Computer Science
NC STATE UNIVERSITY

On the Accurate Detection of Network Service Dependencies In Distributed Systems

(To appear in USENIX LISA 2012)

Barry Wayne Peddycord III

Written Qualifying Examination

July 11, 2012

Advisor: Dr. Peng Ning

Agenda

Computer Science
NC STATE UNIVERSITY

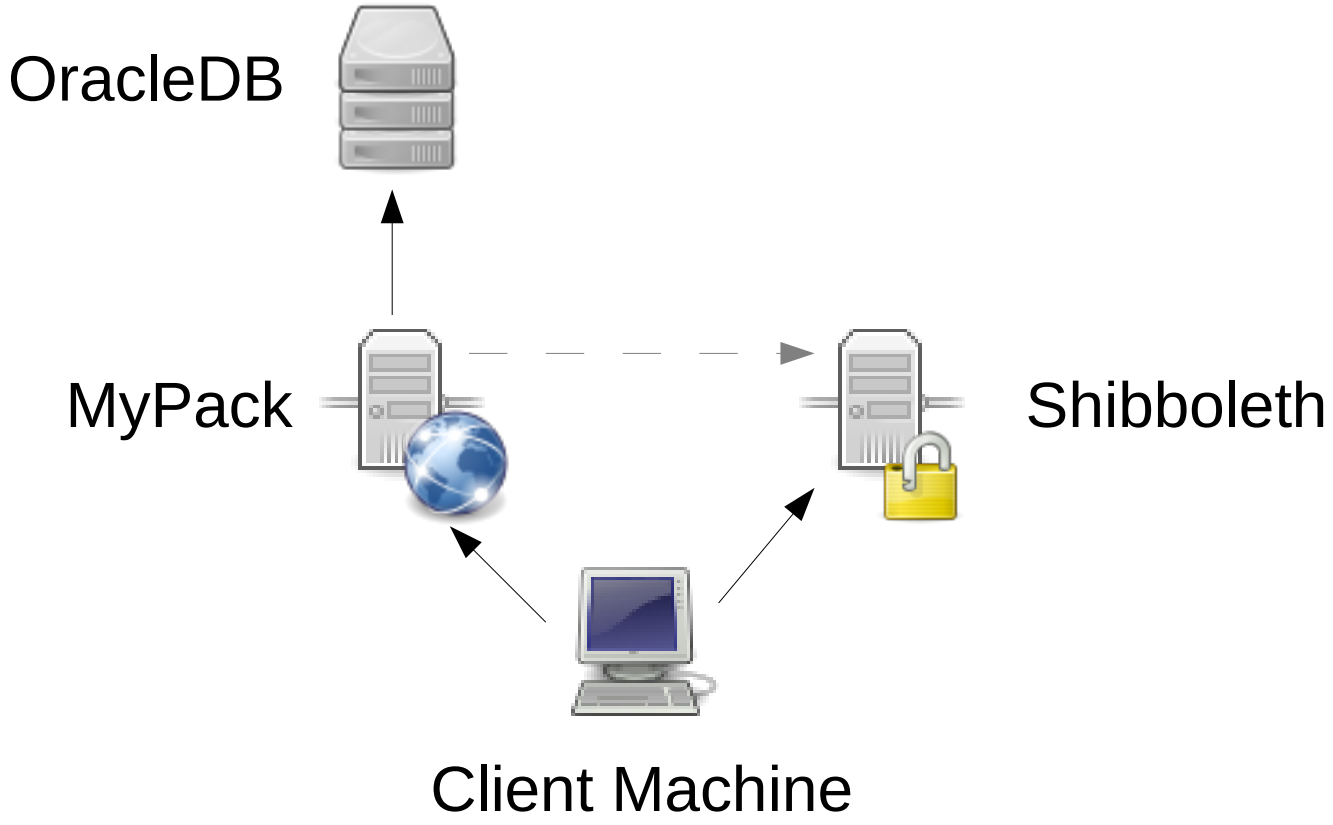
- Introduction
- Motivation
- Related Work
- Our Contributions
  - Logarithm-Based Ranking
  - Service Inference
  - Detection of Network Service Clusters
- Experimental Evaluation
- Conclusion and Future Work

Introduction

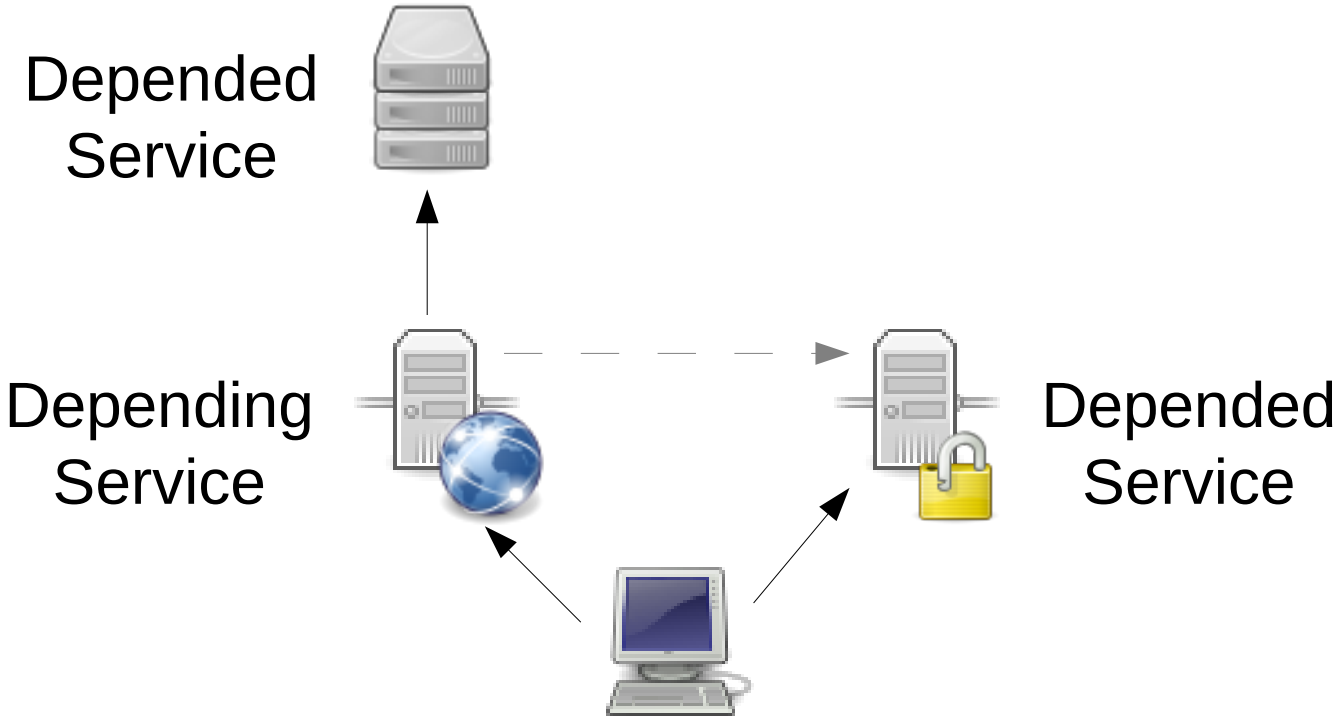
Computer Science
NC STATE UNIVERSITY

- Enterprise networks are comprised of many network services
- These network services depend on other network services for functionality and information
- This can create a complex web of interdependencies

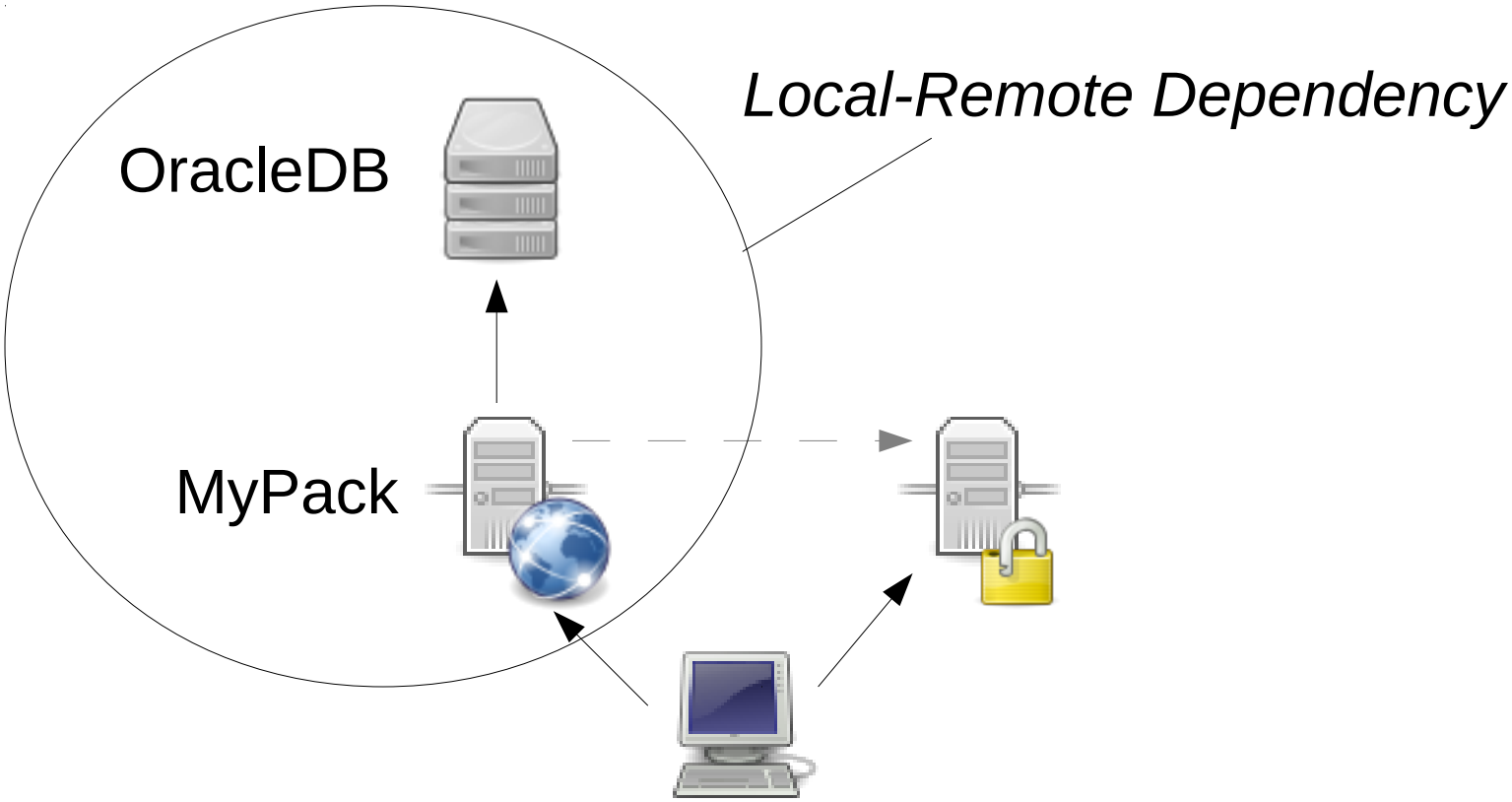
Network Service Dependencies



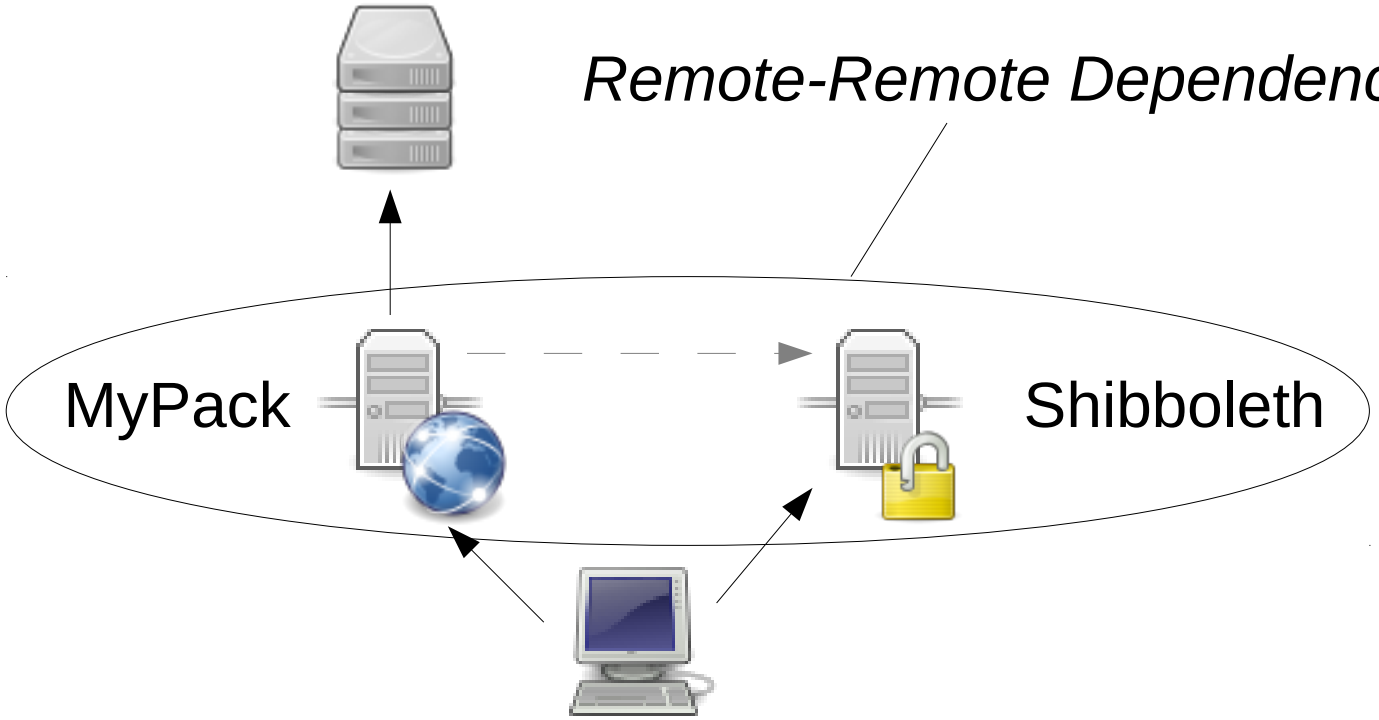
Network Service Dependencies



Network Service Dependencies



Network Service Dependencies



Motivation

Computer Science
NC STATE UNIVERSITY

- If a server goes down, it can potentially take down any of its depending services
- Knowing dependencies...
  - makes it easier to react to reports of system failures
  - makes it easier to identify critical components in order to harden them (through redundancy, etc)
- Automatic approaches can cope with changing configurations better than manual analysis

Intuition

Computer Science
NC STATE UNIVERSITY

- Most dependencies are explicitly defined in network service configuration files
- Automated approaches look for traffic patterns that suggest dependency relationships
- If there is a dependency between services in a network, patterns will be apparent between them

Previous Works

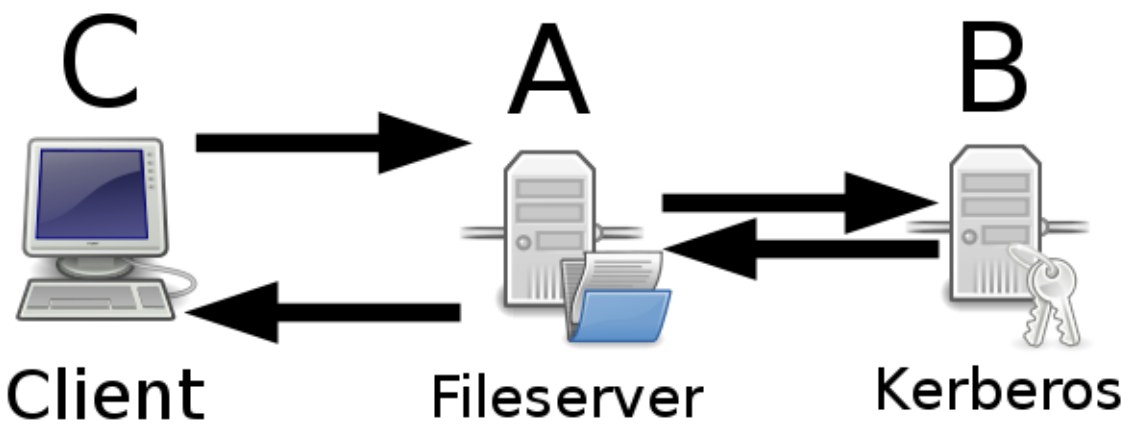
Computer Science
NC STATE UNIVERSITY

- Host-Based
  - Magpie [OSDI 2004]
  - Pinpoint [NSDI 2004]
  - Macroscope [CoNEXT 2009]
- Network Based
  - Sherlock [SIGCOMM 2007]
  - eXpose [SIGCOMM 2008]
  - Orion [OSDI 2008]
  - NSDMiner [INFOCOM 2011]

NSDMiner

Computer Science
NC STATE UNIVERSITY

- NSDMiner [INFOCOM 2011] is an effective network-based approach for finding dependencies
- Looks for nested network flows
- Unobtrusive and more accurate than prior work



Limitations of Prior Work

Computer Science
NC STATE UNIVERSITY

- Ranking of dependency candidates is unintuitive
  - Should true positives have less than 30% confidence?
  - Causes high false positives
- Hard to identify dependencies of infrequently used services
- Can't automatically take advantage of overall server configurations
  - Similar services
  - Redundant service clusters

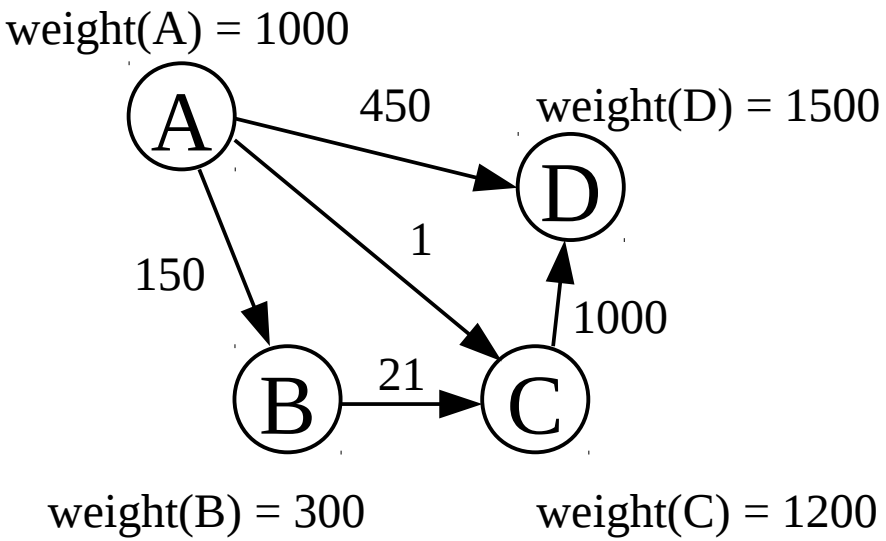
Our Contributions

Computer Science
NC STATE UNIVERSITY

- We develop three novel approaches to improving the detection of network service dependencies
  - A logarithmic based ranking scheme that reduces false positives
  - An inference technique that discovers dependencies of infrequently used services
  - An approach to identify redundant clusters of services in a network
- We implement these as extensions to NSDMiner and evaluate them on production network traffic

Ratio Ranking

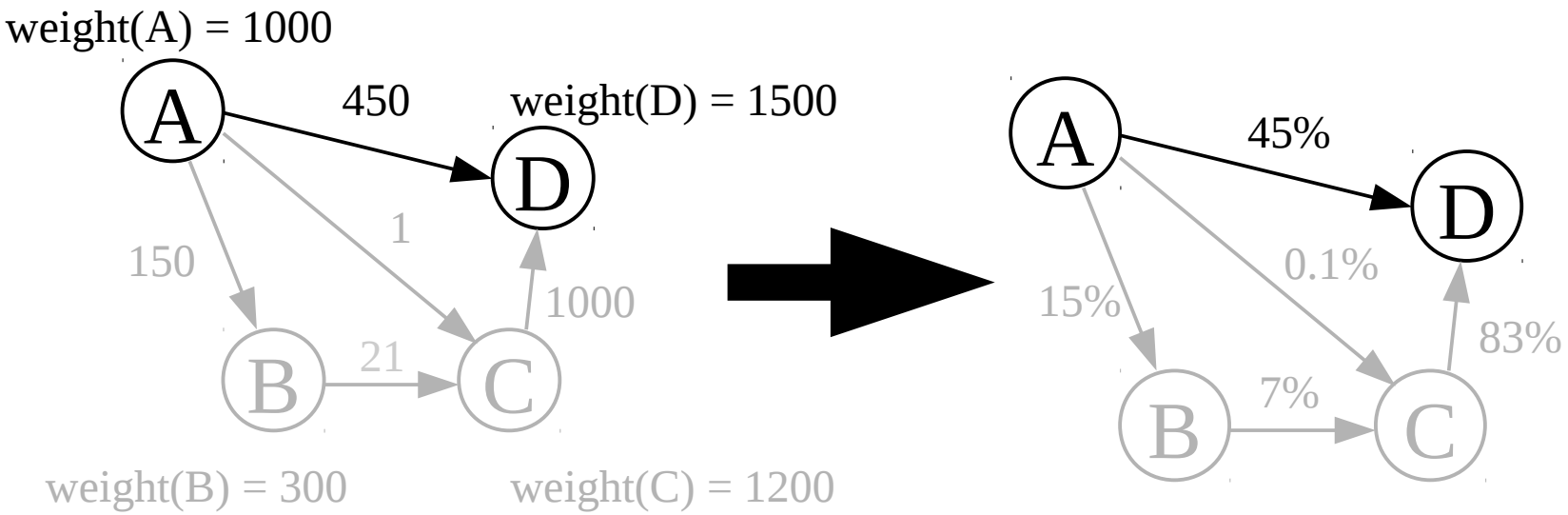
- NSDMiner originally uses a *ratio-based ranking*
- *Confidence* is *weight(edge)/weight(node)*
  - weight(A) = number of times A is accessed
  - weight(A → B) = number of nested flows from A to B



Ratio Ranking (cont)

Computer Science
NC STATE UNIVERSITY

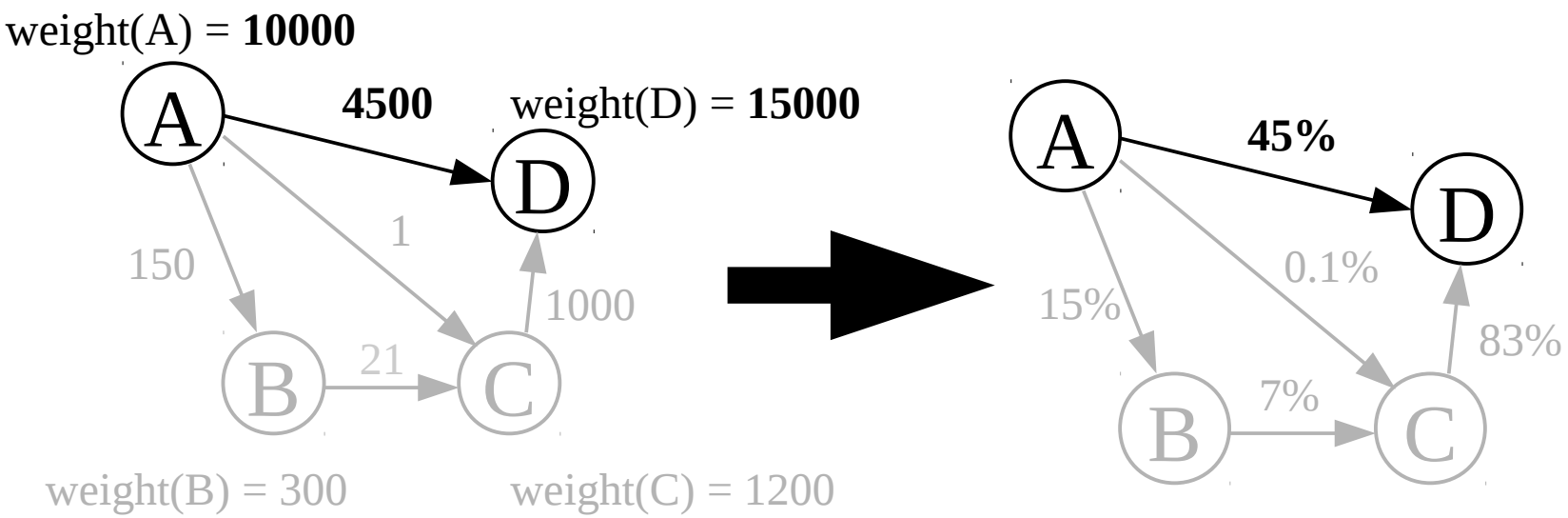
Consider A → D: even though a nested flow to D occurs *every other time* A is accessed, it is only given a confidence of less than 45%.



Ratio Ranking (cont)

Computer Science
NC STATE UNIVERSITY

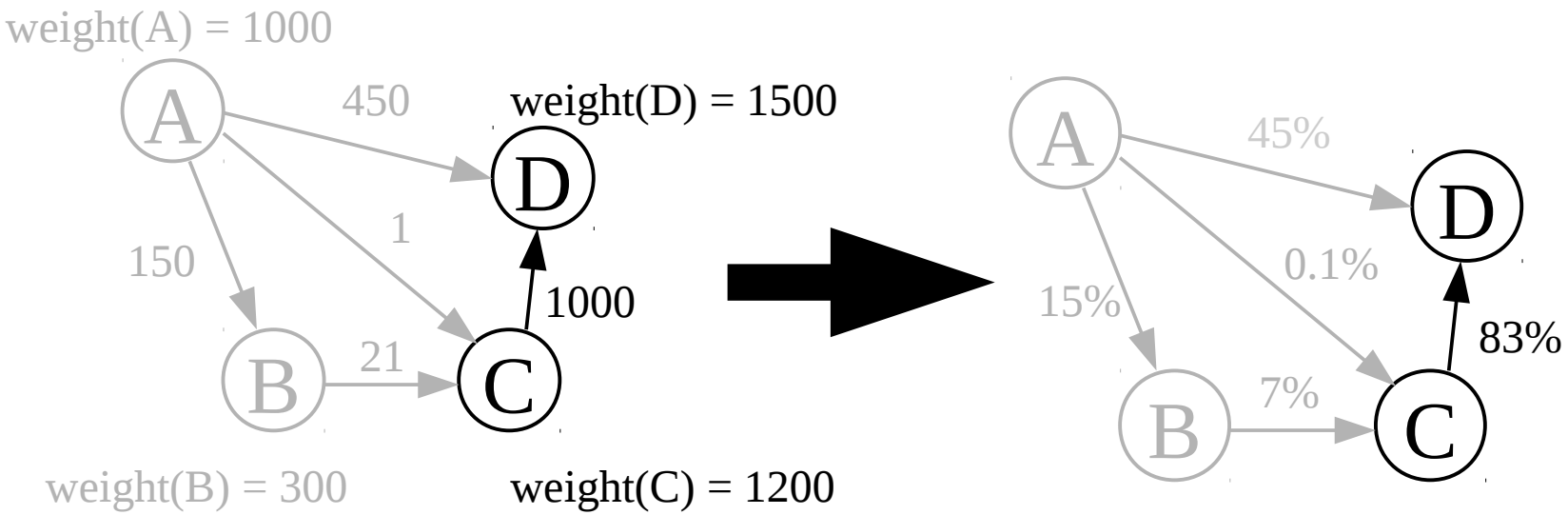
Even if the weights are increased by 10x, the confidence remains the same.



Ratio Ranking (cont)

Computer Science
NC STATE UNIVERSITY

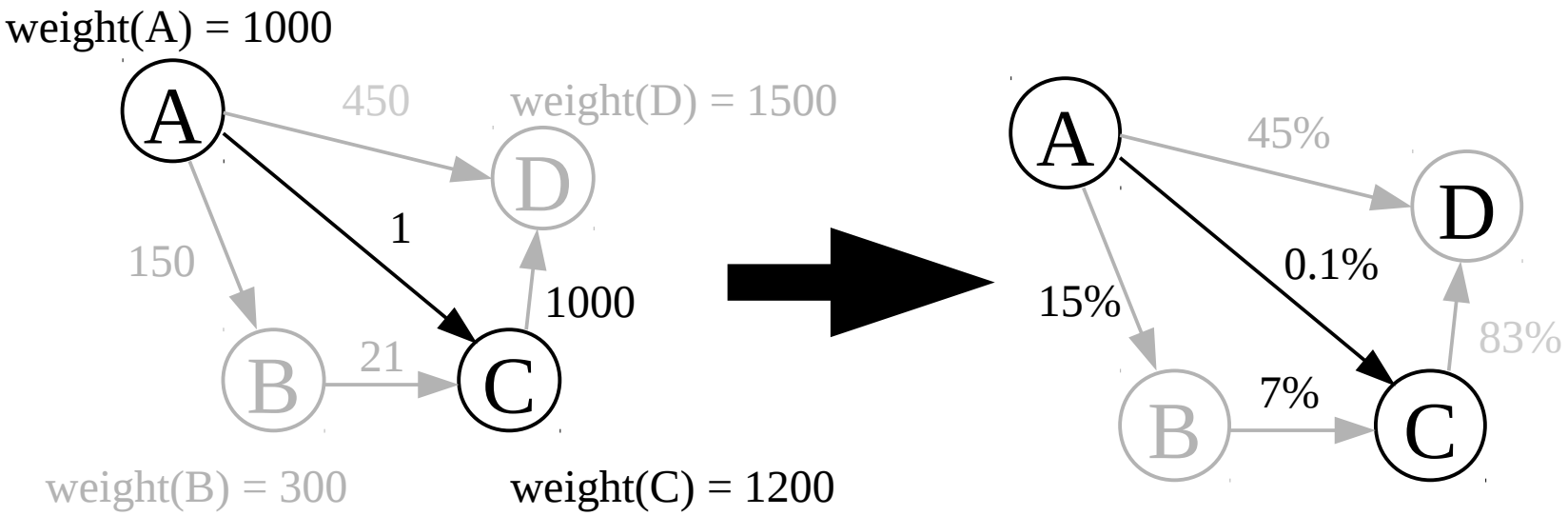
Furthermore, consider C → D: The confidence is very high. Would an additional 100 nested flows result in any more confidence?



Ratio Ranking (cont)

Computer Science
NC STATE UNIVERSITY

On the other hand, seeing 100 nested flows from A to C would be be substantially more convincing, yet the confidence would grow at roughly the same rate



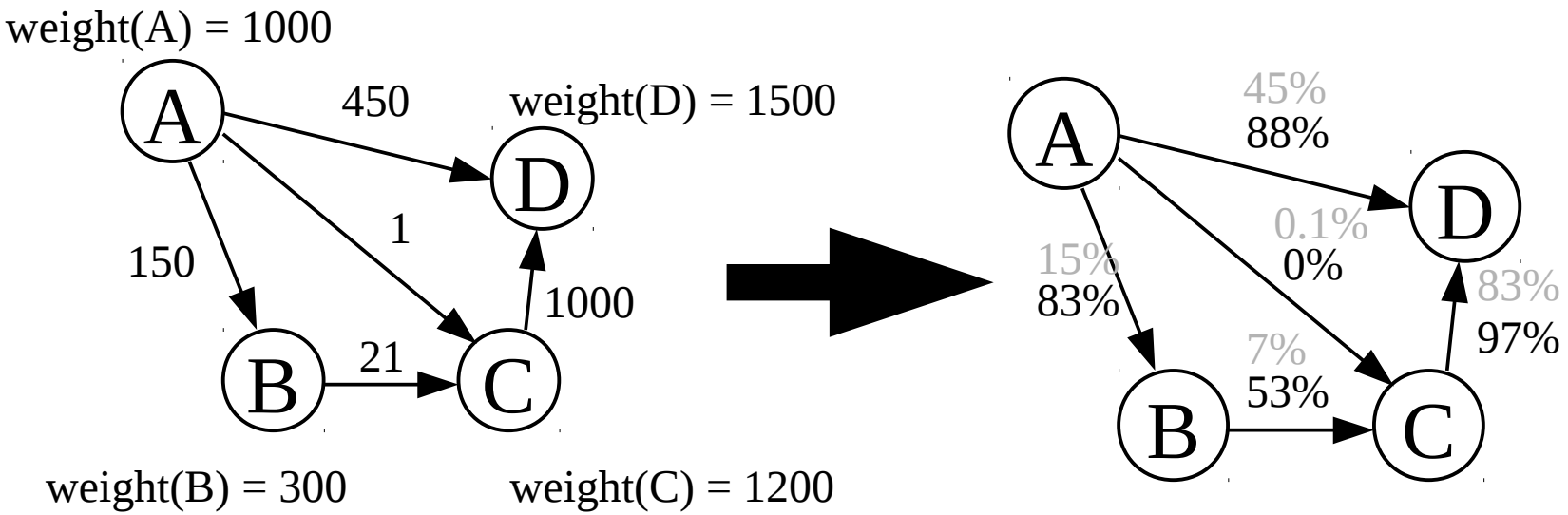
Logarithmic Ranking

Computer Science
NC STATE UNIVERSITY

- Three issues with Ratio Ranking
  - Results in low, unintuitive confidence values
  - Having more traffic that exhibits the same behavior should give us more confidence
  - As more nested flows are observed, the rate of confidence growth should decrease
- Instead, set confidence = $log_{weight(node)} weight(edge)$
  - Equivalent to *log(weight(node))/log(weight(edge))*
- Puts more disparity between true/false positives

Logarithmic Ranking (cont.)

- Logarithm implicitly gives higher values to the same ratio with greater magnitude
- Logarithm also grows quickly before leveling off



Service Inference

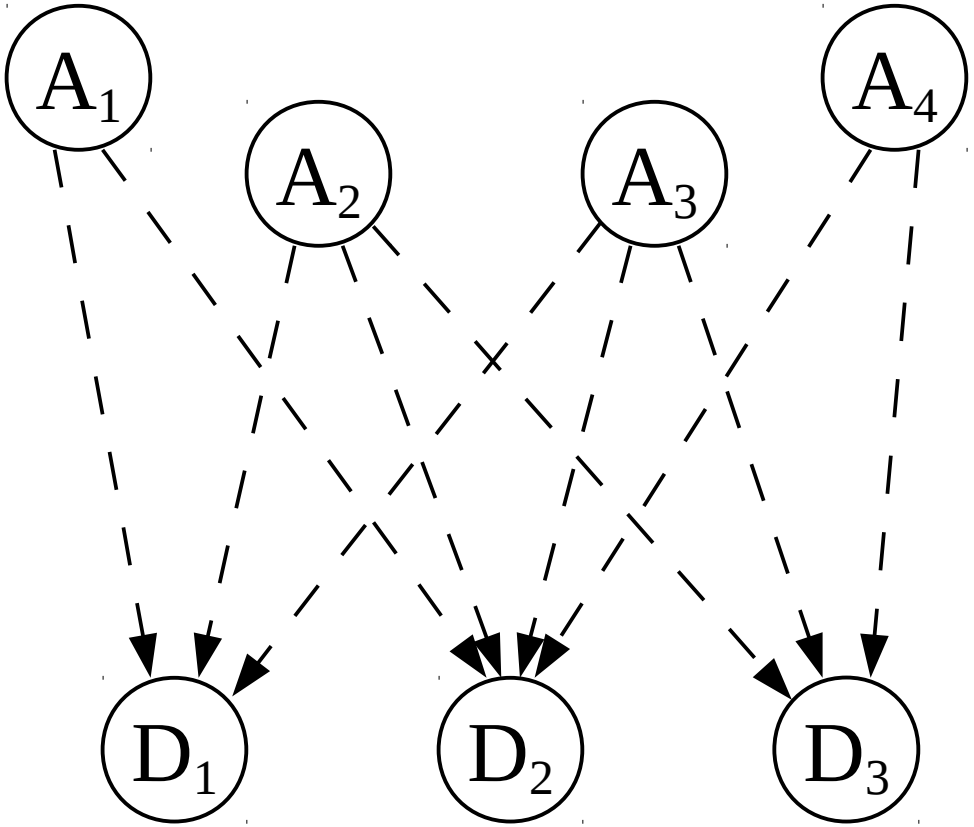
Computer Science
NC STATE UNIVERSITY

- In many networks, there are several hosts offering different instances of the same service
- Some of these services may be accessed more frequently than others
- If we can identify that two instances of services are the same type, we can infer the dependencies of a rarely used service from one that's used often
- We determine similarity using the overlap in their depended services

Service Inference (example)

Computer Science
NC STATE UNIVERSITY

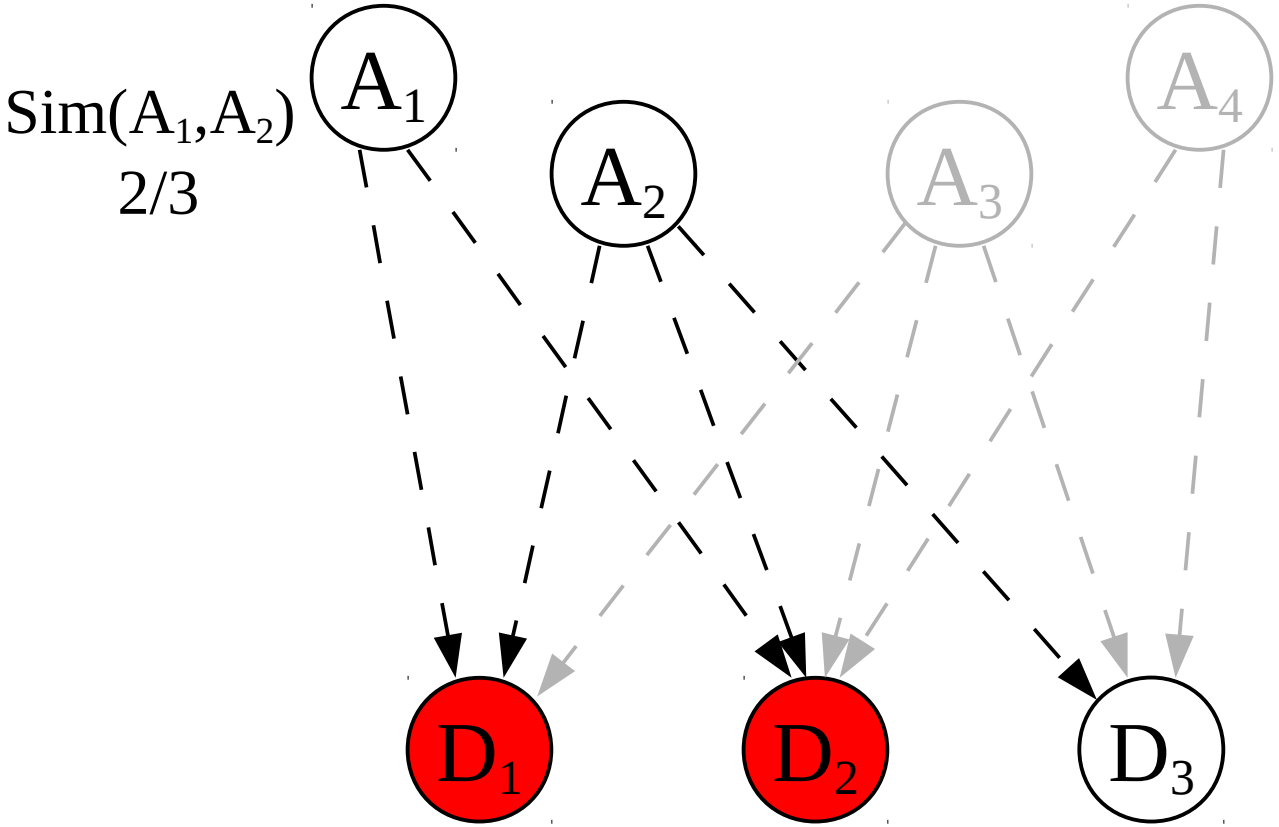
We begin with a communication graph. In this case, weights are not relevant.



Service Inference (example)

Computer Science
NC STATE UNIVERSITY

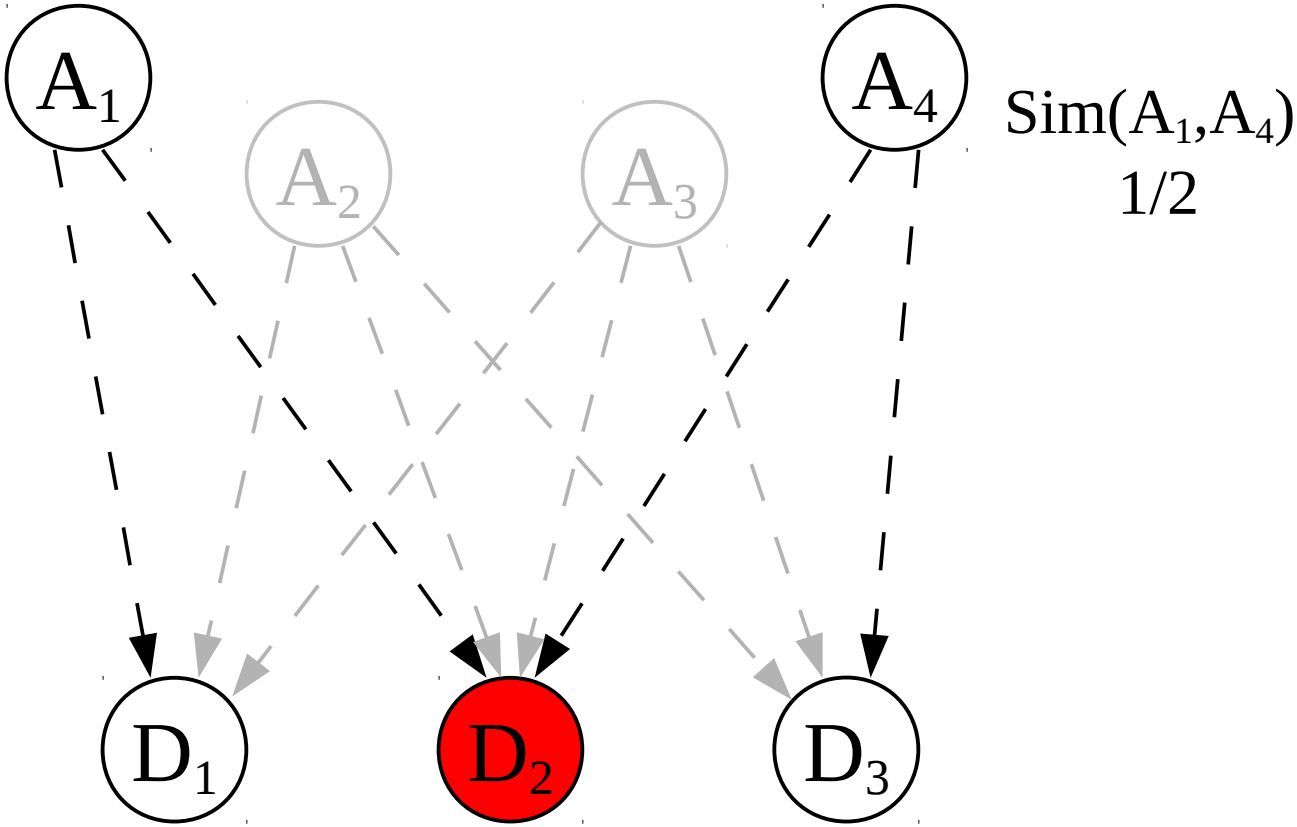
The similarity of two services is defined as the number of common depended services over number of dependencies of the service with the greater number of dependencies.



Service Inference (example)

Computer Science
NC STATE UNIVERSITY

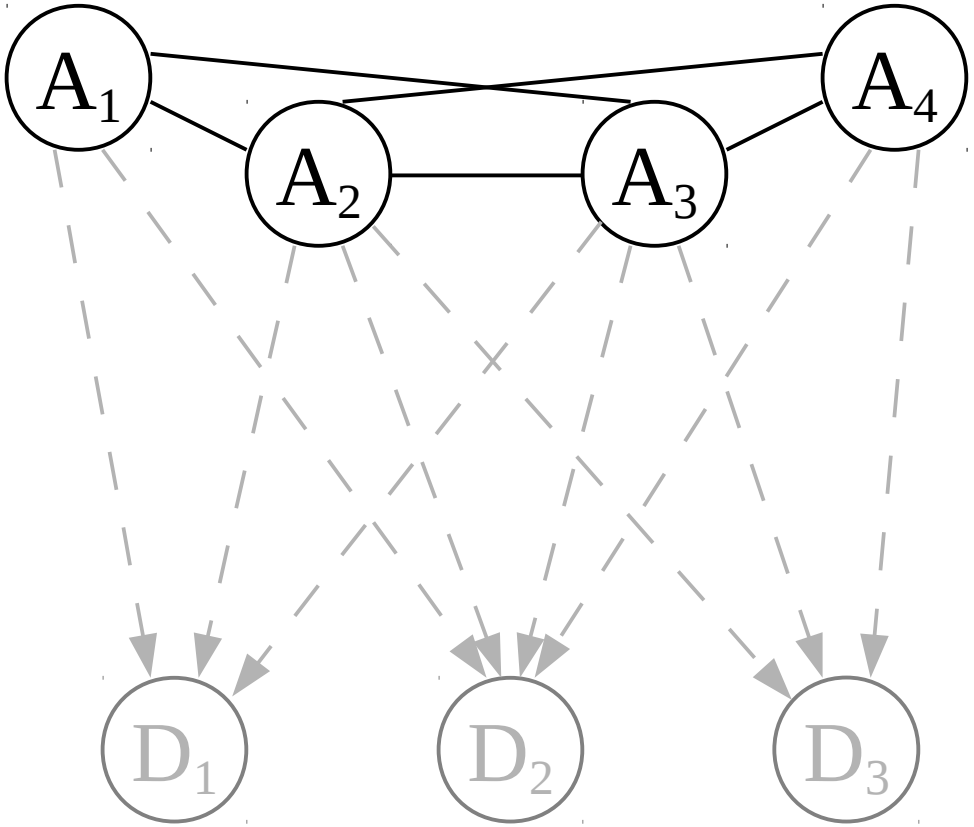
The similarity of two services is defined as the number of common depended services over number of dependencies of the service with the greater number of dependencies.



Service Inference (example)

Computer Science
NC STATE UNIVERSITY

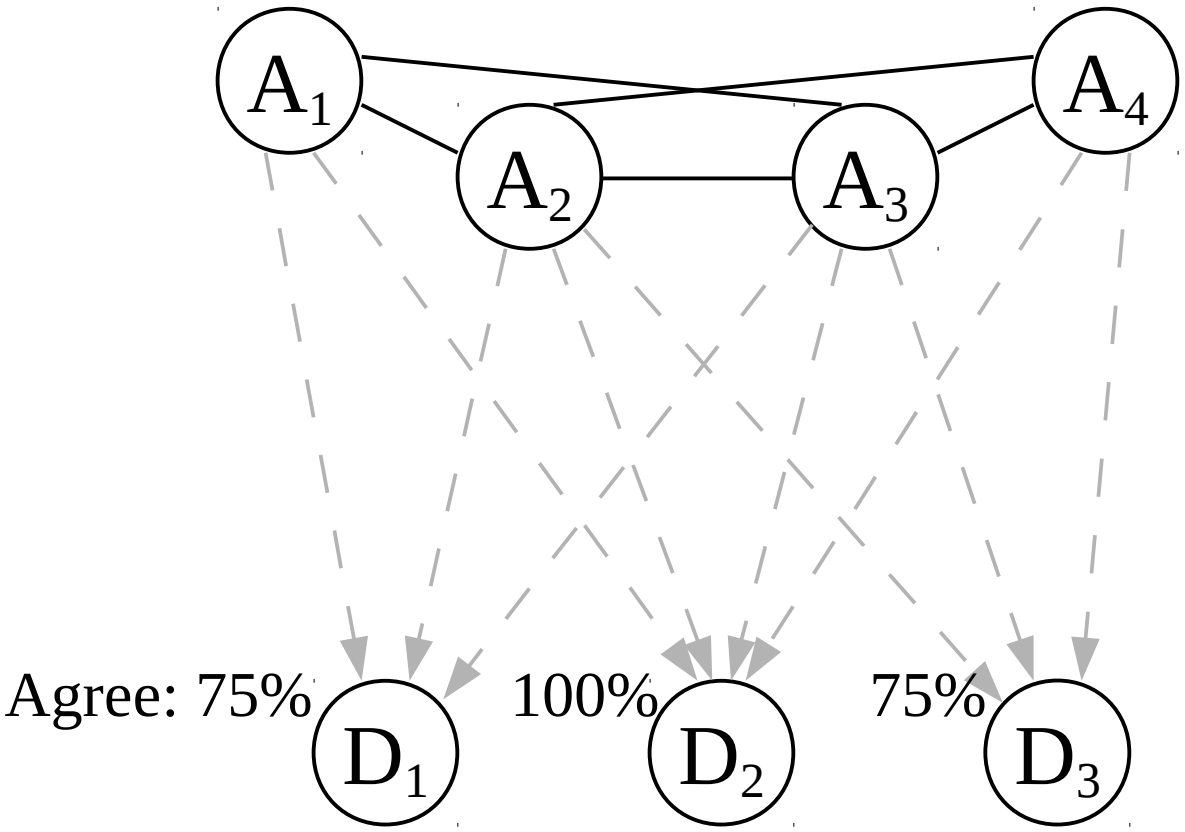
We draw edges between all similar services. Each connected subgraph is considered a similar “group”.



Service Inference (example)

Computer Science
NC STATE UNIVERSITY

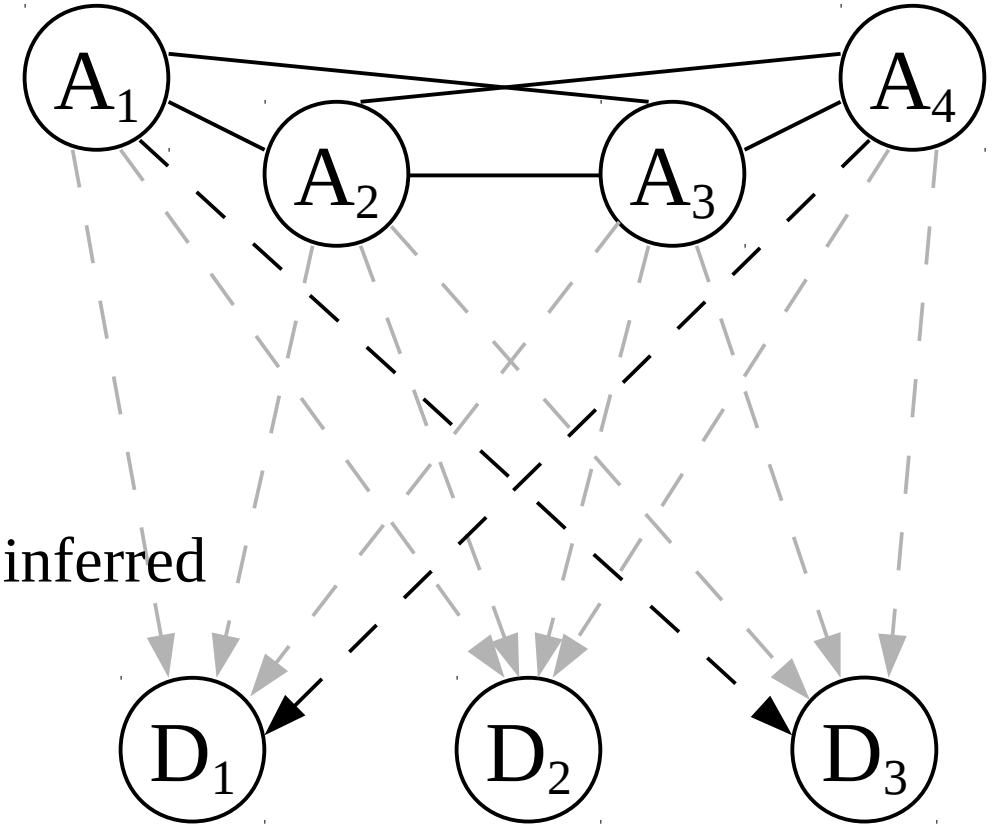
The agreement of a service is equal to the fraction of members of the group that depend on it.



Service Inference (example)

Computer Science
NC STATE UNIVERSITY

Any service in the group that does not depend on an agreed depended service has that dependency inferred.



$A_4 \rightarrow D_1$ and
$D_1 \rightarrow D_3$ are inferred

Service Inference (cont)

Computer Science
NC STATE UNIVERSITY

- This approach is configured with two thresholds:
- The similarity threshold
  - Defines how large groups of similar services become
- The agreement threshold
  - Defines how many dependencies are inferred from these groups
- In networks with lots of redundancy, high thresholds can still yield usable results

Cluster Discovery

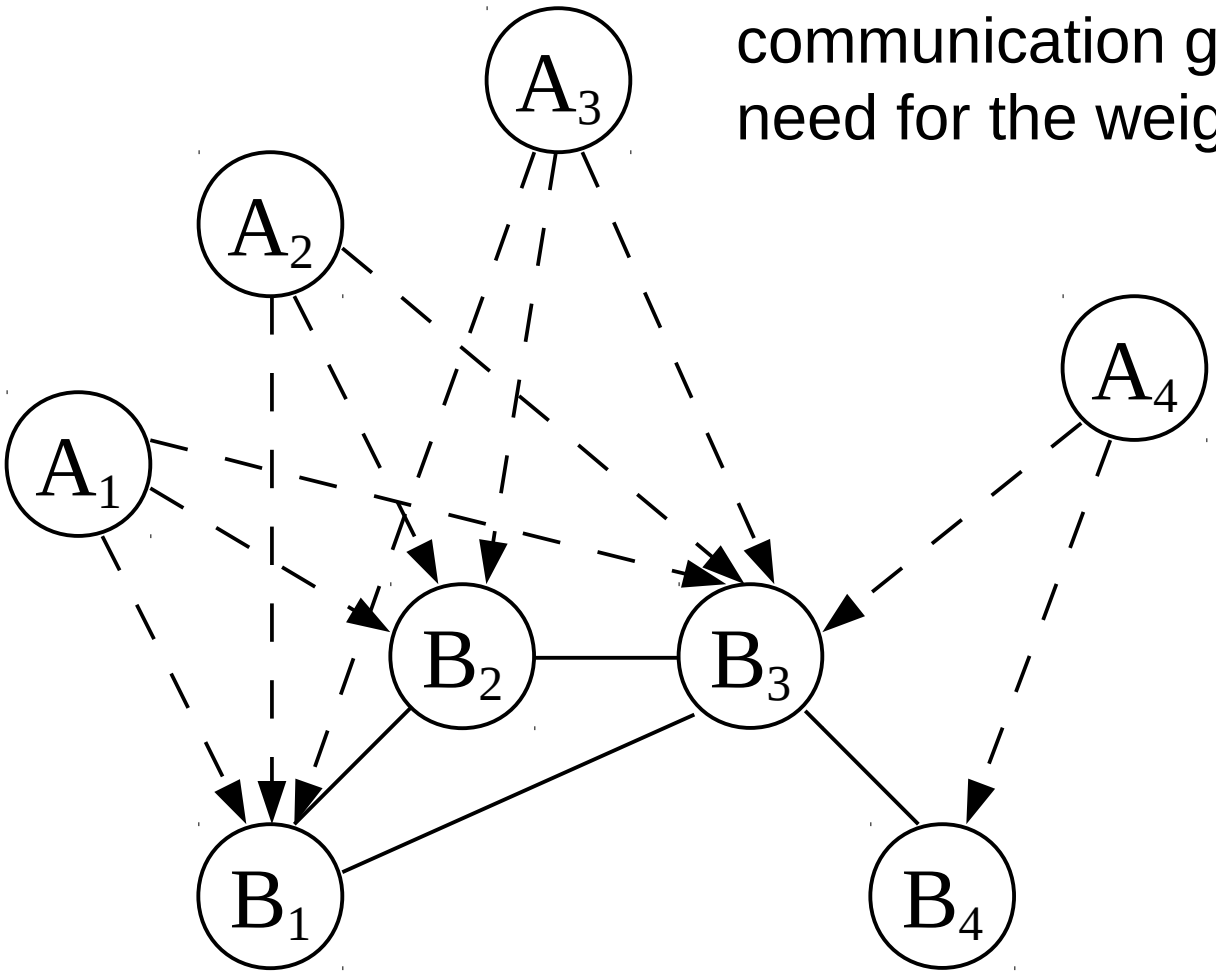
Computer Science
NC STATE UNIVERSITY

- Network services often rely on redundant service clusters
  - Nodes in load-balancing service cluster can be used interchangeably by depending services
  - Nodes in back-up clusters will replace primary nodes in the event that they fail
- In both cases, the services in a cluster will appear in a list of depended services together
- We use this observation to drive our approach to discovering service clusters

Cluster Discovery (example)

Computer Science
NC STATE UNIVERSITY

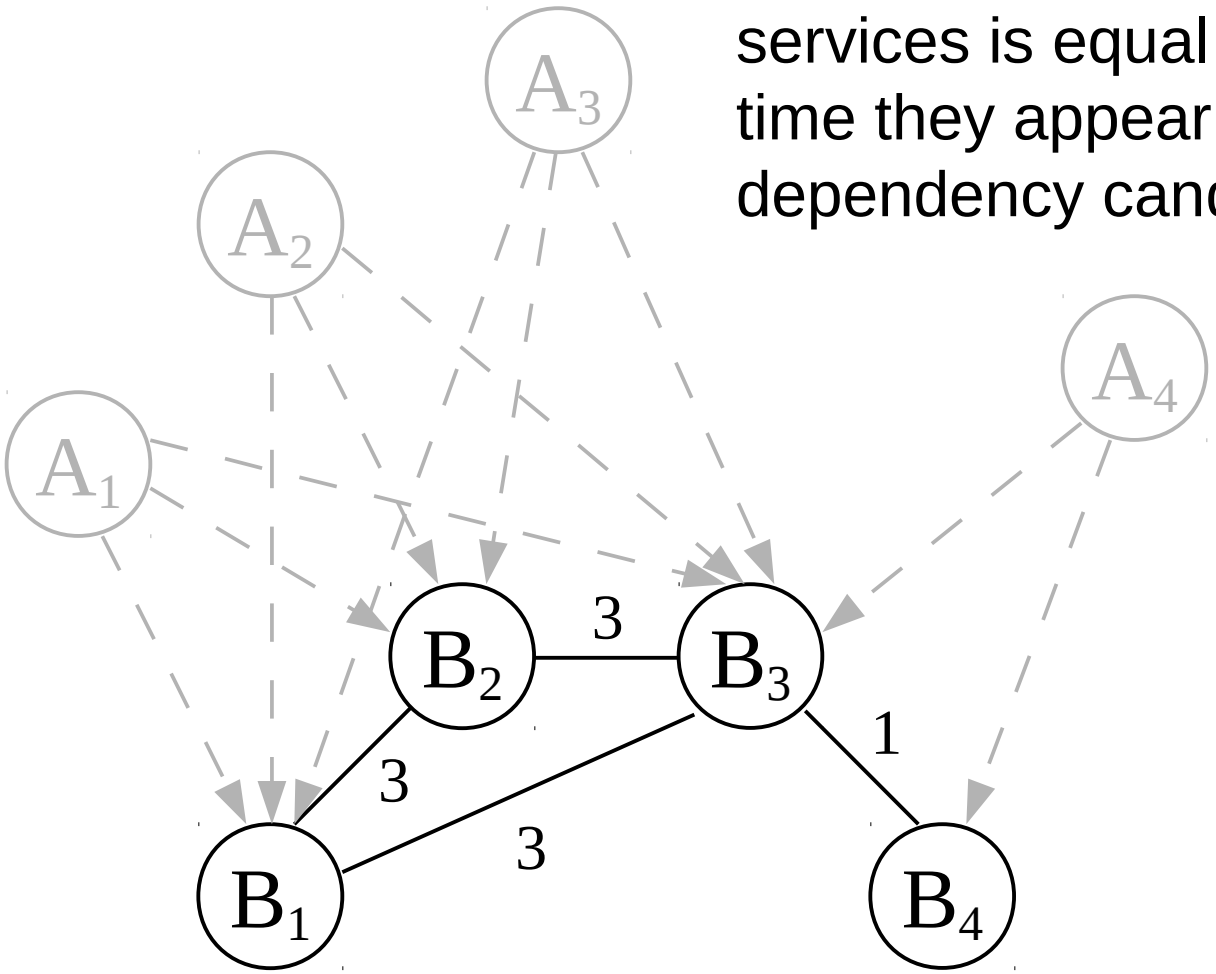
Again, we begin with a communication graph without any need for the weights.



Cluster Discovery (example)

Computer Science
NC STATE UNIVERSITY

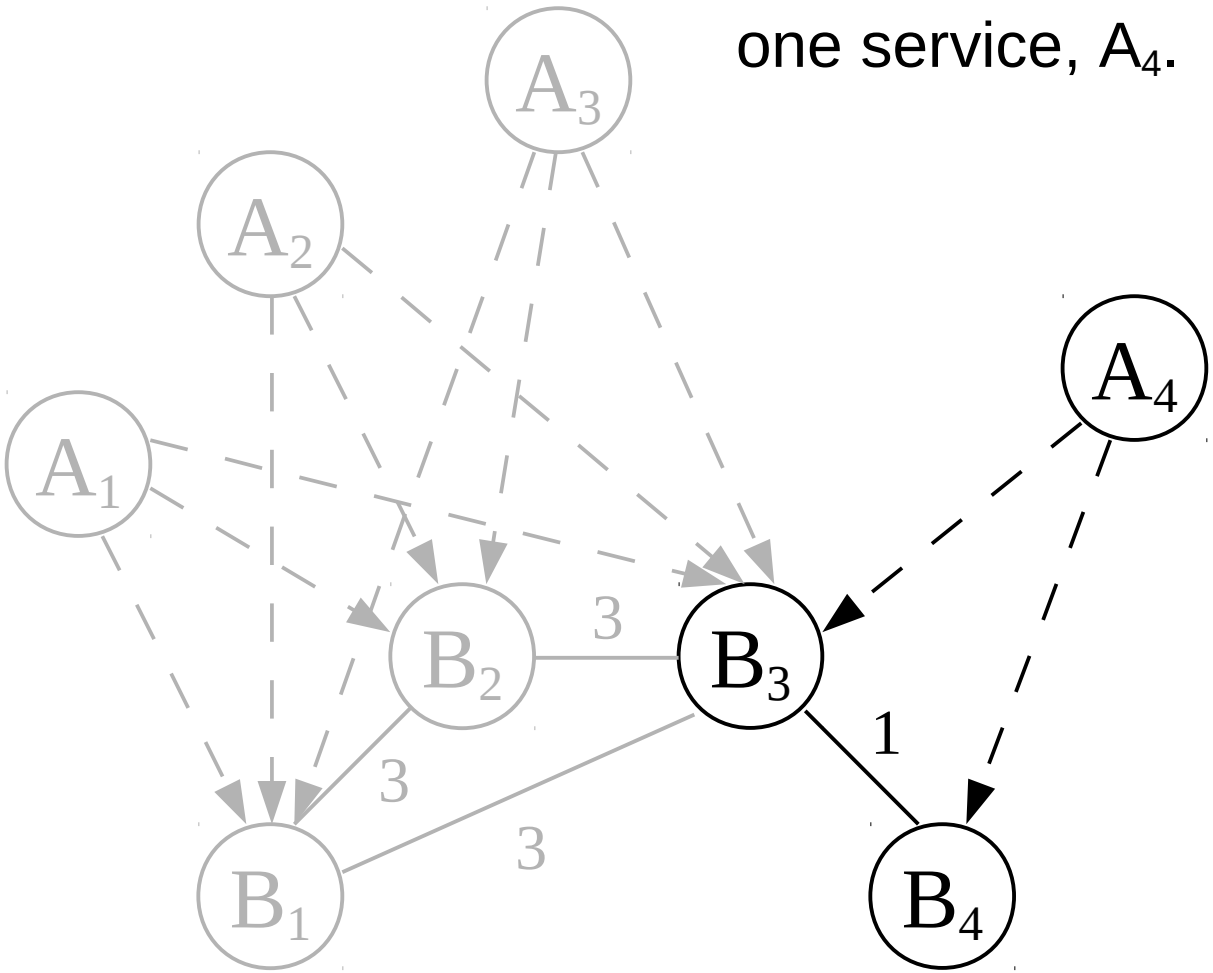
The support of a pair of depended services is equal to the number of time they appear together as dependency candidates.



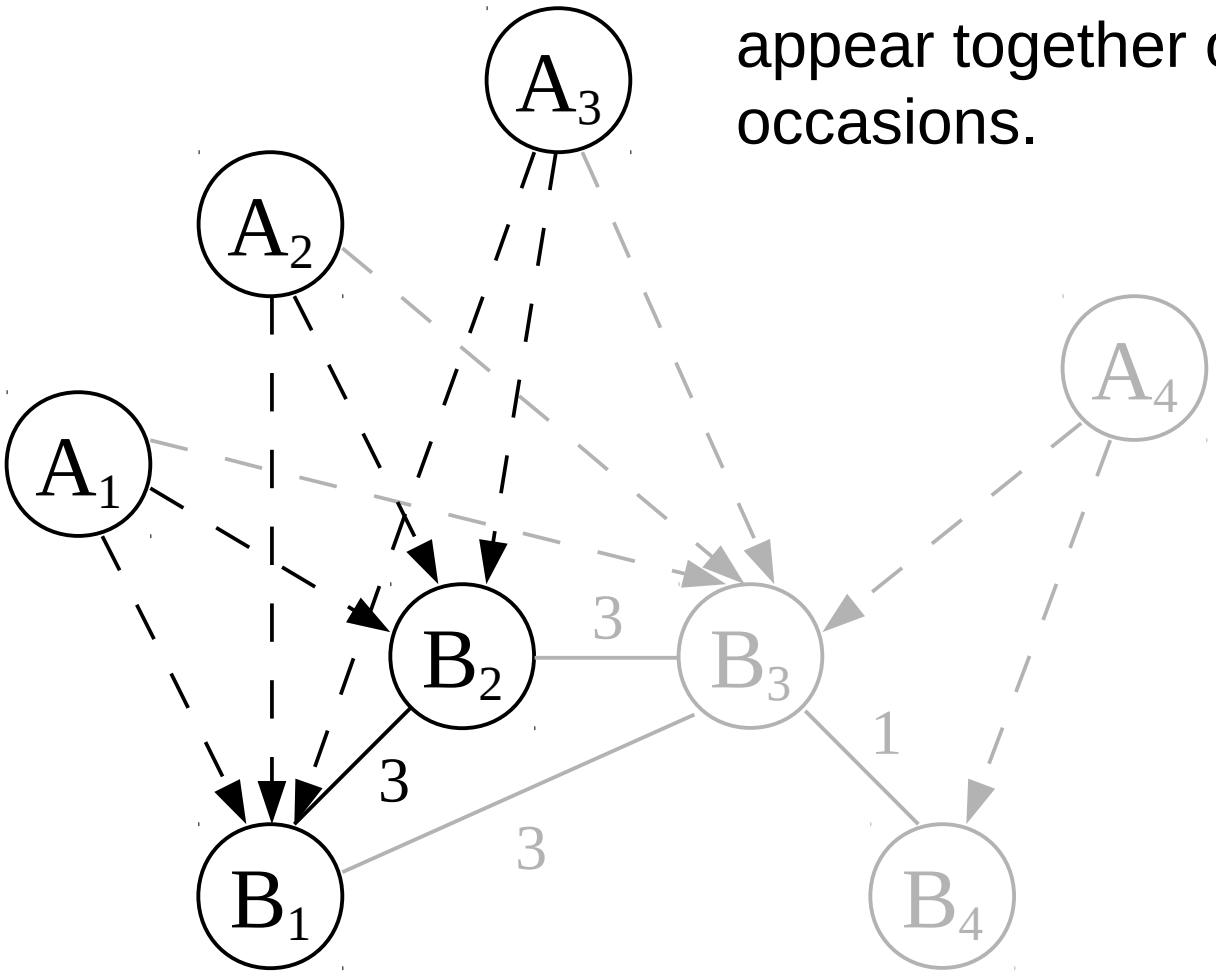
Cluster Discovery (example)

Computer Science
NC STATE UNIVERSITY

B_3 and B_4 only appear together for one service, A_4.



Cluster Discovery (example)

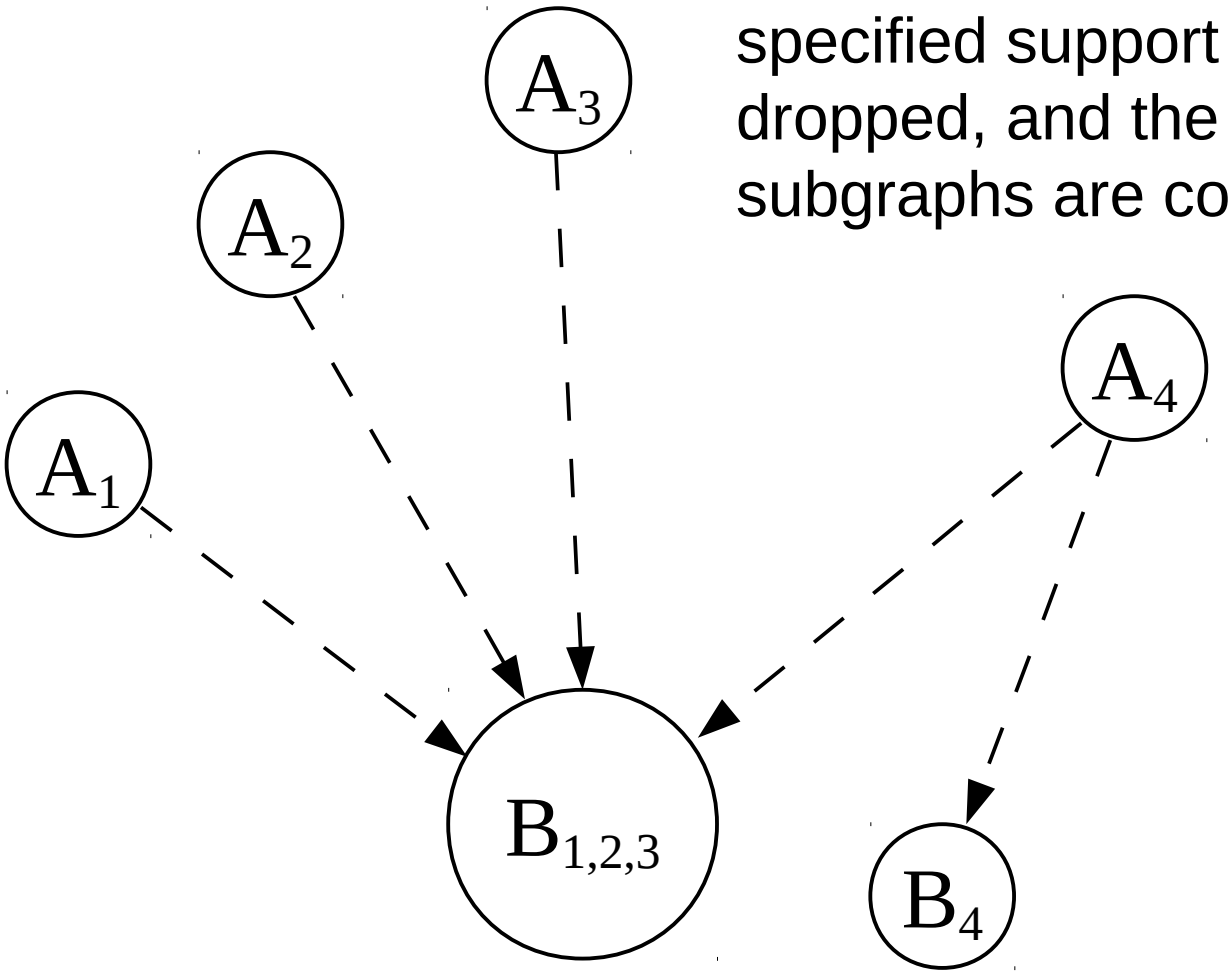


On the other hand, B_1 and B_2 appear together on three separate occasions.

Cluster Discovery (example)

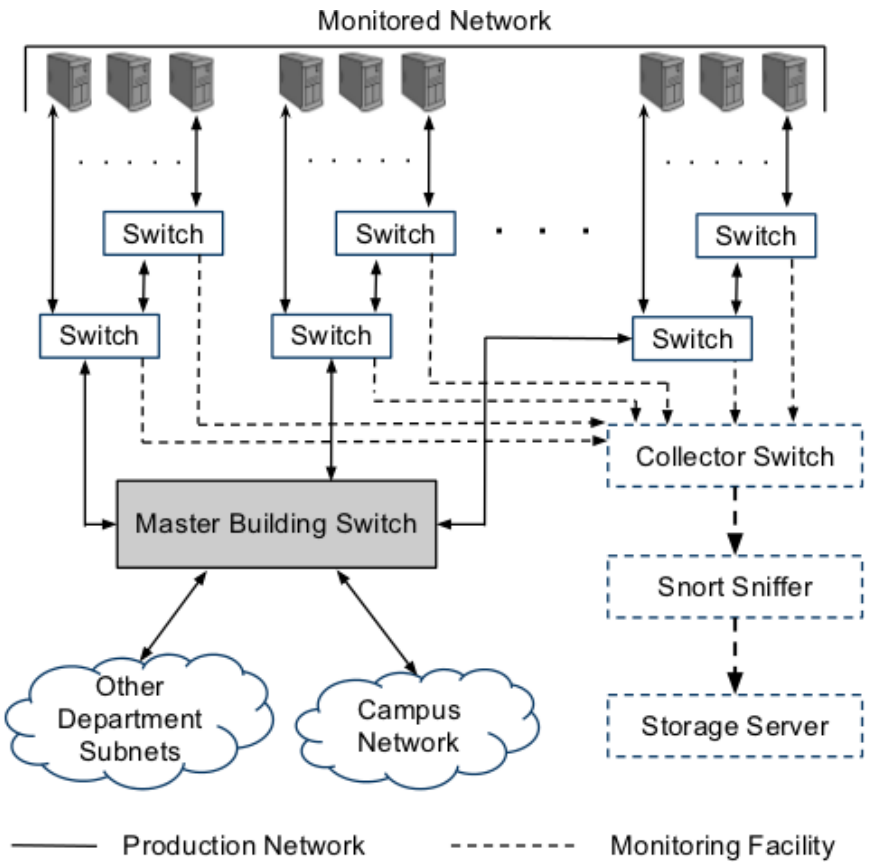
Computer Science
NC STATE UNIVERSITY

Any edges with less than a user-specified support threshold are dropped, and the connected subgraphs are considered clusters.



Evaluation

- We evaluate on the same dataset as NSDMiner's 2011 iteration
- Collected from the traffic on the second floor of EB2's CSC wing
- Ground truth built with help from CSC IT



Logarithmic Ranking

Computer Science
NC STATE UNIVERSITY

- Logarithmic ranking was an improvement, especially for infrequently used services

	Shared Mode						Exclusive Mode					
	Ratio (0.5%)			Log (45%)			Ratio (0.5%)			Log (44%)		
Service	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
webservice	8	9	1	9	12	0	8	4	1	8	7	0
email	3	1	0	3	3	0	3	0	0	3	2	0
ssh(realm-4)	10	1	0	10	1	0	10	1	0	10	1	0
ssh(realm-5)	41	14	10	43	12	8	40	10	11	43	9	8
svn	4	1	0	4	0	0	4	1	0	4	0	0
proxy DHCP	2	1	0	2	0	0	2	1	0	2	0	0
DHCP	1	6	0	1	1	0	1	6	0	1	1	0
endpoint mapper	6	43	0	6	4	0	3	6	3	3	1	3
WDS (RPC)	4	10	0	3	1	1	4	1	0	2	0	2
DFS replication (RPC)	8	4	0	6	3	2	7	0	1	5	1	3
SMB	9	7	1	9	9	1	9	3	1	9	5	1
TFTP	-	1	-	-	1	-	-	0	-	-	0	-
database	-	1	-	-	2	-	-	1	-	-	2	-
Invalid Services	-	34	-	-	29	-	-	30	-	-	17	-
Total	96	133	12	96	78	12	91	64	17	90	46	18

Logarithmic Ranking

Computer Science
NC STATE UNIVERSITY

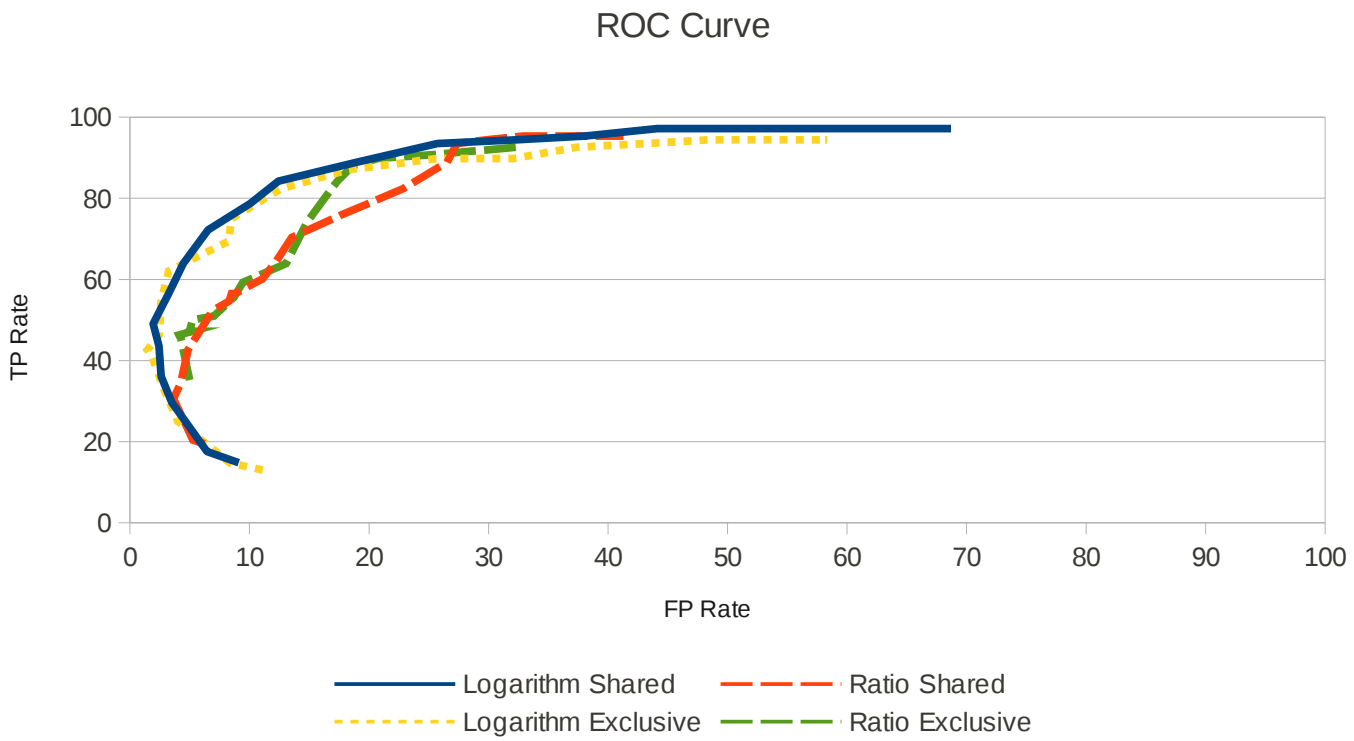
- The greatest improvements in False Positives came from services that were infrequently used.

	Shared Mode						Exclusive Mode					
	Ratio (0.5%)			Log (45%)			Ratio (0.5%)			Log (44%)		
Service	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
webservice	8	9	1	9	12	0	8	4	1	8	7	0
email	3	1	0	3	3	0	3	0	0	3	2	0
ssh(realm-4)	10	1	0	10	1	0	10	1	0	10	1	0
ssh(realm-5)	41	14	10	43	12	8	40	10	11	43	9	8
svn	4	1	0	4	0	0	4	1	0	4	0	0
proxy DHCP	2	1	0	2	0	0	2	1	0	2	0	0
DHCP	1	6	0	1	1	0	1	6	0	1	1	0
endpoint mapper	6	43	0	6	4	0	3	6	3	3	1	3
WDS (RPC)	4	10	0	3	1	1	4	1	0	2	0	2
DFS replication (RPC)	8	4	0	6	3	2	7	0	1	5	1	3
SMB	9	7	1	9	9	1	9	3	1	9	5	1
TFTP	-	1	-	-	1	-	-	0	-	-	0	-
database	-	1	-	-	2	-	-	1	-	-	2	-
Invalid Services	-	34	-	-	29	-	-	30	-	-	17	-
Total	96	133	12	96	78	12	91	64	17	90	46	18

Logarithmic Ranking (cont)

Computer Science
NC STATE UNIVERSITY

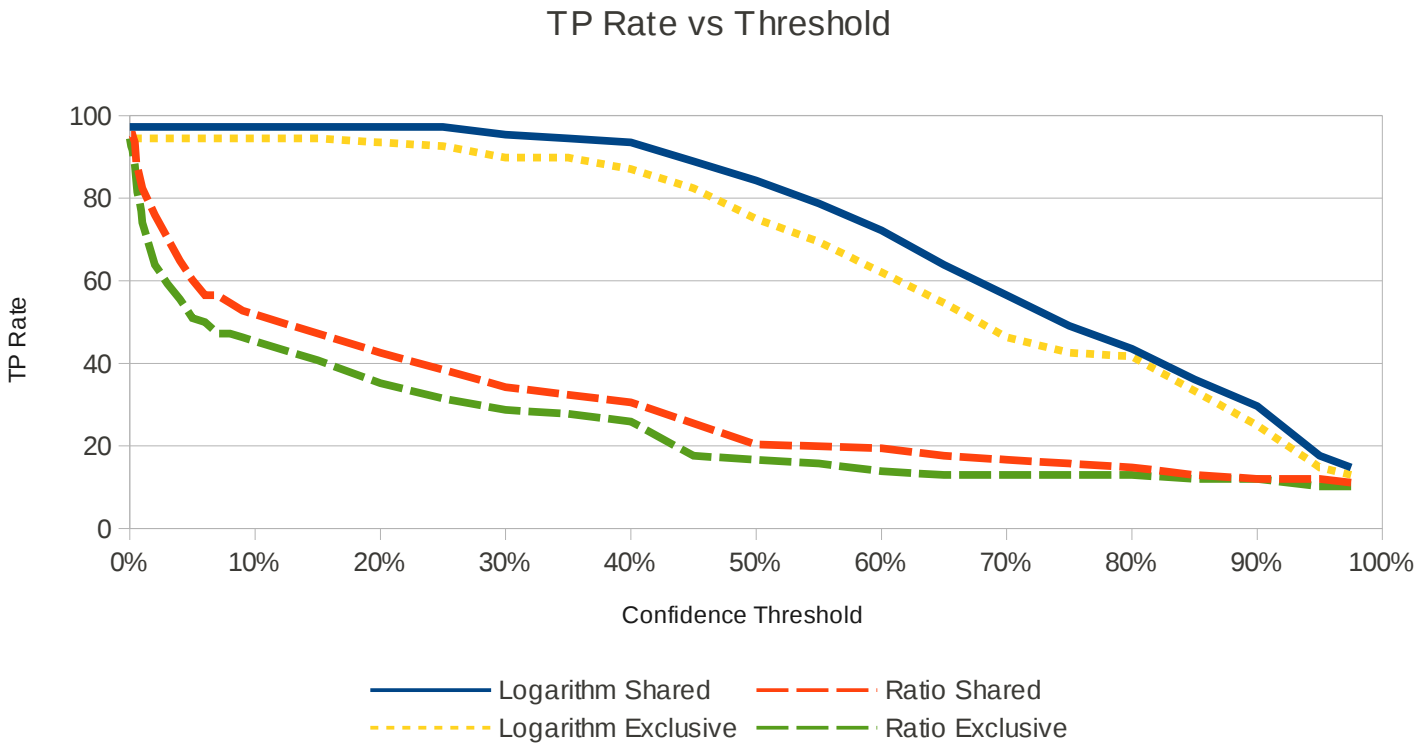
- The Receiver-Operator Characteristic curve shows the overall improvement of log ranking



Logarithmic Ranking (cont)

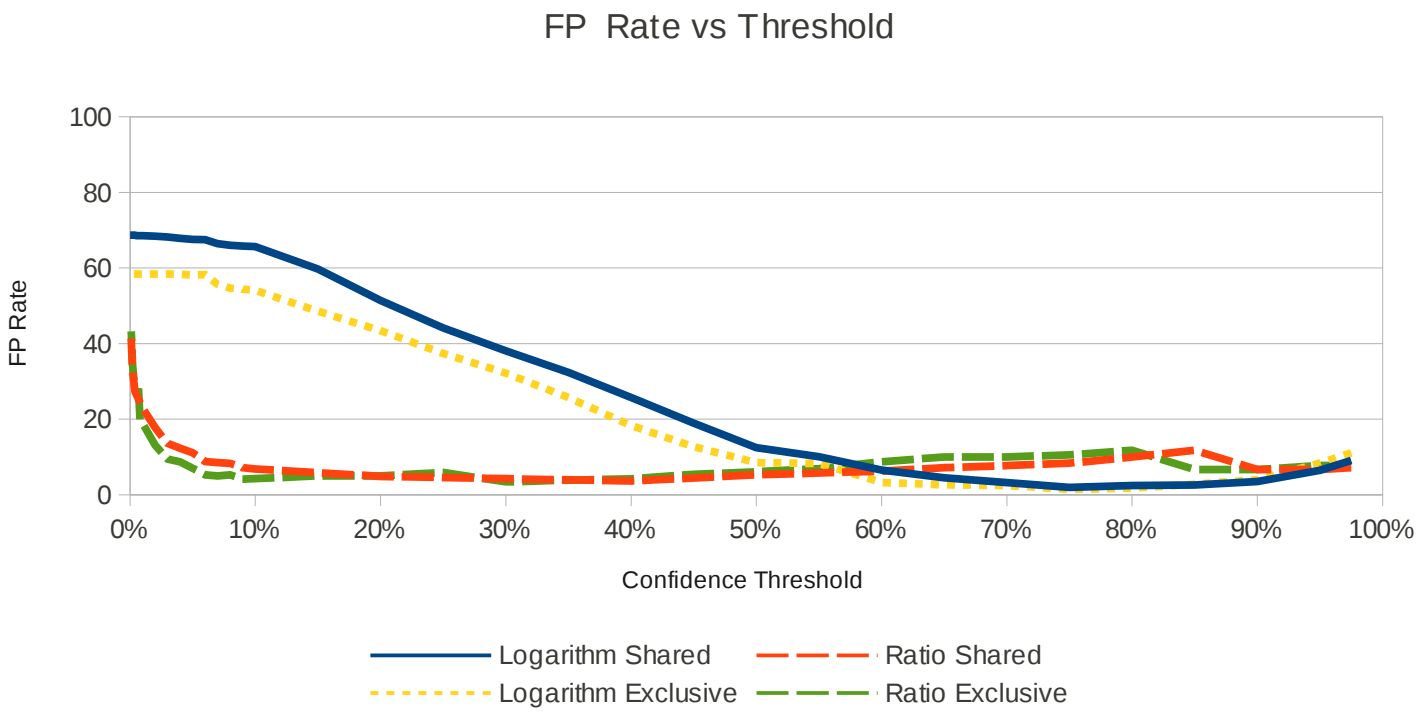
Computer Science
NC STATE UNIVERSITY

- Note that the number of true positives drops gradually as the confidence threshold increases



Logarithmic Ranking (cont)

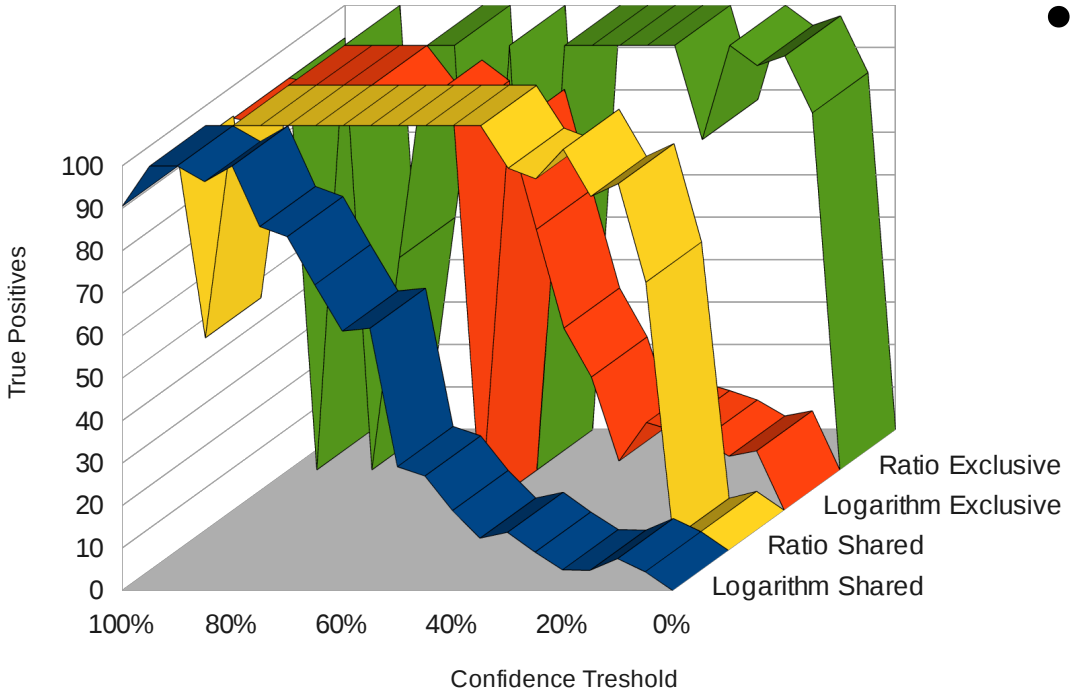
- ... as does the False Positive rate



Logarithmic Ranking (cont)

Computer Science
NC STATE UNIVERSITY

- In logarithmic ranking, the confidence value accurately predicts whether a dependency is true

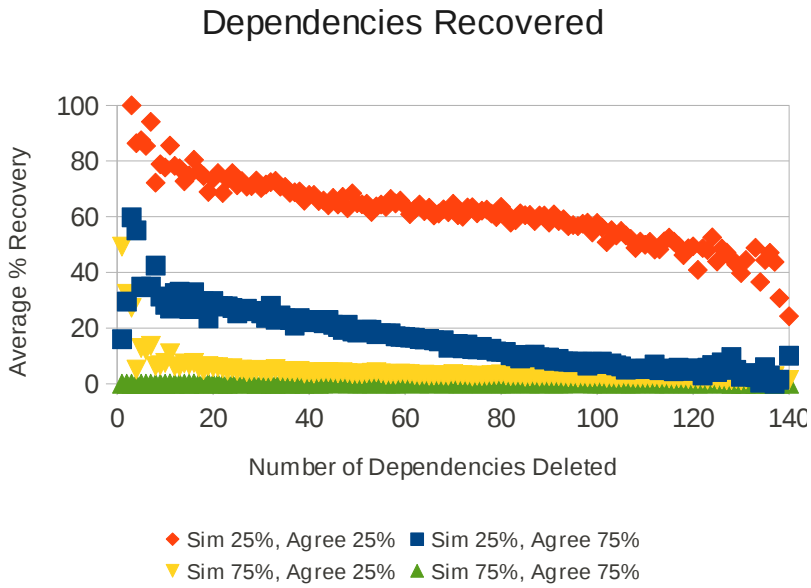


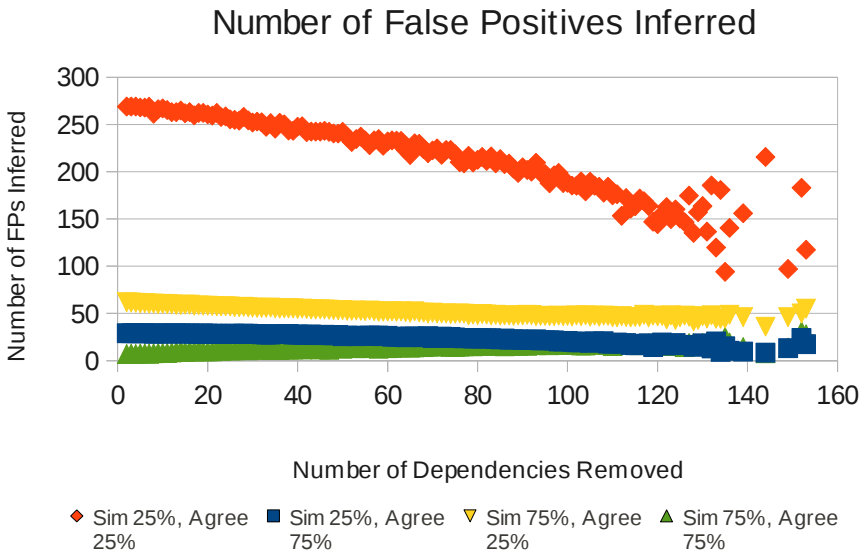
- This graph shows the candidates divided into bins based on their confidence.

Service Inference

Computer Science
NC STATE UNIVERSITY

- We removed flows related to some dependencies from the ground truth and tried to re-infer them
- Note that while False positives were inferred, they still had low confidence values (<30%)

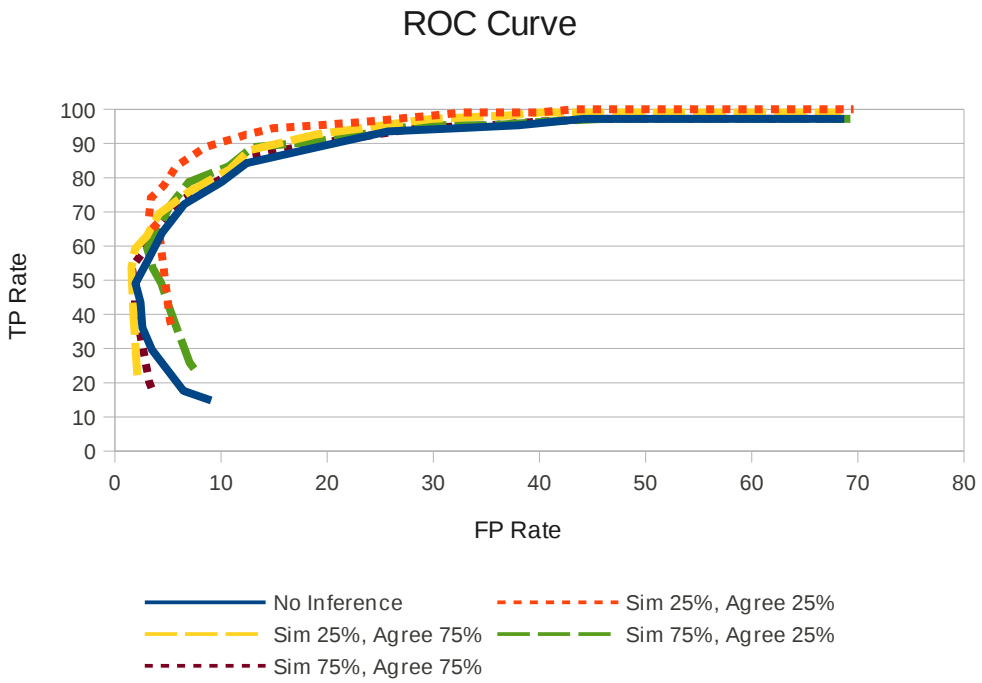




Service Inference (cont)

Computer Science
NC STATE UNIVERSITY

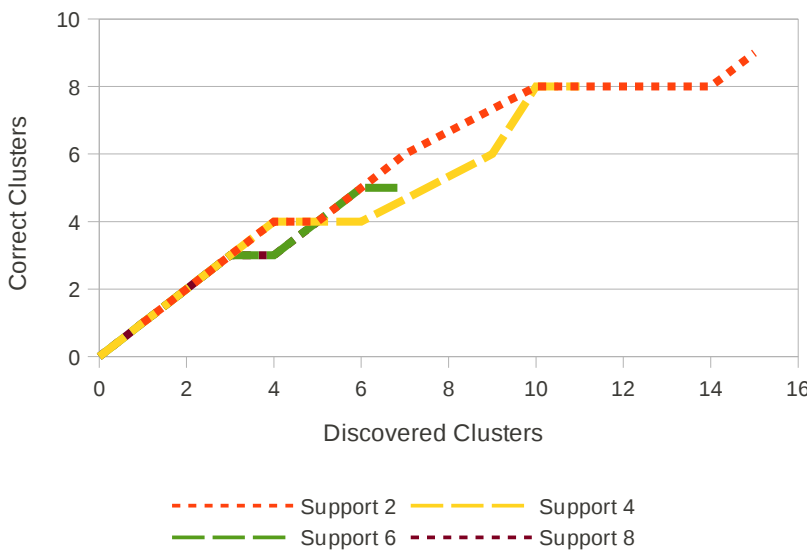
- This shows the ROC curve for various service inference parameters compared to log ranking
- Note that we can achieve 100% coverage!

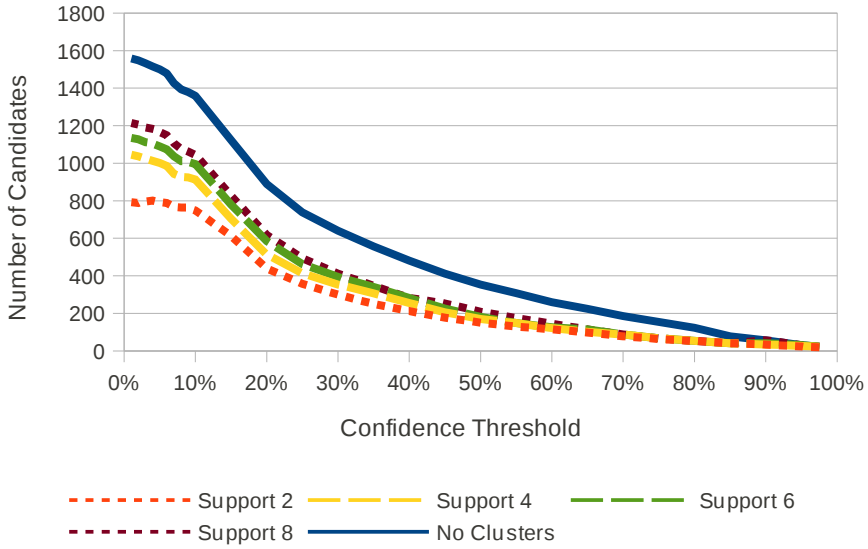


Cluster Discovery

Computer Science
NC STATE UNIVERSITY

- We accurately detect our network's ten clusters
- By aggregating them, we can reduce the size of the output substantially





In Conclusion...

Computer Science
NC STATE UNIVERSITY

- We introduce three novel approaches for the detection of network service dependencies
- We implement and test our approaches on production network traffic
  - Our approach improves false positives and negatives on top of NSDMiner's original improvements

Future Work

Computer Science
NC STATE UNIVERSITY

- Identification of remote-remote dependencies
- Ability to monitor dependencies in real time
- Collection of a newer, larger dataset (first and third floor)

Acknowledgments

- The CSC IT Staff for their support in maintaining our collection infrastructure and discovery of our ground truth
- Arun Natarajan, for helping troubleshoot NSDMiner (even after his graduation)
- This work is supported by the U.S. Army Research Office (ARO) under MURI grant W911NF-09-1-0525

Thank you for your attention!

Computer Science
NC STATE UNIVERSITY