Monitoring Data Fusion for Intrusion Tolerance

Atul Bohara, Gabriel A. Weaver, William H. Sanders

March 11, 2015

Motivation

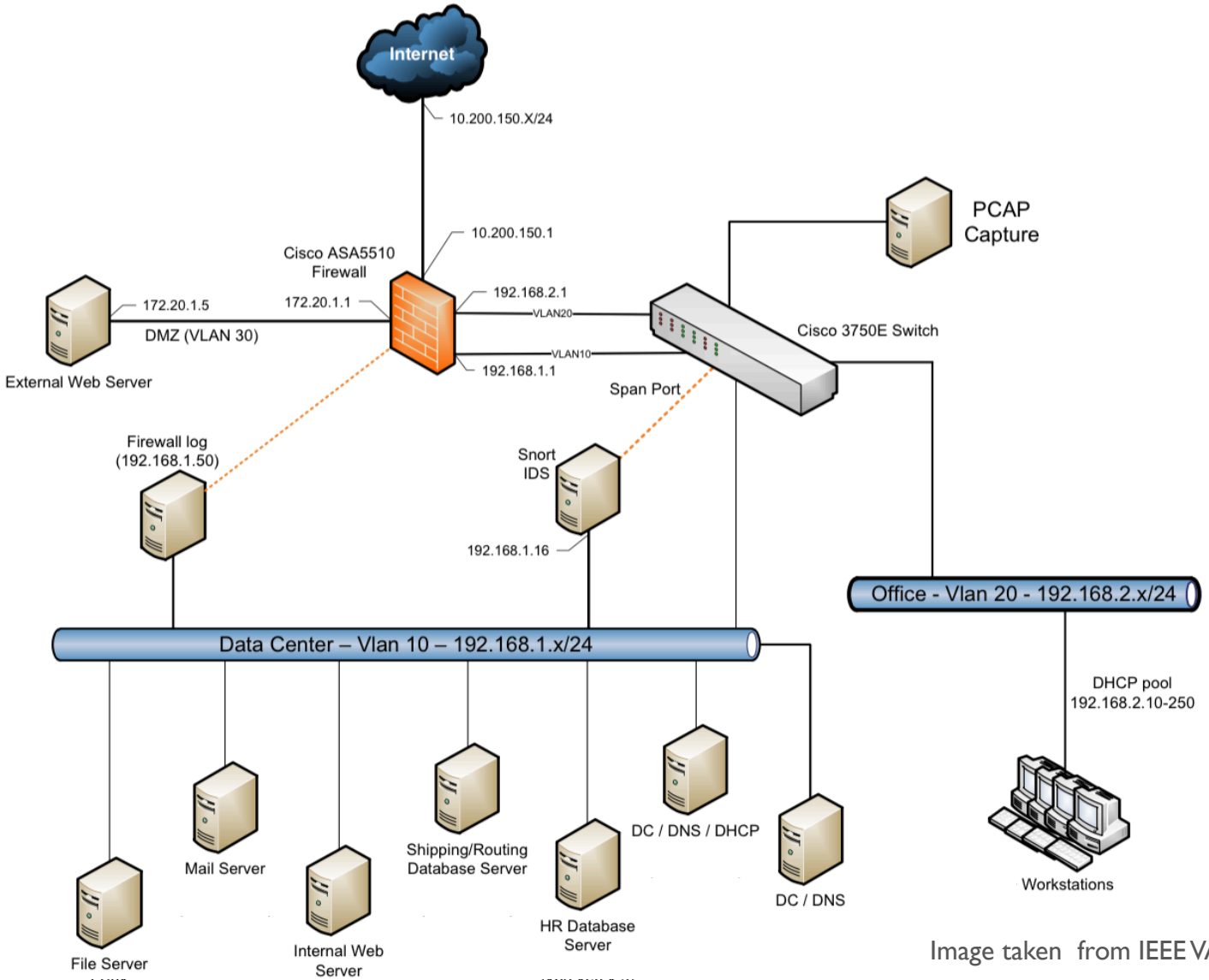


Image taken from IEEE VAST 2011

Motivation

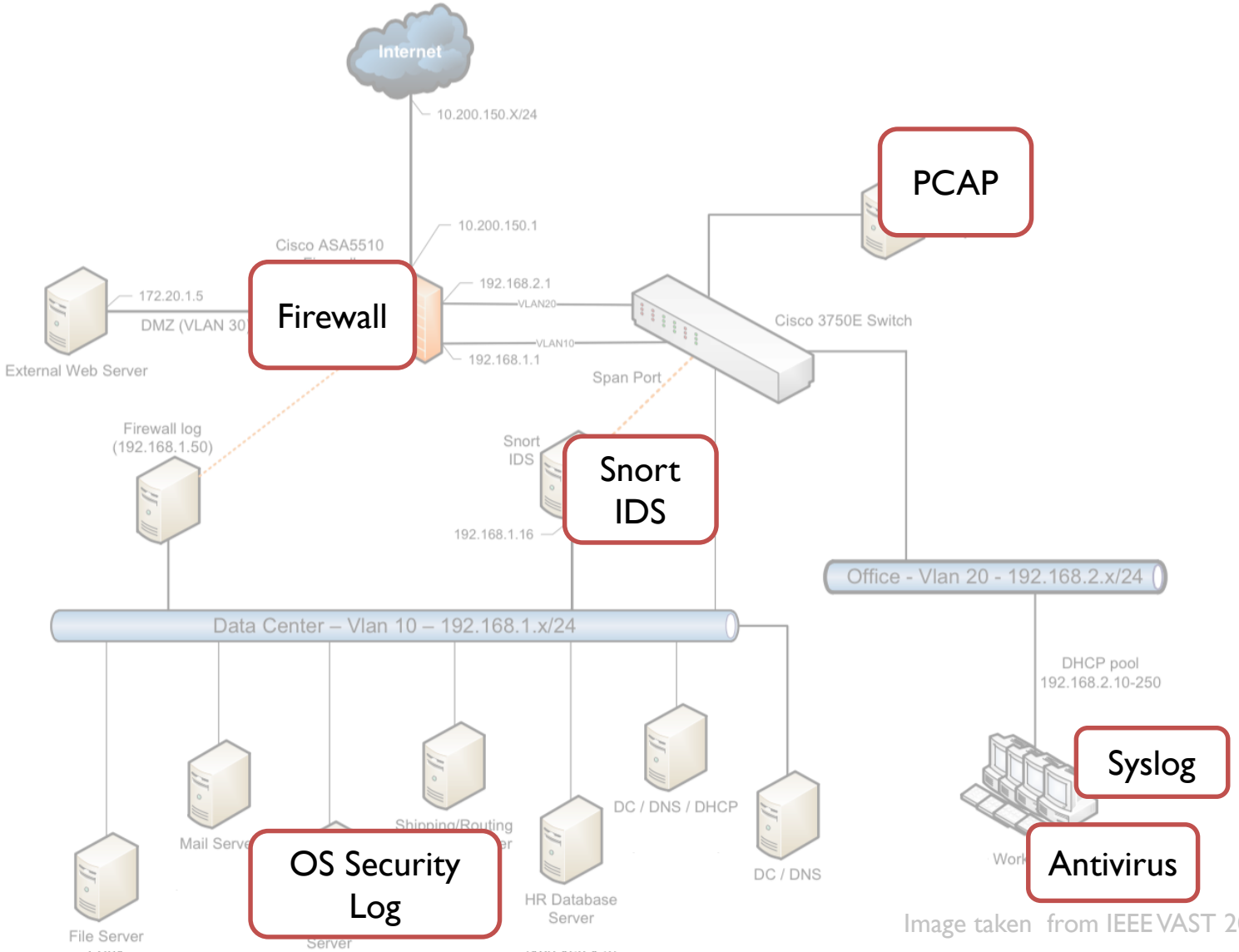


Image taken from IEEE VAST 2011

Problems

- Managing and analyzing these logs becomes time-consuming and error-prone
- Intrusion detection: inefficient, ineffective

- Our approach: Fusion of monitoring information for intrusion tolerance
  - Combine diverse monitoring information

Motivating Example

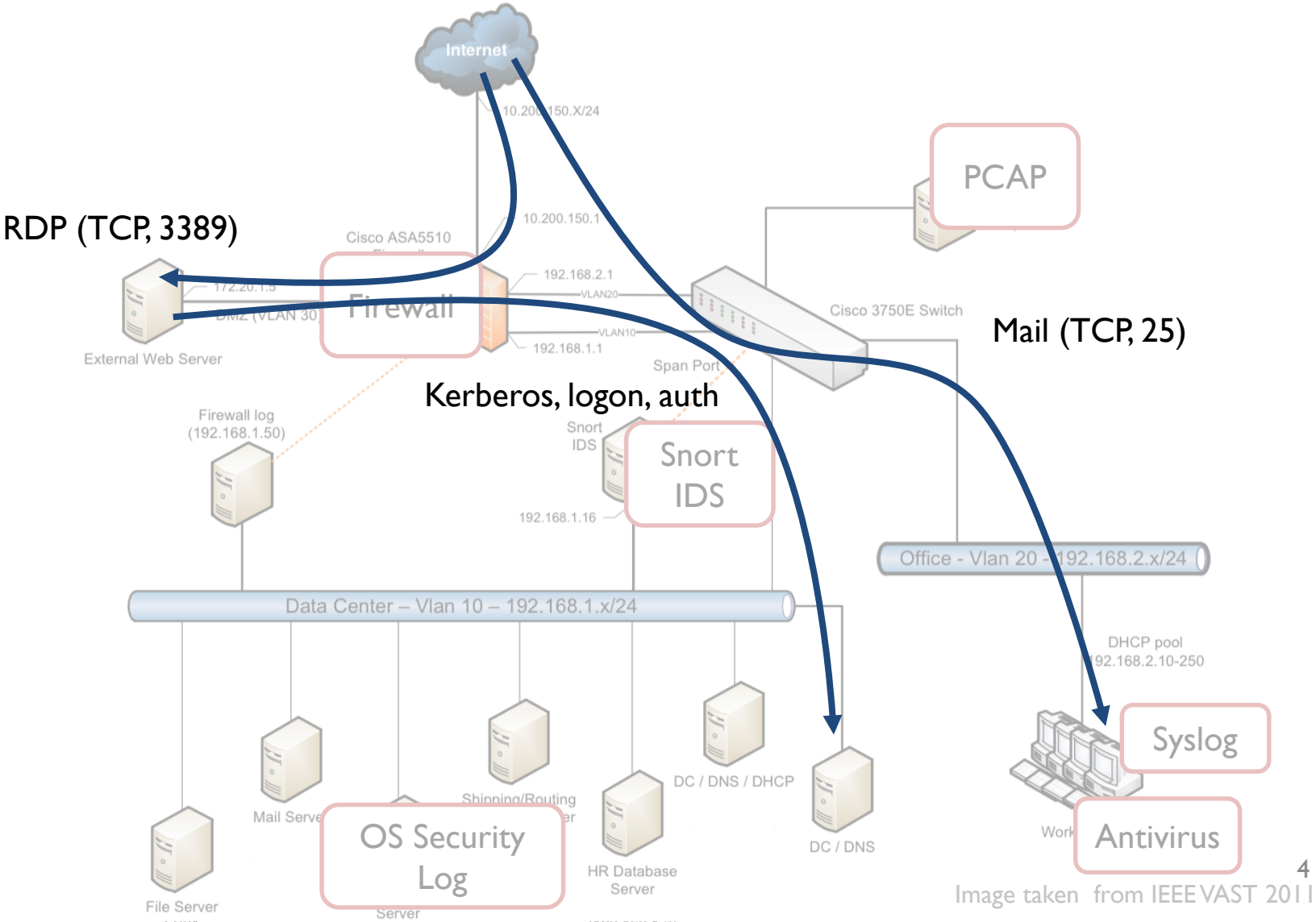
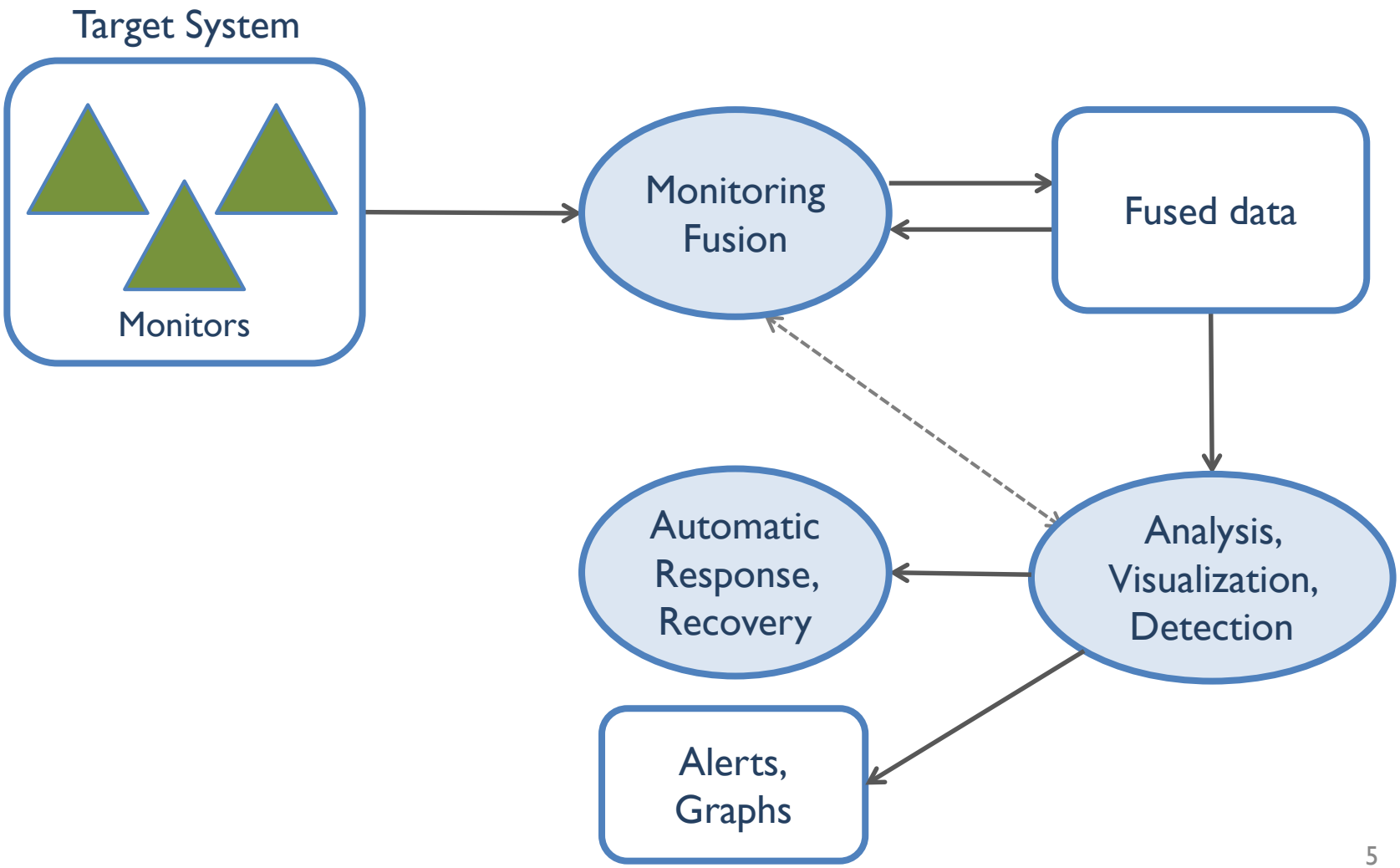


Image taken from IEEE VAST 2011

Overview of Fusion



Goals of Fusion

1. Generate different views of security state of the system

2. Aggregate security monitoring information

3. Facilitate analysis of security attacks through the synthesized information

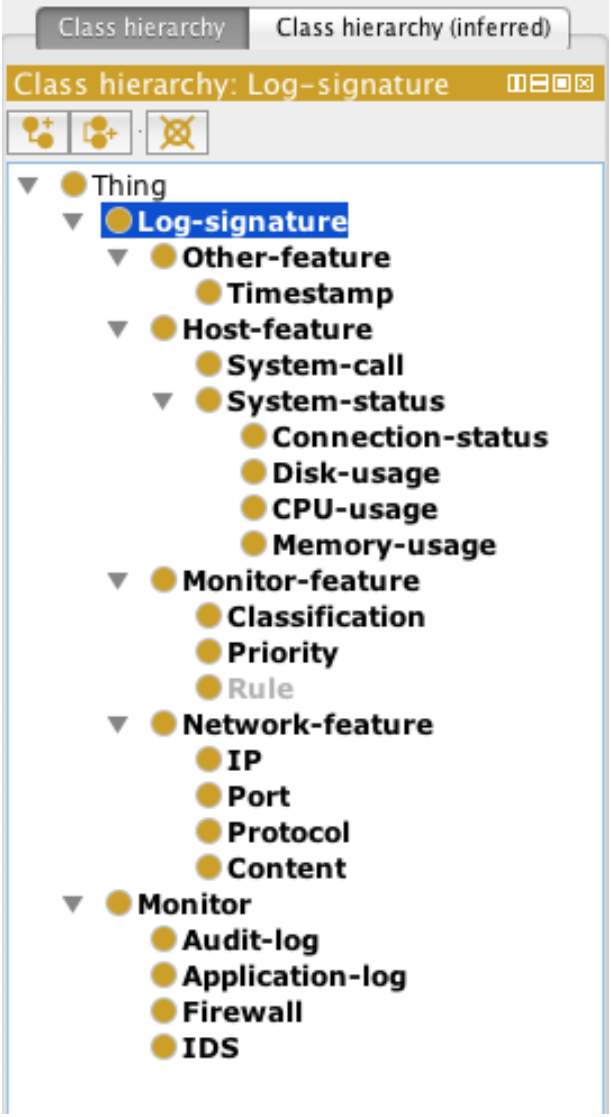
4. Increase the detection capability of monitor deployment through fusion techniques

Before Fusion…

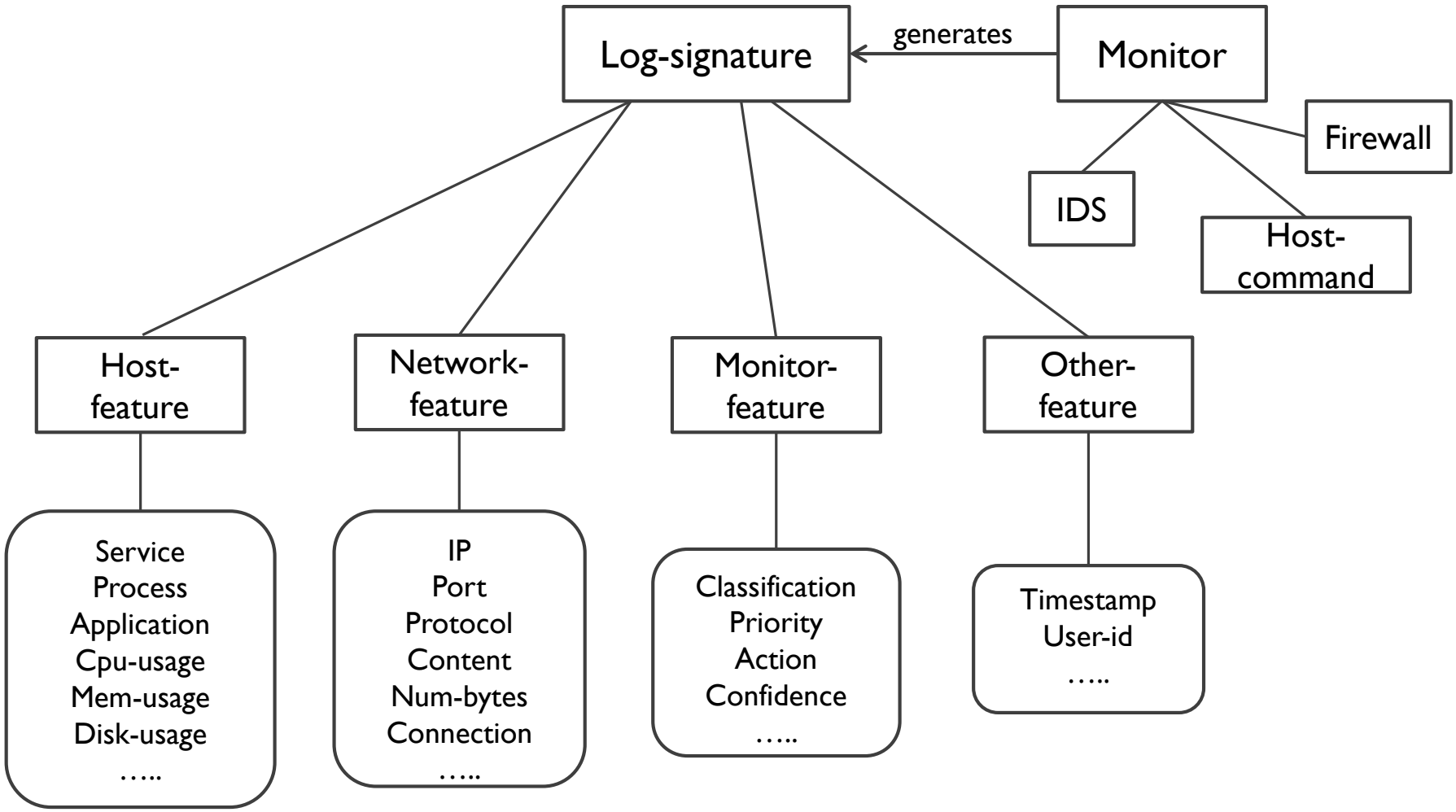
- Normalization of logs
  - Necessary first step to develop a collaborative framework
- Understanding relationships among different fields
- Our approach: ontology for monitor formats

Ontology

- Monitor signature ontology
  - Machine actionable specification of features from diverse monitors
  - Relationship among these features



Ontology


Log-signature
generates
Monitor
Firewall
IDS
Host-command
Host-feature
Network-feature
Monitor-feature
Other-feature
Service
Process
Application
Cpu-usage
Mem-usage
Disk-usage
…..
IP
Port
Protocol
Content
Num-bytes
Connection
…..
Classification
Priority
Action
Confidence
…..
Timestamp
User-id
…..

Goals of Fusion

1. Generate different views of security state of the system

2. Aggregate security monitoring information

3. Facilitate analysis of security attacks through the synthesized information

4. Increase the detection capability of monitor deployment through fusion techniques

First Goal

- Increase visibility into the target system
  - Different information (state) is observed by different monitors
  - Combine them in useful ways

- **Our approach:** CPTL + Hypergraph

Cyber-Physical Topology Language

- Graph based data model to define networked systems and assets
- Set of operations: join, abstract, and contract

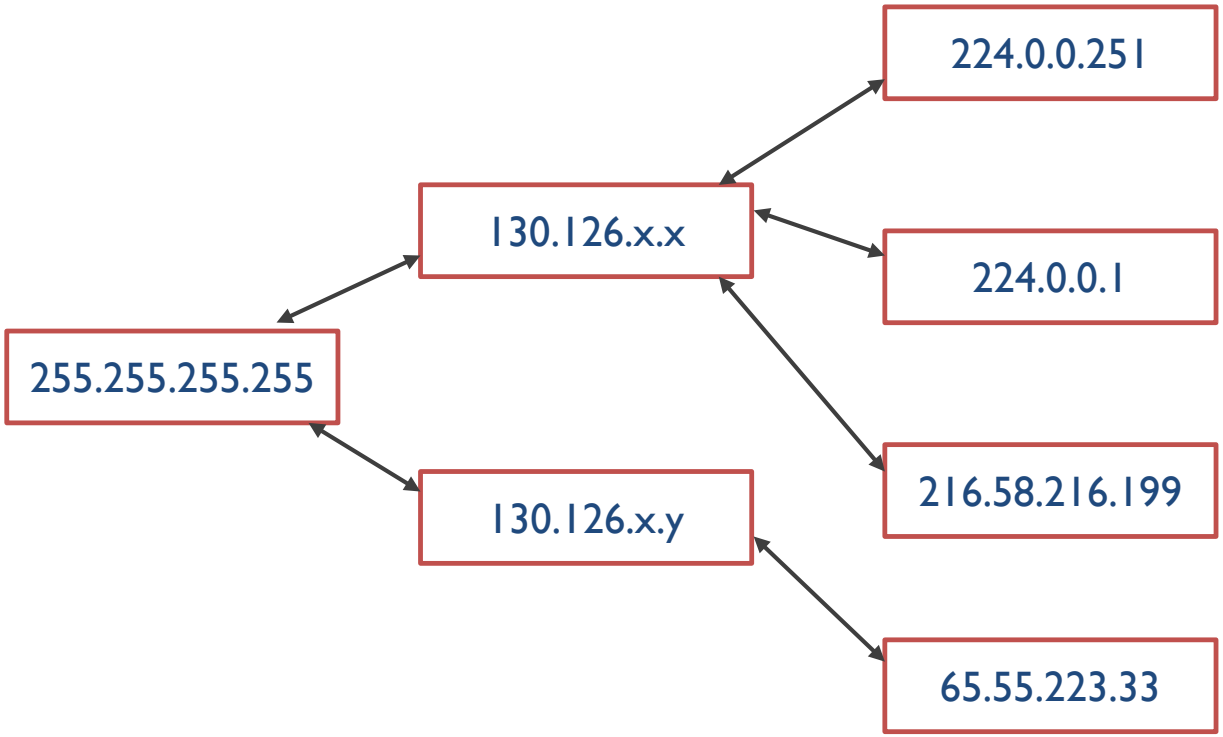
CPTL data model is defined as $(G, K)_I$
Where:
G is a graph
I is an interpretation
K is an ontology
This construct describes an interpretation I of a graph G with respect to an ontology K

- G. A. Weaver, C. Cheh, E. J. Rogers, W. H. Sanders, and D. Gammel, *"Toward a cyber-physical topology language: Applications to nerc cip audit"*, SEGS '13.
- C. Cheh, Masters Thesis, *"The Cyber-Physical Topology Language: Definition and Operations"*, UIUC, 2014.

Achieving the First Goal

1. Define *Views* in CPTL data model
2. Select monitors for desired *view*
3. Extract data from log files of desired monitors
4. Generate *View* using the extracted data

Example View: IPs and Protocol



Example View: IPs and Protocol

View Generator: IPs and Protocols Data Model	
Purpose	View IP addresses within a network and connections among them.
CPTL Data Model	

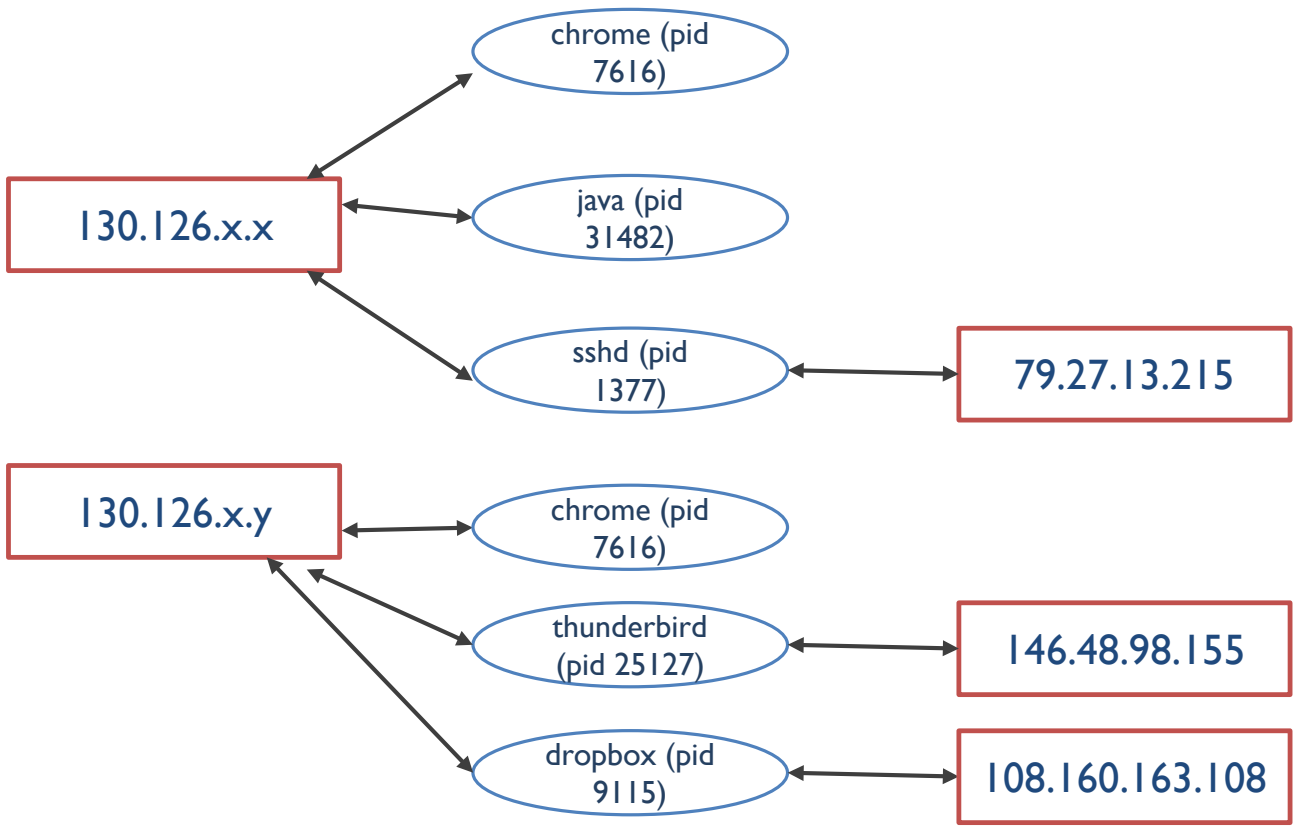
Let corpus $C = \{(s, p, d)\}$ in which $s, d \in IP$ are source and destination IP addresses that communicate using protocol $p \in P$. The sets IP and P are defined by an ontology K.

Using C, construct $G = (V, E)$ as follows:

- $V[G] = S[C] \cup D[C]$. Where $S[C]$ is the set of source IPs in the corpus and $D[C]$ is the set of destination IPs in the corpus.
- $E[G] = \{(s, d) | \exists (s, p, d) \in C\}$

Note that we can construct subviews of the graph by filtering C with a predicate beforehand. For example $C_{ARP} = \{(s, p, d) | p = ARP\}$.

Example View: PID and IP

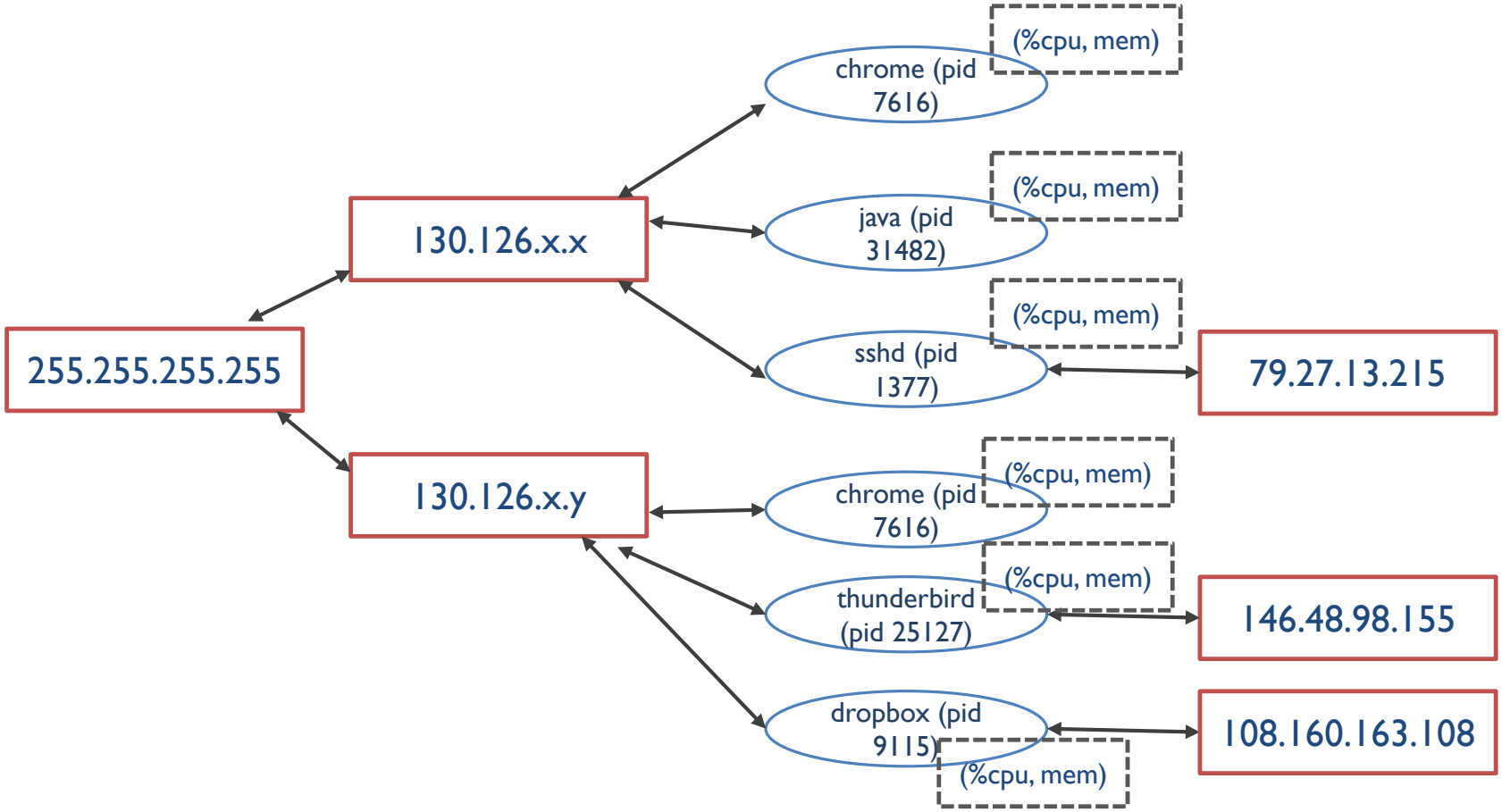


Data taken from www.ntop.org

Example View: PID and IP

View Generator: PID and IP Data Model	
Purpose	A more granular view of how a device uses a network.
CPTL Data Model	Let corpus $C = (p, d)$, where $p \in PID$ and $d \in IP$. Process denoted by p is either running on source IP d or has opened a socket to destination IP d. The sets IP and PID are defined by and ontology K. Using C, construct $G = (V, E)$ as follows: • $V[G] = P[C] \cup D[C]$, where $P[C]$ is the set of PIDs in the corpus and $D[C]$ is the set of IPs in the corpus. • $E[G] = C$ Note that we can construct different views by constructing C to capture traffic from certain users or applications beforehand. We can also construct subviews of the graph by filtering C with predicates of source IPs or destination IPs. For example $C_{SRC} = \{(p, d) \mid d \in S[C]\}$

Composite View: Network, Resources



Data taken from www.ntop.org

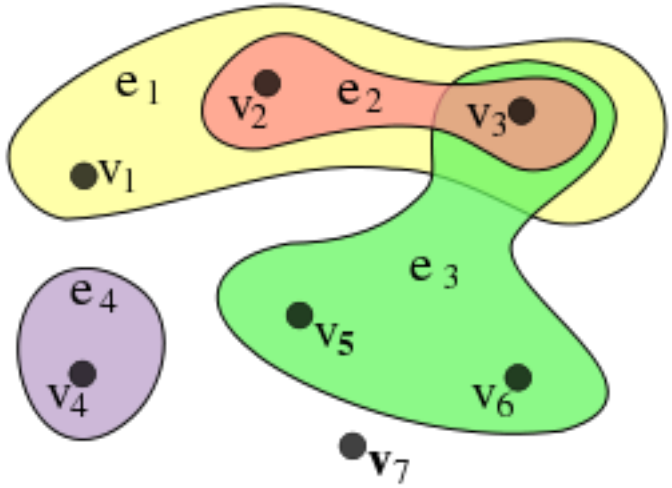
Composite View: Network, Resources

View Generator: IPs, PID and Resource Data Model	
Purpose	Monitor local resource usage with every network activity
CPTL Data Model	Let corpus $C = (s, d, p, c, m)$ in which $s, d \in IP$ are source and destination IP addresses and $p \in PID$ is the corresponding local process. $c \in CPU$ and $m \in Mem$ are percentage CPU usage and Memory usage respectively. The sets IP, PID, CPU, and Mem are defined by and ontology K. Using C, construct $G = (V, E)$ as follows: • $V[G] = S[C] \cup D[C] \cup P[C]$, where $S[C]$, $D[C]$, and $P[C]$ are sets of Source IP address, Destination IP address, and Process IDs in the corpus, respectively. • $S_{VP} = \{CPU, Mem\}$ is the set of vertex attribute values for all the vertices of type $P[C] \in V[G]$, where $P[C]$ is the set of PIDs in the corpus. • $E[G] = E_N \ Join \ E_P$ where E_N and E_P are edge sets of previously mentioned views ($Join$ is over PID)

Where is the Data?

Monitor Corpus (C)	Fields
IP-Protocol view	
tcpdump	Source, Destination, Protocol
PID-IP view	
lsof	PID, DestinationIP
netstat	PID/program name, Local Address, Foreign Address
Network-Resources view	
tcpdump	Source, Destination
netstat	Local Address, Foreign Address, PID/ Program name
top	PID, %CPU, MEM

Introduction to Hypergraph



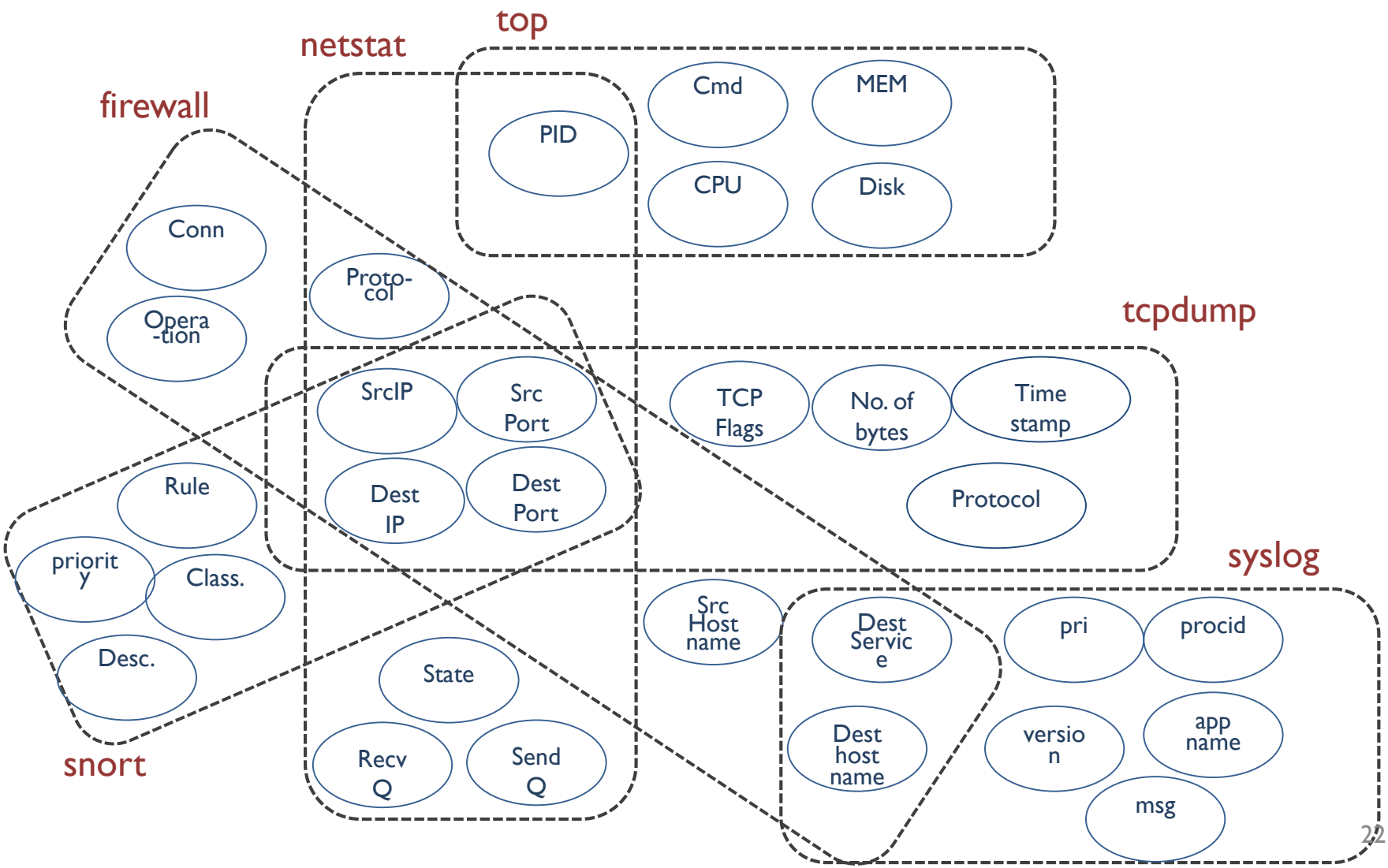
An example of a hypergraph

Vertices: V = {v1, v2, v3, v4, v5, v6, v7}

Hyperedges: E = {e1,e2,e3,e4} = {{v1, v2, v3}, {v2,v3}, {v3,v5,v6}, {v4}}.

Source: wikipedia

Hypergraph for Monitoring Data



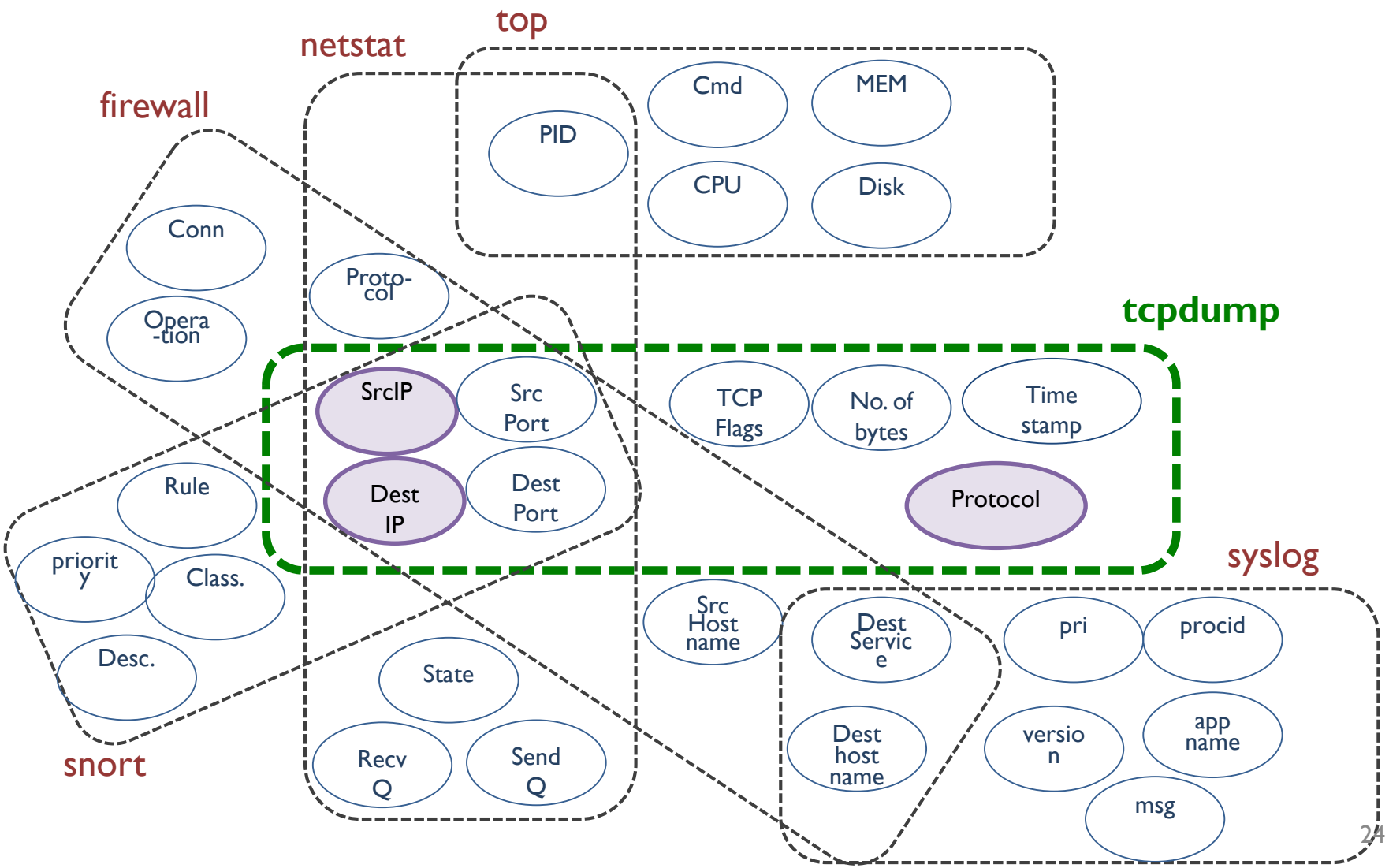
Selecting Required Monitors

- Greedy algorithm for (sub) set cover in *Hg(V, E)*

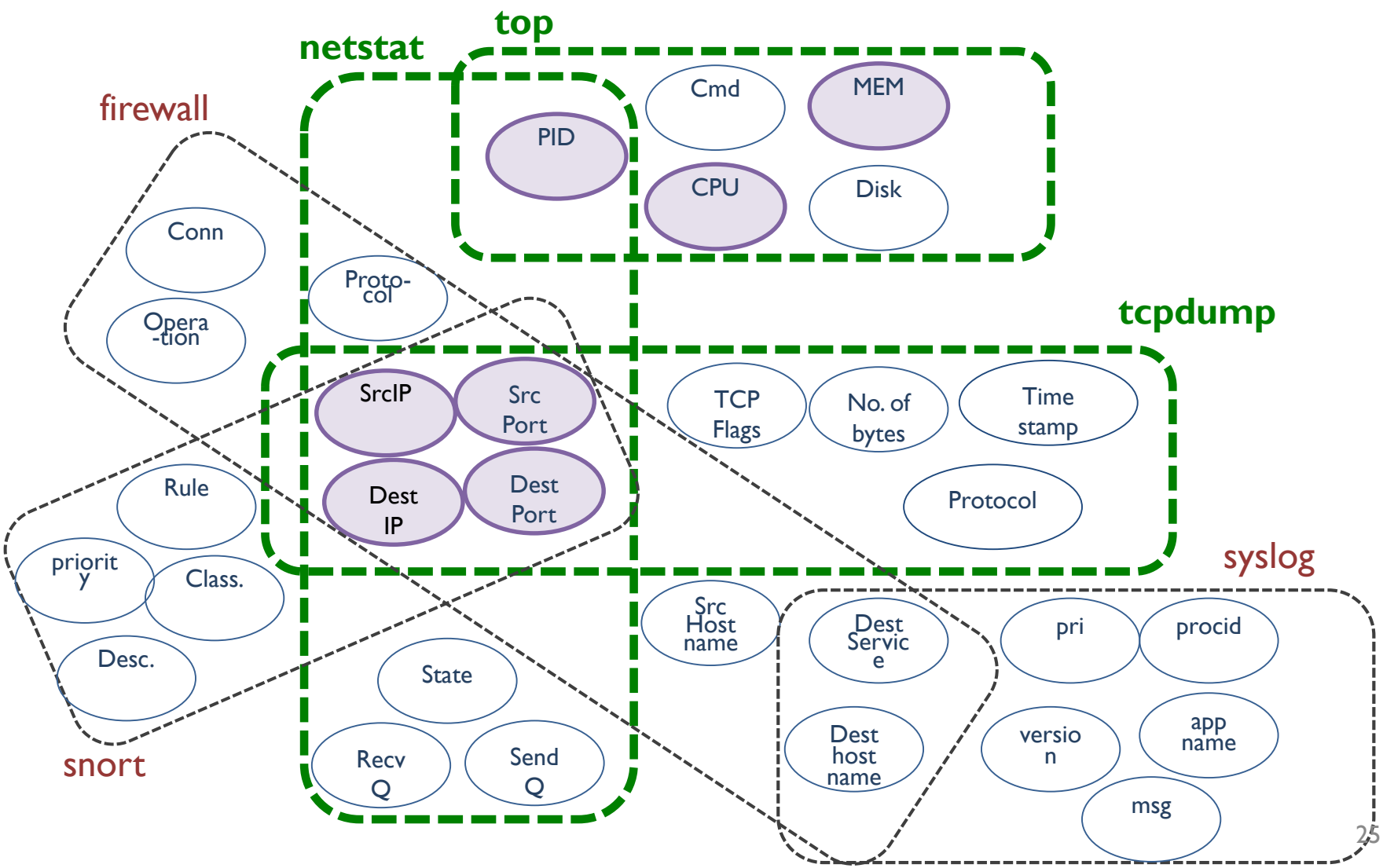
- **Input**: set of fields required to generate the view (I)
- **Output**: set of monitors

- **Algorithm**
  - Repeat until all fields in I are covered
  - In each step, pick the monitor having maximum fields common with currently required field set I
  - Consider only those monitors that have some *fusion feature* common with already selected monitors

Which Monitors for IP/Protocol View?



Which Monitors an Network-Resource View?



Summary

- Combining monitoring data from multiple monitors
  - View generation
- Hypergraph based monitor selection
  - Which monitors we have to depend on
  - Which combinations are more economical than others
  - How much redundancy
  - What impact will be there if monitors are taken out or get compromised

Ongoing Work

- An online way to generate CPTL views from monitoring data automatically
- Experiment with real data sets
- Automatically infer what fields we need, given an attack or query?
  - Currently we need to do this empirically and manually
- Automatic alert correlation to group logically interconnected events into one group

References

1. *"Toward a cyber-physical topology language: Applications to nerc cip audit"*, G. A. Weaver, C. Cheh, E. J. Rogers, W. H. Sanders, and D. Gammel, SEGS '13
2. *"The Cyber-Physical Topology Language: Definition and Operations",* C. Cheh, Masters Thesis UIUC, 2014
3. "*Ontology based cooperative intrusion detection system.*", He, Yanxiang, et al. , Network and Parallel Computing ' 04
4. *"The Intrusion Detection Message Exchange Format (IDMEF)"*, http://www.ietf.org/rfc/rfc4765.txt
5. *"Greedy Algorithms, Divide and Conquer, and DP"*, Lecture scribe CS787 Advanced Algorithms, David Hinkemeyer and Dalibor Zeleny, 2007