Compressing IP Forwarding Tables for Fun and Profit

Gábor Rétvári, Zoltán Csernátony, Attila Körösi,
János Tapolcai
András Császár, Gábor Enyedi, Gergely Pongrácz

Budapest Univ. of Technology and Economics
Dept. of Telecomm. and Media Informatics
{retvari,csernatony,korosi,tapolcai}@tmit.bme.hu

TrafficLab, Ericsson Research, Hungary
{andras.csaszar,gabor.sandor.enyedi,gergely.pongracz}@ericsson.com

MŰEGYETEM 1782

ERICSSON

A Router in the DFZ

- Holds info on the whereabouts of every single IP address
- That ought to be a huge amount of information

A Router in the DFZ

- Holds info on the whereabouts of every single IP address
- That ought to be a huge amount of information
- So a DFZ router must be *huuuuuge*

Cisco CRS-3 line card
up to 8 Gbyte memory
533 MHz DDR2
>300 Watt

http://www.cisco.com/en/US/docs/routers/crs/crs1/4_slot/system_description/reference/guide/10805.pdf

A Router in the DFZ

- Holds info on the whereabouts of every single IP address
- That ought to be a huge amount of information
- So a DFZ router must be *huuuuuge*
- **Or must it?**

ASUS WL 500G Deluxe
32 Mbyte memory
4 Mbyte flash
200 MHz CPU
10 Watt

IP Forwarding Information Base

- A real FIB taken from `taz.bme.hu` (univ. access)
- Stores more than 410K IP-prefix-to-nexthop mappings
- Consulted on a packet-by-packet basis at line speed
  - Longest prefix match
- Takes several Mbytes of fast line card memory
- Some people argue that's a scalability barrier

Report from the IAB Workshop on Routing and Addressing, RFC 4984, 2007.

Zhao et al. *Routing scalability: an operator's view*, JSAC, 2010.

- Some people disagree

Fall et al. *Routing tables: Is smaller really much better?*, HotNets, 2009.

- Don't want to make this a debate on Internet routing scalability

How much information does a FIB actually need to store?

Can we achieve the storage size lower bound, retaining fast lookup?

Towards Compressed IP FIBs

- Store an IP FIB in as small space as possible
  - below 256–512 Kbyte
  - fit FIB into fast memory (SRAM/CPU cache)
  - maintain full forwarding equivalence
  - retain fast lookup!
- Our approach is systematic
  - identify redundancy in common FIB representations
  - eliminate it
  - attain entropy bounds
  - prototype and test on real traffic

Conventional FIB Representations

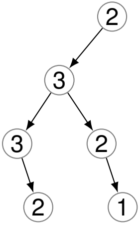
- Next-hops indexed on the alphabet $\Sigma = [0, K]$, $K \ll N$
- **FIB table:** lookup needs looping through all *N* entries
- Memory size is ~20 Mbytes on `taz`

Address/prefix length	Label
-/0	2
0/1	3
00/2	3
001/3	2
01/2	2
011/3	1

Conventional FIB Representations

- Next-hops indexed on the alphabet $\Sigma = [0, K]$, $K \ll N$
- **FIB table:** lookup needs looping through all N entries
- Memory size is ~20 Mbytes on `taz`

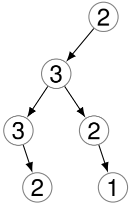
Address/prefix length	Label
-/0	2
0/1	3
00/2	3
001/3	2
01/2	2
011/3	1



- **Binary trie:** search tree over the address space
- Lookup improves to optimal $O(W)$ for W bit address size
- ~4 Mbyte on `taz`

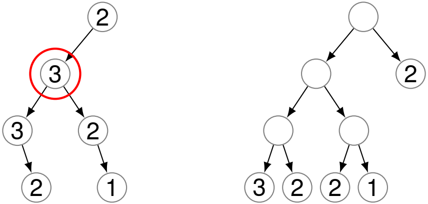
Redundancy in Binary Tries

- **Semantic redundancy:** entries superfluous due to longest prefix match



Redundancy in Binary Tries

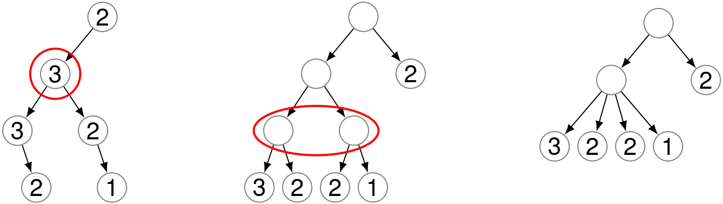
- **Semantic redundancy:** entries superfluous due to longest prefix match



- Leaf-pushing: push interior labels down to leaves
  - ~1.3 Mbytes on `taz`

Redundancy in Binary Tries

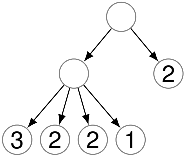
- **Semantic redundancy:** entries superfluous due to longest prefix match



- Leaf-pushing: push interior labels down to leaves
  - ~1.3 Mbytes on `taz`
- **Structural redundancy:** remove excess levels
  - multibit tries have nice structure
  - $<$1 Mbytes

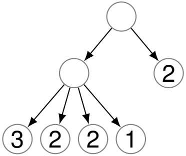
Information-theoretical Redundancy

- Certain labels appear frequently, encode these on fewer bits like Huffman-coding



Information-theoretical Redundancy

- Certain labels appear frequently, encode these on fewer bits like Huffman-coding



i	S_{last}	S_α	
1	1	0	level 0
2	0	0	level 1
3	1	2	
4	0	3	level 2
5	0	2	
6	0	2	
7	1	1	

- **Multibit Burrows-Wheeler transform:** serialize the trie in breadth-first-search order into two strings
  - S_{last}: bitstring encoding the tree structure
  - S_α: string encoding the labels
- Compress S_{last} and S_α to attain entropy bounds

Navigating MBW

- **String self-indexing:** a revolution is going around in TCS
- It is now possible to encode a string to higher-order entropy
- And provide O(1) operations on the compressed form!
  - the encoder supports simple navigational primitives in O(1)
  - lookup on MBW can be implemented in terms of these
- We use `RRR` on S_{last} and `Wavelet trees` on S_α
- Size is optimal in terms of the FIB entropy

$$H_0(p_c) = \sum_{c \in \Sigma} p_c \log \frac{1}{p_c}$$

- p_c is the empirical probability of next-hop labels in the FIB
- In fact, we can even attain higher-order entropy

Experiments on a Linux Prototype

- User space FIB compression, kernel module does lookup
  - could acquire only two real FIBs from the DFZ
  - rest is from collectors that obscure next-hop info
  - contain more than 410K entries

We need your help!
We need your FIBs!

Please, upload any FIB you can put your hands on to
`http://lendulet.tmit.bme.hu/fib_comp`

Output of `show ip bgp` or `show ip route` from a production DFZ router is preferred (but basically anything flies)

Experiments on a Linux Prototype

- User space FIB compression, kernel module does lookup
  - could acquire only two real FIBs from the DFZ
  - rest is from collectors that obscure next-hop info
  - contain more than 410K entries
- MBW compresses beyond zero-order entropy
  - 60–120 Kbytes (!) on FIBs with few next-hops
  - 256–400 Kbytes on FIBs with several hundred next-hops
  - 2–6 bits per prefix
- 3–10 complete rebuilds per second
- Churn out ~100 MBit/sec at 30-50 Kpps/sec

Demo

Discussion

- Contemporary FIBs can be encoded to 256–512 Kbytes with pointerless data structures
  - this is optimal, up to lower order terms
  - well below SRAM/cache size bounds of today
- And lookup is still *theoretically* optimal
  - in practice, two orders of magnitude worse than required
  - but this is only a proof-of-concept

Future?

- Entropy-compressed FIBs with linespeed lookup?
  - can we trade optimized HW away for optimized SW?
  - that is, better FIB compression algorithms in SW

Future?

- Entropy-compressed FIBs with linespeed lookup?
  - can we trade optimized HW away for optimized SW?
  - that is, better FIB compression algorithms in SW
- FIBs contain vast redundancy
  - why?
  - how to get rid of it from the outset?

Future?

- Entropy-compressed FIBs with linespeed lookup?
  - can we trade optimized HW away for optimized SW?
  - that is, better FIB compression algorithms in SW
- FIBs contain vast redundancy
  - why?
  - how to get rid of it from the outset?
- Historic analysis of FIBs entropy
  - how has entropy changed throughout the years?
  - hard to do without real data

`http://lendulet.tmit.bme.hu/fib_comp`