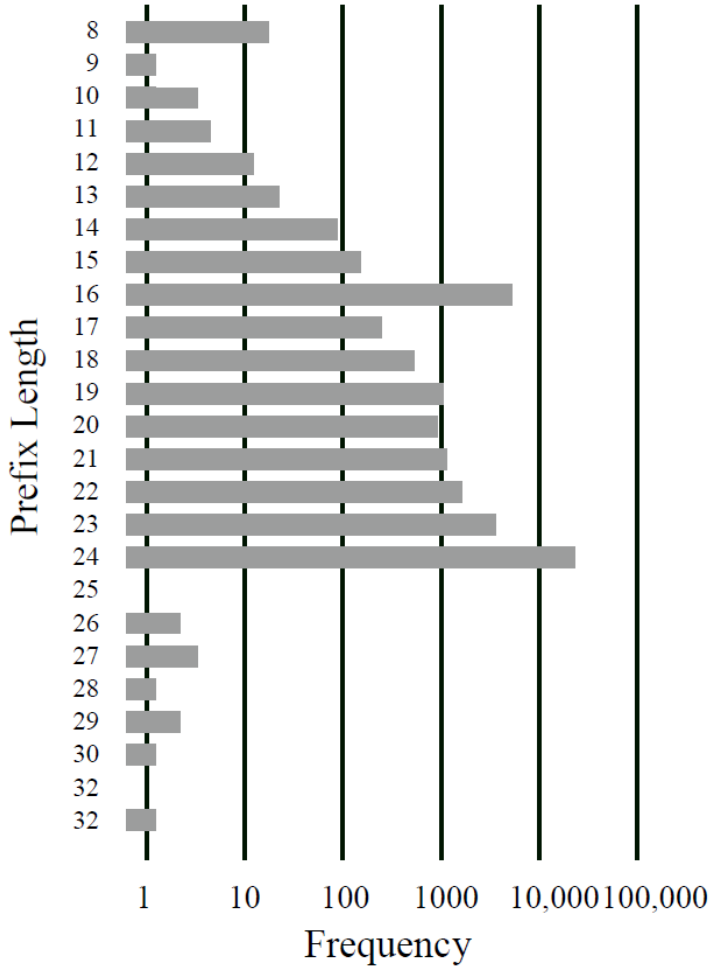
Runtime Analysis

- Every hash look-up in constant time
- At most O (log (w)) hash look-ups
  - At most 7 hash computation for a 128 bit IPv6 Address

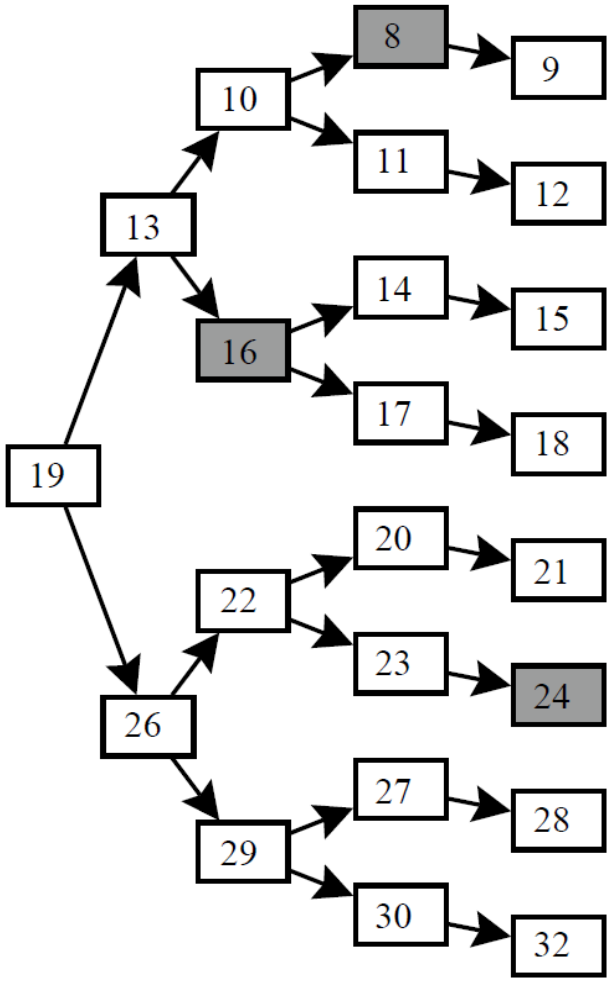


Optimizations

- Adopt to actual data to improve average case behavior
- Some prefix lengths looked up more often
- Some prefix lengths never looked up
- Optimizations do not improve worst case behavior



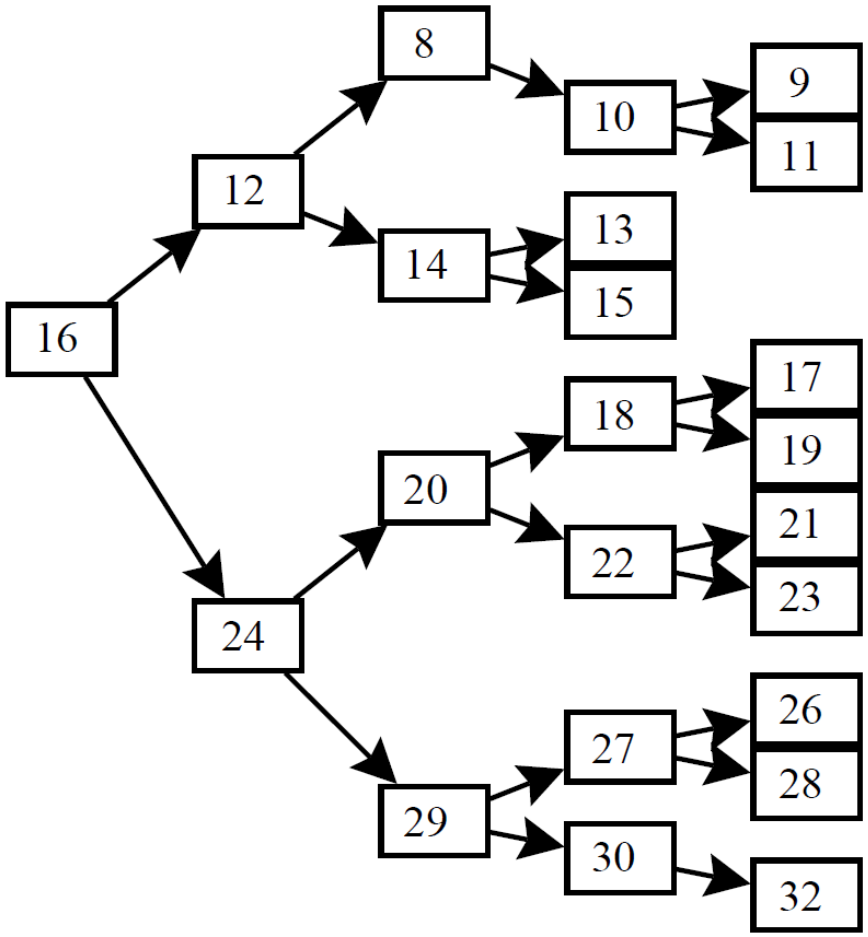
Idea: Remove "empty" Hash Tables from the Search Tree



- Remove not used hash tables from the search tree
- Rebalance the tree

- Average case gets worse, since frequently looked up hash tables are further down the tree

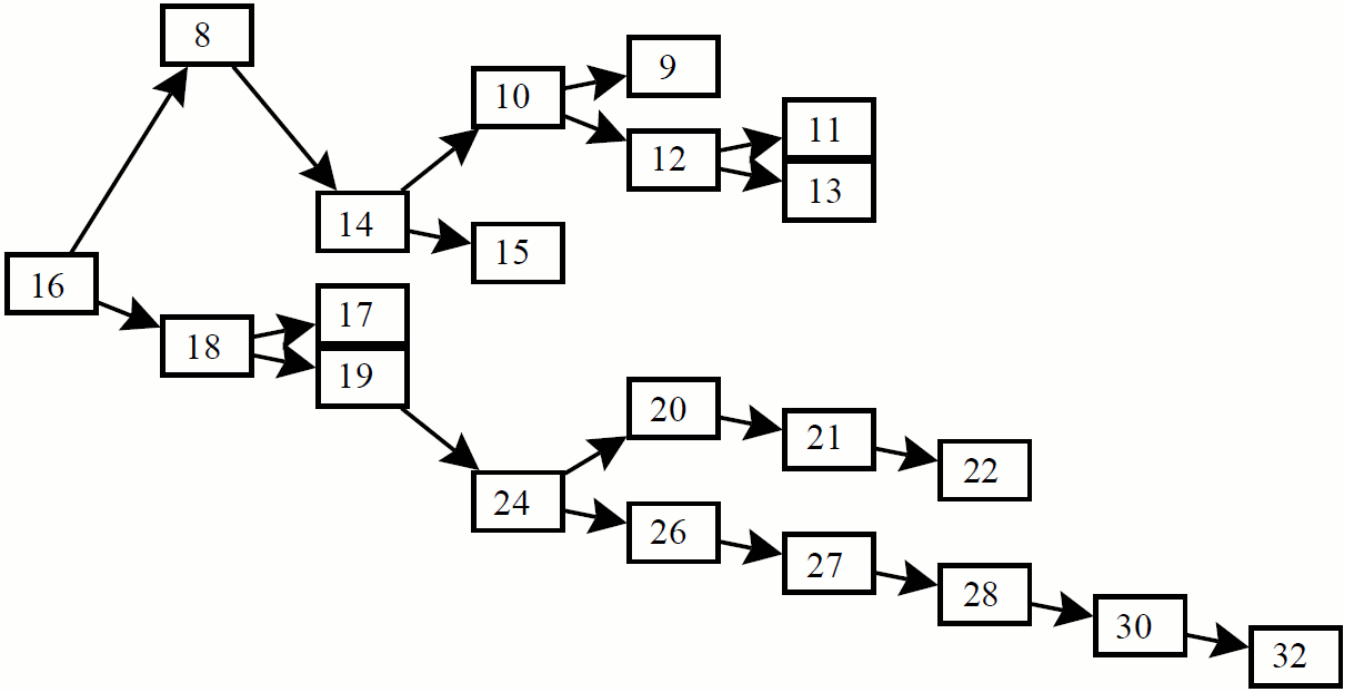
Changing the Tree



- Reorder the tree to have frequently accessed hash tables on top
- Improve average search time
- Worst case still O(log(w))

Changing the Tree

- Even better average case
- Worst case O(w)



Mutating Binary Search

- Observation: For a given matching prefix of length 16, the longest matching prefix usually has only very few possible lengths.

- E.g. if we do a look up for 10111100.01011101.00011101.10001110 we find a matching prefix 10111100.01011101

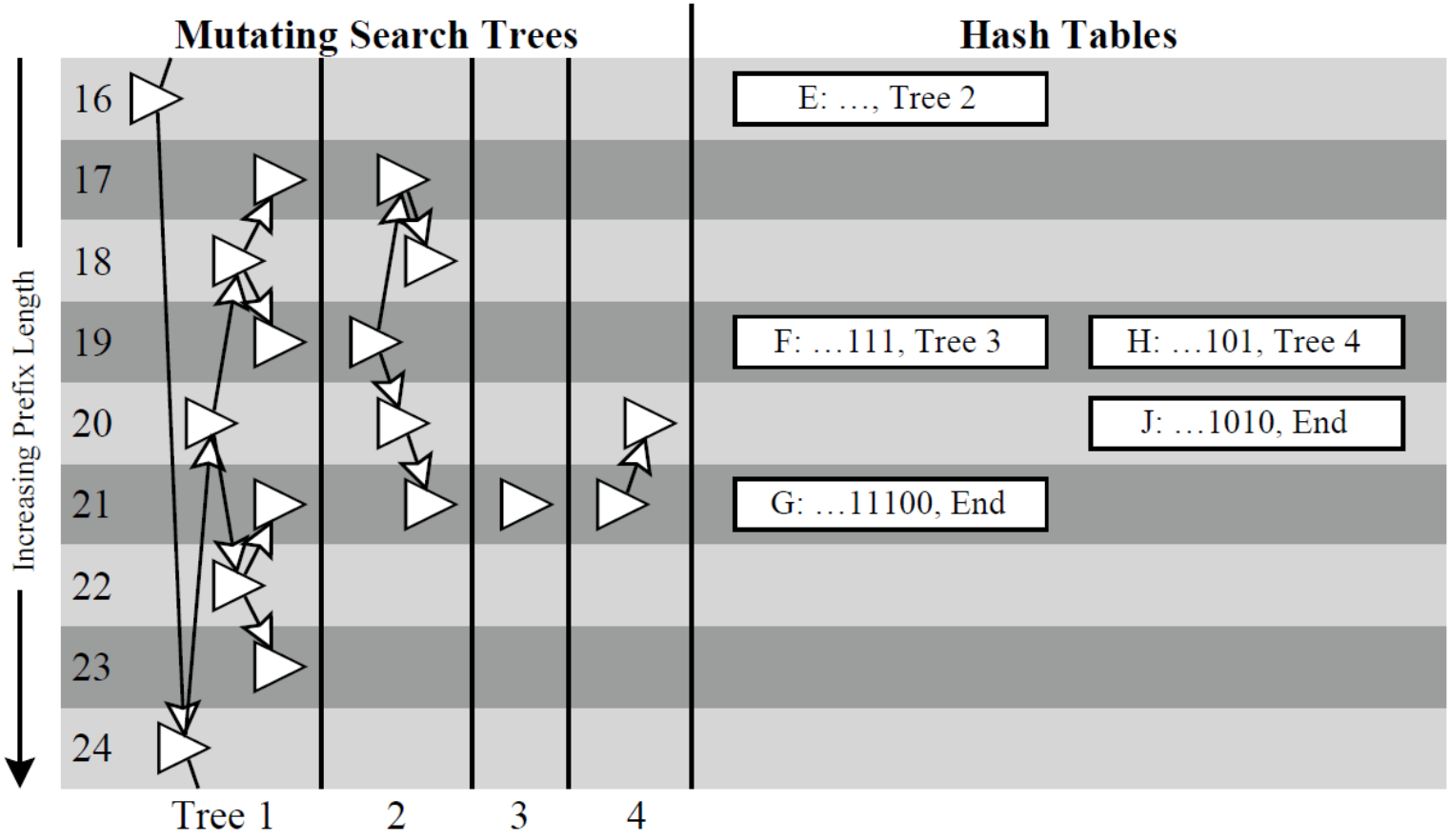
- We know from statistics, that the longest matching prefix for any IP address starting with the prefix 10111100.01011101 either has length 24 or 26.

Mutating Binary Search

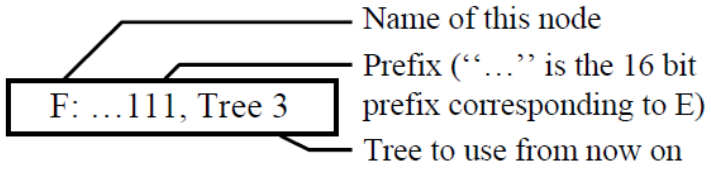
- Use this information to adopt search based on the matching 16 bit prefix.

- Store information on where to continue search in markers.

Mutating Binary Search



Structure of Hash Table Entry:



Mutating Binary Search

- Typically for a given prefix of length 16 only very few possible length of the longest matching prefix.

Distinct Lengths	Frequency
1	4977
2	608
3	365
4	249
5	165
6	118
7	78
8	46
9	35
10	15
11	9
12	3

Results

- Algorithm for longest prefix matching
- O(log(w))
- Evaluation (on a 200MHz Pentium Pro)
  - 80 ns for IPv4 lookup
  - 200 ns for IPv6 lookup
- Router throughput no longer limited by the speed of the lookup

Router Plugins
A Software Architecture for Next Generation Routers

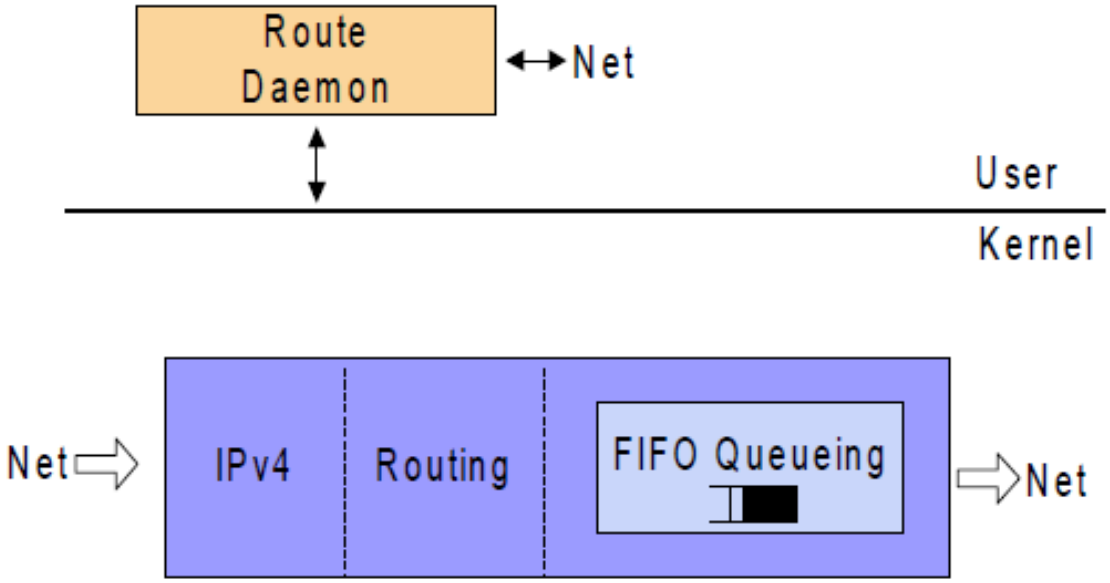
Dan Decasper, Zubin Dittia, Guru Parulkar,
and Bernhard Plattner

In ACM SIGCOMM Proceedings, 1998

Today's Routers

Routers today:

- Monolithic operating systems
- Not easily upgradable and extensible

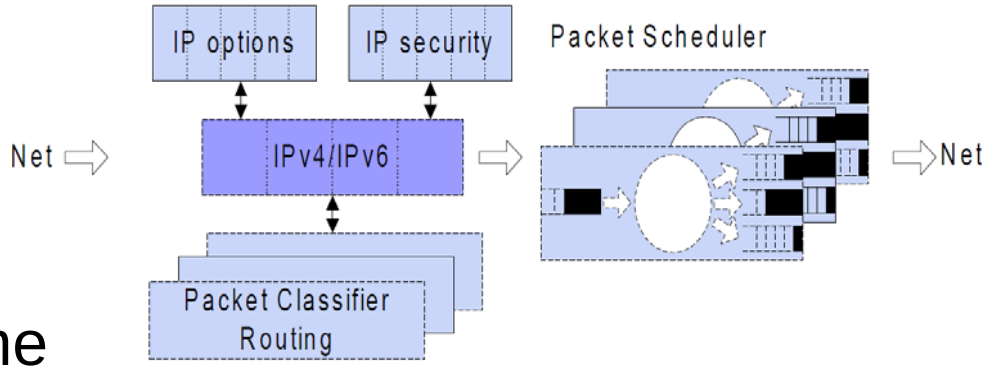


Design Goals

- Modularity
- Extensibility
  - New Protocols (IPv6)
  - Security
- Flexibility
- Performance

Plugins

- Modules
  - Packet scheduler
  - Longest prefix matching
- Add and configure at runtime
- Run in kernel space



Gates

- Possible entry points for plugins in the IP core

Examples:

- IPV6 option processing
- IP security
- Packet scheduling
- Best-matching prefix algorithm

Flow / Filter

- Filter used to bind packet flow to a plugin instance

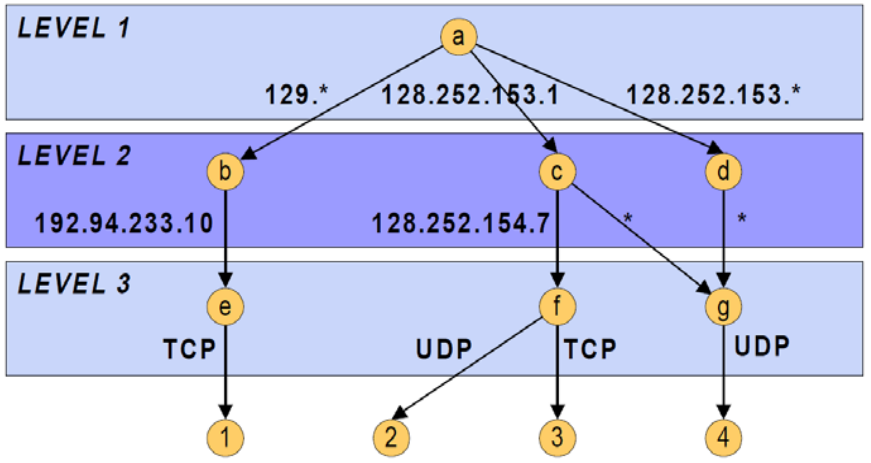
- Six-tuple
  - Source address
  - Destination address
  - Protocol
  - Source port
  - Destination port
  - Incoming interface

- Wildcards allowed

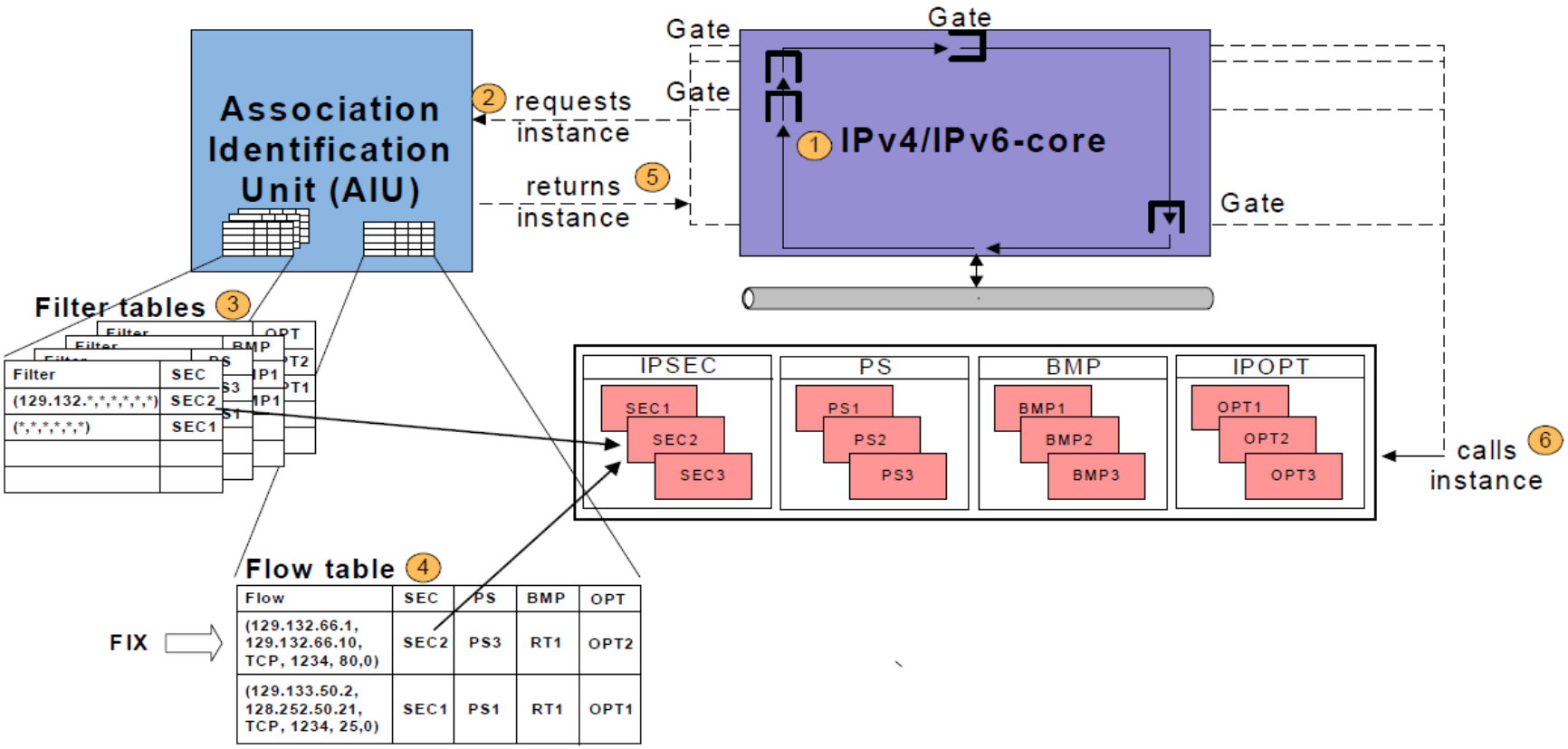
Association Identification Unit

- Binding of plugin instances and filters
- Performance critical
- Directed acyclic graph (DAG)

#	Source Address	Destination Address	Protocol
1	129.*	192.94.233.10	TCP
2	128.252.153.1	128.252.153.7	UDP
3	128.252.153.1	128.252.153.7	TCP
4	128.252.153.*	*	UDP



Example


Association Identification Unit (AIU)
2 requests instance
returns 5 instance
Gate
Gate
Gate
Gate
1 IPv4/IPv6-core
Filter tables 3
Filter | SEC
(129.132.*,*,*,*,*,*) | SEC2
(*,*,*,*,*,*) | SEC1
IPSEC
SEC1
SEC2
SEC3
PS
PS1
PS2
PS3
BMP
BMP1
BMP2
BMP3
IPOPT
OPT1
OPT2
OPT3
calls 6 instance
Flow table 4
FIX
Flow | SEC | PS | BMP | OPT
(129.132.66.1, 129.132.66.10, TCP, 1234, 80,0) | SEC2 | PS3 | RT1 | OPT2
(129.133.50.2, 128.252.50.21, TCP, 1234, 25,0) | SEC1 | PS1 | RT1 | OPT1

Results

- Implemented in NetBSD
- Efficient (only 8% slower than best effort)
- Extensible
- Flexible

Conclusions

- Short overview over some important routing problems
  - Increasing number of users
  - large traffic
  - Inflexible routers
- Algorithm for faster routing
  - A fast algorithm with good asymptotic runtime
  - Various optimizations to typical lookups
- Powerful software routing architecture
  - Can easily be adapted to new problems with routing
  - Very efficient

Questions?