CP-Trie: Cumulative PopCount based Trie for IPv6 Routing Table Lookup in Software and ASIC

MD Iftakharul Islam, Javed I Khan

Department of Computer Science
Kent State University
Kent, OH, USA.

IEEE HPSR, 2021

Outline

Routing table lookup

- Also known as ”IP lookup” or ”FIB lookup”
- It involves performing the Longest prefix match (LPM) in order to find the next-hop.

Table: An example routing table

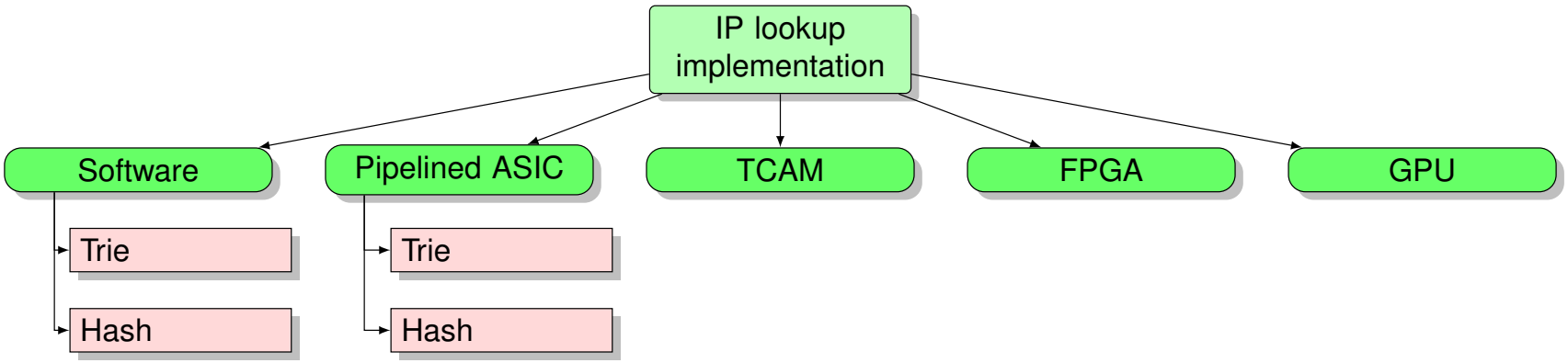
Route	Prefix	Next-hop
r1	200a:410:8000::/40	2
r2	200a:410:8080::/44	3
r3	200a:410:8000:702::/64	1
r4	200a:410:8000:702::0df/128	2

- **Lookup**(200a:410:8088:500::300) $\Longrightarrow$ $\{r1, r2\}$ $\Longrightarrow$ 3
- IP lookup is generally performed using **Trie** or **Hash** based algorithms. Here, Trie based algorithm is our main focus.

Challenges of IP lookup

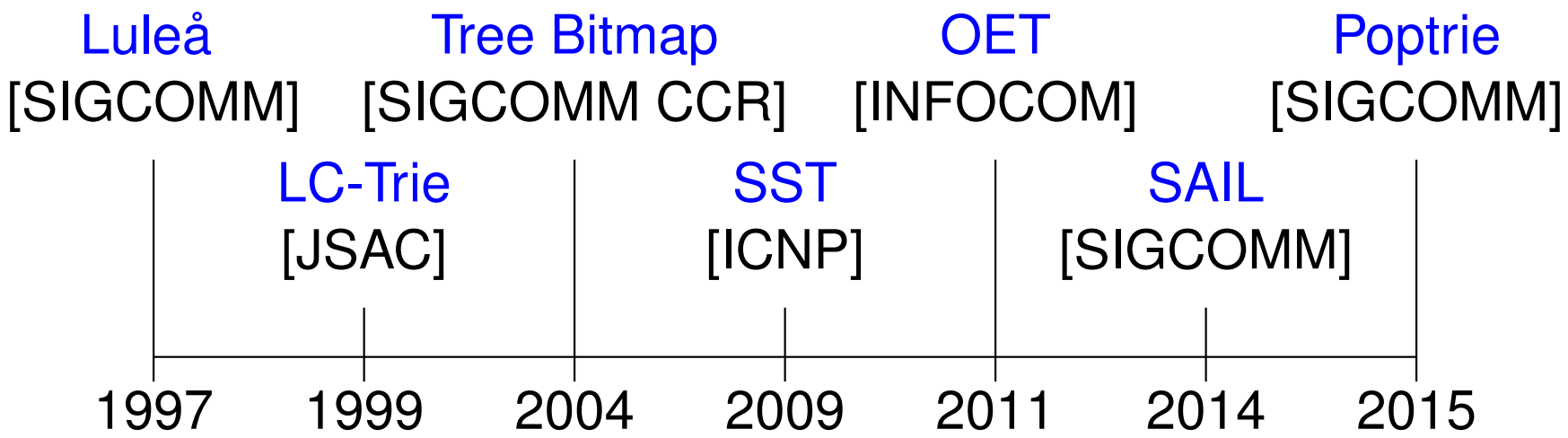
- Extremely high throughput needed.
  - 4.8 Tb/s Brodcom Jericho2 chip (used in many core routers) can forward 2 billion packets per second.
- Forwarding table of a core router can be extremely large
  - Already exceeded $100k+$ IPv6 entries and growing very rapidly.
- An IPv6 prefix can be 128 bits while IPv4 prefix can be 32 bits long. Thus, IPv6 lookup may need $4\times$ processing and memory compared to IPv4 lookup.

Existing solutions



- High-speed routers generally use ASIC and TCAM.
- This paper focuses on Software and ASIC based IP lookup.
- Pipelined ASIC use algorithmic solution (like software) in hardware.
- Software and ASIC based IP lookup solutions can be categorized as Trie and Hash based solutions.
- Trie based solutions are the main focus of this paper.

Trie based IP lookup algorithms

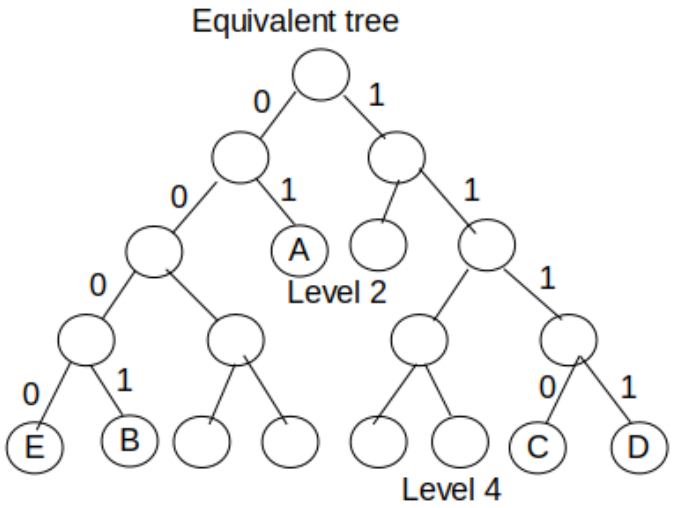


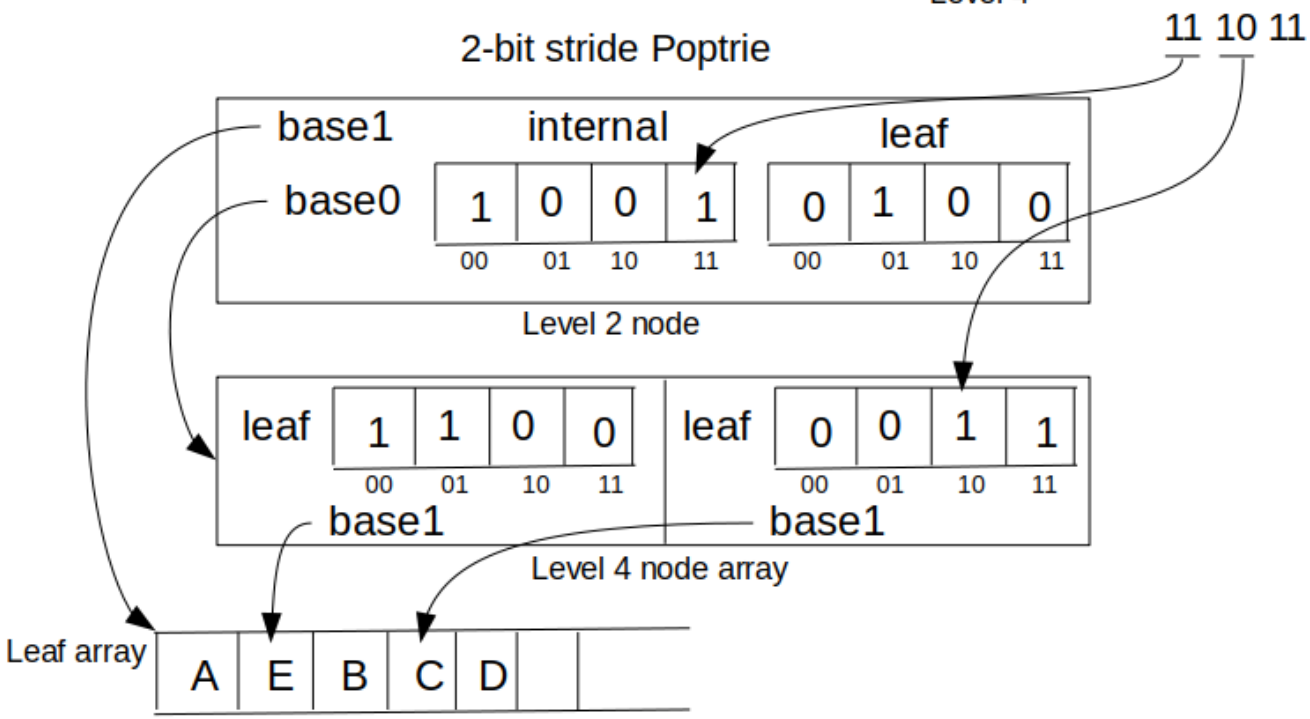
- **Poptrie** is the state of the art IP lookup because it requires very small memory and can perform very fast lookup.
- Poptrie however uses 6-bit stride.
- **CP-Trie** is extension of **Poptrie** which can uses $8 - 16$ bit stride, thus results more efficient lookup.

Poptrie based IP lookup

Forwarding table

Prefix (in binary)	Next-hop
01*	A
0001*	B
1110*	C
1111*	D
0000*	E



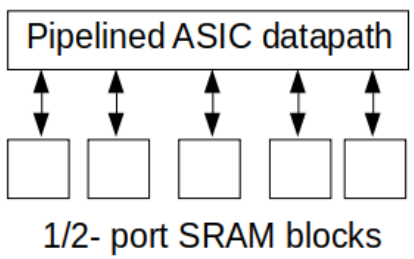


Limitation of Poptrie

- Poptrie uses 6-bit stride. This is because PopCount CPU instruction can only process 64 bits ($2^6 = 64$).
- As a result, Poptrie splits an IPv6 routing table table into 20 levels: level 16, 22, 28, 34, 40, 46, 52, 58, 64, 70, 76, 82, 88, 94, 100, 106, 112, 118, 124 and 130.
- In CP-Trie, we store cumulative PopCount along with bitmap with allows it to use longer stride ($8 - 16$ bit).
- CP-Trie splits an IPv6 routing table into 15 levels: level 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96, 104, 112, 120 and 128.

Benefits of longer stride

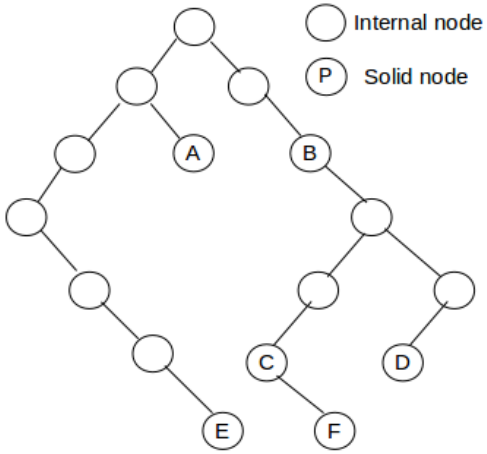
- Fewer number of steps
  - Results in faster lookup
  - Lower power consumption in ASIC
- Fewer memory accesses (critical in pipelined ASIC).
  - In pipelined ASIC, # of memory accesses = # of SRAM blocks needed (each pipeline stage has to access SRAM in parallel.).

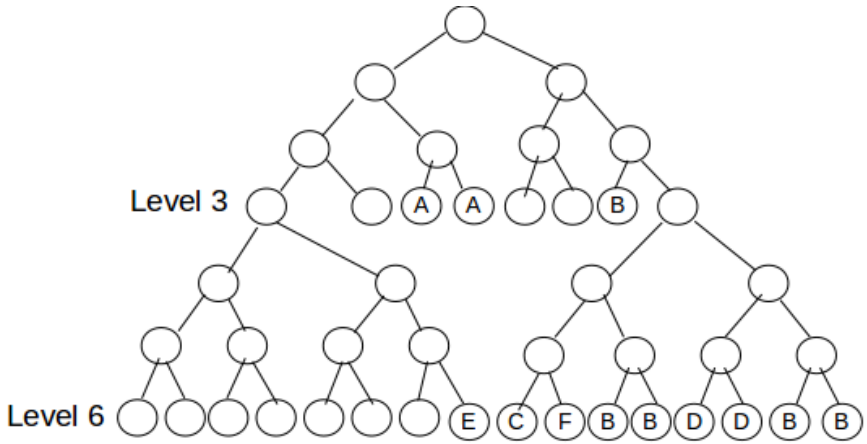


  - It also has been found that fewer large SRAM blocks are more area efficient than many smaller SRAM blocks. This is because 85% area of a SRAM is used for control logic.
  - This is why, reducing the number of memory access reduces the number of SRAM blocks needed in ASIC which results in smaller area.

CP-Trie based IP lookup

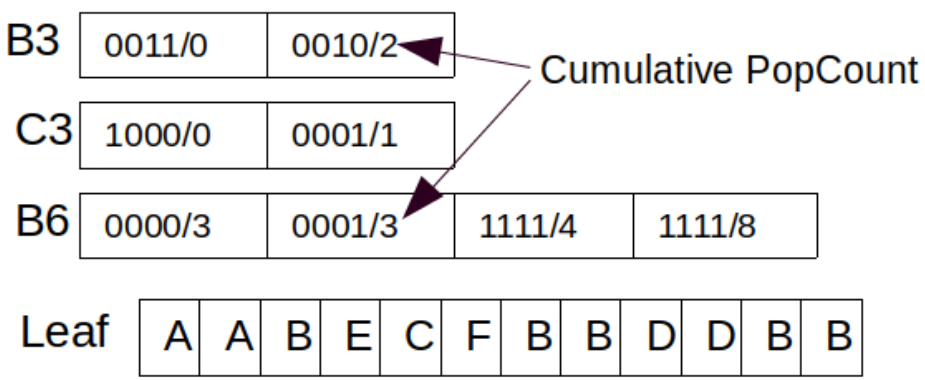
Prefix (in binary)	Next-hop
01*	A
11*	B
11100*	C
11110*	D
000111*	E
111001*	F

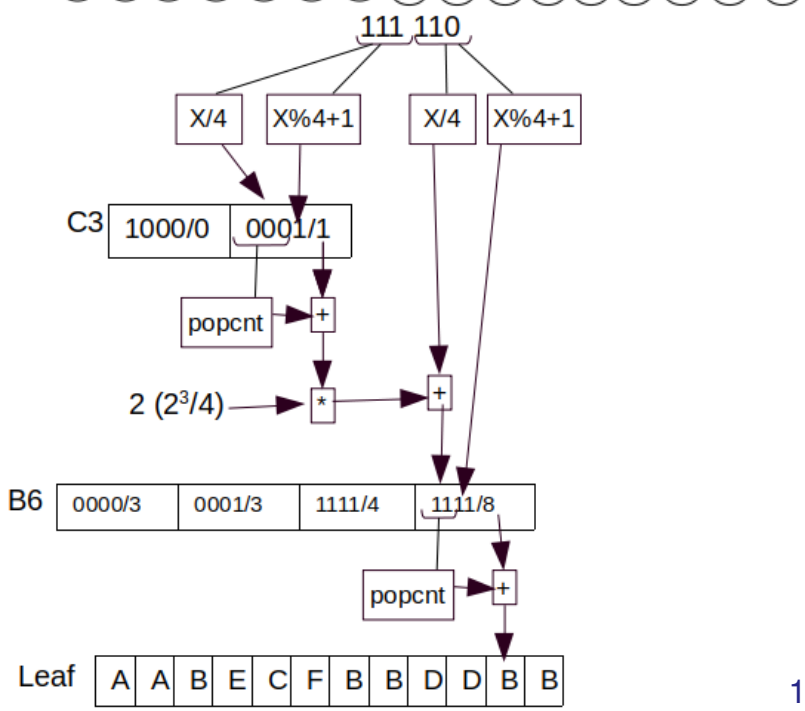




B3 = 00110010
C3 = 10000001
B6 = 0000000111111111

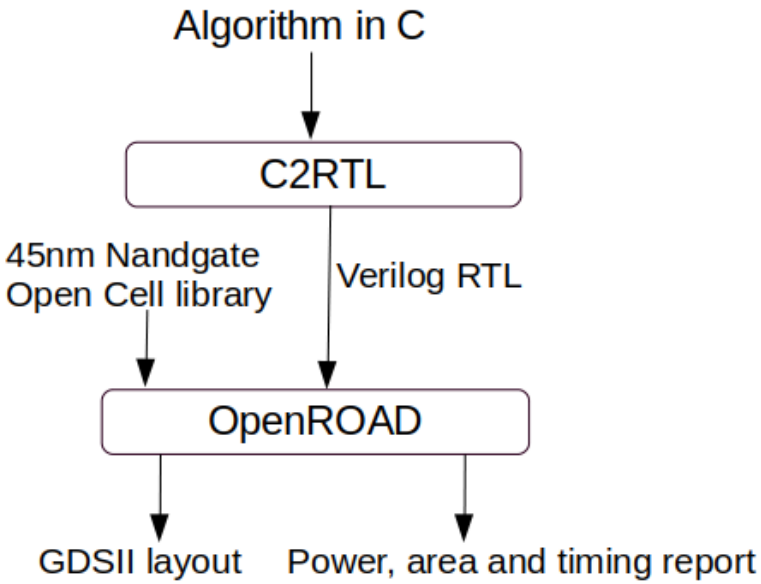
By grouping every 4 bits, we get:





Implementations

- Implemented both Poptrie and CP-Trie in C and Verilog RTL. Source code is publicly available at `https://tamimcse.github.io/cp-trie/`.
- We use a high-level synthesis (HLS) tool named C2RTL [**HPSR 2021**] to generate the Verilog RTL code.
- We generated physical chip layout using OpenROAD EDA. We also evaluate power, area and timing characteristics of ASIC using OpenROAD.



Evaluation in ASIC

	Poptrie	CP-Trie
Clock speed	1 GHz	1 GHz
Internal Power	76.5 mW	64.6 mW
Switching Power	24.4 mW	22.2 mW
Leakage Power	1.15 mW	0.926 mW
Total Power	102.05 mW	87.726 mW
Area	0.0658 mm^2	0.0523 mm^2

- SRAM implemation is missing in Open Cell Library. We use registers instead.
- Poptrie needs 79 SRAM blocks and CP-Trie needs 59 SRAM blocks. So, we expect CP-Trie to be even more efficient in area and power compared to Poptrie when SRAM is incorporated.

Evaluation in Software

Table: FIB dataset

Name	# of prefixes	Longest prefix length
fib0	105, 363	48
fib1	102, 126	48
fib2	79, 431	64
fib3	103, 067	128
fib4	105, 957	128
fib5	102, 739	64
fib6	104, 235	48
fib7	100, 899	64
fib8	102, 731	128

- We obtained the data set from RouteView project. The snapshot was taken on January 17, 2021 at 3 PM EST.

Memory consumption

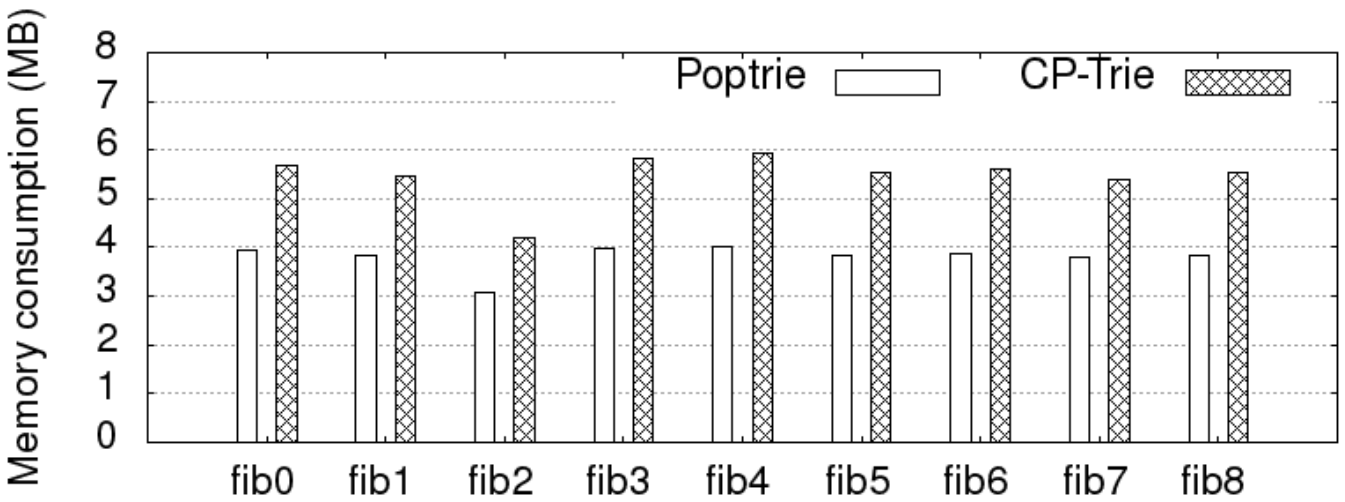
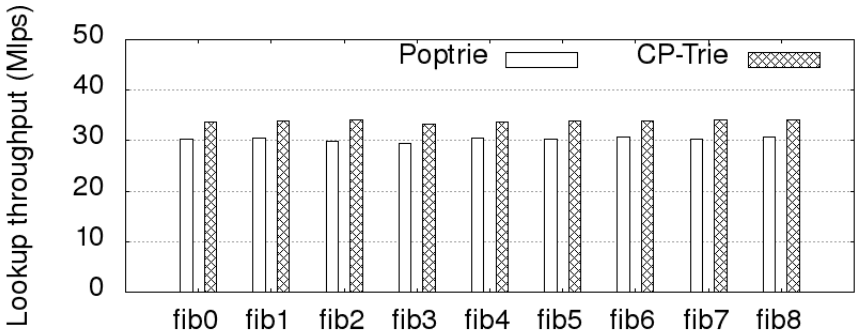


Figure: Memory consumption for different FIBs

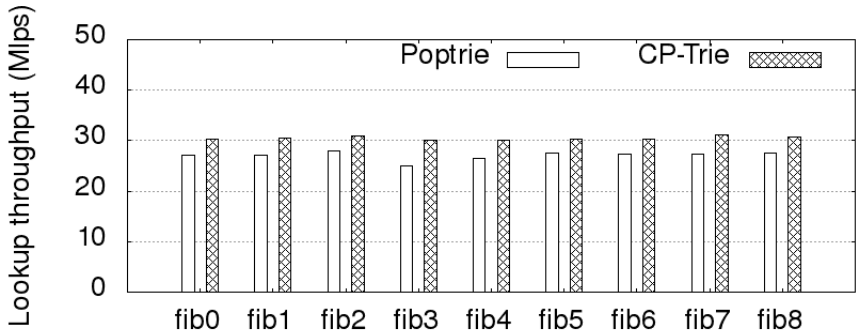
Lookup traffics

- **Prefix traffic** consists of the prefixes in our FIBs. Prefix traffic ensures that the lookups are spread across a FIB.
- **Repeated traffic** consists of 2^{24} randomly selected prefixes from the FIBs (with possible duplicates) and adding random bits at the ”don’t care” fields of the prefixes. Each IP in repeated traffic is looked up 10 times repeatedly. Repeated traffic is **analogous to real Internet trace**.
- **Sequential traffic** consists of 2^8 sequentially increasing IPv6 addresses. The start address of the sequential traffic is selected such that all the IPs in sequential traffic requires matching up to the last level of the FIB. We use sequential traffic to measure the worst lookup throughput for a FIB.
- **Random traffic** consists of 2^{24} random IPv6 addresses within 2000::/4.

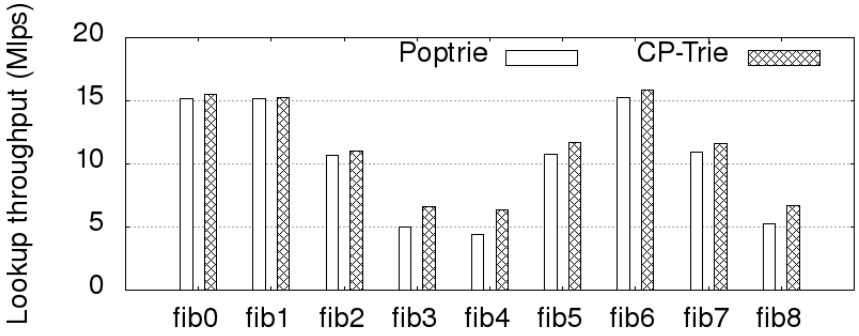
Lookup performance



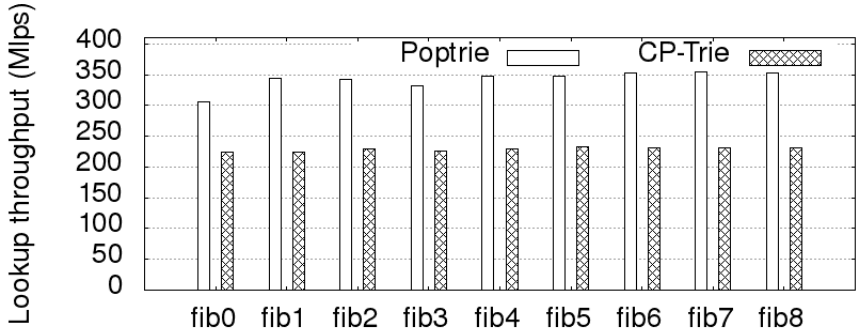
(a) Prefix traffic



(b) Repeated traffic (analogous to real Internet trace)



(c) Sequential traffic



(d) Random traffic (not a realistic traffic)

Update performance

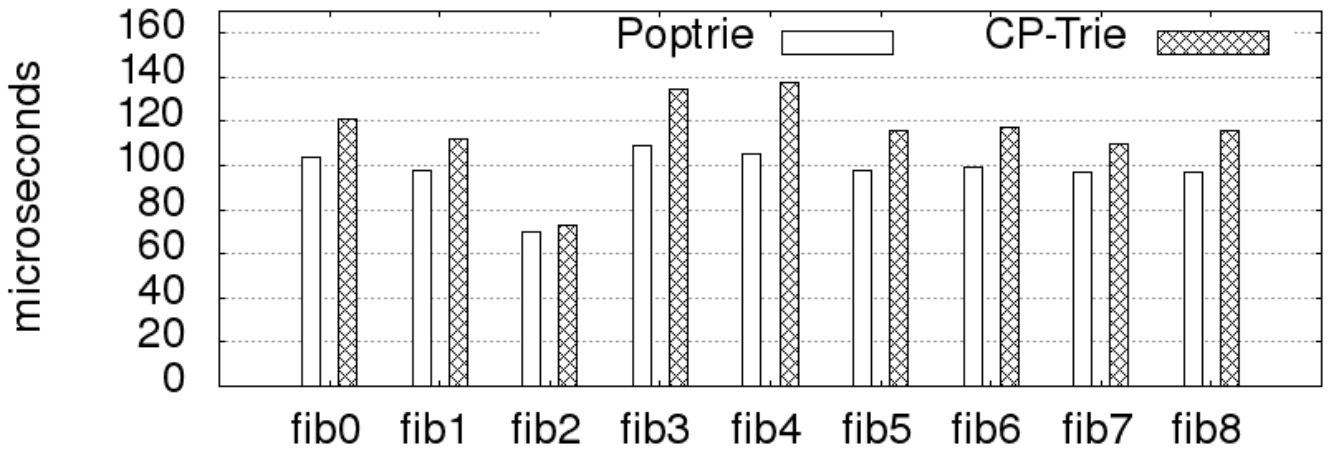


Figure: Average insertion time per prefix (smaller is better)

Thank You