Middleware Management

19 March 2007

Mike Spreitzer, Gosia Steinder, Chunqiang Tang, Constantin Adam, Asser Tantawi , Wolfgang Segmuller, Ian Whalley

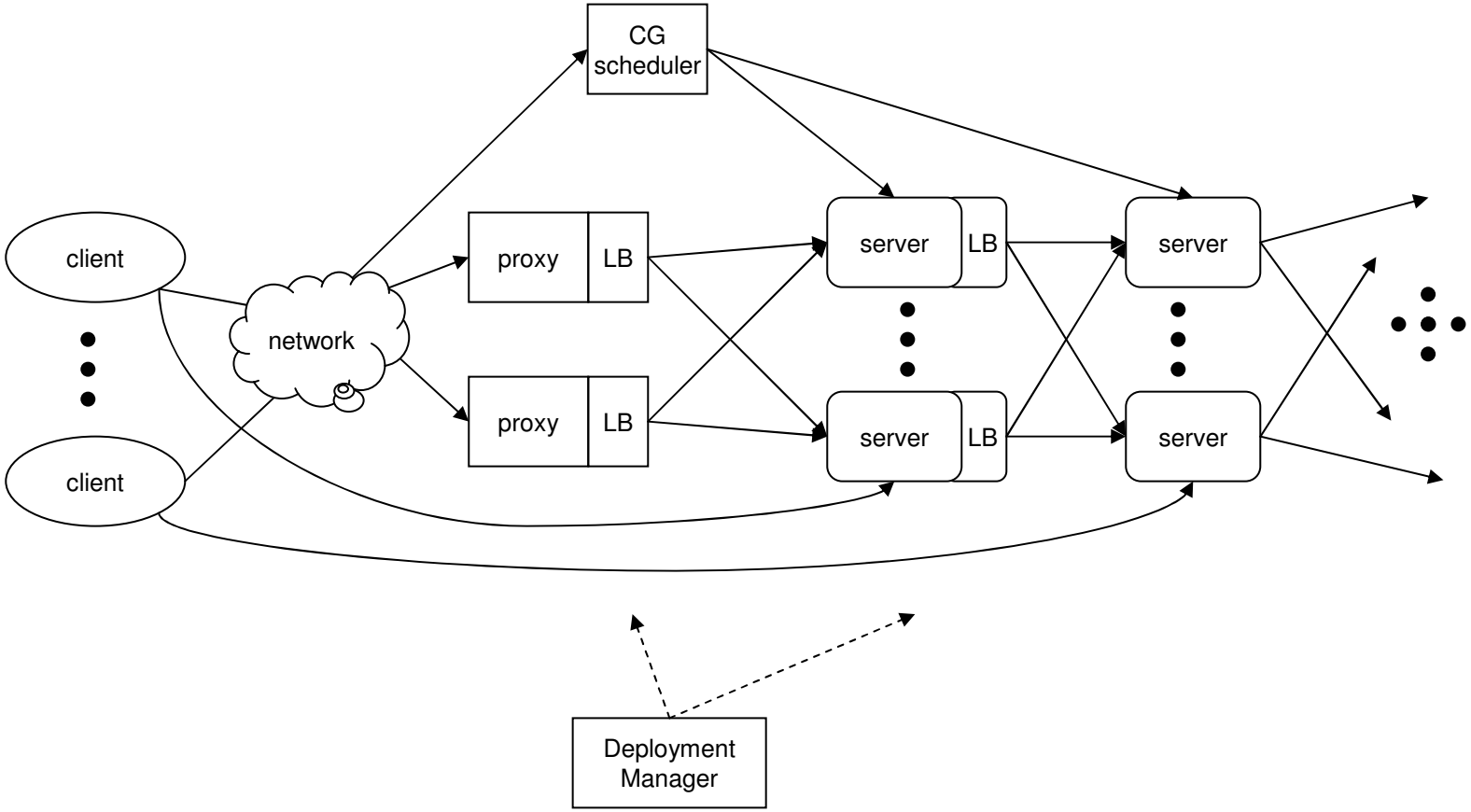
Overview

- Describe some real enterprise middleware
- Identify distributed systems problems, technologies in that middleware
- There is much more to it than messaging
- It is not all based on group communication
- Discuss usage of P2P technology in this middleware

Some Middleware Supports..

- HTTP[S]
- SIP[S]
- JMS, other Message Queuing APIs
- GIOP (CORBA)
- Object Grid (OG): in memory RDB
- Compute Grid (CG): long-running jobs
- Data-based partitioning for HTTP, OG, DB
- Layered products: portal, workflow, commerce, …

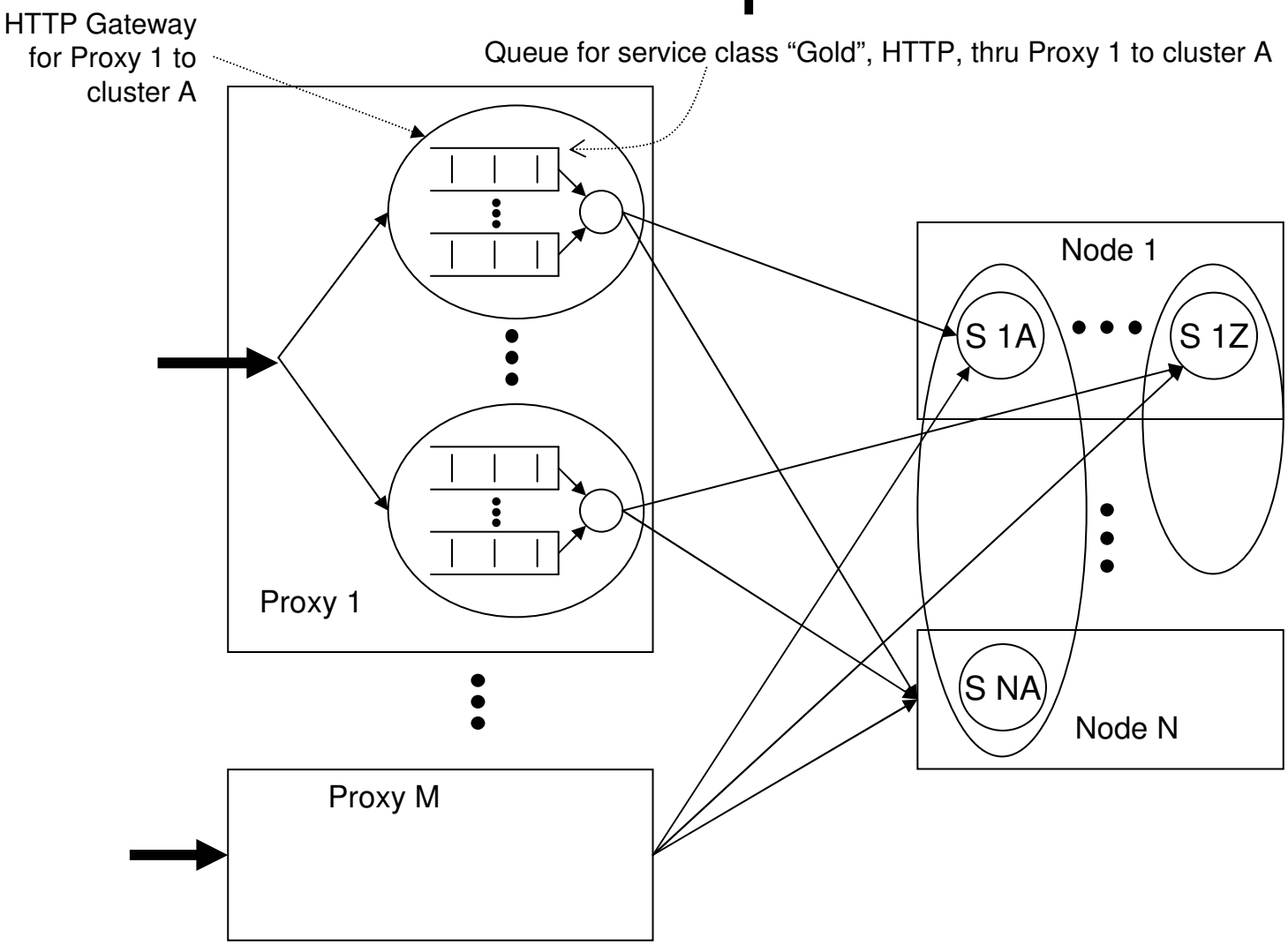
Topology


CG
scheduler
client
network
proxy
LB
proxy
LB
server
LB
server
LB
server
server
client
Deployment
Manager

About Clusters

- A “cluster” is a set of processes
  - .. that have the same user code deployed
- A given piece of user code is generally deployed to one cluster
  - Exceptions: mulitple editions, geographic distribution
- A cluster can have many different pieces of user code deployed
  - .. but they usually all serve the same “application”
- Typical large system: many small clusters

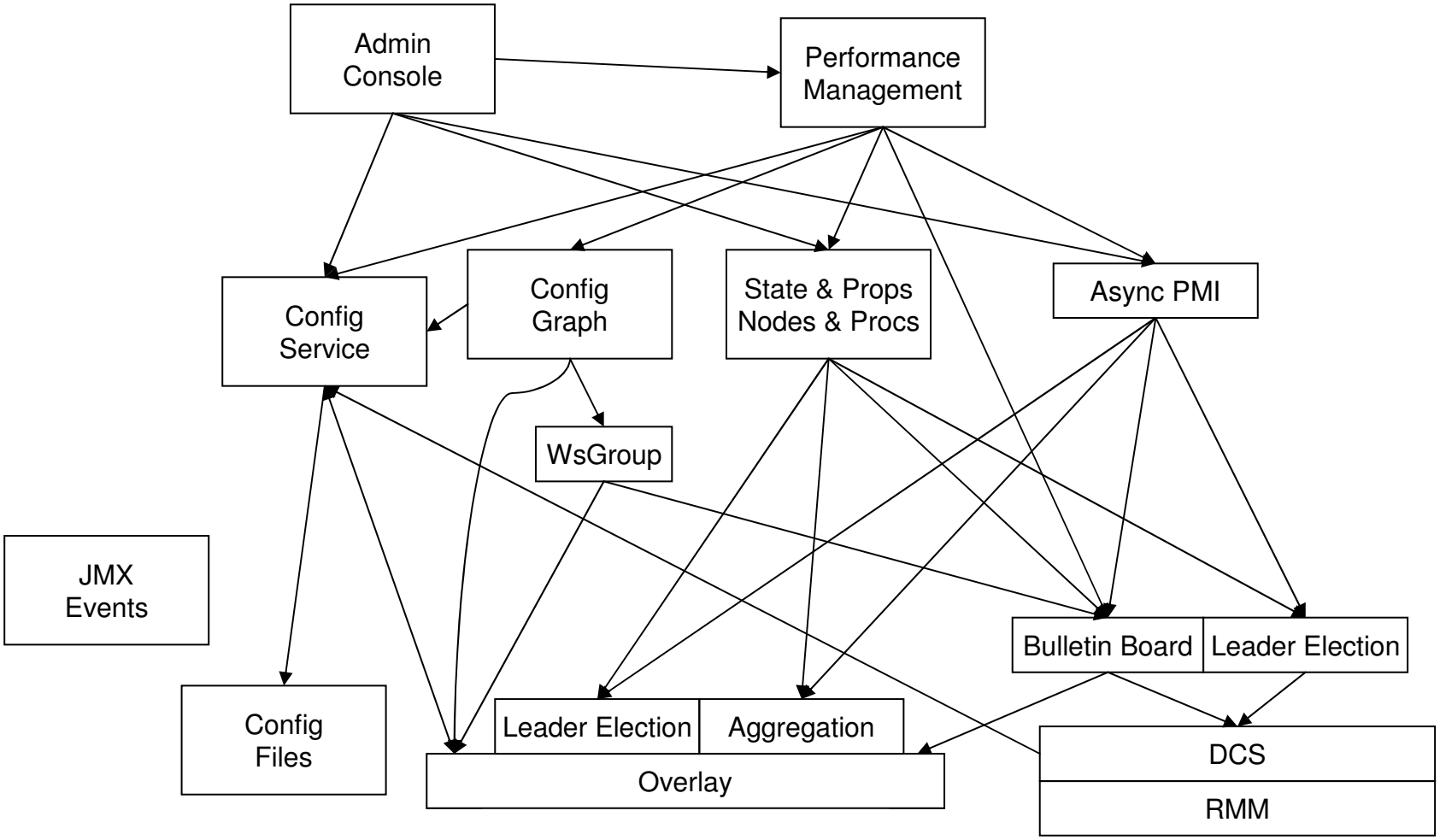
HTTP Request Flows



Some Infrastructure

- Confguration
- Administration Console (a web app)
- JMX events
- state & system properties of nodes & processes
- Async PMI: wide variety of performance metrics
- Bulletin Board: write/subscribe distributed memory
- Generic data gather+aggregate
- Leader election
- DCS/RMM
- random, nearly uniform degree, graph overlay

Layering (Past, Present, Future)



Messaging vs. State

Pub/Sub Messaging:

- Each subscriber is notified of each message published
- p.subscribe: □ (q.publish(msg) => ◊ p.deliver(msg))

Write/Sub Memory:

- Whenever a subscriber does not know the current value, it will eventually be notified of current value
- p.subscribe: □ (p.stale => ◊ p.update)
- Allows one notification to cover several writes

Generic Data Gathering with Aggregation

- Given
  - a query Q(process)
  - a commutative and associative function ○
  - p1, p2, … pN are the processes in the system
- Returns Q(p1) ○ Q(p2) ○ … Q(pN)
- Query is flooded, building an on-demand tree
- Replies are collected and combined along the tree
- Spatial, temporal redundancy makes it robust

Performance Management

- Three feedback control loops + on-line profiler
- Placement Controller: start/stop processes
- [d]WLM: load balancing with dynamic weights
- MFM: admission control & queuing at entry for overload protection & differentiated service
- Profiler: estimates CPU work for each kind of request (using a certain categorization)

Service Classes

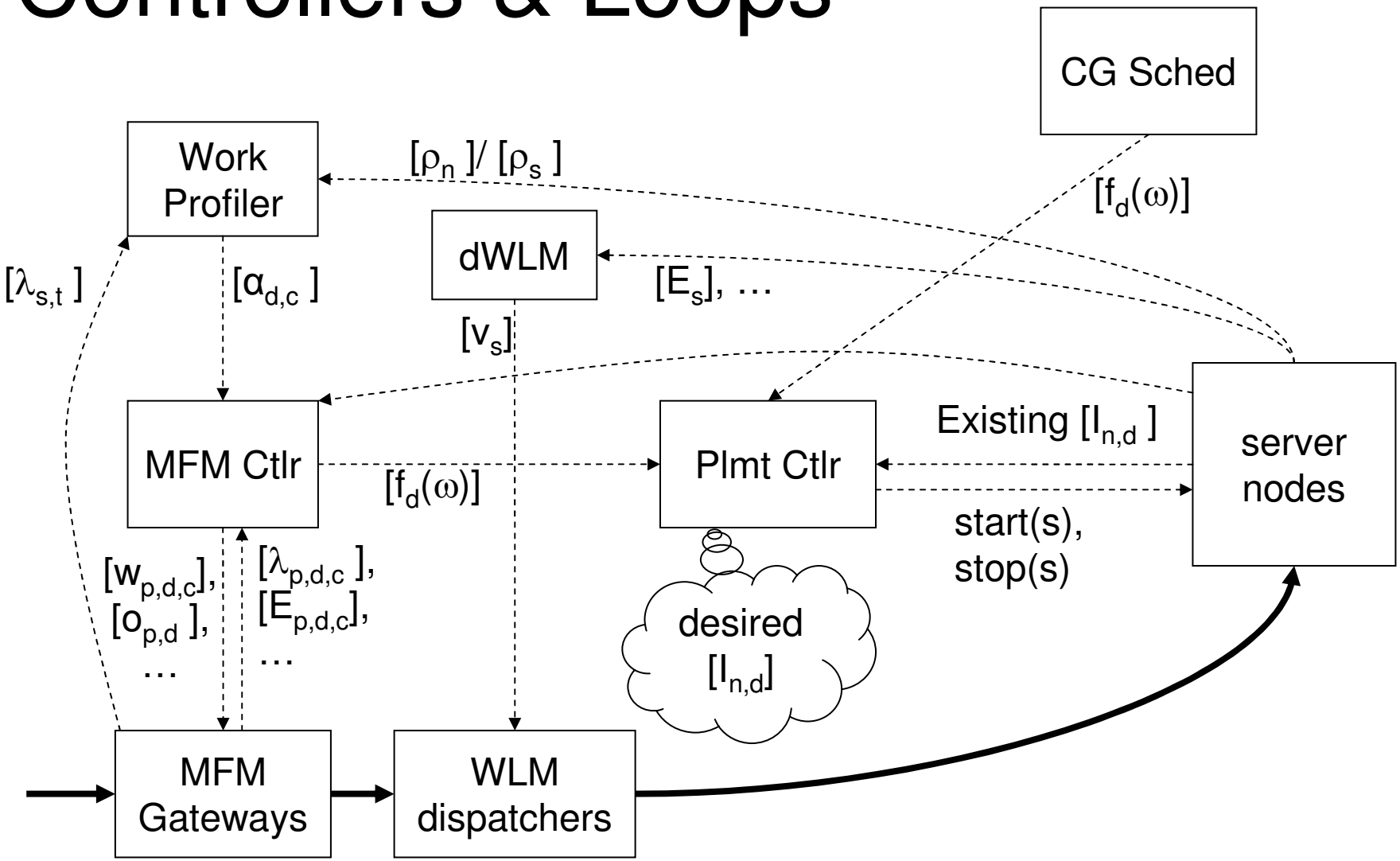
- A service class has two parts:
  1. a set of incoming messages, grouped into transaction classes
     - (which may involve multiple deployment targets);
  2. a goal, consisting of
     - a type (discretionary, average response time, or percentile response time),
     - value(s) (if relevant), and
     - an importance (if relevant), in the range 1 (most important) to 99 (least important).

Management Objectives

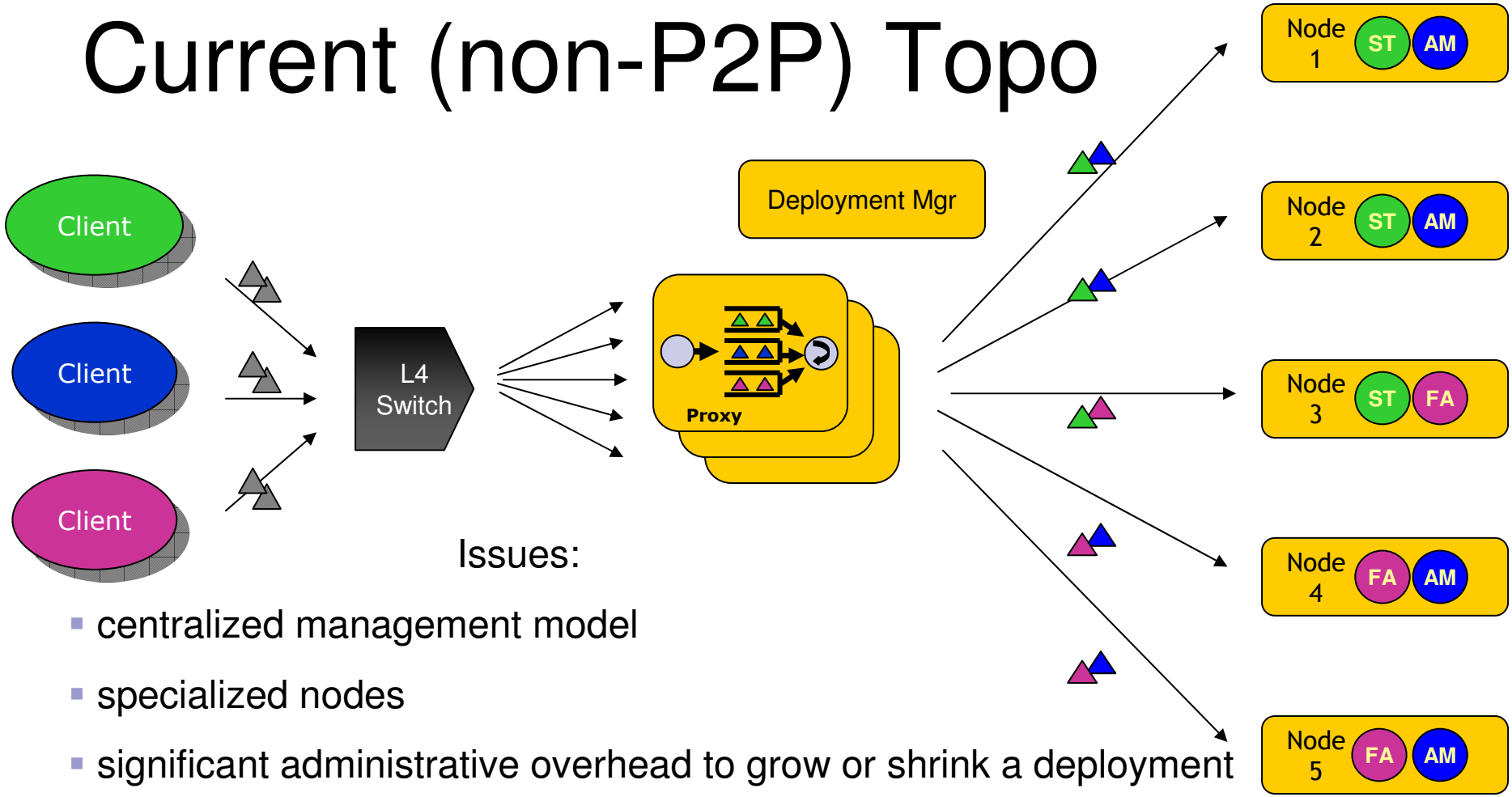
- Fairness: equalize utility of predicted performance
- Maximize (for non-CG):

  min [protocol+proxy p, entry cluster d, service class c] $U_c(Perf_{p,d,c})$
- MFM controls flow given current placement, sends utility↔power relationships to Plmt Ctlr
- Similar story for Compute Grid
- Plmt Ctlr takes utility↔power relationship per cluster, other params, updates placement
- Independently for each cluster, dWLM equalizes service times among the servers in the cluster

Controllers & Loops



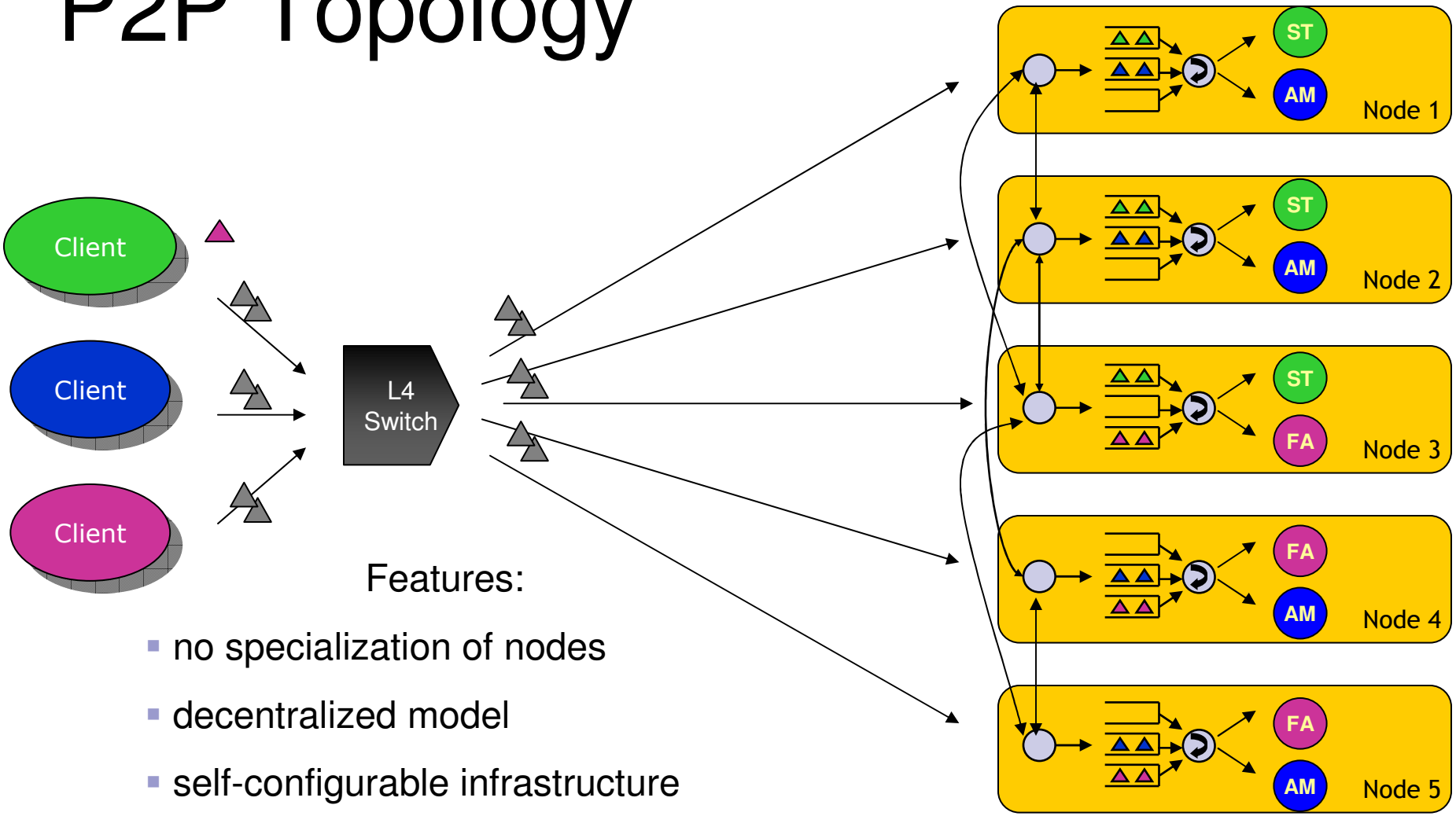
Current (non-P2P) Topo



Issues:

- centralized management model
- specialized nodes
- significant administrative overhead to grow or shrink a deployment
- not scalable communication infrastructure
- centralized and not scalable performance management controllers

P2P Topology



Features:

- no specialization of nodes
- decentralized model
- self-configurable infrastructure
- scalable architecture and management

Decentralized Routing & Placement

- "A Service Middleware that Scales in System Size and Applications", Constantin Adam et. al., IM 2007
- Random, nearly uniform degree, graph overlay
- Each node holds a fixed amount of info for each cluster:
  - k processes, based on distance and freshness of information
- An update is propagated until it makes no difference
  - (rather than being reliably broadcast)
- Each node independently will periodically adjust the set of clusters it runs, based on limited information

Bibliography

- "Dynamic Application Placement for Clustered Web Applications"; A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, A. Tantawi; 15th Int'l World Wide Web Conf. (WWW) 2006
- "A Scalable Application Placement Controller for Enterprise Data Centers"; Chunqiang Tang, Malgorzata Steinder, Michael Spreitzer, Giovanni Pacifici; 16th Int'l World Wide Web Conf. (WWW) 2007
- "A Decentralized Application Placement Controller for Web Applications"; C. Adam, G. Pacifici, M. Spreitzer, R. Stadler, M. Steinder, C. Tang; IBM Research Report RC 23980, June 2006
- "Managing the Response Time for Multi-tiered Web Applications"; Giovanni Pacifici, Wolfgang Segmuller, Michael Spreitzer, Malgorzata Steinder, Asser Tantawi, Ian Whalley; IBM Research Report RC23942 (2006)
- "Managing the Response Time for Multi-tiered Web Applications"; Giovanni Pacifici, Wolfgang Segmuller, Michael Spreitzer, Malgorzata Steinder, Asser Tantawi, Alaa Youssef; IBM Research Report RC23651 (2005)
- "Dynamic Estimation of CPU Demand of Web Traffic"; Giovanni Pacifici, Wolfgang Segmuller, Mike Spreitzer, Asser Tantawi; 1st Int'l Conf. Perf. Evaluation Methodologies and Tools (VALUETOOLS 2006)
- "GoCast: Gossip-enhanced Overlay Multicast for Fast and Dependable Group Communication"; Chunqiang Tang, Rong N. Chang, Christopher Ward; 2005 Int'l Conf. on Dependable Systems and Networks (DSN'05)