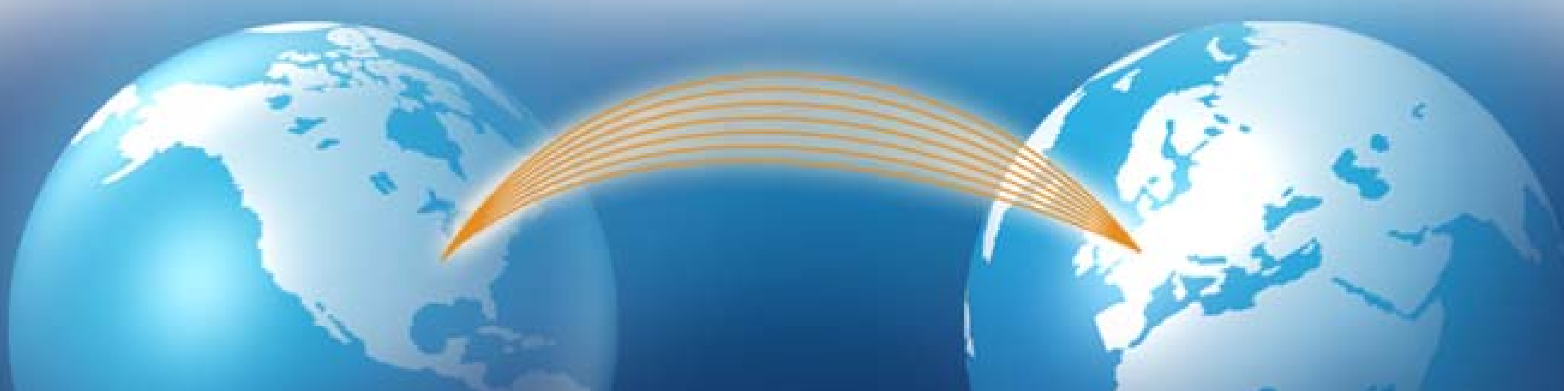
Distributed Event Correlation and Self-Managed Systems

J.P. Martin-Flatin, CERN, Switzerland
jp.martin-flatin@ieee.org
http://cern.ch/jpmf/



Self-Star 2004, Bertinoro, Italy, 2 June 2004

Outline

- Taxonomy
- Problem statement
- New organizational model
- Dynamic dependency graphs
- Research perspectives

Taxonomy

Proposed Definitions

- Self-organization:
  - Emergent properties (e.g., MAS in DAI)
  - Darwinism (only some emergent properties are good)
- Self-management:
  - Not externally managed
  - Local decision making
  - Includes self-configuration, self-monitoring, self-repair
- Self-configuration:
  - Can automatically retrieve its config (e.g., DHCP or Jini)
  - ...or can automatically adapt its config to new environment
- Self-repair (self-healing):
  - Can correct faults without intervention of 3rd party

Self-Managed System

- Autonomous (see robots or intelligent systems in DAI):
  - Can detect problems that occur within its subsystems (self-monitoring)
  - Can find the causes of these problems
  - Can correct these problems without the intervention of any third party (self-repair)
- Autonomous does not mean schizophrenic:
  - May receive orders and new configs from a trusted manager
  - May exploit data coming from an external source

Problem Statement

Self-Managed Systems: Examples

- Ad hoc networks & sensor networks:
  - Many nodes
  - Tough resource constraints:
    - limited power budget
  - Failures have a limited impact (e.g., 1 user)
  - Focus: self-configuration
- E-business servers:
  - Few nodes, possibly only one
  - Almost no resource constraints
  - Stringent service availability constraints
  - Focus: self-repair, high availability
  - Failures have a big impact (e.g., 1M users)

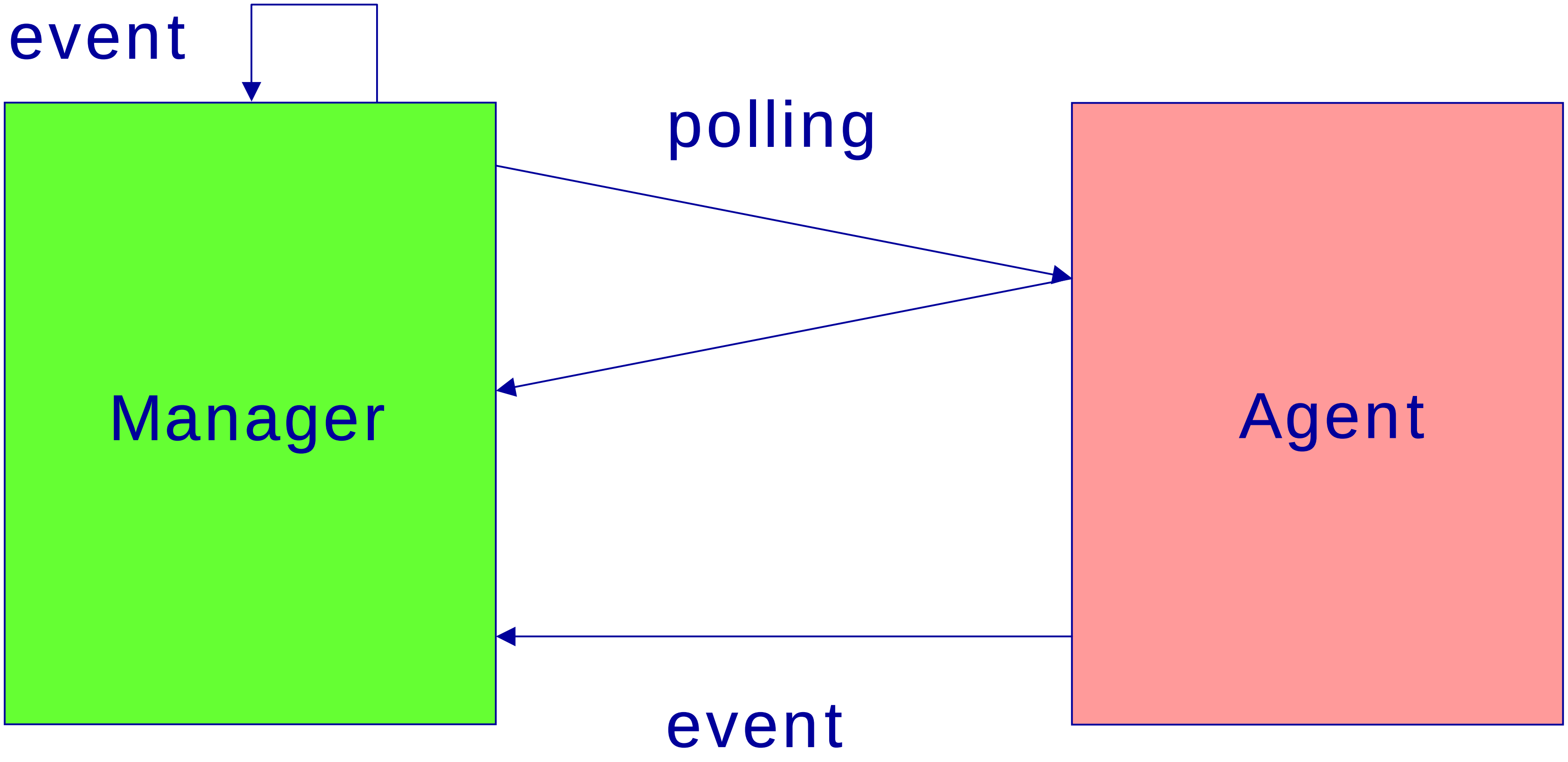
High Availability Can Be Offered by...

- Fault-tolerant systems:
  - Expensive hardware (redundancy)
  - Expensive software (niche market)
  - Expensive to operate (rare admin skills)
  - Guaranteed service
- Management systems:
  - Monitoring
  - Fault management (reactive)
  - Performance management + trend analysis (proactive)
  - Less expensive
  - Best effort service

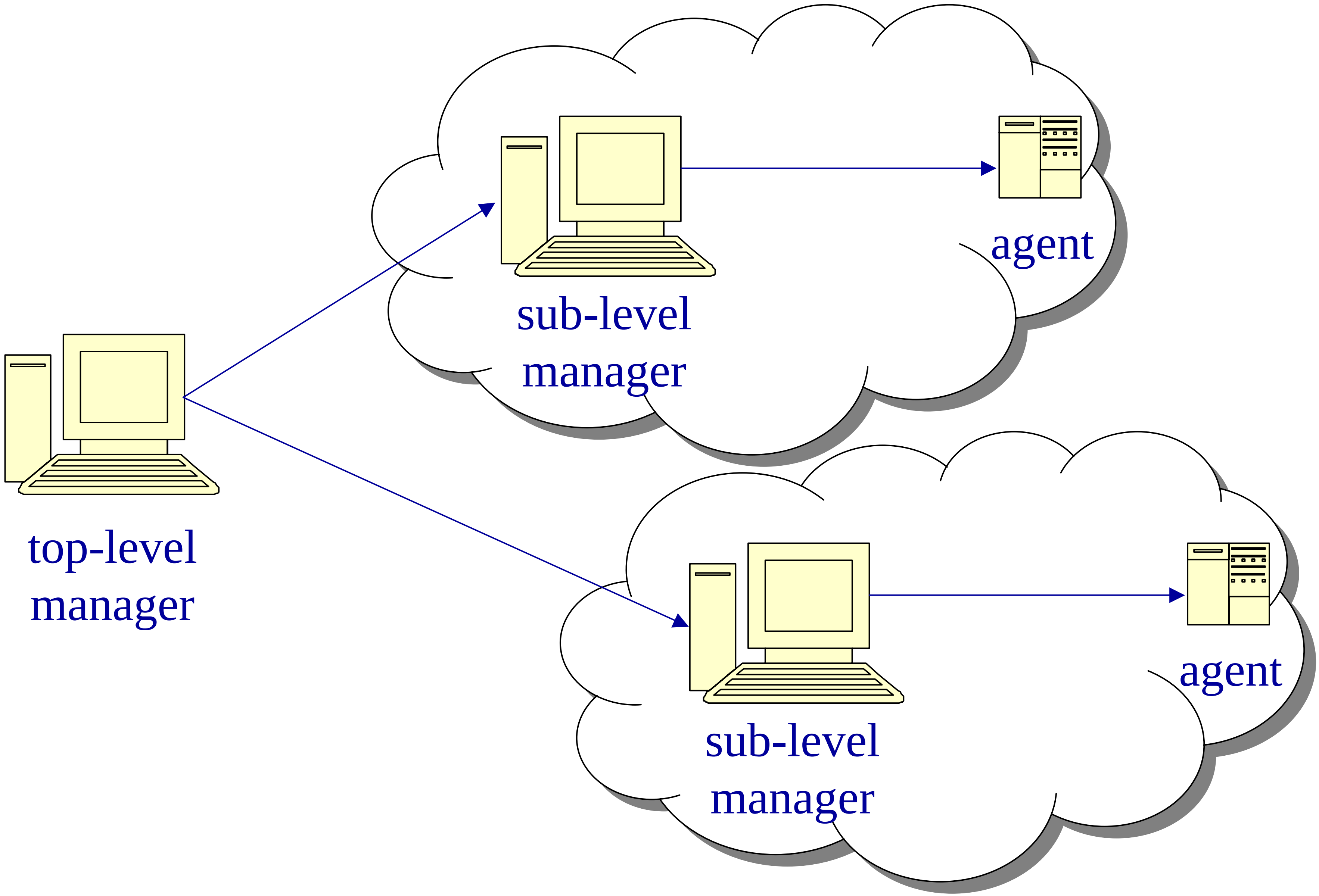
Problem

- High service availability constraints are imposed on e-business servers
- Today's management systems are inadequate to meet these constraints

Manager-Agent Paradigm



Domain-Based Hierarchical Distribution



Event Correlation

- Smart part of the management application
- Objective:
  - Identify current problems and their causes:
    - root cause analysis
  - If possible, solve these problems automatically:
    - e.g., trigger execution of scripts
  - Otherwise report them:
    - warn administrator, log to DB, etc.
- Integrated event correlation:
  - The same correlator can deal with network events, systems events, application events, service events, etc.
  - Linked to trouble-ticket system or helpdesk (user complaints)

Event Correlation Techniques

- State transition graphs (FSMs)
- Rule-based reasoning
- Case-based reasoning
- Model-based reasoning
- Binary coding (codebooks)
- Probabilistic dependency graphs:
  - Bayesian networks
  - Belief networks
- Neural networks
- ...

Where Are Event Correlators Located?

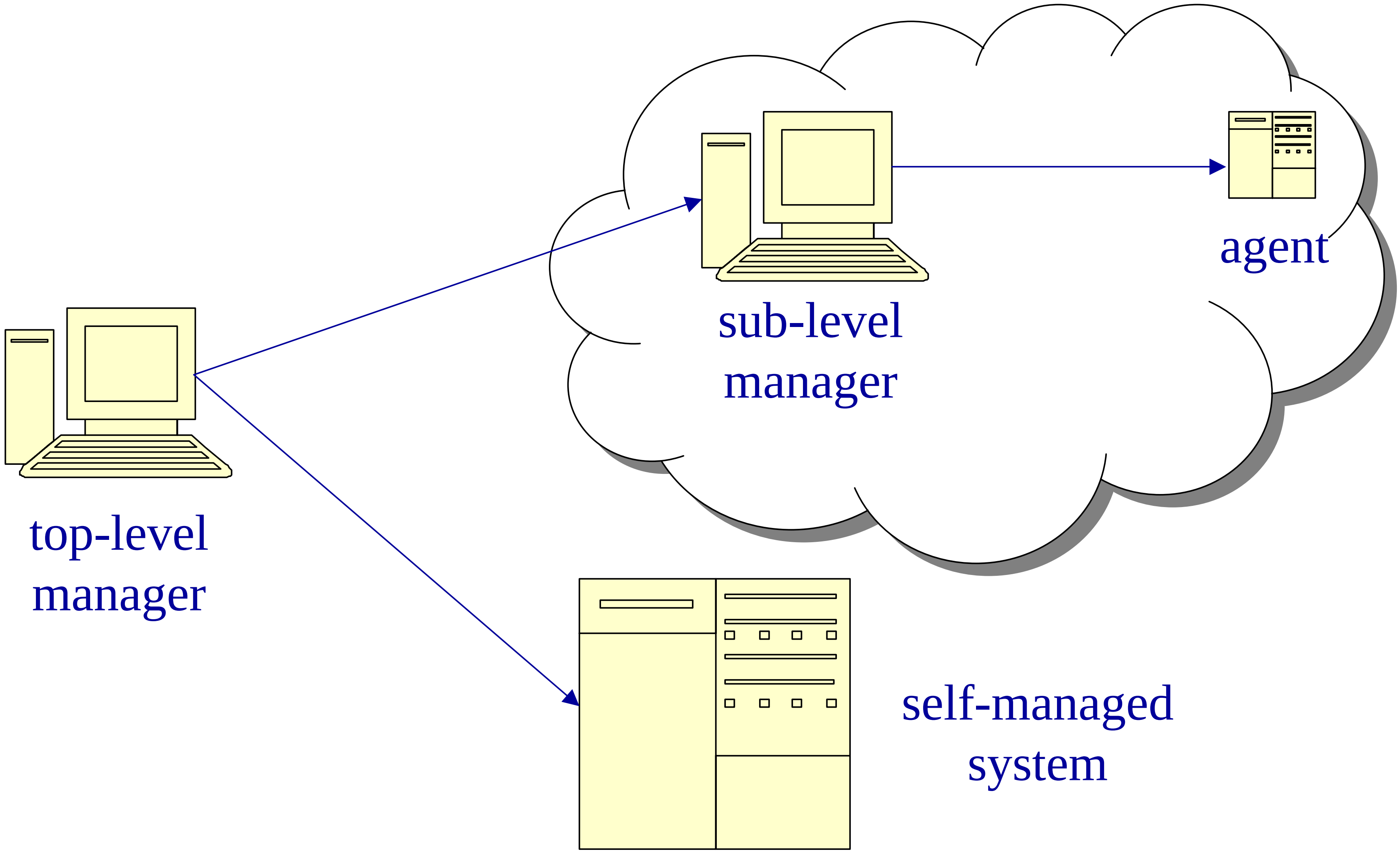
- Centralized event correlation:
  - one correlator for the entire organization
- Distributed event correlation:
  - one correlator per domain

What Is Wrong with E-Business Servers?

- Rules of thumb for defining a domain:
  - At most a few 100 agents
  - 1 to 10 OIDs per agent are polled every 10-15 min
  - At most a few 1000 OIDs to monitor per poll cycle
  - Most agents do not send events
  - Event correlation takes less than 10-15 min
- E-business application server:
  - Multiple machines, sometimes geographically dispersed
  - Usually 1000s of OIDs to monitor
  - The domain manager cannot cope with the work load
  - Latency becomes too large to guarantee SLAs
  - Network overhead may also become too large

New Organizational Model

One Correlator per Self-Managed System



Advantages

- SLM relieved of a lot of work
- If not geographically distributed:
  - Polling is done locally, so:
    - Internal problems remain internal: no network overhead imposed on external machines
    - We can afford to do more polling over short periods of time
  - Events are generated locally, so it is easier to cope with event bursts:
    - Event delivery time is shorter
    - Marshalling and unmarshalling can be reduced

Dynamically Built Dependency Graphs

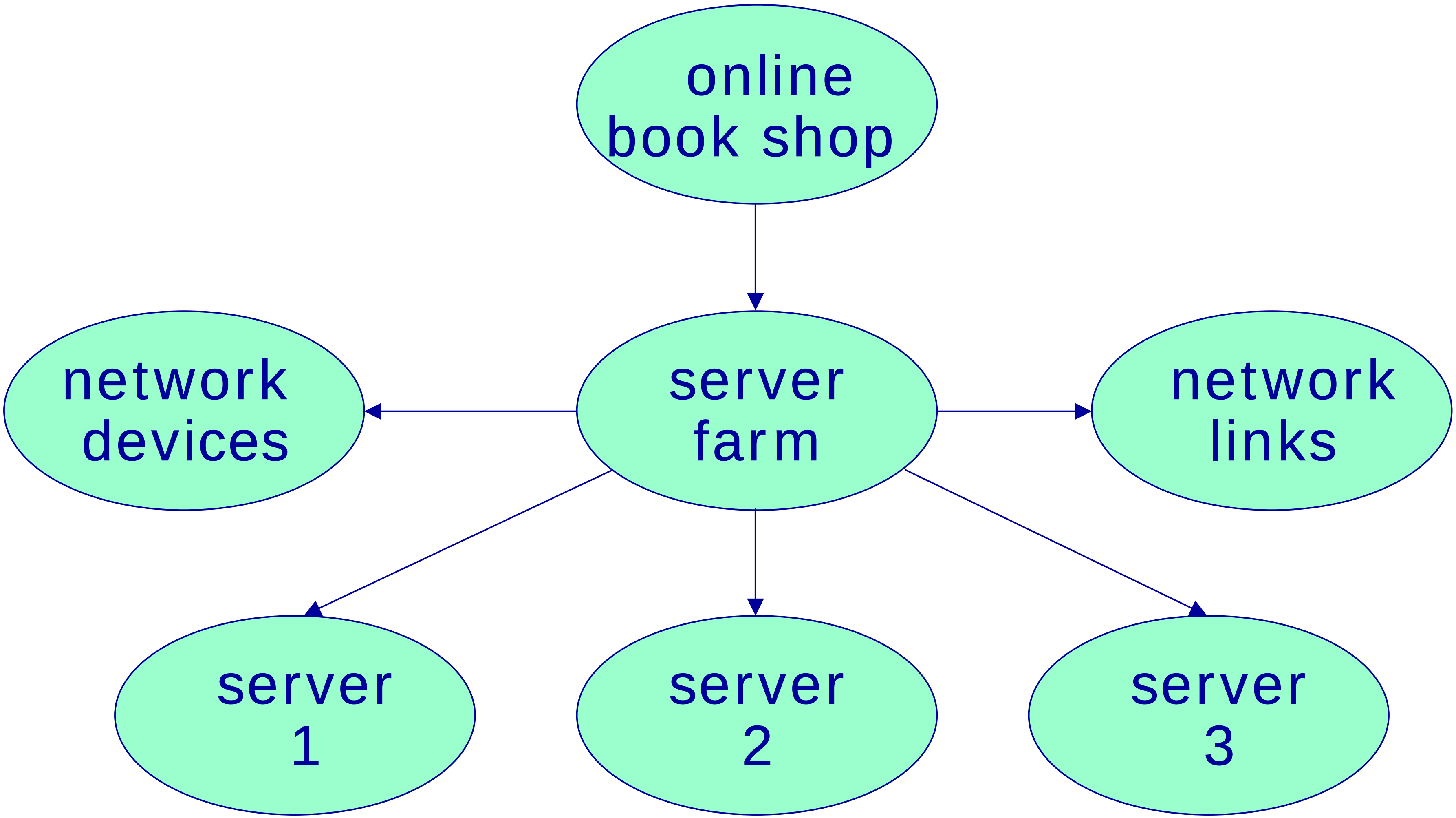
Dependency Graphs

- Capture dependencies between:
  - subsystems (coarse-grained)
  - components (fine-grained)
- Hardware or software components
- Some dependencies are static (long-lived):
  - e.g., C program compiled for a given CPU on a PC
- Some dependencies are dynamic (short-lived):
  - e.g., route between two hosts when we use a dynamic routing protocol and load-balancing across multiple paths
- Some dependencies are less obvious than others...

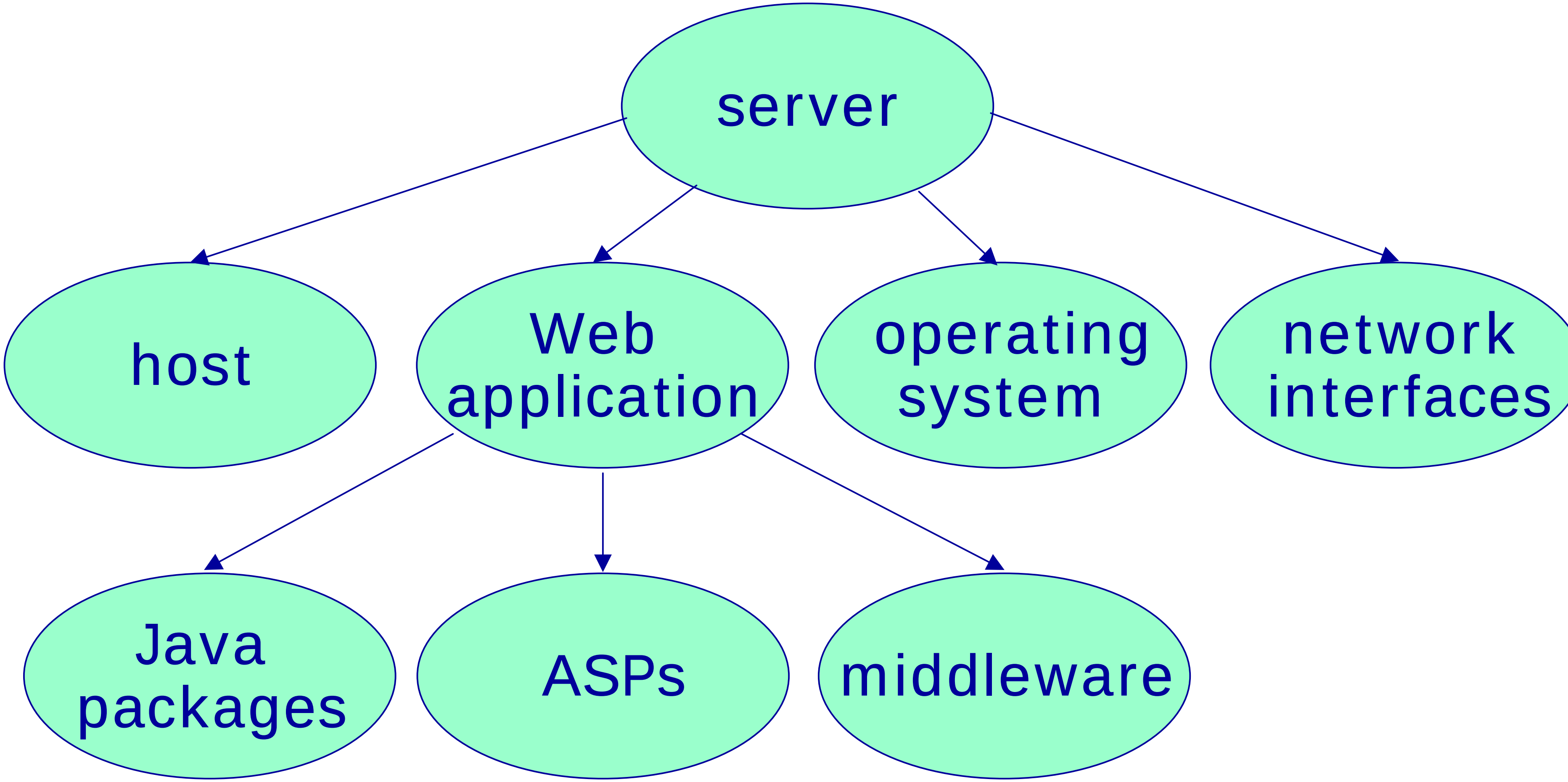
Advantages of Dynamic Refinement of Dependency Graph (1/2)

- The self-managed system need not keep permanently up-to-date:
  - A complete model of the world
  - A complete dependency graph
- While progressing in its investigation of a problem, the self-managed system can:
  - Poll its subsystems and components for more info on an ad hoc basis
  - Refine its model of the world
  - Refine its dependency graph

Partial Dependency Graph



Dependency Graph Refinement



Advantages of Dynamic Refinement of Dependency Graph (2/2)

- The self-managed system can discover new dependencies on-the-fly:
  - Useful for capturing dynamic dependencies
  - Difference between expected and totally unexpected dependencies
- The self-managed system can temporarily switch a component into debugging mode to get very detailed info:
  - Not possible with external correlators
  - Requires pre-existing instrumentation in hardware/software components

Disadvantages

- We lose the end-to-end vision:
  - Nothing changed if the clients of the e-business server are outside the management domain
  - Big change if they are inside
- The self-managed system depends on external monitors for end-to-end monitoring data:
  - e.g., SLA monitoring

Conclusion

Summary

- E-business servers should be self-managed
- Local polling and notifications reduce overhead and latency
- Event bursts are easier to cope with
- Dependency graph refinement allows for adaptation
- Realistic framework for capturing dynamic dependencies

Directions for Future Research

- Learn on-the-fly what partial dependency graph should be constantly kept up-to-date
- Automate (fully or partially):
  - refinement of dependency graphs
  - discovery of dynamic dependencies
- Find better algorithms for distributing event correlation:
  - Work on disjoint sets of data