On the Accurate Identification of Network Service Dependencies in Distributed Systems

Barry Peddycord III
NC State University
bwpeddyc@ncsu.edu
@isharacomix

Dr. Peng Ning
NC State University
pning@ncsu.edu

Dr. Sushil Jajodia
George Mason University
jajodia@gmu.edu

Computer Science
NC STATE UNIVERSITY

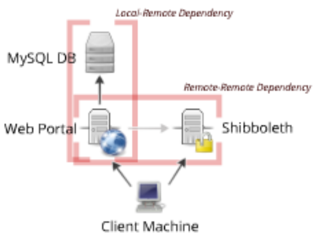
GEORGE MASON UNIVERSITY

USENIX LISA '12
#NSDMiner

This work is supported by the U.S. Army Research Office (ARO) under MURI grant W911NF-09-1-0525

Motivation

Example



Problem

Network Service Dependencies

- Defined by configuration parameters and source code
  - Each service does it differently!
- Often very intricate and subtle
- Hard to keep track
  - How good are YOUR docs?
- Want to identify them automatically

Why bother?

Know thyself

- If dependencies are discovered after a failure occurs, it's too late
- Knowing in advance
  - Improves response time
  - Allows pro-active action to be taken on mission-critical services
- Networks are dynamic

Prior Work

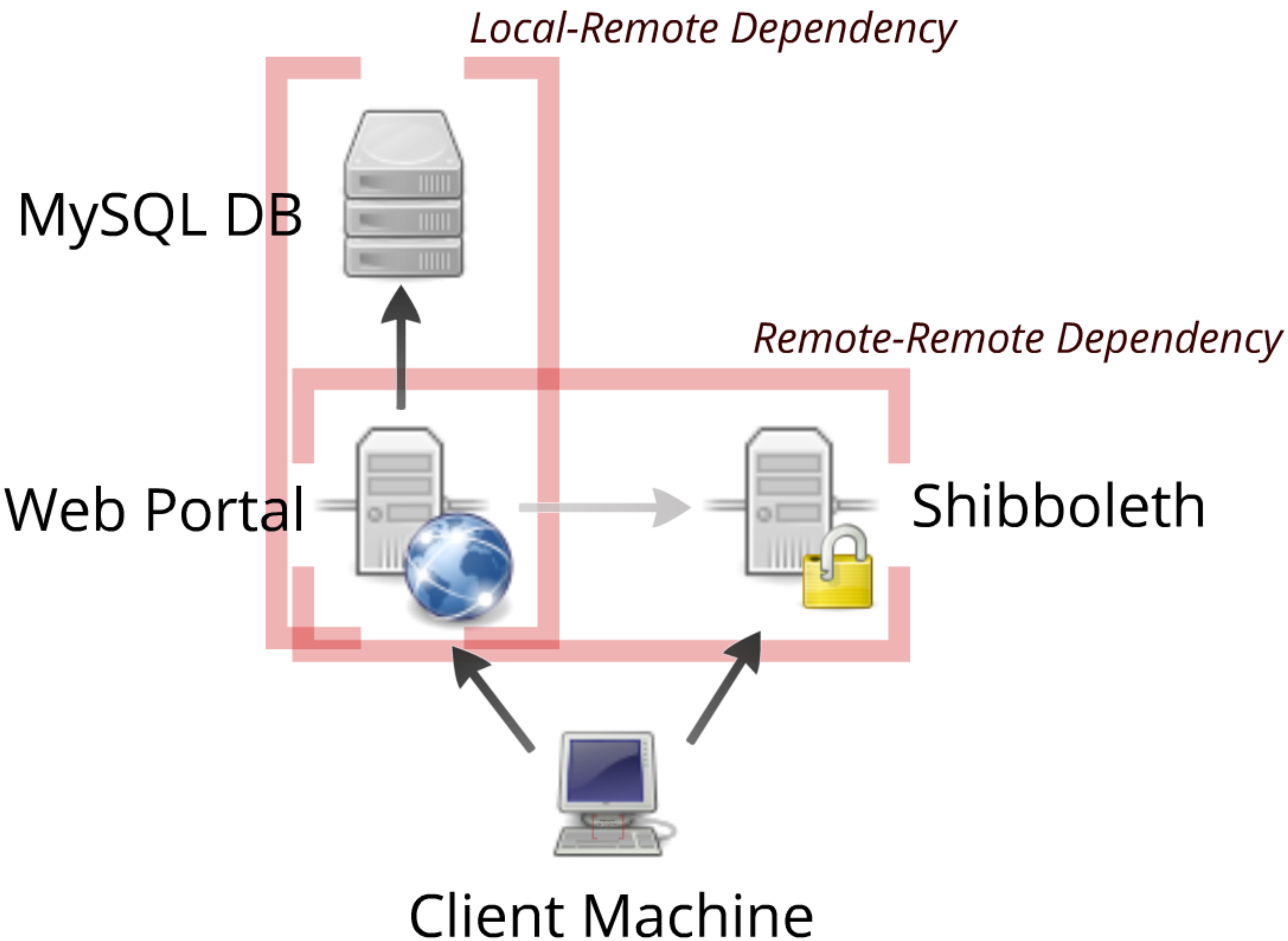
Two Paradigms

- Patterns in the behavior of the network can model its structure
- Previous approaches fall into two categories:

Host-Based

Network-Based

Example


Local-Remote Dependency
MySQL DB
Remote-Remote Dependency
Web Portal
Shibboleth
Client Machine

Problem

A network service is a software application that runs on a server and listens on a port for connections from other applications.

A dependency is a relationship between two services A and B such that A (the depending service) contacts B (the depended service) to complete a task.

Network Service Dependencies

- Defined by configuration parameters and source code
  - Each service does it differently!
- Often very intricate and subtle
- Hard to keep track
  - How good are YOUR docs?
- Want to identify them automatically

A network service is a software application that runs on a server and listens on a port for connections from other applications.

A dependency is a relationship between two services A and B such that A (the depending service) contacts B (the depended service) to complete a task.

Problem

A network service is a software application that runs on a server and listens on a port for connections from other applications.

A dependency is a relationship between two services A and B such that A (the depending service) contacts B (the depended service) to complete a task.

Network Service Dependencies

- Defined by configuration parameters and source code
  - Each service does it differently!
- Often very intricate and subtle
- Hard to keep track
  - How good are YOUR docs?
- Want to identify them automatically

Why bother?

Know thyself

- If dependencies are discovered after a failure occurs, it's too late
- Knowing in advance
  - Improves response time
  - Allows pro-active action to be taken on mission-critical services
- Networks are dynamic

Prior Work

Two Paradigms

- Patterns in the behavior of the network can model its structure
- Previous approaches fall into two categories:

Host-Based

Accurate, but intrusive

- Install an agent (i.e. a kernel module) to track socket/application behavior
  - Magpie [OSDI 2004]
  - Pinpoint [NSDI 2004]
  - Macroscope [CoNEXT 2009]
- Intrusiveness makes them unattractive
  - Security risks
  - Resource contention

Network-Based

Treat hosts as black boxes

- Data-mine on-the-wire network traffic to extract relationships
  - Sherlock [SIGCOMM 2007]
  - eXpose [SIGCOMM 2008]
  - Orion [OSDI 2008]
  - NSDMiner [INFOCOM 2011]
- High-false positive/false negative rates

Host-Based

Accurate, but intrusive

- Install an agent (i.e. a kernel module) to track socket/application behavior
  - Magpie [OSDI 2004]
  - Pinpoint [NSDI 2004]
  - Macroscope [CoNEXT 2009]
- Intrusiveness makes them unattractive
  - Security risks
  - Resource contention

Network-Based

Treat hosts as black boxes

- Data-mine on-the-wire network traffic to extract relationships
  - Sherlock [SIGCOMM 2007]
  - eXpose [SIGCOMM 2008]
  - Orion [OSDI 2008]
  - NSDMiner [INFOCOM 2011]
- High-false positive/false negative rates

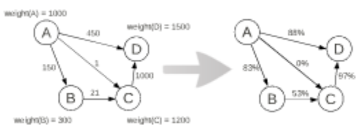
NSDMiner

Intuition

Ranking

Confidence equals:

$$\frac{\log(\text{ number of times A is accessed })}{\log(\text{ number of nested A->B flows })}$$



Why logs?

Two Important Properties

- Not all nested flows are equal
- Give candidates with more evidence more weight
  - "every other flow" means more when it's 10000 than 100
- Later flows are worth less
  - Is 90% less convincing than 95%?

Post-processing

Given a Communication Graph

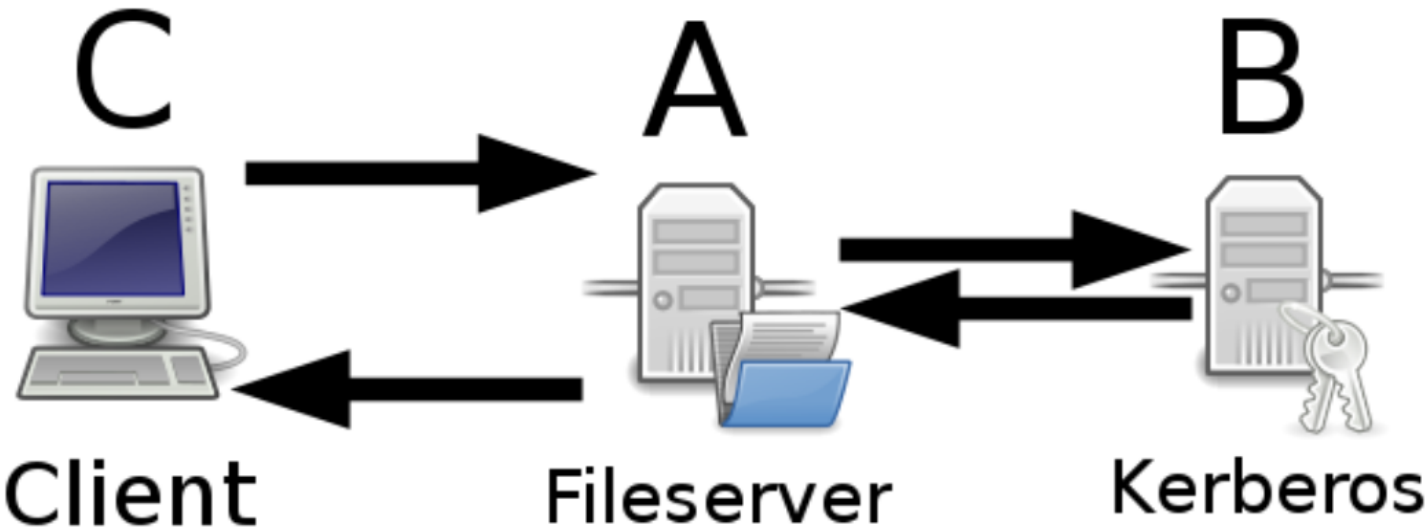
- Less-used services are vulnerable to false positive, false negative
- Post-processing uses overall structure to fine-tune results

The Output

List of Dependency Candidates

- Returns each network service and all of its dependency candidates
- Dependencies ordered by most-likely to least-likely
- Should be verified by hand, so a few false positives are acceptable

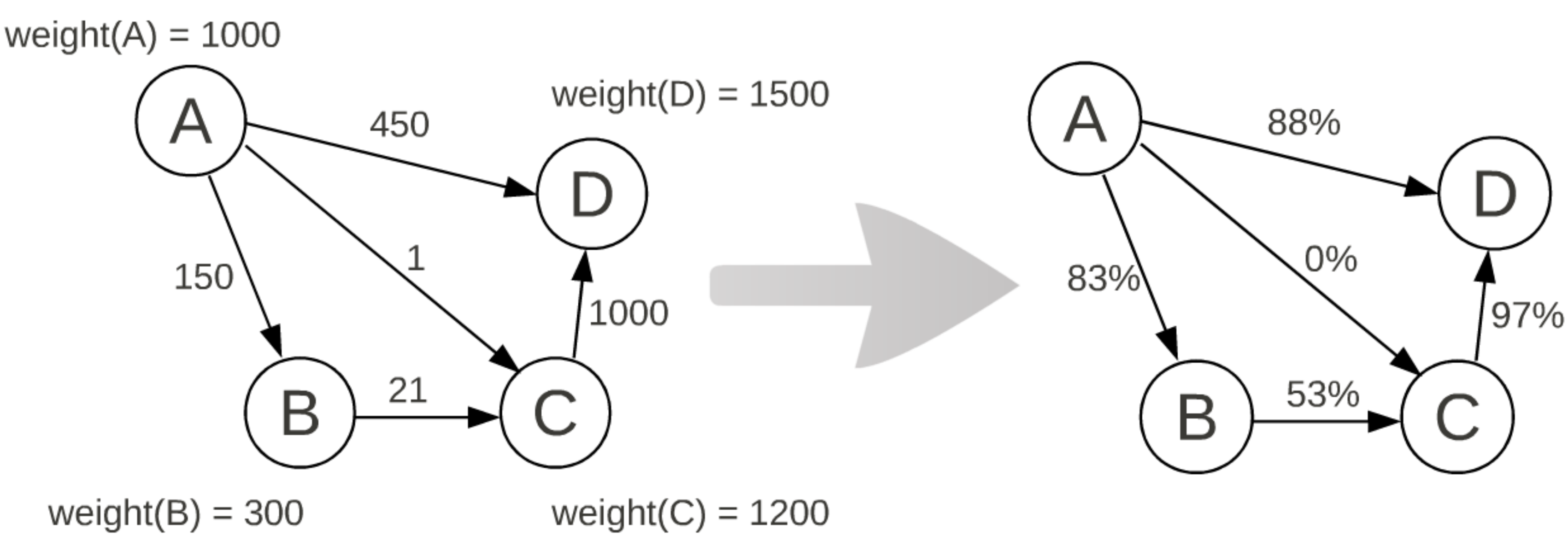
Intuition



Ranking

Confidence equals:

$$\frac{\log(\text{ number of times A is accessed })}{\log(\text{ number of nested A->B flows })}$$



Post-processing

Given a Communication Graph

- Less-used services are vulnerable to false positive, false negative
- Post-processing uses overall structure to fine-tune results

Inference

Clustering

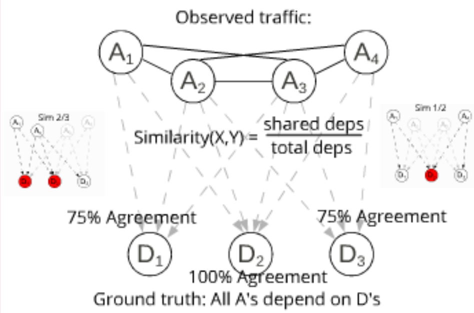
Inference

Intuition

Consider a Web Host

- Many servers are configured the same way (HTTPD) with the same dependencies (MySQL, SMTP, etc)
- Some are more popular than others, having more traffic
- Identify dependencies of less used servers by identifying 'similar' services

Example



Algorithm

- Identify all pairs of similar services above a certain similarity threshold
- Combine pairs into similarity groups
- Calculate agreement on dependency candidates
- Infer dependencies from members of similarity group to most agreed-upon candidates

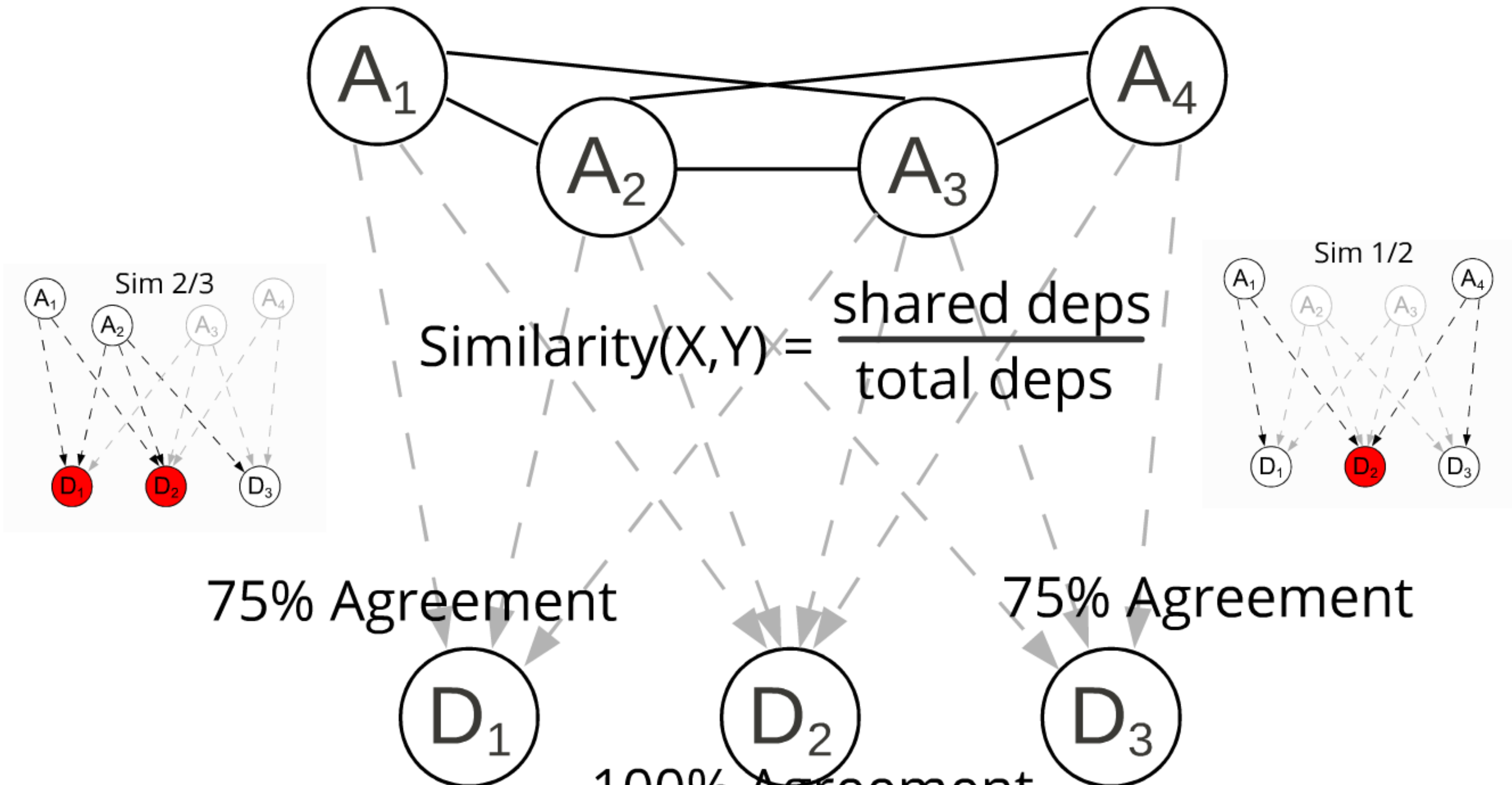
Intuition

Consider a Web Host

- Many servers are configured the same way (HTTPD) with the same dependencies (MySQL, SMTP, etc)
- Some are more popular than others, having more traffic
- Identify dependencies of less used servers by identifying 'similar' services

Example

Observed traffic:



$$\text{Similarity}(X,Y) = \frac{\text{shared deps}}{\text{total deps}}$$

Ground truth: All A's depend on D's

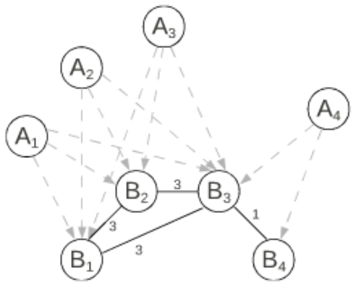
Clustering

Intuition

Backups and Load-Balancing

- In a load-balancing cluster, a depending service will eventually utilize all cluster nodes
- In a backup-cluster, a service will use the primary nodes until they fail, then move to backup nodes
- In both cases, if a service uses one node in a cluster, it uses them all

Example



Algorithm

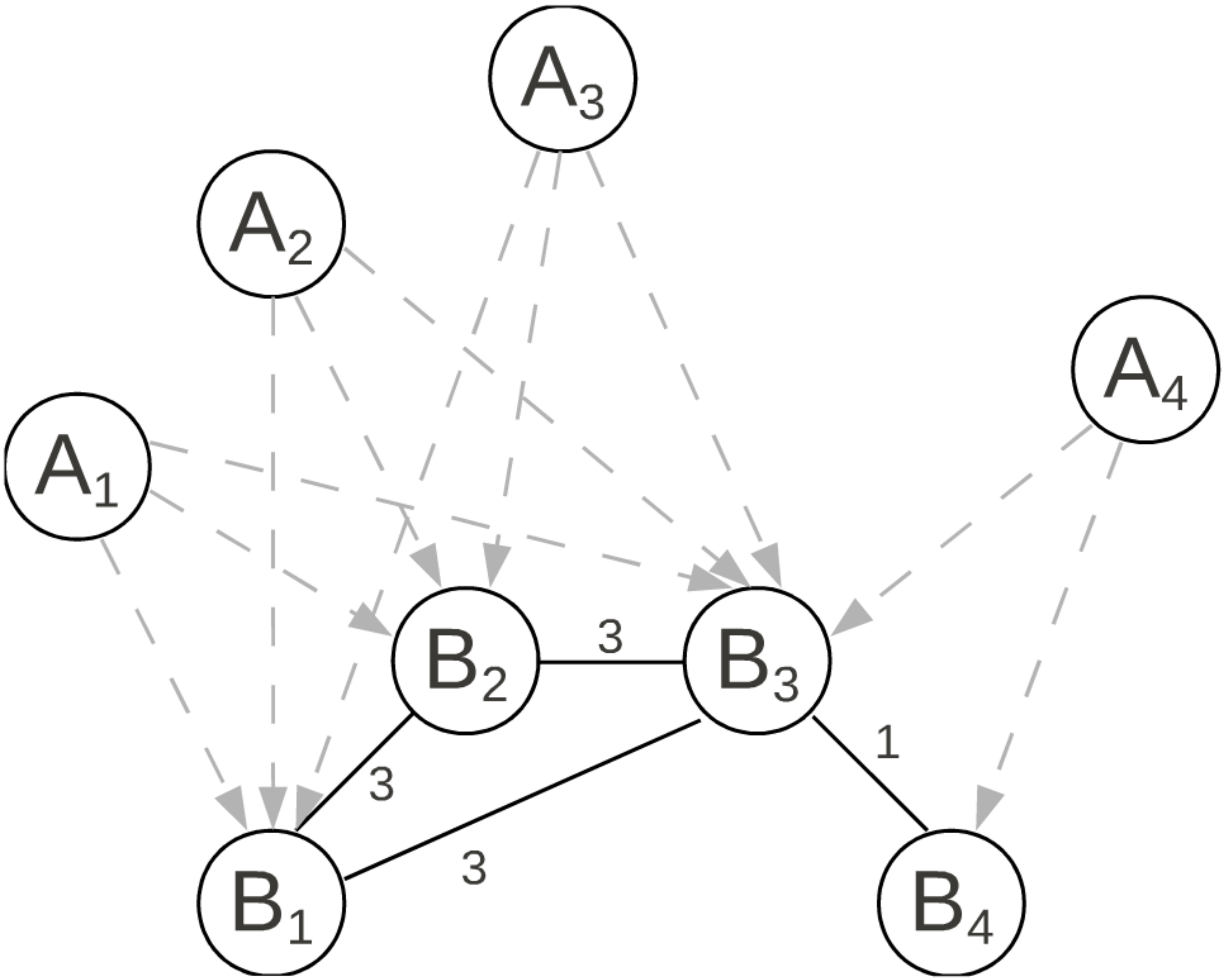
- Count the number of times that pairs of services are depended upon by the same service
- For services that have support above a certain threshold, these services are considered to be in clusters
- Re-interpret services that depend on services in clusters as depending on the entire cluster itself.

Intuition

Backups and Load-Balancing

- In a load-balancing cluster, a depending service will eventually utilize all cluster nodes
- In a backup-cluster, a service will use the primary nodes until they fail, then move to backup nodes
- In both cases, if a service uses one node in a cluster, it uses them all

Example



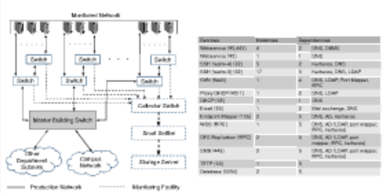
The Output

List of Dependency Candidates

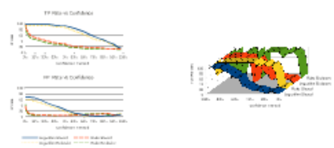
- Returns each network service and all of its dependency candidates
- Dependencies ordered by most-likely to least-likely
- Should be verified by hand, so a few false positives are acceptable

Evaluation

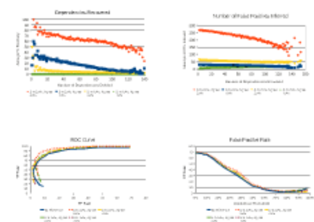
Design



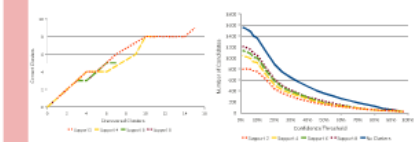
NSDMiner



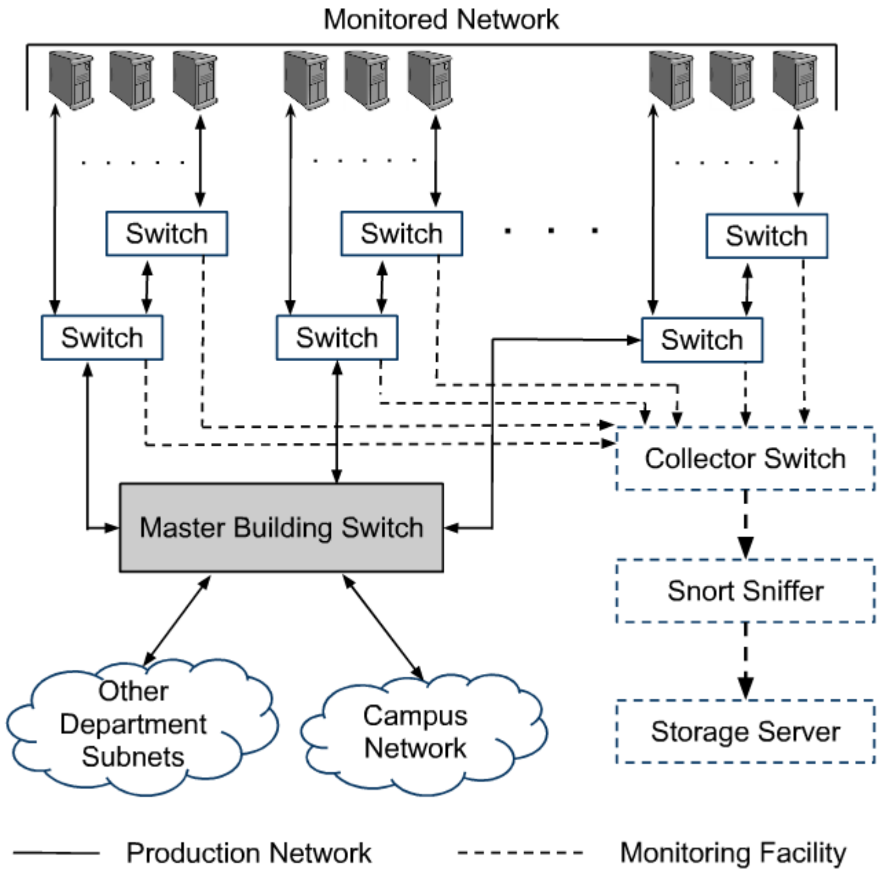
Inference



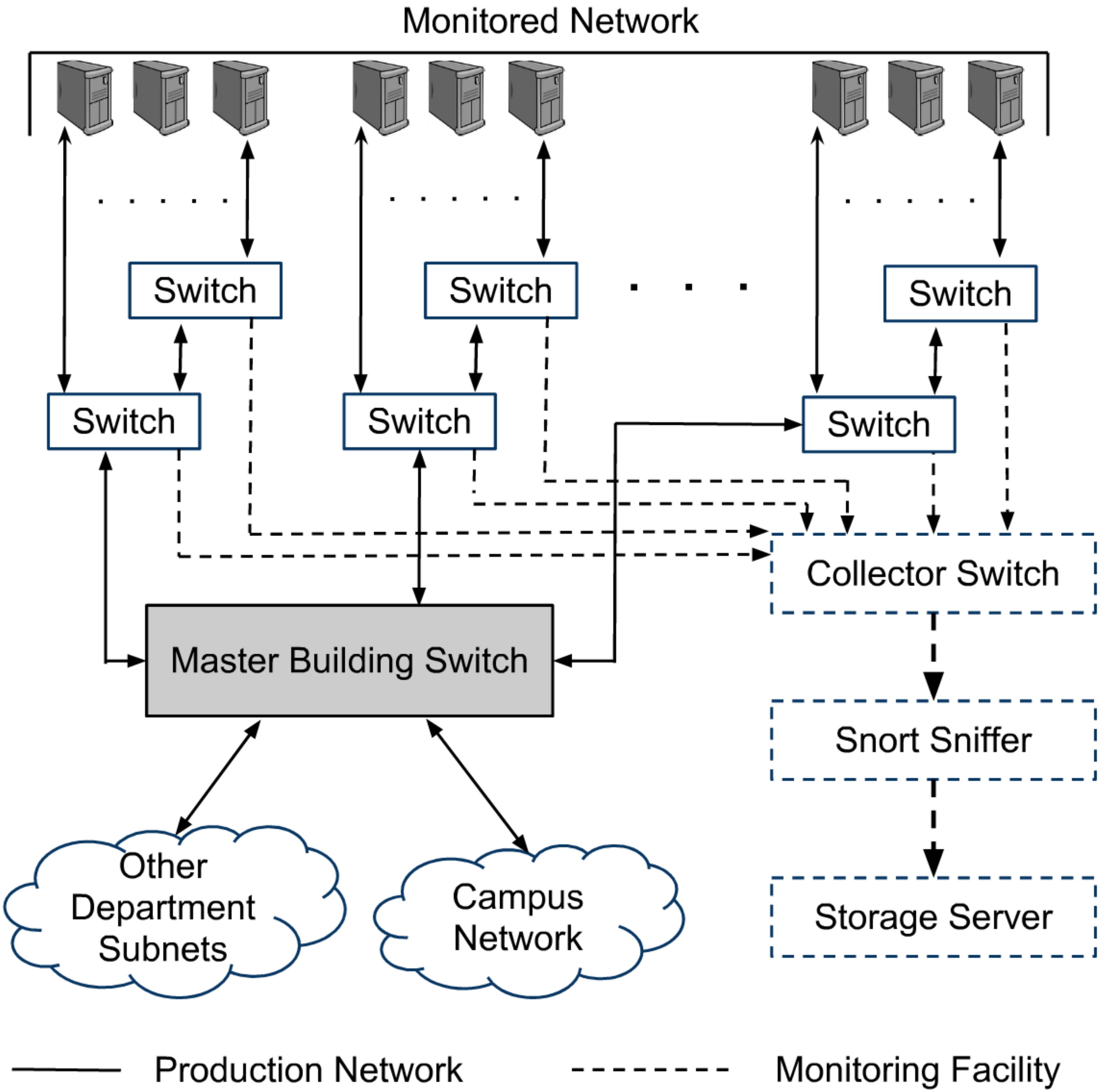
Clustering



Design



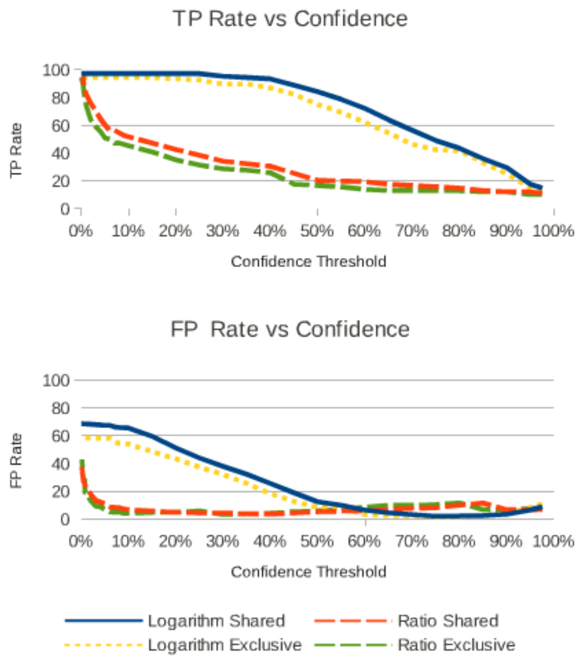
Services	Instances	Dependencies	
Webservice (80,443)	4	2	DNS, DBMS
Webservice (80)	1	1	DNS
SSH (realm-4) (22)	5	2	Kerberos, DNS
SSH (realm-5) (22)	17	3	Kerberos, DNS, LDAP
SVN (8443)	1	4	DNS, LDAP, Port Mapper, RPC
Proxy DHCP (4011)	1	2	DNS, LDAP
DHCP (68)	1	1	DNS
Email (25)	1	2	Mail exchange, DNS
Endpoint Mapper (135)	2	3	DNS, AD, Kerberos
WDS (RPC)	1	5	DNS, AD (LDAP, port mapper, RPC, kerberos)
DFS Replication (RPC)	2	5	DNS, AD (LDAP, port mapper, RPC, kerberos)
SMB (445)	2	5	DNS, AD (LDAP, port mapper, RPC, kerberos)
TFTP (69)	1	0	
Database (3306)	2	0	



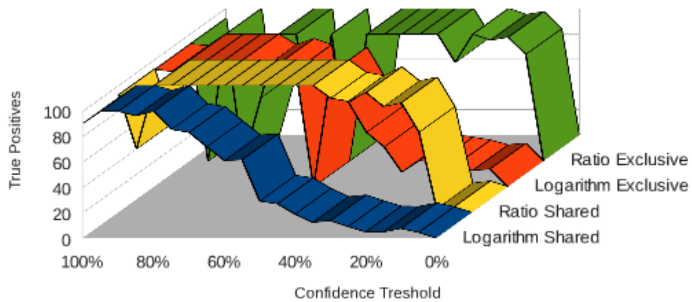
Services
Webser
Webser
SSH (re
SSH (re
SVN (84
Proxy D
DHCP (
Email (2
Endpoin
WDS (R
DFS Re
SMB (4
TFTP (6
Databas

Services	Instances	Dependencies	
Webservice (80,443)	4	2	DNS, DBMS
Webservice (80)	1	1	DNS
SSH (realm-4) (22)	5	2	Kerberos, DNS
SSH (realm-5) (22)	17	3	Kerberos, DNS, LDAP
SVN (8443)	1	4	DNS, LDAP, Port Mapper, RPC
Proxy DHCP (4011)	1	2	DNS, LDAP
DHCP (68)	1	1	DNS
Email (25)	1	2	Mail exchange, DNS
Endpoint Mapper (135)	2	3	DNS, AD, Kerberos
WDS (RPC)	1	5	DNS, AD (LDAP, port mapper, RPC, kerberos)
DFS Replication (RPC)	2	5	DNS, AD (LDAP, port mapper, RPC, kerberos)
SMB (445)	2	5	DNS, AD (LDAP, port mapper, RPC, kerberos)
TFTP (69)	1	0	
Database (3306)	2	0	

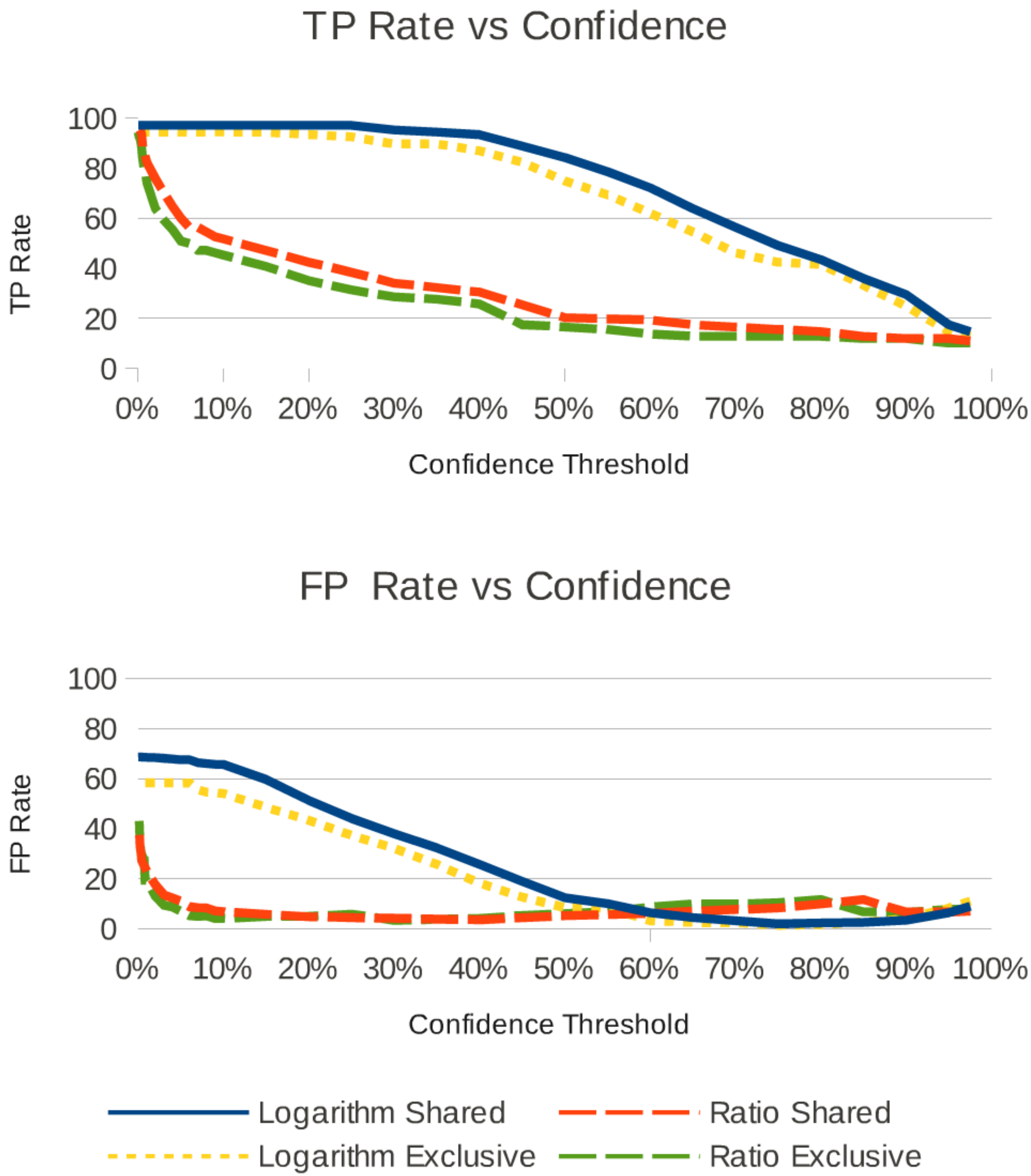
NSDMiner



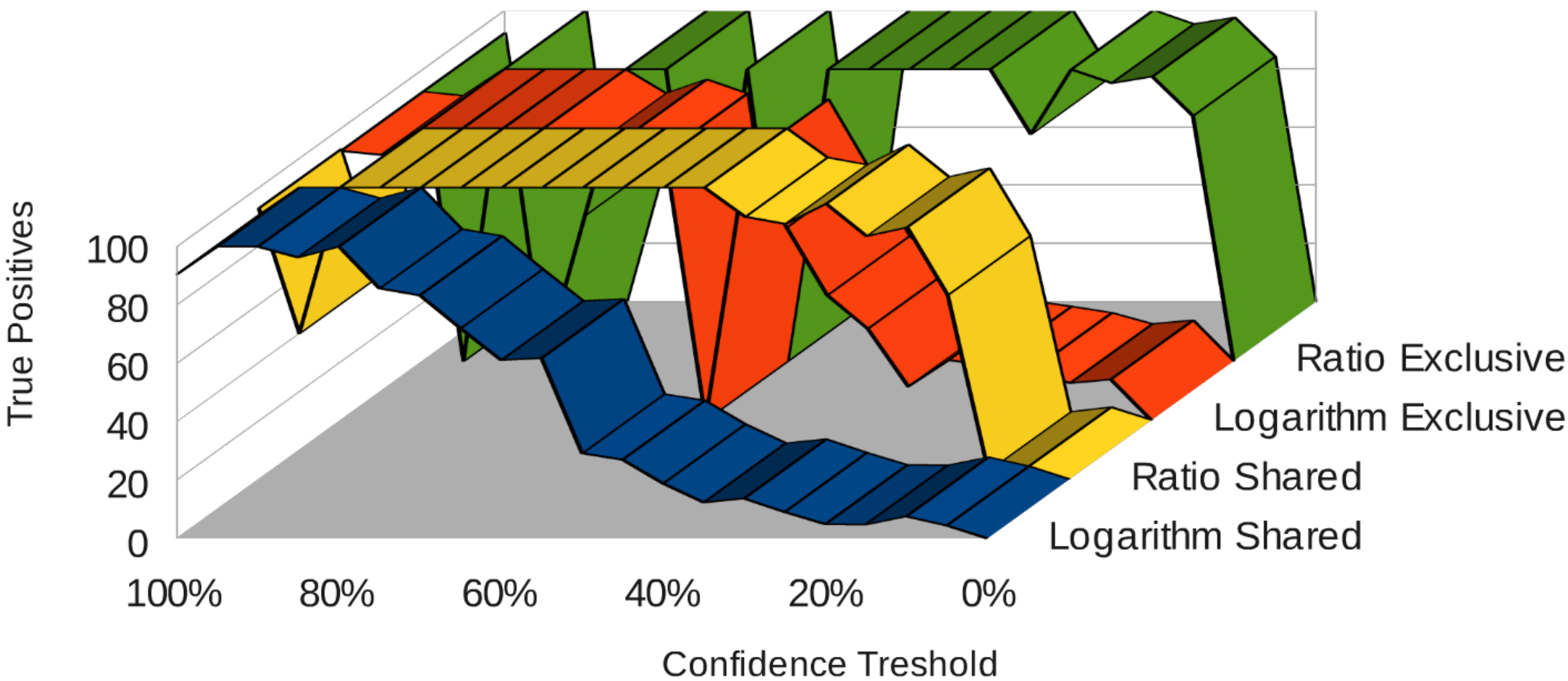
TP Rate vs Confidence
TP Rate
100
80
60
40
20
0
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Confidence Threshold
FP Rate vs Confidence
FP Rate
100
80
60
40
20
0
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Confidence Threshold
Logarithm Shared
Ratio Shared
Logarithm Exclusive
Ratio Exclusive



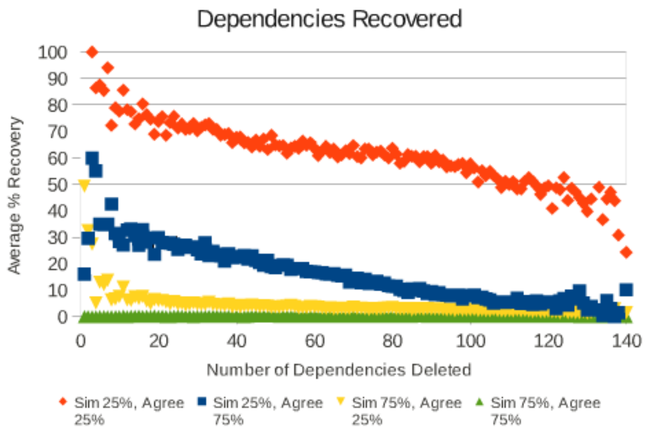
True Positives
100
80
60
40
20
0
100% 80% 60% 40% 20% 0%
Confidence Treshold
Ratio Exclusive
Logarithm Exclusive
Ratio Shared
Logarithm Shared



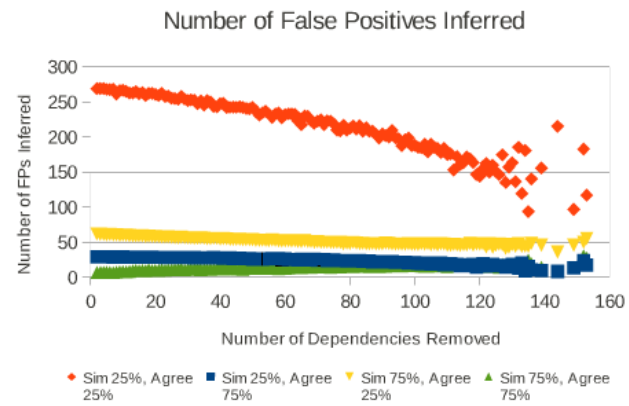
TP Rate vs Confidence
TP Rate
100
80
60
40
20
0
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Confidence Threshold
FP Rate vs Confidence
FP Rate
100
80
60
40
20
0
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Confidence Threshold
Logarithm Shared
Ratio Shared
Logarithm Exclusive
Ratio Exclusive


True Positives
100
80
60
40
20
0
100%
80%
60%
40%
20%
0%
Confidence Treshold
Ratio Exclusive
Logarithm Exclusive
Ratio Shared
Logarithm Shared

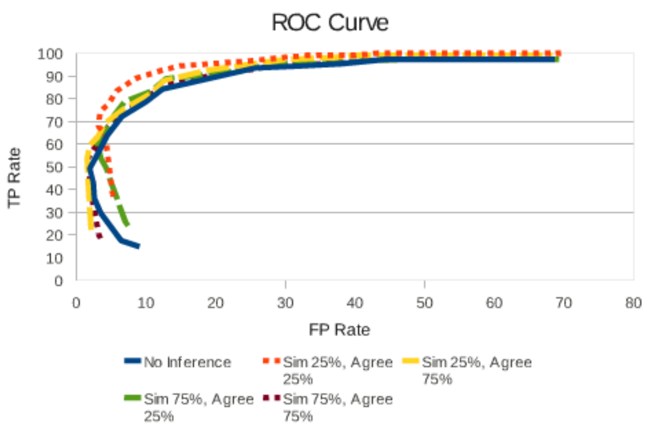
Inference



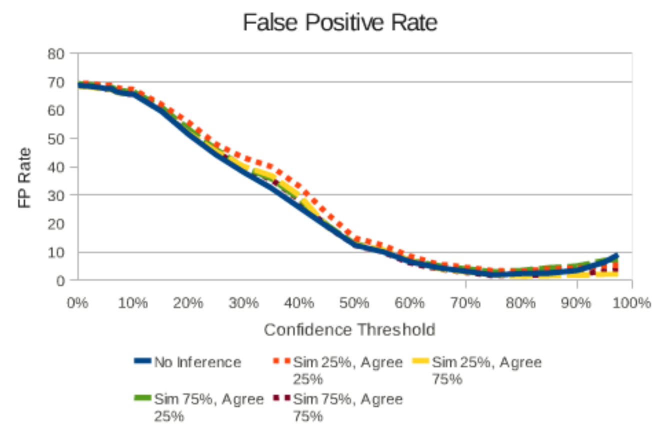
Dependencies Recovered
Average % Recovery
100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
Number of Dependencies Deleted
Sim 25%, Agree 25%
Sim 25%, Agree 75%
Sim 75%, Agree 25%
Sim 75%, Agree 75%



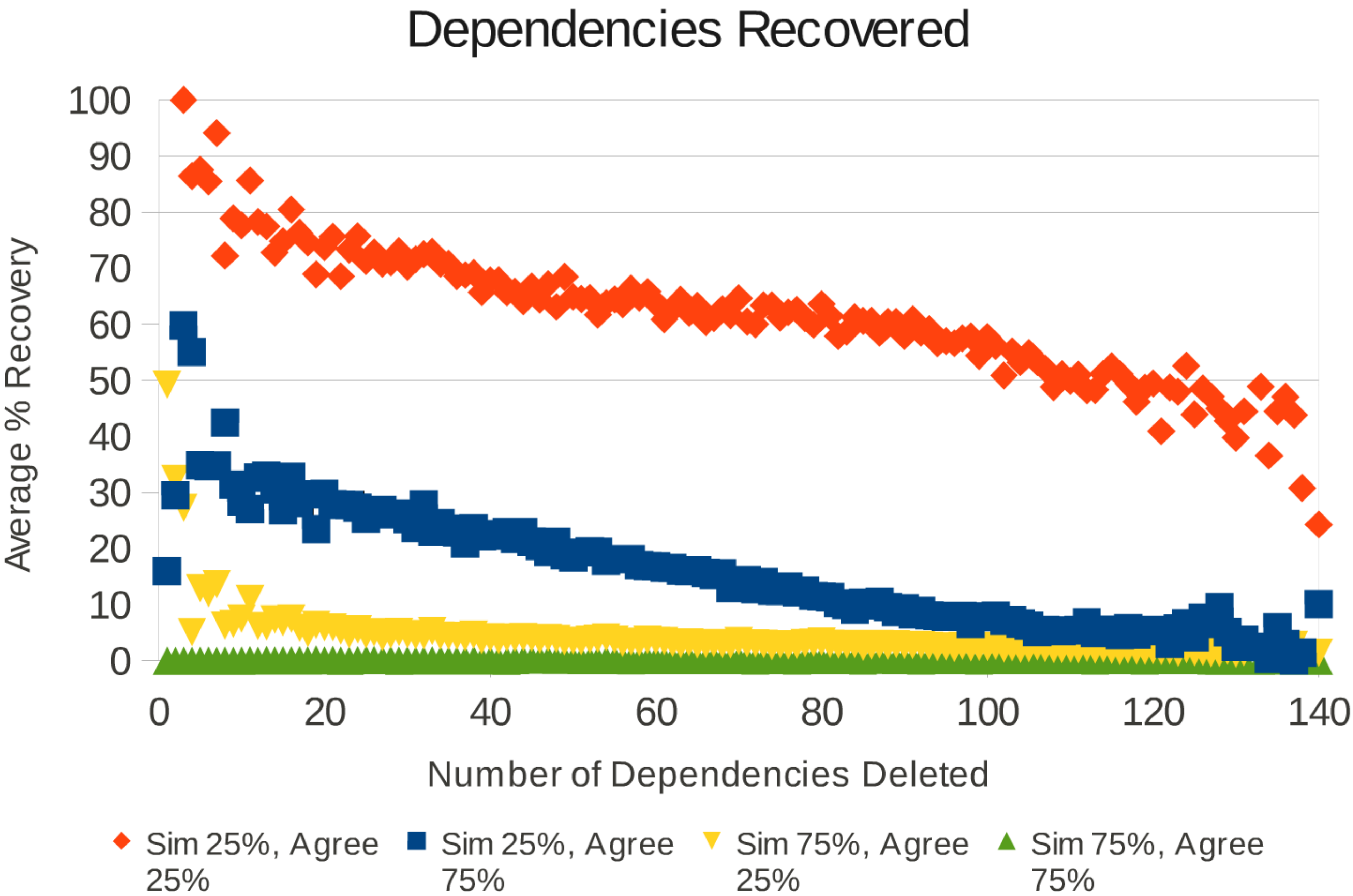
Number of False Positives Inferred
Number of FPs Inferred
300
250
200
150
100
50
0
0
20
40
60
80
100
120
140
160
Number of Dependencies Removed
Sim 25%, Agree 25%
Sim 25%, Agree 75%
Sim 75%, Agree 25%
Sim 75%, Agree 75%

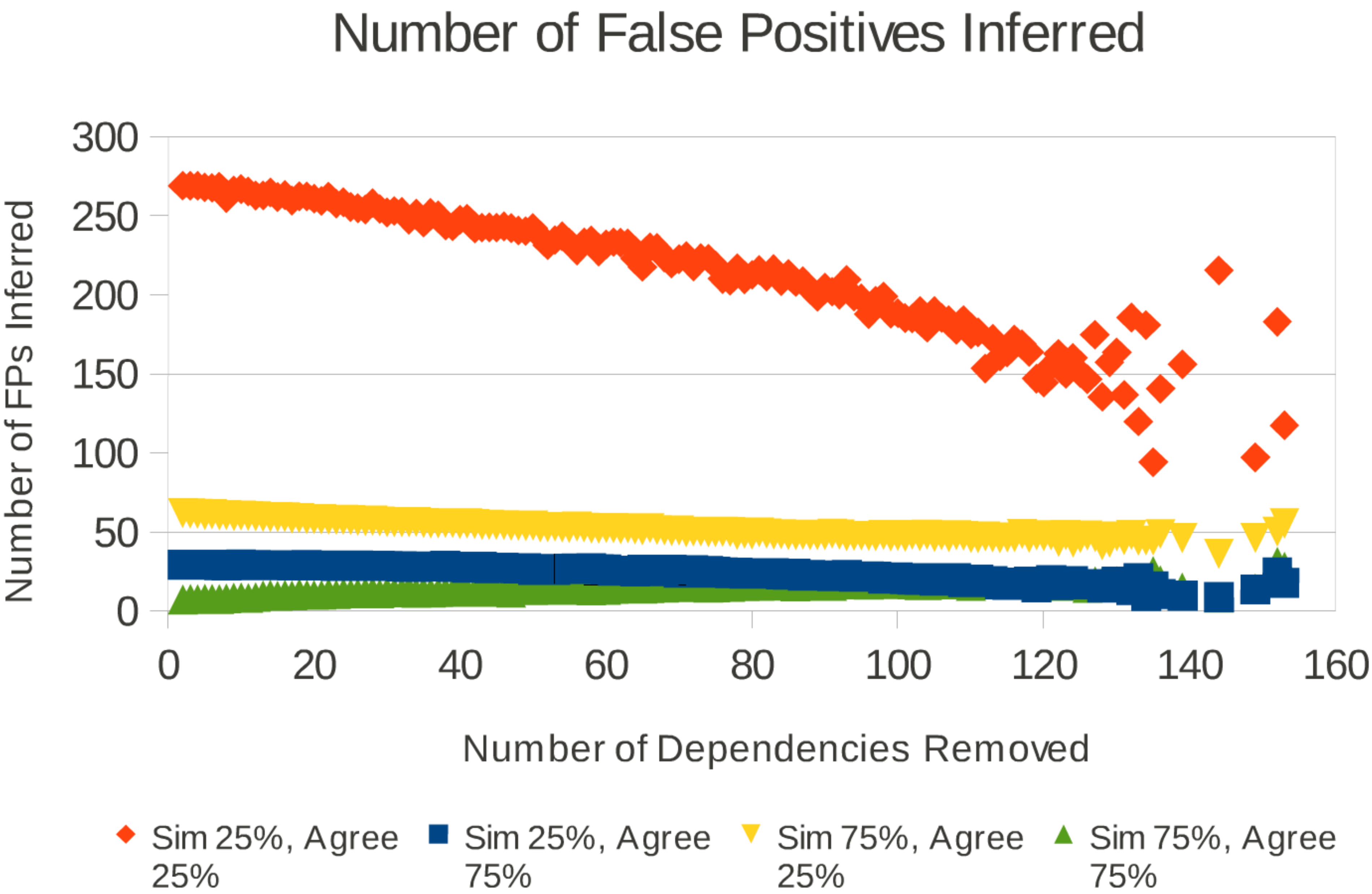


ROC Curve
TP Rate
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
FP Rate
No Inference
Sim 25%, Agree 25%
Sim 25%, Agree 75%
Sim 75%, Agree 25%
Sim 75%, Agree 75%

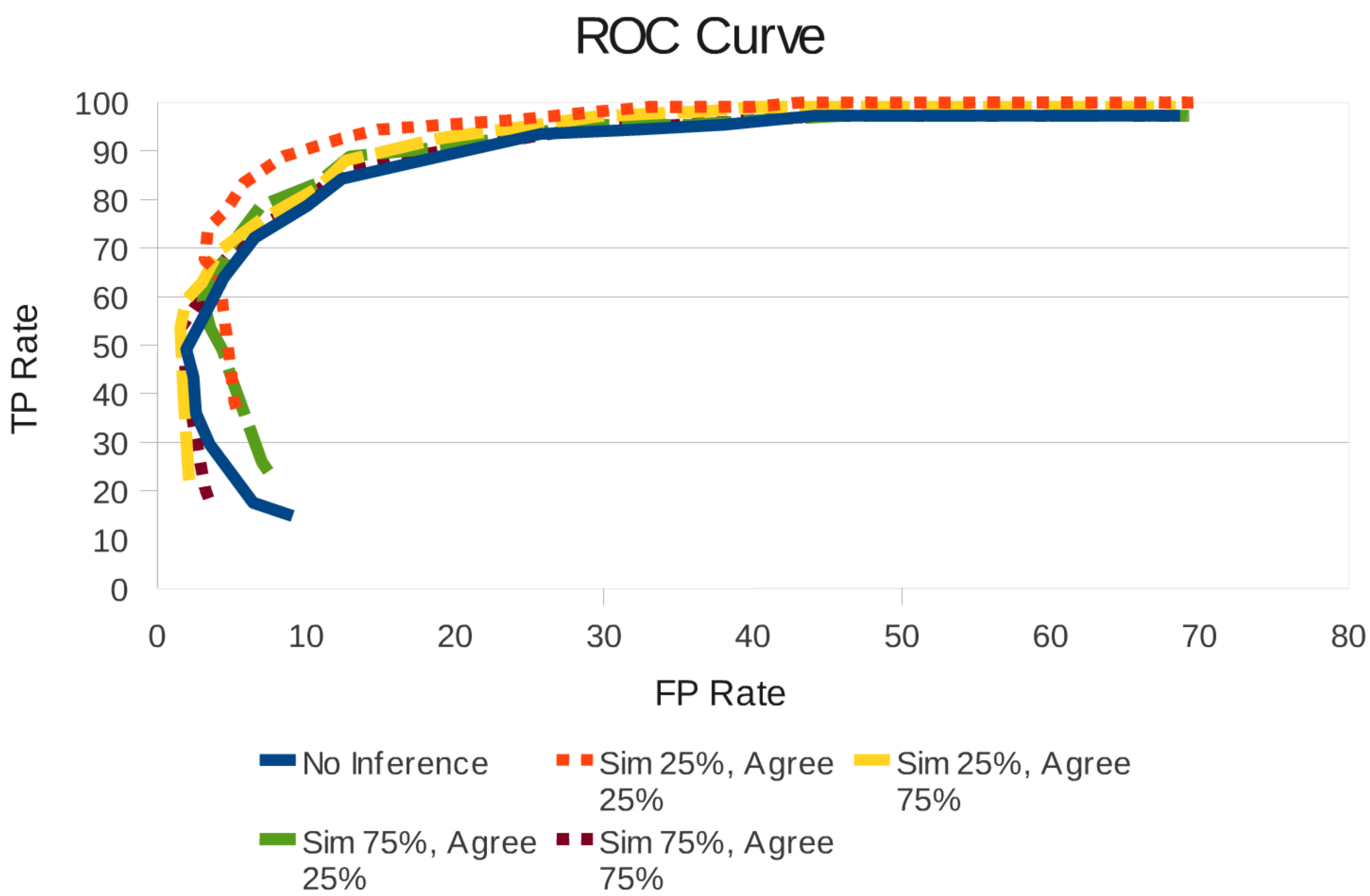


False Positive Rate
FP Rate
80
70
60
50
40
30
20
10
0
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Confidence Threshold
No Inference
Sim 25%, Agree 25%
Sim 25%, Agree 75%
Sim 75%, Agree 25%
Sim 75%, Agree 75%

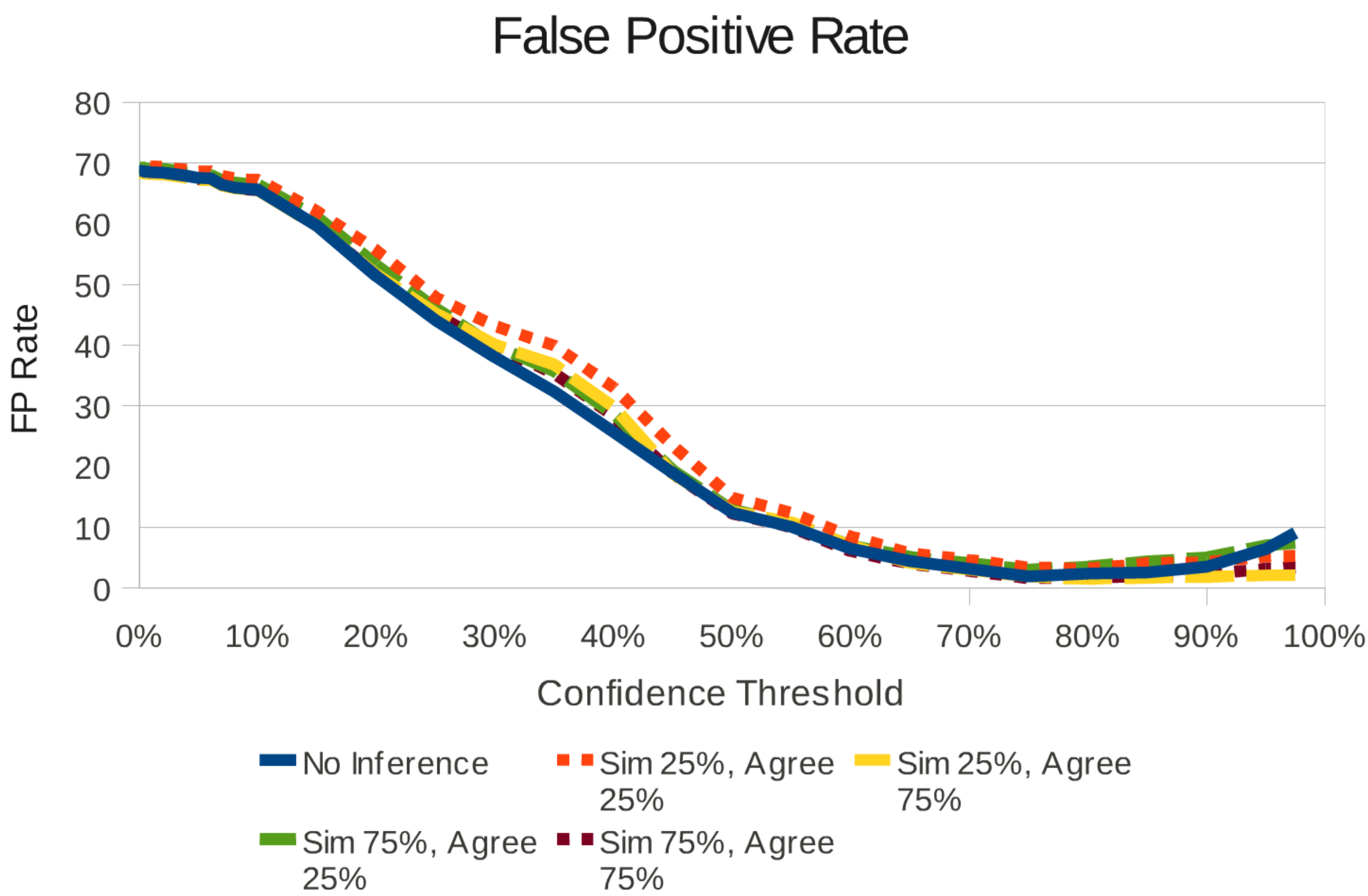

Dependencies Recovered
Average % Recovery
100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
Number of Dependencies Deleted
Sim 25%, Agree 25%
Sim 25%, Agree 75%
Sim 75%, Agree 25%
Sim 75%, Agree 75%



Number of False Positives Inferred
Number of FPs Inferred
300
250
200
150
100
50
0
0
20
40
60
80
100
120
140
160
Number of Dependencies Removed
Sim 25%, Agree 25%
Sim 25%, Agree 75%
Sim 75%, Agree 25%
Sim 75%, Agree 75%



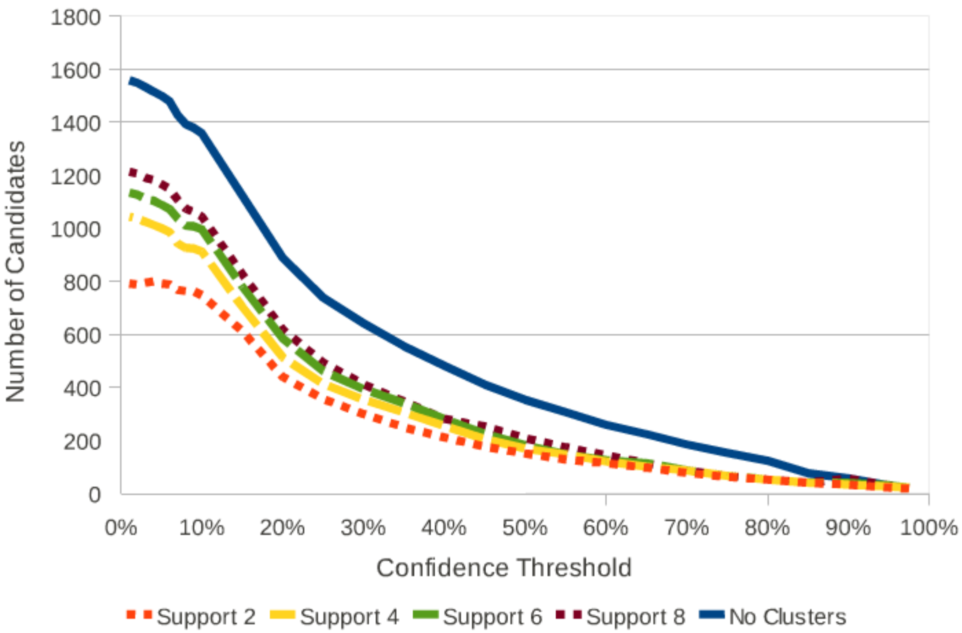
ROC Curve
TP Rate
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
FP Rate
No Inference
Sim 25%, Agree 25%
Sim 25%, Agree 75%
Sim 75%, Agree 25%
Sim 75%, Agree 75%



False Positive Rate
FP Rate
80
70
60
50
40
30
20
10
0
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Confidence Threshold
No Inference
Sim 25%, Agree 25%
Sim 25%, Agree 75%
Sim 75%, Agree 25%
Sim 75%, Agree 75%

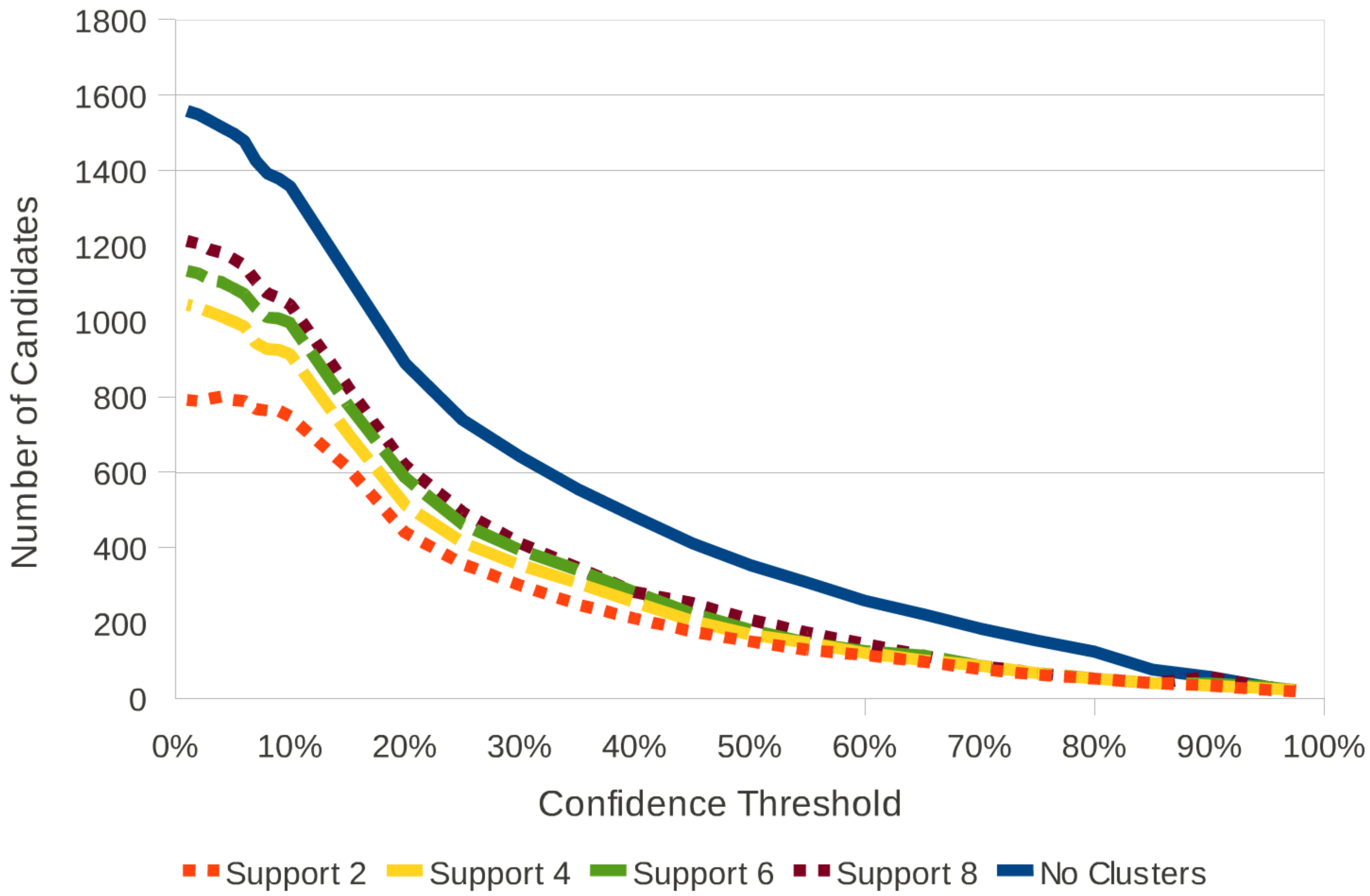
Clustering

Correct Clusters
10
8
6
4
2
0
0
2
4
6
8
10
12
14
16
Discovered Clusters
Support 2
Support 4
Support 6
Support 8



Number of Candidates
1800
1600
1400
1200
1000
800
600
400
200
0
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Confidence Threshold
Support 2
Support 4
Support 6
Support 8
No Clusters

Correct Clusters
10
8
6
4
2
0
0
2
4
6
8
10
12
14
16
Discovered Clusters
Support 2
Support 4
Support 6
Support 8



1800
1600
1400
1200
1000
800
600
400
200
0
Number of Candidates
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Confidence Threshold
Support 2
Support 4
Support 6
Support 8
No Clusters

Deployment

Open Source

Available on SourceForge

- Written as a Python Module
  - 'import nsdminer'
- Comes with a command-line interface for processing data

What's needed

Collect the Data

- Collect all network traffic from network switches
  - Export netflows from switches
  - Use packet mirroring - forward and save all pcap headers of packets
- Usually a week of packets is needed

Using NSDMiner

Just install and run!

- Run 'nsdminer' to process your data
  - Command line options let you choose various parameters
  - Detailed in the paper and README
- Output will be a list of services, dependencies, and confidence values

Going Beyond

- Extend and improve NSDMiner using the features of the 'ndminer' Python library.
- Use it in your own networks and let us know how it works for you in the SourceForge forum!

Open Source

Available on SourceForge

- Written as a Python Module
  - 'import nsdminer'
- Comes with a command-line interface for processing data

What's needed

Collect the Data

- Collect all network traffic from network switches
  - Export netflows from switches
  - Use packet mirroring - forward and save all pcap headers of packets
- Usually a week of packets is needed

Using NSDMiner

Just install and run!

- Run 'nsdminer' to process your data
  - Command line options let you choose various parameters
  - Detailed in the paper and README
- Output will be a list of services, dependencies, and confidence values

Going Beyond

- Extend and improve NSDMiner using the features of the 'ndminer' Python library.
- Use it in your own networks and let us know how it works for you in the SourceForge forum!

Conclusions

NSDMiner

- Non-intrusive and fairly accurate
- Open Source Python Module
  - http://sf.net/p/nsdminer
- Future work includes
  - Making it work in real time
  - Identifying remote-remote dependencies

On the Accurate Identification of Network Service Dependencies in Distributed Systems

Barry Peddycord III
NC State University
bwpeddyc@ncsu.edu
@isharacomix

Dr. Peng Ning
NC State University
pning@ncsu.edu

Dr. Sushil Jajodia
George Mason University
jajodia@gmu.edu

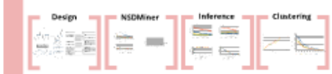
Computer Science
NC STATE UNIVERSITY

GEORGE MASON UNIVERSITY

Motivation

NSDMiner

Evaluation



Deployment

Conclusions

NSDMiner

- Non-intrusive and fairly accurate
- Open Source Python Module
  - http://sf.net/p/nsdminer
- Future work includes
  - Making it work in real time
  - Identifying remote-remote dependencies

USENIX LISA '12

#NSDMiner

This work is supported by the U.S. Army Research Office (ARO) under MURI grant W911NF-09-1-0525