From Packet to Flow: Network Security Algorithms to Break Bottleneck

Jun Li
Research Institute of Information Technology
School of Information Science and Technology
Tsinghua University, Beijing, China

Many contributions from my colleagues and students, especially Yaxuan Qi, Bo Xu, and Xin Zhou

Outline

- Why from Packet to Flow?
- Features and Bottlenecks
  - Packet Classification
  - Stateful Inspection
  - Deep Inspection
- Algorithms and Performance
  - Fast Packet Classification: AggreCuts
  - Efficient State Management: SigHash
  - High Performance Content Inspection: MRSI
- Summary

Why from Packet to Flow?

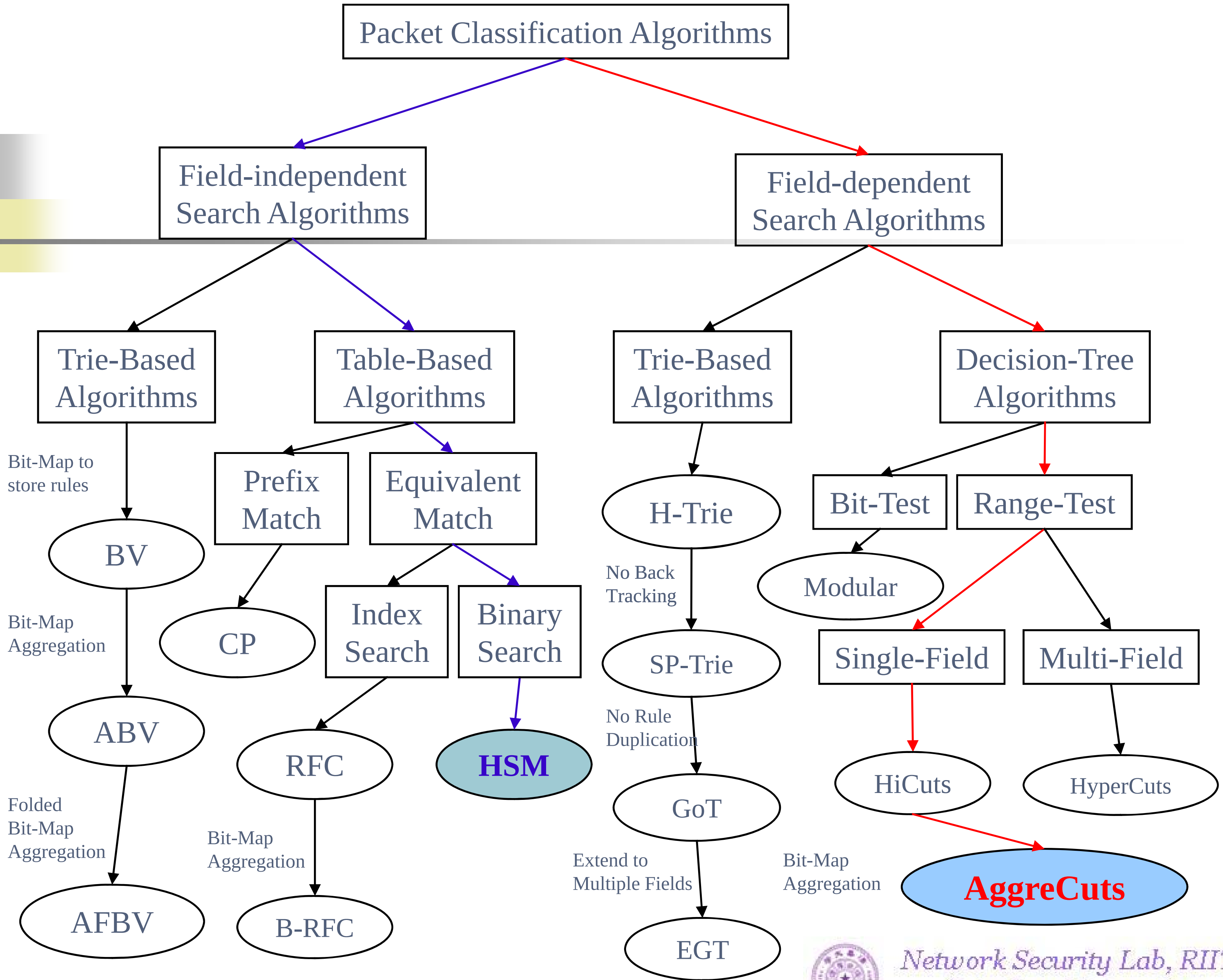
- Increasing sophistication of applications
  - Stateful inspection firewalls
  - Deep inspection in IDS/IPS
- Continual growth of network bandwidth
  - OC192 or higher link speed
  - Millions of concurrent connections
- Requirement for holistic defense
  - Against complex and blended network threats
  - Integrated security features in unified security architecture
  - Unified Threat Management (UTM)

Features and Bottlenecks

- Packet Classification
  - High-speed with modest memory
- Stateful Inspection
  - Large number of connections
  - Order-preserving
- Deep Inspection
  - Enormous signatures
  - Various signature characteristics

Novel Algorithms (1)

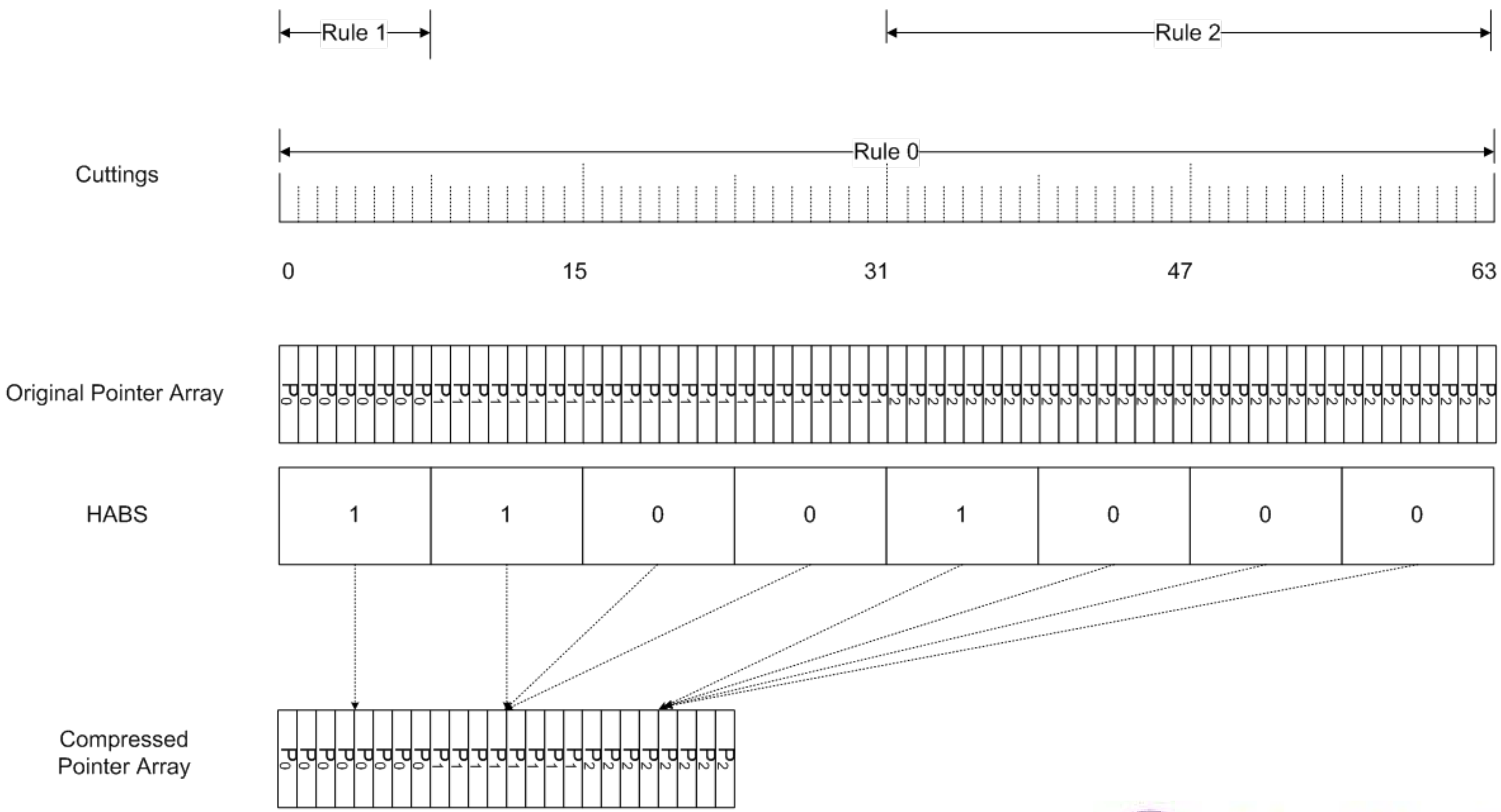
- Packet classification algorithm (AggreCuts)
  - Aggregation Cuttings
    - Multi-dim range match
  - Worst-case bounded and adjustable
    - Limited decision tree depth
    - No linear search
  - Efficient memory storage
    - Space aggregation with bitmap
    - Support different memory hierarchies



Packet Classification Algorithms
Field-independent Search Algorithms
Field-dependent Search Algorithms
Trie-Based Algorithms
Table-Based Algorithms
Trie-Based Algorithms
Decision-Tree Algorithms
Bit-Map to store rules
BV
Bit-Map Aggregation
ABV
Folded Bit-Map Aggregation
AFBV
Prefix Match
CP
Equivalent Match
Index Search
Binary Search
RFC
Bit-Map Aggregation
B-RFC
HSM
H-Trie
No Back Tracking
SP-Trie
No Rule Duplication
GoT
Extend to Multiple Fields
EGT
Bit-Test
Range-Test
Modular
Single-Field
Multi-Field
HiCuts
HyperCuts
Bit-Map Aggregation
AggreCuts

Space Aggregation

- Space Aggregation



Decision-tree

- Data-structure

2-bit d2c	2-bit b2c	8-bit HABS	20-bit Next Node Address Base
00	00	10001000	0x40000

01	00	11000000	0x50000
00	01	10000000	0x52000
11	00	10100000	0x55000
10	00	10100001	0x58000

11	00	10100000	0x60000
11	00	10100000	0x60100
01	01	10110100	0x62000
10	00	10001000	0x62500

01	00	10001001	0x63000
00	02	10000100	0x63700

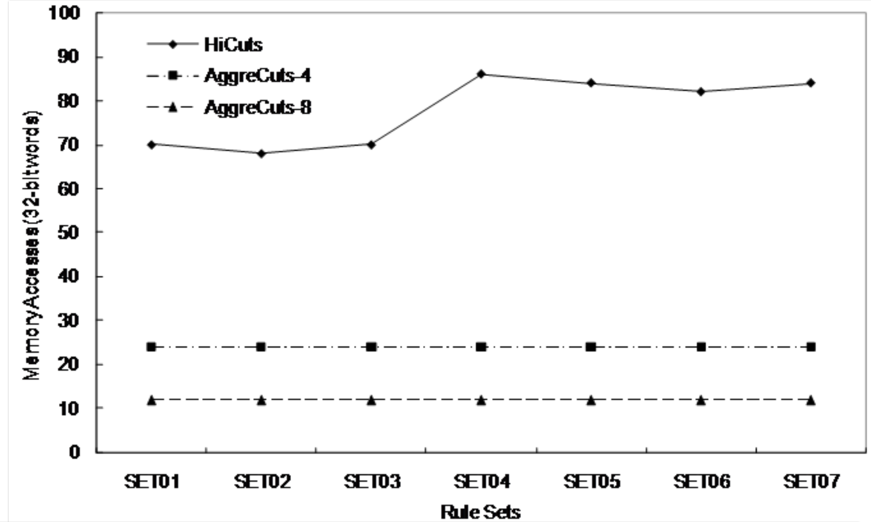
11	01	HABS	0x65000
01	00	HABS	0x65200
01	00	HABS	0x67000
01	00	HABS	0x67800

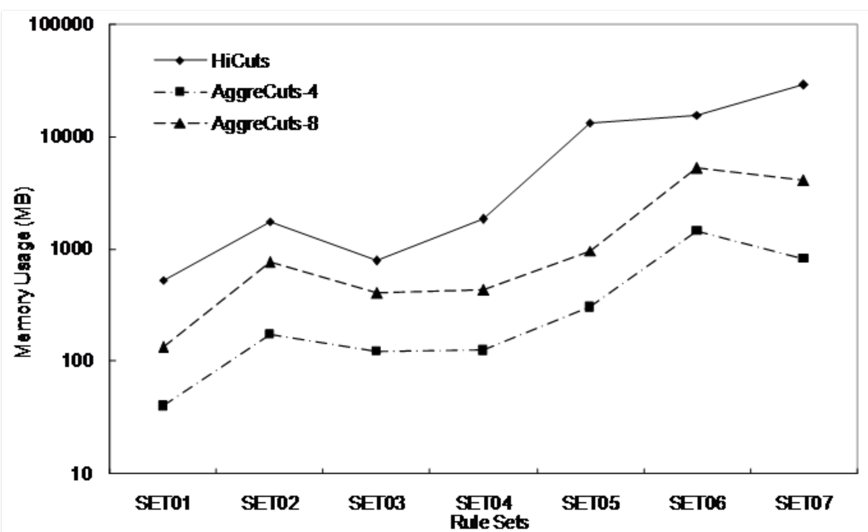
10	01	10101010	0x68200
10	01	10101010	0x68500
00	01	11100000	0x70000
01	00	10000101	0x71000
11	00	11000010	0x71500
00	01	10000010	0x73000

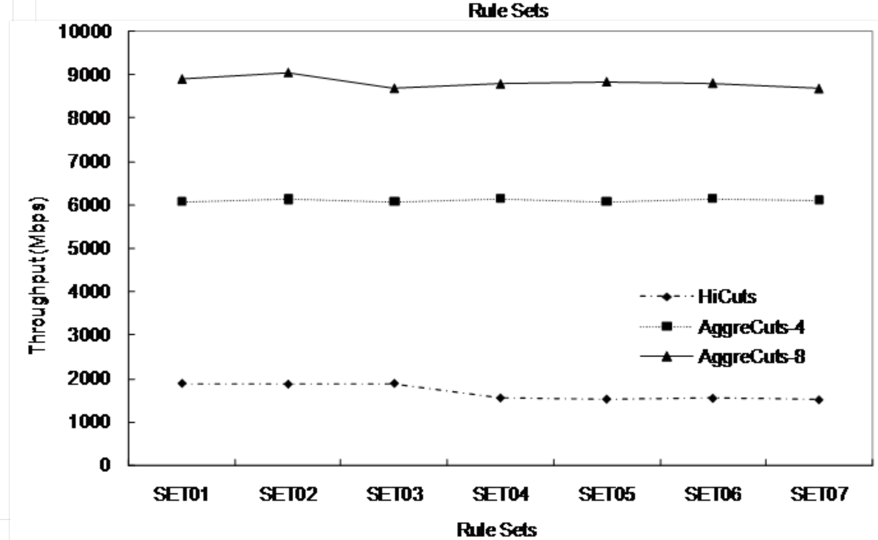
Bits	Description	Value
31:30	dimension to Cut (d2c)	d2c=00: src IP; d2c=01: dst IP; d2c=10: src port; d2c=11: dst port.
29:28	bit position to Cut (b2c)	b2c=00: 31~24; b2c=01: 23~16 b2c=10: 15~8; b2c=11: 7~0
27:20	8-bit HABS	if *w*=8, each bit represent 32 cuttings; if *w*=4, each bit represent 2 cuttings.
19:0	20-bit Next-Node CPA Base address	The minimum memory block is $2^w/8*4$ Byte. So if *w*=8, 20-bit base address support 128MB memory address space; if *w*=4, it supports 8MB memory address space.

AggreCuts vs. HiCuts

- Performance Evaluation
  - Memory Usage:
    - an order of magnitude less
  - Memory Access:
    - 3~8 times less
  - Throughput on IXP2850:
    - 3~5 times faster





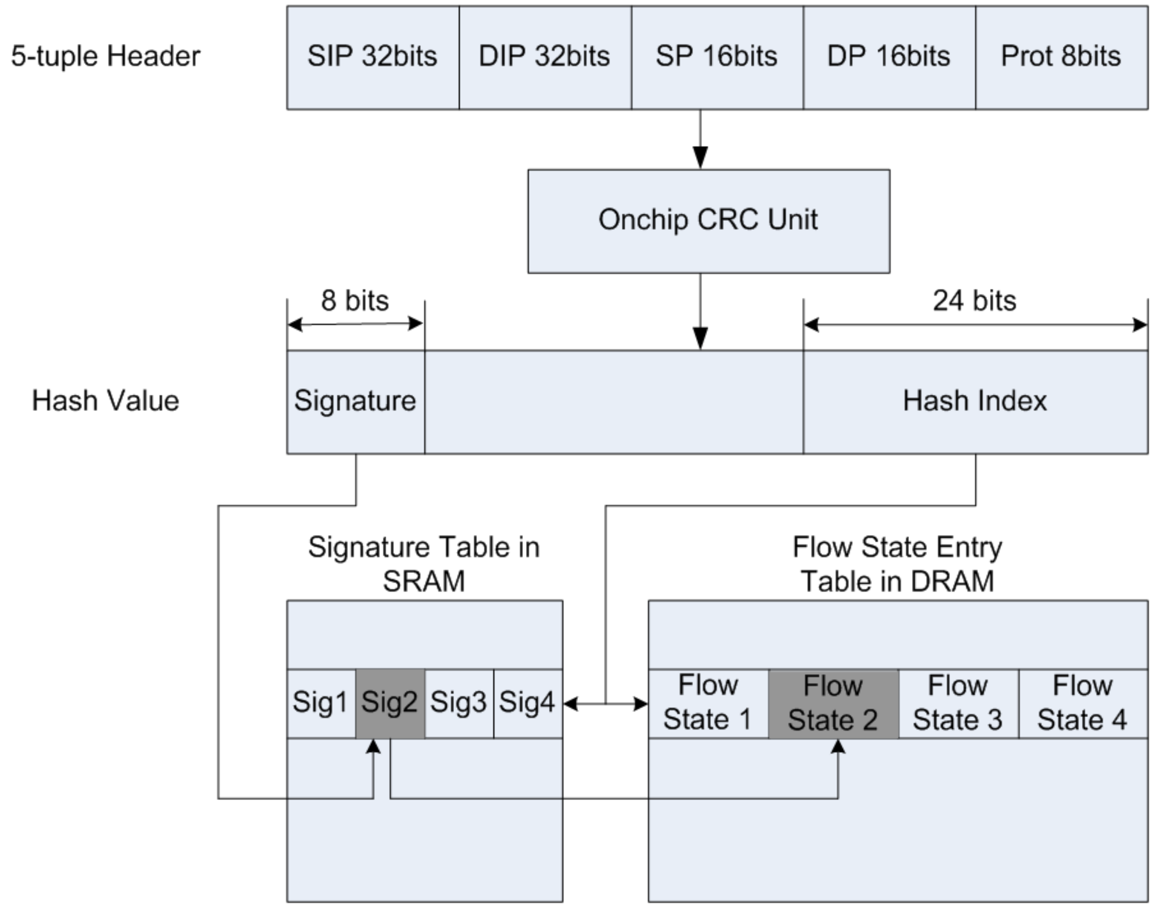


Novel Algorithms (2)

- Stateful inspection algorithm (SigHash)
  - Signature based hashing
    - Support large concurrent connections
    - Efficient memory usage
    - High speed TCP handshakes
  - Per-flow packet order preserving
    - External Packet order preserving
    - Internal Packet order preserving

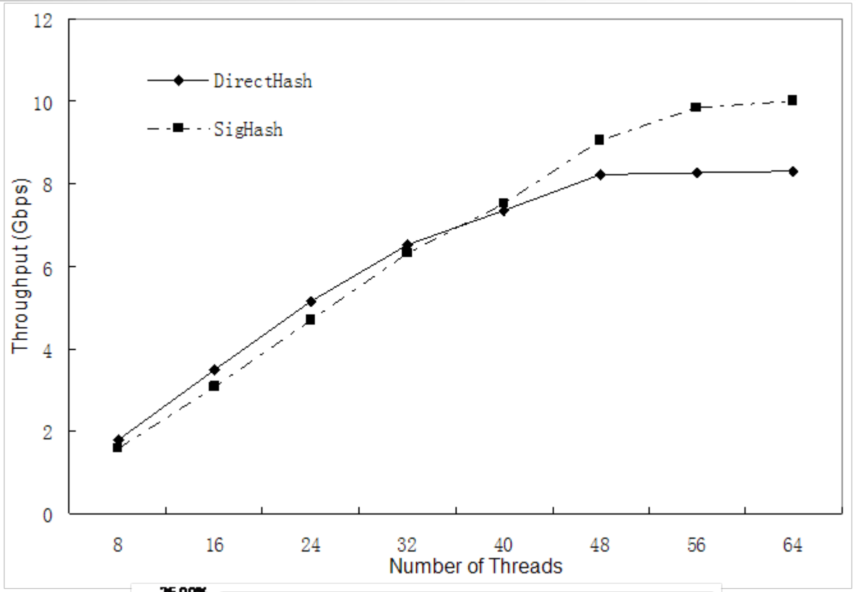
Signature-based Hash

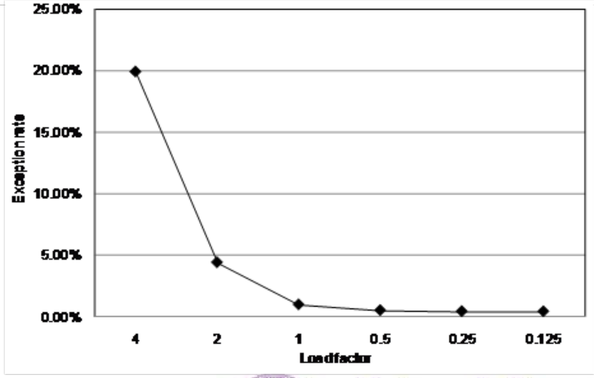
- Signature-based Hashing
  - *m* signatures for *m* different states with same hash value
  - Resolving collision in SRAM (fast, word-oriented)
  - Storing states in DRAM (large, burst-oriented)



SigHash Performance

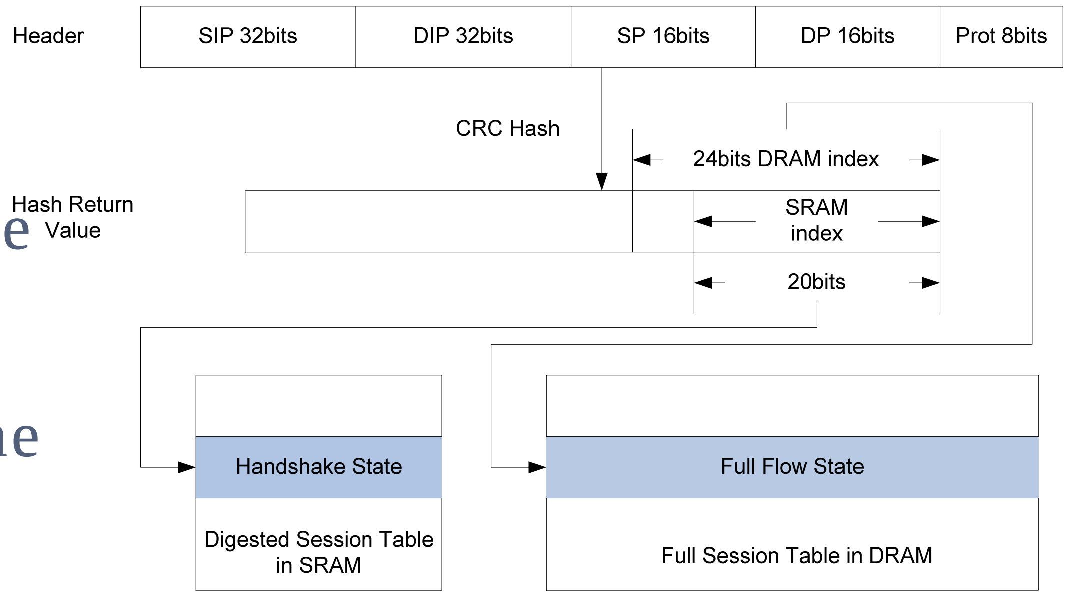
- Throughput
  - 10Gbps
  - (SRAM+DRAM)
  - 8Gbps
  - (DRAM only)
- Connections
  - 10M on IXP2850
- Collision
  - Less than 1%
  - Depends on different load factors





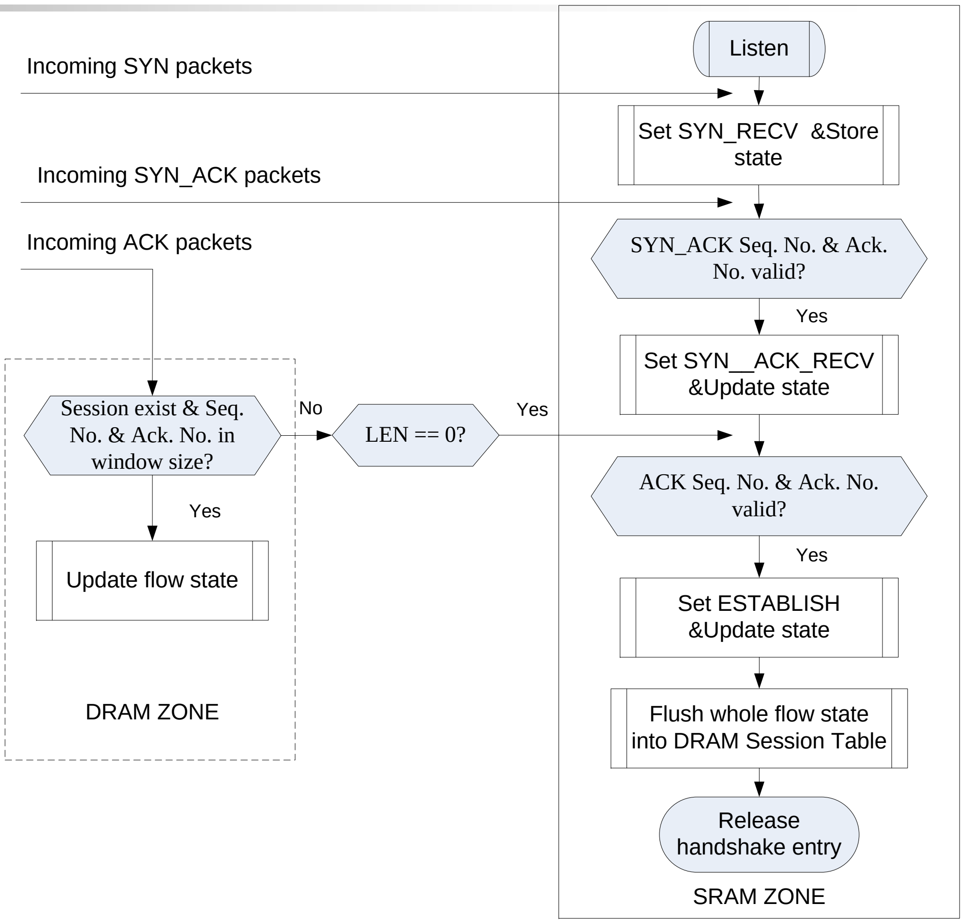
Handshake-separated Hash

- Handshake-separated Hash (IntelliHash)
  - Process handshake packets in SRAM, data packets in DRAM, sharing the same hash value
  - Speedup session creation
  - Enhance anti-DoS capability



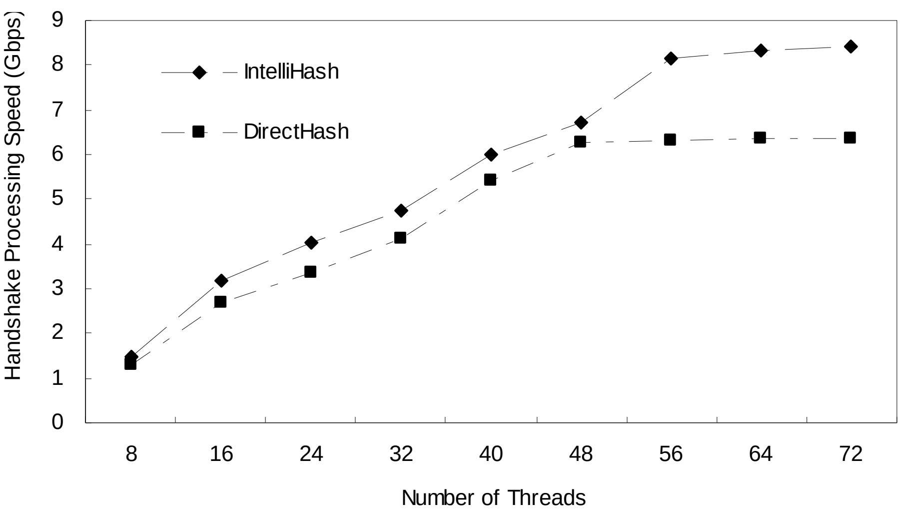
IntelliHash Procedure

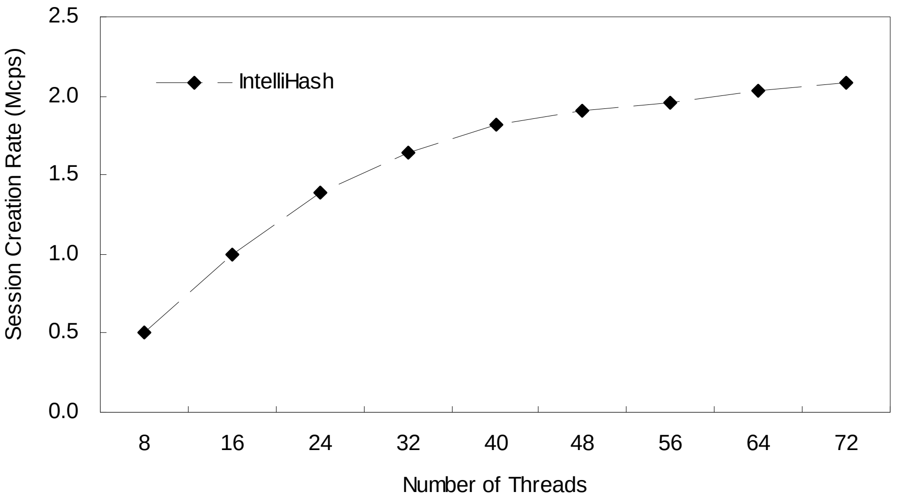
- Handshake packets processing
  - Process SYN/SYN_ACK packets in SRAM
  - Process ACK packets in DRAM; if (LEN==zero && session!=exist), process in SRAM Zone



IntelliHash

- Performance Evaluation
  - Handshake packets processing speed
    - 8.5G (IntelliHash)
    - 6.5G (DirectHash)
  - Session Creation Rate
    - Up to 2M connections per second (IntelliHash)





Per-flow Packet Ordering

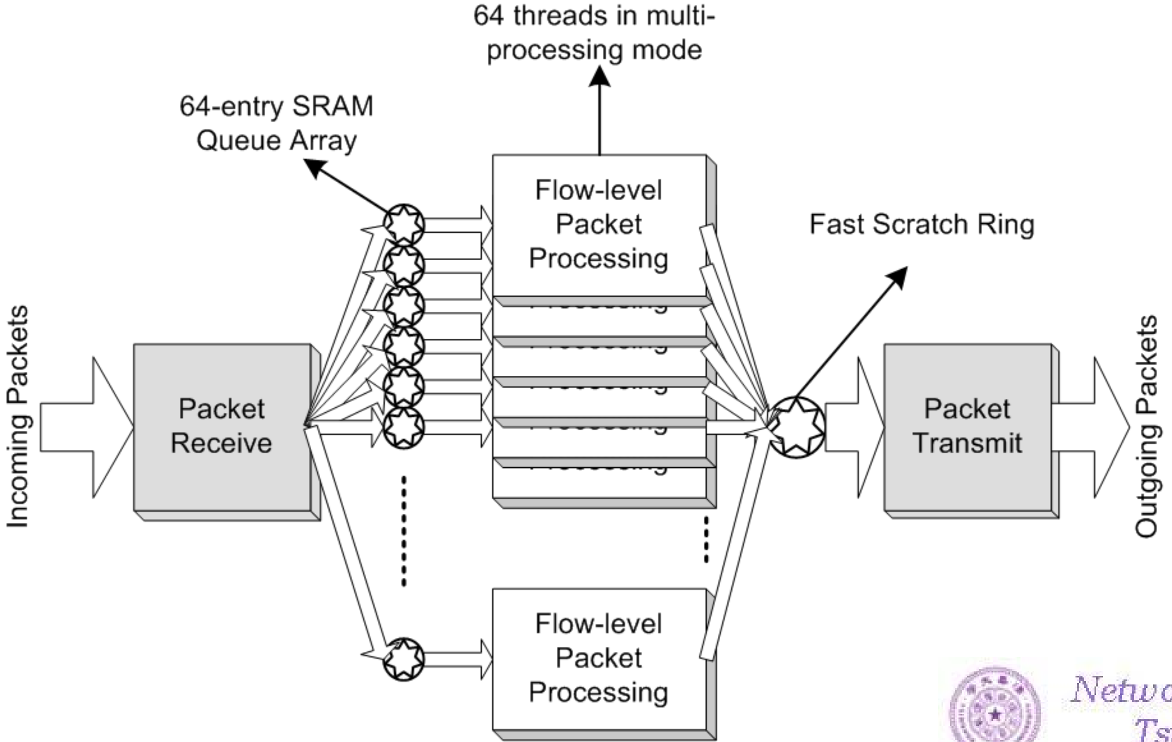
- Packet Order-preserving
  - Typically, only required between packets on the same flow.
- External Packet Order-preserving (EPO)
  - Sufficient for processing packets at network layer.
  - Fine-grained workload distribution (packet-level)
  - Need locking
- Internal Packet Order-preserving (IPO)
  - Required by applications that process packets at semantic levels.
  - Coarse-grained workload distribution (flow-level)
  - No need for locking

Per-flow Packet Ordering

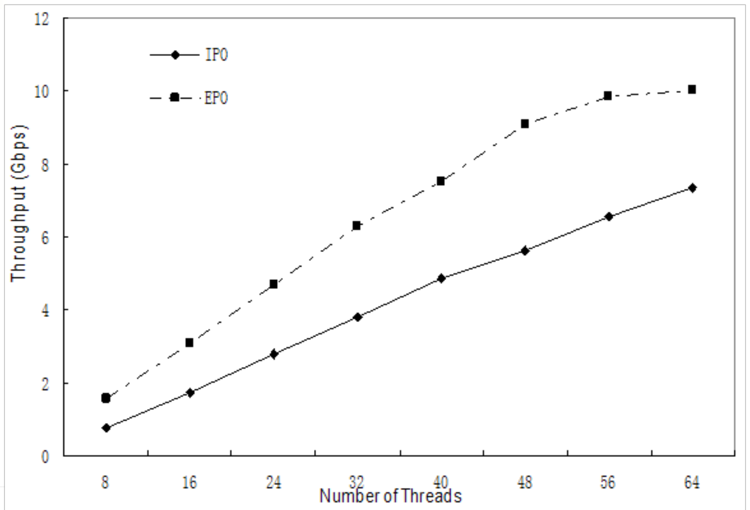
- External Packet Order-preserving (EPO)
  - Ordered-thread Execution
    - Ordered critical section to read the packet handles off the scratch ring
    - The threads then process the packets, which may get out of order during packet processing
    - Another ordered critical section to write the packet handles to the next stage
  - Mutual Exclusion by Atomic Operation
    - Packets belong to the same flow may be allocated to different threads to process
    - Mutual exclusion can be implemented by locking
    - SRAM atomic instructions

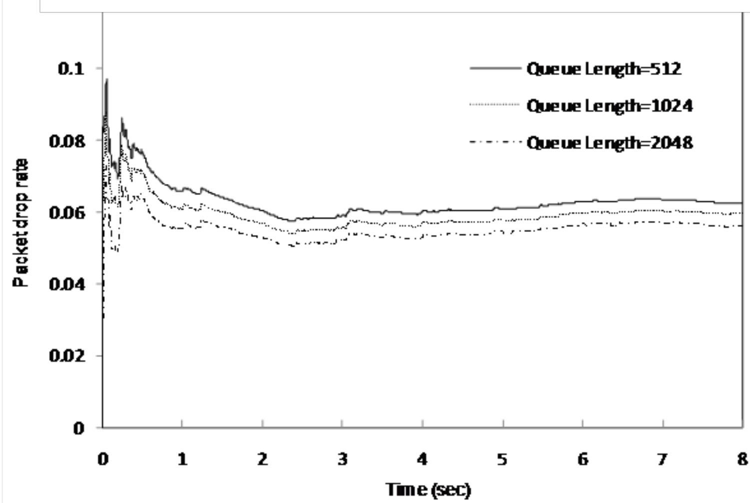
Per-flow Packet Ordering

- Internal Packet Order-preserving (IPO)
  - SRAM Q-Array
  - Workload Allocation by CRC Hashing on Headers



Per-flow Packet Ordering

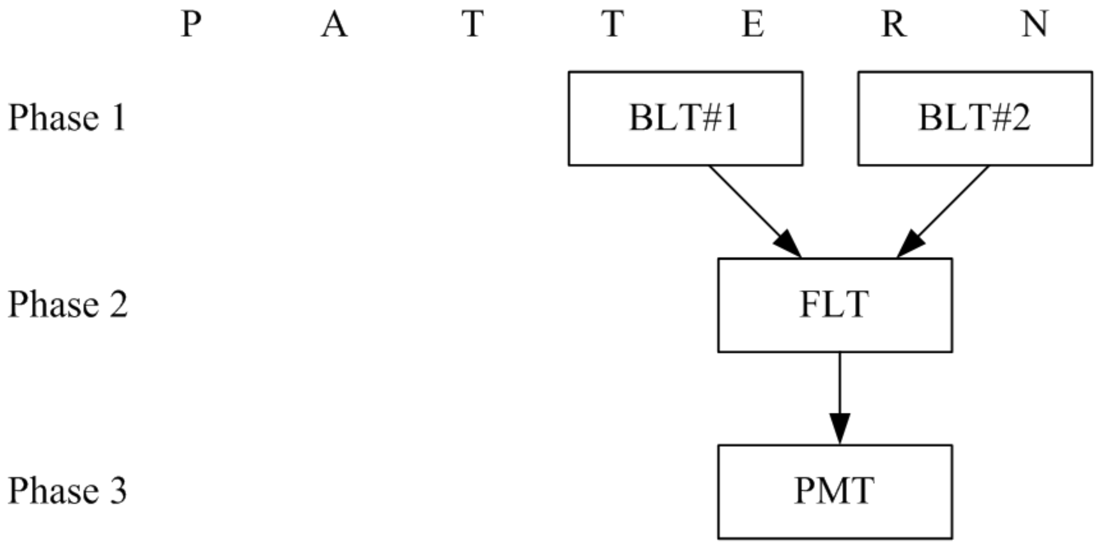




- Performance Evaluation
  - Throughput
    - EPO is faster, 10Gbps
    - IPO has linear speed up, 7Gbps
  - Workload Allocation
    - Hashing via On-chip CRC
    - Nearly balanced workload

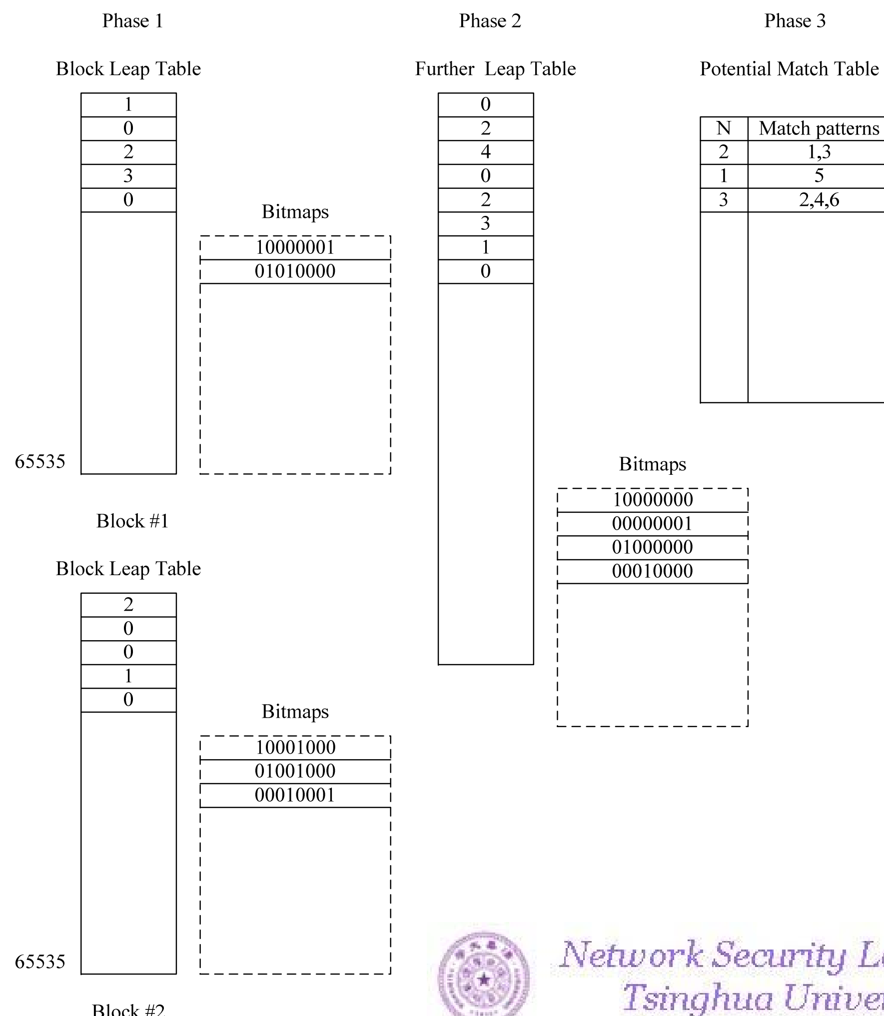
Novel Algorithms (3)

- RSI (Recursive Shift Indexing)
  - Reduce the number of useless matching
  - Pro: trade-off space with time
    - Directly using four-character block to create the BLT will use memory up to $256^4 \rightarrow 4$ GB

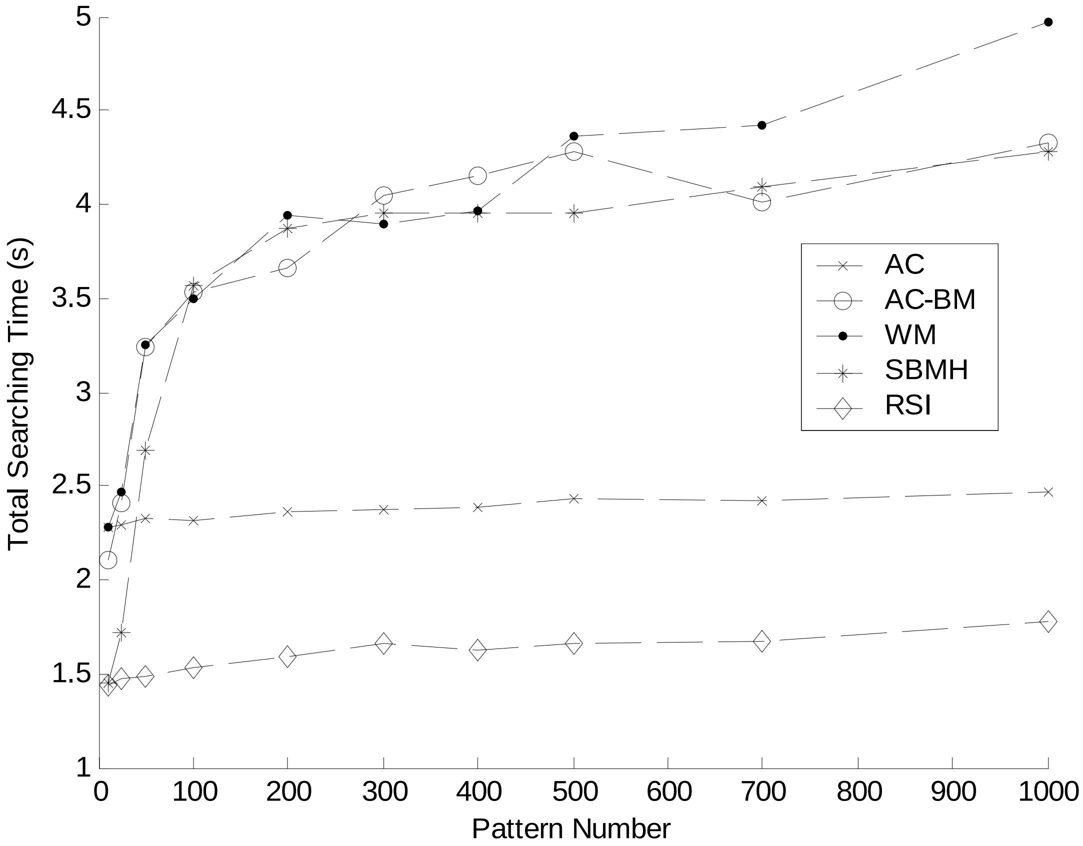


RSI Data Structure

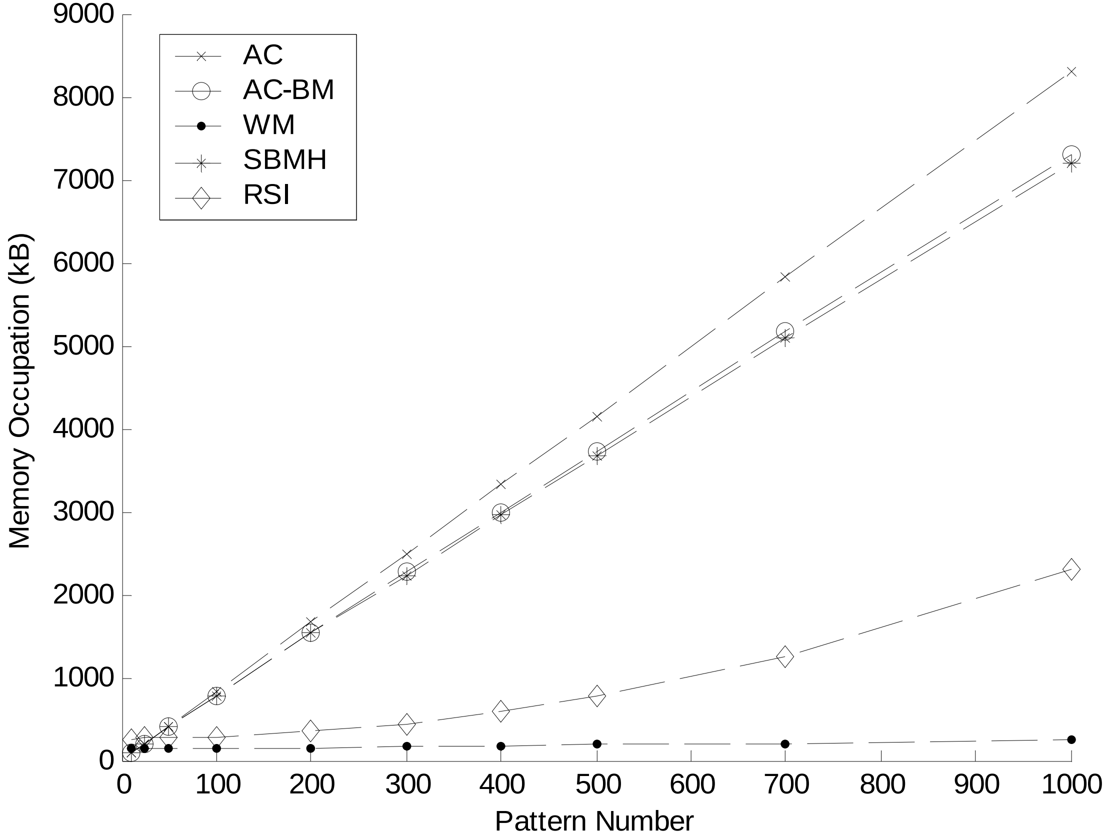
- Bitmaps are used for preprocessing and deleted after that



RSI Temporal Performance



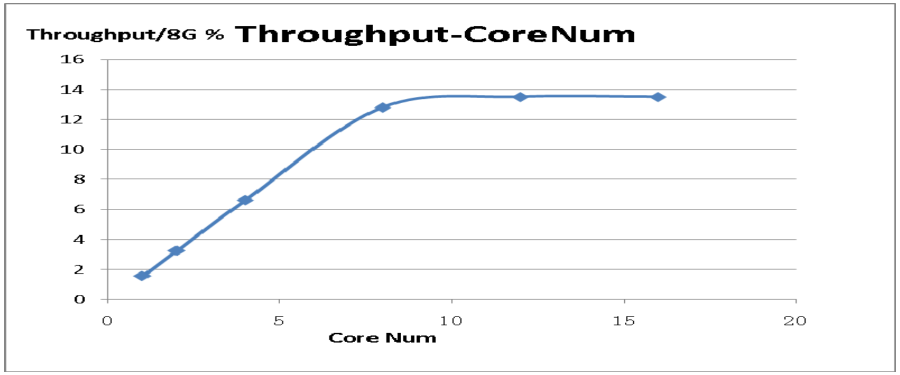
RSI Spatial Performance



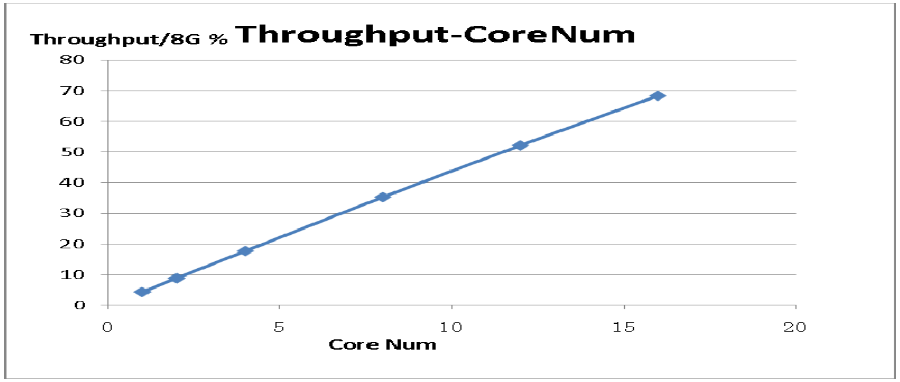
Break the Real Bottleneck

- Current version of Clam-AV
  - The basic signatures are handled by BMEXT that uses the last 3 characters of a signature to generate shifts
- Large dataset characteristics
  - ClamAV: 78k basic rules
- Our proposal: hybrid algorithms
  - DFA for short signatures: DFA-based algorithm implemented on fast on-chip memory
    - Space efficient
    - High performance (5.5G vs 1.2G on Octeon)
  - HASH for long signatures: Hash-based algorithm with larger shifts than BMEXT
    - Search with shifts/skips: i.e. MRSI

DFA Performance Limit



DFA size = 100MB, Len=512Byte
1.2Gbps on Octeon 3860



DFA size = 100KB, Len=512Byte
5.5Gbps on Octeon 3860

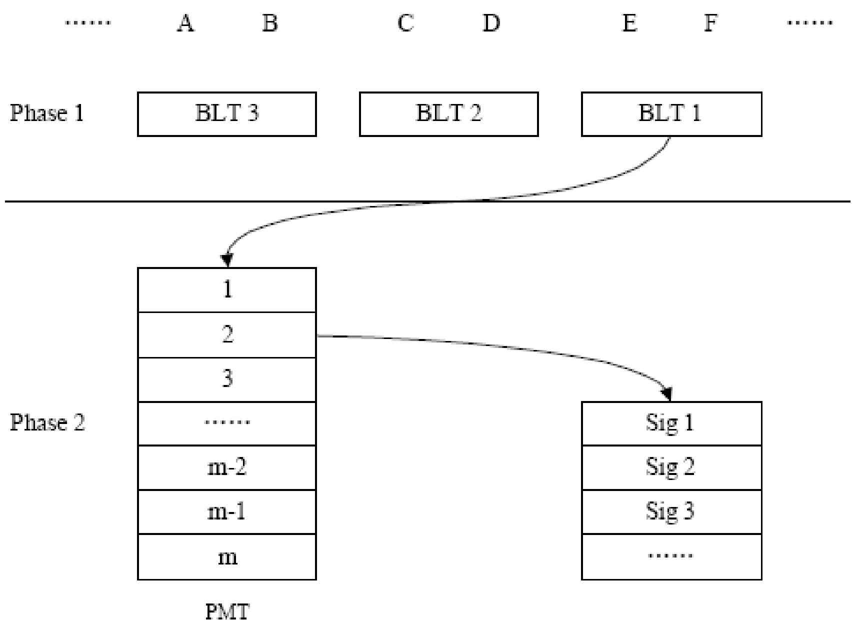
Statistics of ClamAV Signatures

Idx	Total Number	Average Length	Min Length	Len<9 Num
0	29611	67.5	10	0
1	46954	123.7	4	8
2	164	106.8	28	0
3	1402	110.7	14	0
4	355	46.6	17	0
5	0	n/a	n/a	0
6	15	105.1	17	0

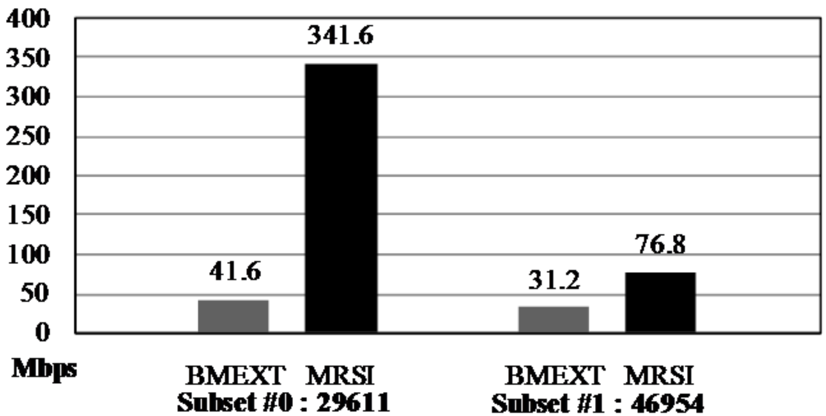
- Large scale signature set
- Longer average length
- Very few short signatures

MRSI

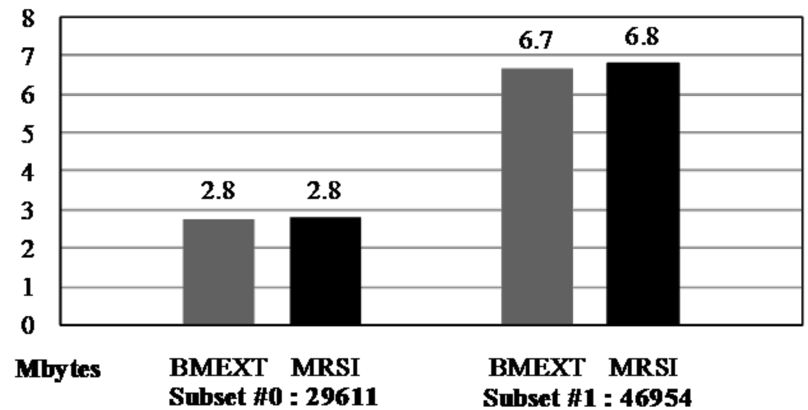
- Use three BLTs
  - Increase the probability of getting leap
- Omit Phase 2 in original RSI data structure
  - Solve memory occupation expansion
  - Improve preprocessing speed



MRSI Performance

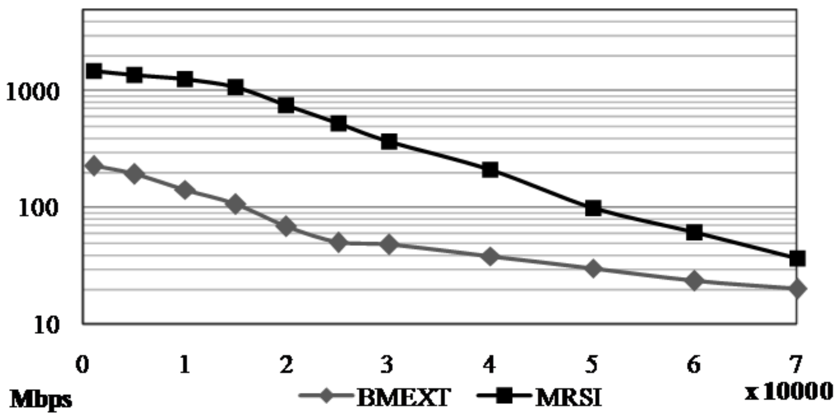


MRSI vs. BMEXT: Scanning Speed

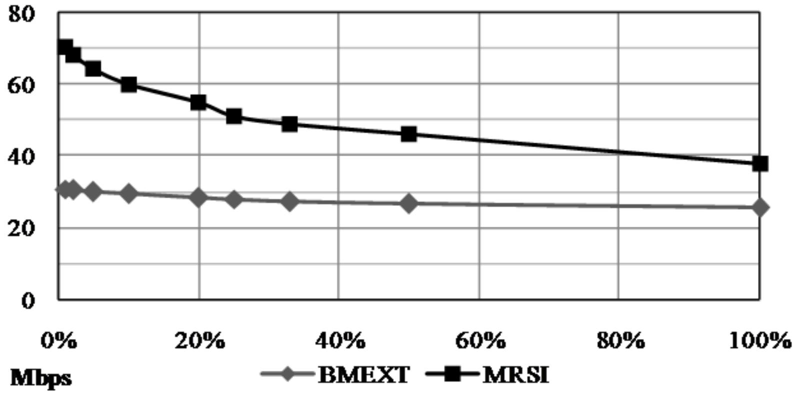


MRSI vs. BMEXT: Memory Usage

MRSI Performance

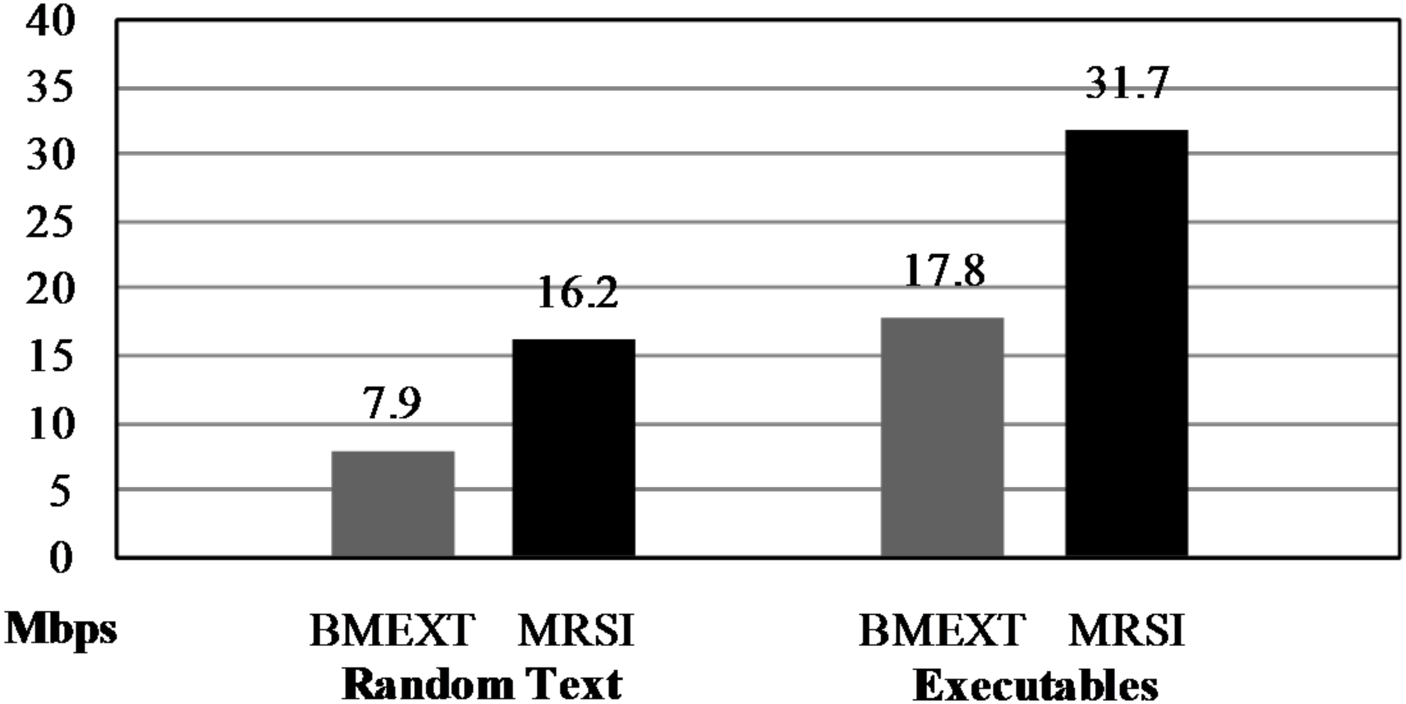


MRSI vs. BMEXT: Scalability



MRSI vs. BMEXT: Performance under Attacks

MRSI Performance in AV



Real System Performance on Clam-AV

Summary

- Analyze the real problem
  - Packet classification
  - Stateful Inspection
  - Deep Inspection
- Propose new algorithms
  - Hardware aware
  - Time-space tradeoff
- Break the real bottleneck

Reference

- [1] Yaxuan Qi, Bo Xu, Fei He, Baohua Yang, Jianming Yu and Jun Li, Towards High-performance Flow-level Packet Processing on Multi-core Network Processors, Proc. ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2007.
- [2] Yaxuan Qi, Baohua Yang, Bo Xu, and Jun Li, Towards System-level Optimization for High Performance Unified Threat Management, Proc. 3rd International Conference on Networking and Services (ICNS), 2007.
- [3] Bo Xu, Yaxuan Qi, Fei He, Zongwei Zhou, Yibo Xue and Jun Li, Fast Path Session Creation on Network Processors, Proc. 28th International Conference on Distributed Computing Systems (ICDCS), 2008. (to appear)
- [4] Xin Zhou, Bo Xu, Yaxuan Qi and Jun Li, MRSI: A Fast Pattern Matching Algorithm for Anti-virus Applications, Proc. 7th International Conference on Networking, 2008.

Thanks

http://security.riit.tsinghua.edu.cn