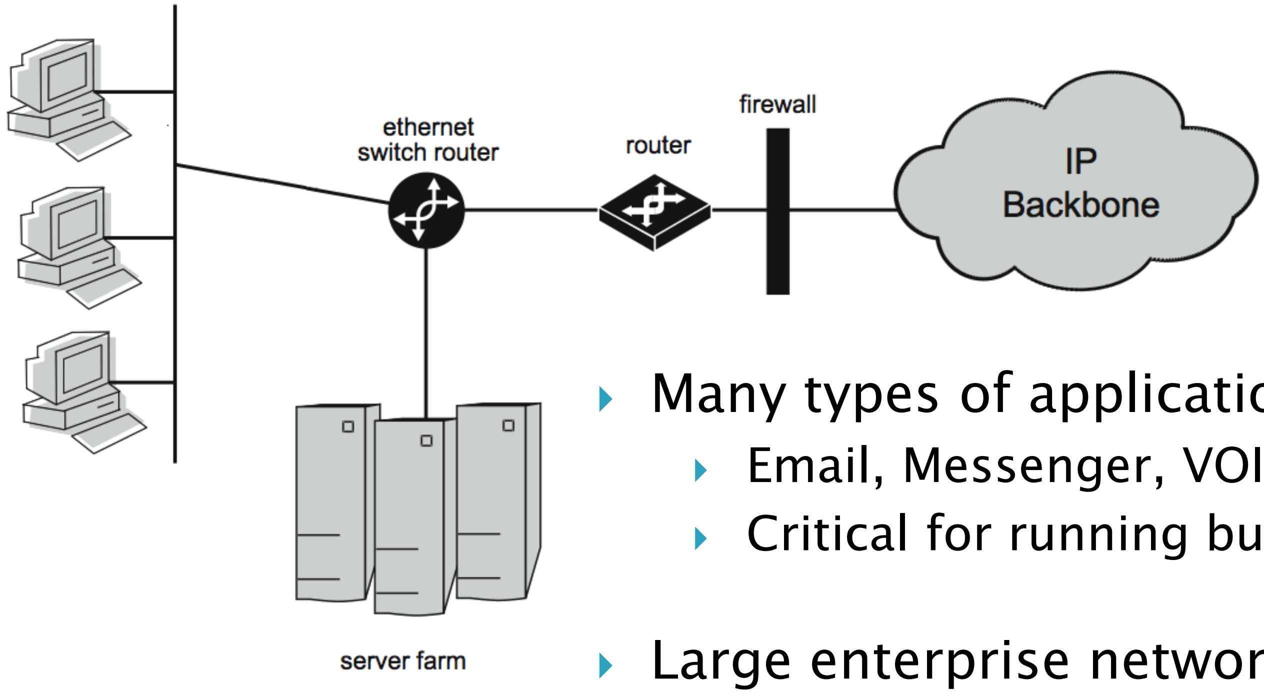
Automating Network Application Dependency Discovery: Experiences, Limitations, and New Solutions

Ming Zhang, Microsoft Research

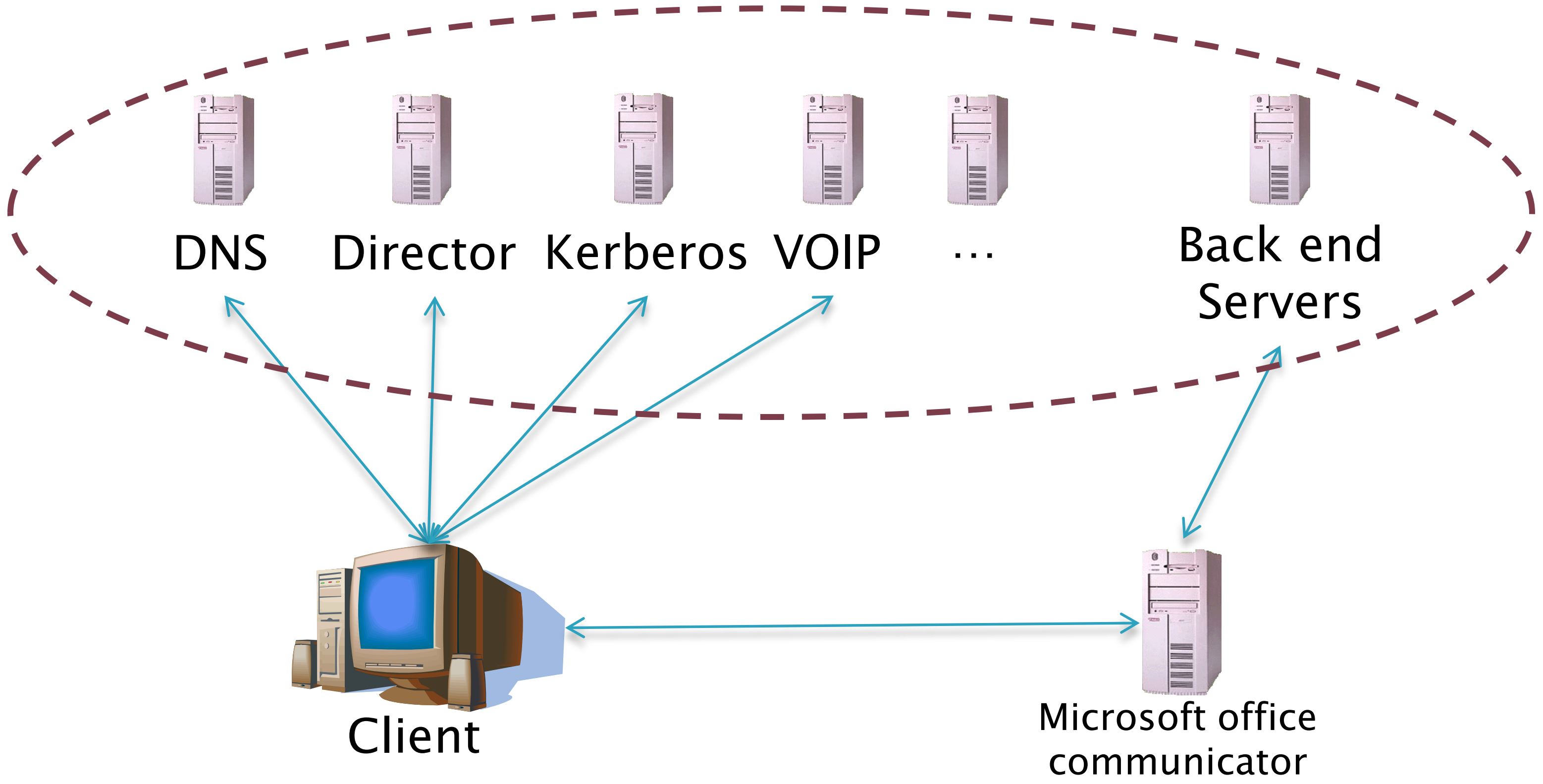
Joint work with Xu Chen, Z. Morley Mao (UMich), Victor Bahl (Microsoft Research)

Enterprise network management is complicated



- Many types of applications
  - Email, Messenger, VOIP, etc.
  - Critical for running business
- Large enterprise networks
  - 1,000s network applications
  - 1,000s staffs in IT support
  - $$ millions of dollars spent every year

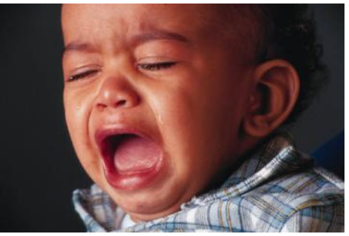
Why enterprise network management is complicated



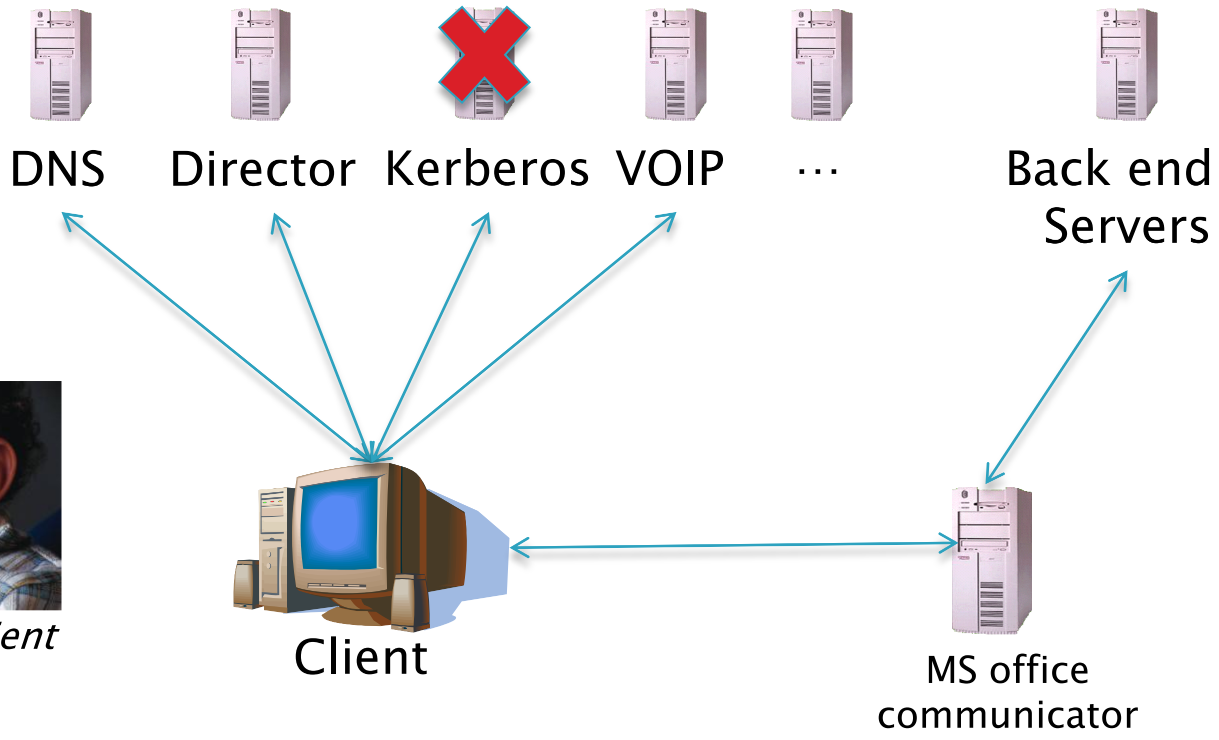
Challenges in discovering service dependency

- Heterogeneous applications
  - Functionality
  - Deployment setups
- Knowledge distributed across layers and locations
- Applications evolve continuously

Why service dependency is useful



Why my OC client doesn't work?



All OC servers are running fine…
Only some clients have this problem…

Current solutions

- Based on human knowledge
  - Expensive
  - Error-prone
  - Hard to keep up-to-date information

- We need automated solution for dependency extraction

Related work

- Co-occurrence based dependency discovery
  - Sherlock & eXpose [SIGCOMM'07 & 08]
- Execution causality path extraction
  - Project 5 [SOSP'03] & WAP5 [WWW'06]

Our contributions

- Introduce a new technique to discover dependencies based on *spike in delay distribution*
- Identify the limitations of dependency discovery based on temporal analysis
- Evaluate our technique on five dominant applications in Microsoft's enterprise network
- Significantly improve the accuracy of dependency discovery over prior work

Outline

- Overview
- Dependency discovery techniques
- Deployment & results
- Conclusion

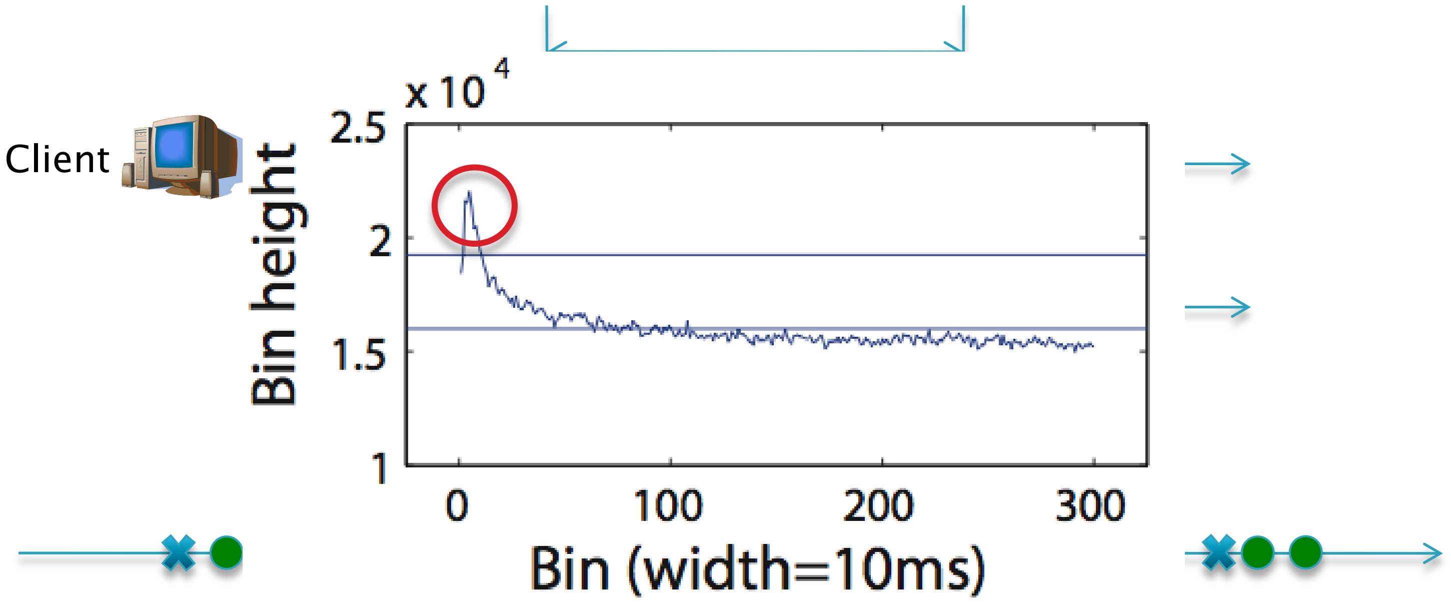
Design goals

- Applicable to variety of applications
  - Passive sniffing
  - Only parse into TCP/IP headers

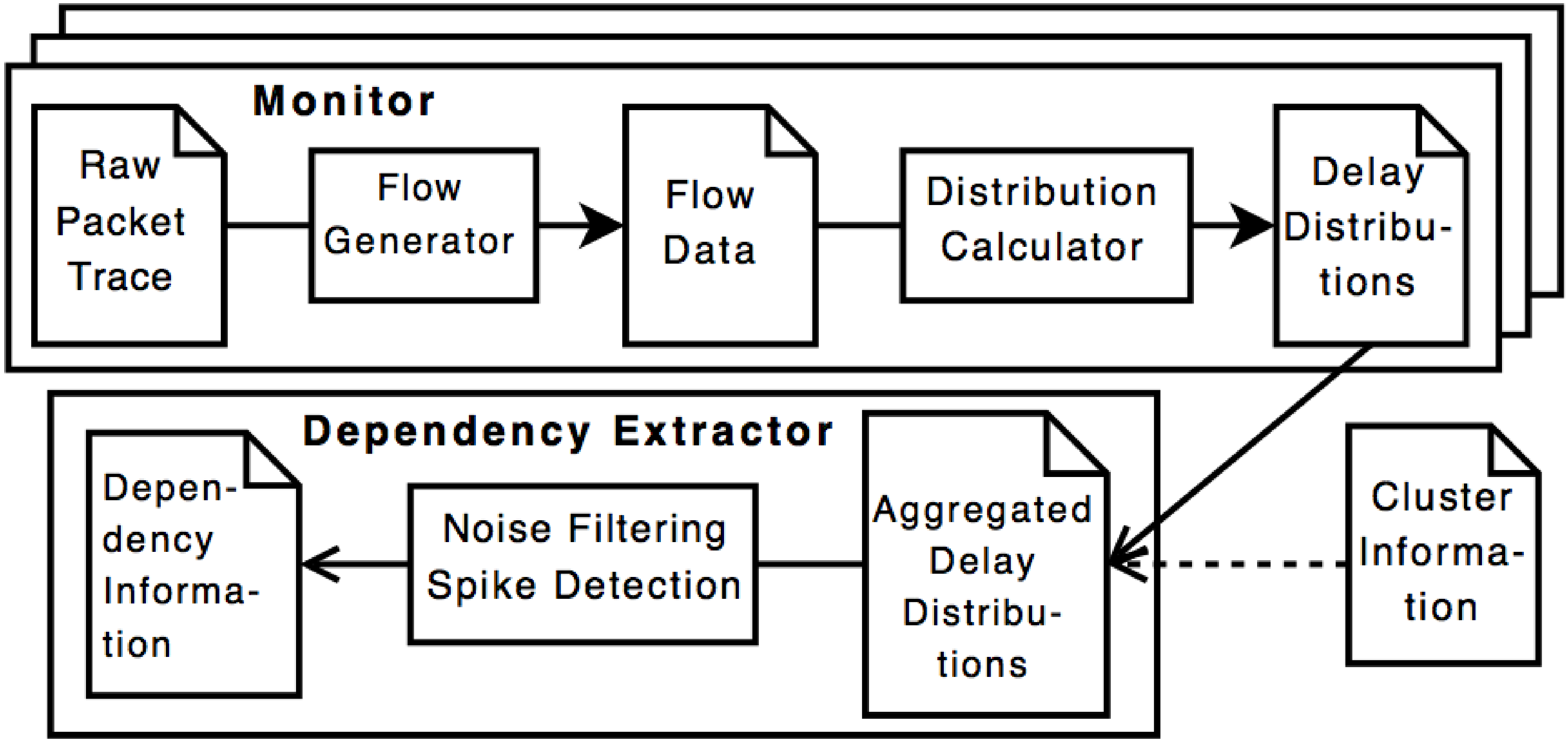
- Minimizing false dependencies without losing true dependencies
  - Hard to recover missing true dependencies
  - Minimize the effort to filter false dependencies

Orion

- Key idea: time delay between dependent services reflects typical processing and network delay



Dependency discovery process



Outline

- Overview
- Dependency discovery techniques
- Deployment & results
- Conclusion

Inferring application messages

- Problem: dependency exists between application messages

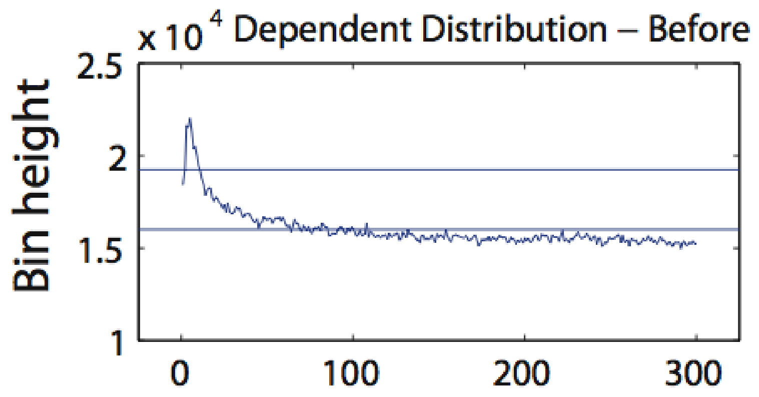
- Only rely on TCP/IP headers
  - Aggregate packets into flows

- Benefits
  - Reduce bias introduced by long flows
  - Reduce number of pairs

Dealing with scalability

- Service: (ip, port, proto)
- Problem:
  - Too many service pairs
- Solutions
  - Ignore transient "services"
  - Only consider service pairs that are close in time

Filtering noise in delay distributions



Overcoming a lack of samples

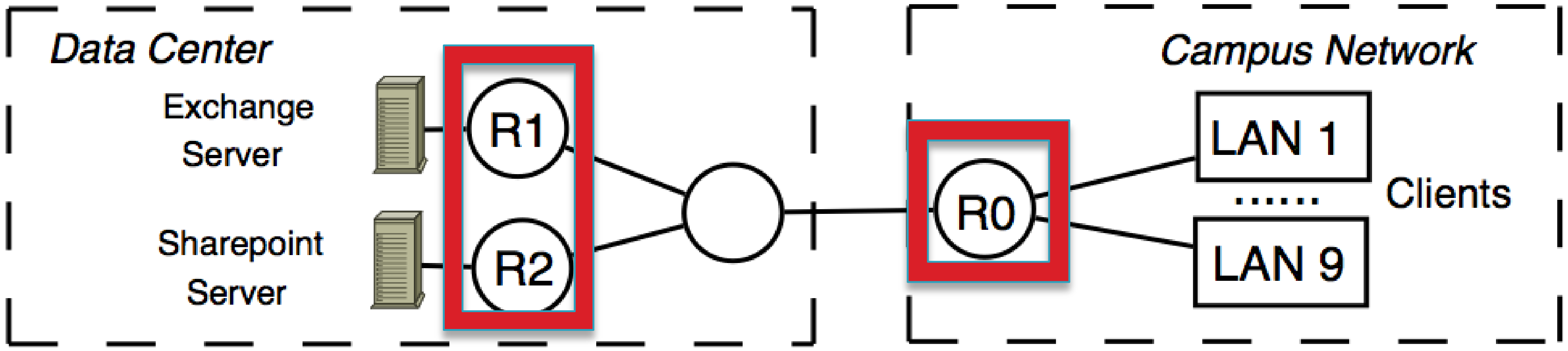
- Problem: Orion requires fair number of samples to infer dependency

- Solution:
  - Client aggregation
    - clients have similar dependencies
  - Server aggregation
    - Same application hosted on a cluster of servers
  - Port aggregation
    - Same service hosted on different ports

Outline

- Overview
- Dependency discovery techniques
- Deployment & results
- Conclusion

Orion deployment



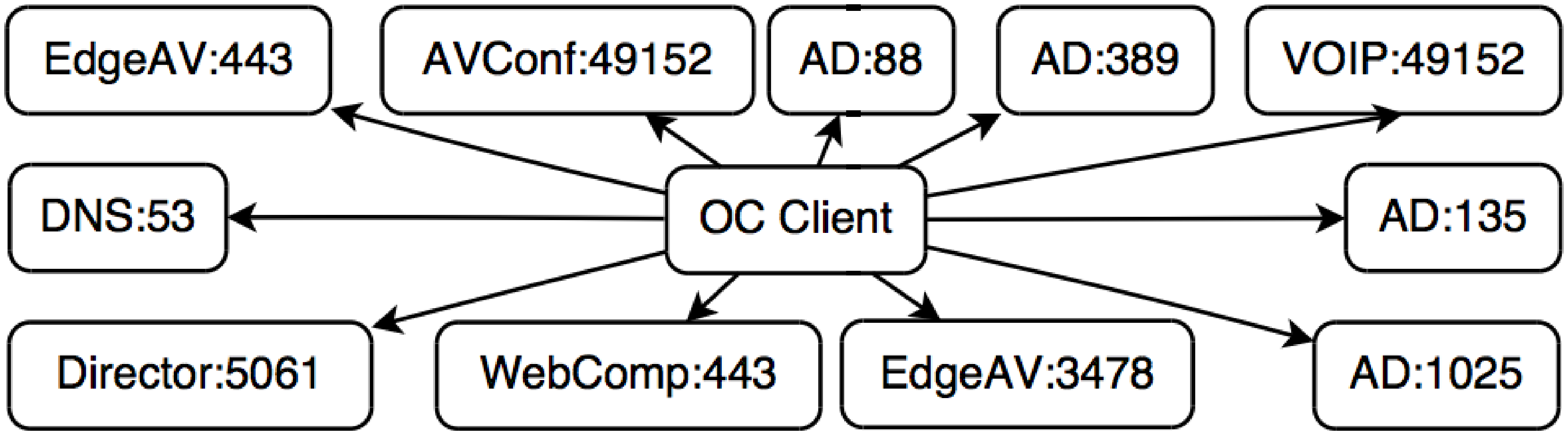
- Five dominant applications
  - Exchange, Office Communicator, Source Dept, Distributed File System, Web
- Traffic on R0
  - Over 2,000 clients
- Traffic on R1/R2
  - Email service used by over 10,000 users
  - Largest internal web portal

Accuracy of dependency discovery

- Evaluation criteria
  - Missed dependencies – false negative
  - Incorrectly-inferred dependencies – false positive
  - Reduction ratio *97.9% – 99.6% for Orion*

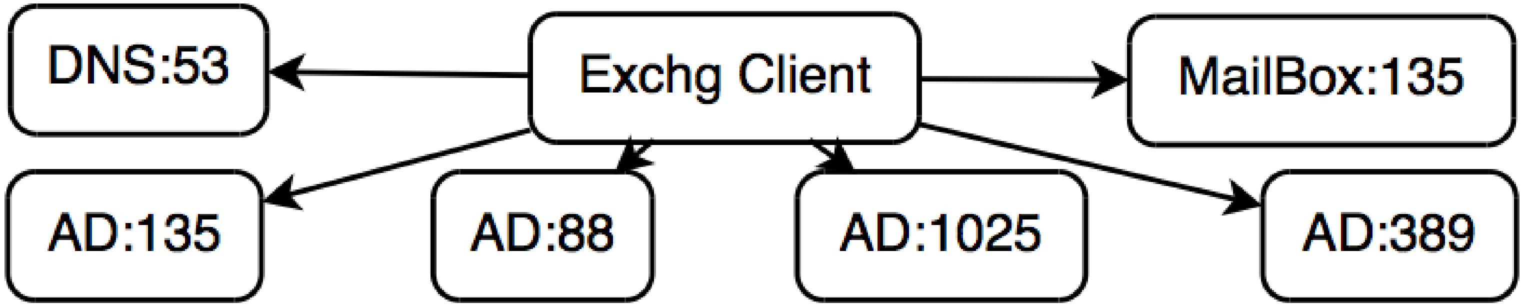
	Exchange		DFS		Sharepoint		OC client		SD client	
	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP
Orion	**0**	**26**	**0**	**13**	**0**	**3**	**0**	**77**	**0**	**4**
Sherlock:10	0	178	0	102	0	65	2	125	0	52
Sherlock:100	0	57	0	93	0	168	1	85	0	29
eXpose	1	443	0	570	0	565	1	1416	0	323

Dependencies of Office Communicator



OC client dependencies

Dependencies of Exchange



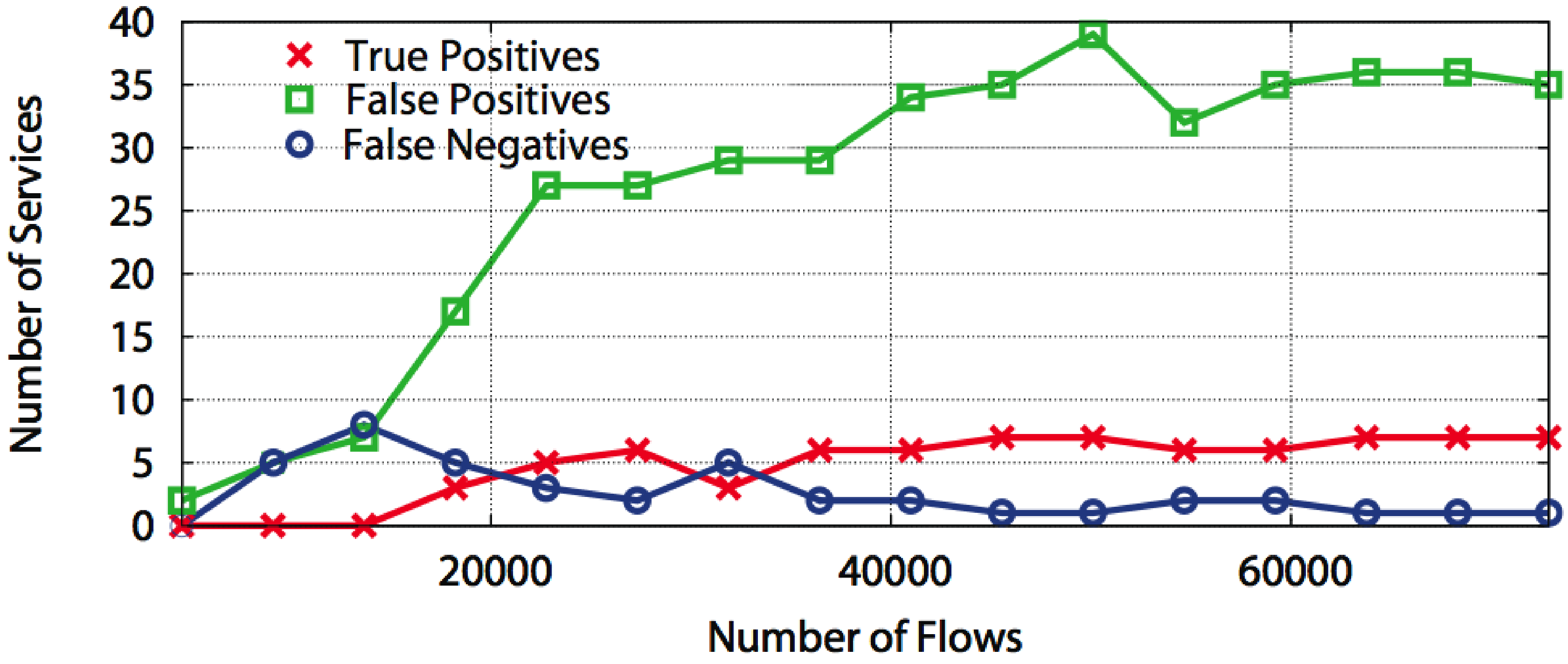
Exchange client dependencies

Effects of noise filtering & flow generation

	Exchange		DFS		Sharepoint		OC client		SD client	
	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP
Orion	**0**	**26**	**0**	**13**	**0**	**3**	**0**	**77**	**0**	**4**
noFilter	0	49	0	25	0	6	0	159	1	19
noFlow	0	2488	0	988	0	534	0	3594	0	198

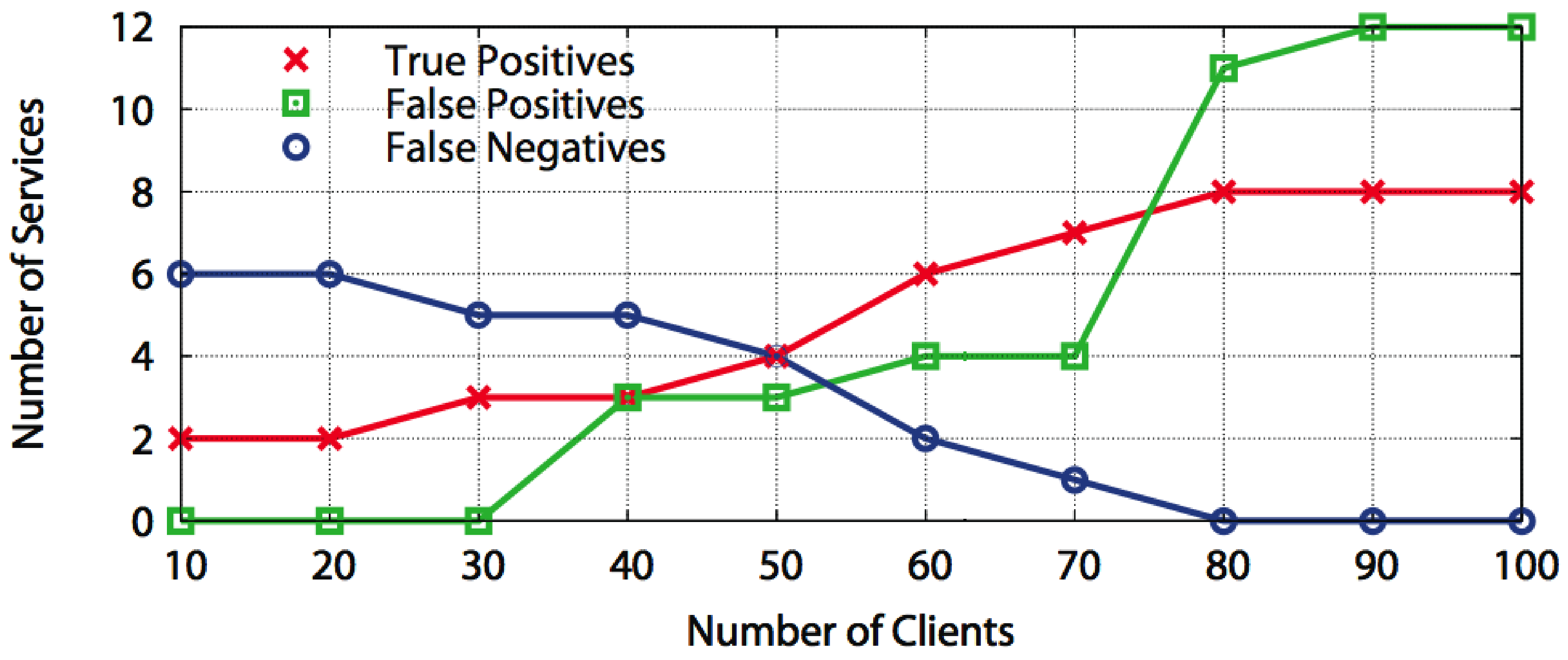
Convergence

Exchange



Impact of aggregation

DFS: distributed file system



Limitations

- Relatively long discovery time
- Not applicable to P2P applications
- May miss certain types of interactions
- May include false positives

Conclusion

- Lessons learned
  - Temporal analysis has inherent limitations
  - False positive can be reduced to a manageable level
- Summary
  - A new technique to discovery dependency based on spike in delay distribution
  - Evaluate using production enterprise applications

Thank you!

Questions?