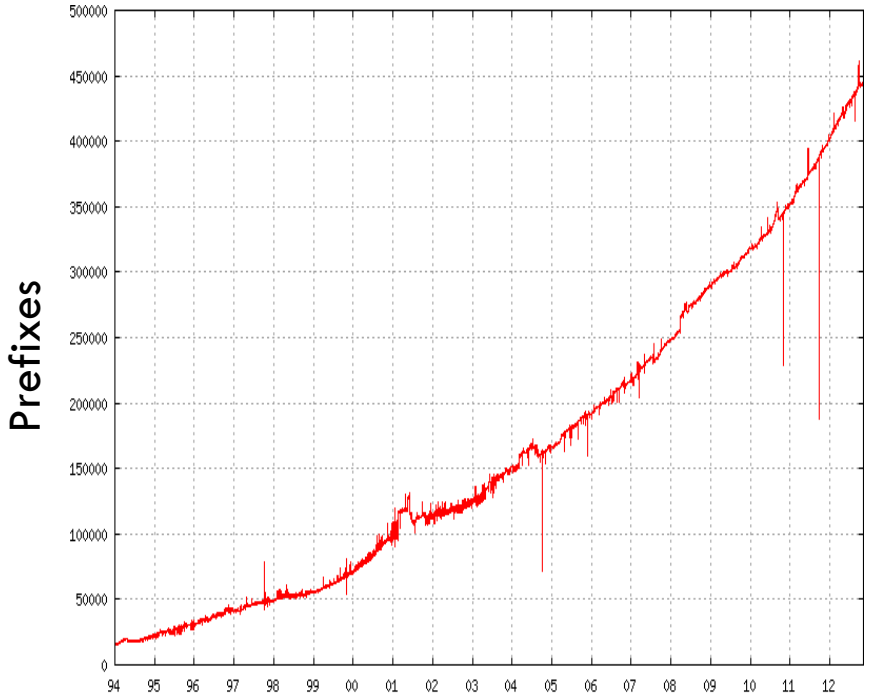
A FRESH LOOK AT SCALABLE FORWARDING THROUGH ROUTER FIB CACHING

Kaustubh Gadkari, Dan Massey and Christos Papadopoulos

Colorado State University

Problem: RIB/FIB Growth

- Global RIB directly affects FIB size
- FIB growth is a big concern:
  - Lookups need to keep up with increasing line speeds
  - FIB memory is small, expensive
  - Makes network provisioning hard
  - IPv6 growth may make things worse



The obligatory Geoff Huston plot, http://bgp.potaroo.net

Why Cache?

Performance! (cost too)

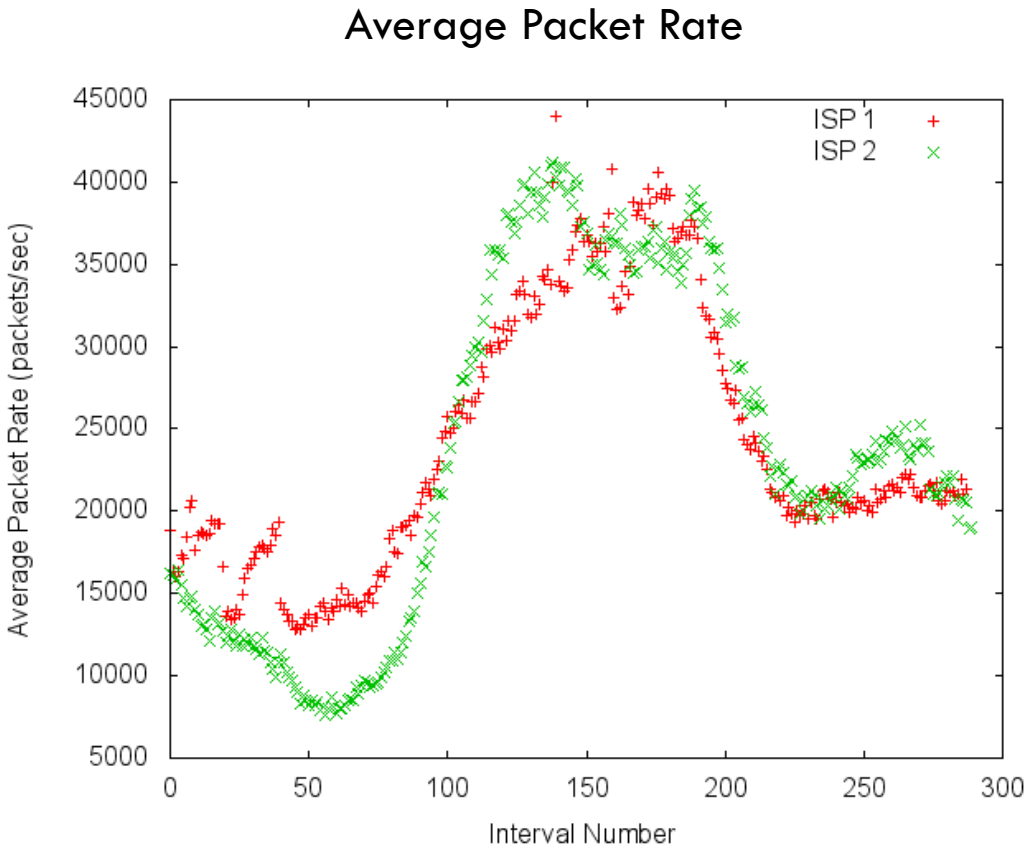
- Two potential benefits:
- Reduce the memory bandwidth required for FIB accesses
  - When FIB is accessed on forwarding decisions
- Better FIB compression on forwarding chips
  - Less compression (faster access) for the cache
  - More compression (slower access) for the rest
  - Potentially needed for Tb/s chips

Is There Traffic Locality?

Traces from our local friendly ISP, FRGP: Thanks guys!

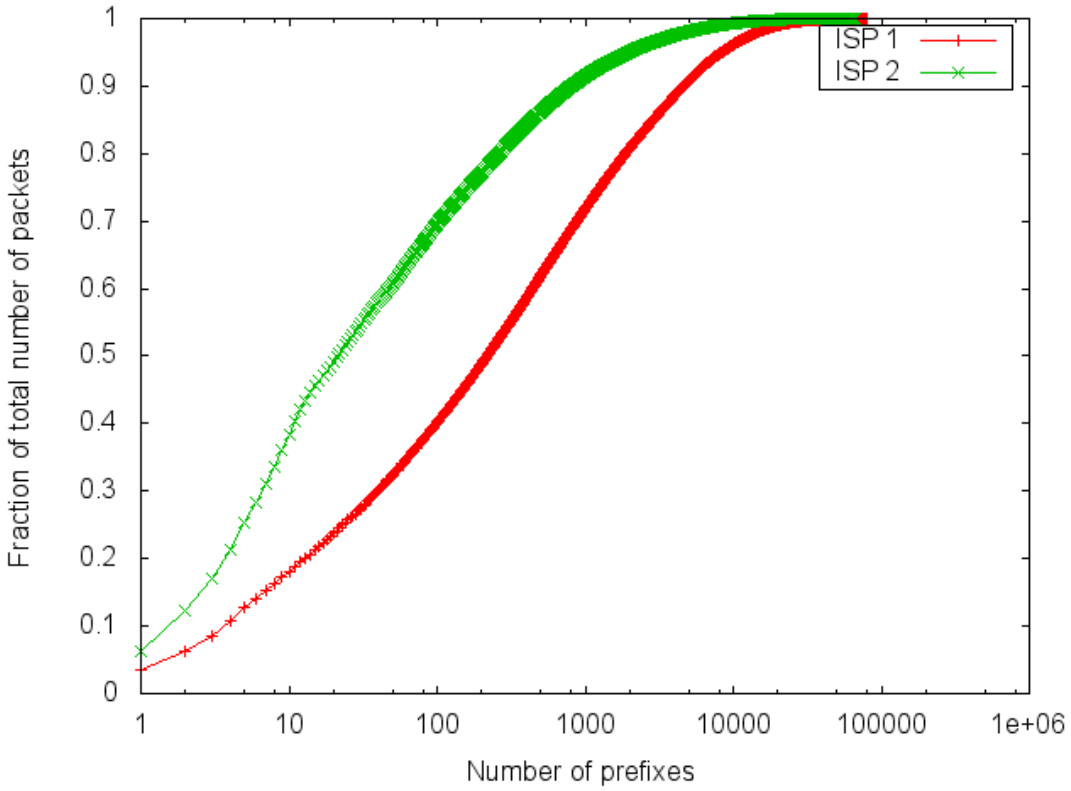
Test subjects: two 24H traces at two tier-1 provider links (1Gb/s) at a regional ISP

Link	Number of Packets
ISP-1	2,084,398,007
ISP-2	2,050,990,835



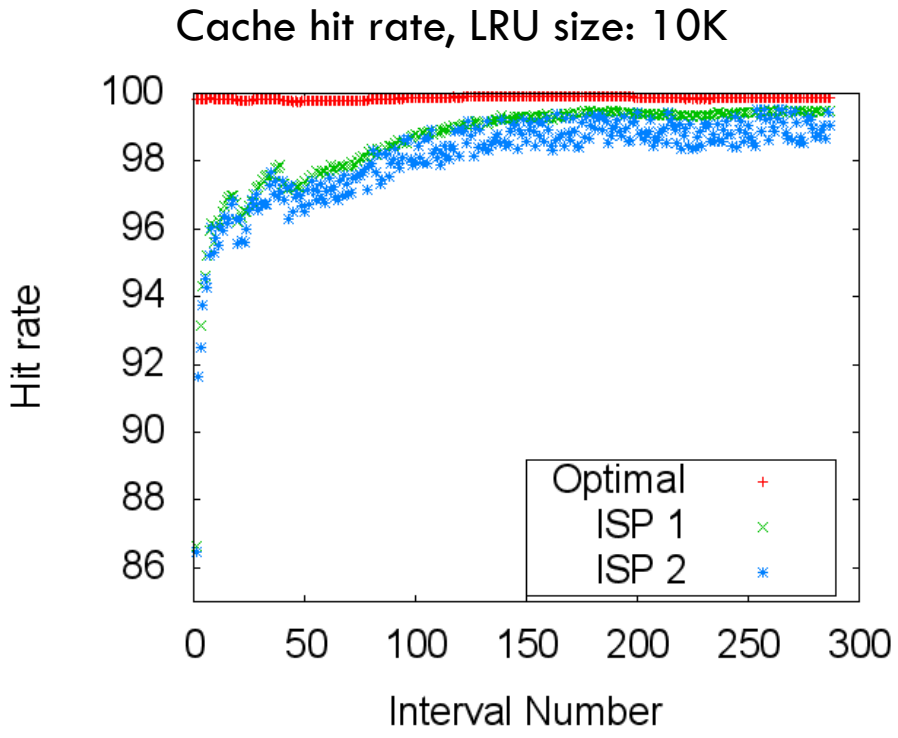
Yes, There is Locality!

- ~80k prefixes carry 99% of all traffic.
- ~1K prefixes carry 90% of the traffic
  - (but we already knew that, see Rexford's work and others)



Caching Results - LRU

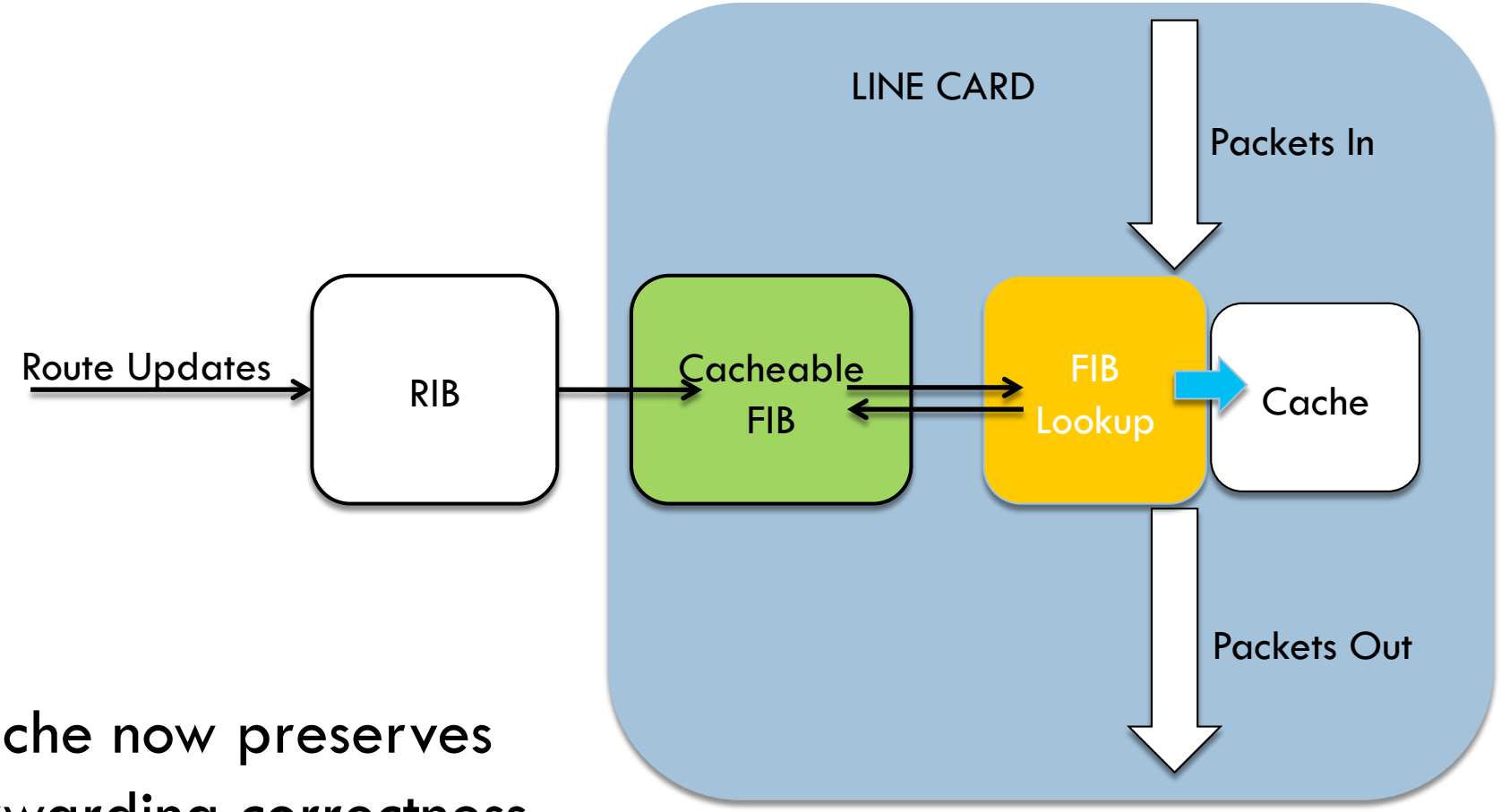
- Hit rate 96 - 99%
  - 24H trace, 5min intervals
- Close to optimal performance
  - Even at cache warm up hit rate = ~87-91%
- **Caching works!**
- Great! Now how do we build gear that use it?



Barriers to FIB Caching

- Three barriers to FIB caching
  - Cache hiding
  - Handling cache misses
  - Robustness

Secret Sauce: Cacheable FIB



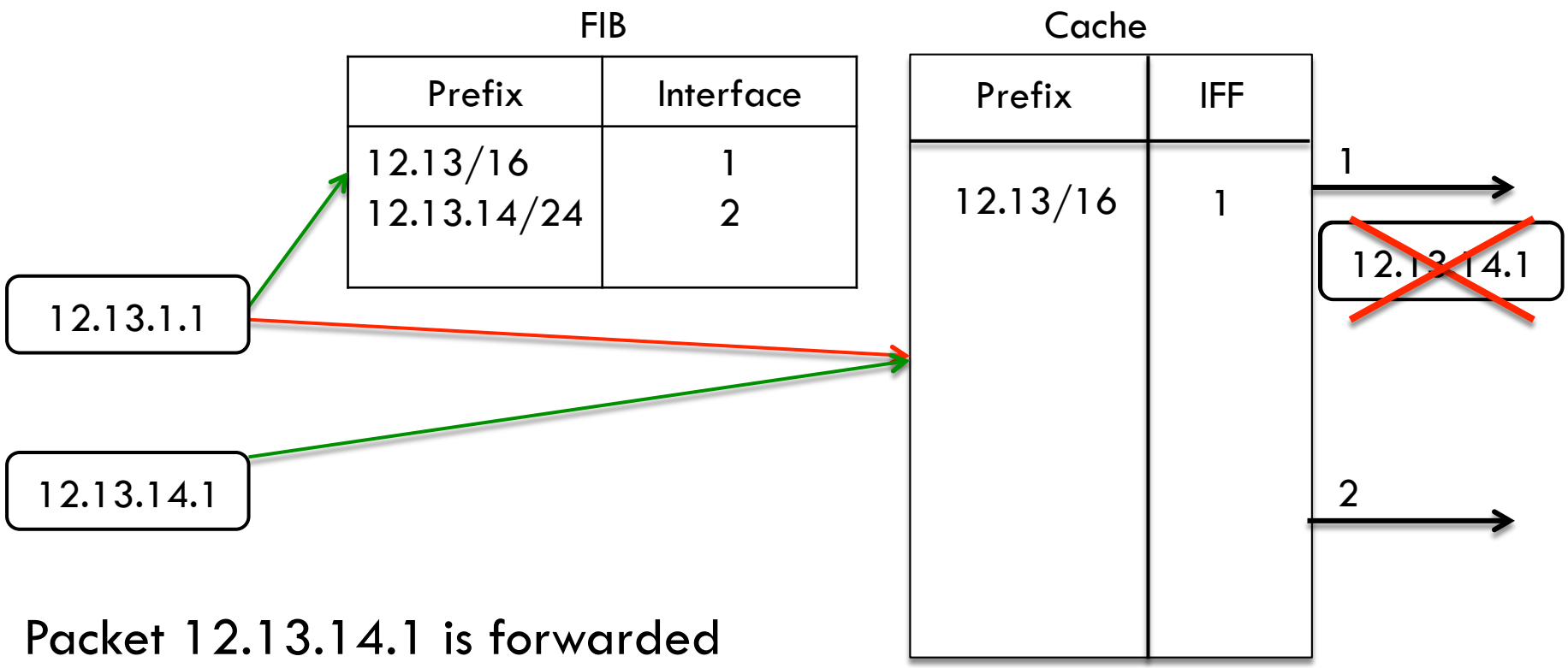
Cache now preserves forwarding correctness

The Cache Hiding Problem

- Consider the following snippet of a FIB
- The /16 covers (hides) the /24

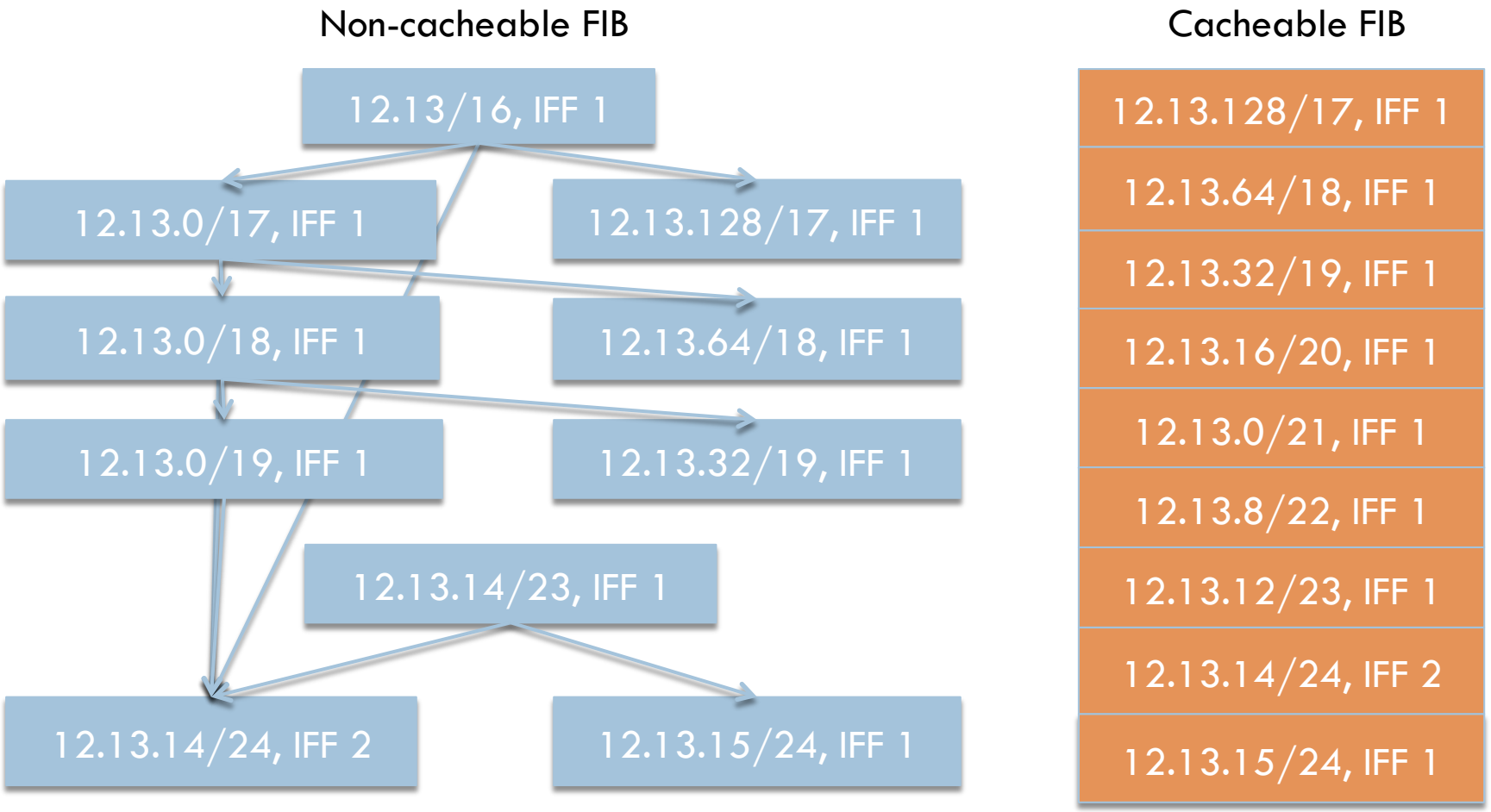
Prefix	Interface
.	.
.	.
12.13/16	1
12.13.14/24	2
.	.
.	.

Cache Hiding Impairs Forwarding



Packet 12.13.14.1 is forwarded incorrectly: 12.13/16 in the cache hides the true match

Solving Cache Hiding – Hole Filling



Have We Exploded the FIB?

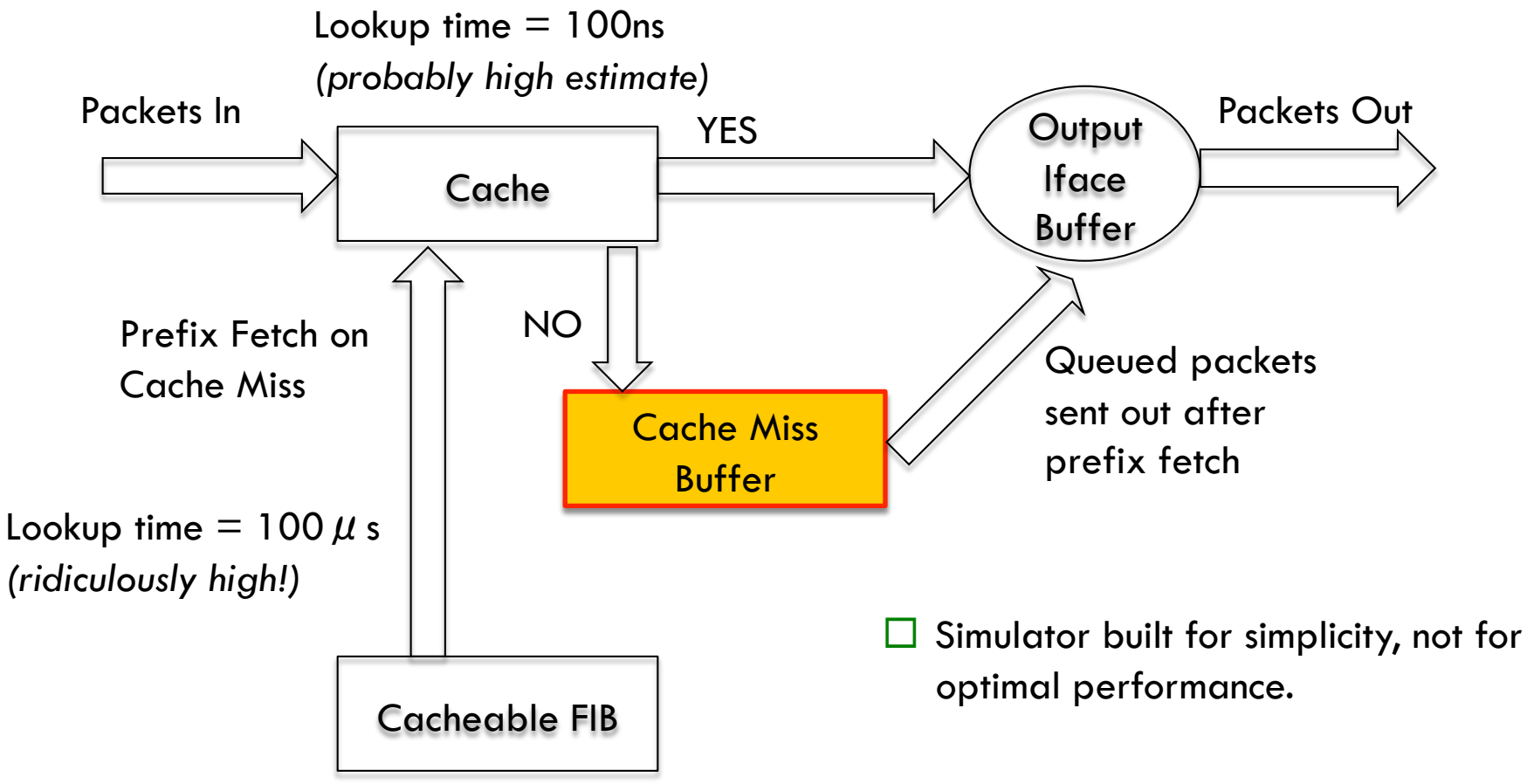
- **Actually, NO!**
- At FRGP, we go from 397,878 to 432,422 entries
  - 6.5% increase in size
- Other ISPs are similar (Route Views data)
- Caching makes this increase irrelevant anyway

Peer Name	% Increase
GBLNETRU	6.8%
CENIC	6.8%
Sprint	6.6%
APAN	6.6%
ESNet	6.6%
AOL	6.5%
Hurricane- Electric	6.5%
Sprint Canada	6.4%
Level3	6.4%
AT&T	6.4%

The Happy 99% vs. the Sad 1%

- What happens to the 1% of packets that miss the cache?
- They are queued until the cache is updated
- But what does that queue look like?

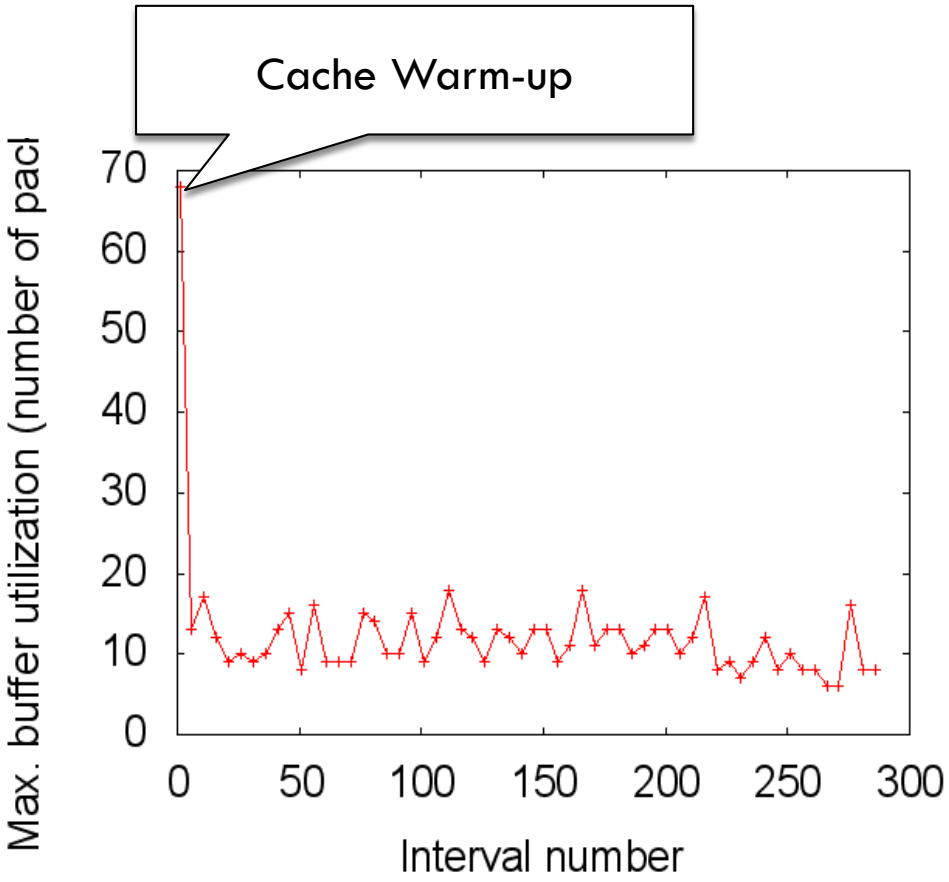
Queuing Simulator



- Simulator built for simplicity, not for optimal performance.

Cache Miss Buffer Utilization

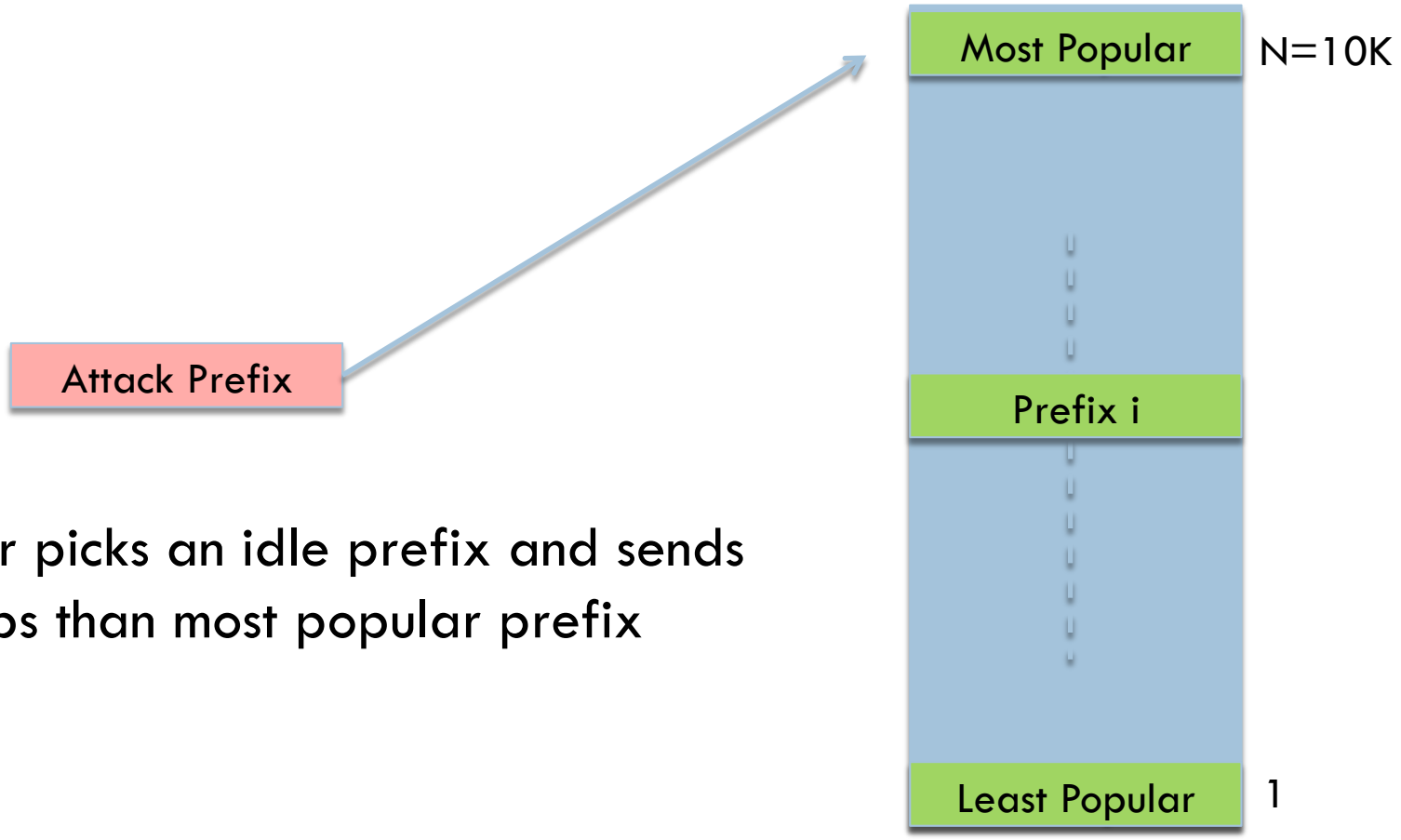
- No data packets queued!
  - 956K total misses, 752K SYNs, rest SYNACKs
- Buffer utilization is very low
  - approx. 10 packets in any given interval
- Small buffers needed to queue packets



Attacking the Cache

- Attacking a LRU cache is trivial :-(
  - Just send a train of packets to N idle prefixes
- Should have simulated LFU .. (next step)
- Research question: what is the appropriate cache replacement algorithm?
  - we plan to take a shot
- But with LFU, what rate does an attacker need to send packets to blow the cache?

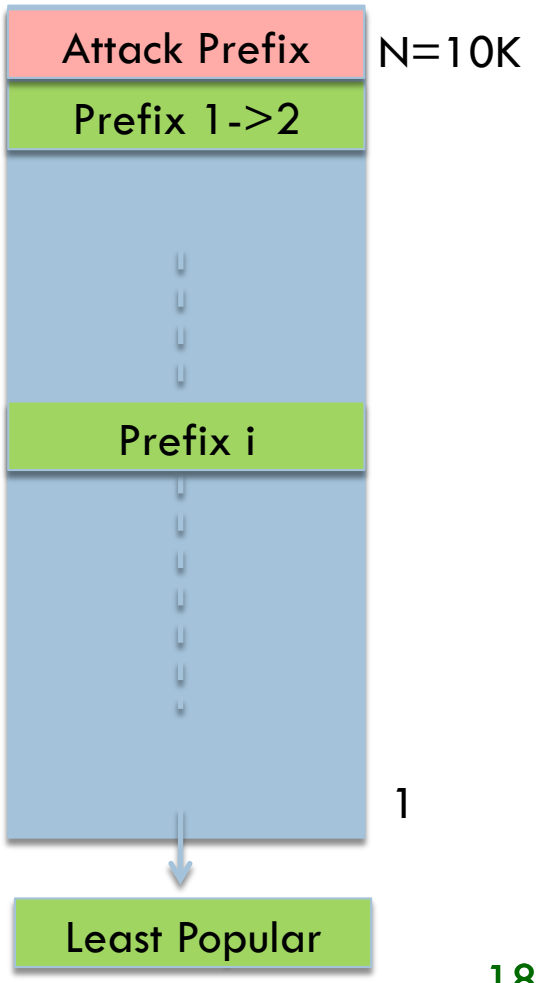
Attack on an LFU Cache



Attacker picks an idle prefix and sends more pps than most popular prefix

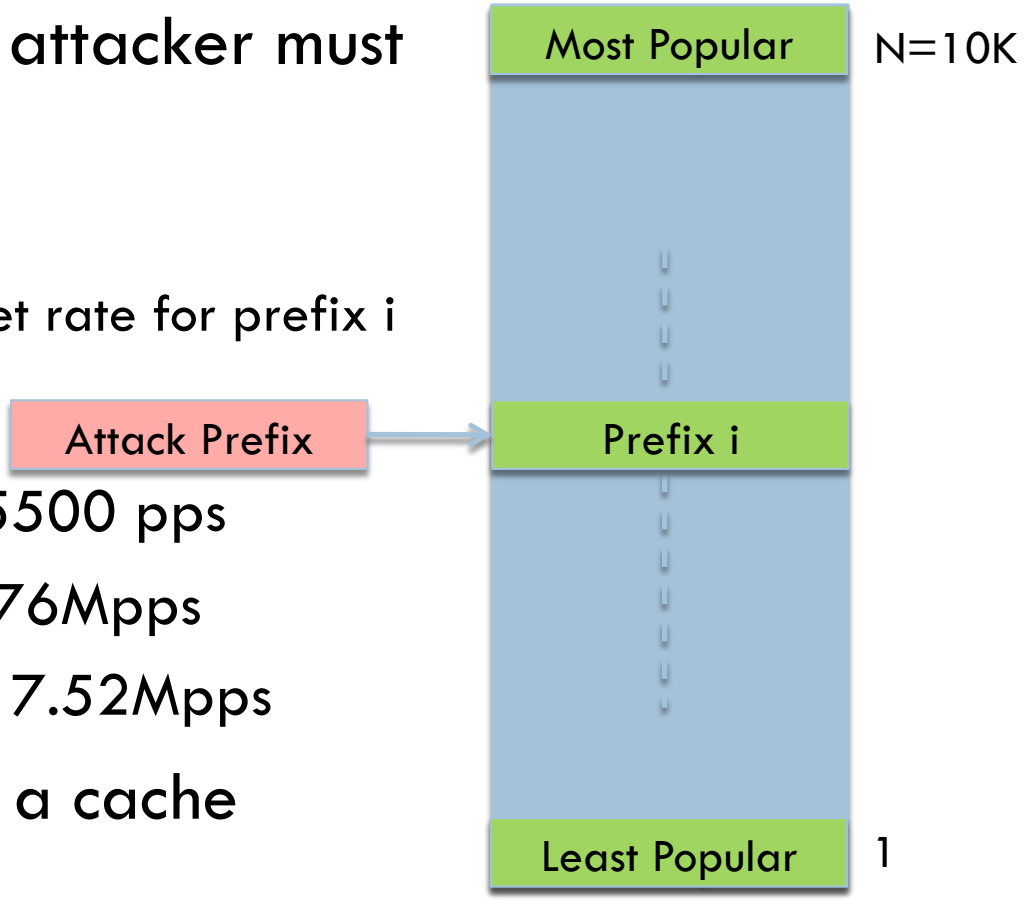
Result of the Attack

- Attacker prefix becomes top prefix
- Prefix 1 becomes prefix 2
- Least popular prefix is evicted
- *To evict all prefixes from cache attacker has to use N idle prefixes each at a rate higher than the most popular prefix*



Generalizing the Cache Attack

- To evict *prefix i* the attacker must send:
  - $P_{attack} >= P_i * i$
    - where P_i is the packet rate for prefix i
- From our trace:
  - i = 100, P_{attack} >= 5500 pps
  - I = 5K, P_{attack} >= 8.76Mpps
  - I = 10K, P_{attack} >= 17.52Mpps
- Flooding attack, not a cache attack!



Limitations – Future Work

- Do these observations carry to the core?
  - Need a trace from the core to investigate
  - Can you give us one? :-)
  - ..but recent trends point towards a traffic concentration to datacenters
- LRU, LFU cache replacement – tradeoff between performance and robustness?
- Better analysis of cache misses
  - Memory bandwidth demand
  - Who suffers from misses?

Conclusions

- Yet another reminder of traffic locality – and that caching works
  - 96-99% hit rate with a 10K cache at edge
  - no cache hiding problem
  - low queuing delay while updating the cache
  - cache fairly robust – attacks against top prefixes infeasible?
- So let's build gear with caches!
  - Unless we change physics, may be the only way forward with Tb/s speeds and large Global RIB (IPv6)

Thank You!

- Work funded by the DHS PREDICT project
- Data provided by Front Range GigaPop (FRGP)
- Thanks to the following people for fruitful discussions:
  - Jon Turner – Washington University, St. Louis
  - Will Eatherton – Juniper Networks
  - Chang-Hong Wu – Juniper Networks

Start Backup slides

One Solution: Cache the FIB

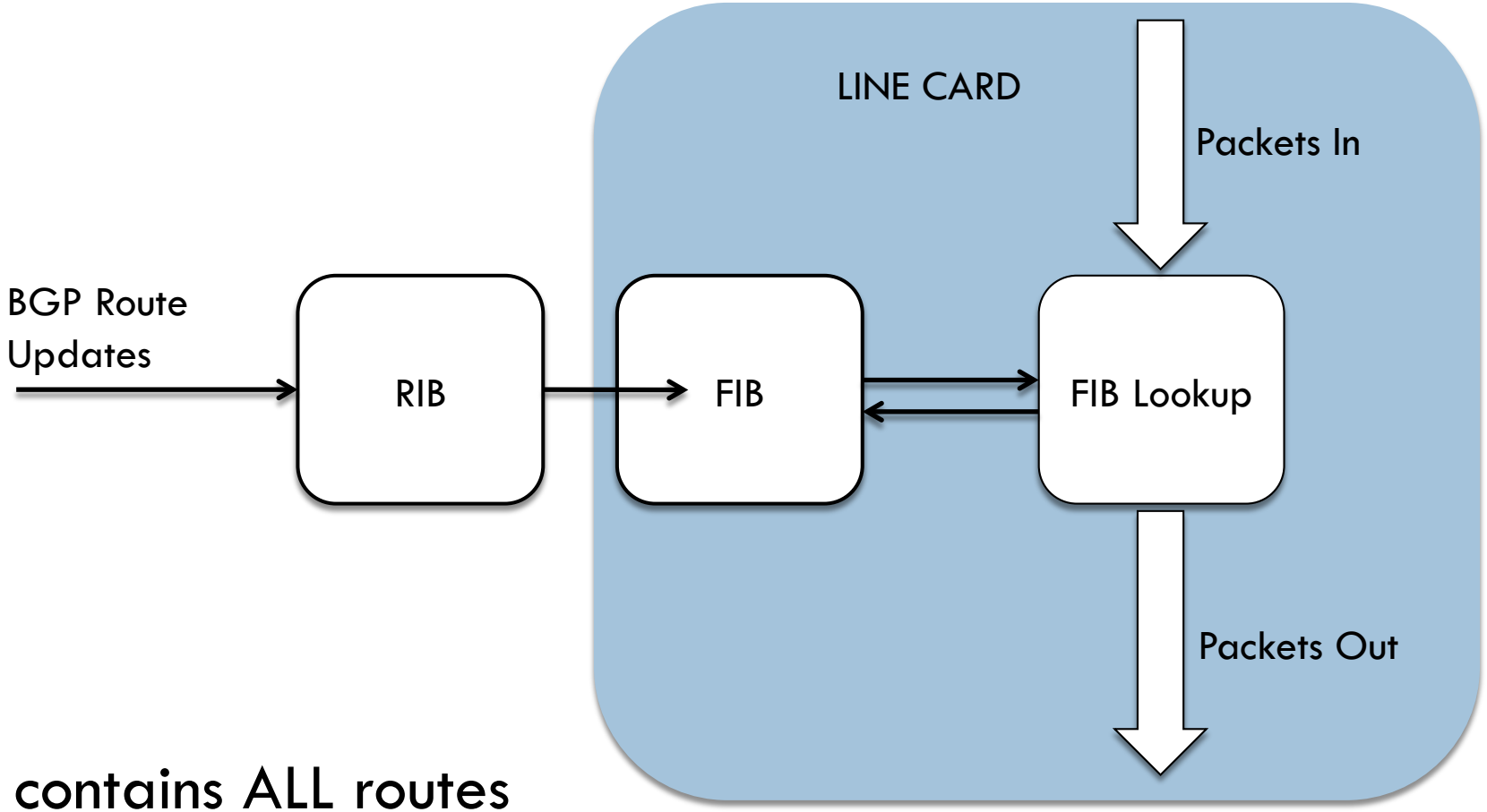
- Locality of network traffic is well-known
- LRU caching of /24s shown to provide 99%+ hit rate with a 100k entry cache
- Us: 99%+ hit rates with a 10k entry cache without de-aggregating prefixes
- We also investigate effects of cache misses and cache attacks

Solution: Making FIB Cacheable

Main idea:

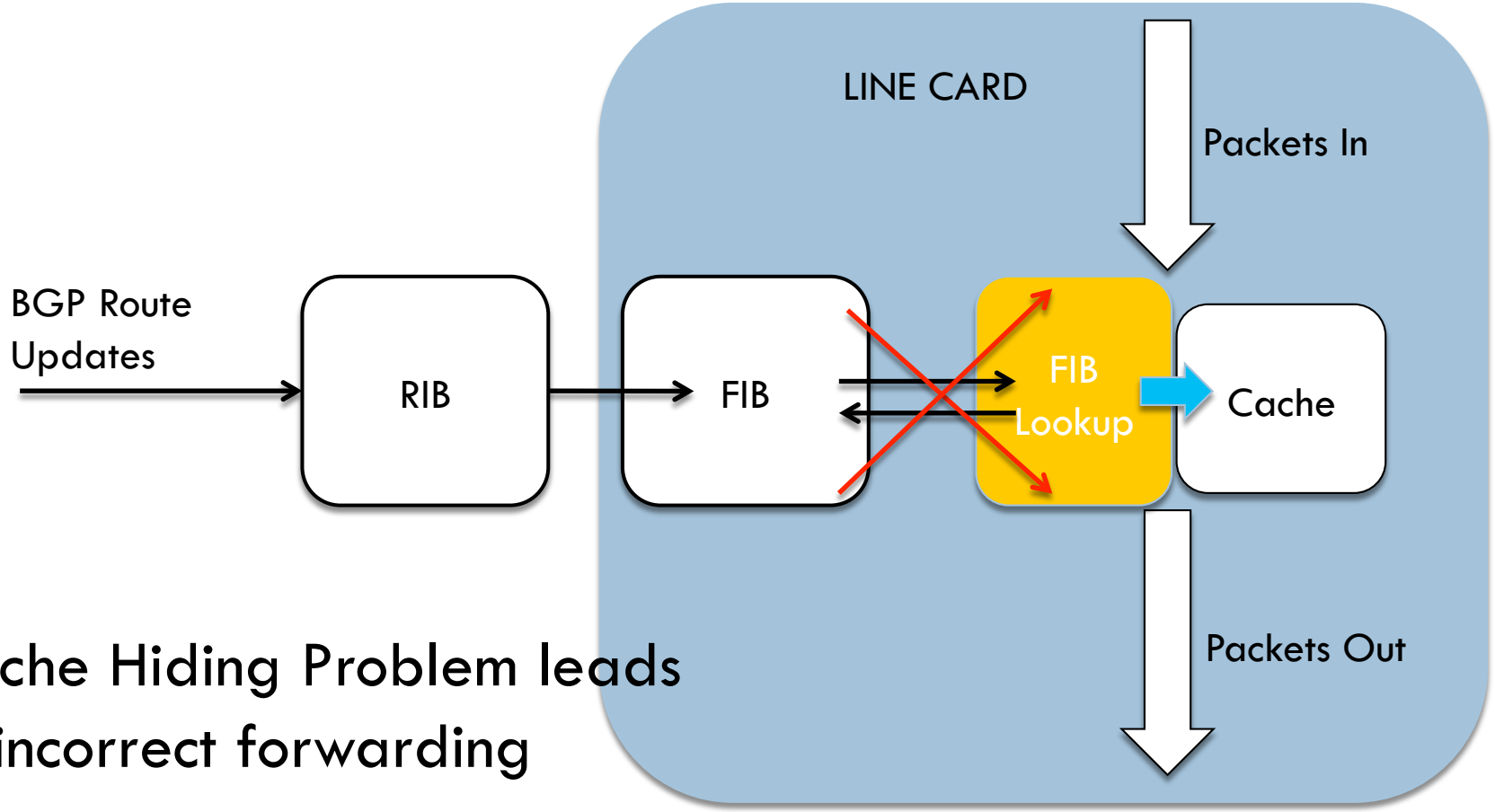
- Start with existing FIB
- Process FIB to replace all "hidden" prefixes with prefixes that cannot be hidden
- Produce new, cacheable FIB
- Serve the cache from the cacheable FIB

Current FIB Architecture



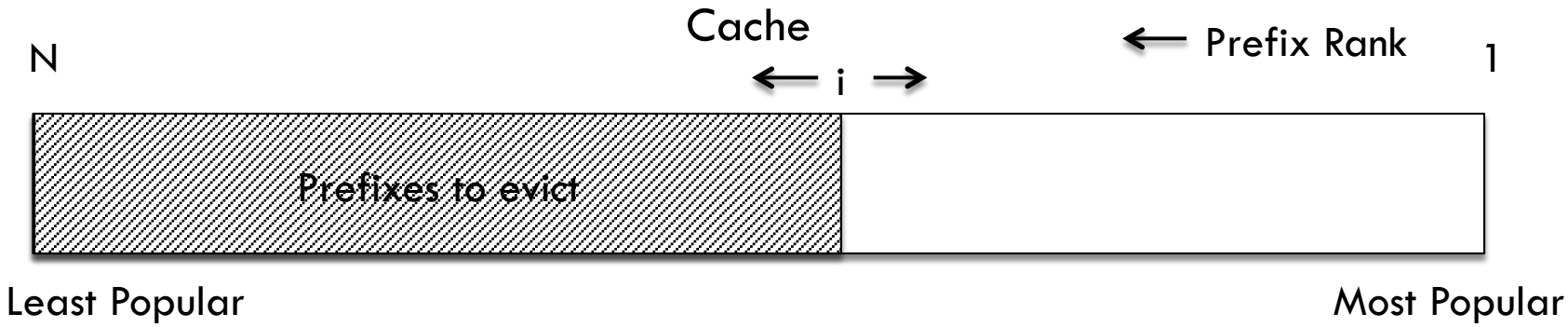
FIB contains ALL routes

Current FIB Not Cacheable!



Cache Hiding Problem leads to incorrect forwarding

Evaluation



To i_{th} prefix and all prefixes below it from cache, the required attack rate is

$$P_{attack} >= P_i * i$$

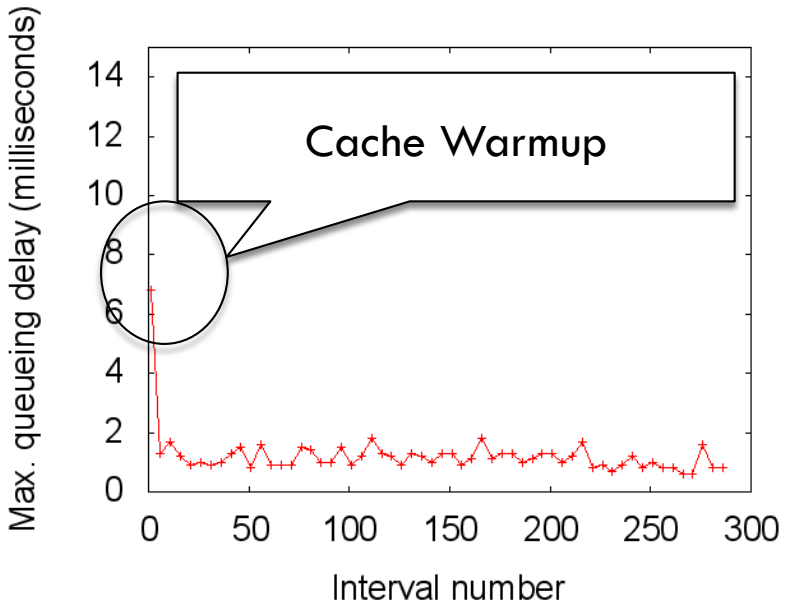
Assuming N = 10K entries, and *k* being the number of prefixes to evict

For k = 1, P_{attack} >= 1 pps
For k = 5K, P_{attack} >= 8.76M pps
For k = 10K, P_{attack} >= 17.52M pps

Queuing Delay

- Queued packets should not incur large delays

- Average delay is 1.1 ms.
  - Not counting cache warmup



TCAM Limitations

- TCAMs can do ~1.6B searches per second
  - 800M searches per second in the worst case
- Line cards typically have enough TCAM memory to store 512K IPv4 entries and 256K IPv6 entries
- TCAM lookups are fast (<20ns), but can they keep up with increasing line speeds (>= 100Gbps)?
  - 8,333,333 lookups per second, assuming all 1500 byte packets
  - 223,696,213 lookups per second, assuming all 60 byte packets

Implications of FIB Growth

- Routers can crash when they run out of memory (Chang02 *et. al.*)
- Some operators filtering out small prefixes (mostly /24s) rather than upgrade (Ballani09 *et.al.*)
  - Some parts of the Internet may become unreachable
- Network provisioning is harder
  - Difficult to estimate usable lifetime of routers and upgrade costs.

Impact of Cache Hiding

- Have to treat related prefixes at *atomic blocks*
  - All cache operations performed on entire block
- Largest atomic block size is 2557 prefixes
- Leads to cache thrashing
- Increases cache sizes
- Cache operations now more complex

Our proposal - Hole Filling

- Fill in *holes* between prefixes to eliminate cache hiding
- Add additional entries to the FIB
  - Trade FIB size for FIB cacheability
- Basic idea: Every prefix block is covered by *non-overlapping prefixes*
- This set is *optimal* (adds minimum number of prefixes required to cover the address space)

Publications

- Dynamics of Prefix Usage at an Edge Router, Kaustubh Gadkari, Dan Massey and Christos Papadopoulos, 12th Passive and Active Measurements Conference (PAM 2011), March 2011
- Fingerprinting Custom Botnet Protocol Stacks, Steve DiBenedetto, Kaustubh Gadkari, Nicholas Diel, Andrea Steiner, Dan Massey and Christos Papadopoulos, Workshop on Secure Network Protocols (NPSec 2010) (in conjunction with ICNP 2010), October 2010
- Dynamics of RIB Usage at an Edge Router (Poster), Kaustubh Gadkari, Steve DiBenedetto, Dan Massey and Christos Papadopoulos, 18th International Conference on Network Protocols (ICNP 2010), October 2010
- Characterizing TCP Resets in Established Connections, Nicholas Diel, Kaustubh Gadkari, Steve DiBenedetto, Andrea Steiner and Christos Papadopoulos, Technical Report CS-08-102.

Quantifying Cache Hiding

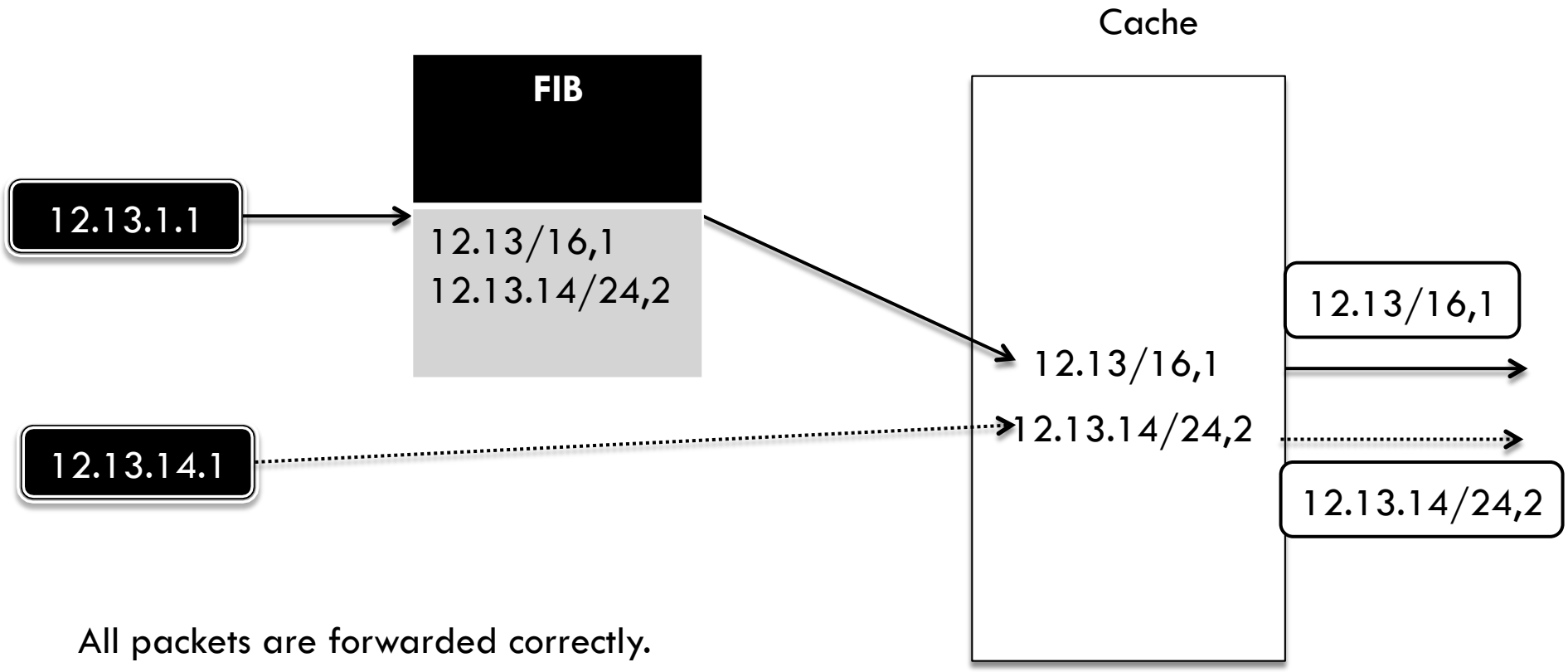
- Treat prefix groups as *atomic blocks.*

Peer Name	Number of Prefixes in Table	Size of Largest Atomic Block
GBLNETRU	345643	2557
CENIC	341122	2557
Sprint	338169	2557
APAN	344810	2557
ESNet	340874	2556
AOL	338247	2556
Hurricane-Electric	340402	2556
Sprint Canada	339509	2556
Level3	337701	2555
AT&T	338368	2552

Line Card Memory Limitations

- RIB stored in RAM (DRAM)
  - Large : *N* GB
  - Cheap : Approx. $200 per GB
  - Scaling not considered a problem
- FIB stored on line cards (SRAM/TCAM)
  - Small : *N* MB
  - Expensive : Approx. $4000 per GB
  - FIB is the union of all RIBs
  - Scaling is considered a problem
- IPv6 impact is unknown
  - Size of FIB after IPv6?
  - Lookups at speeds of 100G+?

Addressing Cache Hiding



All packets are forwarded correctly.

Solution Space

Table (FIB/RIB) / Method	FIB	RIB
Architectural	• Caching	• Edge-core separation
Configuration-only	• Aggregation	• Aggregation

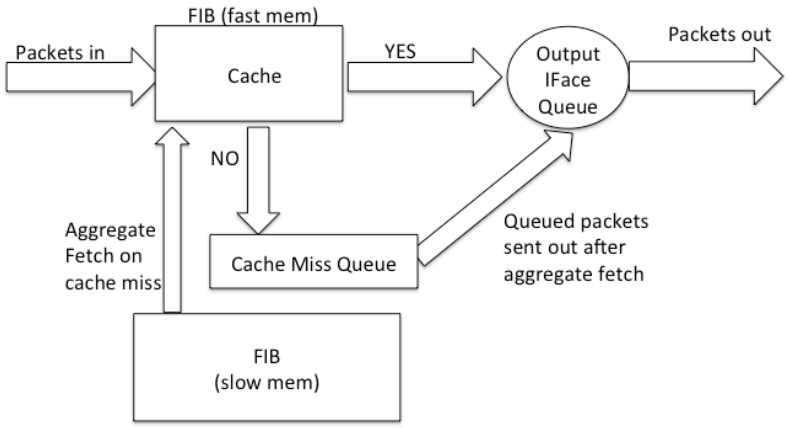
Uni-class Caching

- Kim *et. al.* show that route caching is feasible, and may be necessary.
- Cache only /24 prefixes.
  - This mitigates the *cache hiding problem*.
- Achieves 99%+ hit rates with cache sizes of 100k entries.

Aggregation

- Better than the naïve solution – leads to smaller FIBs.
- Compress FIB entries based on next-hop information.
- Can achieve 30 – 70% compression, based on aggregation algorithm.

Cache Miss Evaluation



- Built simulator to evaluate effect of cache misses.
- Simulator built for simplicity, not for optimal performance.

Handling Dynamic Route Updates

- Investigate how dynamic updates affect route caches
  - Cache entries can be invalidated
- How many updates actually affect cache entries?
- How do we handle those updates that do affect cache entries?

Implementing Caching on Current Hardware

- Investigate whether we can implement the caching scheme in current router hardware.
- Ideally, routers should not need new hardware to use our caching solution.

Evaluation

- To evict N_{th} entry from cache, attacker must send enough packets to that prefix and all cache entries below it
- To evict *i* entries, attack rate required is

$$P_{attack} >= P_i * i$$

- In low traffic interval, most popular prefix received 240K packet in 5 mins, at an avg. packet rate of 803 pps
- Assuming cache size of 10K, attacker needs to send 8M pps to blow away cache
  - 17.52M pps in high traffic interval
- This traffic rate is higher than the line speed at our capture point (2M pps, assuming 60 byte packets)

Threat Model

- Attacker knows, or can determine, set of popular and unpopular aggregates
- Attacker can send packets at line speed
- Attacker aims only to replace legitimate cache entries with bogus ones
  - Attacks on other infrastructure (e.g. DDoS) will trigger other defenses
- Attack cannot be stealthy
  - Attack rate must compete with legitimate traffic
- What is the packet rate required?

Future Work

- We will investigate how to handle dynamic route updates

- Further, we will investigate whether our caching solution can be implemented on existing router hardware

What is Optimal Caching?

- Defined on a cache size N, finite network trace
- When evicting a prefix choose the one that will not be used for the longest time in the future
  - This includes the current prefix!
  - Robust against one-off packets
- Theoretical algorithm – please do not try to implement in practice