Provenance Segmentation

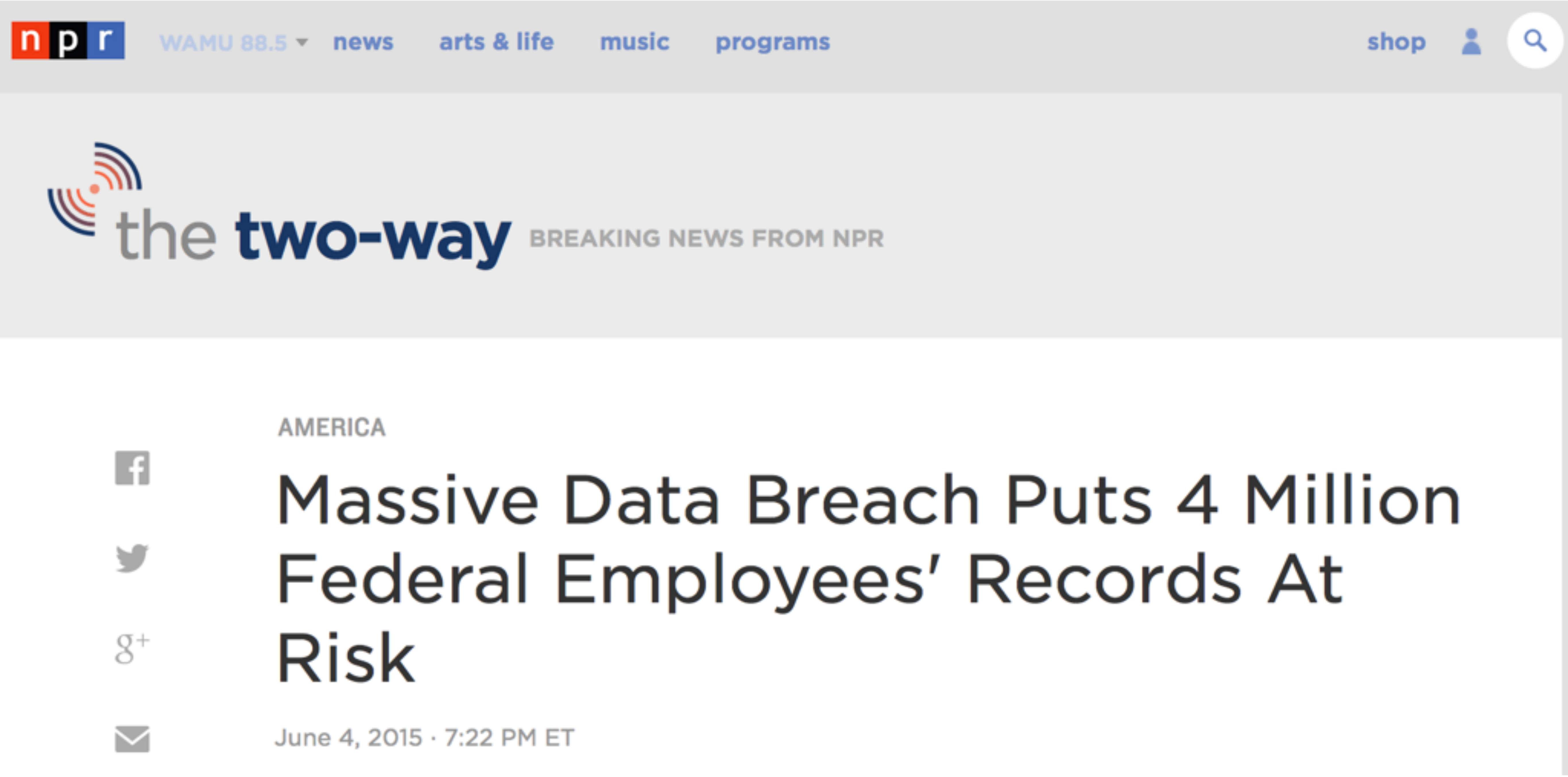
parc
A Xerox Company

galois

THE UNIVERSITY of EDINBURGH

Rui Abreu, Dave Archer,
Erin Chapman, James Cheney,
Hoda Eldardiry, Adria Gascon

TaPP 2016
June 9, 2016



AMERICA

Massive Data Breach Puts 4 Million Federal Employees' Records At Risk

June 4, 2015 · 7:22 PM ET

Office of Personnel Management breach (2015)

Context

- "Advanced persistent threats" (APTs) are stealthy, long-term, resourceful attackers
    - Simulate normal user behavior most of the time
    - Lateral attacks, avoid violating fixed security policy
- *Transparent Computing*: try to fight APTs through pervasive recording and analysis of provenance
    - $60m DARPA research program (2015-19)

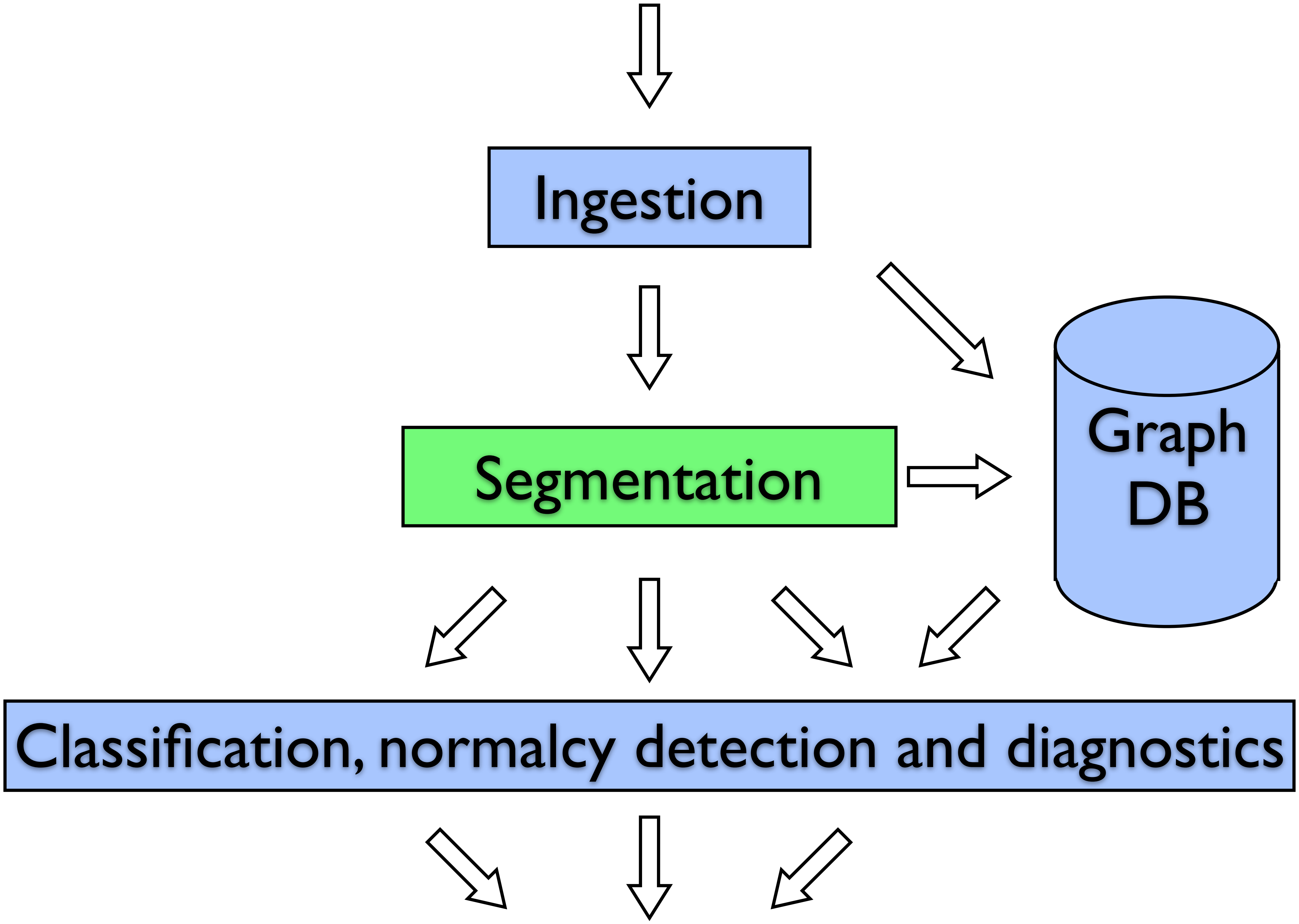
Provenance and Security

- Security OF provenance
  - provenance abstraction (ProvAbs, ProPub, provenance redaction etc.)
  - privacy views (Davidson et al. 2011)
- vs. using provenance FOR security
  - provenance based access control (Thuraisingham et al., Sandhu et al.)
  - "transparent computing" (this work)

Problem

- Other projects are developing systems (e.g. SPADE, DTrace, RecProv) to *record* + *generate* provenance
- We are developing an *analysis* system called *ADAPT* (A Diagnostics Approach to Advanced Persistent Threat detection)
- The incoming provenance data is much too massive for existing analysis techniques we have in mind
- Also, some of the techniques expect preprocessing into "uniform" chunks
  - (e.g. all user activity over 1-hour periods)

Incoming provenance data from recorder (e.g. SPADE)



Ingestion

Segmentation

Graph DB

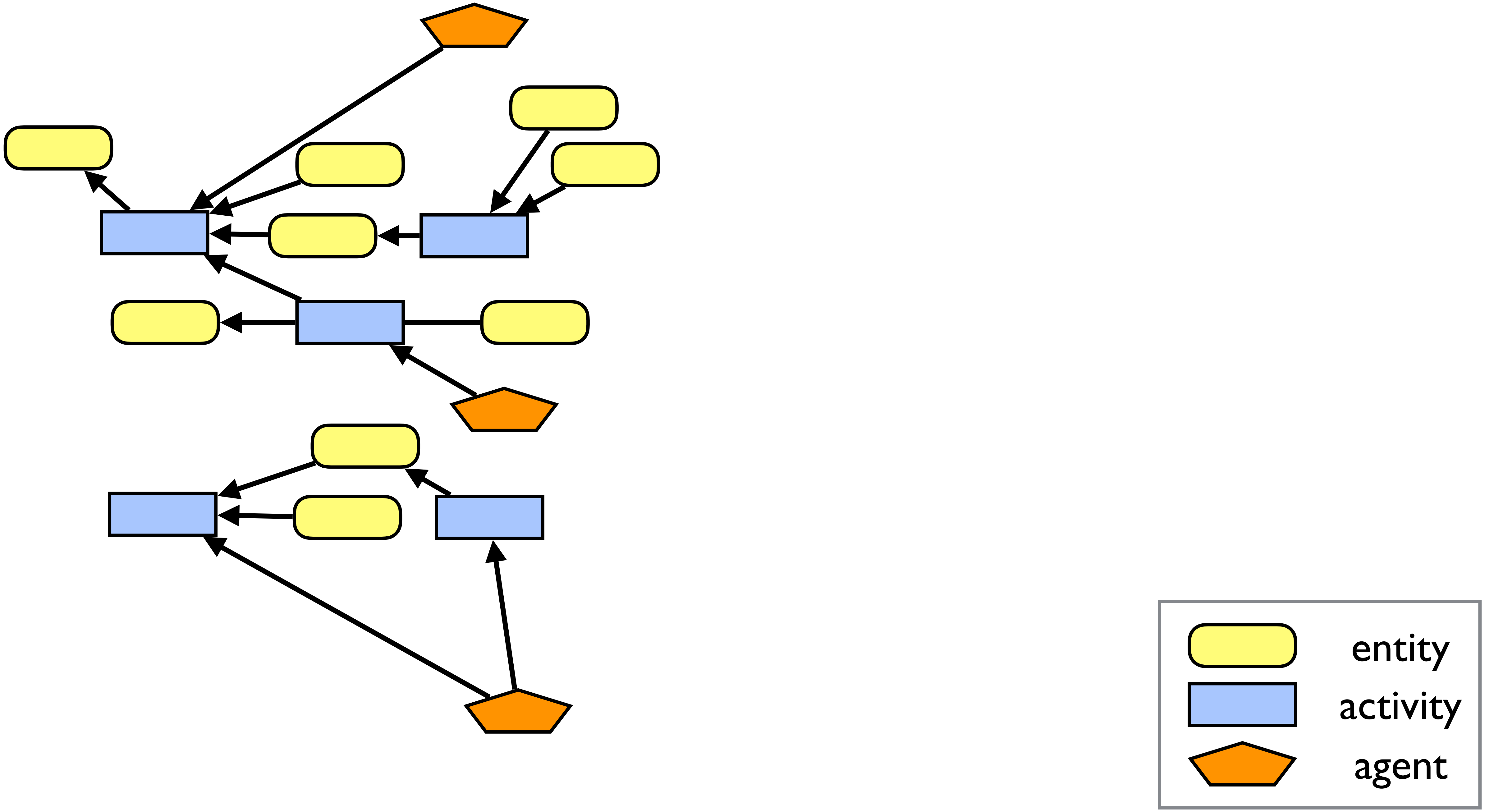
Classification, normalcy detection and diagnostics

Attack subgraph / warning

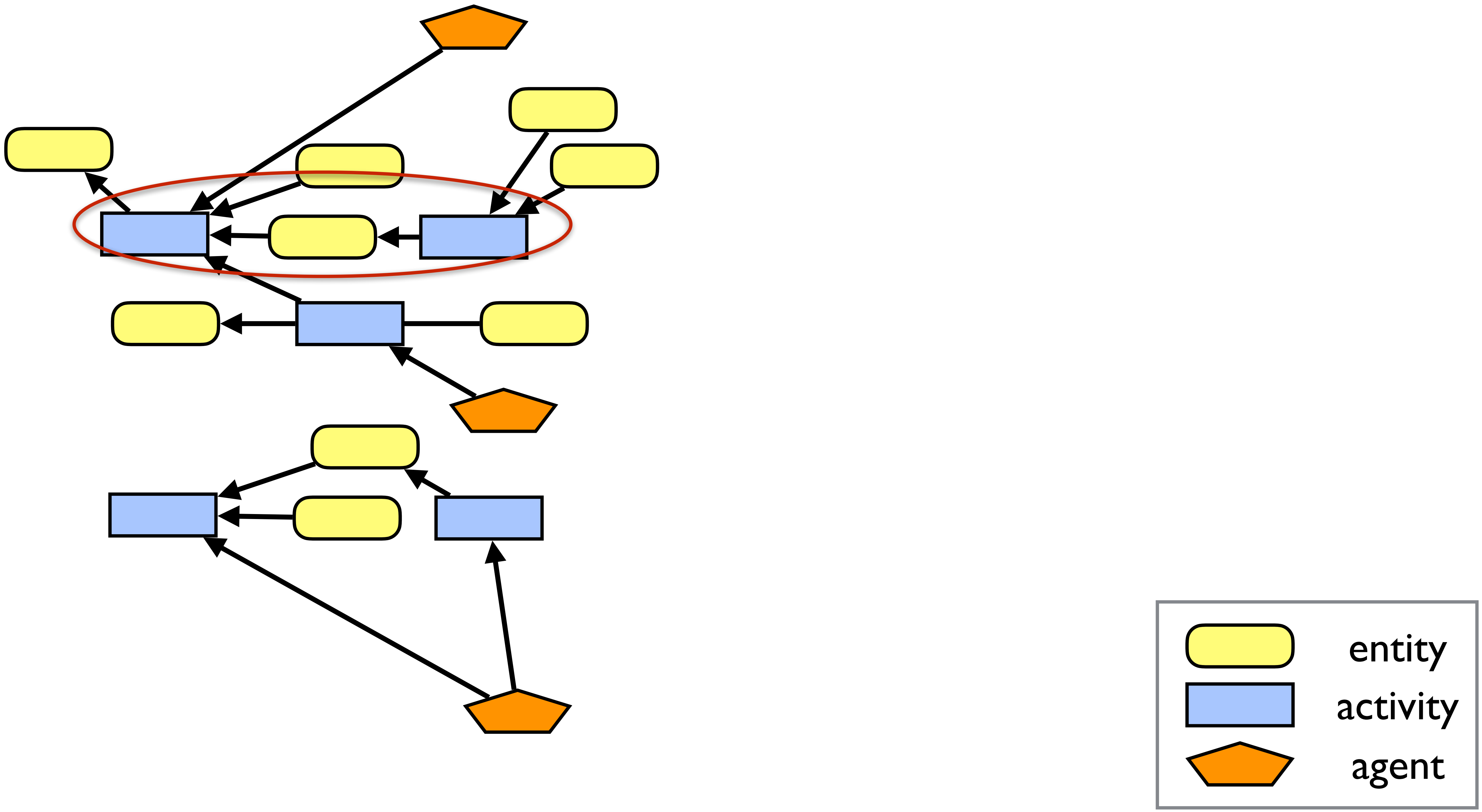
Terminology

- *Raw graph*: the unadulterated input graph from recorder
- *Segment*: subgraphs that isolate interesting features
- *Segment layer*: a summary graph with nodes corresponding to summarizing relationships between them
- *Batch segmentation*: construct segment layer for a given input graph
- *Incremental segmentation:* given incoming stream of provenance data, incrementally construct/add to segment layer

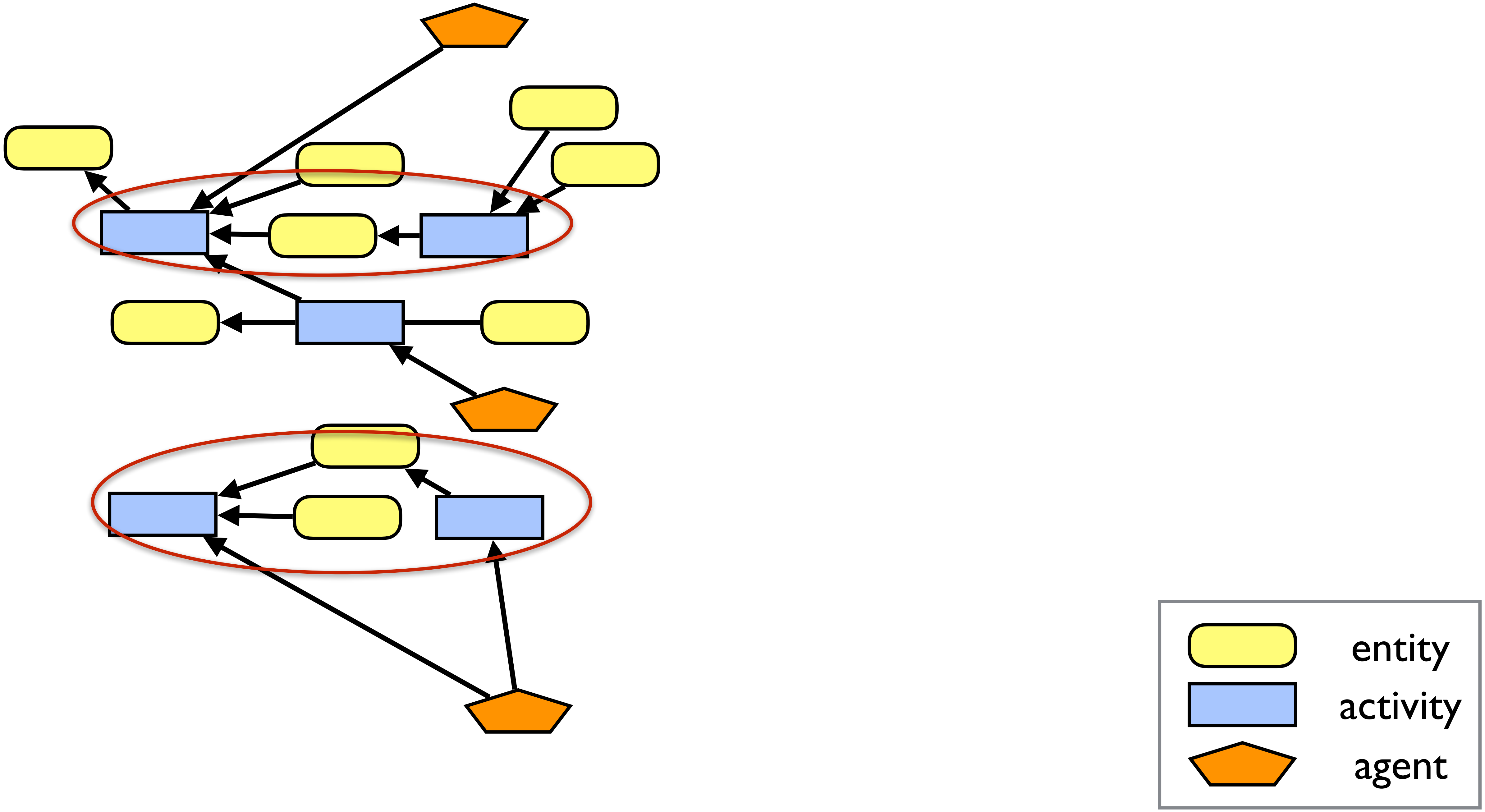
Example: monitoring agents



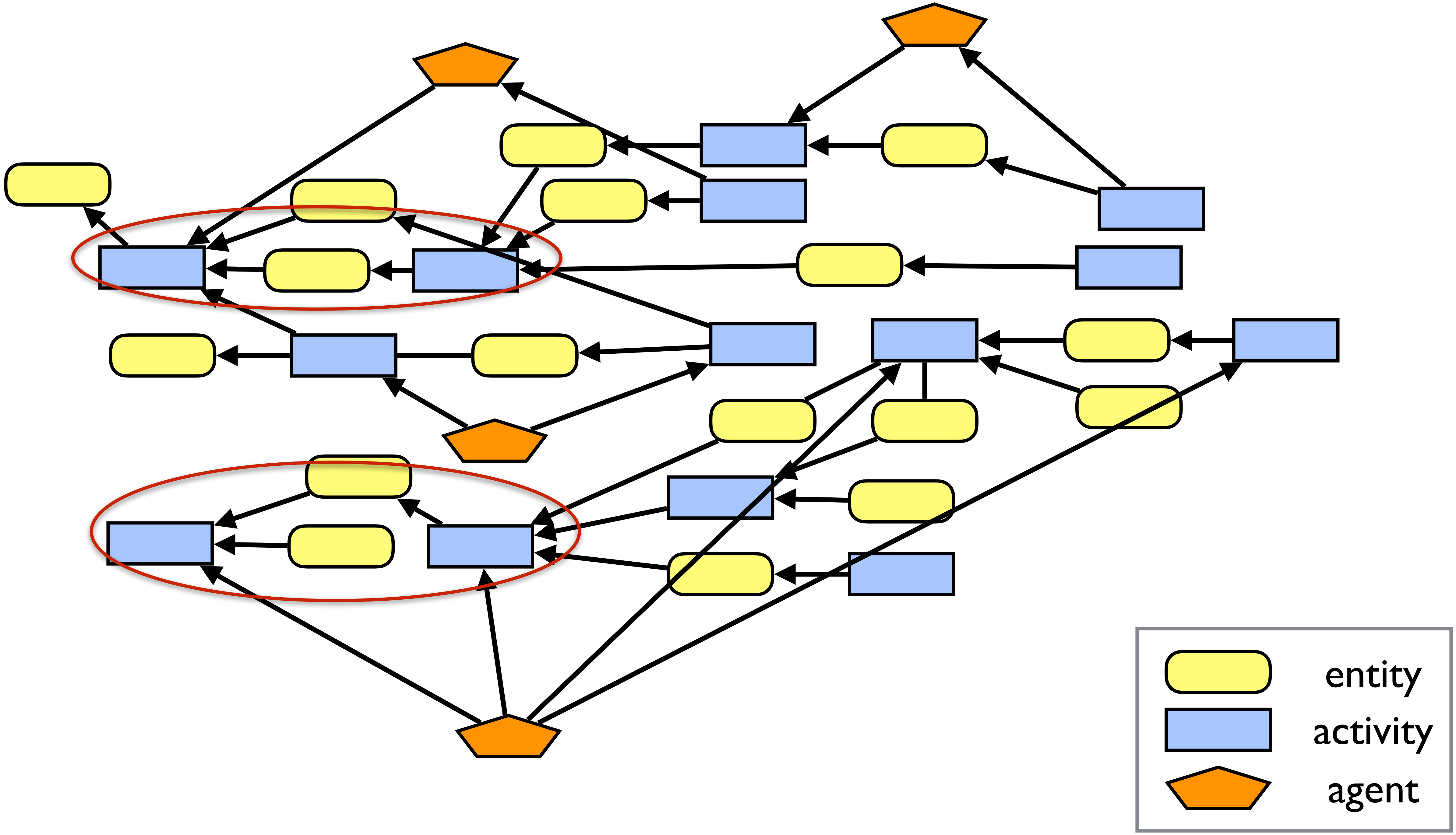
Example: monitoring agents



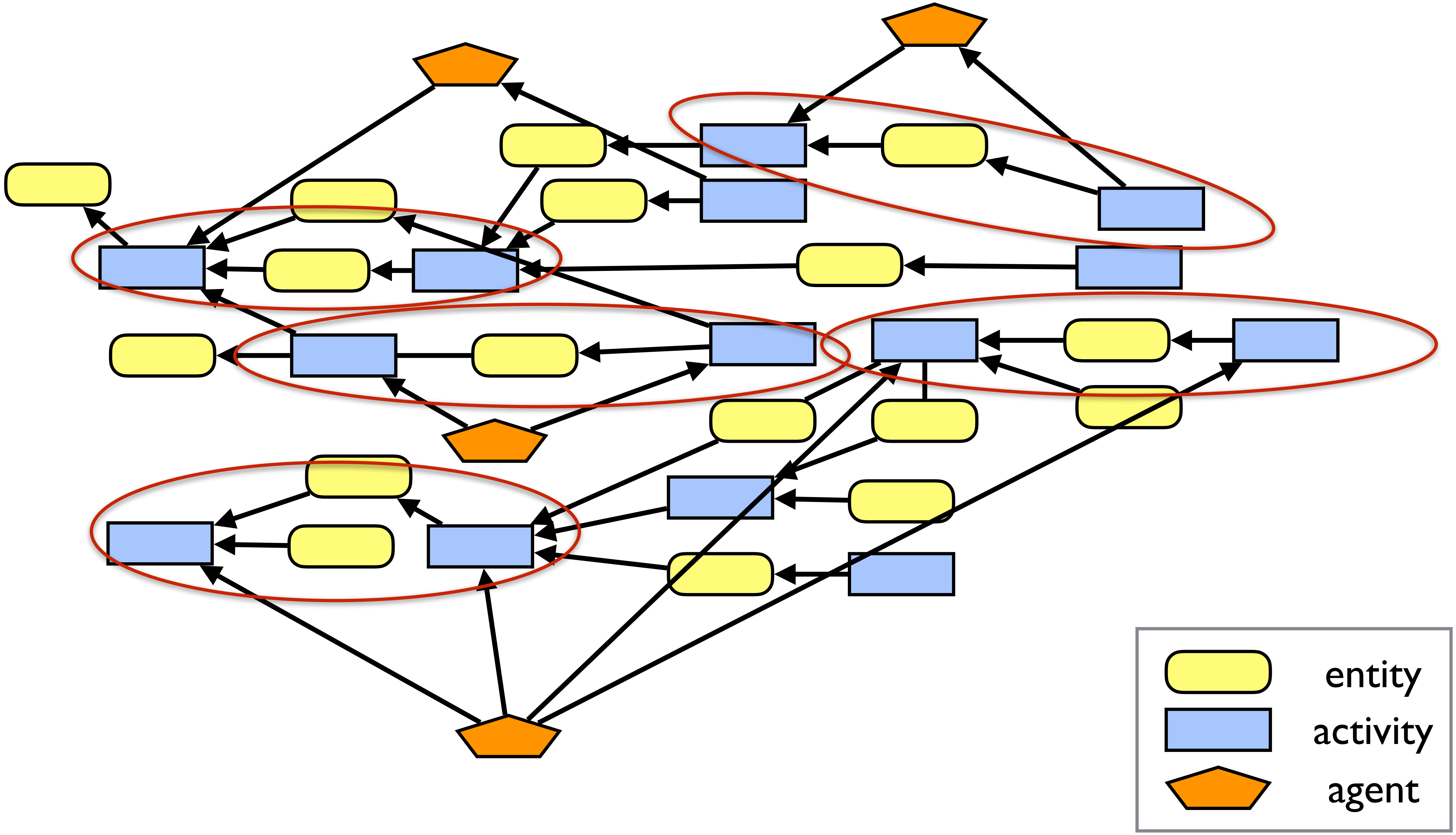
Example: monitoring agents



Example: monitoring agents



Example: monitoring agents



Complications

- **Scale**: Segmentation needs to be fast hence criteria need to be very simple
- **Retention**: Does segment layer suffice for analysis or do we need to keep raw data? (For how long?)
- **Representation**: Segments need not be convex. How to represent arbitrary subgraphs?
- **Incomplete information**: What to do if time information or edges missing?

Segmentation criteria

- Different applications may need different things
    - (e.g. user vs. process centric, large vs. small time windows)
- *Segmentation criteria* to make this configurable
- Two basic kinds of segments so far:
    - by radius (up to N, from a "seed" node, following certain edge types only)
    - by time (regular time intervals)

Example

```
segment byPidTime(pid=X, startTime=T)
     by radius 3 from PID=X
         following {"wasDerivedFrom", "used", "wasGeneratedBy",
                    "wasAssociatedWith", "wasInvalidatedBy"}
    and time window 24:00:00 from 2013-03-16T00:00:00 starting T
```

- This says:
  - starting at each node with a PID, construct segments of up to 3 edges away, following usual PROV event edges
  - and split into 1-day time windows starting on March 16, 2013
- Variables `X`, `T` bound to PID and start time of segment, respectively.
- "and" means segment using two different strategies and combine using pairwise intersection

Validity of segmentation

- What do other components need / expect from segment layer?
- Acyclicity: helpful but not necessary
- Can view segment layer as an "index" or "summary" of full graph
- (We made progress on agreeing on this since finalizing paper
    - this was part of the goal of writing up in early stage...)

Conclusions and next steps

- Work on segmentation (and rest of system) in progress *as we speak*
- Future work:
    - Using (Titan/Gremlin) graph queries to extract segments, instead of in-memory processing?
    - Incremental segmentation: efficiently maintain segment layer as graph grows
    - Adaptive segmentation: can we learn good segmentation criteria?
    - What will we learn from running system?