Host-Based Anomaly Detection

Wahab Hamou-Lhadj, PhD, ing.
Software Behaviour Analysis (SBA) Research Lab
ECE, Concordia University
Montreal, QC, Canada
wahab.hamou-lhadj@concordia.ca

Feb. 6, 2014
Ottawa, ON, Canada

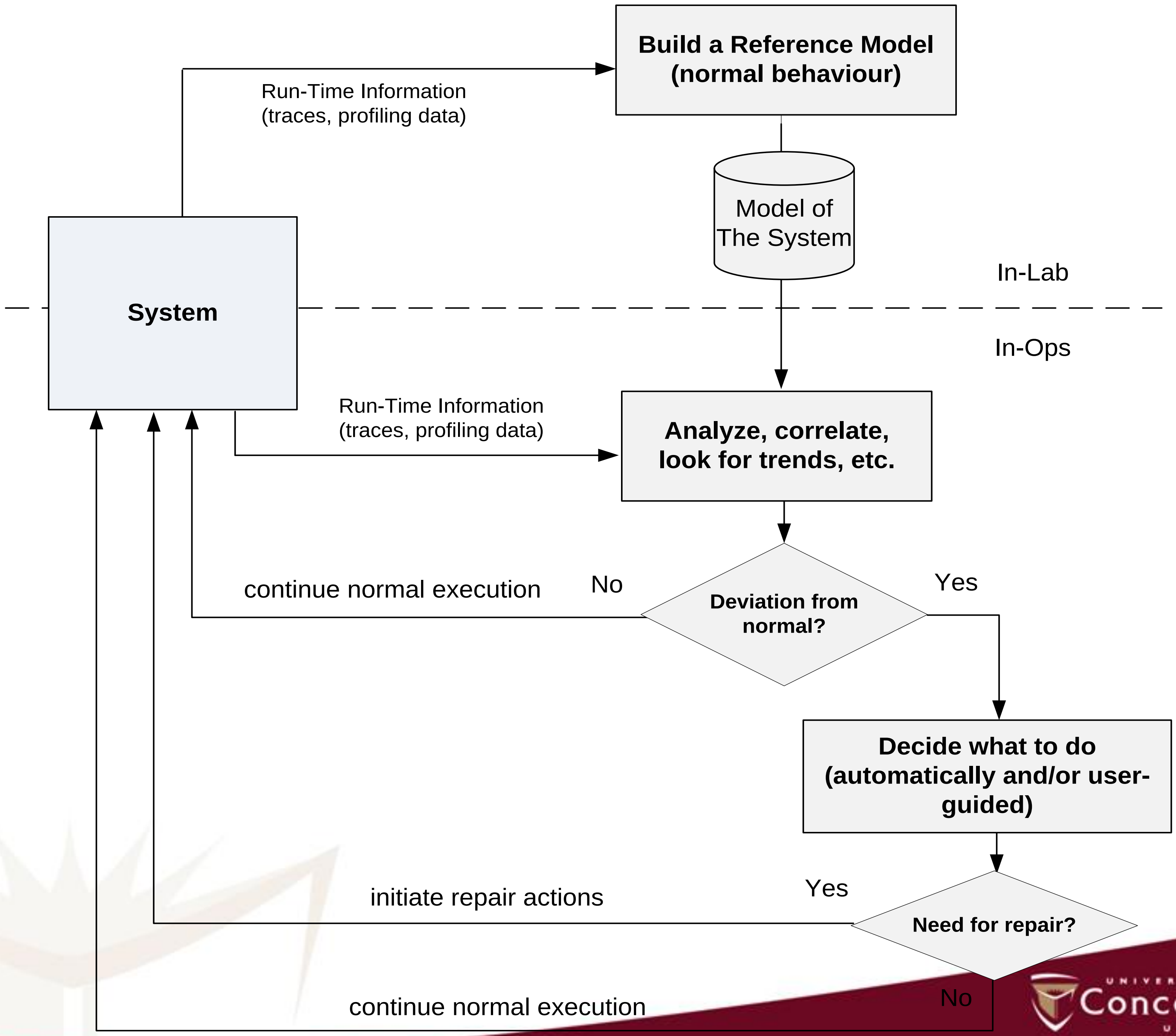
Objectives

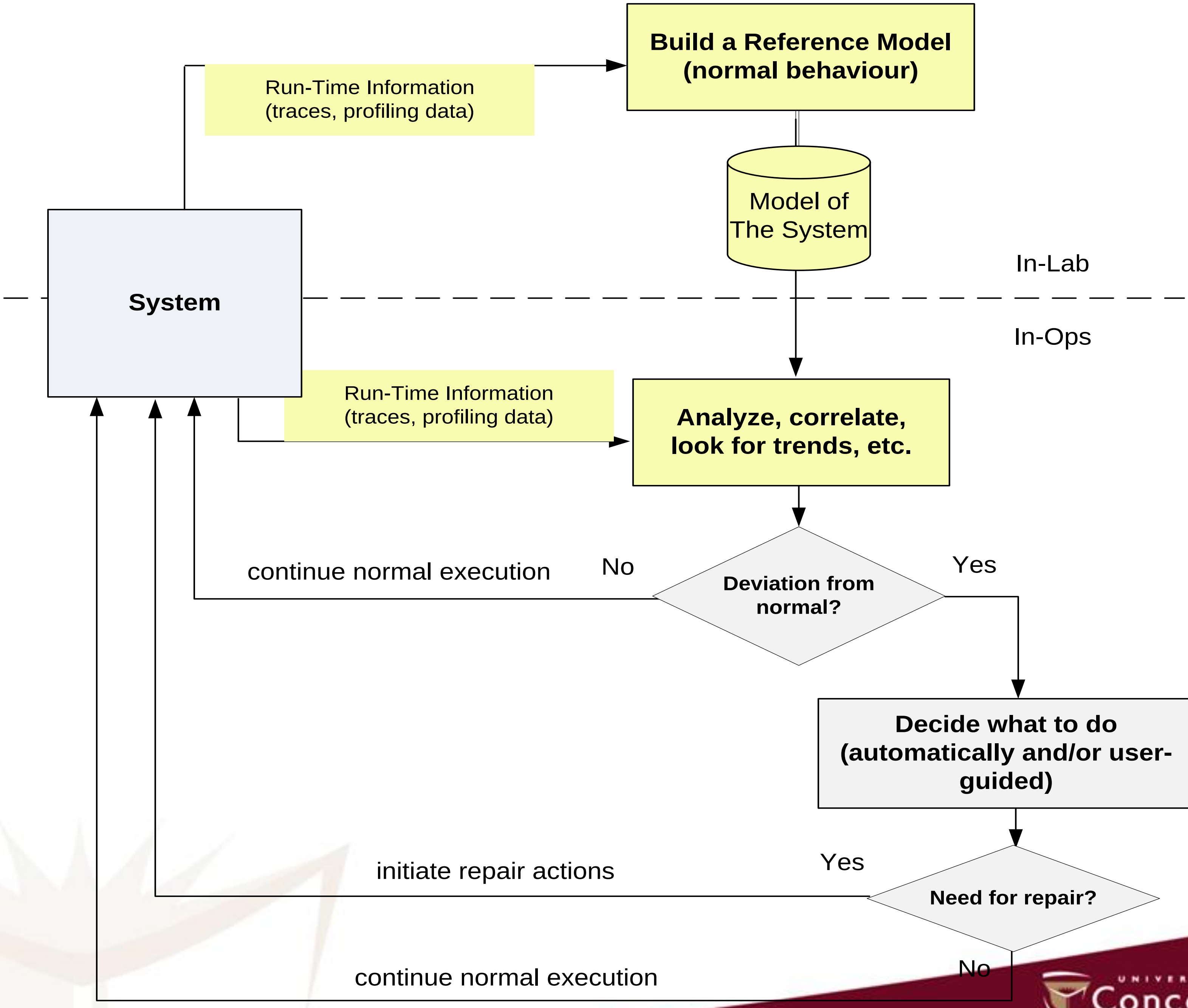
- Protect **host systems against cyber-attacks** (web-based exploitation, simulated social engineering, etc.)
- Model **system health** and develop **modular, adaptive,** and **scalable** Anomaly Detection Systems (ADS) at the **system call level**
- Reduce **false positives (alarms)** and improve the **true positives**
- Provide preliminary analysis/recommendations for future research and directions

Université Concordia University

Background on ADS

- Monitors computer or network activity for signs of intrusions and alert administrators
- Signature based Detection
  - Looks for known patterns
  - Detects only known attacks
- Anomaly Detection
  - Looks for deviations from normal behavior
  - Detects even unknown attacks (zero day exploits)

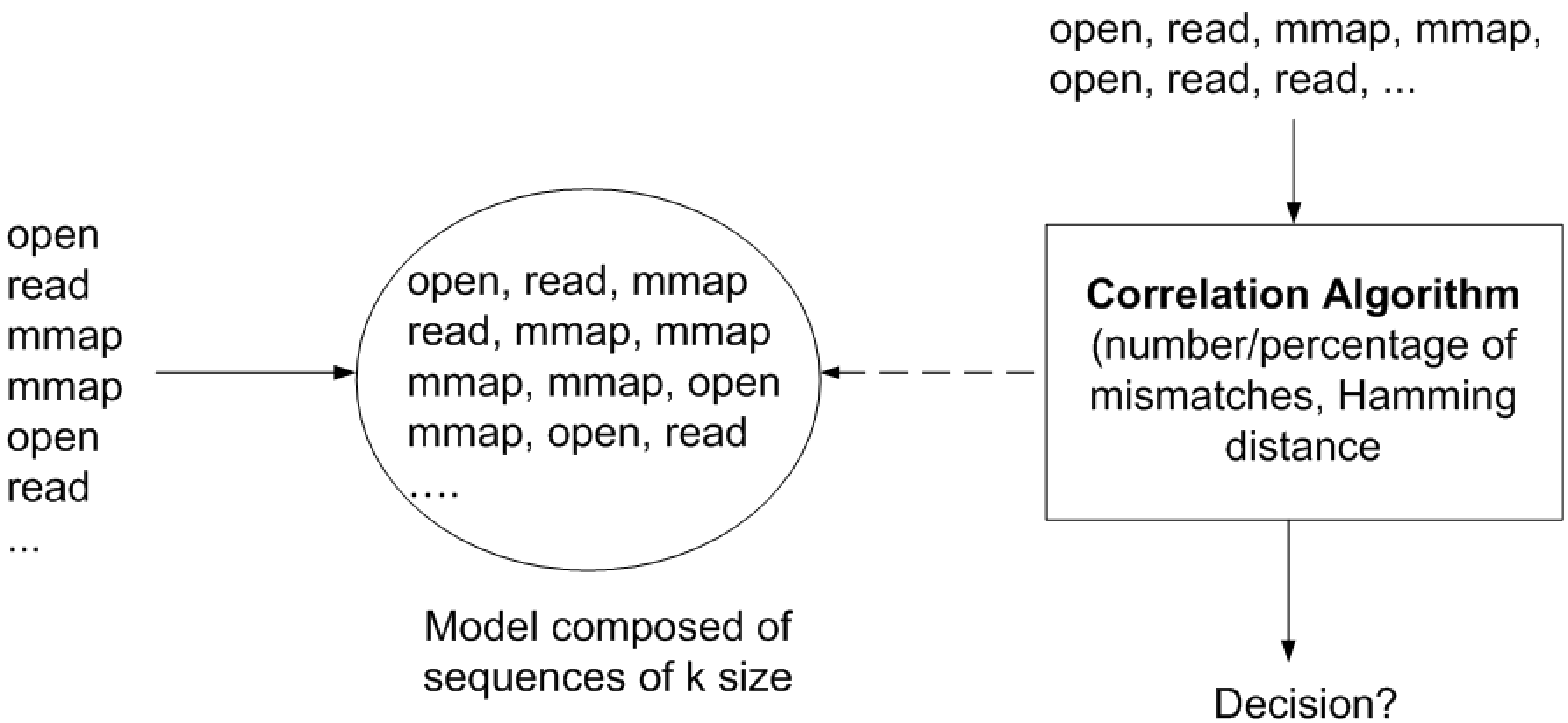

Build a Reference Model
(normal behaviour)
Run-Time Information
(traces, profiling data)
Model of
The System
In-Lab
System
In-Ops
Run-Time Information
(traces, profiling data)
Analyze, correlate,
look for trends, etc.
continue normal execution
No
Deviation from
normal?
Yes
Decide what to do
(automatically and/or user-
guided)
Yes
initiate repair actions
Need for repair?
No
continue normal execution


Build a Reference Model
(normal behaviour)
Run-Time Information
(traces, profiling data)
Model of
The System
In-Lab
System
In-Ops
Run-Time Information
(traces, profiling data)
Analyze, correlate,
look for trends, etc.
continue normal execution
No
Yes
Deviation from
normal?
Decide what to do
(automatically and/or user-
guided)
Yes
initiate repair actions
Need for repair?
No
continue normal execution

Existing Work

- Several techniques have been used to model the normal behavior of a system
  - Sliding window technique
  - HMM
  - Neural networks (two-class)
  - Clustering
  - Varied length n-gram technique
  - Context Free Grammar

Example: Sliding Approach (STIDE)



Challenges – False alarms

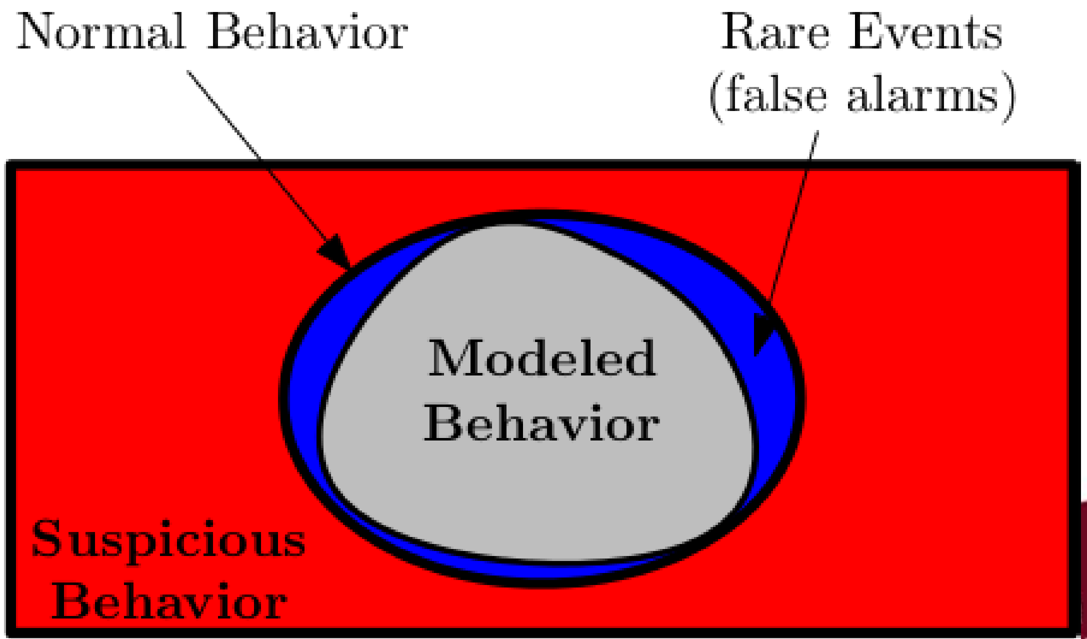
- High false alarms **reduce confidence** and could lead to deactivation of the ADS
- Causes:
  - Unrepresentative normal data for training and attack data for validation and testing
  - Inappropriate model or feature selection
  - Poor optimization of models parameters
  - Over fitting (leads to poor generalization)
  - Inadequate assumptions such as static environments

Université Concordia University

Challenges: Adaptability

- ADSs are often designed using limited data
  - collection and analysis of representative data from each process (different versions, OS, etc.) is costly

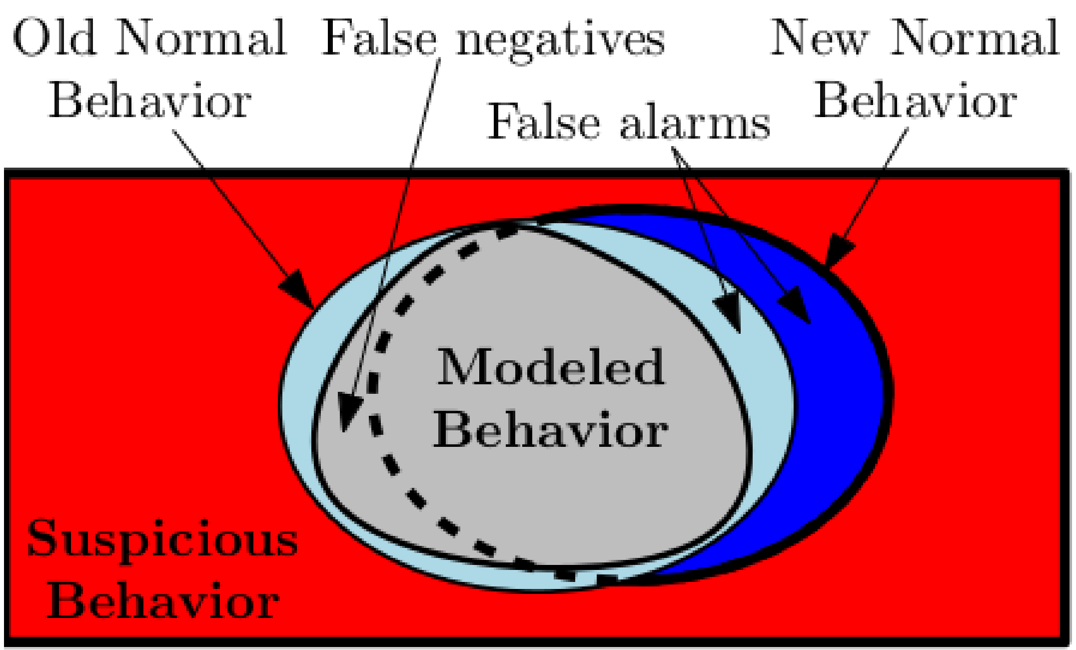
Anomaly detector will have **incomplete** view of normal system behavior



In Practice

- Dynamic environment
  - Changes in normal process behavior due, for instance, to application update

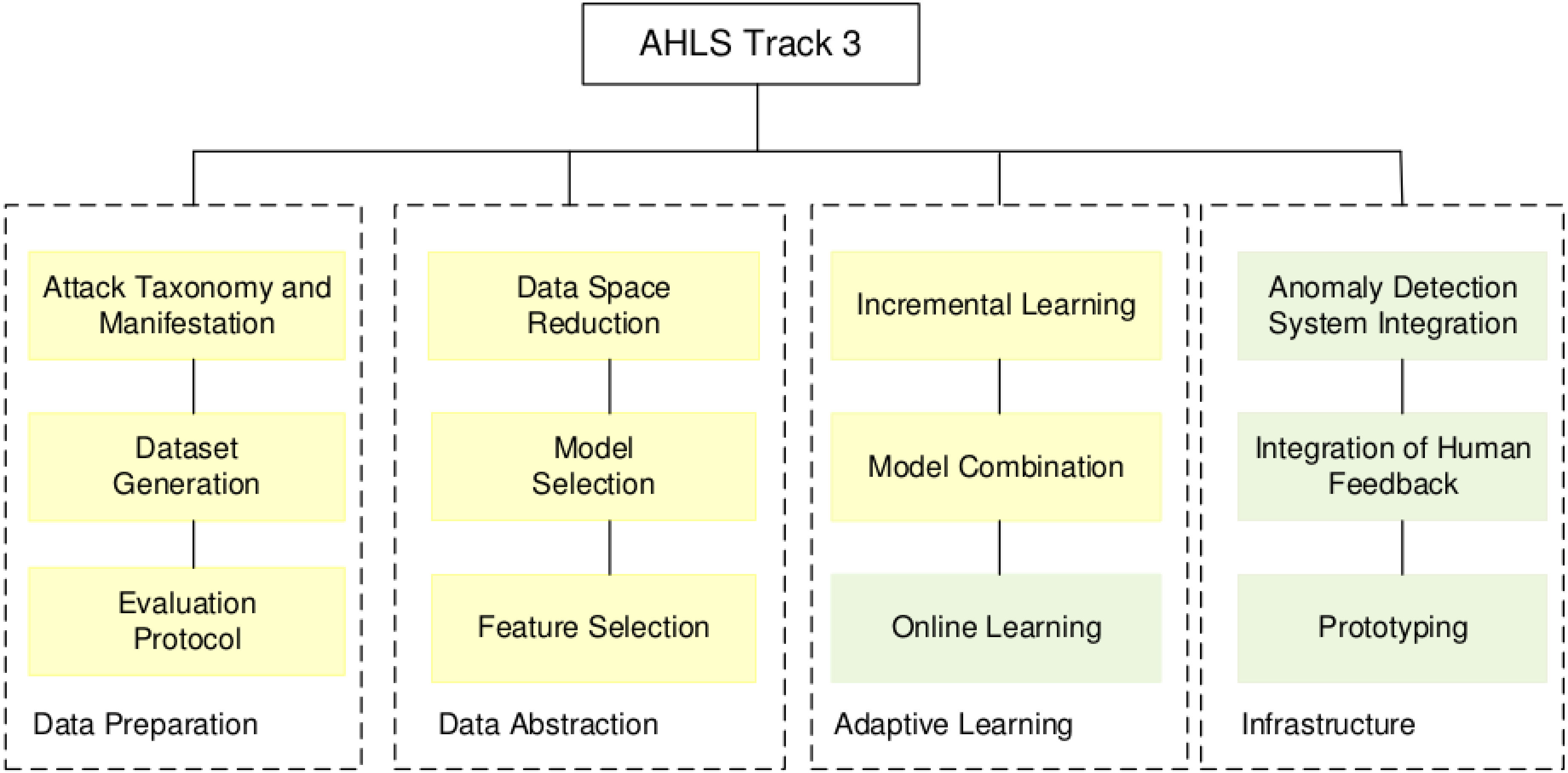
Internal model of normal behavior **diverges** with respect to the underlying data



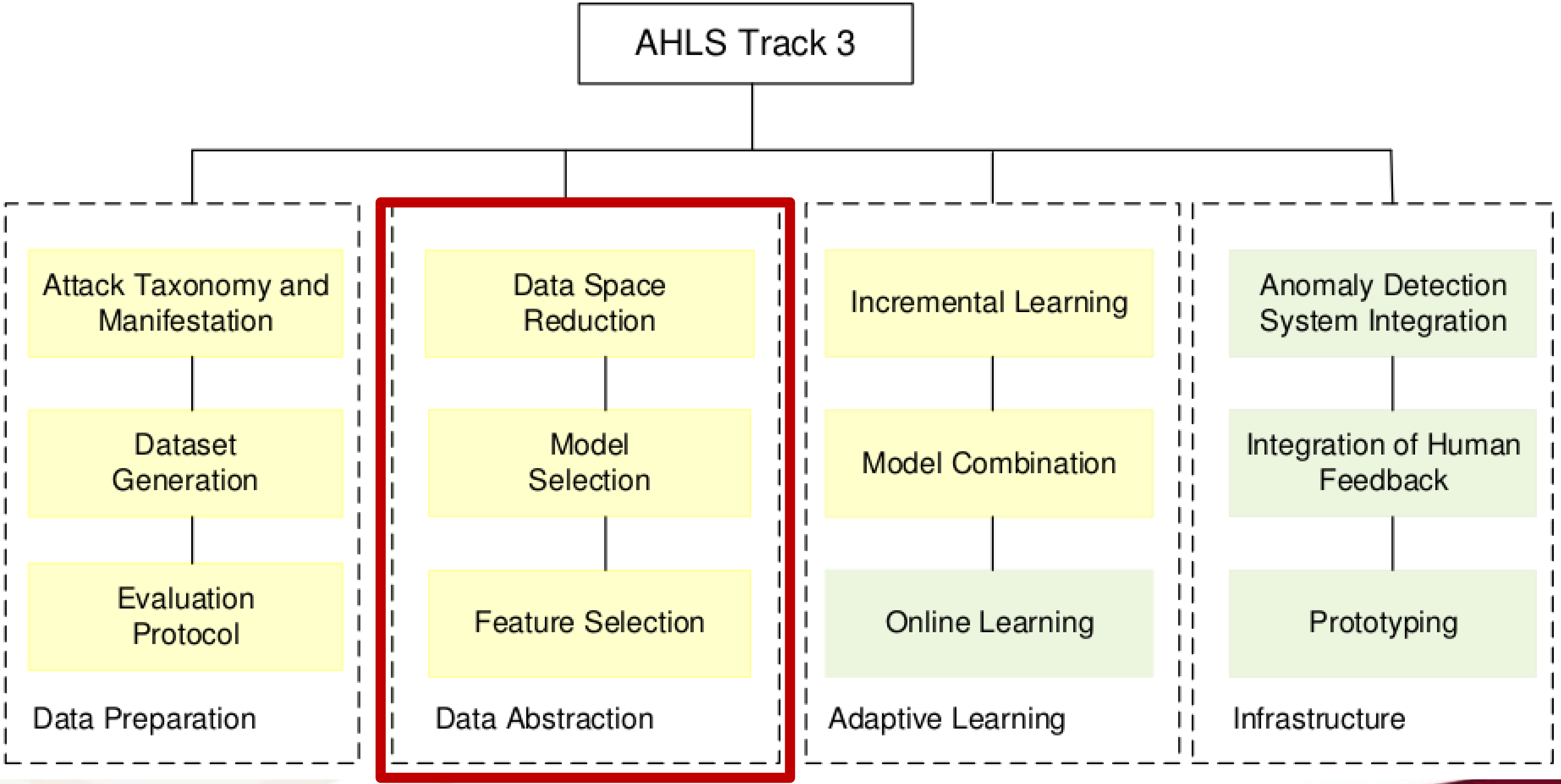
ADS Requirements

- ADS should:
  - Account for rare normal events (false alarms)
  - Be scalable and modular: can add, replace or remove models or features over time
  - Handle large data spaces
  - Accommodate new data

Advanced Host-Level Surveillance



Advanced Host-Level Surveillance



Kernel State Modeling (KSM)

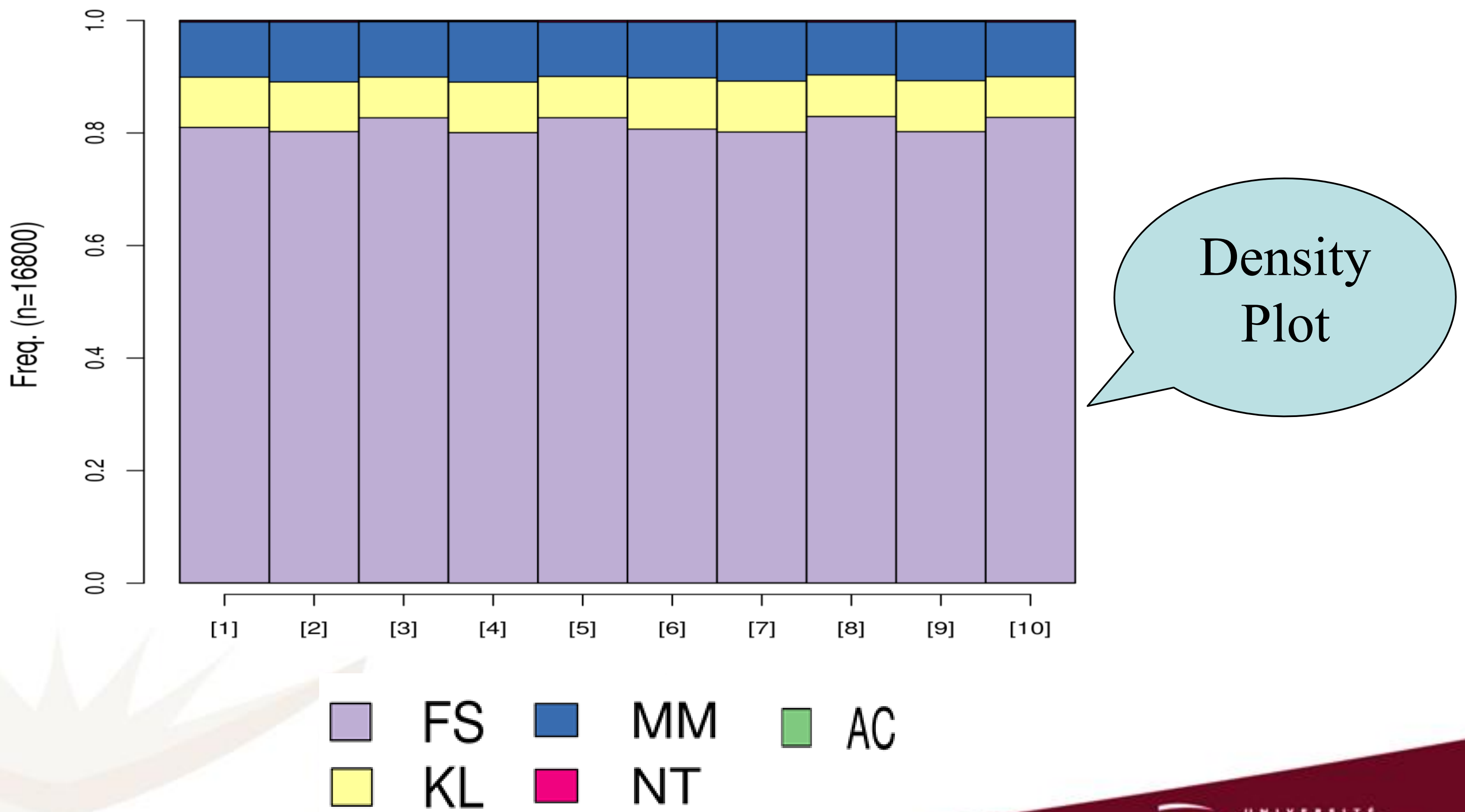
- KSM is an anomaly detection technique
  - Transforms system calls into kernel modules, called states
  - Detect anomalies at the level of interaction of kernel states
  - Reduces data space used in training and testing
  - Favors efficiency while keeping accuracy

Transforming System Calls into States of Kernel Modules

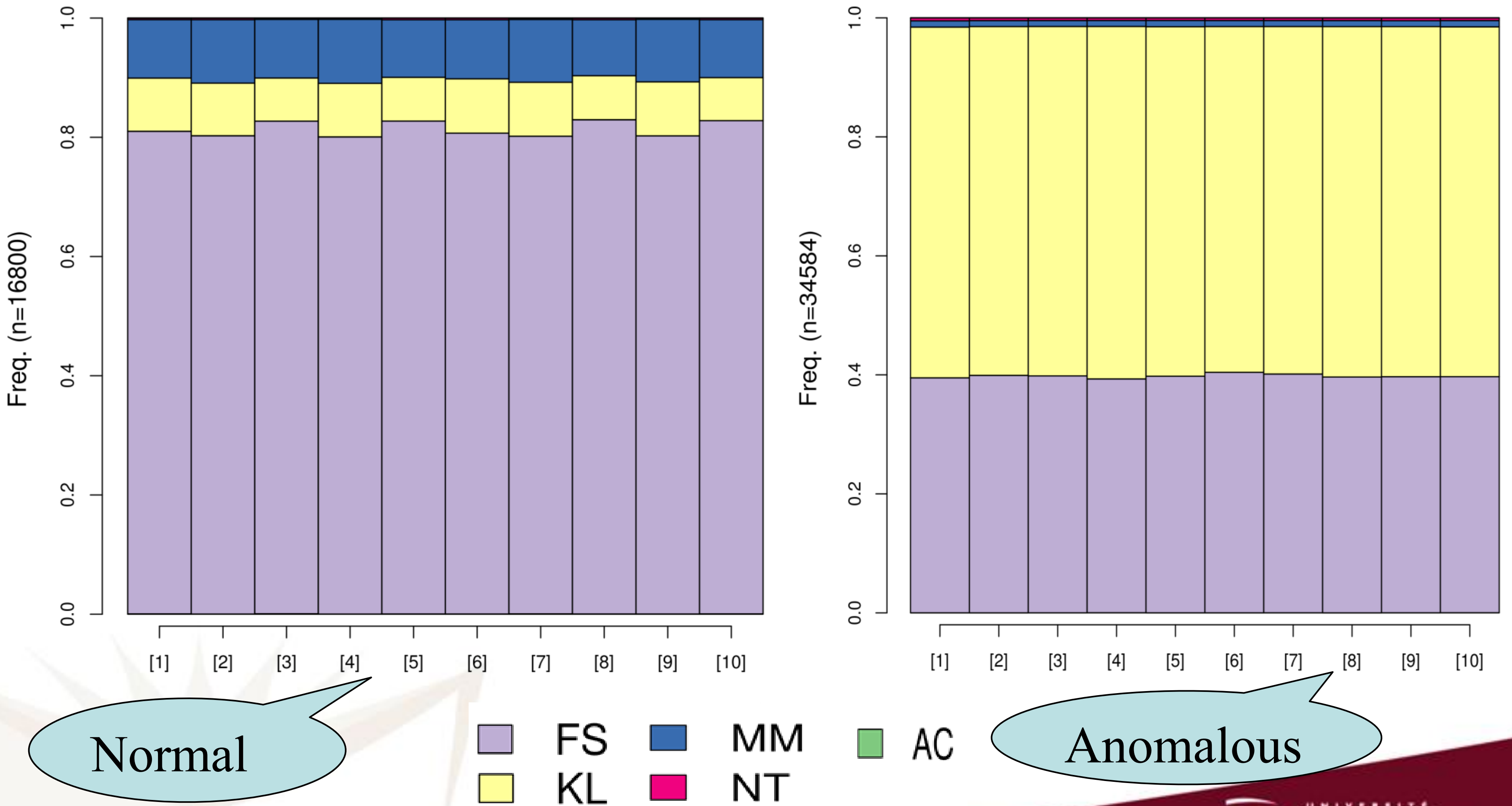
State	Module in Linux Source Code	# of System Calls
AC	Architecture	10
FS	File System	131
IPC	Inter Process Communication	7
KL	Kernel	127
MM	Memory Management	21
NT	Networking	2
SC	Security	3
UN	Unknown	37

[Source]: http://syscalls.kernelgork.com

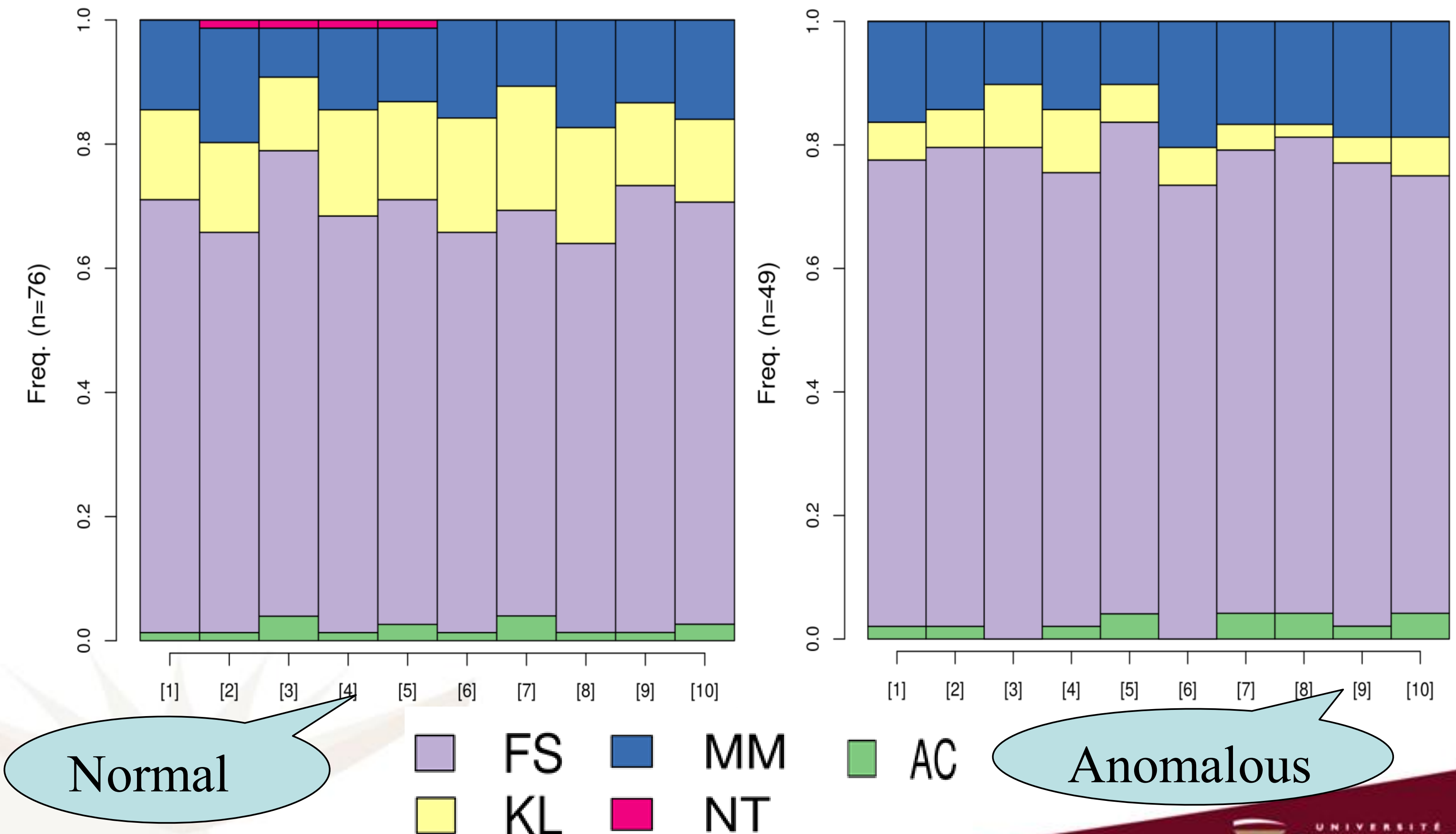
KSM and Density Plots



Anomaly Detection in Firefox



Anomaly Detection in Login Utility



Automatically Detecting Anomalies

- To determine significant deviation **threshold (alpha):**
  - Divide normal dataset into training set, validation set, and testing set
  - Extract **probabilities** from training set
  - Evaluate on validation **set and adjust alpha**
  - Measure accuracy on testing set

Case Study 1: ADFA Linux Dataset

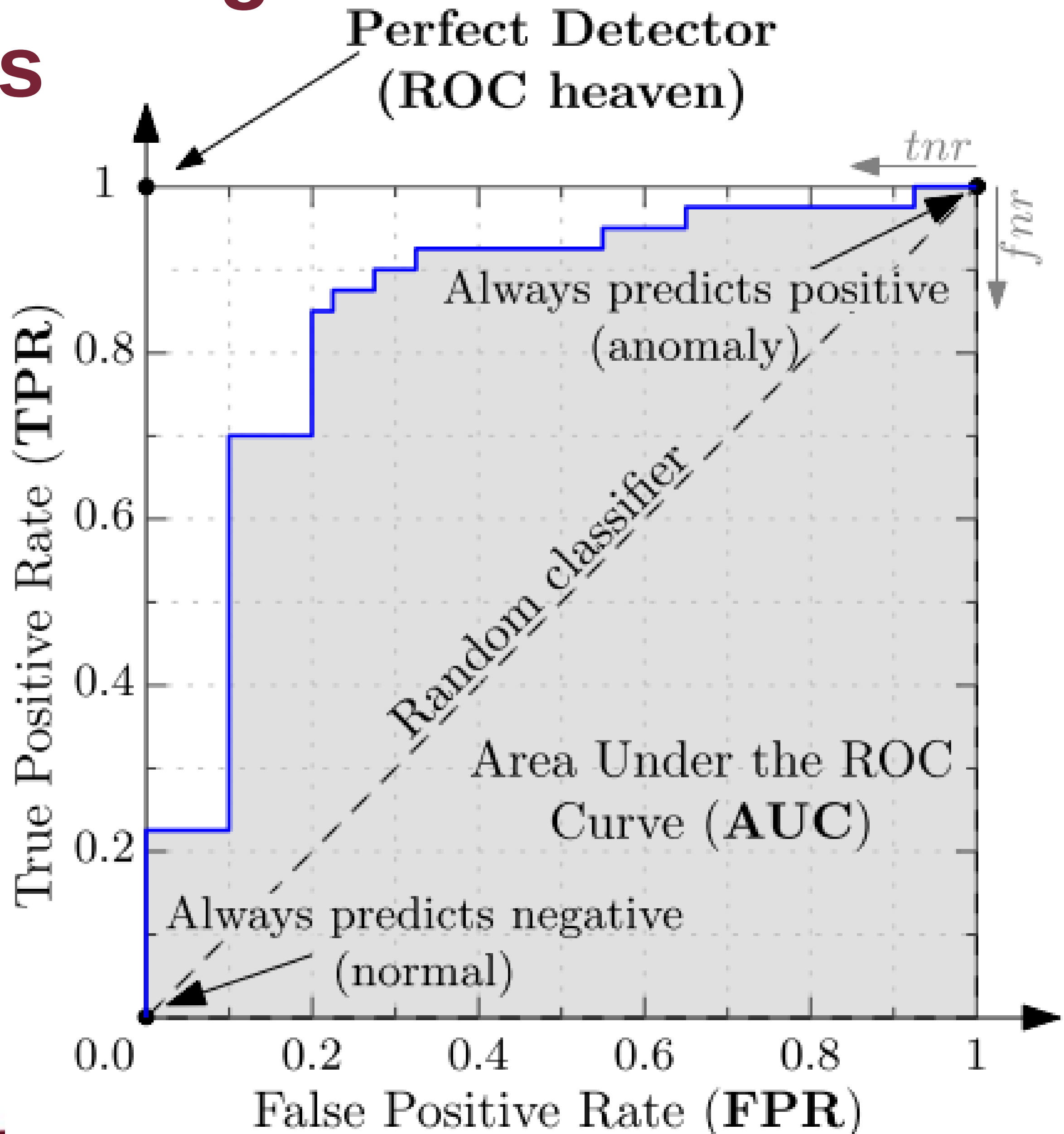
- A host with Ubuntu 11.04, Apache 2.2.17, PHP 5.3.5, TikiWiki 8.1, FTP server, MySQL 14.14 and an SSH server
  - web-based exploitation
  - simulated social engineering
  - poisoned executable,
  - remotely triggered vulnerabilities,
  - remote password brute force attacks
  - system manipulation

Case Study 1: ADFA Linux Dataset

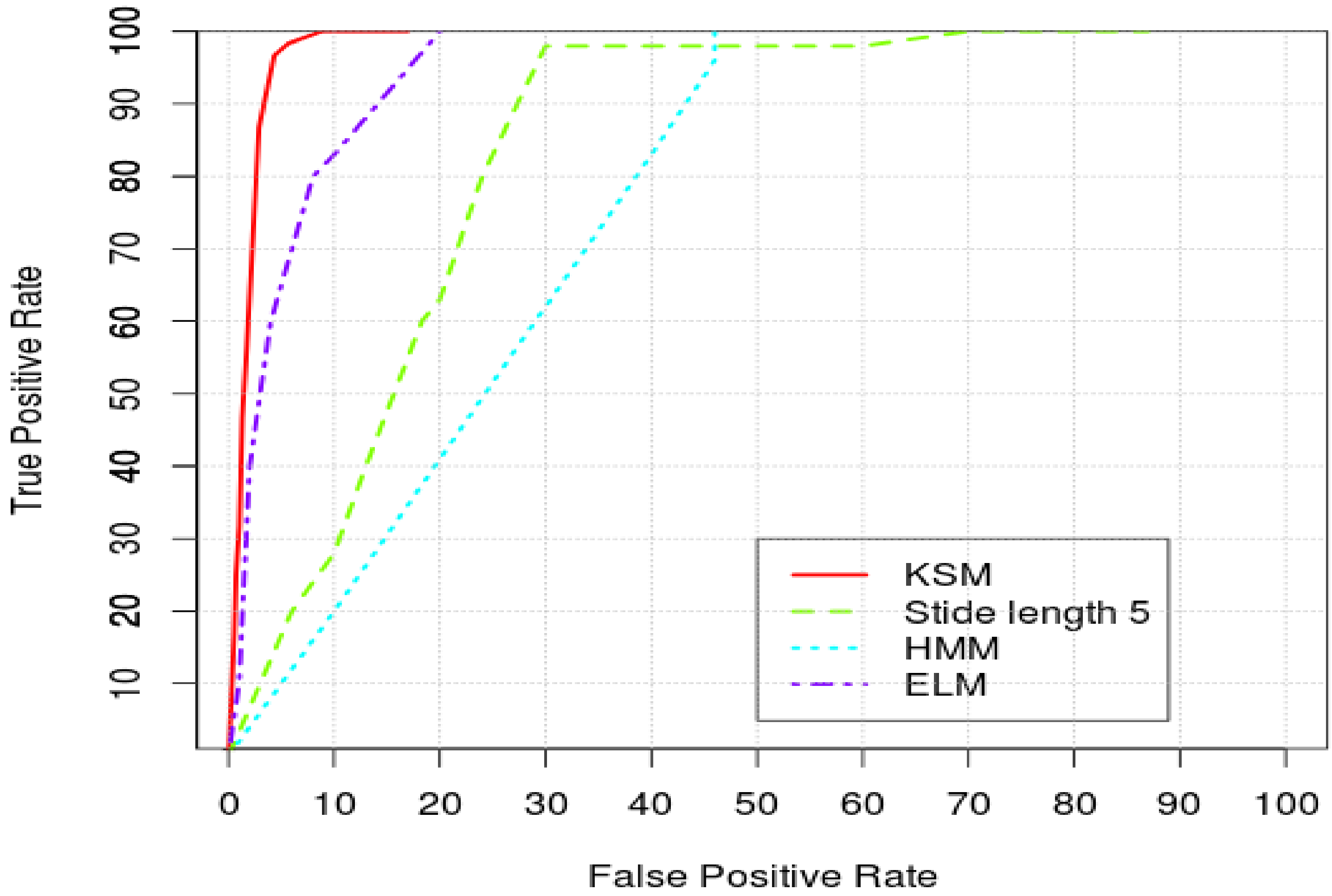
Training Set	
# of training traces	**833**
Validation Set	
# of attacks	**20**
# of normal traces	**1000**
Testing Set	
# of attacks	**40**
# of normal traces	**3373**

Receiver Operating Characteristics (ROC) Curves

- True Positive: anomaly detected as anomaly
- False Positive: normal detected as anomaly



Case Study 1: ADFA Linux Dataset



Case Study 2: Dataset

Program	# Normal Traces			#Attack Types	#Attack Traces
	Training	Validation	Testing		
Login	4	3	5	1	4
PS	10	4	10	1	15
Stide	400	200	13126	1	105
Xlock	91	30	1610	1	2
Firefox	125	75	500	5	19

Case Study 2: Results

Program	Technique	TP rate	FP rate
Login	KSM (alpha=0.00)	100%	0.00%
	Stide (win=6)	100%	40.00%
	Stide (win=10)	100%	40.00%
	HMM (states=10)	100%	40.00%
PS	KSM (alpha=0.02)	100%	10.00%
	Stide (win=6)	100%	10.00%
	Stide (win=10)	100%	10.00%
	HMM (states=5)	100%	30.00%
Xlock	KSM (alpha=0.04)	100%	0.00%
	Stide (win=6)	100%	1.50%
	Stide (win=10)	100%	1.50%
	HMM (states=5)	100%	0.00%

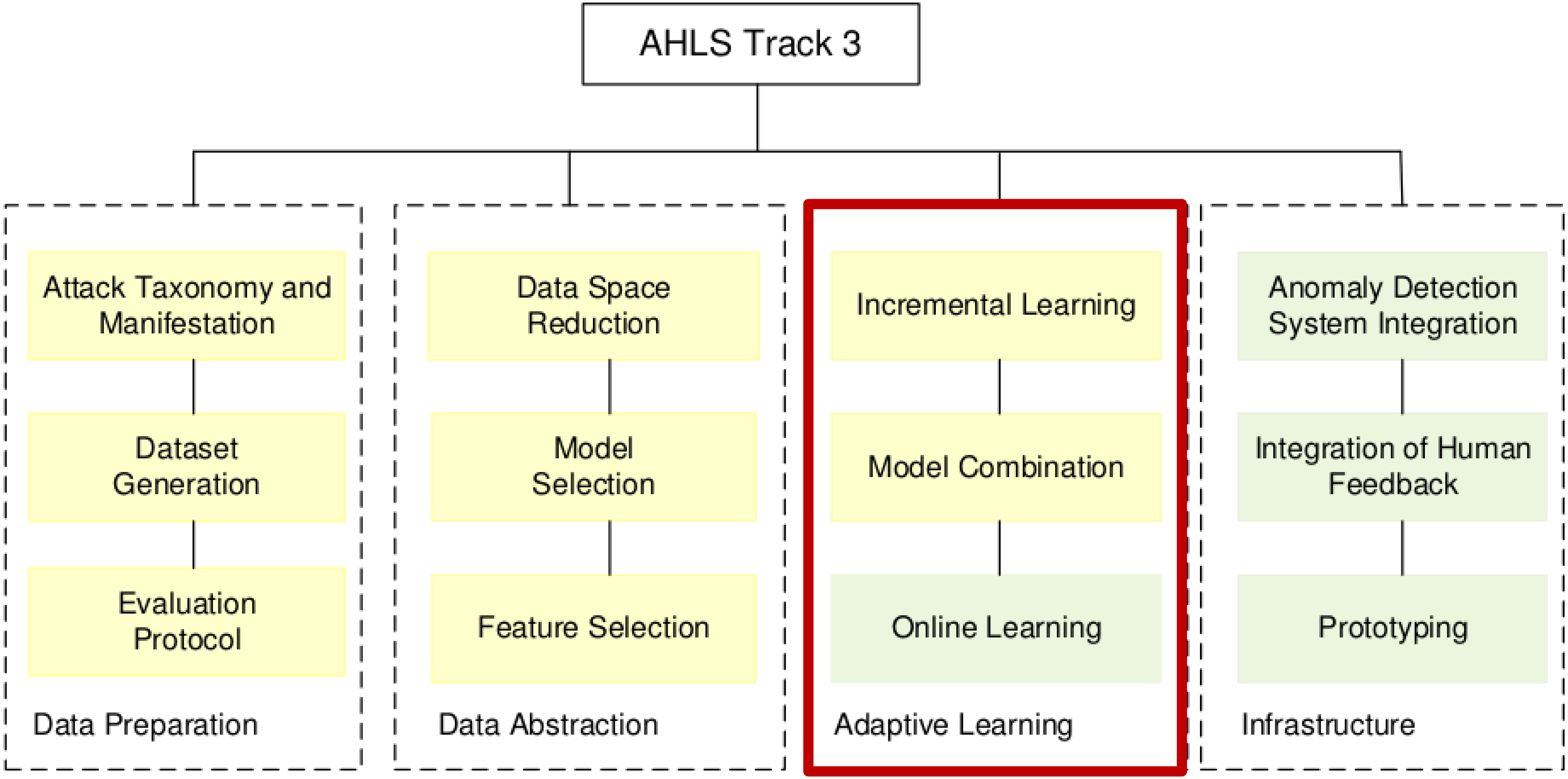
Case Study 2: Results

Program	Technique	TP rate	FP rate
Stide	KSM (alpha=0.06)	100%	0.25%
	Stide (win=6)	100%	4.97%
	Stide (win=10)	100%	5.25%
	HMM (states=5)	100%	0.25%
Firefox	KSM (alpha=0.08)	100%	0.60%
	Stide (win=6)	100%	44.60%
	Stide (win=10)	100%	49.20%
	HMM (states=5)	100%	1.40%

Case Study 2: Execution Time

	Size of All Traces	KSM	Stide	HMM
Login	26.2KB	4.46 sec	0.03 sec	56.43 min
PS	29.6KB	5.14 sec	0.11 sec	46.24 min
Xlock	47.4MB	1.51 min	12.3 min	13.37 hr
Stide	36.2MB	5.85 min	8.53 min	2.3 day
Firefox	270.6MB	9.35 min	4.17 hr	4.03 day

Research Threads



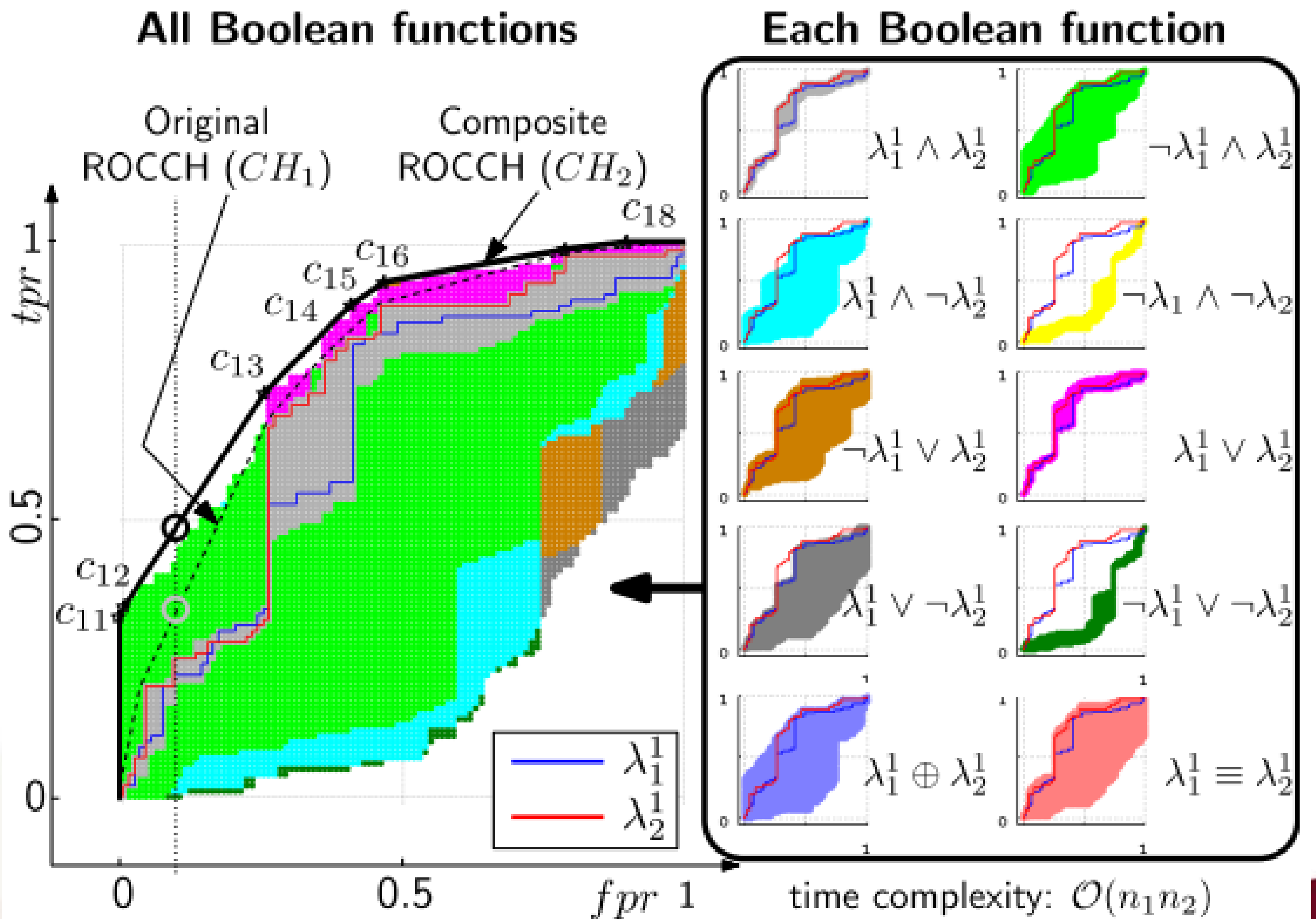
Model Combination

- A single classifier or model may not provide a good approximation to the underlying data structure or distribution
  - No dominant classifier for all data distributions (“no free lunch” theorem)
  - True data distribution is usually unknown
  - Limited amount of (labeled) data is typically provided during training

IBC: Iterative Boolean Combination in the ROC Space

- For each threshold from the first detector and each threshold from the second detector:
  - Combine the responses using all Boolean functions
  - Select thresholds and Boolean functions that improve the ROC space

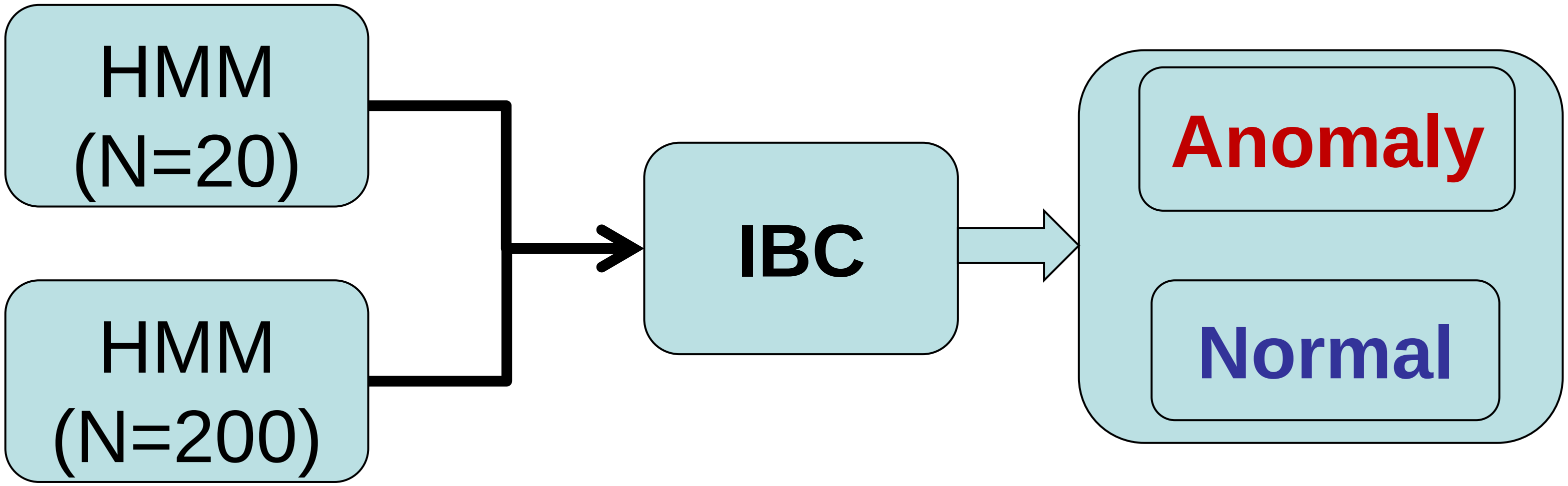
IBC - Example

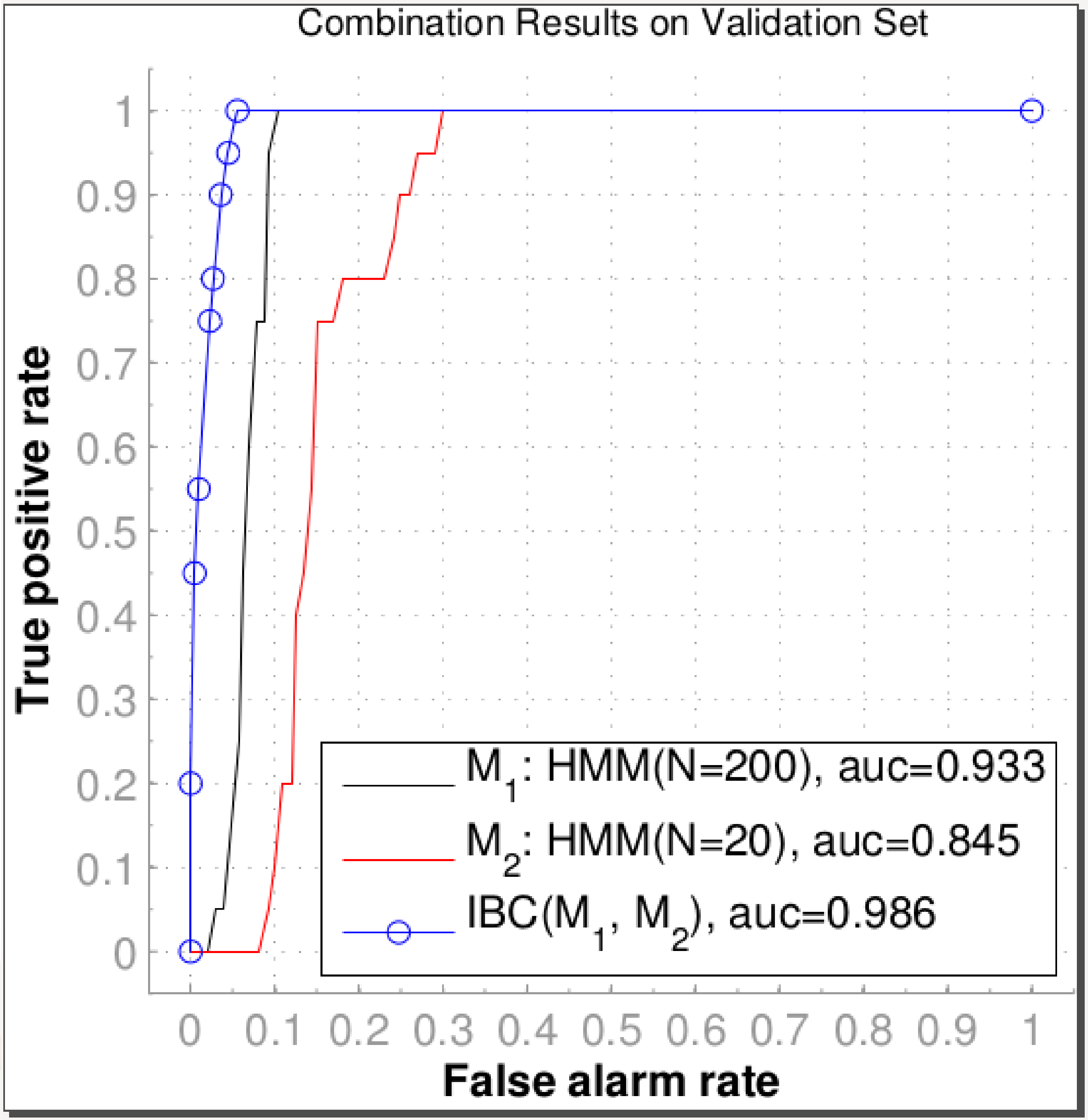

All Boolean functions
Each Boolean function
Original ROCCH (CH_1)
Composite ROCCH (CH_2)
c_{11} c_{12} c_{13} c_{14} c_{15} c_{16} c_{18}
tpr
fpr
λ_1^1
λ_2^1
$\lambda_1^1 \wedge \lambda_2^1$
$\neg\lambda_1^1 \wedge \lambda_2^1$
$\lambda_1^1 \wedge \neg\lambda_2^1$
$\neg\lambda_1 \wedge \neg\lambda_2$
$\neg\lambda_1^1 \vee \lambda_2^1$
$\lambda_1^1 \vee \lambda_2^1$
$\lambda_1^1 \vee \neg\lambda_2^1$
$\neg\lambda_1^1 \vee \neg\lambda_2^1$
$\lambda_1^1 \oplus \lambda_2^1$
$\lambda_1^1 \equiv \lambda_2^1$
time complexity: $\mathcal{O}(n_1 n_2)$

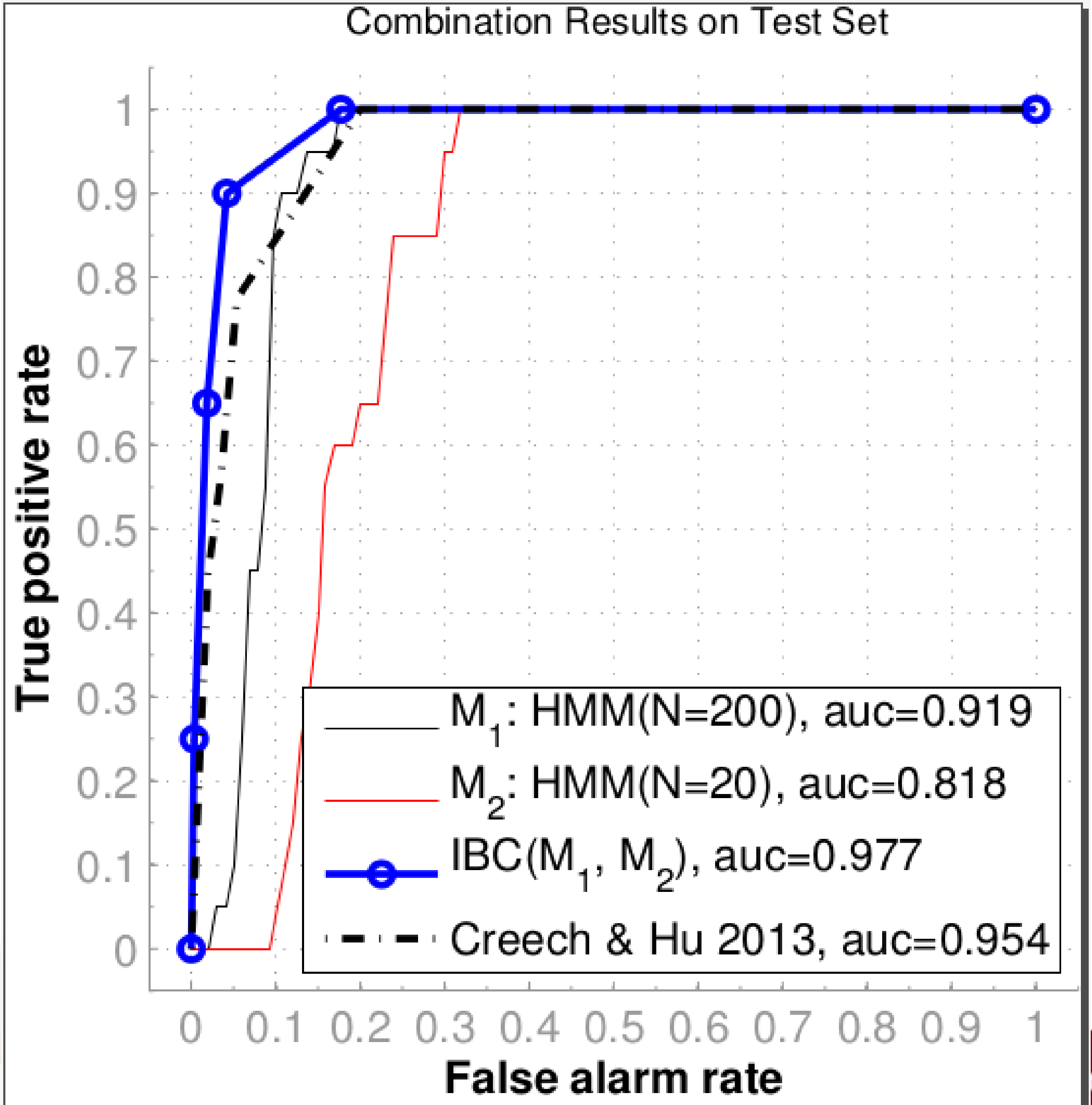
Experimental Methodology

Training Set	
# of training traces	**833**
Validation Set	
# of attacks	**20**
# of normal traces	**1000**
Testing Set	
# of attacks	**40**
# of normal traces	**3373**

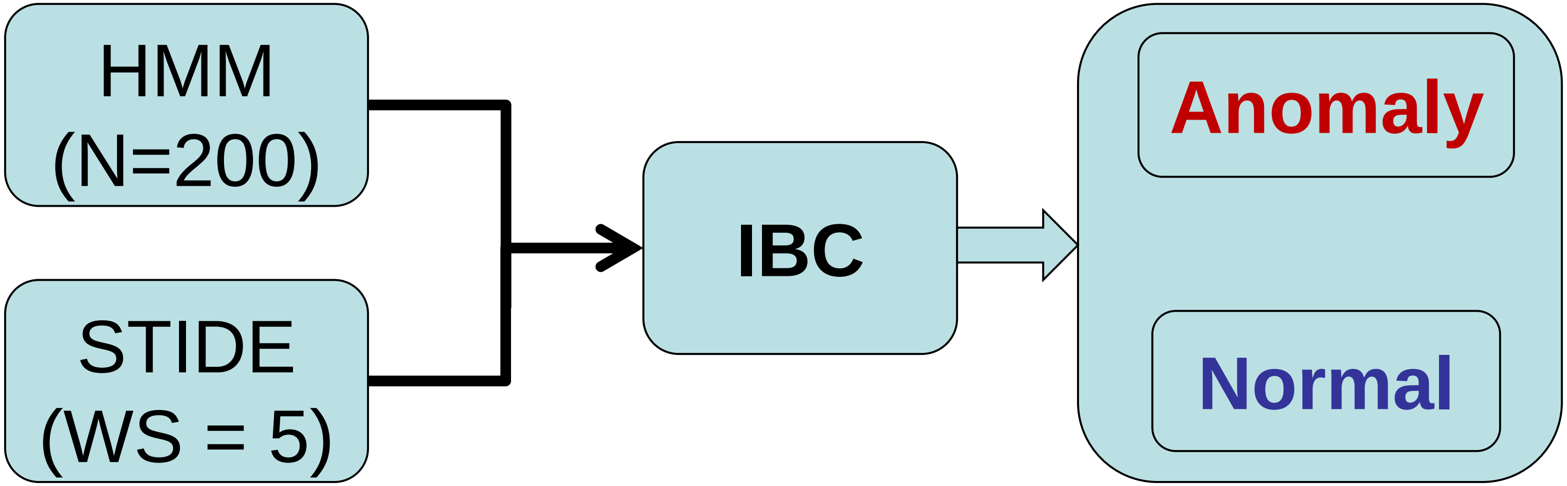
Combination of Responses from Different HMMs

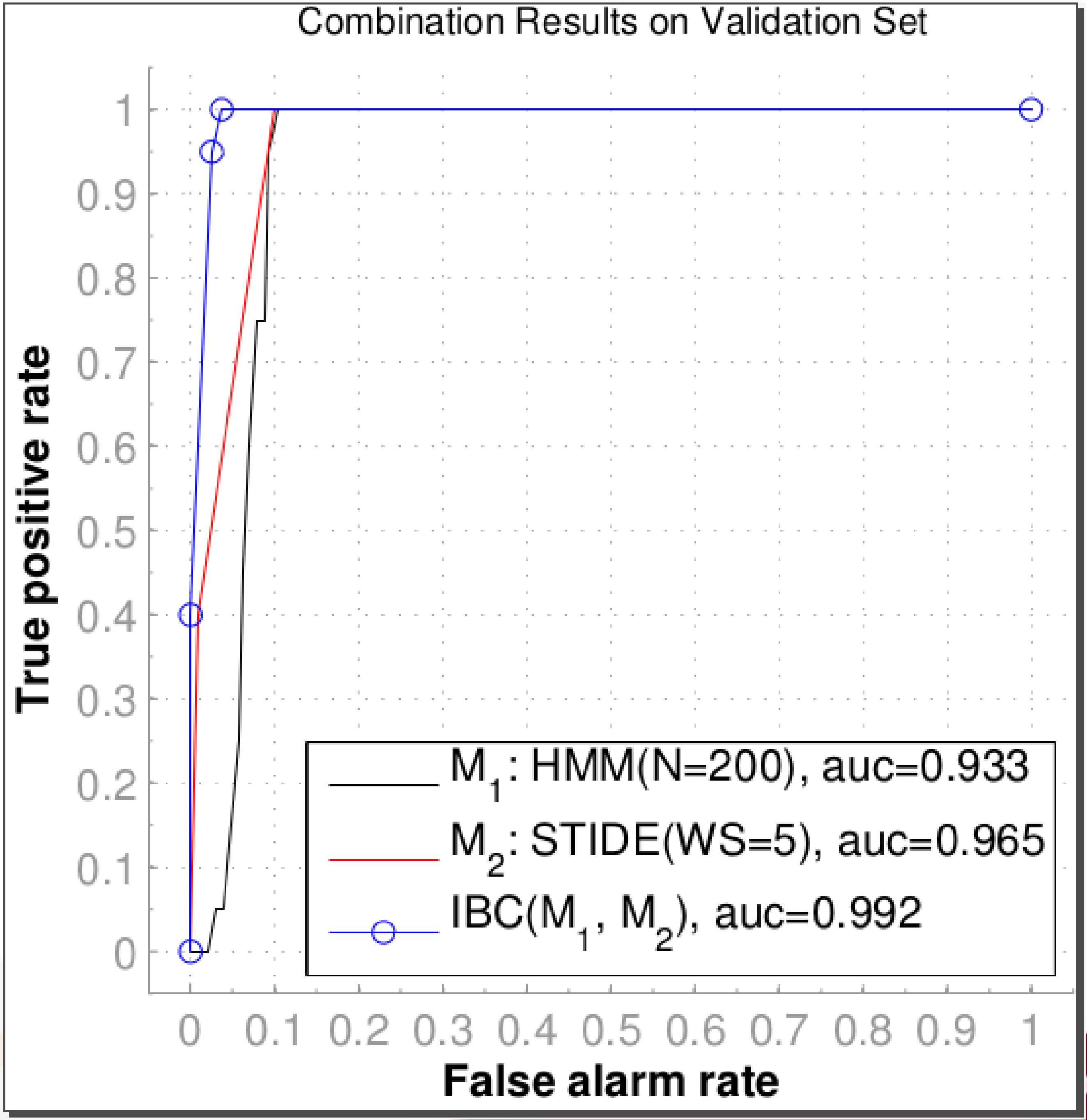


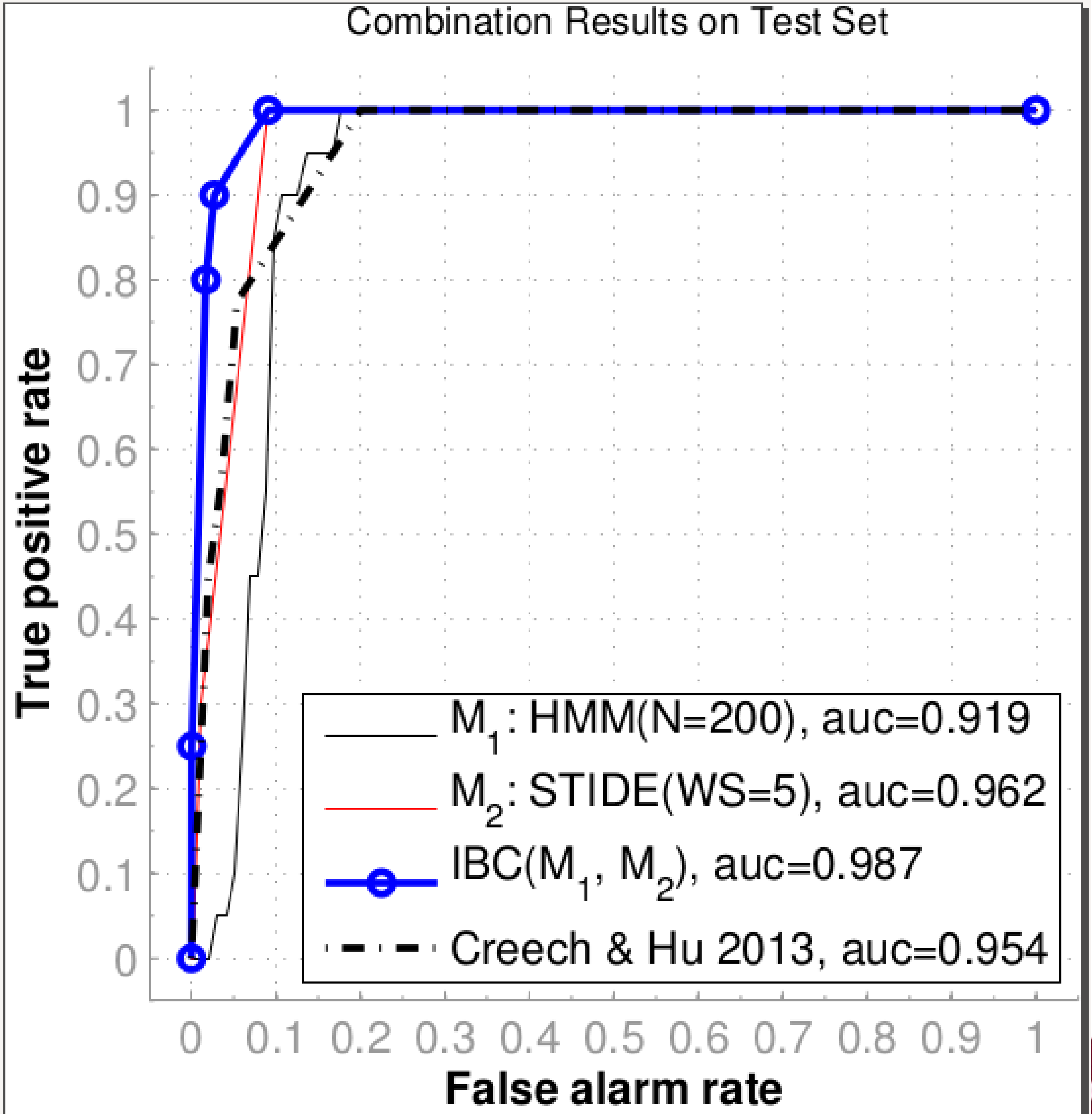

Combination Results on Validation Set
True positive rate
False alarm rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
M1: HMM(N=200), auc=0.933
M2: HMM(N=20), auc=0.845
IBC(M1, M2), auc=0.986


Combination Results on Test Set
True positive rate
False alarm rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
M_1: HMM(N=200), auc=0.919
M_2: HMM(N=20), auc=0.818
IBC(M_1, M_2), auc=0.977
Creech & Hu 2013, auc=0.954

Combination of HMM and STIDE Responses

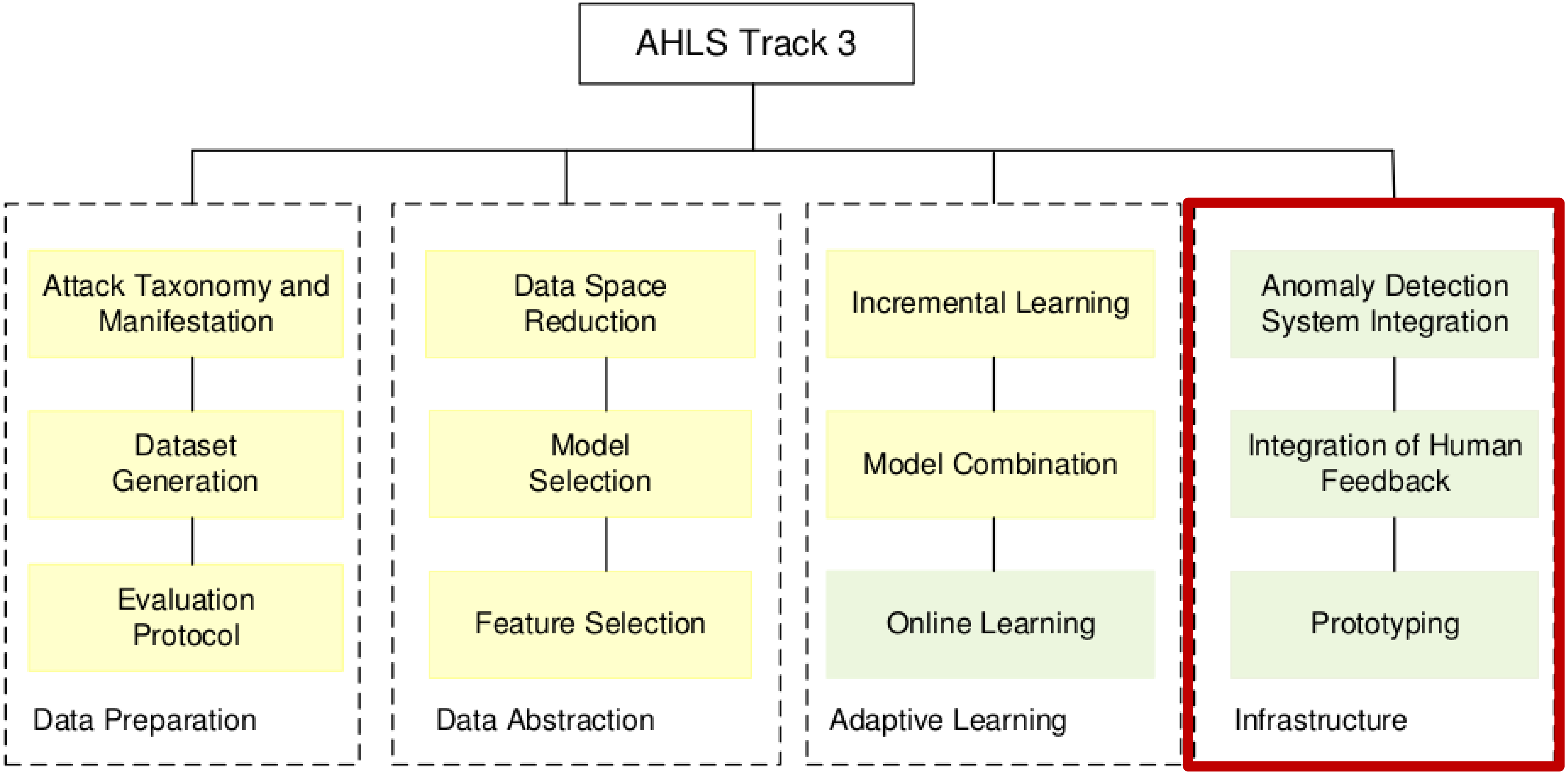



Combination Results on Validation Set
True positive rate
False alarm rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
M1: HMM(N=200), auc=0.933
M2: STIDE(WS=5), auc=0.965
IBC(M1, M2), auc=0.992



Université Concordia University

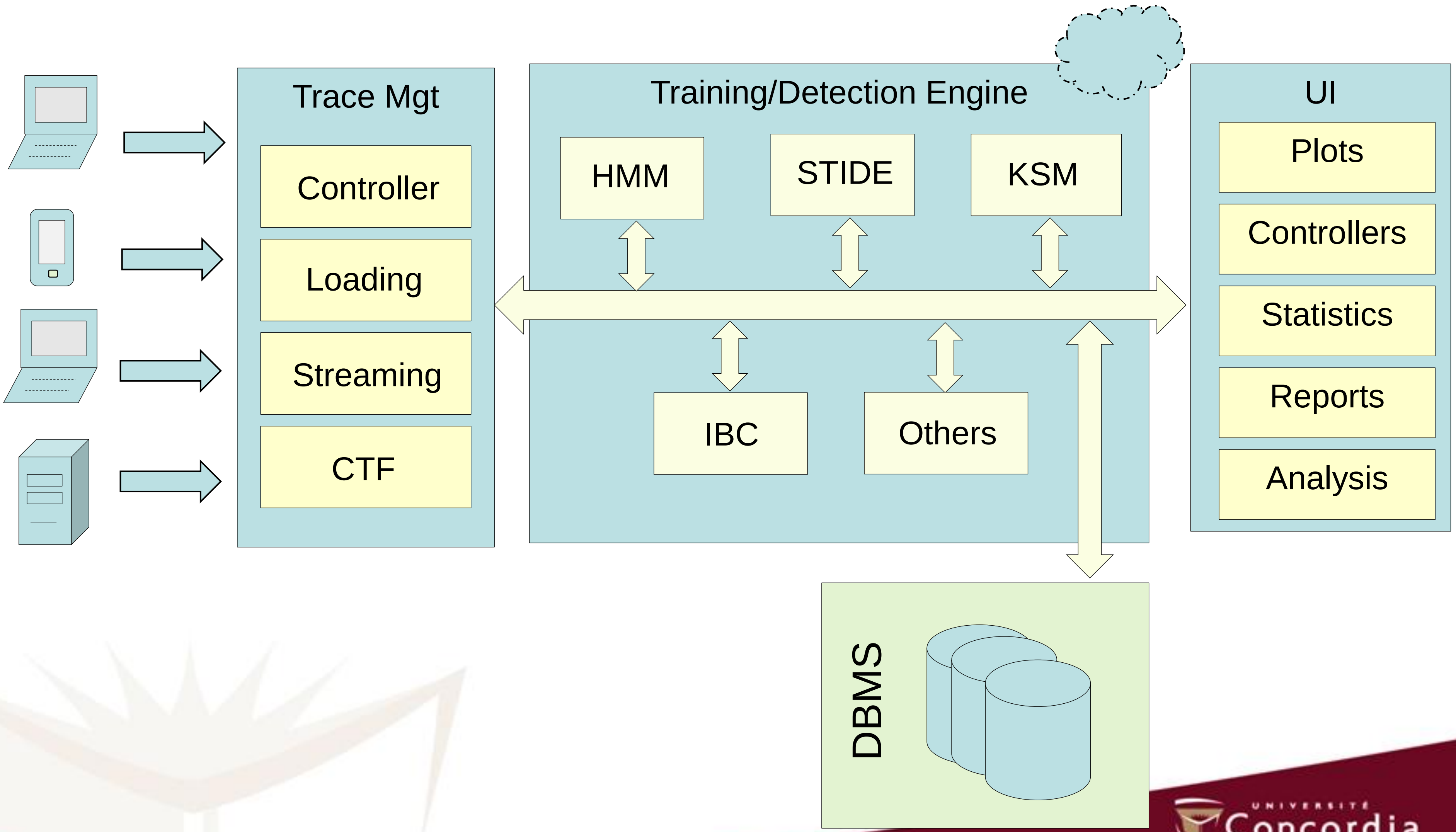
Research Threads

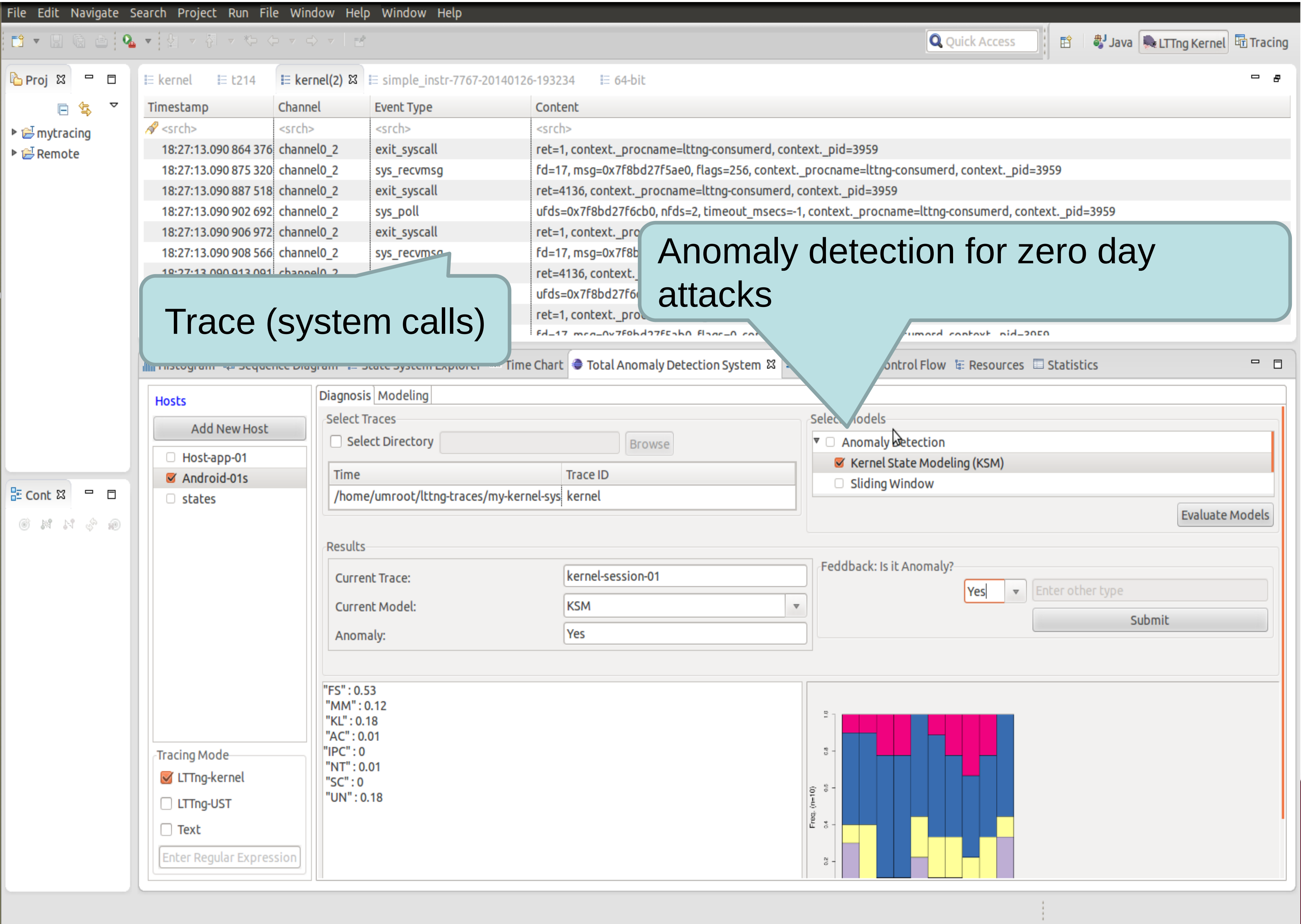


TotalADS

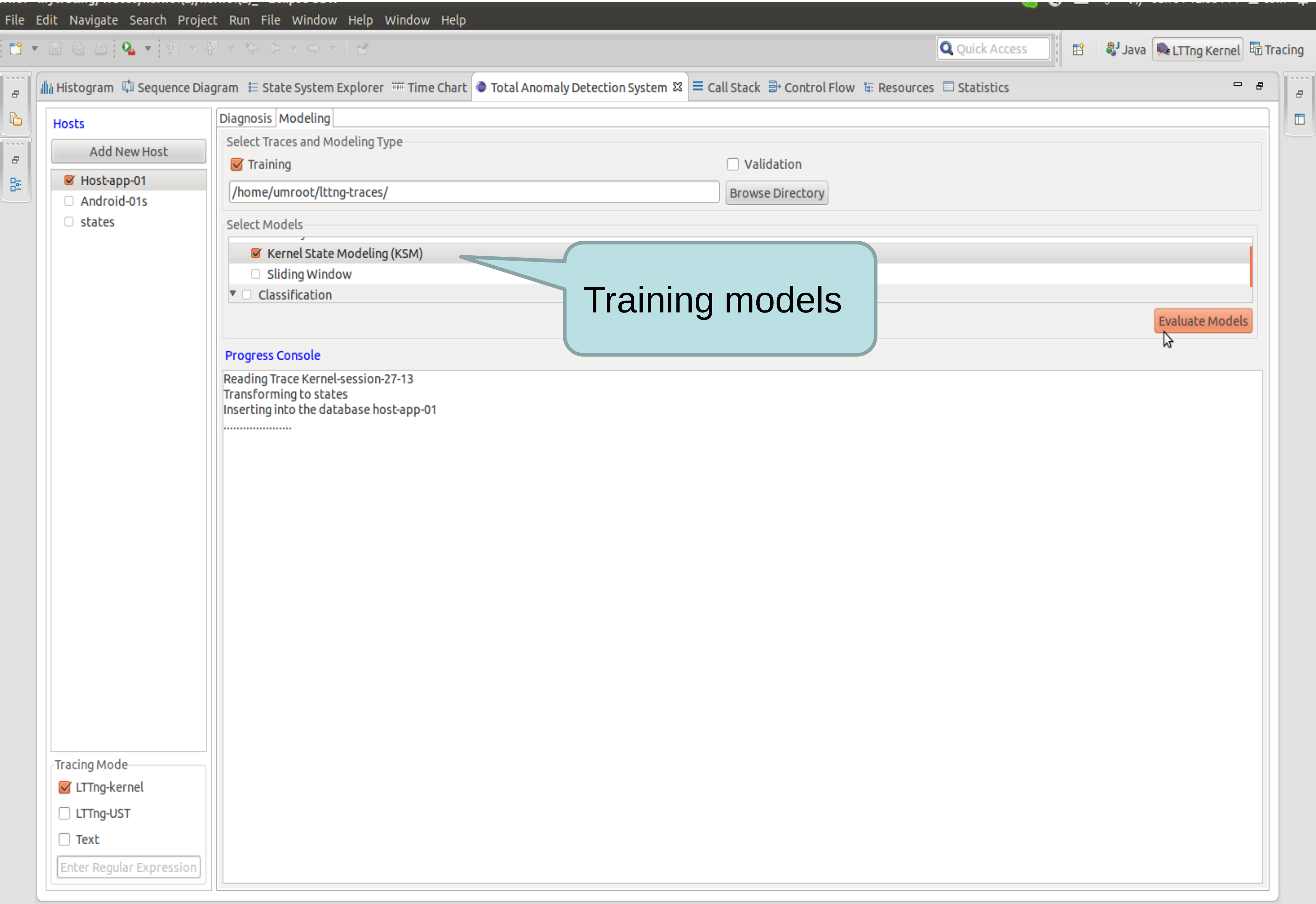
- TotalADS is an integrated Anomaly Detection System Environment
  - Eclipse Plug-in
  - Open Source
  - Based on TMF (Tracing and Monitoring Framework)
  - Supports STIDE, HMM, KSM, IBC
  - Supports a combination of classifiers
  - Supports trace analysis and forensic analysis
  - Supports CTF (Common Trace Format)

Architecture





Trace (system calls)
Anomaly detection for zero day attacks
File Edit Navigate Search Project Run File Window Help Window Help
Quick Access
Java
LTTng Kernel
Tracing
kernel
t214
kernel(2)
simple_instr-7767-20140126-193234
64-bit
Timestamp
Channel
Event Type
Content
18:27:13.090 864 376 channel0_2 exit_syscall ret=1, context._procname=lttng-consumerd, context._pid=3959
18:27:13.090 875 320 channel0_2 sys_recvmsg fd=17, msg=0x7f8bd27f5ae0, flags=256, context._procname=lttng-consumerd, context._pid=3959
18:27:13.090 887 518 channel0_2 exit_syscall ret=4136, context._procname=lttng-consumerd, context._pid=3959
18:27:13.090 902 692 channel0_2 sys_poll ufds=0x7f8bd27f6cb0, nfds=2, timeout_msecs=-1, context._procname=lttng-consumerd, context._pid=3959
18:27:13.090 906 972 channel0_2 exit_syscall
18:27:13.090 908 566 channel0_2
mytracing
Remote
Time Chart
Total Anomaly Detection System
Resources
Statistics
Hosts
Add New Host
Host-app-01
Android-01s
states
Diagnosis
Modeling
Select Traces
Select Directory
Browse
Time
Trace ID
/home/umroot/lttng-traces/my-kernel-sys
kernel
Anomaly Detection
Kernel State Modeling (KSM)
Sliding Window
Evaluate Models
Results
Current Trace:
kernel-session-01
Current Model:
KSM
Anomaly:
Yes
Feddback: Is it Anomaly?
Yes
Enter other type
Submit
"FS" : 0.53
"MM" : 0.12
"KL" : 0.18
"AC" : 0.01
"IPC" : 0
"NT" : 0.01
"SC" : 0
"UN" : 0.18
Tracing Mode
LTTng-kernel
LTTng-UST
Text
Enter Regular Expression


File Edit Navigate Search Project Run File Window Help Window Help
Quick Access
Java
LTTng Kernel
Tracing
Histogram
Sequence Diagram
State System Explorer
Time Chart
Total Anomaly Detection System
Call Stack
Control Flow
Resources
Statistics
Hosts
Add New Host
Host-app-01
Android-01s
states
Diagnosis
Modeling
Select Traces and Modeling Type
Training
Validation
/home/umroot/lttng-traces/
Browse Directory
Select Models
Kernel State Modeling (KSM)
Sliding Window
Classification
Training models
Evaluate Models
Progress Console
Reading Trace Kernel-session-27-13
Transforming to states
Inserting into the database host-app-01
....
Tracing Mode
LTTng-kernel
LTTng-UST
Text
Enter Regular Expression

Conclusion

- **Research threads:** Data preparation, data abstraction, adaptive learning, and infrastructure
- **ADS requirements:** low false positive rate, scalability, and adaptability
- **KSM:** Abstraction is not the enemy of accuracy
- **IBC:** Combining detectors provides better results than using a single detector
- **TotalADS:** An environment for integrating multiple anomaly detection systems

Future Plans

- Continue experimenting with KSM and IBC on other datasets (preferably generated at DRDC)
- Combine additional detectors using IBC
- Start working on adaptive/incremental learning
- Continue improving the maturity level of TotalADS
- Integrate this work with work done at other universities
- Transfer knowledge to DRDC

Thank You

Wahab Hamou-Lhadj, PhD, ing.

Software Behaviour Analysis (SBA) Research Lab
Concordia University
Montreal, QC, Canada

www.ece.concordia.ca/~abdelw/sba