NDN-NIC: Name-based Filtering on Network Interface Card

Junxiao Shi, Teng Liang, Hao Wu, Bin Liu, Beichuan Zhang

Published in ICN '16 Proceedings of the 3rd ACM Conference on Information-Centric Networking

황인찬

2016. 10. 03

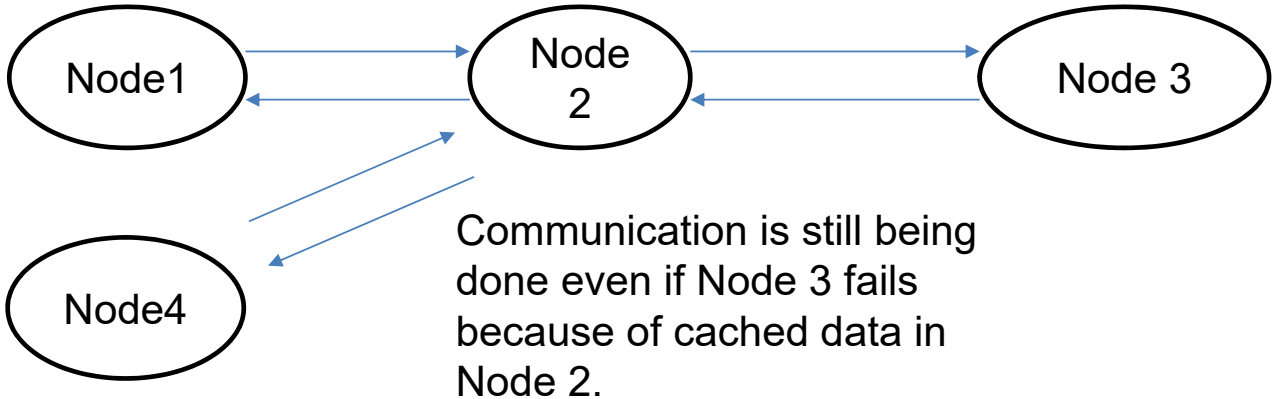
Introduction

- In named data networking, consumers(clients) send Interests to get data from the producer(server).
- Traditionally, networking is a host-to-host conversation, focusing on a node itself.

  But, NDN emphasizes on data.
- It is known to work better in content distribution, security, mobility support, and application development.
- However, current NIC cards, having small memory, filter packets based on destination addresses.
- Thus, name-based filtering is done by software. There is overhead at CPU.
- This paper takes advantage of Bloom filter to enable name based filtering at NIC.

What is named data networking?

- It was originally announced by Van Jacobson in 2009.
- NDN names nodes and data packets at the network layer.
- Its network paradigm is not session based, but all communication in NDN is local, operating in hop-by-hop fashion.



NDN data structure

- To make NDN communication possible, Each node must maintain CS, PIT, FIB data structures.
  - FIB forwards Interest packets to the sources of data
  - CS caches all data packets passing by each node
  - PIT keeps track of Interests forwarded upstream to the content source. Returned data packets will trace this PIT to travel to the consumer.

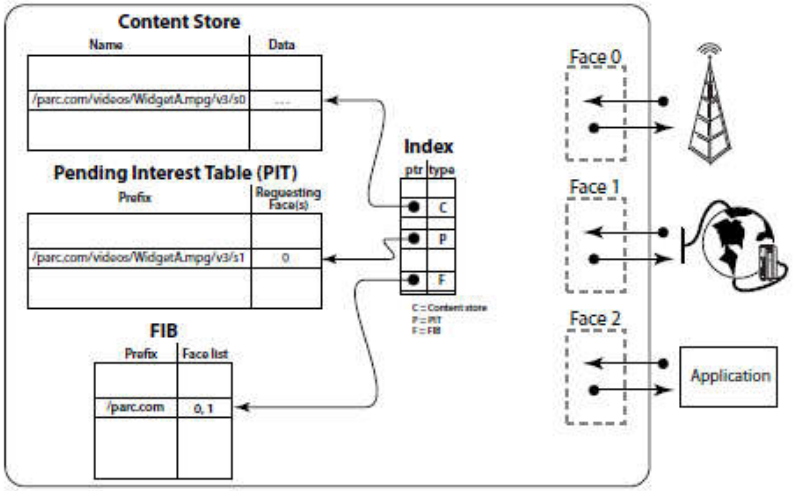


Figure 3: CCN forwarding engine model

- NDN name in each data structure is human readable character. They are longer than the traditional IP addresses. One of the obstacles to implement NDN over the current network infrastructure is this fact. NDN names do not fit into the small memory of a network device. Thus, software filtering causes a lot of power consumption and CPU overhead.

Overview

- Traditionally, packet filtering at the NIC card based on the destination MAC address. But NDN does not carry those data in packets. Currently, NDN software performs name based filtering.
- A typical NDN end host has many name rulesets due to the content it serves.
- Exploitation of Bloom filter will allow a NIC card to store those name rulesets to filter out irrelevant NDN packets to reduce CPU usage and power consumption.
- NIC finds a match between the destination MAC address of an incoming packet and a list of acceptable unicast and multicast addresses. If there is a exact match, then the packet will be accepted.

Name Matching in NDN

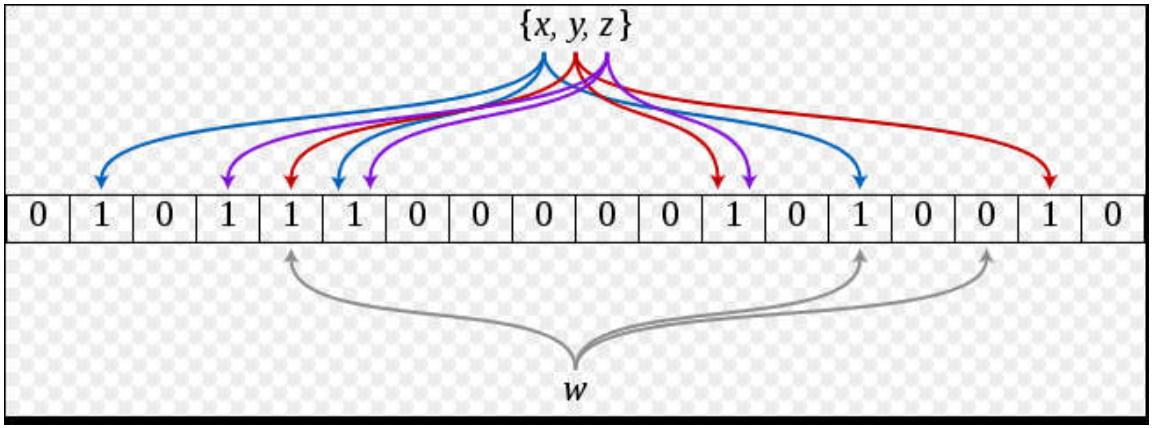
- Both Interest and data carry content names. Data satisfy an Interest if the Interest name is the same as the data name or a prefix of the data name.
- NFD makes a forwarding decision based on the name rulesets stored in three tables.
  - If an Interest is looking for "/A/b" in CS, it could match "/A/b, /A/b/1, A/b/1/2, /A/b/3." It made multiple matching results. Then, the best match will be returned to the consumer.
  - NFD inserts Interest name into the PIT if there is no exact match in it.
  - Next, NFD performs longest prefix match on FIB, the results could be "/A/b, /A, /."
  - If there is not a match, it is dropped.

Name Matching in NDN

- When a data packet with “/A/b/1” came to a node, NFD performs longest prefix match on PIT.
- Then, the match results are “/, /A, /A/b, /A/b/1” If there is no match, the packet is dropped.
- NFD puts the data into the CS, which involves an exact match look up.

Bloom filter

- Bloom filter is a bit array of m bits, all sets to 0. There are also K different hash functions defined.
- To add an element, K hash functions hash an element and get K array positions.
- Set the bits at all these positions to 1.
- To query an element, K hash functions hash an element, and produced array positions are queried. If those queried positions are 1. Then, the elements are in the set. But false positive can happen.
- In the picture, elements x, y, z are in the set. But element w is not.



Counting Bloom filter

- Bloom filter does not allow the removal of an element.
- Each array element will have “n” bit in size. When an element is added, and there is already 1 in the position. It increases the element value by 1.
- When a removal operation takes place, it subtracts the value by 1. When there is no value. The queried element is not a member.

Design Overviews

- IP hosts have a handful of unicast/multicast addresses whereas an NDN node has hundreds of FIB and PIT entries, and more number of CS entries.
- An average NIC has a few kilobytes of memory.
- IP addresses are stable in NIC card. PIT and CS change frequently.
- NDN NIC must be updated frequently. Its matching rules are different for each table.
- Bloom filter(BF) is the answer to save the entries in NIC.
- Even if a packet passes BF for false positive, NFD will drop the packet if its name is not present in NDN tables.

Design Overviews

- Bloom filter does not allow an element to be removed. But counting bloom filter will let an element to be deleted.
  - Every single bit is replaced with a multi-bit counter.
- Since there are different matching strategies between FIB, PIT, and CS, Three different BFs are made in NIC.
- NDN-NIC hardware filters NDN packets
- NDN-NIC driver will maintain three counting bloom filter arrays. They keep being uploaded to the NDN-NIC.
- FIB entry is added to the NIC when it has a nexthop.

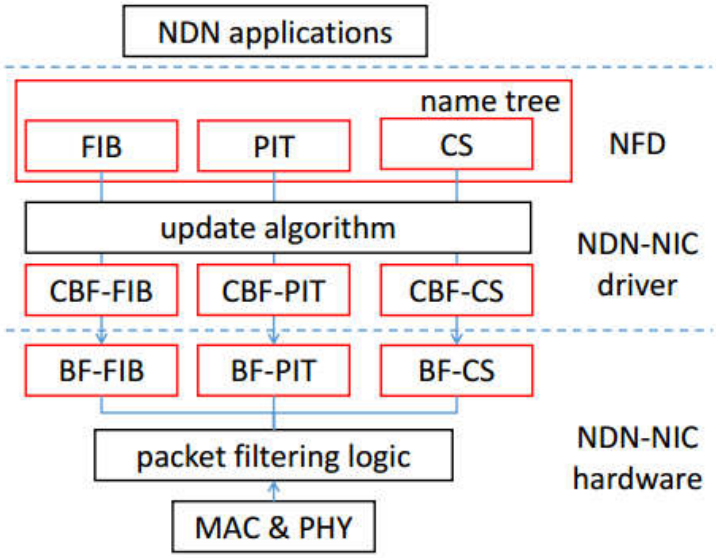


Figure 1: Overall architecture

Design overview

- PIT updates are done when Interests from the local consumers forwarded to the network are detected.
- Every CS entry name is added to the NIC.
- Updating algorithms, such as Direct mapping, Basic CS, Active CS are proposed to make changes to CBFs.
  - Direct mapping : adds every name in a table to the BF.
  - Basic CS and Active CS : reduce the number of names in the BFs.

Name Filter on Hardware

- When an Interest("/A/b") arrives, the NIC queries BF-FIB with all prefixes of the name ("/, /A, /A/b")
  - Queries BF-CS with the exact match, and BF-PIT with all the prefixes of the data name.
- H3 hash functions are used because it works in any hardware logic.
- The driver will update CBF as it is necessary. If a digit in CBF becomes zero, then it generates a new BF array and vice versa.
- Not every CBF update will result in BF update.

Direct Mapping

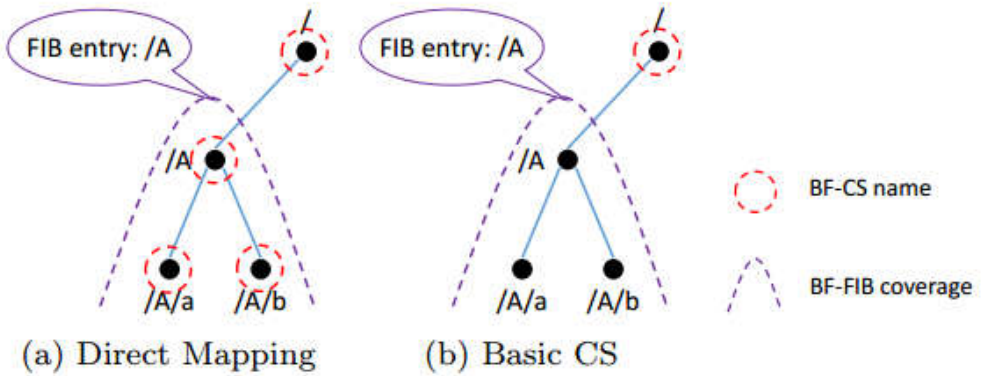
- Direct mapping is simple, FIB maps to BF-FIB, PIT maps to BF-PIT, CS maps to BF-CS.
- Filtering accuracy is high with 4.33% of false positive probability. But its value is increased as there are more entries.
- Hashing comes with CPU overhead. DM incurs high BF update overhead on hardware.
- Basic CS and active are proposed to add less names to BFs.
  - They look into CS because most names come from CS.

Basic CS

- When there is a FIB entry, /A, and two CS entries, /A/a and /A/b. Direct mapping (DM) add /A to BF-FIB, and add /A/a, /A/b, /A, / to BF-CS.
- Basic CS adds / to BF-CS. All other Interest names are detected by BF-FIB.
  - Then, the packet will be forwarded to NFD in the software.
  - It is effective when data come from the local producer.
- When a producer quits, its FIB entry is removed.

  Data from it is valid in the cache.

  CS entry names under the former FIB prefix must be readded to the BF-CS.



(a) Direct Mapping (b) Basic CS

Figure 2: BF-CS usage, DM vs Basic CS

Active CS

- Active CS creates appropriate prefixes into BF-FIB so that BF-CS usage is reduced.
- Transformation : it replaces a single BF-CS name with a BF-FIB prefix.
- Aggregation : it aggregates multiple BF-FIB prefixes up to a shorter prefix in BF-FIB.
  - Transformation reduces BF-CS use. It increases BF-FIB use.
  - Aggregation reduces BF-FIB use.

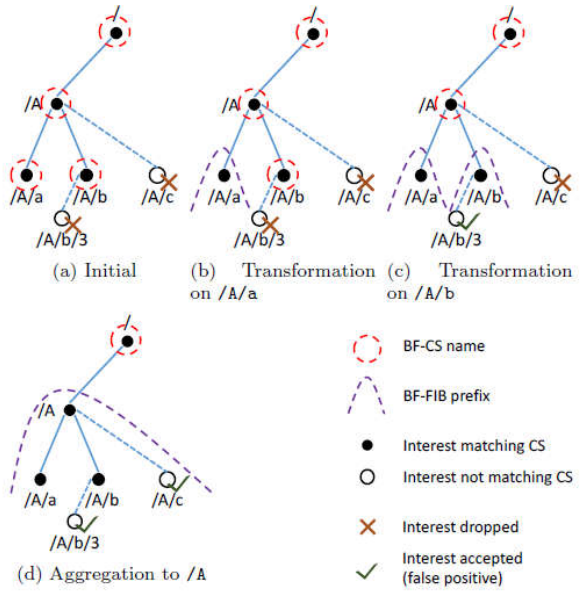
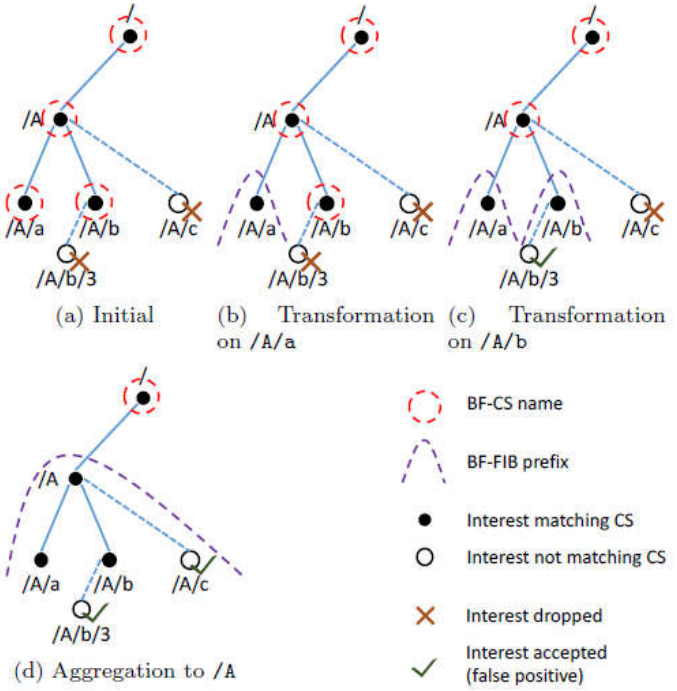


Figure 3: Two basic operations in Active CS

Active CS

- Figure illustrates a CS with two entries /A/a and /A/b. Transformation on those names will remove two names from BF-CS, and add two prefixes to BF-FIB.
- Aggregation to /A replaces two BF-FIB prefixes with one shorter prefix in BF-FIB.
- All CS entry names and their prefixes are matching at least one bloom filter.
- Interests /A/b/3 and /A/c would match the newly added BF-FIB prefixes /A/b and /A.
- The Interests will be dropped because there is no match in CS entry.



(a) Initial (b) Transformation on /A/a (c) Transformation on /A/b

(d) Aggregation to /A

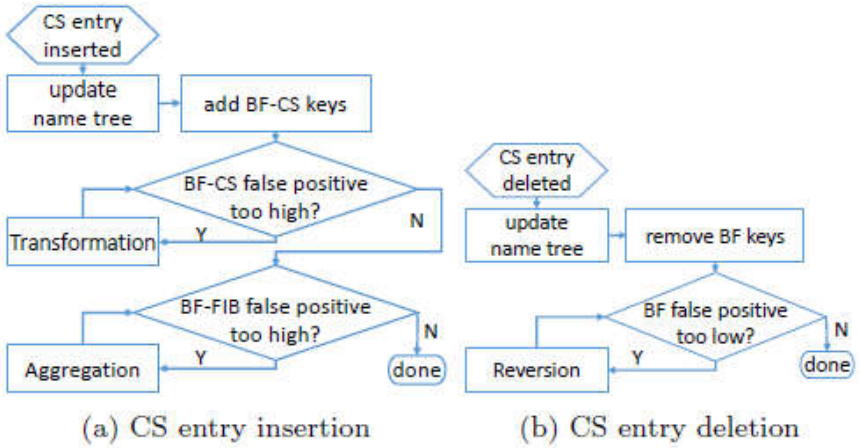
Figure 3: Two basic operations in Active CS

Active CS strategies

- Because false positive increases, it imposes an upper bound threshold on BF-CS/BF-FIB false positive, keep as many names in BF-CS/BF-FIB as allowed, does not do Transformation or Aggregation until BF false positives are too high.
- A name tree node is eligible for Transformation if its name is in BF-CS. A name tree node is eligible as a target of Aggregation if it has two or more descendants whose names are in BF-FIB.
- When BF-CS/BF-FIB false positive probability is below a lower bound, we undo the two operations to reduce prefix match false positives.

Active CS flowchart

- When a CS entry is inserted,
  the name tree is updated.
  Those names not matched by BF-CS
  and BF-FIB are added to BF-CS.
- If BF's false positive probability is over
  the upper bound threshold,
  Transformation and Aggregation are done
  until they fall below the threshold.



(a) CS entry insertion (b) CS entry deletion

- After a CS entry is deleted, the name tree is updated. When its name is removed from BF-FIB and BF-CS, they check BF false positive probabilities. Reversion will begin when both BF's false positive probabilities are below the lower bound threshold.

Active CS flowchart

- Reversion is done on the shortest BF-FIB prefix to maximize the reduction of prefix match false positives.
- For transformation, every name tree node remembers the distance to the deepest eligible descendant, and a pointer to the child containing that descendant.
- For aggregation, every node remembers similar distance and pointer fields to the deepest eligible descendent, and also keep track of how many of its descendants are in BF-FIB to show whether a node itself is eligible.
- For reversion, every node maintains the distance to the shallowest descendant having a BF-FIB prefix, and a pointer to the child containing that descendant.

Active CS flowchart

- With those additional maintained information.
  Active CS locates the desired node
  by walking down the name tree
  from the root following the pointers.
  It has h(x) time complexity.

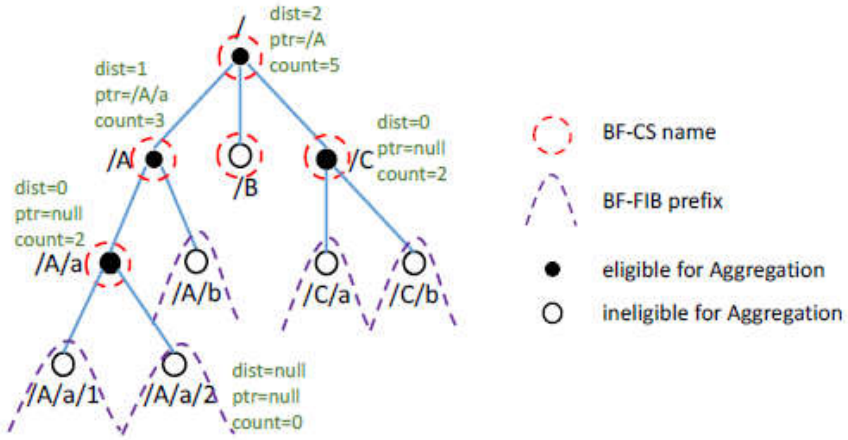


Figure 5: Name tree node fields for Aggregation
Fields for some nodes are omitted due to space constraint.

Overall Filtering Accuracy

- As compared to 100% CPU usage,
  65536 bits for BF-FIB and BF-CS.
  256 bits for BF-PIT.
- Each BF uses two H3 hash functions
  - Active CS parameters achieve the best result for NFS trace.
  - DM filters out 92.26%. Basic CS does 92.92%. Active CS does 96.30%.
- CPU usage is reduced by 92.23% with DM, 92.83% with Basic CS, and 95.92% with Activie CS.
- Optimal NIC filters out 97.51% packets and reduce CPU usage by 97.49%.

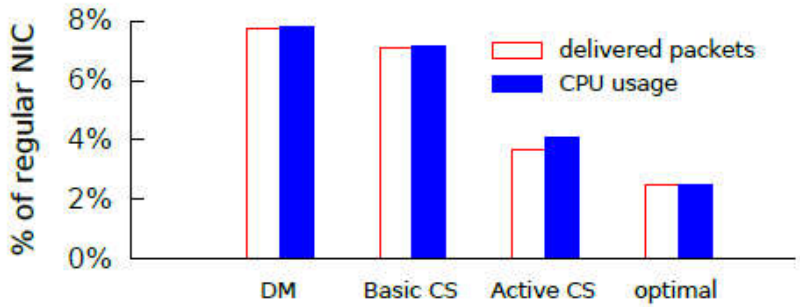


Figure 6: NDN-NIC overall filtering accuracy

Bloom filter characteristics

- Larger BF improves filtering accuracy because of less BF false positives.
- Significant accuracy improvement when BF-CS size is increased from 4096 to 65536 bits.
- Two hash functions give the most improvement in filtering accuracy.

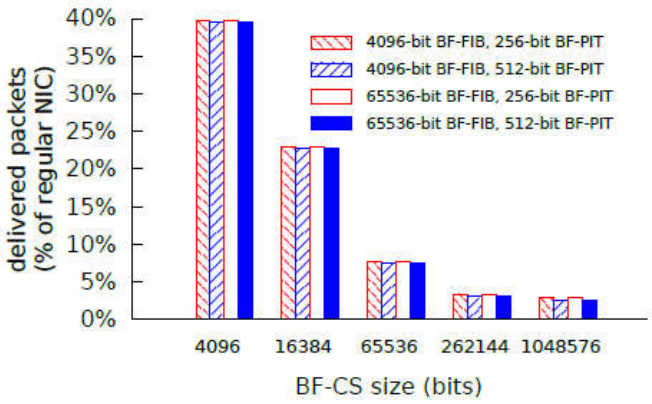


Figure 7: Effect of Bloom filter size, Direct Mapping

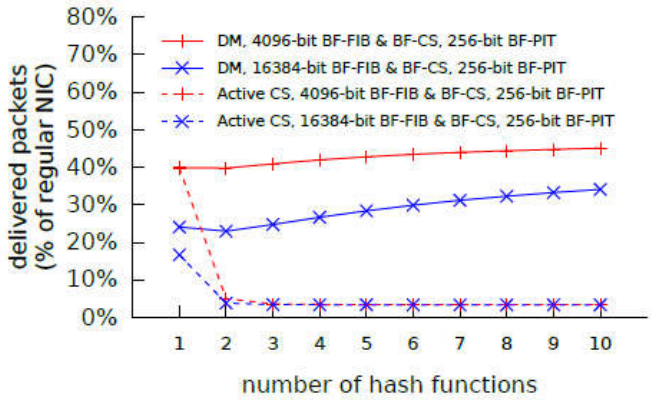


Figure 9: Effect of number of hash functions

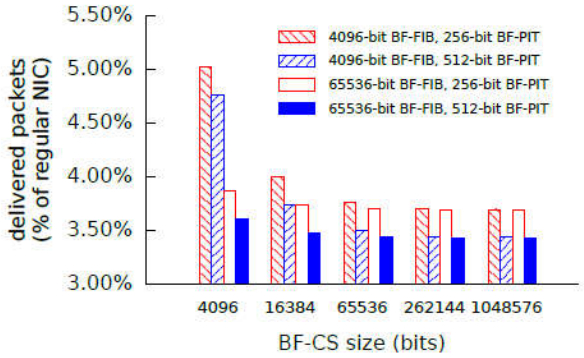


Figure 8: Effect of Bloom filter size, Active CS

Update Algorithms

- Basic CS is similar to DM in CPU usage.
- When BF size is high, DM and Basic CS work efficiently because their false positives are already low.

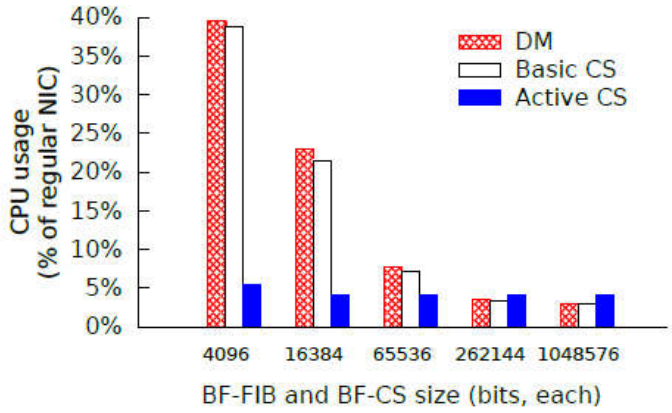
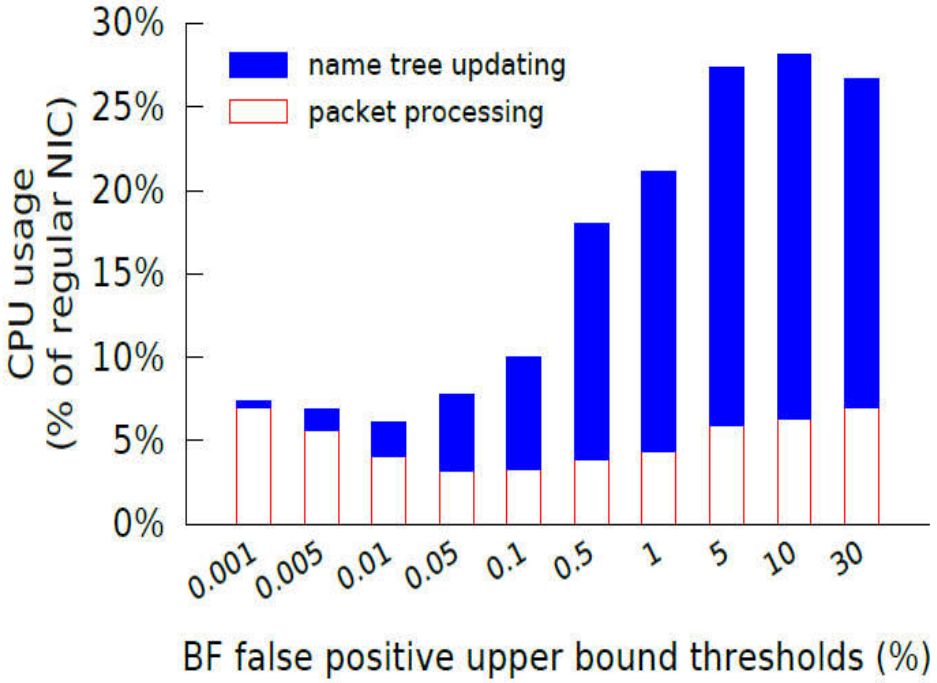


Figure 10: Comparison among update algorithms

Active CS parameters

- Name tree updating cost increases with higher thresholds because more names are kept in BF-CS, resulting in more nodes to update.
- Packet processing cost is minimized at 0.05% threshold.
- Lower thresholds cause more CS entry names to be aggregated onto shorter BF-FIB increasing prefix match false positives. Higher thresholds more BF false positives.



Active CS parameters

- "Fixed" settings use the same degree threshold for all names.
- "Name length dependent" settings provides different degree thresholds at different name lengths.
- "Tailored" settings derive degree thresholds from the traffic.
- Shorter prefixes with larger degree threshold , adding a short prefix to BF-FIB causes more prefix match false positives than longer ones.

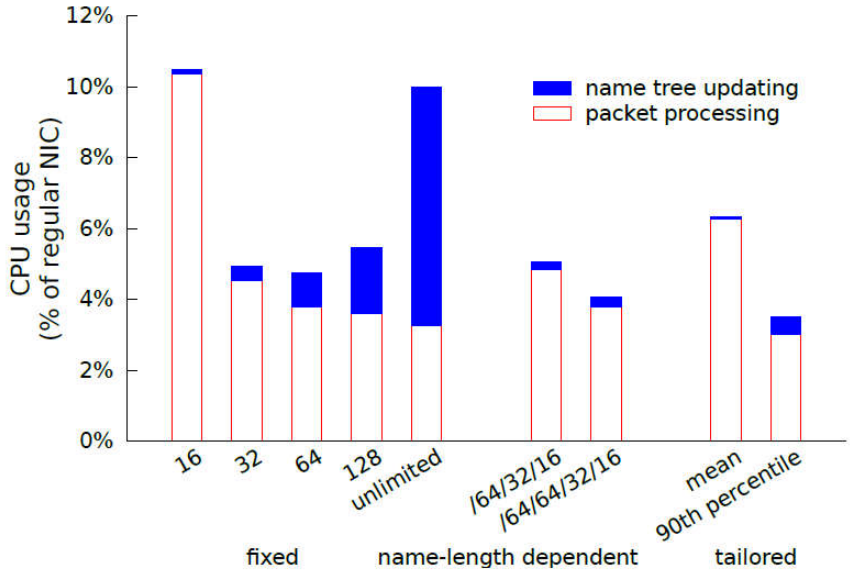


Figure 12: Effect of degree threshold

Active CS parameters

- A NIC processes one byte of outgoing traffic per clock cycle. BF updates share the bandwidth. Outgoing packet traffic is less.
- BF updates consume between 14.8 million and 22.5 million clock cycles, depending on BF sizes and algorithms.
- It means 6.21% to 9.45% overhead above the clock cycles on outgoing packets.
- Larger BF require more updates since there are more bits in BFs.
- Active CS incurs less BF update overhead than DM and Basic CS because many CS entry operations occur under BF-FIB.

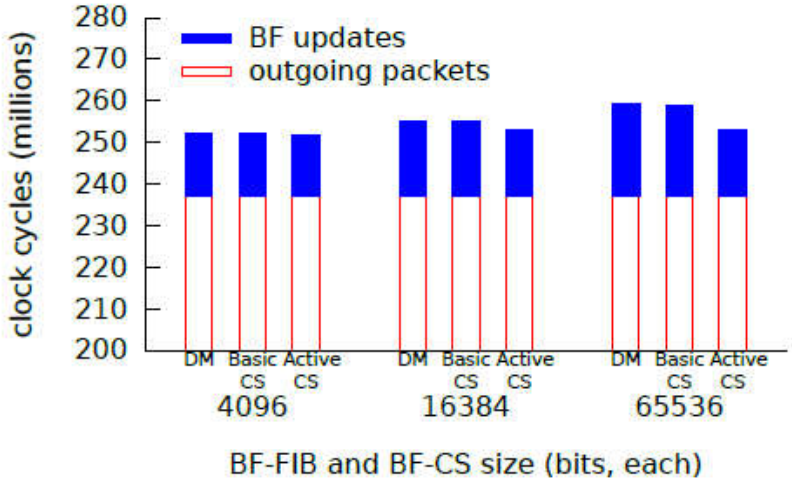


Figure 13: BF update overhead

Conclusion

- Bloom filter supported NDN NIC filters out most of irrelevant packets.
- It reduces the CPU usage, and energy consumption.