Efficient Memory Utilization on Network Processors for Deep Packet Inspection

Yan Luo

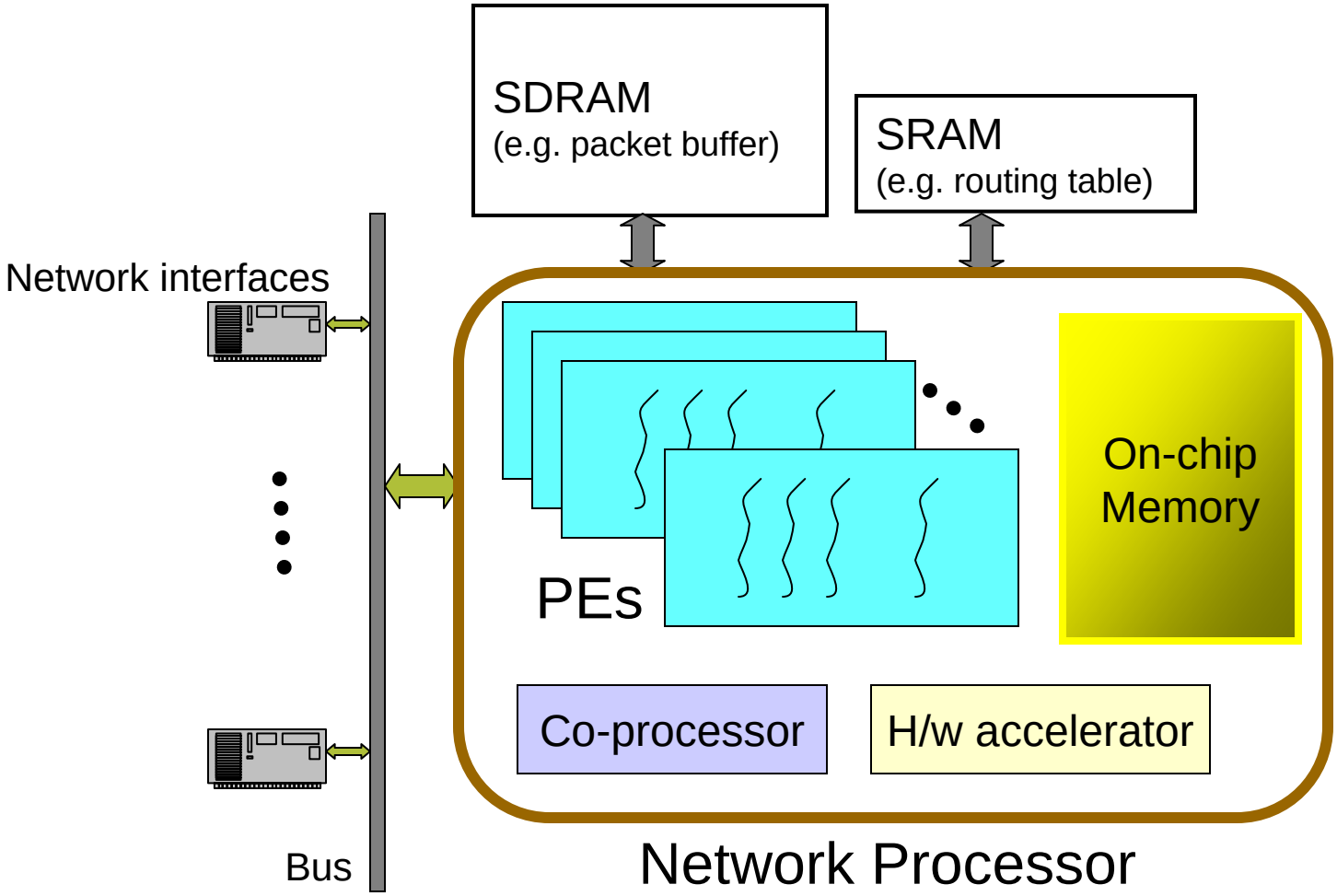
Assistant Professor

Dept. of Electrical and Computer Engineering
University of Massachusetts Lowell

Outline

- Overview of Network Processors and Deep Packet Inspection
- Efficient Memory Utilization on Network Processors for DPI
- Ongoing Research
- Prior Works on Network Processors

Generic Network Processor Architecture



DPI and its Challenges

- Deep Packet Inspection
  - Inspect packet header & payload
  - Detect network intrusion, worms, spam, etc.

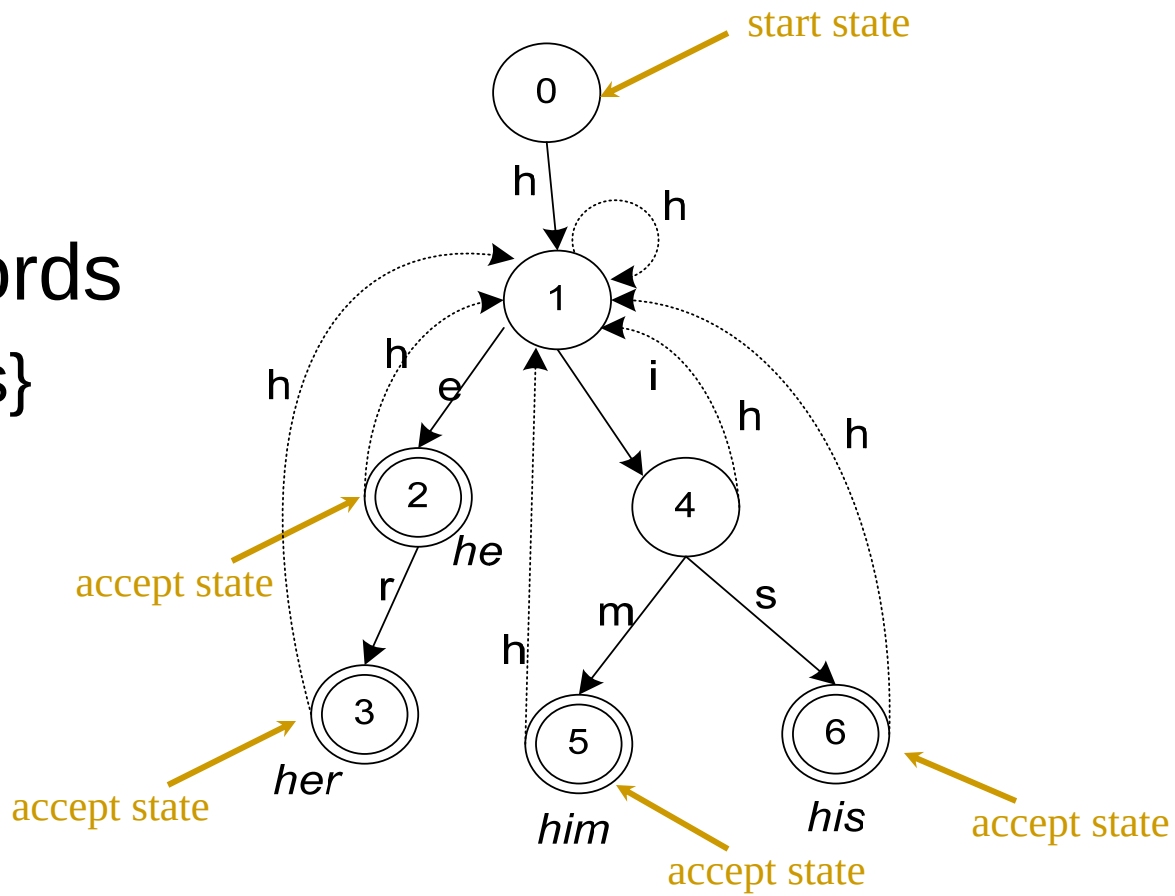
    Examples: Bro, Snort, etc.
- Challenges
  - Matching predefined multiple keywords (also called rules).
  - Keywords can be any size.
  - Keywords can be anywhere in the payload of a packet.
  - Rule Matching at line speed
  - Flexibility to accommodate new rule sets

Classical Aho-Corasick (AC) DFA

- Example

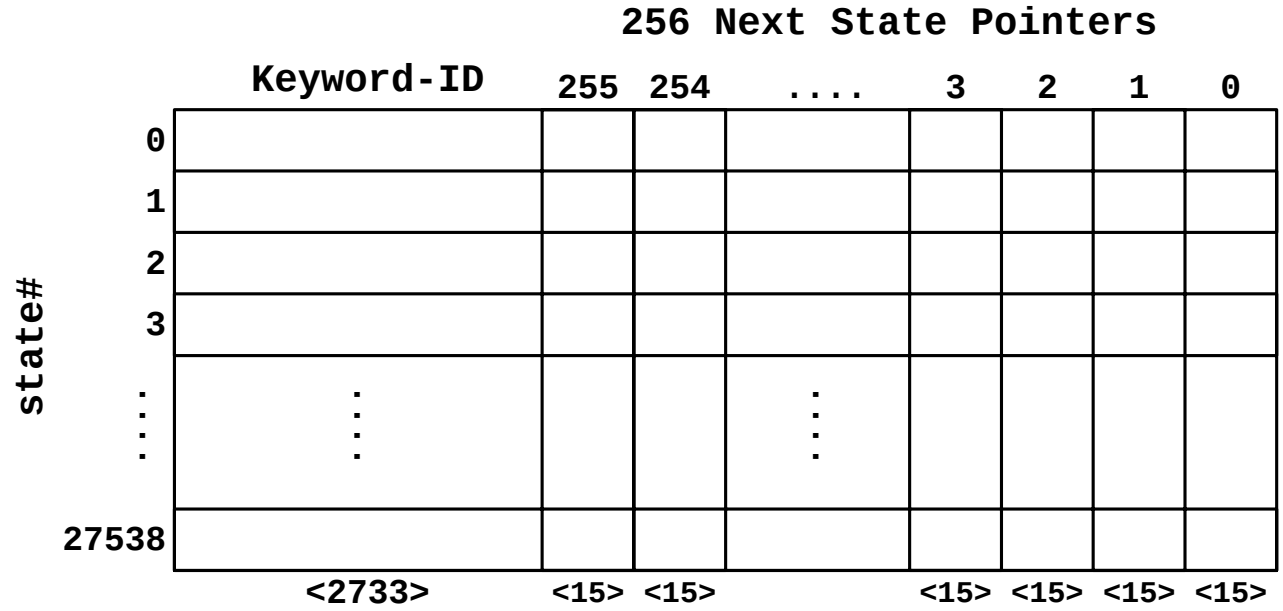
  A set of keywords

  {he, her, him, his}



[1] Aho and Corasick, "Efficient string matching: An aid to bibliographic search". *Communications of the ACM* 18 (6): 333–340.

Storage of AC DFA

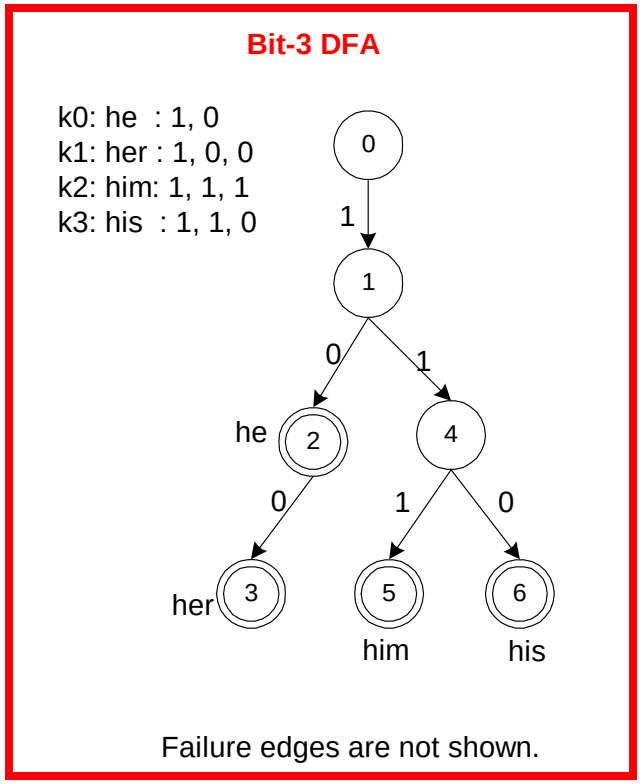


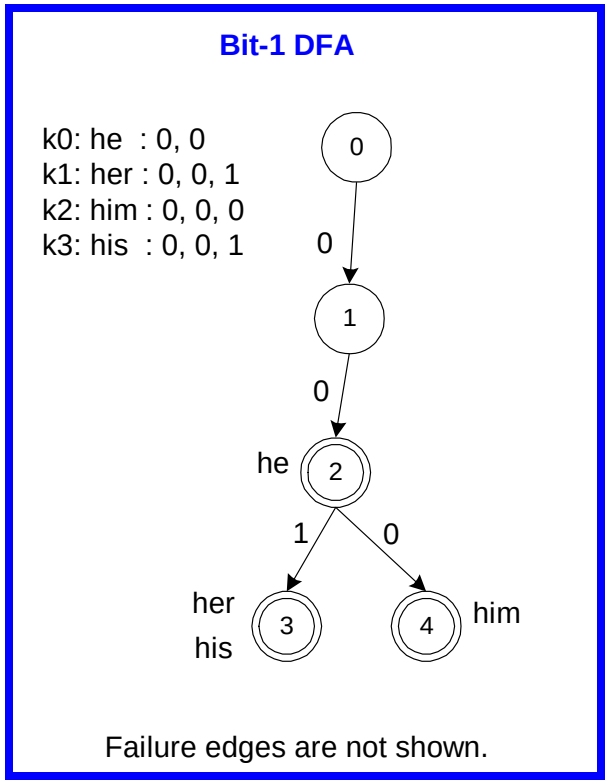
- Snort rule set (Dec'05): 2733 keywords
- DFA has over 27,000 states
  - width = 15 bits
- 256 next state pointers per state
- keyword-ID width = 2733 bits
- 27538 x (2733 + 256 x 15)b = 22 MB too big for on-chip memory

Bit-AC DFA

k0	h	e		0110 1000	0110 0101	
k1	h	e	r	0110 1000	0110 0101	0111 0010
k2	h	i	m	0110 1000	0110 1001	0110 1101
k3	h	i	s	0110 1000	0110 1001	0111 0011

Need 8 bit-DFA

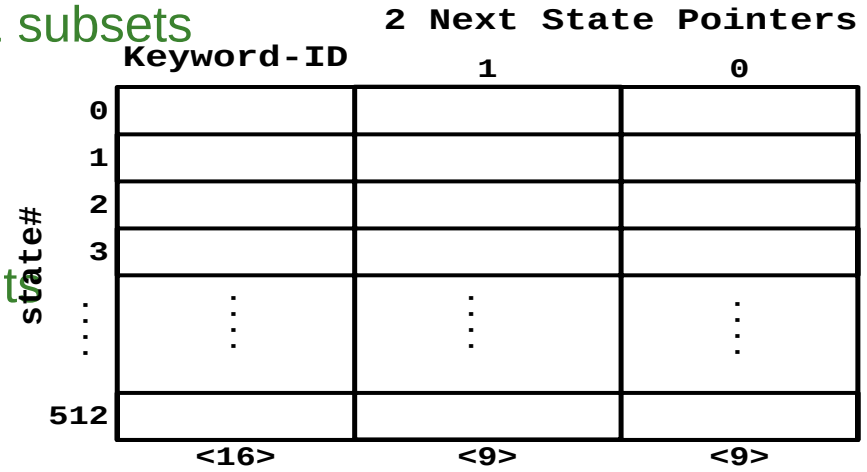




[2] Tan, Brotherton and Sherwood, Bit-split string-matching engines for intrusion detection and prevention, ACM TACO, March 2006

Bit-AC DFA

- Shrinks the width of keyword-ID
  - By dividing 2733 keywords into 171 subsets
    - Each subset has 16 keywords
    - Width change from 2733 to 16 bits
- Reduces next state pointers
  - By dividing each input byte into 8 bits
    - From 256 to 2 pointers
    - Need 8 bit-DFA
- Extra benefits
  - The number of states (per DFA) reduces from ~27,000 to ~300 states.
  - The width of next state pointer reduces from 15 to 9 bits.
- # of DFA
  - With 171 subsets, each subset has 8 DFA.
  - Total 171 x 8 = 1,368 DFA
- Memory reduced from 22 MB to 2 MB



Can we do better to reduce the memory usage?

Byte-AC DFA

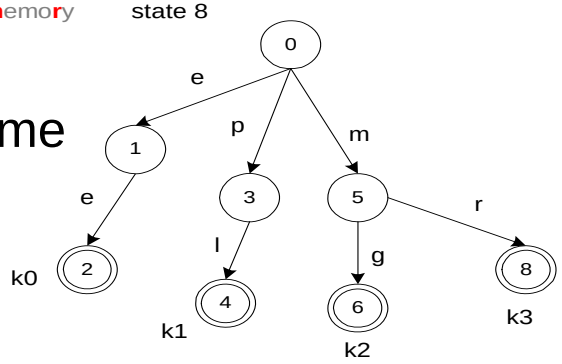
		byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3
k0	elements	e	l	e	m	e	n	t	s
k1	parallel	p	a	r	a	l	l	e	l
k2	manage	m	a	n	a	g	e		
k3	memory	m	e	m	o	r	y		

- Considering 4 bytes at a time
- 4 DFA
- ≤ 9 states per DFA
- 256 next state pointers

Benefit: reduce # of states per DFA

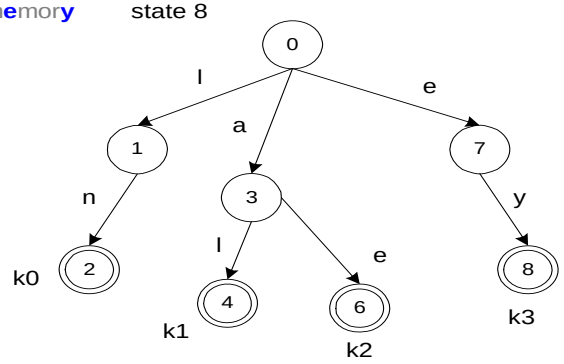
Byte 0

Match Found at
k0: elements state 2
k1: parallel state 4
k2: manage state 6
k3: memory state 8



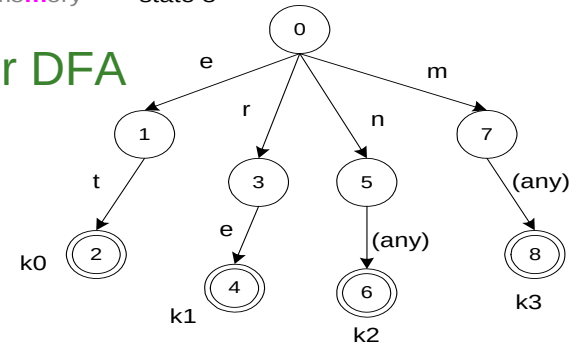
Byte 1

Match Found at
k0: elements state 2
k1: parallel state 4
k2: manage state 6
k3: memory state 8



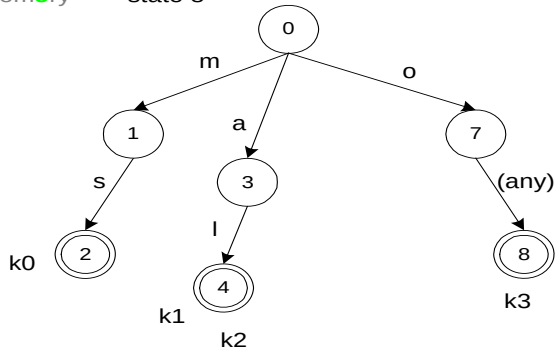
Byte 2

Match Found at
k0: elements state 2
k1: parallel state 4
k2: manage state 6
k3: memory state 8



Byte 3

Match Found at
k0: elements state 2
k1: parallel state 4
k2: manage state 4
k3: memory state 8



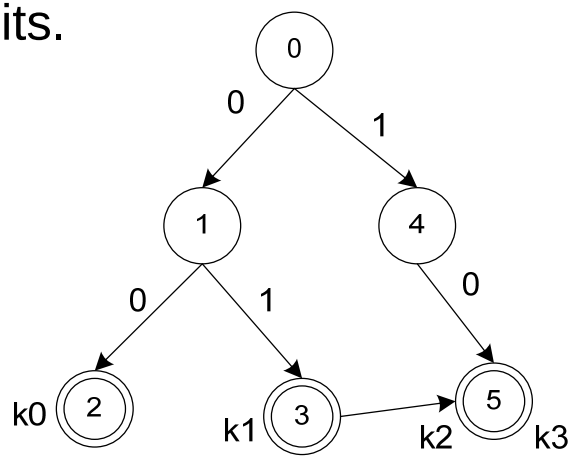
Bit-Byte AC DFA

									byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3
k0	e	l	e	m	e	n	t	s	0110 **0**101	0110 1100	0**1**10 0101	0110 1101	0110 **0**101	0110 1110	0**1**11 0100	0111 0011
k1	p	a	r	a	l	l	e	l	0111 **0**000	0110 0001	0**1**11 0010	0110 0001	0110 **1**100	0110 1100	0**1**10 0101	0110 1100
k2	m	a	n	a	g	e			0110 **1**101	0110 0001	0**1**10 1110	0110 0001	0110 **0**111	0110 0101	x**x**xx xxxx	xxxx xxxx
k3	m	e	m	o	r	y			0110 **1**101	0110 0101	0**1**10 1101	0110 1111	0111 **0**010	0111 1001	x**x**xx xxxx	xxxx xxxx

- 4 bytes at a time
- Each byte divides into bits.
- 32 DFAs (= 4 x 8)
- ≤ 6 states per DFA
- 2 next state pointers

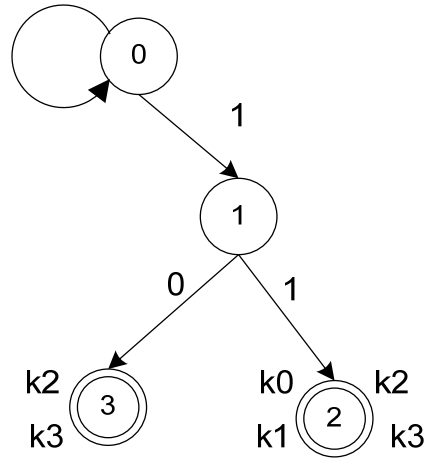
bit 3, Byte 0

k0: **e**lem**e**nts 0,0
k1: **p**ara**l**lel 0,1
k2: **m**ana**g**e 1,0
k3: **m**emo**r**y 1,0



bit 6, Byte 2

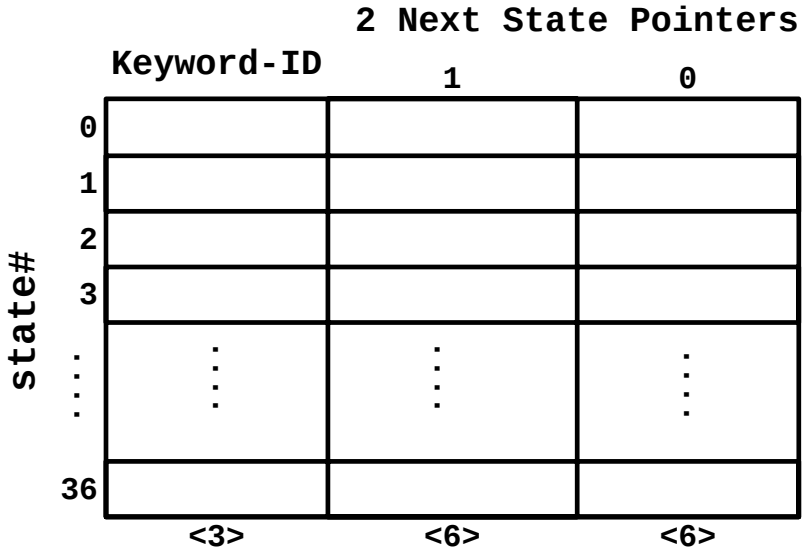
k0: el**e**men**t**s 1,1
k1: pa**r**al**l**el 1,1
k2: ma**n**age 1,x
k3: me**m**ory 1,x



[3] Piyachon and Luo, Efficient Memory Utilization on Network Processors for Deep Packet Inspection, ACM ANCS 2006

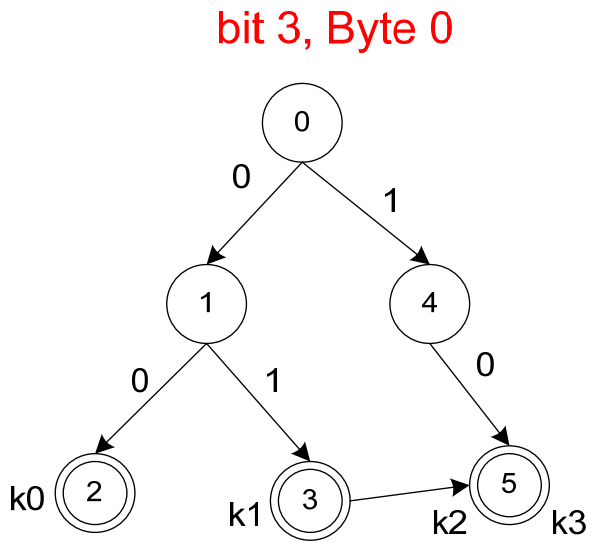
Benefits of Bit-Byte AC DFA

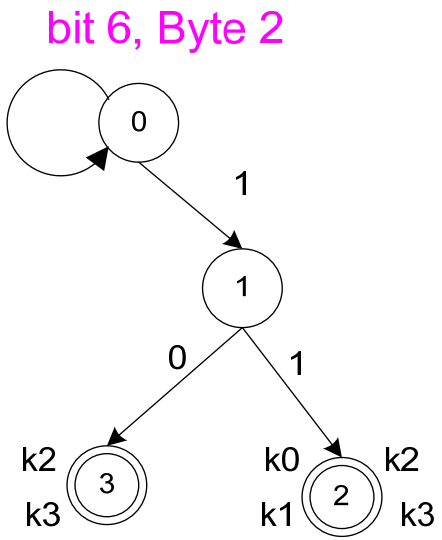
- Keeping the width of keyword-ID as low as Bit-AC DFA.
- Keeping next state pointers as small as Bit-AC DFA.
- Keeping the # of states per DFA as low as Byte-AC DFA.
- Memory Reduction
  - Snort (Dec'05): 2733 keywords
  - 4 bytes at a time
  - ≤ 36 states per DFA
  - 2 next state pointers
    - width = 6 bits
  - keyword-ID width = 3 bits
  - 29152 DFAs (= 911 x 32)
  - 29152 x 36 x (3 + 2 x 6) = 1.9 MB



Bit-Byte-DFA: Searching

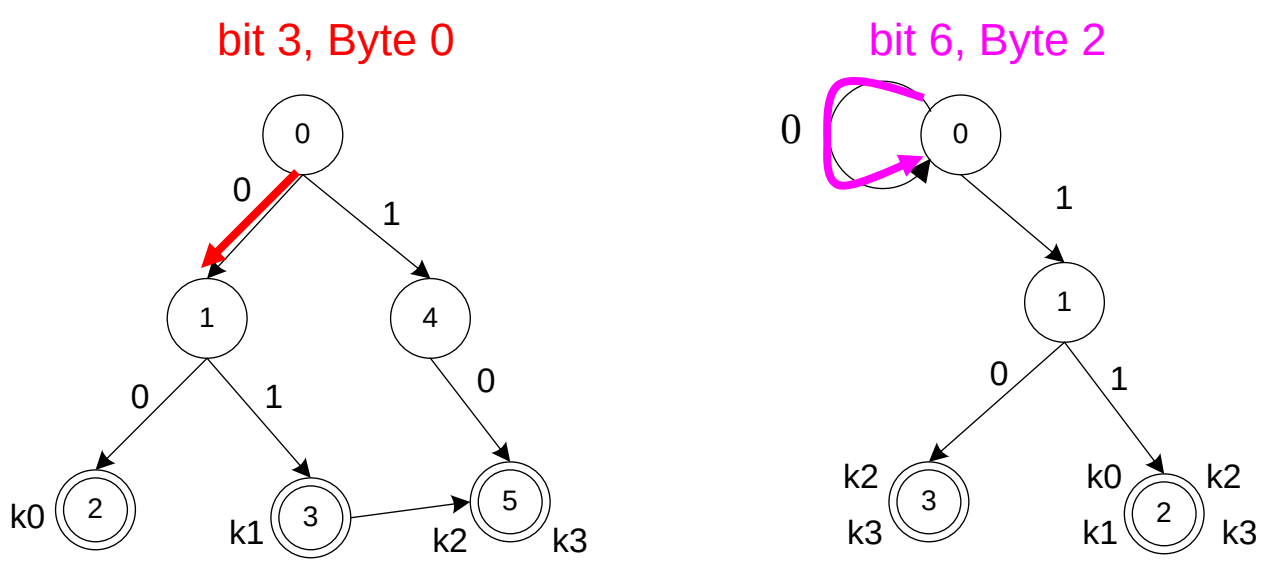
	byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3
input												





Bit-Byte-DFA: Searching

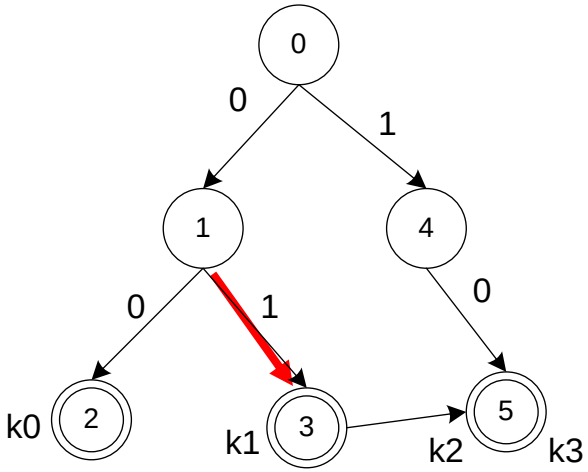
	byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3
input	a	b	1	2								
	0110 0001	0110 0010	0011 0001	0011 0010								



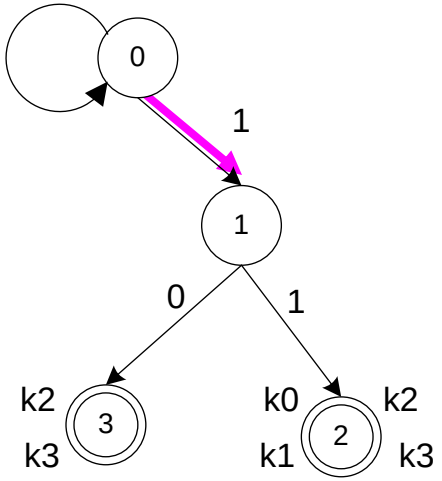
Bit-Byte-DFA: Searching

	byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3
input	a	b	1	2	m	e	m	o				
	0110 0001	0110 0010	0011 0001	0011 0010	0110 1101	0110 0101	0110 1101	0110 1111				

bit 3, Byte 0



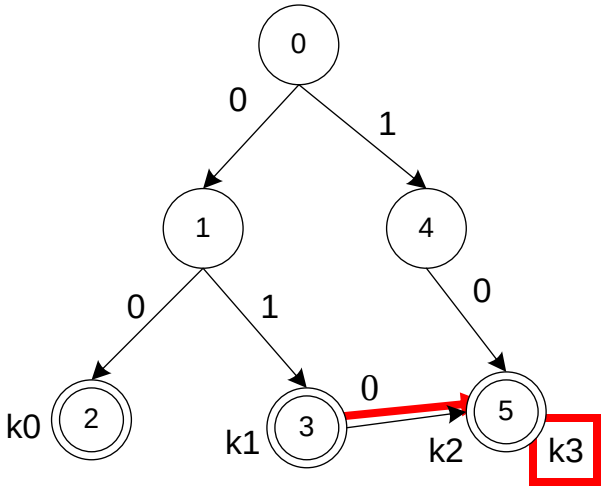
bit 6, Byte 2



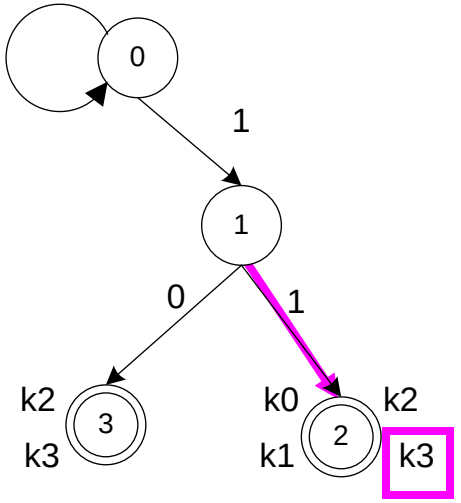
Bit-Byte-DFA: Searching

	byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3
input	a	b	1	2	m	e	m	o	r	y	e	f
	0110 0001	0110 0010	0011 0001	0011 0010	0110 1101	0110 0101	0110 1101	0110 1111	0111 0010	0111 1001	0110 0101	0110 0110

bit 3, Byte 0



bit 6, Byte 2



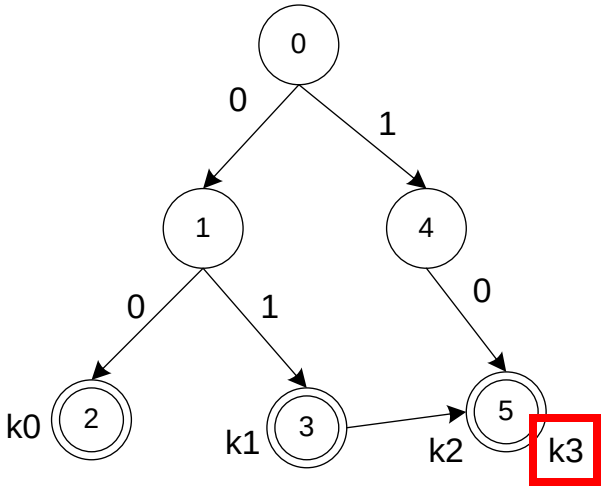
Bit-Byte-DFA: Searching

	byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3	byte0	byte1	byte2	byte3
input	a	b	1	2	m	e	m	o	r	y	e	f
	0110 0001	0110 0010	0011 0001	0011 0010	0110 1101	0110 0101	0110 1101	0110 1111	0111 0010	0111 1001	0110 0101	0110 0110

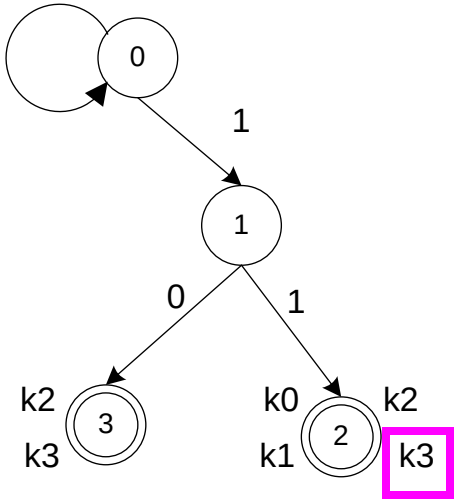
Match with 'memory'

Only when all **32** bit-DFA find the match of k3

bit 3, Byte 0

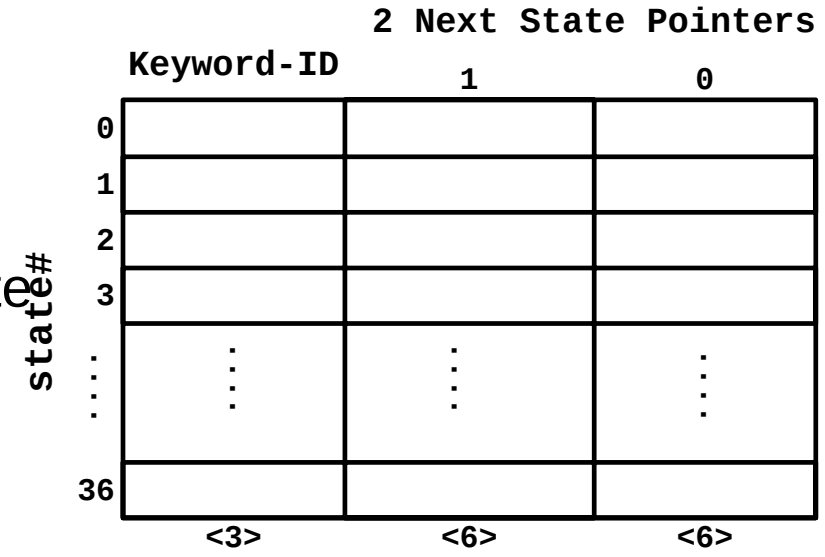


bit 6, Byte 2



Find the Optimal Setting to Minimize Memory Footprint

- When *k* = keywords per subset
  - The width of keyword-ID = *k* bits
  - *k* = 1, 2, 3, … , *K*
  - when *K* = total number of keywords
- *b* = # of bits extracted from each byte
  - *b* = 1, 2, 4, 8
  - # of next state pointers = 2^b
- *B* = # of bytes considered at a time
  - *B* = 1, 2, 3, …
  - *B* = 4 in previous example
- Total Memory (*T*) is a function of *k*, *b*, and *B.*
  - $T = f(k, b, B)$



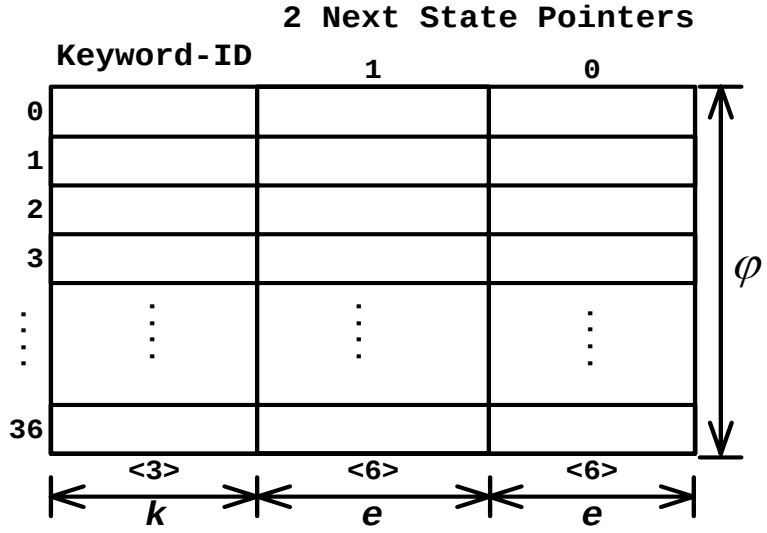
Total Memory Footprint

	Definition	
K	The number of all Keywords in the whole rule set	
k	The number of keyword in each subset	1, 2, 3, É, K
b	GroupedBit: The number of bits grouped to divide 8 bits (of a byte)	1, 2, 4, 8
B	The number of bytes considered at a time	1, 2, 3, É
θ	The number of next state pointers	2, 4, 16, 256
φ_{ij}	The number of states in i^{th} bit-level AC in i^{th} subset	
e_{ij}	The number of bits used to encode states in i^{th} bit-level AC in j^{th} subset	

when $N_{subset} = \left\lceil \frac{K}{k} \right\rceil$, $\frac{N_{bitDFA}}{N_{subset}} = \frac{8B}{b}$, $e_i = \lceil \log_2 \varphi_i \rceil$ and $\theta = 2^b$

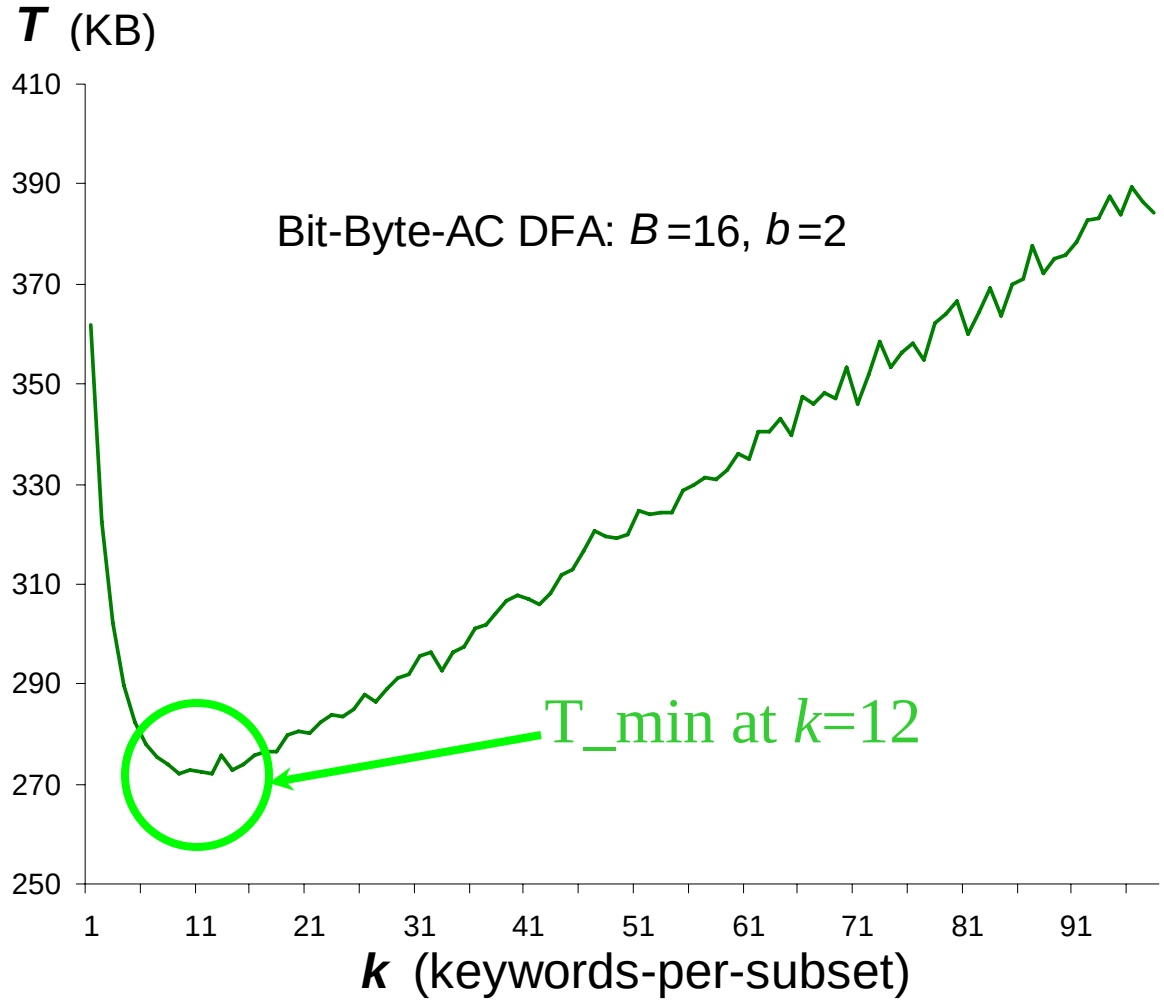
Total memory of all bit-ACs in all subset

$$T(k,b,B) = \sum_{j=1}^{\lceil K/k \rceil} \sum_{i=1}^{8B/b} \varphi_{ij} \cdot \left(k + \theta \cdot e_{ij}\right)$$

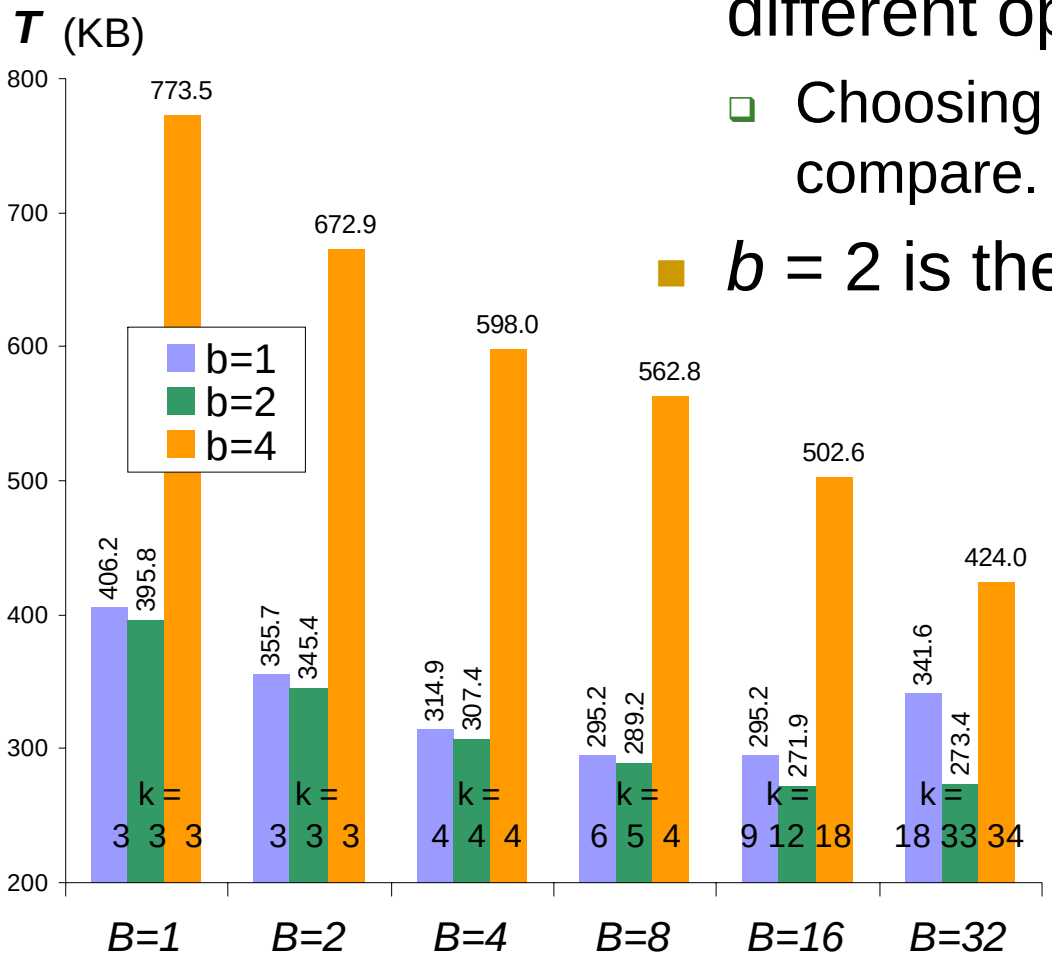


Find the optimal *k*

- Each pair of (*b*, *B)* has one optimal *k* for a minimal *T*.

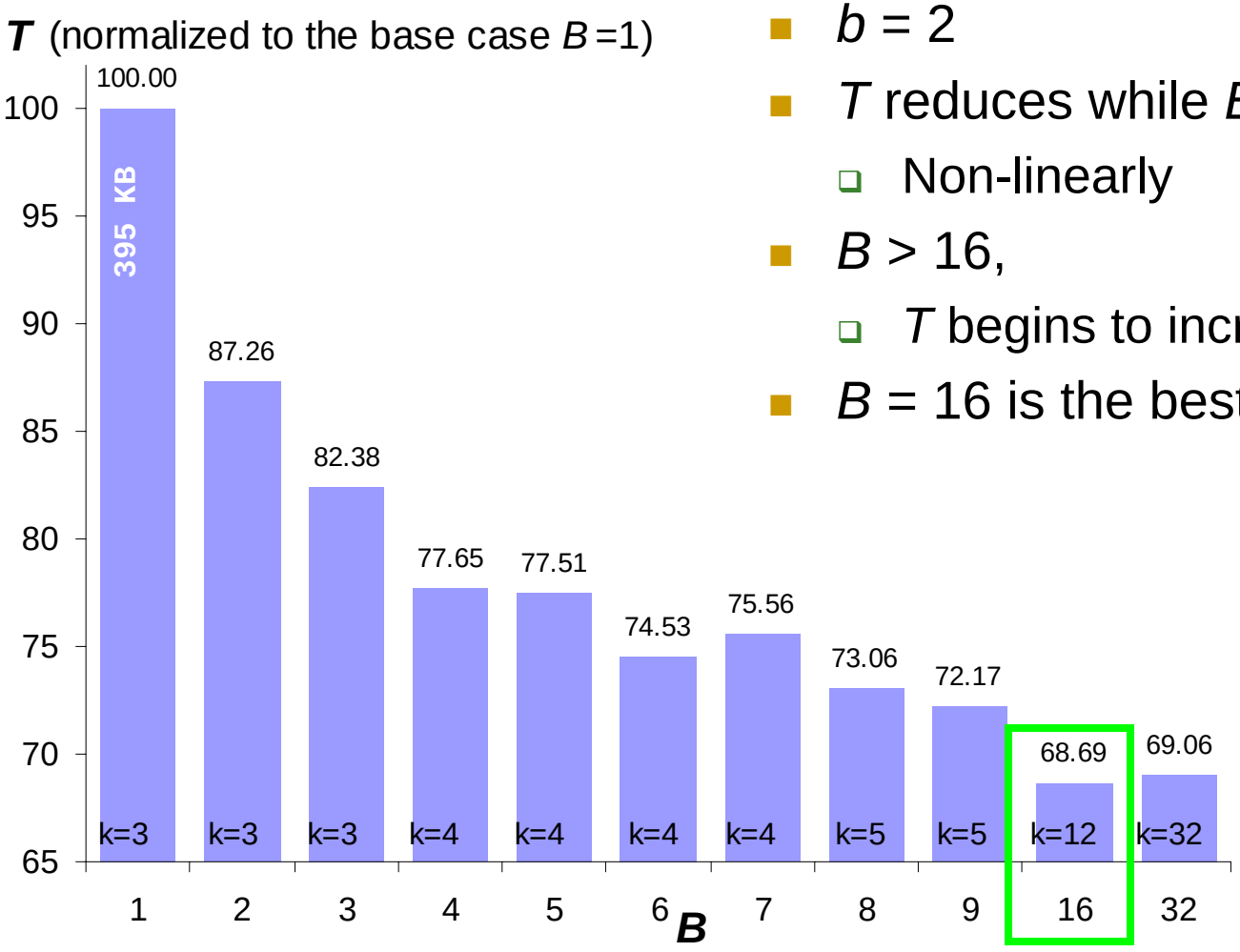


Find the optimal *b*



- Each setting of *k*, *b*, and *B* has different optimal point.
  - Choosing only the optimal setting to compare.
- *b* = 2 is the best.

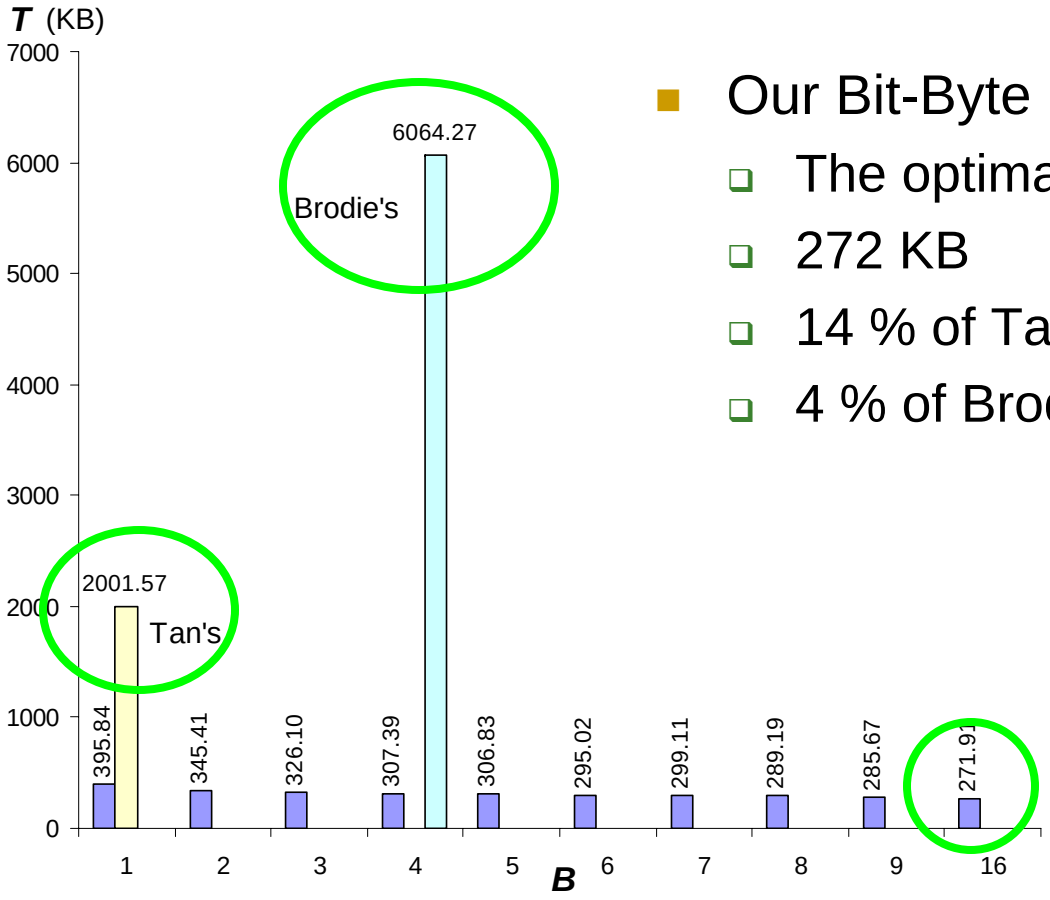
Find the optimal *B*



- $b = 2$
- T reduces while B increases.
  - Non-linearly
- $B > 16$,
  - T begins to increase.
- $B = 16$ is the best for Snort (Dec'05).

Comparing with Existing Works

- Tan-Sherwood's, Brodie-Cytron-Taylor's, and Ours

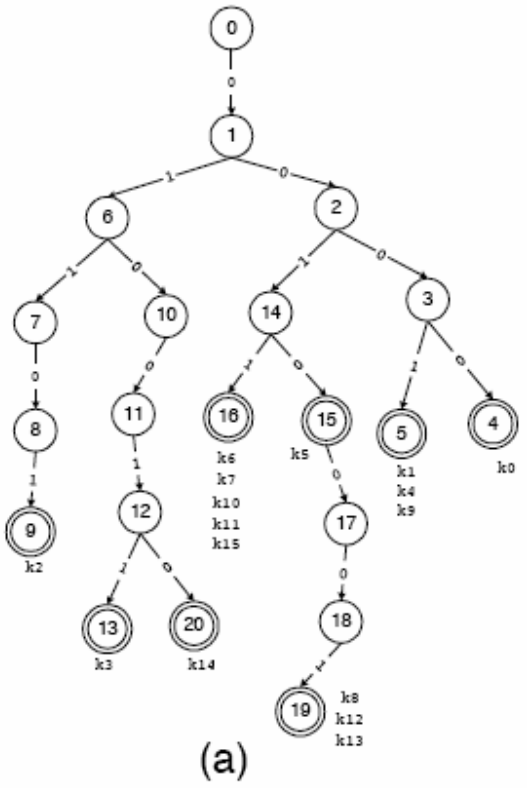


- Our Bit-Byte DFA when *B*=16
  - The optimal point at *b*=2 and *k*=12
  - 272 KB
  - 14 % of Tan's (2001 KB)
  - 4 % of Brodie's (6064 KB)

Performance Evaluation

- NePSim2
  - An open source IXP24xx/28xx simulator
  - http://faculty.uml.edu/yluo/pmwiki/index.php?n=NePSim.NePSim
- NP Architecture based on IXP2855
  - 16 MicroEngines (MEs), 8 threads per ME
  - 512 KB on-chip memory
  - 1.4 GHz
- Bit-Byte AC DFA: b=2, B=16, k=12
  - T = 272 KB
  - Up to 5 Gbps

Ongoing Work (I): Compact DFAs through State Re-labeling



state #	keywordID Matrix <16>	Next State Matrix 1 <9>	0 <9>
0	0000 0000 0000 0000	0	1
1	0000 0000 0000 0000	6	2
2	0000 0000 0000 0000	14	3
3	0000 0000 0000 0000	5	4
4	0000 0000 0000 0001	5	4
5	0000 0010 0001 0010	7	10
6	0000 0000 0000 0000	7	10
7	0000 0000 0000 0000	0	8
8	0000 0000 0000 0000	9	0
9	0000 0000 0000 0100	0	0
10	0000 0000 0000 0000	0	11
11	0000 0000 0000 0000	12	0
12	0000 0000 0000 0000	13	20
13	0000 0000 0000 1000	0	0
14	0000 0000 0000 0000	16	15
15	0000 0000 0010 0000	0	17
16	1000 1100 1100 0000	0	0
17	0000 0000 0000 0000	0	18
18	0000 0000 0000 0000	19	0
19	0011 0001 0000 0000	0	0
20	0100 0000 0000 0000	0	0
21	0000 0000 0000 0000	0	0
22	0000 0000 0000 0000	0	0
...	...	...	...
...	...	...	...
...	...	...	...
511	0000 0000 0000 0000	0	0

Piyachon and Luo, Compact State Machines for High Performance Pattern Matching, In submission.

Ongoing Work (II): High Performance Regular Expression Matching with FPGAs

- Motivation
  - Massive parallel logic units
  - Abundant parallel memory elements on chip
- Techniques
  - Compact DFA storage using banked memory
  - Parallel matching engine
  - Probabilistic state clustering

Prior Works on Network Processors

- SpliceNP: A content-aware switch based on IXP2400 NP
  - using TCP splicing technique
  - 2~5.7x throughput improvement over Linux box
  - *ACM ANCS'05*
- Power saving on NP based routers
  - Clock-gating PEs based on traffic variation during time of a day.
  - *ACM DAC'05*
- NePSim: the first cycle-accurate open source network processor simulator with power estimation.
  - Being downloaded (1500+ downloads) and used worldwide
  - *IEEE Micro Sept/Oct, 2004*

Thank you !

Question?

Yan_Luo@uml.edu

978-934-2592