清華大學

Tsinghua University

A new novel method in Signature Matching area

Presenter

Ali（阿里）

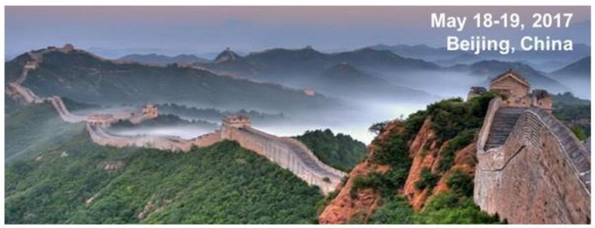
1 March 2017

Introduction

ACM/IEEE Symposium on Architectures for Networking and Communications Systems

- Overview
- Call for Papers (PDF)
- Call for Posters (PDF)
- Paper Submission
- Committees
- ANCS Email List
- Past Events

ANCS 2017 Conference



The 13th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '17) will be held **May 18-19** in **Beijing, China**.

ACM/IEEE ANCS is the premier forum for presenting and discussing original research that explores the relationship between the algorithms and architectures of data communication networks and the hardware and software elements from which these networks are built, including both experimental and theoretical analysis. To recognize and foster the increasing importance of research into the co-design of computer and network systems, the conference also places an emphasis on systems issues arising from the interaction of computer and network architectures.

News

February 17, 2017	Call For Posters is online.
Dec 17, 2016	Paper submission site is online.
Oct 11, 2016	Call For Papers is online.
Sep 30, 2016	ANCS 2017 website is online.

Important Dates

Papers - Abstract submission deadline	January 7, 2017 (optional)
Papers - Full paper submission deadline	January 21, 2017
Papers - Notification of acceptance	March 8, 2017
Papers - Camera-ready submission deadline	April 1, 2017
Posters - Abstract submission deadline	March 20, 2017
Posters - Notification of acceptance	April 3, 2017
Posters - Camera-ready submission deadline	April 10, 2017

Sponsors

SIGARCH acm sigcomm IEEE computer society HUAWEI ZTE TNList

Contacts

Web Chair: Zhi Liu

Introduction – Contd.

ANCS 2016

Introduction – Contd.

ANCS 2016

O^{3}FA: A Scalable Finite Automata-based Pattern-Matching Engine for Out-of-Order Deep Packet Inspection

Outline

- Introduction to Signature Matching
- O^3FA Design
  - Overview
  - Example
  - Optimization
  - Components
- Evaluation

Introduction to SM

Signature Matching

- IDS
- IPS
- DPI
- …

Pattern Matching

As Fast As Possible!

Regular Expressions

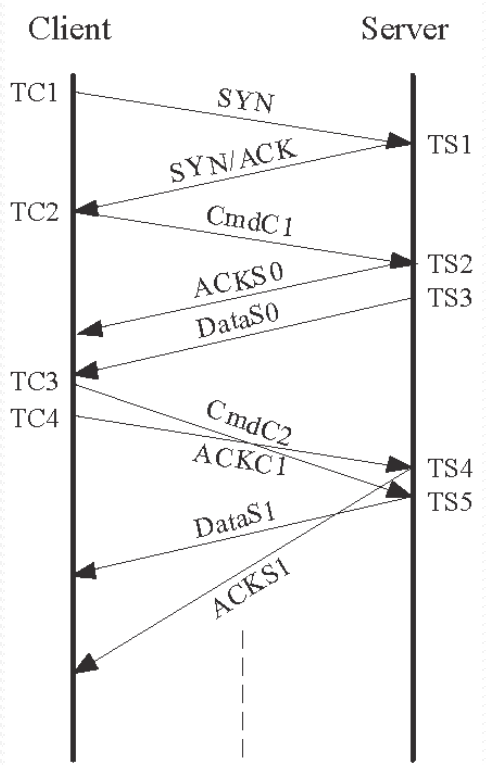
A.*B.{C+}

DFA vs NFA

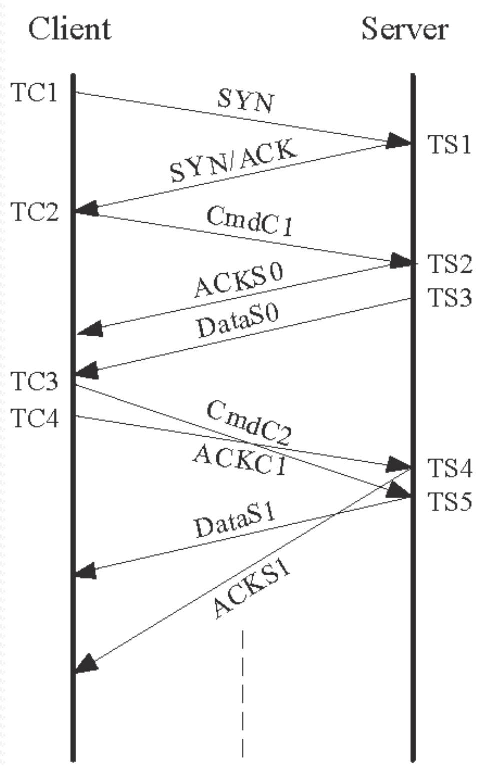
- **DFA Based**
  - Running → Fast
  - Storage → Awful

- **NFA Based**
  - Running → Awful
  - Storage → Compact
- **Others**

Re-ordering Packets

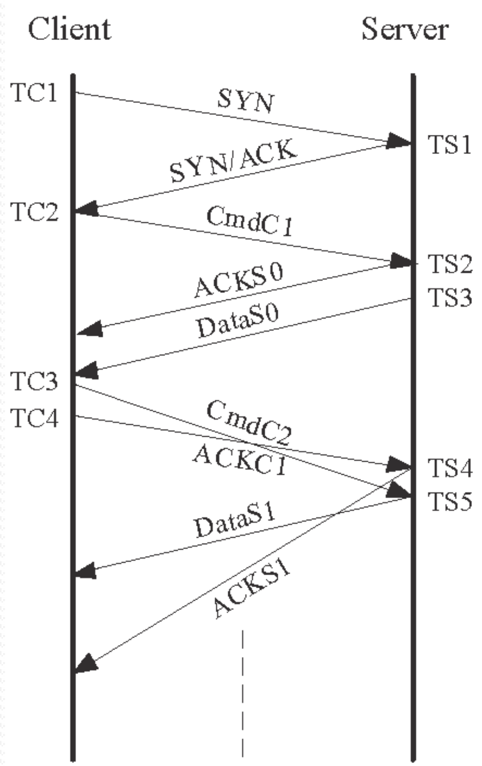


Re-ordering Packets



- **Pattern**
  - $b.*\, cde$
- **Input packets**
  - $P_3: dead$
  - $P_1: caba$
  - $P_2: dcac$

Re-ordering Packets

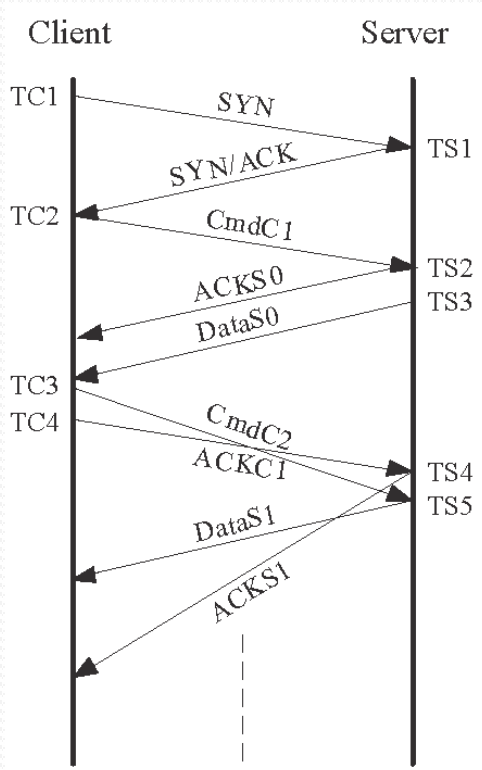


- **Pattern**
  - $b.* cde$
- **Input packets**
  - $P_3: dead$
  - $P_1: caba$
  - $P_2: dcac$

Flow re-assembly

$cabadcacdead$

Re-ordering Packets



- **Pattern**
  - $b.*\ cde$
- **Input packets**
  - $P_3: dead$
  - $P_1: caba$
  - $P_2: dcac$

Flow re-assembly

O^3FA Design

O^3FA overview

- **Pattern**
  - $b.*\,cde$
- **Input packets**
  - $P_3: dead$
  - $P_1: caba$
  - $P_2: dcac$

O^3FA overview – Contd.

- **Pattern**
  - $b.*\,cde$
- **Input packets**
  - $P_3: dead$
  - $P_1: caba$
  - $P_2: dcac$

- **Observation 1**
  - If a regular expression R is matched across a set of packets $P_1, .., P_N$, then
    - The suffix of P_1 must match a prefix of R
    - The prefix of P_N must match a suffix of R.

O^3FA overview – Contd.

- **Pattern**
  - $b.*\ cde$
- **Input packets**
  - $P_3: dead$
  - $P_1: caba$
  - $P_2: dcac$

- **Observation 1**
  - If a regular expression *R* is matched across a set of packets $P_1,.., P_N$, then
    - The suffix of P_1 must match a prefix of *R*
    - The prefix of P_N must match a suffix of *R*.

✓ Detect suffix *de* in P_3.
✓ Detect prefix *ba* in P_1.
✓ Concatenate detected segments with P_2.

badcacde

O^{3}FA overview – Contd.

Observation 2

- Given a regular expression *R* in the form sp1.*sp2 and an input stream of the form *M1M*M2*, where *M1* matches *sp1* and *M2* matches *sp2*,
  - any modifications to *I* that substitutes *M** with a shorter segment will not affect the match outcome.

O^3FA overview – Contd.

- **Pattern**
  - $b.*cde$
- **Input packets**
  - $P_{10}: dead$
  - $P_1: caba$
  - $P_3, P_4, P_5, P_6, P_7, P_8, P_9$
  - $P_2: dcac$

O^3FA overview – Contd.

- **Pattern**
  - $b.*cde$
- **Input packets**
  - $P_{10}: dead$
  - $P_1: caba$
  - $P_3, P_4, P_5, P_6, P_7, P_8, P_9$
  - $P_2: dcac$

Suffix(P_1) | P_2 | prefix(P_{10})

badcacde

O^3FA overview

- One or more DFAs
- Supporting FAs
  - Prefix FAs
  - Suffix FAs

O^3FA example

*abc.*def, ghk*

1. *Creating sub-patterns*
   - .*abc.*
   - .*def.*
   - .*ghk

O³FA example – contd.

2. *Creating prefix and suffix sets*

- .*abc.*
  - prefix: {*.*abc.*, .*abc, .*ab, .*a*}
  - Suffix: {*.*abc.*, abc.*, bc.*, c.**}
- .*def.*
  - prefix: {*.*def.*,.*def, .*de, .*d*}
  - Suffix: {*.*def.*, def.*, ef.*, f.**}
- .*ghk
  - prefix: {*.*ghk, .*gh, .*g*}
  - Suffix: {*.*ghk, ghk, hk, k*}

O^{3}FA example – contd.

3. *Simplifications in the prefix and suffix sets*

- Prefix Set: {*.*abc.*, .*abc, .*ab, .*a, .*def.*,.*def, .*de, .*d, .*ghk, .*gh, .*g*}
- Suffix Set: {*.*abc.*, abc.*, bc.*, c.*, .*def.*, def.*, ef.*, f.*, .*ghk, ghk, hk, k*}

O³FA example – contd.

3. *Simplifications in the prefix and suffix sets*

- Prefix Set: {*.*abc.*, .*abc, .*ab, .*a, .*def.*,.*def, .*de, .*d, .*ghk, .*gh, .*g*}
- Suffix Set: {*.*abc.*, abc.*, bc.*, c.*, .*def.*, def.*, ef.*, f.*, .*ghk, ghk, hk, k*}

Observation 1

⇓

- Prefix Set: {*.*abc.*, .*abc, .*ab, .*a, .*def.*,.*def, .*de, .*d, .*ghk, .*gh, .*g*}
- Suffix Set: {*.*abc, abc, bc, c, .*def, def, ef, f, .*ghk, ghk, hk, k*}

O³FA example – contd.

3. Simplifications in the prefix and suffix sets

- Prefix Set: {*.*abc.*, .*abc, .*ab, .*a, .*def.*,.*def, .*de, .*d, .*ghk, .*gh, .*g*}
- Suffix Set: {*.*abc, abc, bc, c, .*def, def, ef, f, .*ghk, ghk, hk, k*}

O^{3}FA example – contd.

3. *Simplifications in the prefix and suffix sets*

- Prefix Set: {.*abc.*, .*abc, .*ab, .*a, .*def.*,.*def, .*de, .*d, .*ghk, .*gh, .*g}
- Suffix Set: {.*abc, abc, bc, c, .*def, def, ef, f, .*ghk, ghk, hk, k}

Removing similar elements

- Prefix Set: {.*abc.*, .*ab, .*a, .*def.*, .*de, .*d, .*gh, .*g}
- Suffix Set: {.*abc, abc, bc, c, .*def, def, ef, f, .*ghk, ghk, hk, k}

O^{3}FA example – contd.

3. *Simplifications in the prefix and suffix sets*

- Prefix Set: {*.*abc.*, .*ab, .*a, .*def.*, .*de, .*d, .*gh, .*g*}
- Suffix Set: {*.*abc, abc, bc, c, .*def, def, ef, f, .*ghk, ghk, hk, k*}

O^3FA example – contd.

3. *Simplifications in the prefix and suffix sets*

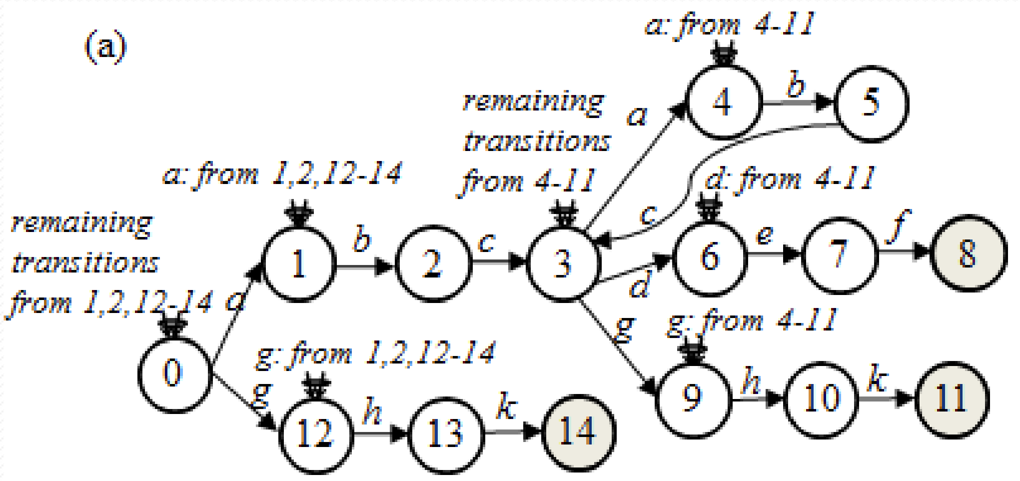
- Prefix Set: {*.*abc.*, .*ab, .*a, .*def.*, .*de, .*d, .*gh, .*g*}
- Suffix Set: {*.*abc, abc, bc, c, .*def, def, ef, f, .*ghk, ghk, hk, k*}

Removing elements that are covered by more general elements in the same set

- Prefix Set: {*.*abc.*, .*ab, .*a, .*def.*, .*de, .*d, .*gh, .*g*}
- Suffix Set: {*.*abc, bc, c, .*def, ef, f, .*ghk, hk, k*}

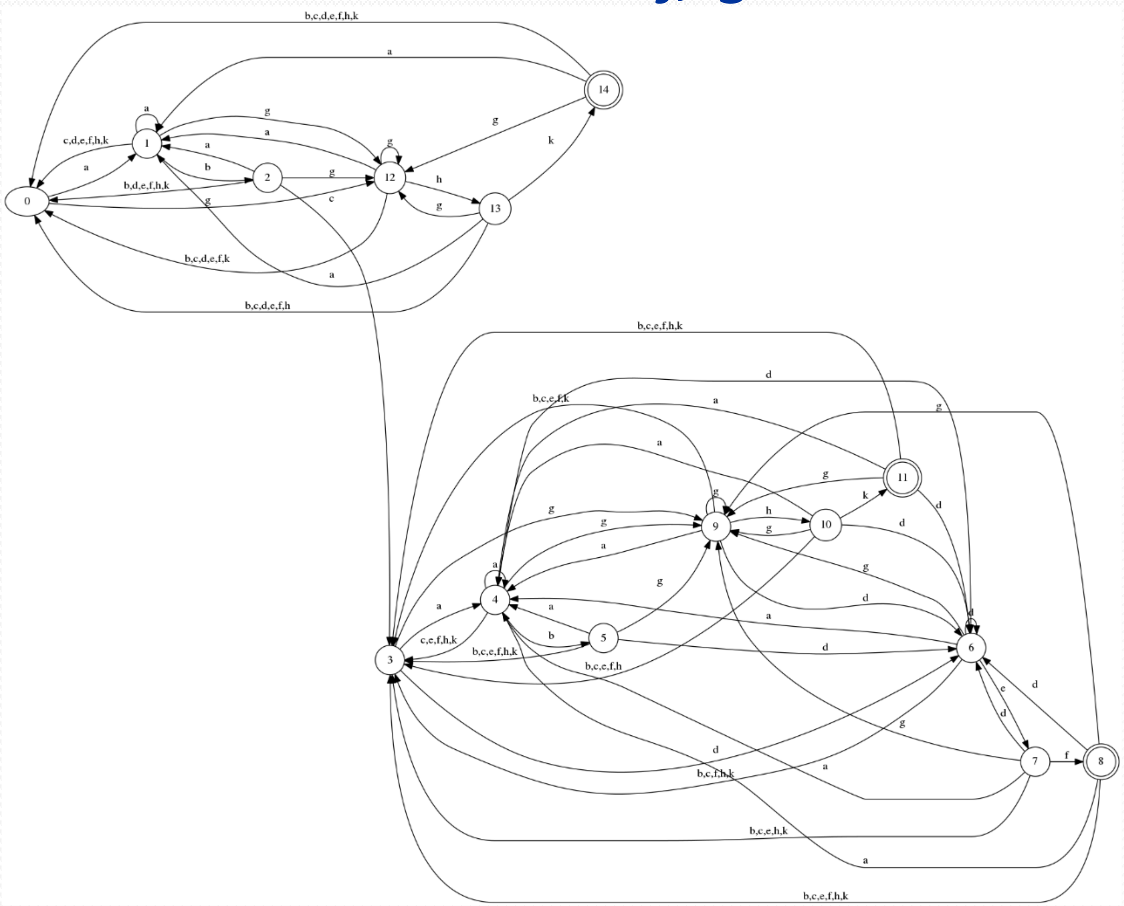
O³FA example – DFA

*abc.*def, ghk*



O³FA example – DFA (Contd.)

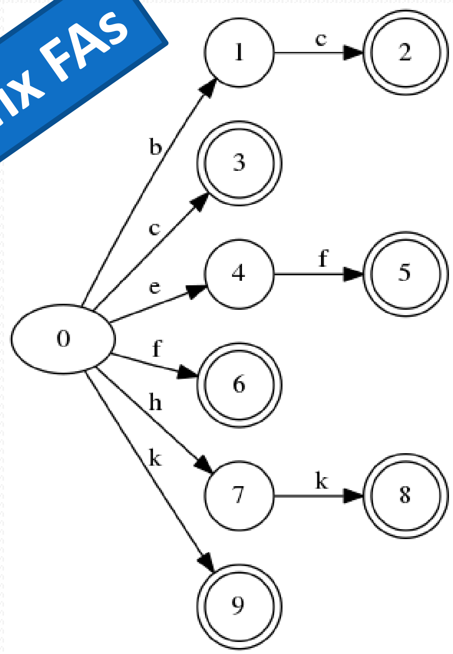
*abc.*def, ghk*

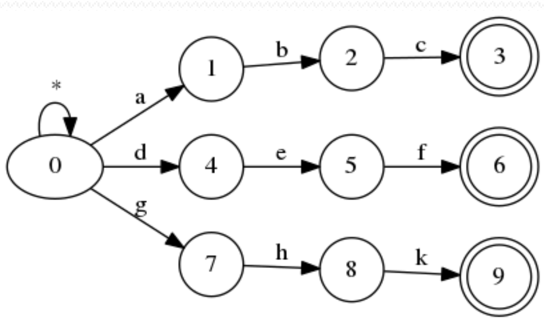


O3FA example – Prefix and Suffix FAs

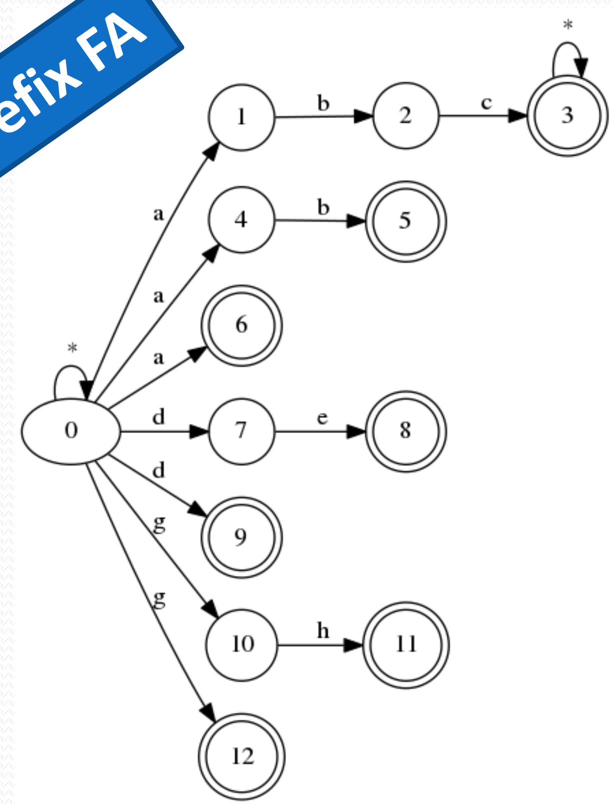
*abc.*def, ghk*

Suffix FAs





Prefix FA

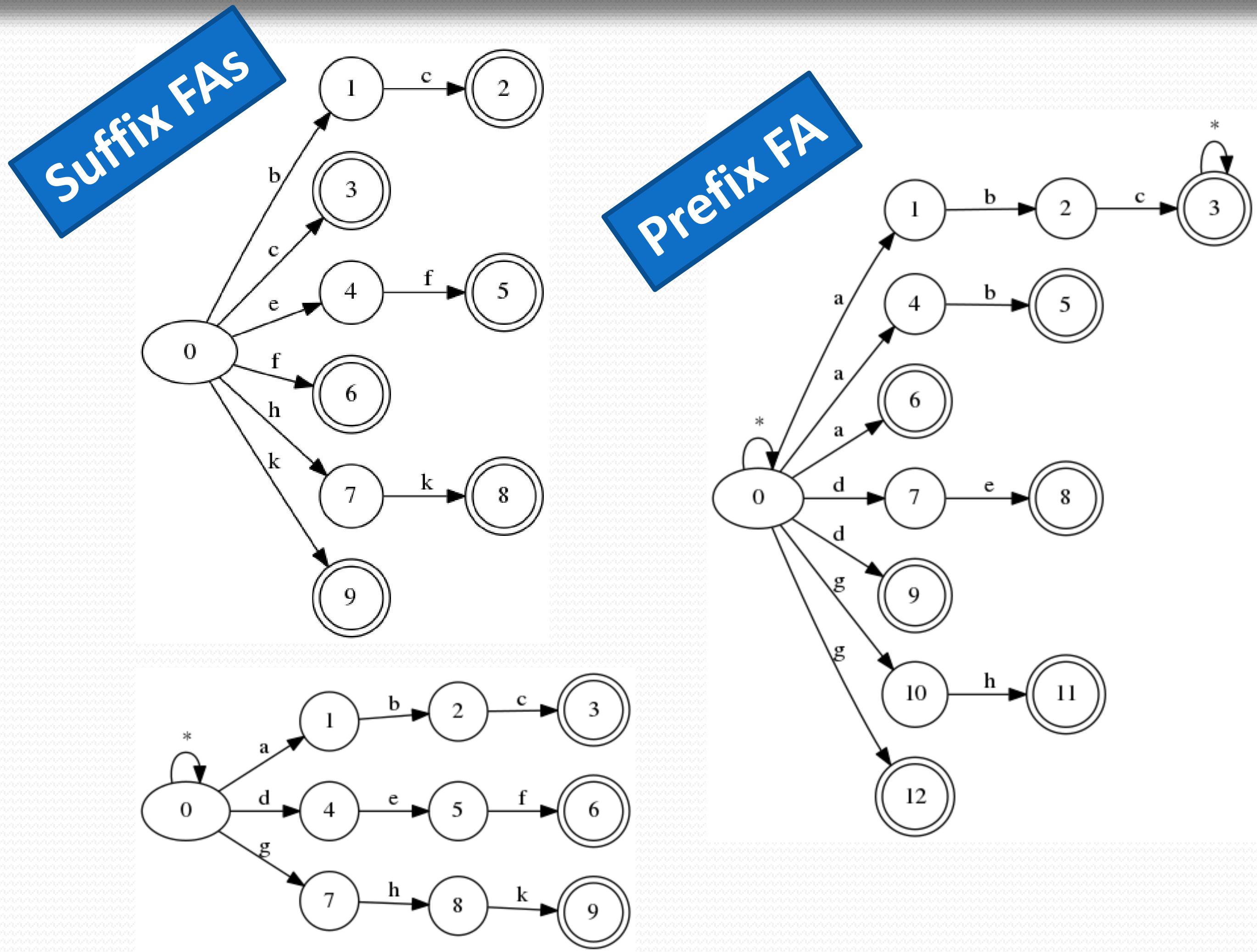


O^3FA example – Suffix FAs

P_3 = *adef*

P_1 = *bhab*

P_2 = *cegh*

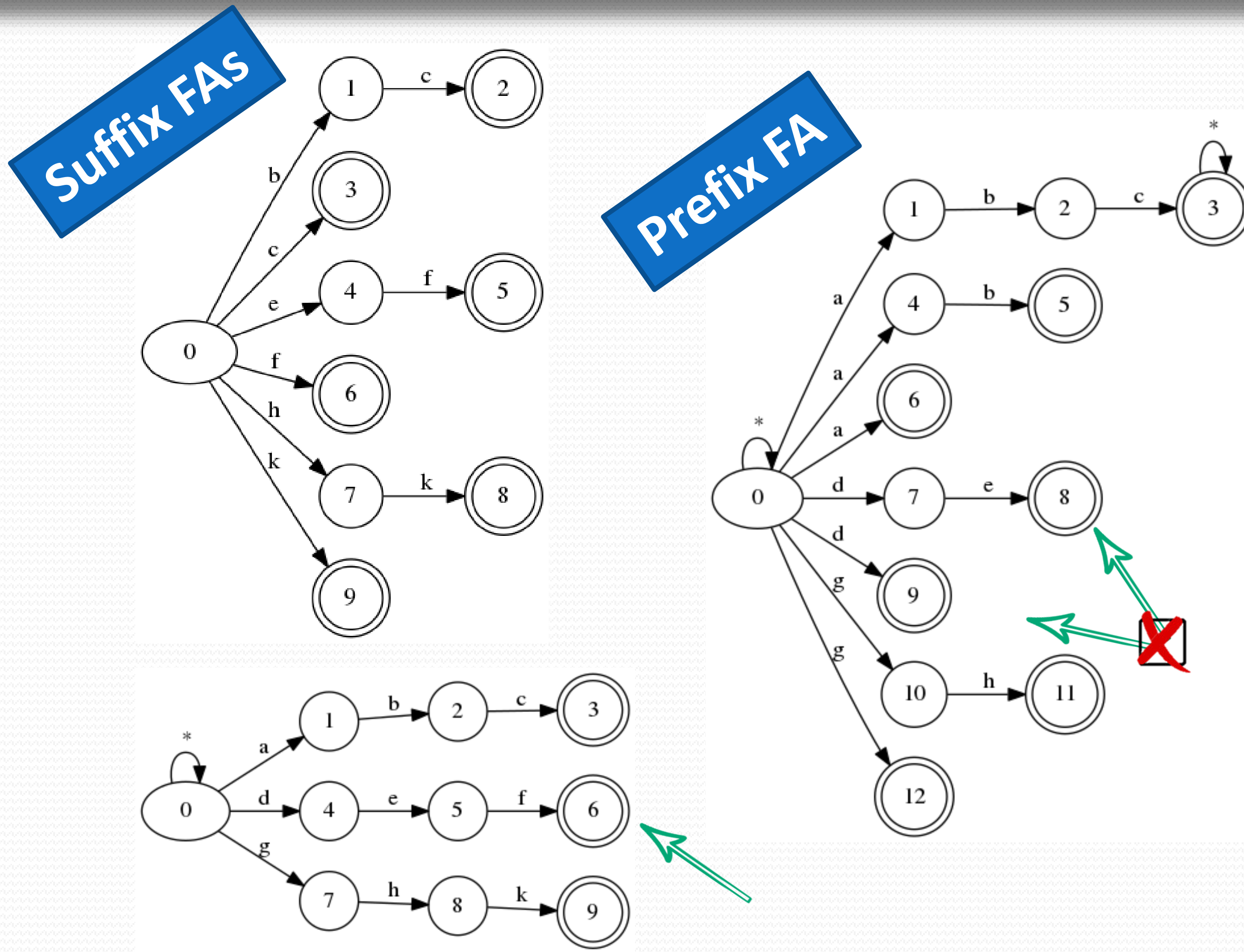


O^3FA example – Suffix FAs

$P_3 = adef$

$P_1 = bhab$

$P_2 = cegh$



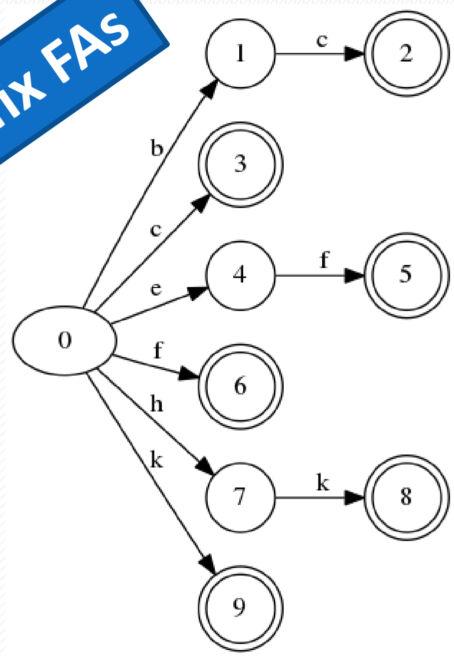
O^3FA example – Suffix FAs

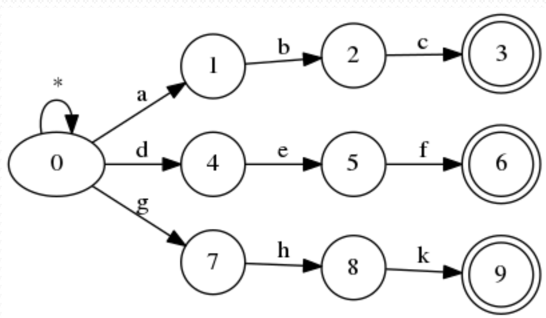
$P_3 = adef$

$P_1 = bhab$

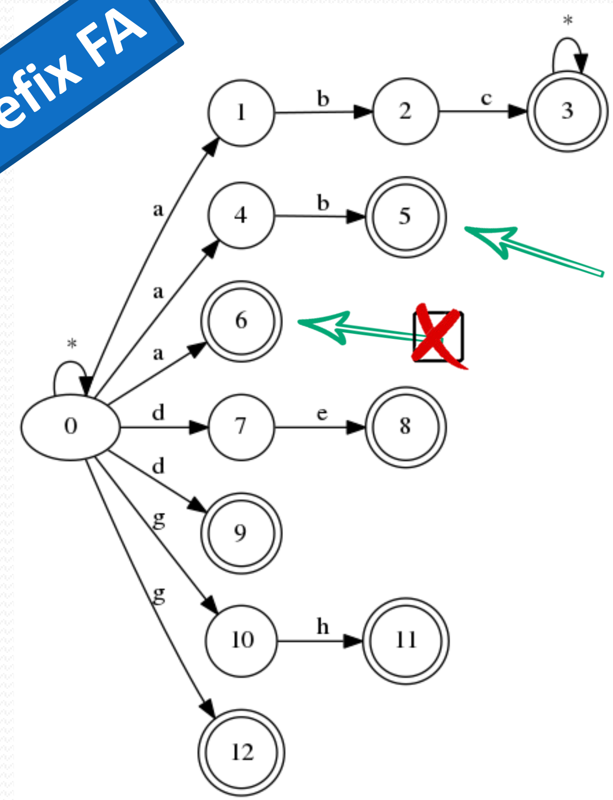
$P_2 = cegh$

Suffix FAs





Prefix FA



O^3FA example – Suffix FAs

$P_3 = adef$

$P_1 = bhab$

$P_2 = cegh$ ⟹

Suffix(P_1) | P_2 | prefix(P_3)

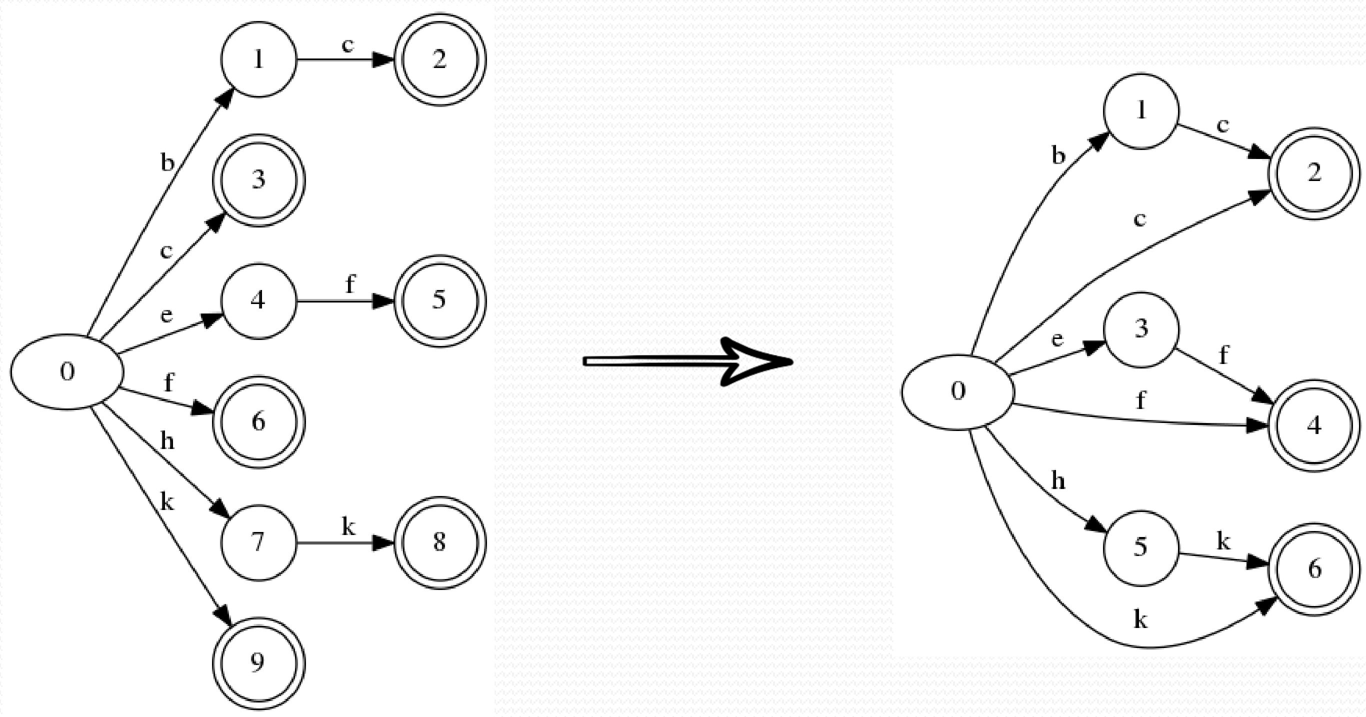
abceghdef

O^3FA: Optimizations

- Using Index tags to avoid false positive

O³FA: Optimizations

- Using Index tags to avoid false positive
- Compressing Suffix FAs



O^{3}FA: Optimizations

- Using Index tags to avoid false positive
- Compressing Suffix FAs
- Creating DFA instead of NFA from suffix or prefix sets

Evaluation

Experimental Setup

- 176 backdoor rules
- 304 spyware rules
- 500 dot-star rules (5%, 10%, and 20%)
- 500 range-star rules (50% and 100%)
- 500 exact match rules

- 16 input streams
- Break 1MB input streams to 64KB packets
- Re-order packets

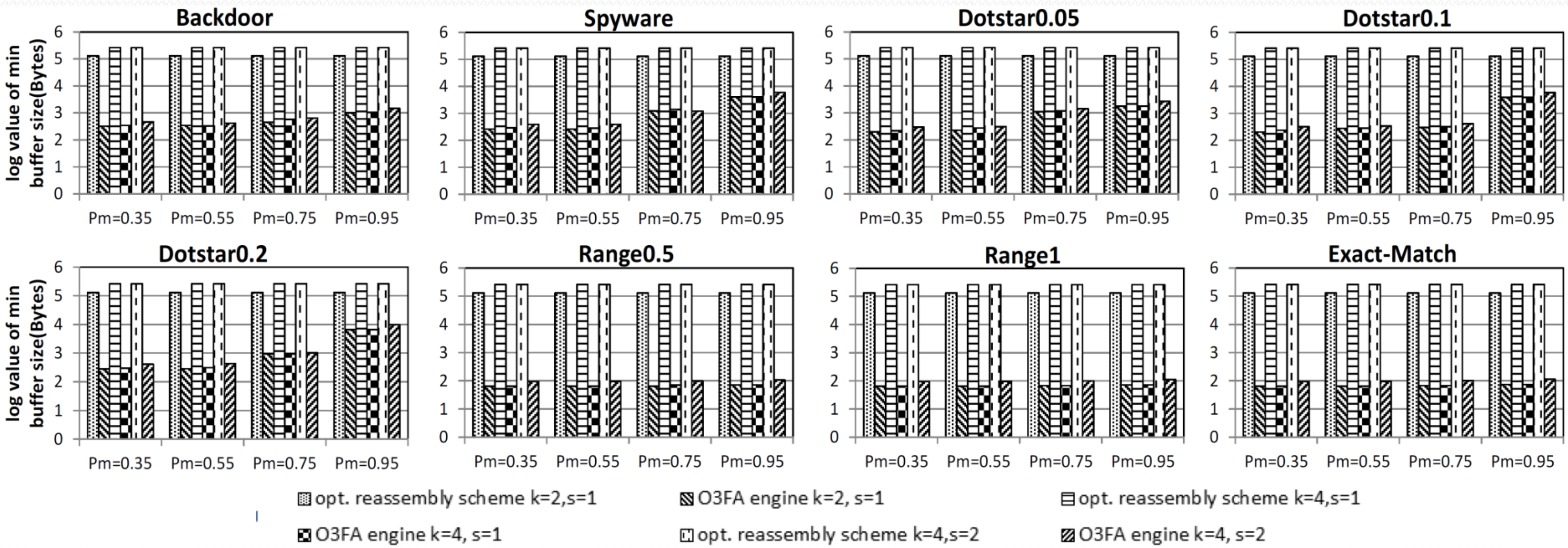
Analysis Aspects

- *O^3FA Memory Footprint*
- *Buffer Size Savings*
- *Memory Bandwidth Overhead*
- *Traversal Overhead*

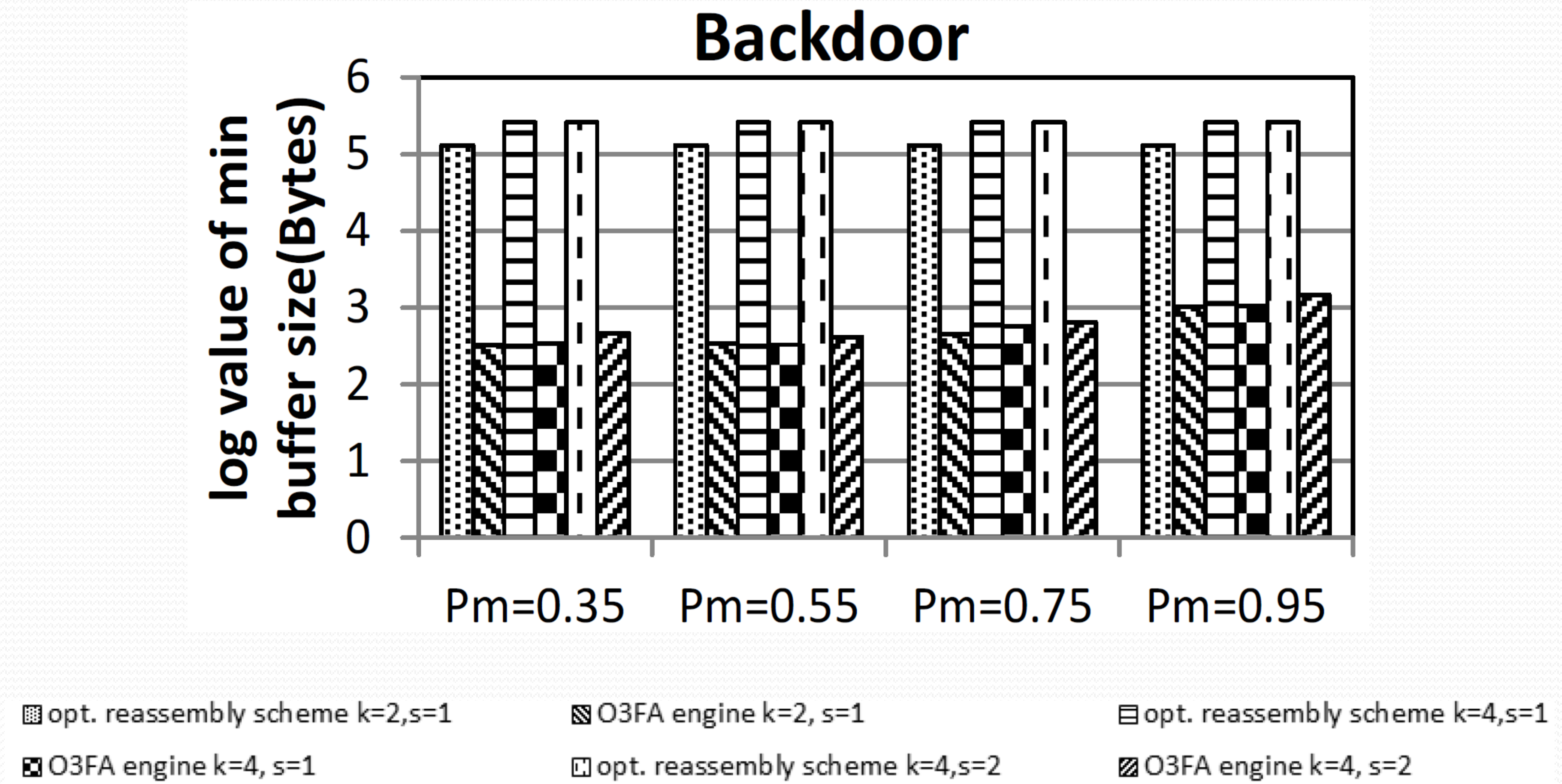
O³FA Memory Footprint

Dataset	FA Kernel			
	Regular multi-DFAs		Supporting-FAs	
	# of DFA	Memory Footprint	# of FA States	Memory Footprint
Backdoor	8	60	4k	0.62
Spyware	10	56	12k	1.35
Dotstar0.05	15	26	26k	3.58
Dotstar0.1	8	60	25k	3.12
Dotstar0.2	14	100	23k	2.76
Range0.5	1	5.6	24k	2.43
Range1	1	5.8	24k	2.05
Exact-match	1	4.7	17k	1.92

Buffer Size Savings



Buffer Size Savings – Contd.



Memory Bandwidth Overhead

Table 2. Ratio between the number of csNFA states traversed and the number of input characters processed (%)

Dataset	k=2, s=1				k=4, s=1				k=4, s=2			
	P_M=0.35	P_M=0.55	P_M=0.75	P_M=.95	P_M=0.35	P_M=0.55	P_M=0.75	P_M=0.95	P_M=0.35	P_M=0.55	P_M=0.75	P_M=0.95
Backdoor	0.0144	0.0202	0.0278	0.0349	0.0347	0.0337	0.0440	0.1010	0.0144	0.0202	0.0278	0.0349
Spyware	0.0590	0.1002	0.1163	0.1286	0.1158	0.1942	0.2188	0.1853	0.0590	0.1002	0.1163	0.1286
Dotstar0.05	0.0804	0.0838	0.1173	0.2595	0.1733	0.1394	0.1517	0.3927	0.0804	0.0838	0.1173	0.2595
Dotstar0.1	0.0526	0.0715	0.1054	0.2610	0.1129	0.1184	0.1701	0.3974	0.0526	0.0715	0.1054	0.2610
Dotstar0.2	0.0363	0.0611	0.1142	0.2977	0.0531	0.1112	0.1806	0.3622	0.0363	0.0611	0.1142	0.2977
Range0.5	0.0973	0.1015	0.2170	0.2238	0.1839	0.1865	0.3488	0.3831	0.0973	0.1015	0.2170	0.2238
Range1	0.0638	0.1180	0.2181	0.3927	0.1697	0.1910	0.3319	0.6929	0.0638	0.1180	0.2181	0.3927
E-M	0.0391	0.0627	0.1460	0.3140	0.1407	0.1395	0.1959	0.4374	0.0391	0.0627	0.1460	0.3140

Traversal Overhead

Ratio between Number of extra character processed and the size of the input stream

Dataset	k=2, s=1				k=4, s=1				k=4, s=2			
	P_M=0.35	P_M=0.55	P_M=0.75	P_M=0.95	P_M=0.35	P_M=0.55	P_M=0.75	P_M=0.95	P_M=0.35	P_M=0.55	P_M=0.75	P_M=0.95
Backdoor	0.0114	0.0102	0.0346	0.3277	0.0211	0.0139	0.0732	0.5140	0.0119	0.0076	0.0288	0.3376
Spyware	0.0059	0.0058	0.1333	2.4362	0.0101	0.0090	0.2635	3.6701	0.0057	0.0049	0.0753	2.4427
Dotstar0.05	0.0103	0.0076	0.2645	1.0132	0.0220	0.0389	0.4492	1.5218	0.0134	0.0221	0.2679	1.0135
Dotstar0.1	0.0041	0.0129	0.0116	2.2671	0.0120	0.0304	0.0183	3.3866	0.0073	0.0136	0.0111	2.2464
Dotstar0.2	0.0083	0.0092	0.0160	3.4655	0.0164	0.0173	0.0225	5.2268	0.0098	0.0101	0.0112	3.4838
Range0.5	0.0007	0.0011	0.0032	0.0137	0.0017	0.0020	0.0054	0.0214	0.0009	0.0012	0.0033	0.0128
Range1	0.0006	0.0011	0.0033	0.0123	0.0014	0.0020	0.0051	0.0153	0.0008	0.0012	0.0033	0.0102
E-M	0.0006	0.0011	0.0033	0.0168	0.0014	0.0022	0.0054	0.0214	0.0008	0.0012	0.0033	0.0159

Summary & Discussion

- O^3FA memory footprint is less than 100MB
- O^3FA state buffers can be 20x - 4000x smaller
- O^3FA bandwidth is linear in the number of incoming characters
- O^3FA traversal efficiency is comparable to that of conventional flow reassembly methods

- Supports only variable length operators
  - .*, C*, [C_i, C_j]*, .+, C+, …

Thanks for your attention!