Hardware Prefetchers and Security:
A Two-Edged Sword

Biswabandan Panda

CARS@CSE-IITK

Indian Institute of Technology Kanpur

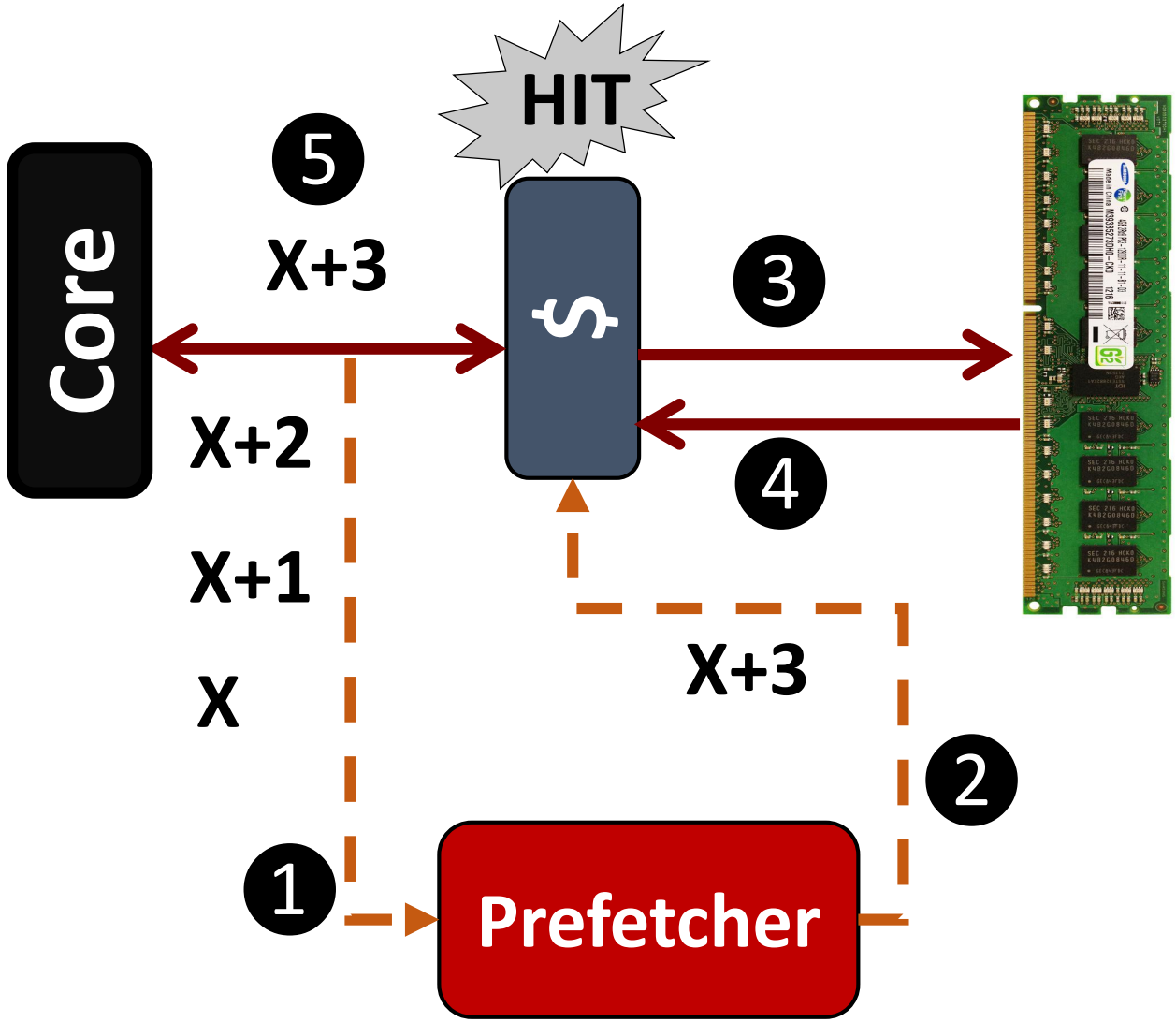
MAST 2019

October 11th, 2019

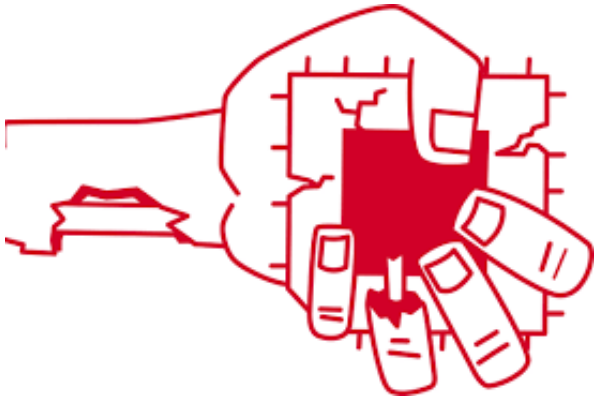
Hardware Prefetchers ?

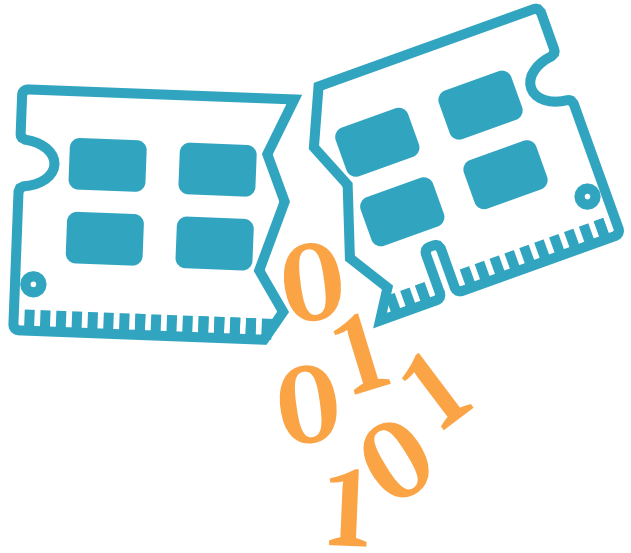
Pre-fetchers: Fetch Before Time (event) to save time!

Hardware Prefetchers-101



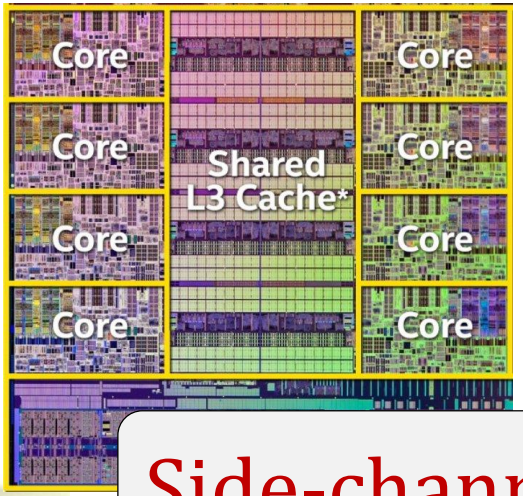
Security? Microarchitecture Attacks

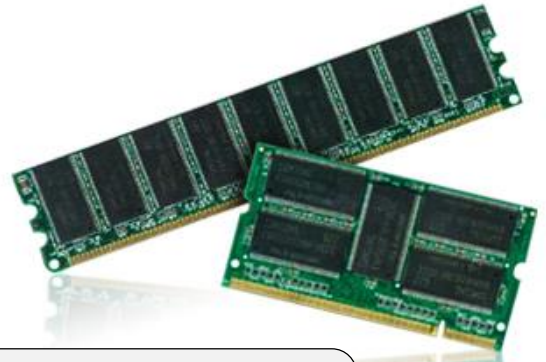




Security? Microarchitecture Attacks

Spy

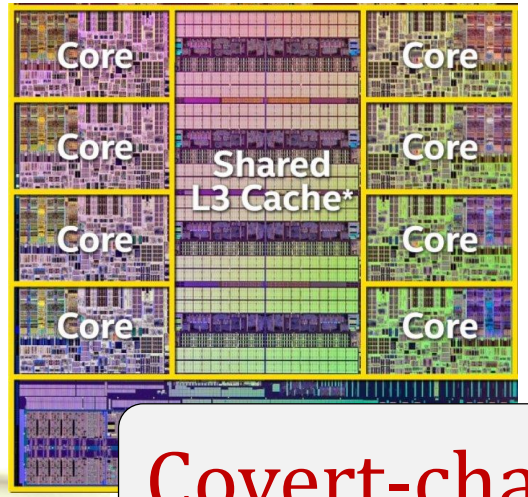


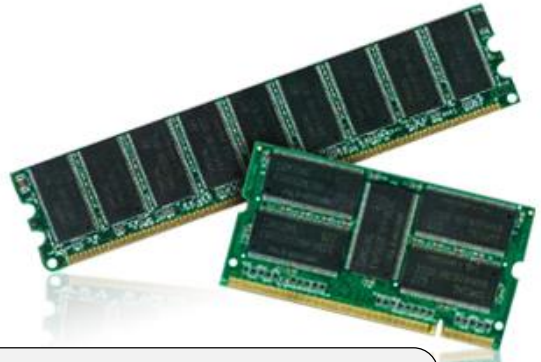


Side-channel attacks

Victim

Let's play

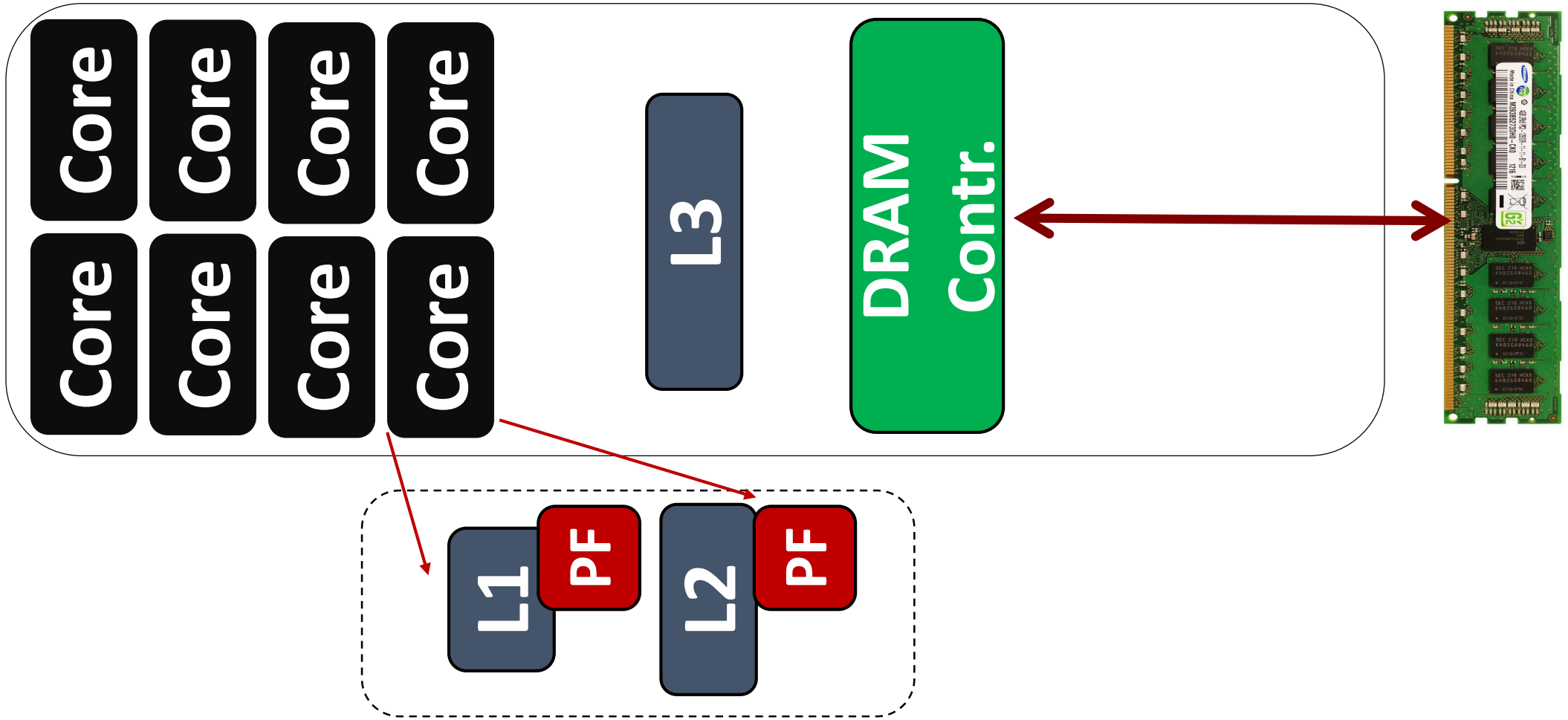




Covert-channel attacks

Oh Yes!!

Current Multicore Systems



Two Edged Sword

Hardware Prefetching: ***(Mitigating cross-core timing channels)***

Back-invalidation Triggered Prefetching [PACT '19]

Hardware Prefetching: ***(Creating a cross-thread timing channel)***

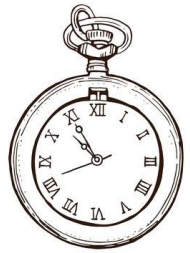
Whispering Streamers [HASP@ISCA'19, Poster@MICRO '19]

Fooling the Sense of a Cross-Core Eviction based Attacker by Prefetching the Common Sense [Biswa, PACT ‘19]

Fooling the Sense of a Cross-Core Eviction based Attacker by Prefetching the Common Sense [Biswa, PACT ‘19]

Microarchitecture Attacks at the LLC

Attacks at the LLC exploit timing channels:
LLC miss > LLC hit



Flush + Reload

Evict + Reload

Prime + Probe

clflush

Eviction based attacks

Focus of this talk

Information Leakage: An Example

```
x ⟵ 1
for i ⟵ |e|-1 downto 0 do
    x ⟵ x^2 mod n
    if (e_i = 1) then
        x = xb mod n
    endif
done
return x
```

square

reduce

multiply

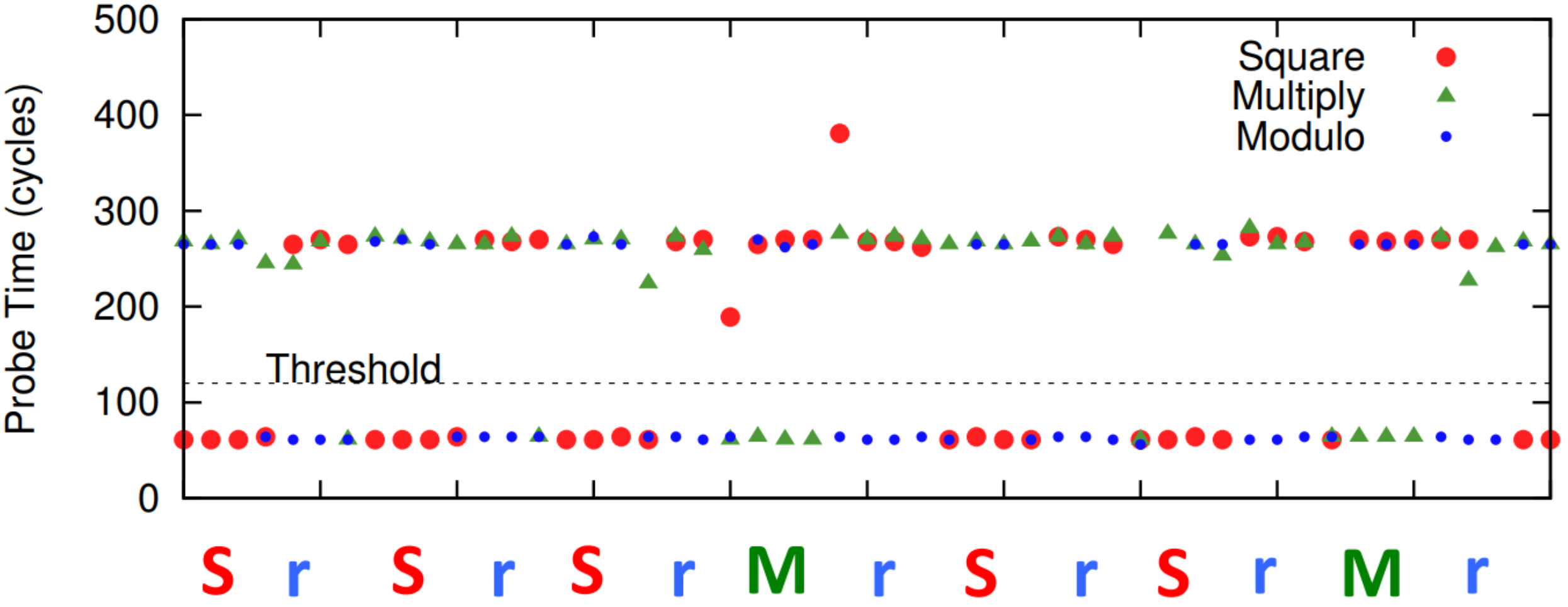
Modular exponentiation, b^e mod n

Exponent *e* is used for decryption

e_i = 0, Square Reduce (SR)
e_i = 1, SRMR

Attacker tries to get the e

Information Leakage: An Example [Usenix Security '14]



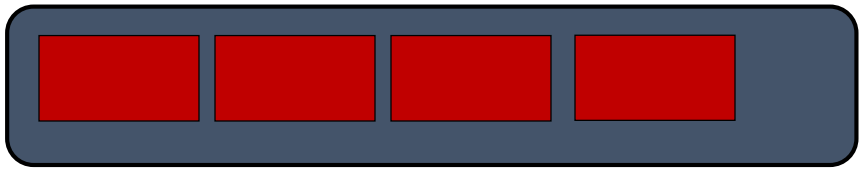
Threat Model

Knowing the victim *has accessed a cache set* can be considered as a *successful* attack

Fooling the Sense of a Cross-Core Eviction based Attacker by Prefetching the Common Sense

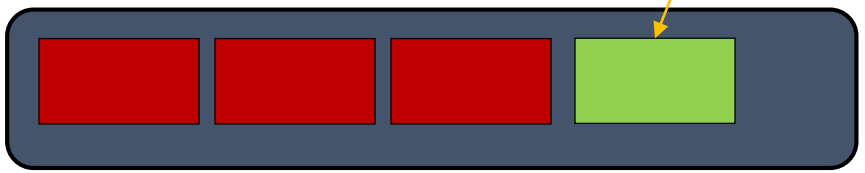
Prime + Probe at the LLC

Step 0: Spy *fills* the entire shared cache (a few sets)



LLC

Prime + Probe at the LLC



LLC

Step 0: Spy *fills* the entire shared cache (a few sets)

Step 1: Victim *evicts* cache blocks while running

Prime + Probe at the LLC

LLC

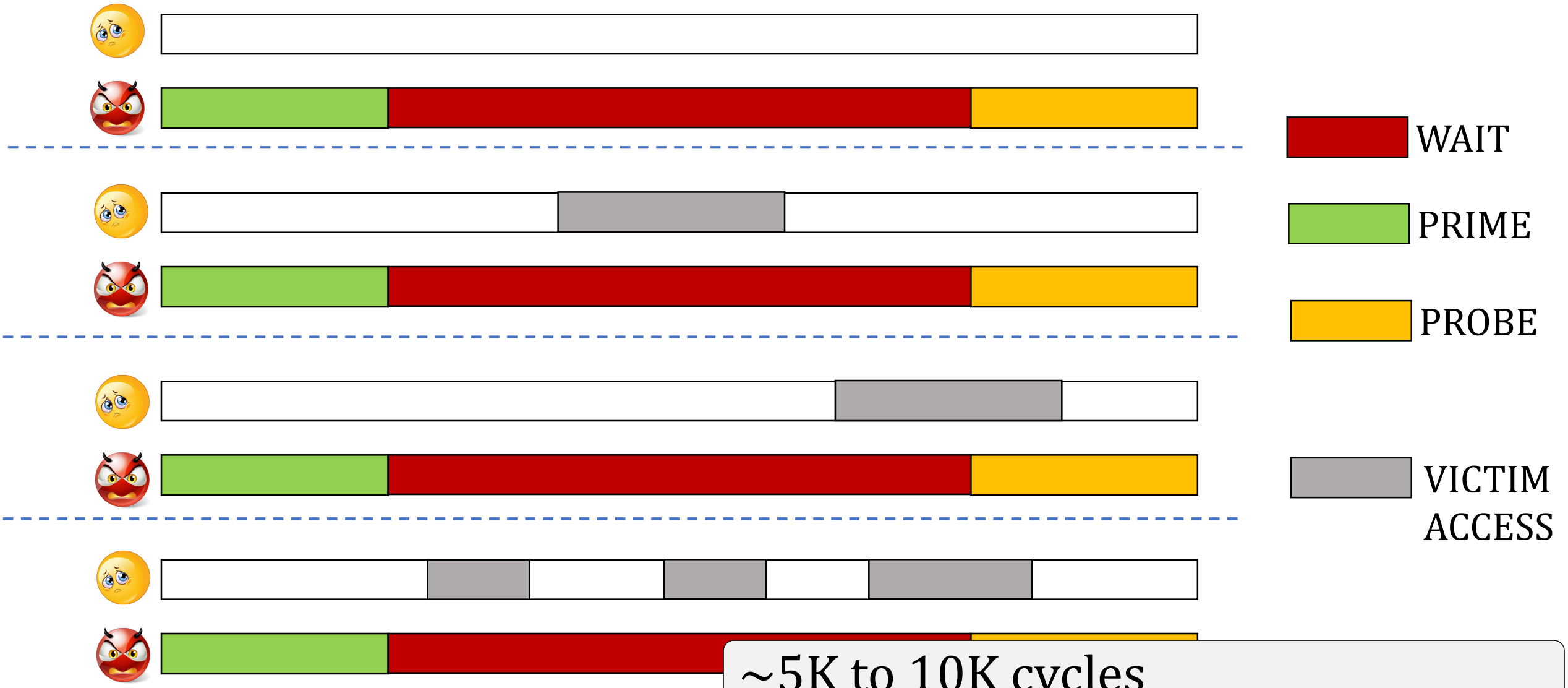
Step 0: Spy *fills* the entire shared cache (a few sets)

Step 1: Victim *evicts* cache blocks while running

Step 2: Spy *probes* the cache set

If *misses* then victim has accessed the set

Notion of Time Gap



~5K to 10K cycles

Prior Mitigation Techniques

Cache Partitioning [TACO '12, MICRO'18]

Fuzzy Timers [ISCA '12]

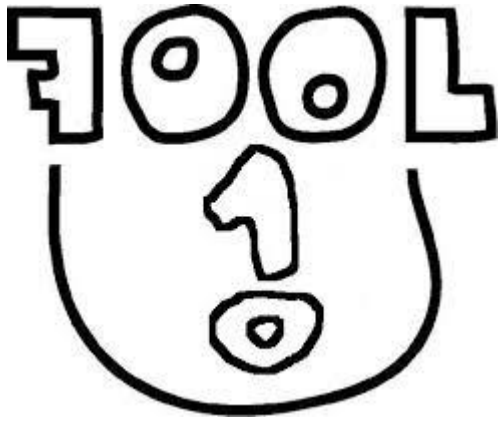
Random Permutation Cache [ISCA '07]

Secure Cache Hierarchy [DAC '17]

Secure Cache Replacement Policy [ISCA '17]

Depends on ISA changes, OS support, and runtime support

Our goal: Minimal changes to hardware only ☺



Fooling the Sense of a Cross-Core Eviction based Attacker by Prefetching the Common Sense

Fooling the Sense

Fooling the sense of an attacker:

time(LLC hit) = time(LLC miss)

How?

Hardware Prefetching

But, prefetch what?

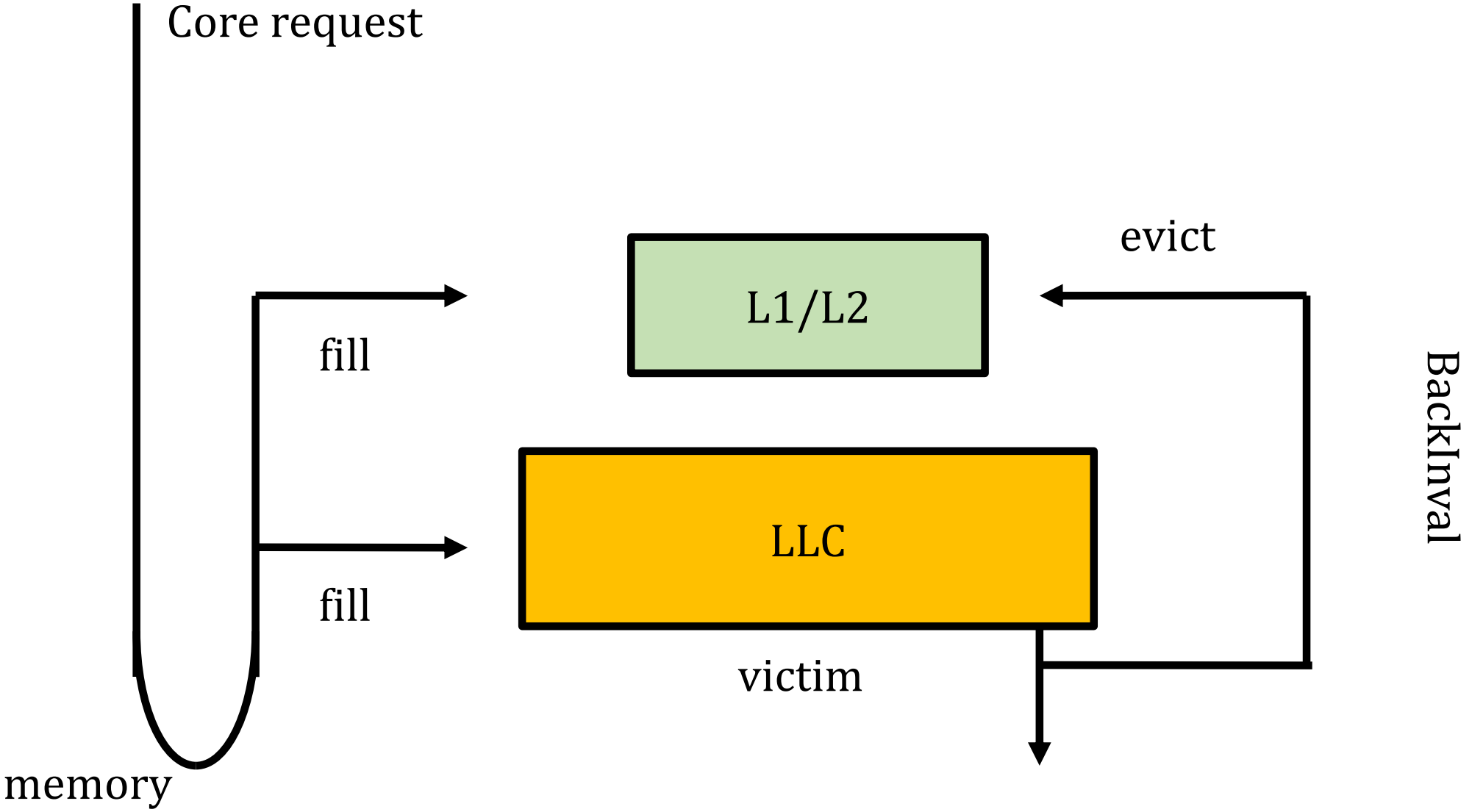
Our Thesis

(1) All the eviction attacks target inclusive LLCs

(2) Inclusion Victims (back-invalidations) help the attackers

(3) Cross-core back-invalidations are rare and benign

Inclusive Caches



Our Thesis

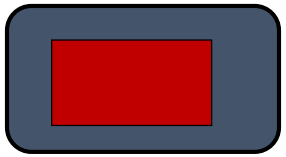
(1) All the eviction attacks target inclusive LLCs

(2) Inclusion Victims (back-invalidations) help the attackers

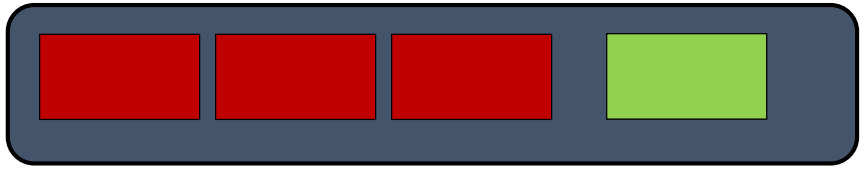
(3) Cross-core back-invalidations are rare and benign

Let’s Revisit the Eviction Attacks

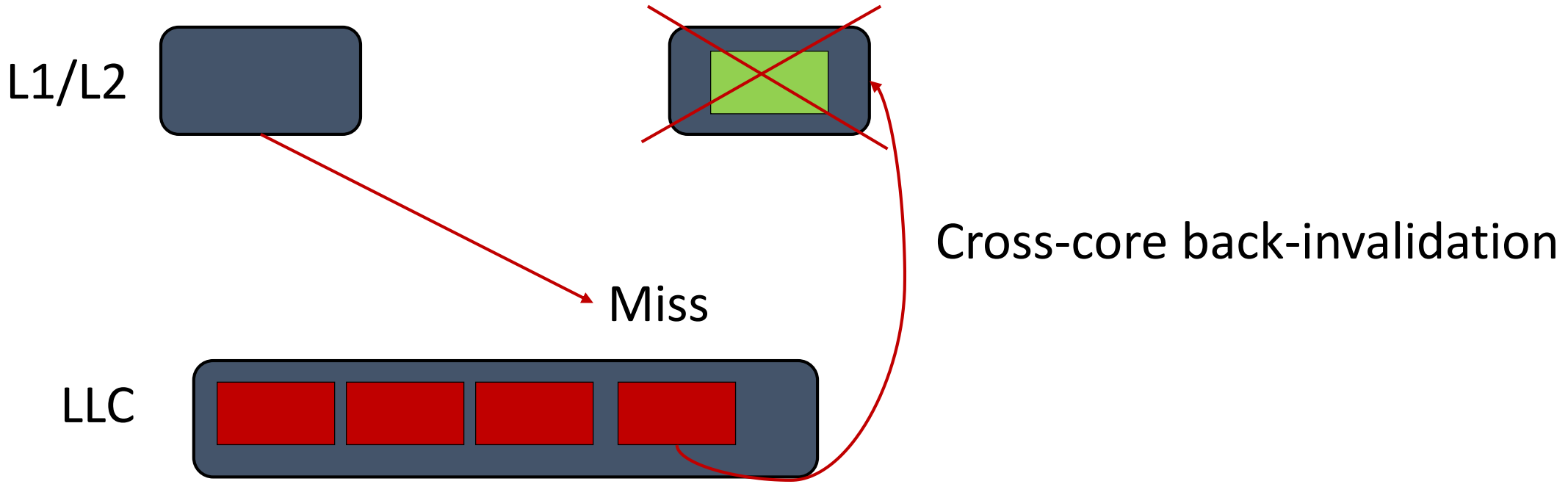
L1/L2



LLC



Let’s Revisit the Eviction Attacks



Let's Revisit the Eviction Attacks

Attacker knows whether victim has accessed a set or not

Miss

L1/L2

Miss

LLC

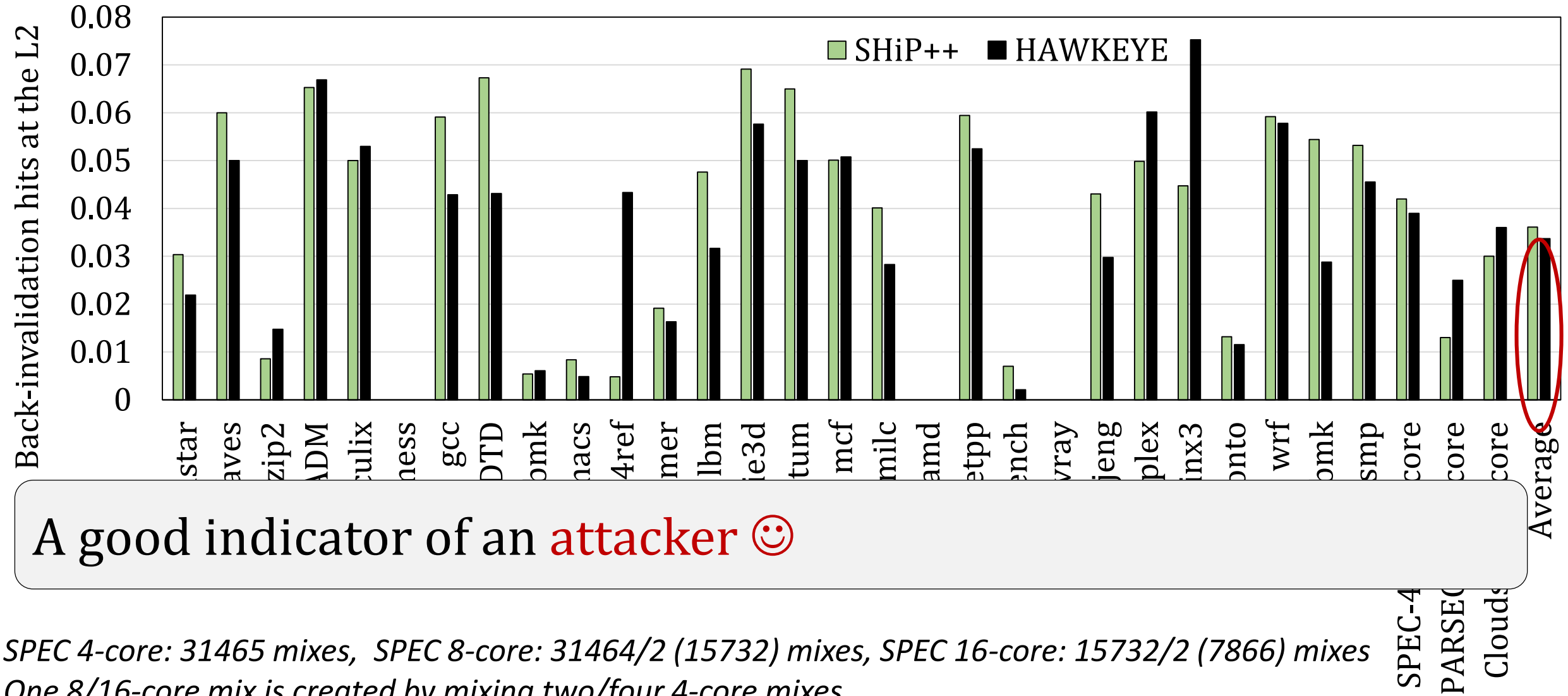
Our Thesis

(1) All the eviction attacks target inclusive LLCs

(2) Inclusion Victims (back-invalidations) help the attackers

(3) Cross-core back-invalidations are rare and benign

Let's Quantify it for Benign Applications



A good indicator of an attacker ☺

SPEC 4-core: 31465 mixes, SPEC 8-core: 31464/2 (15732) mixes, SPEC 16-core: 15732/2 (7866) mixes
One 8/16-core mix is created by mixing two/four 4-core mixes.

Fooling the Sense of a Cross-Core Eviction based Attacker by Prefetching the Common Sense

Our Proposal: 10K Feet View

Fooling the Sense of the Attacker by Prefetching Common Sense

Fooling the sense of an attacker:
time(LLC hit) = time(LLC miss)

Common sense:
Prefetch the *back-invalidation hits (before attacker probes)*
Back-invalidation Triggered Prefetcher (BITP)

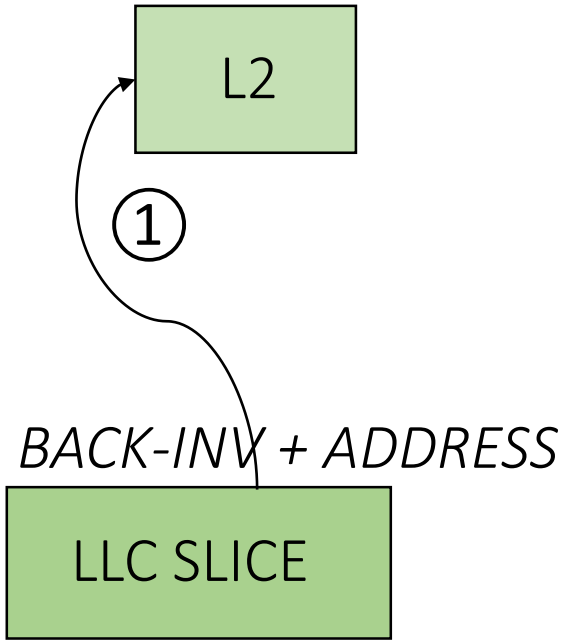
Why not Prevent Cross-Core Eviction?

SHARP [ISCA ‘17] does the same,
affects the replacement priority chain

Significant performance degradation with state-of-the-art replacement policies and many issues [WOOT@USENIX ‘19]

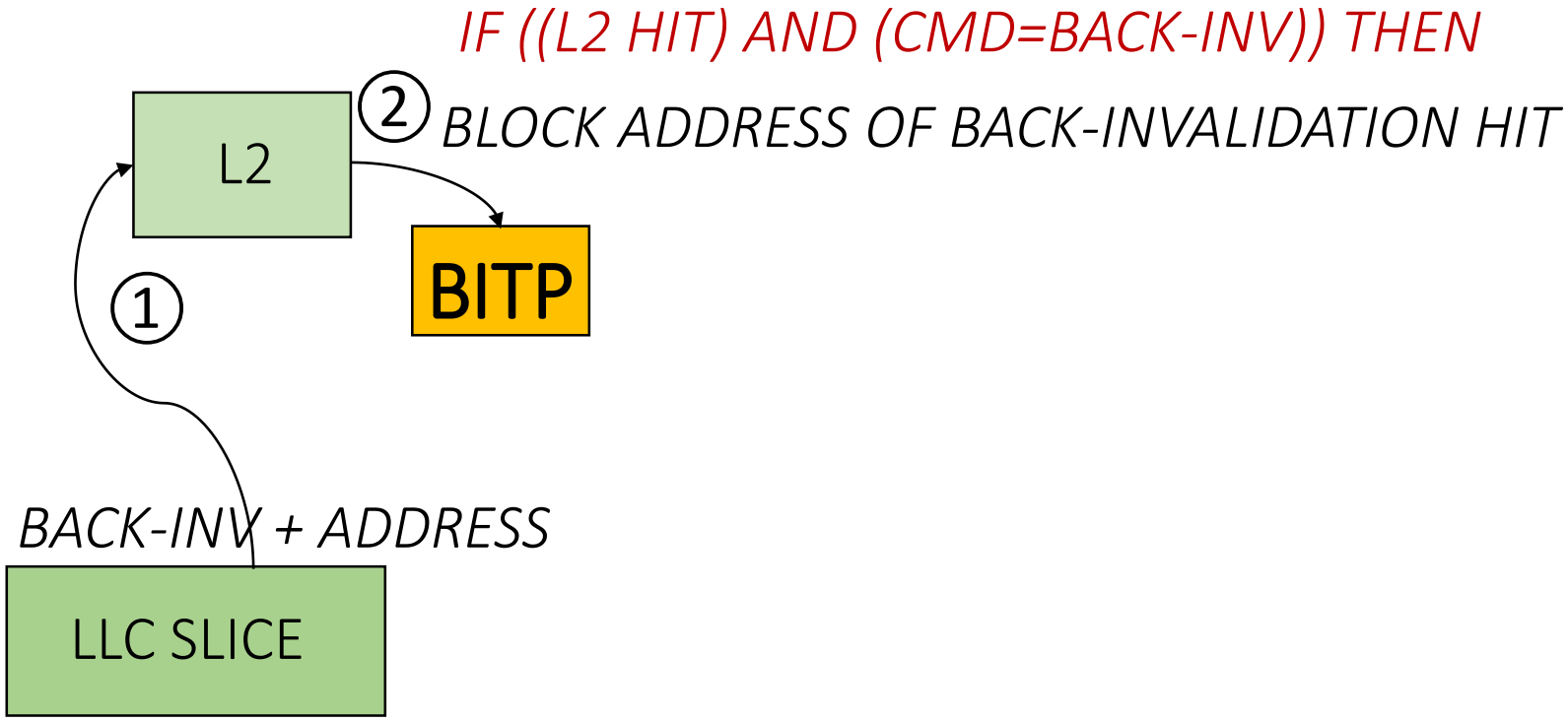


Our Proposal at the Per-core L2 Level



BACK-INV: BACK-INVALIDATION COMMAND

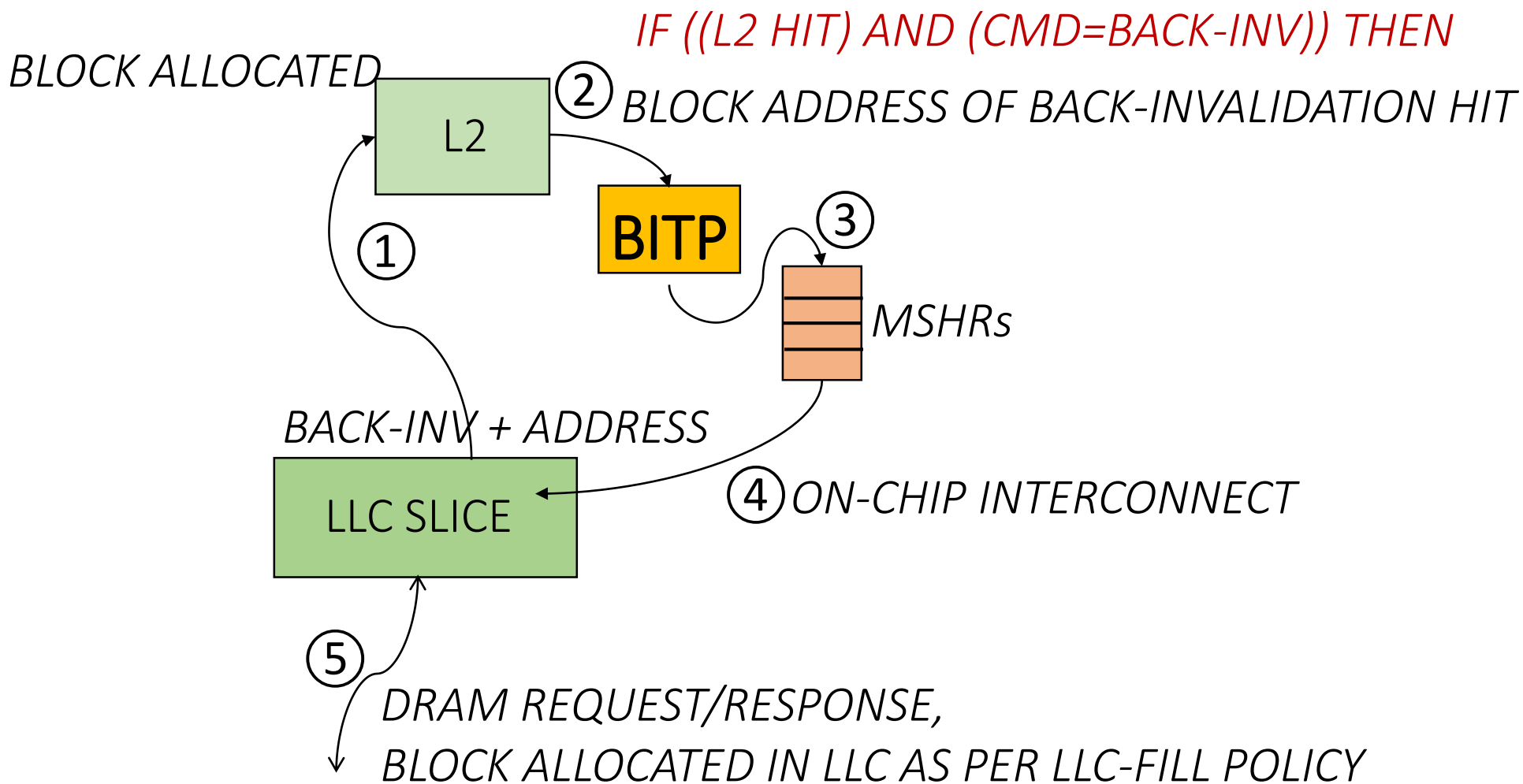
Our Proposal at the Per-core L2 Level



CMD: COMMAND

BACK-INV: BACK-INVALIDATION COMMAND

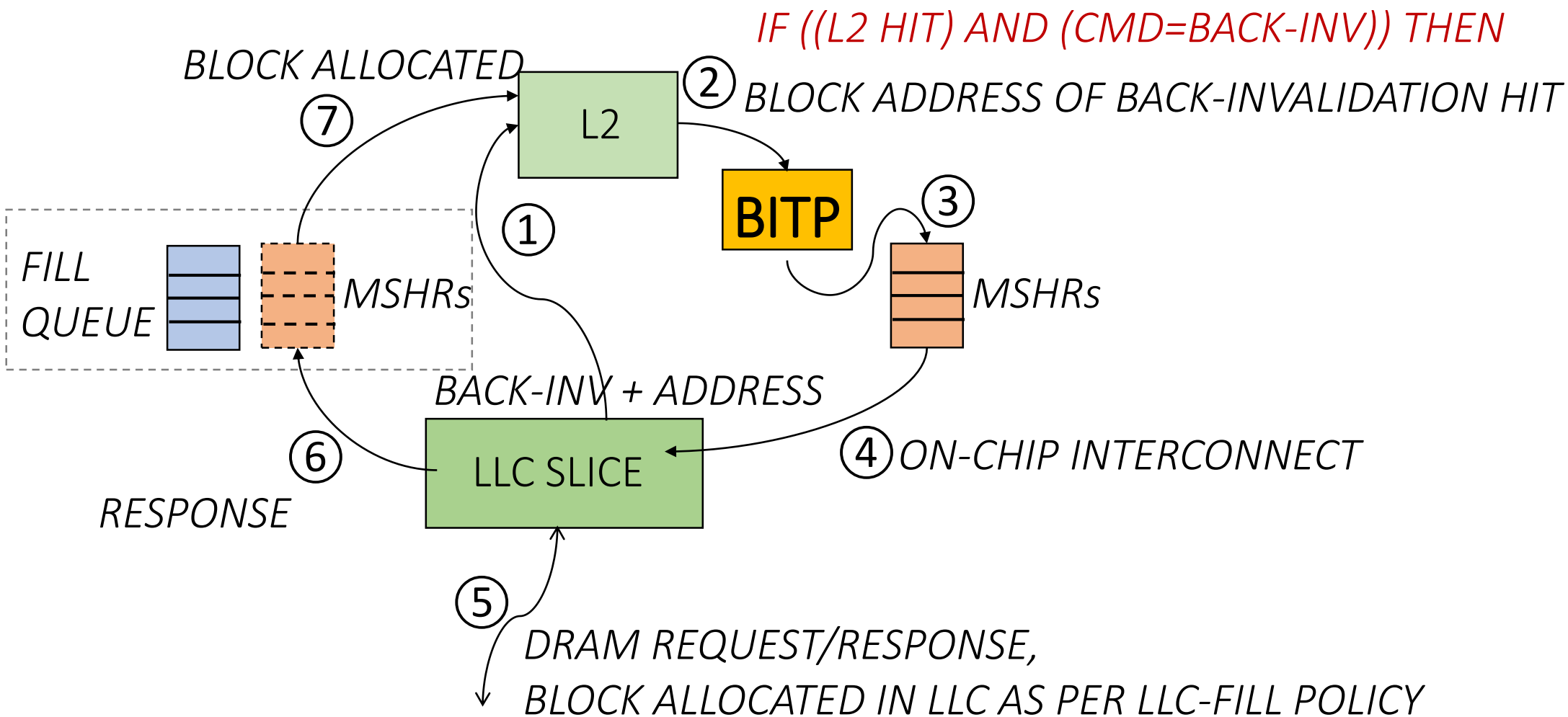
Our Proposal at the Per-core L2 Level



CMD: COMMAND

BACK-INV: BACK-INVALIDATION COMMAND

Our Proposal at the Per-core L2 Level



CMD: COMMAND

BACK-INV: BACK-INVALIDATION COMMAND

BITP

BITP works independent of the core/thread id

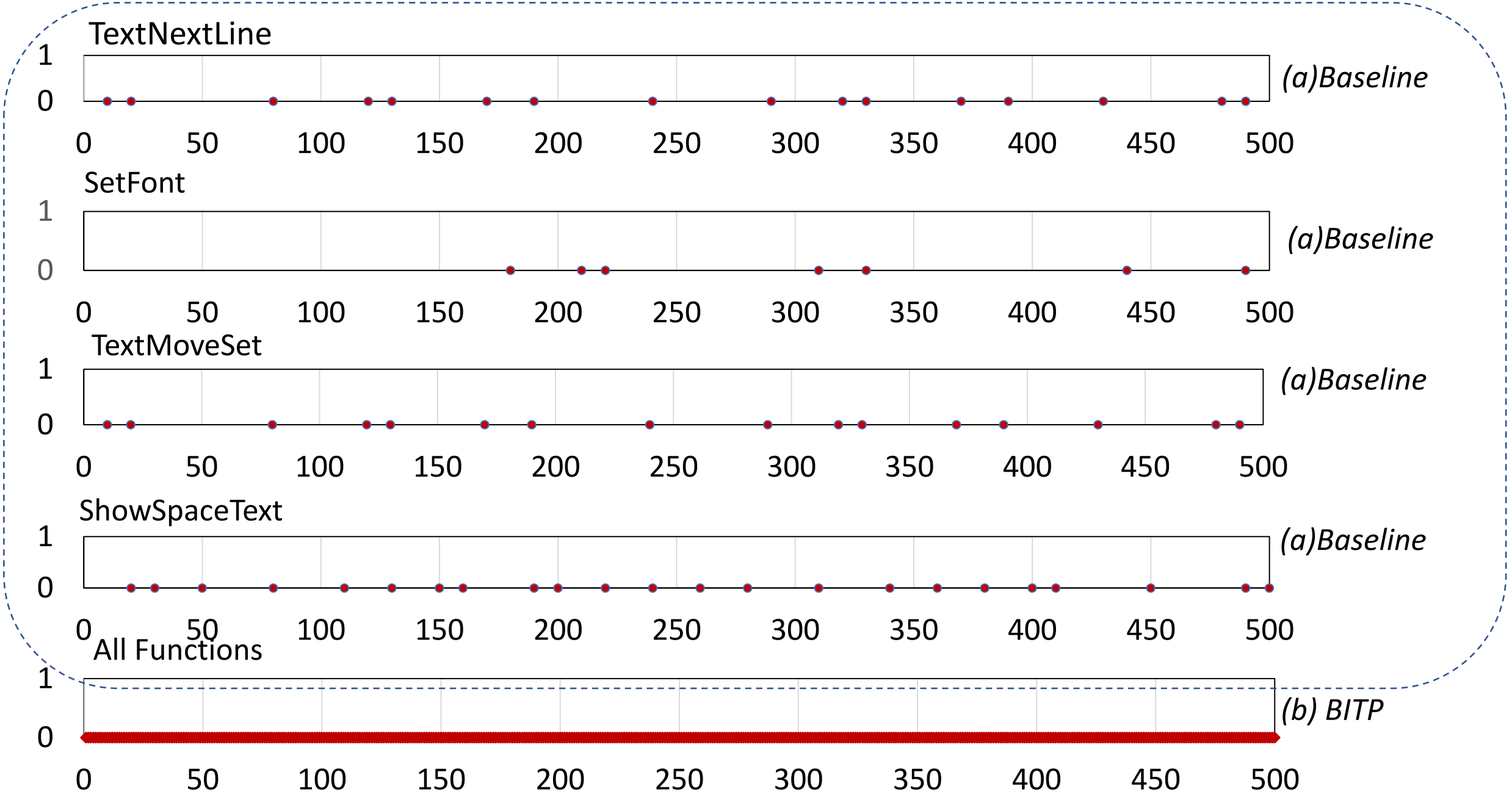
Thread migration does not affect BITP

Works irrespective of LLC replacement policy

Does not affect the priority chain

No support from OS/compiler/runtime system/ISA ☺

A quick security evaluation:
BITP on Poppler (a PDF rendering library)

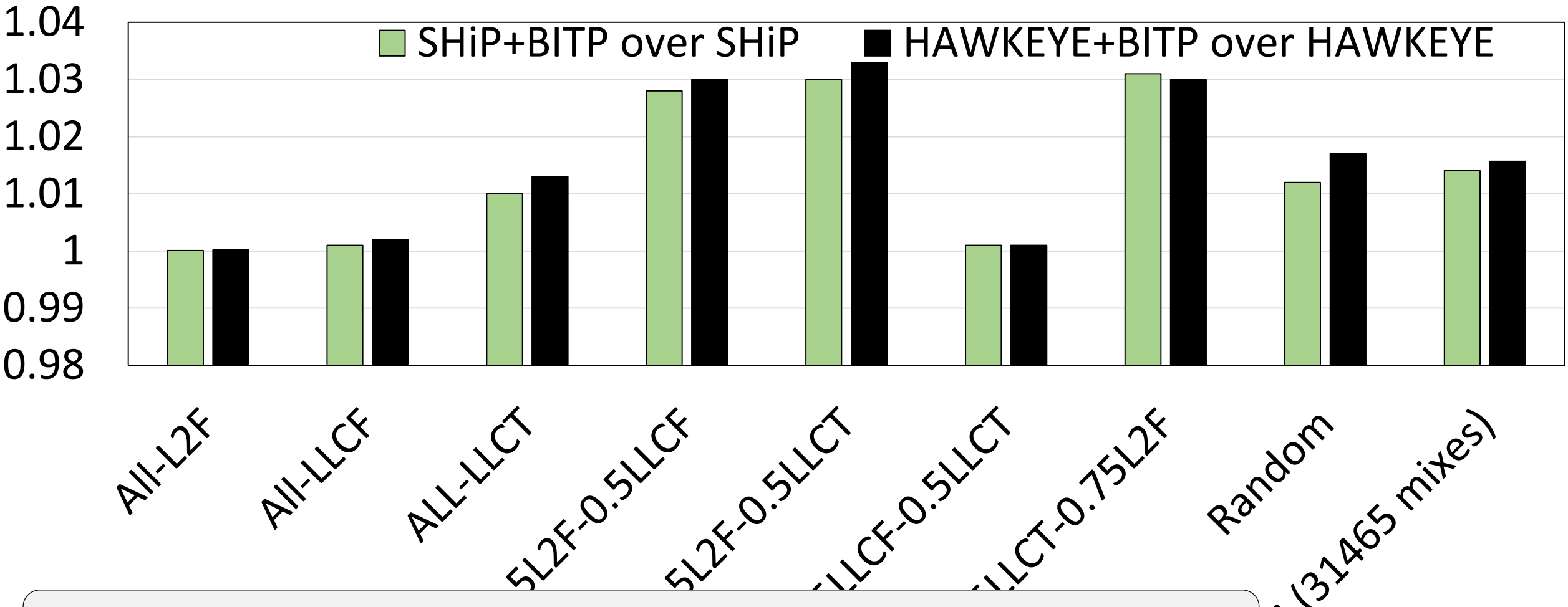

TextNextLine
1
0
0 50 100 150 200 250 300 350 400 450 500
(a)Baseline
SetFont
1
0
0 50 100 150 200 250 300 350 400 450 500
(a)Baseline
TextMoveSet
1
0
0 50 100 150 200 250 300 350 400 450 500
(a)Baseline
ShowSpaceText
1
0
0 50 100 150 200 250 300 350 400 450 500
(a)Baseline
All Functions
1
0
0 50 100 150 200 250 300 350 400 450 500
(b) BITP

No More Inclusive LLCs ☺ So BITP ☹

Yes, recent machines have employed non-inclusive LLCs

However, Coherence Directory is inclusive even if the LLC is non-inclusive [S&P '19]: Intel and AMD machines, So BITP ☺

Effect on System Performance



Marginal performance gain: win-win situation ☺

Key Takeaways

Prefetcher can fool a cross-core LLC attacker

Prefetching on Back-invalidation hits

~zero hardware overhead and no performance loss

Key Takeaways

Hardware Prefetching: ***(Mitigating cross-core timing channels)***

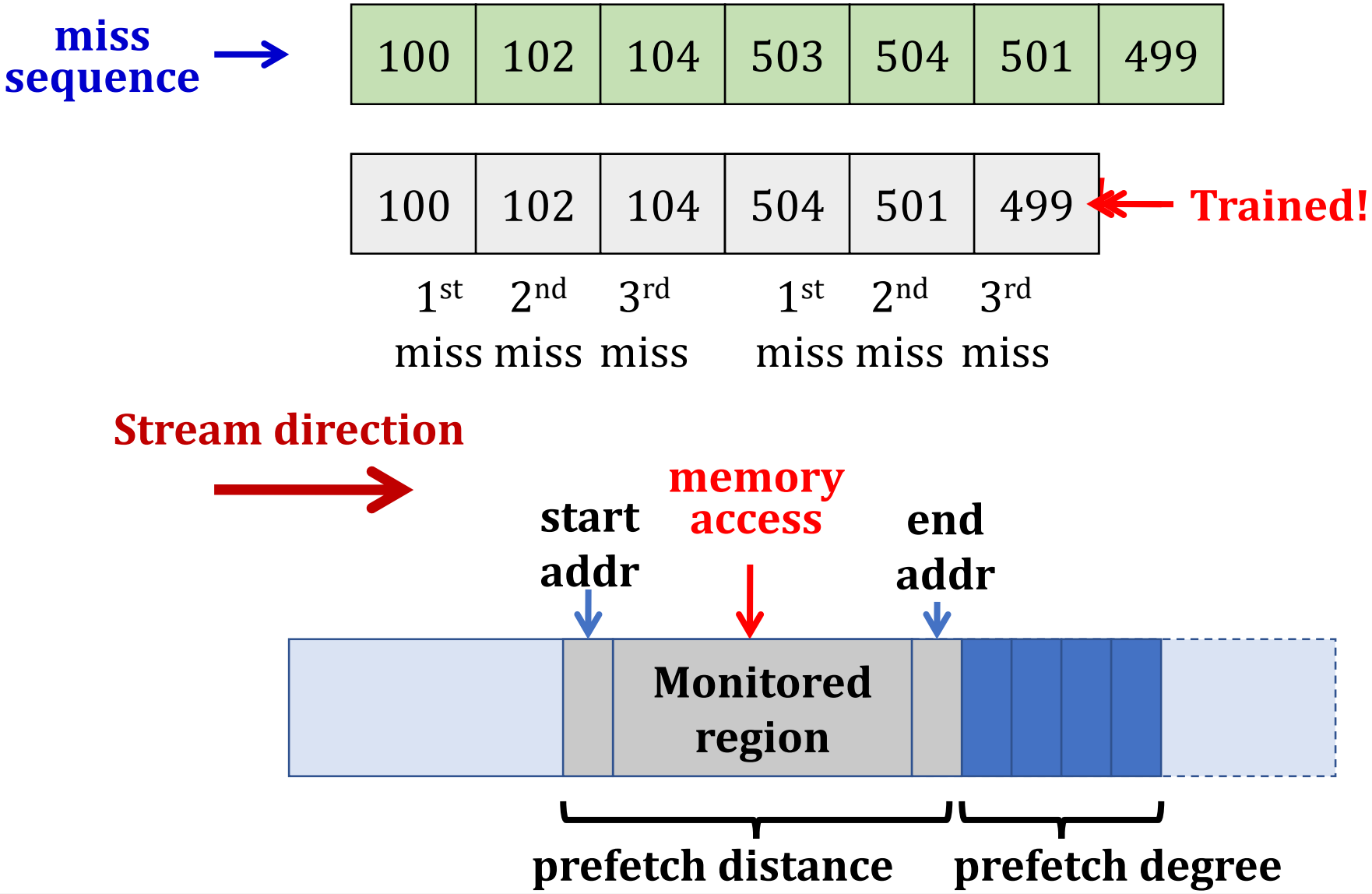
Back-invalidation Triggered Prefetching [PACT '19]

Hardware Prefetching: ***(Creating a cross-thread timing channel)***

Whispering Streamers [HASP@ISCA '19, Poster@MICRO '19]

Whispering Streamers: [Adi, Biswa, and Prakhar HASP@ISCA ‘19 and Poster@MICRO ’19]

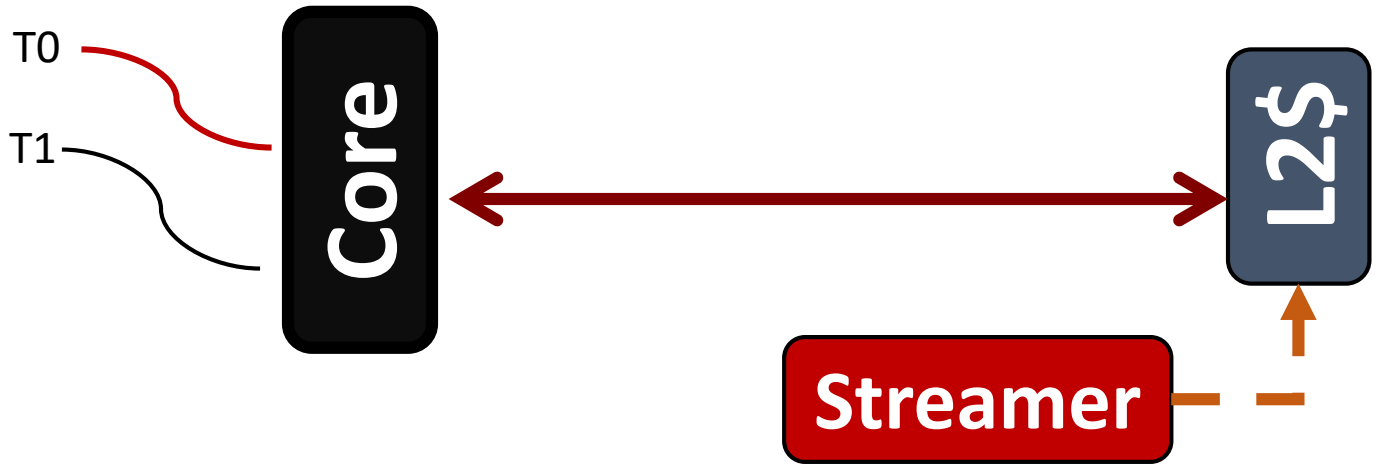
Streamers: What We Know?



Streamers: Our Hunch

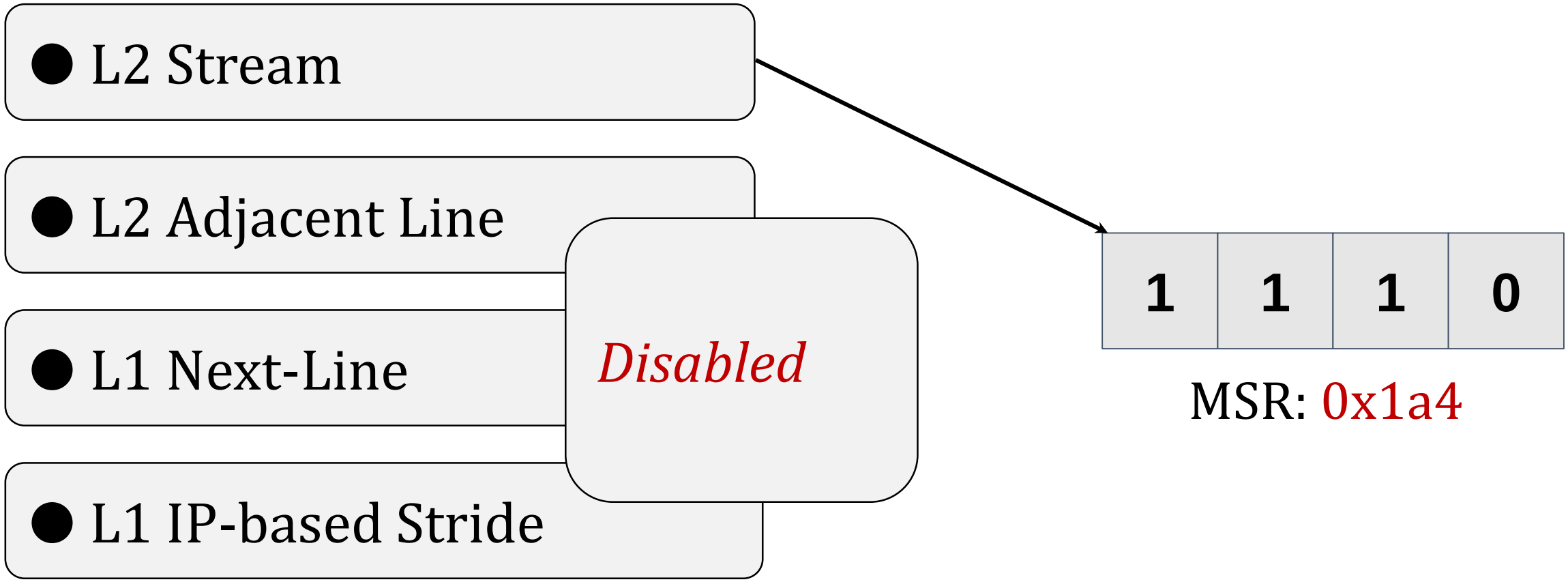
Streamers are shared between hyper threads

A cross-thread covert-channel in an SMT core



What We Do not Know? Many ☹

Other Prefetchers in Intel Machines



Our Goal: What Are We Trying to Answer?

Is the Stream Prefetcher shared b/w SMT threads?

What triggers the stream prefetcher with cross-threads?

Does Streamer Trigger Multiple Prefetching?

Can the Streamer leak?

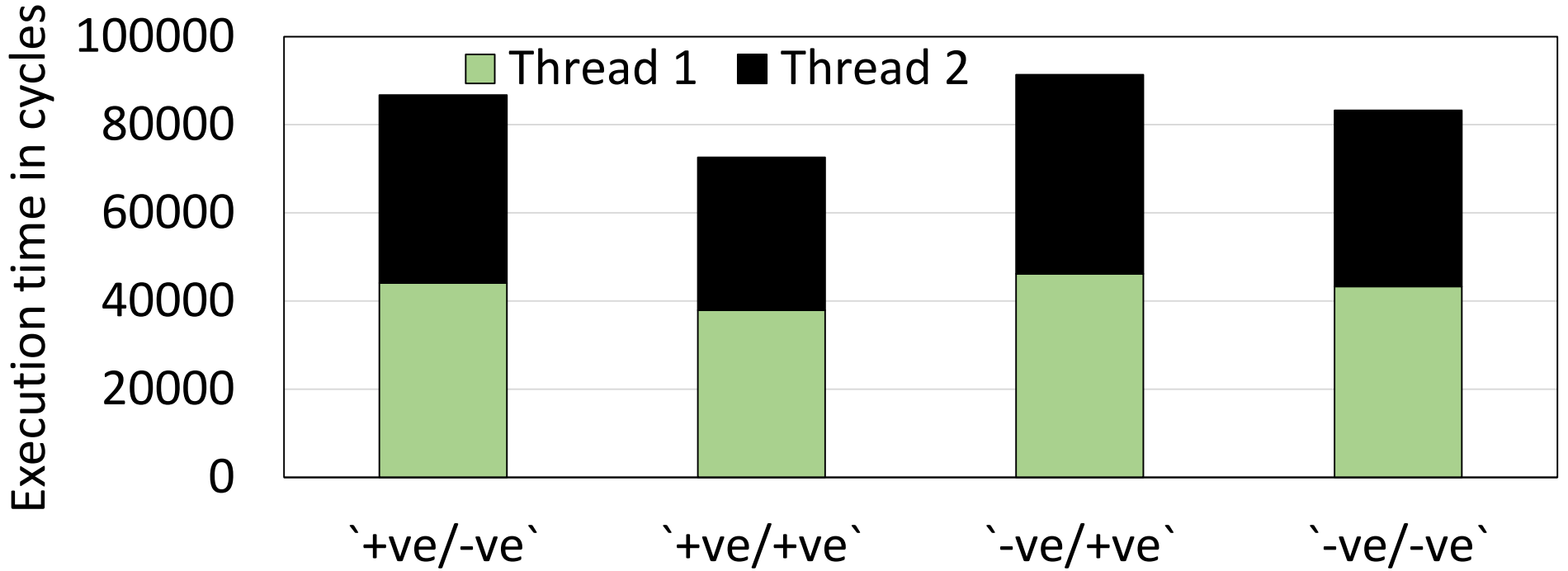
Our Setup and Assumptions

Two SMT threads on an isolated core accessing 4KB OS pages

All prefetchers except the streamer disabled

Intel Kaby Lake i5, 4 cores (SMT), 64KB L1, 256KB L2, 1.5MB L3/core

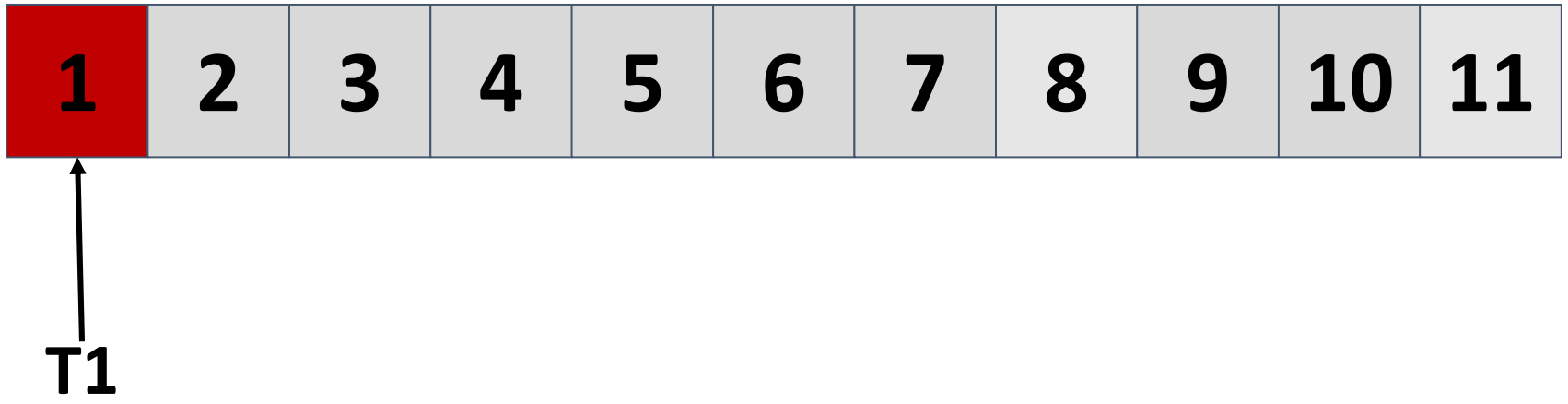
Experiment 1: Is the Streamer Shared?



Accesses made to OS pages covering a large array of 4MB

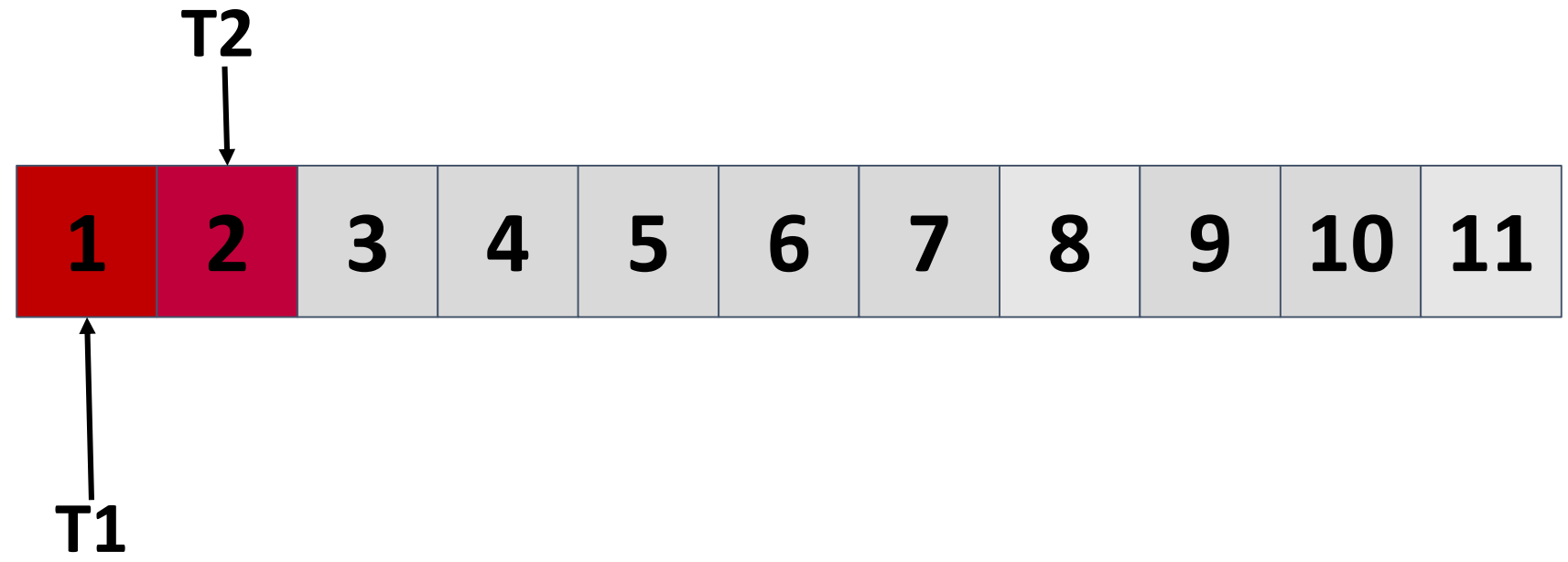
Insight: Stream table entry is shared b/w SMT threads

Experiment 2: How does the Streamer get Triggered?



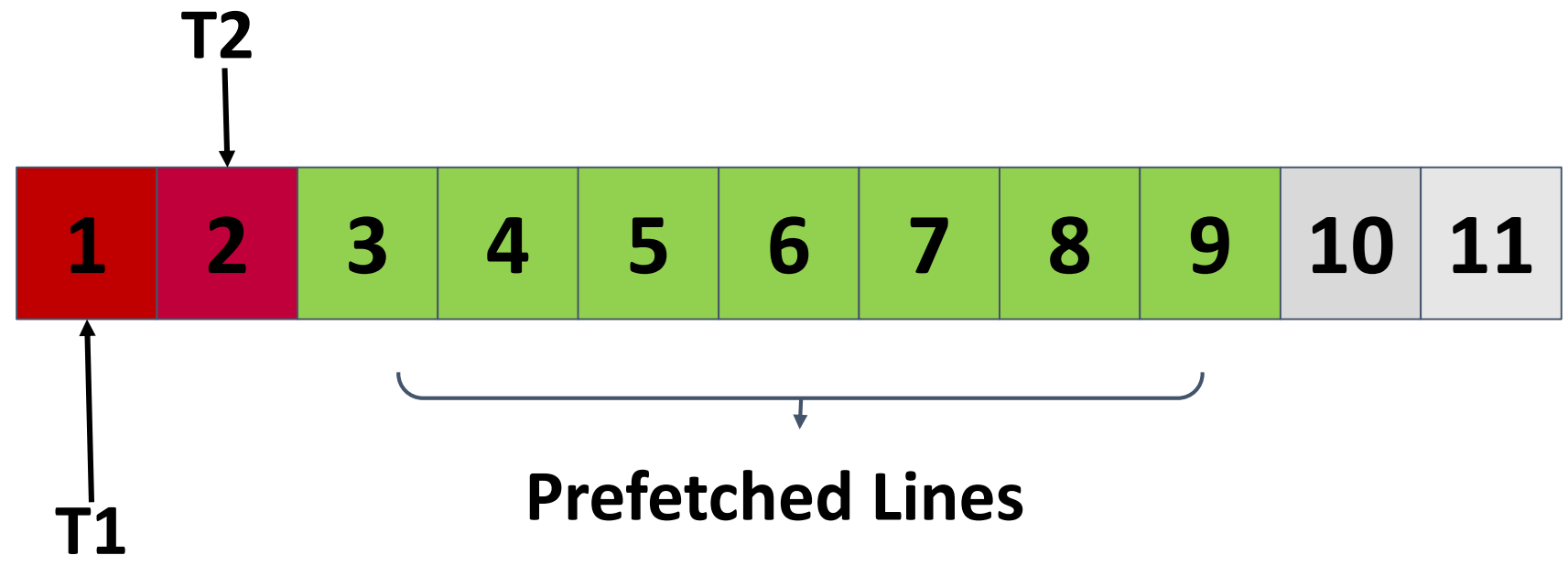
Thread1 accesses line1

Experiment 2: How does the Streamer get Triggered?



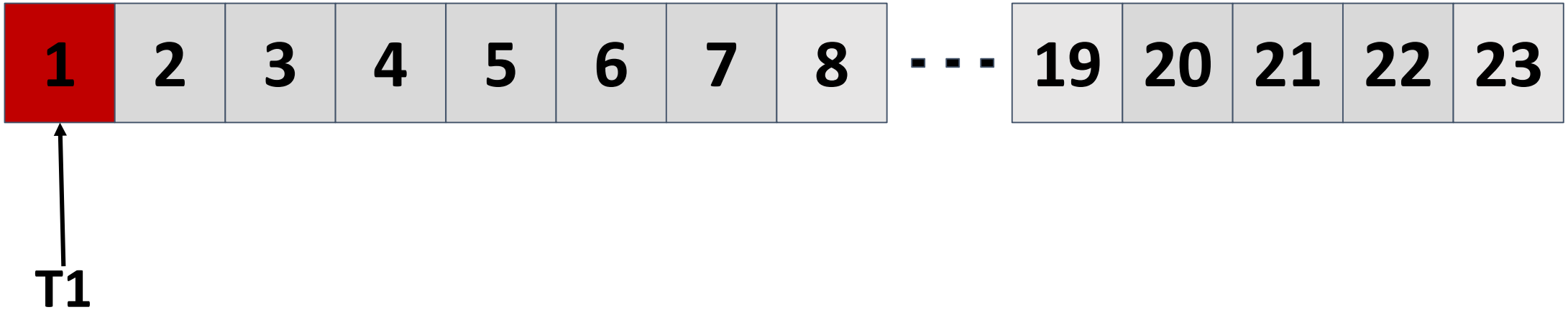
Trigger : Thread2 accesses line2

Experiment 2: How does the Streamer get Triggered?



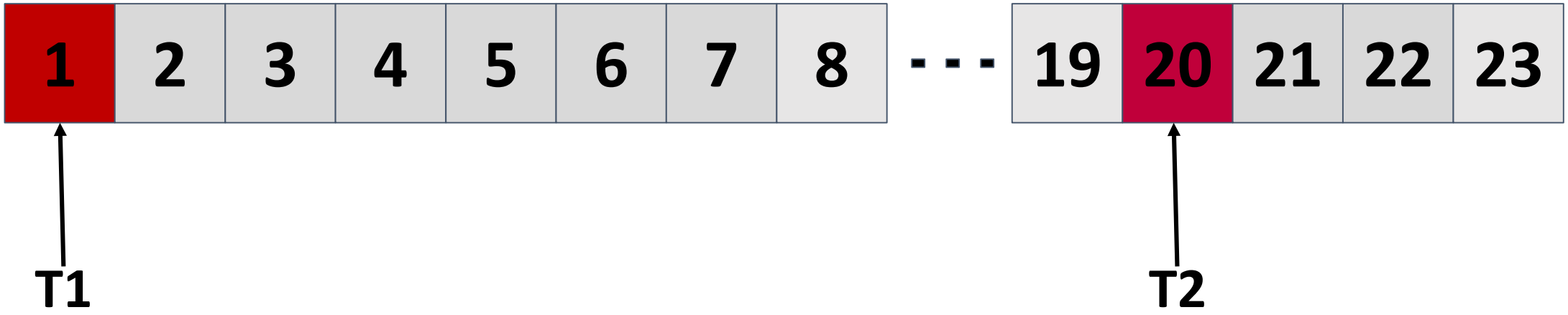
Insight: Two misses (even accesses)
can trigger the streamer, prefetch degree varies from four to eight

Experiment 3: Does Streamer Trigger Multiple Prefetching?



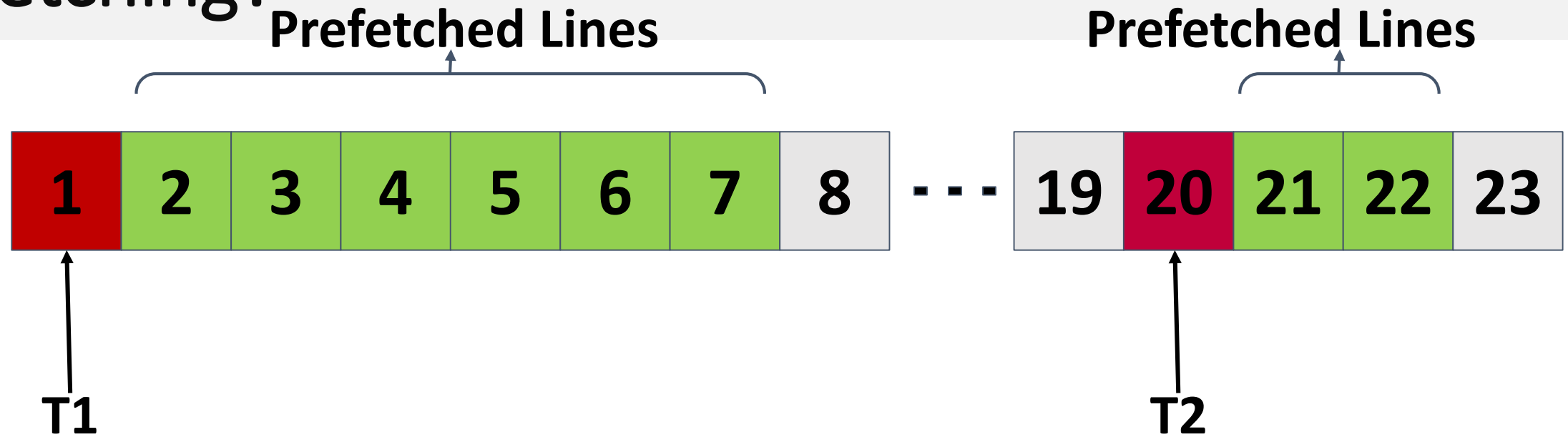
Thread1 accesses line1

Experiment 3: Does Streamer Trigger Multiple Prefetching?



Trigger : Thread2 accesses line20

Experiment 3: Does Streamer Trigger Multiple Prefetching?



Insight: Prefetching happens at the trigger and the previous access

So Where Are We?

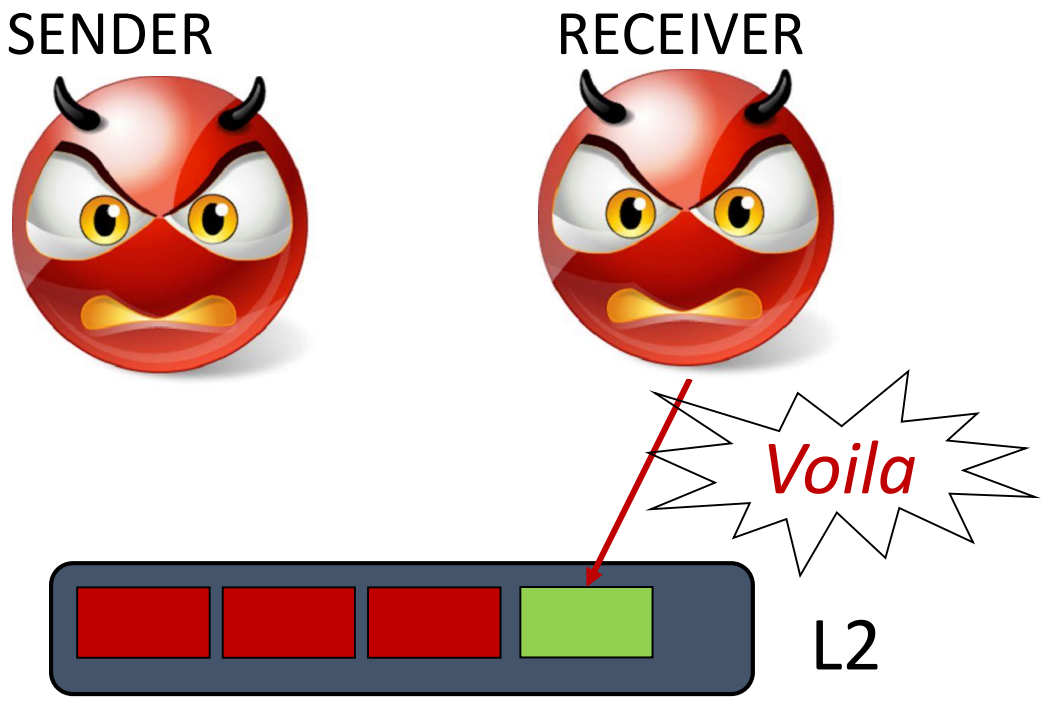
Shared b/w SMT threads on the same core

Two accesses can trigger the streamer

Prefetching happens at trigger and the previous access

Can the Streamer leak?

Flush+Reload For Covert Channel: 101



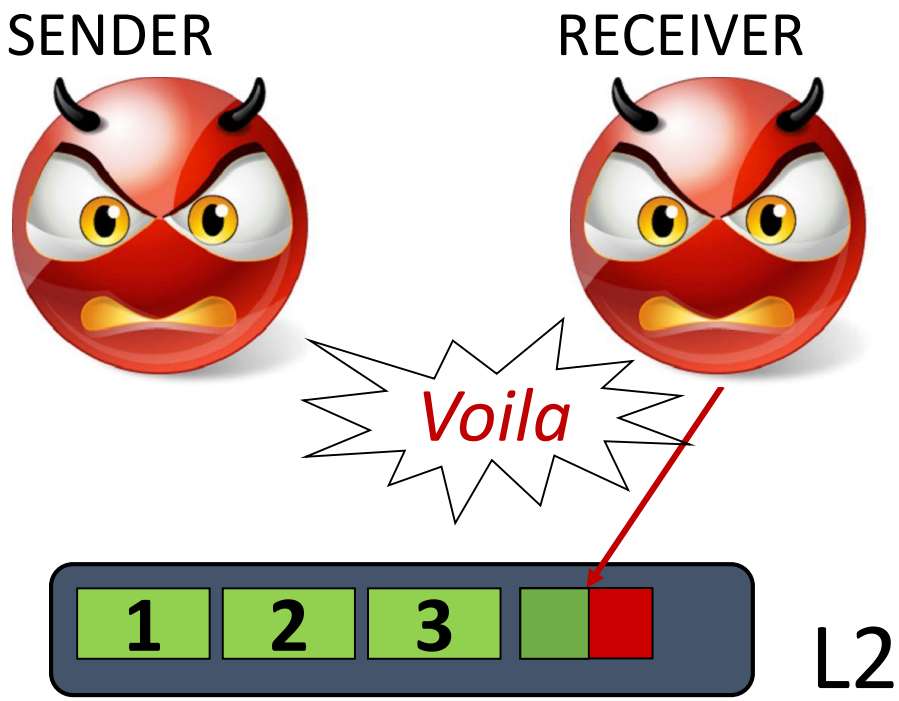
Step 0: Sender and receiver agree on a few cache sets

Step 1: Sender *does not flush/flushes* cache sets while running

Step 2: Receiver *reloads* the cache sets

Low latency: 1,
high latency: 0

Our Proposal: Flush+Prefetch+Reload



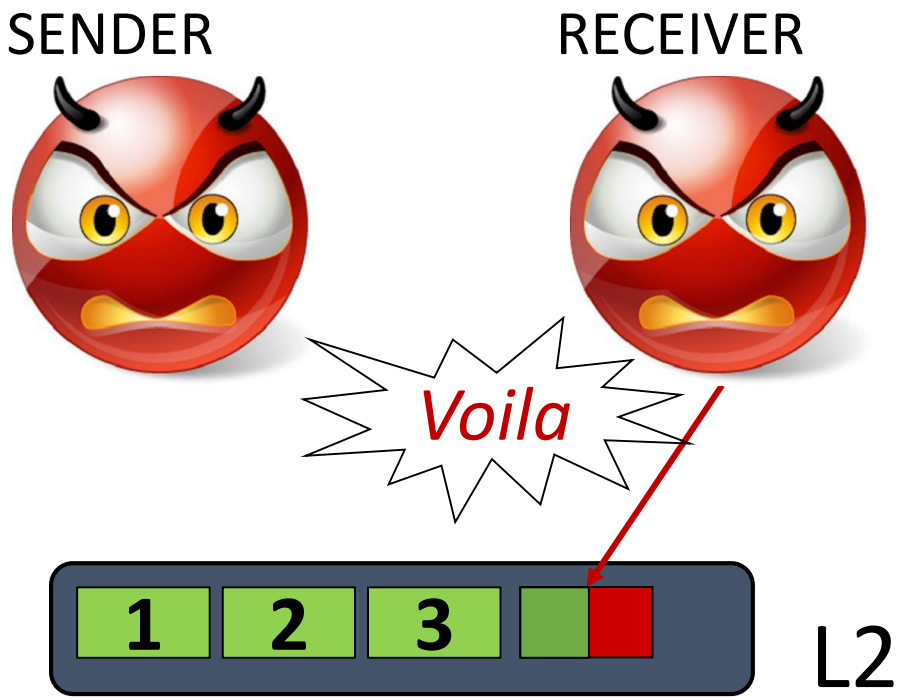
Step 0: Sender *flushes* all cache lines in a page

Step 1: Sender accesses ***line1***, ***line2***, and ***line3*** lines of the page

Step 2: Receiver accesses ***line4*** of the same page

Hit means sender sent bit 1,
Miss bit 0

Our Proposal: Flush+Prefetch+Reload



Bandwidth ***13.49 KB/s***

Accuracy ***91.3%***

So Where Are We?

Shared b/w SMT threads on the same core

Two accesses can trigger the streamer

Prefetching happens at trigger and the previous access

Yes! as a covert channel and possibly as a side channel

Takeaways

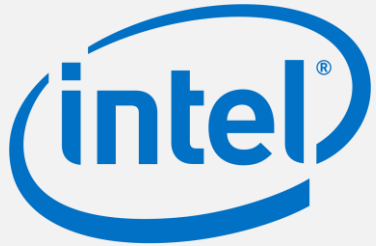
Prefetchers can mitigate eviction based attacks

Existing prefetchers can be leaky too

What next: Side channel attack using Streamer, stay tuned, may be MAST ‘20 ☺

"It takes two to speak the truth - one to speak and another to hear" - Henry David Thoreau

Thank You



Wanna Join Us (CARS@CSE-IITK)?

This could be you