IBM Research

# Bulletin Board: A Scalable and Robust Eventually Consistent Shared Memory over a Peer-to-Peer Overlay

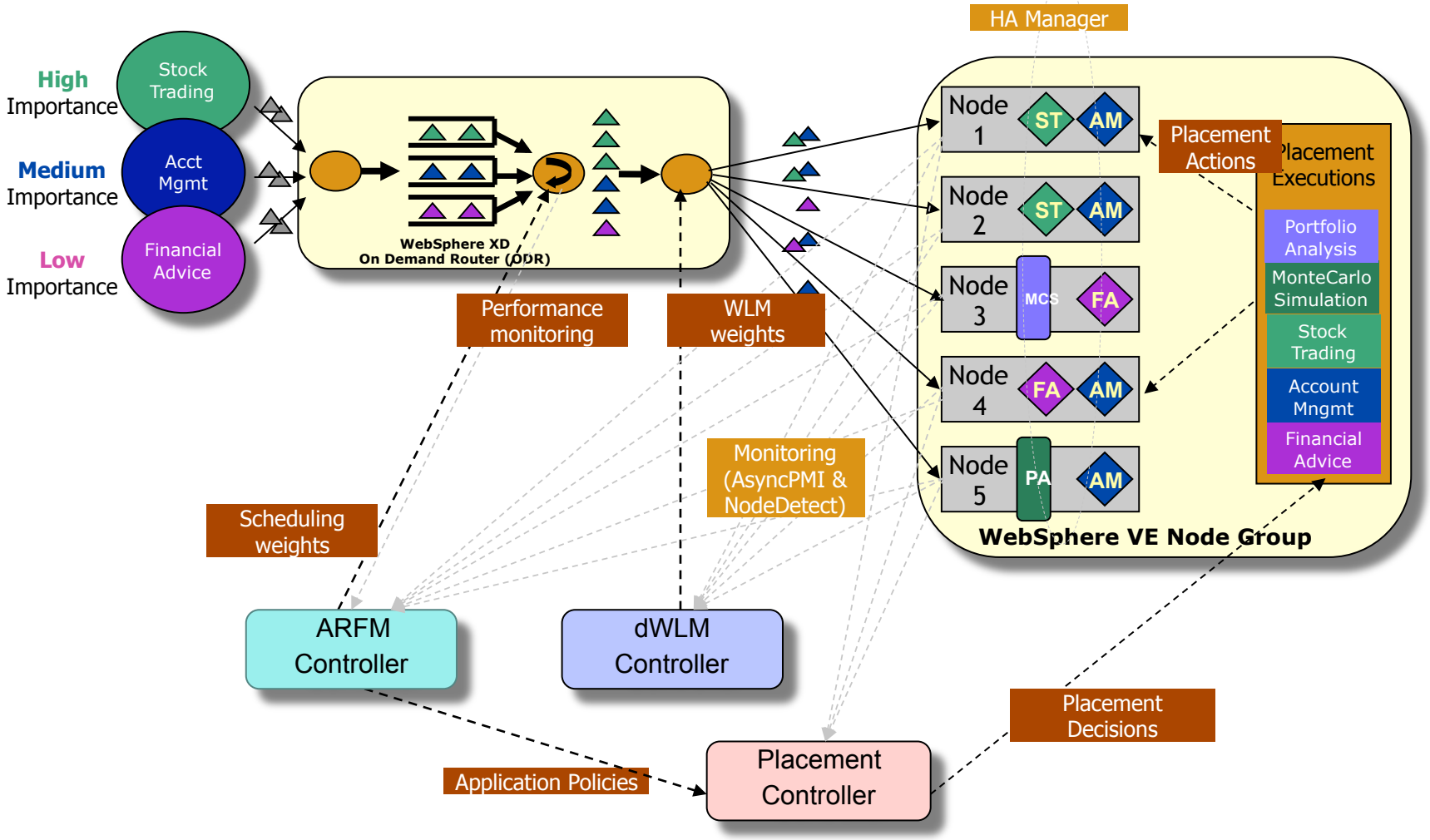
Vita Bortnikov

*Gregory Chockler*

Alexey Roytman

Mike Spreitzer

# Background: Resource Management in WebSphere Virtual Enterprise (WVE)



# What is Bulletin Board?

Platform service for facilitating group-based information sharing in a data center

- Critical component of WVE
- Scalable consistency model (≠ Inconsistent!)
- Primary application: monitoring and control
  - Useful for other weakly consistent services as well

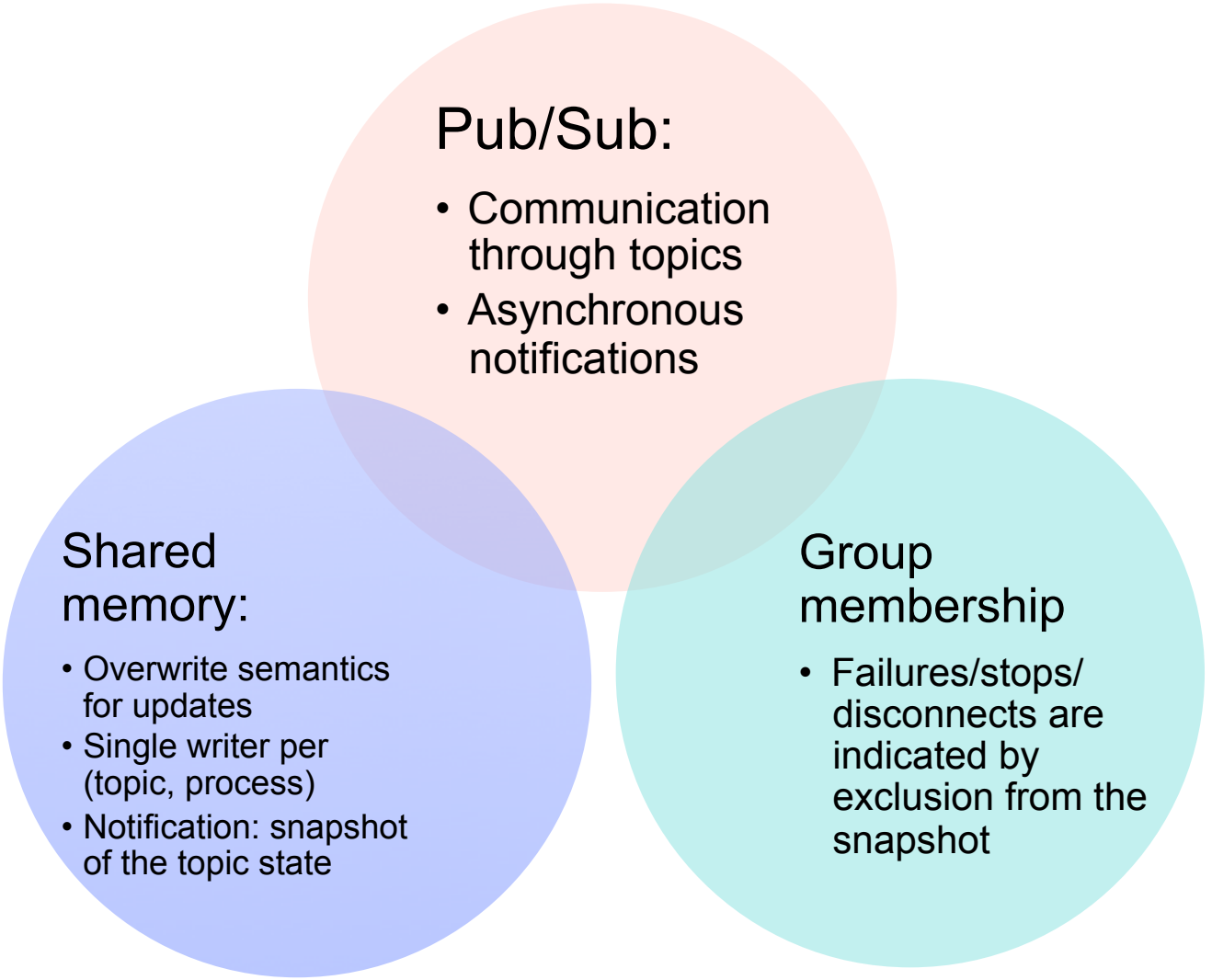
# Motivation and Contribution

- Prior implementation was not designed to grow 10X
  - Based on Virtually Synchronous group communication
  - Robustness, stability, high runtime overheads as the system grew beyond several 100s processes
  - Static hierarchy introduced configuration problems

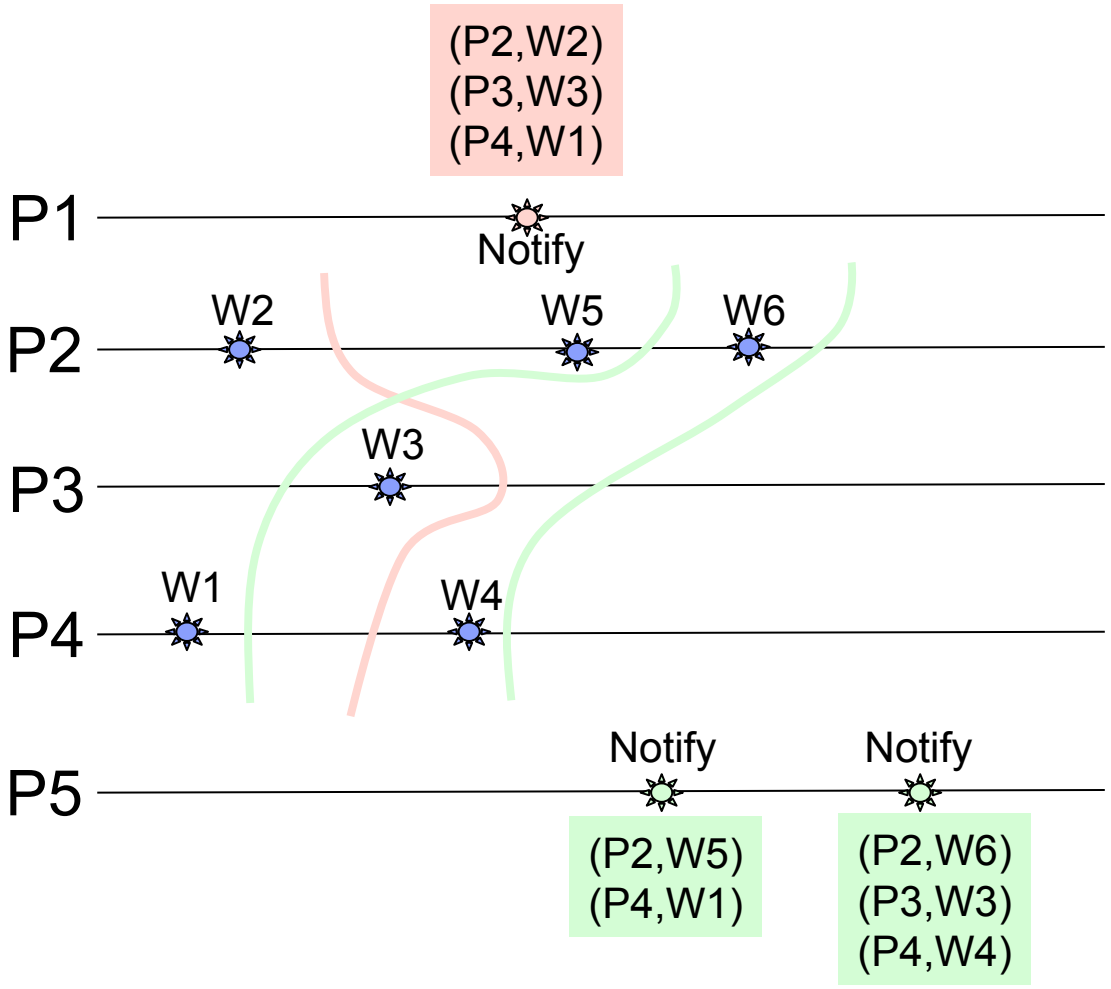
Our goal: Provide a new implementation to resolve the scaling and stability issues of the prior one

*within a short time…*

# Write-Sub Service Model



# Consistency Semantics (single topic)




**PRAM Consistency:**

Notified snapshots are consistent with the process order of writes

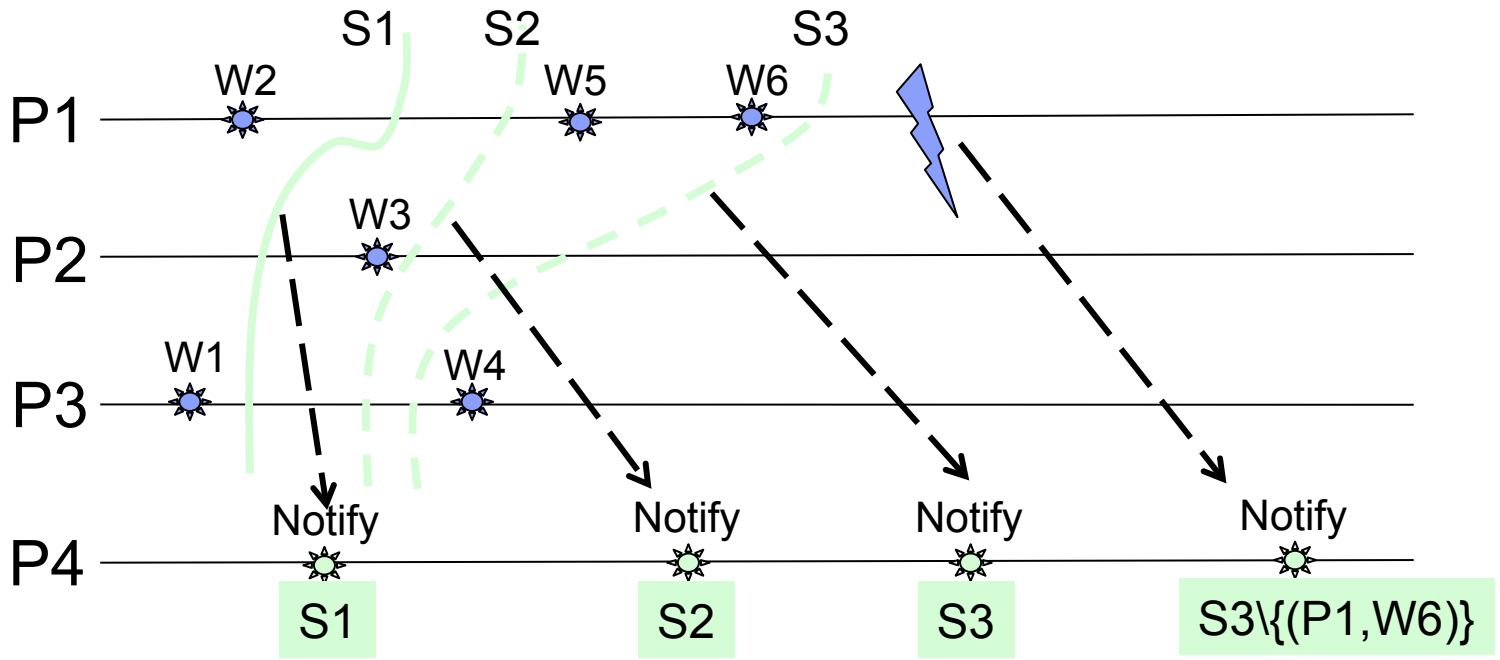
# Liveness Semantics (single topic)

**Eventual Inclusion:**

Eventually each write by a correct and connected process is included into the notified snapshot

**Eventual Exclusion:**

Eventually each permanent failure or disconnect is detected and notified



# Performance and Scalability Goals

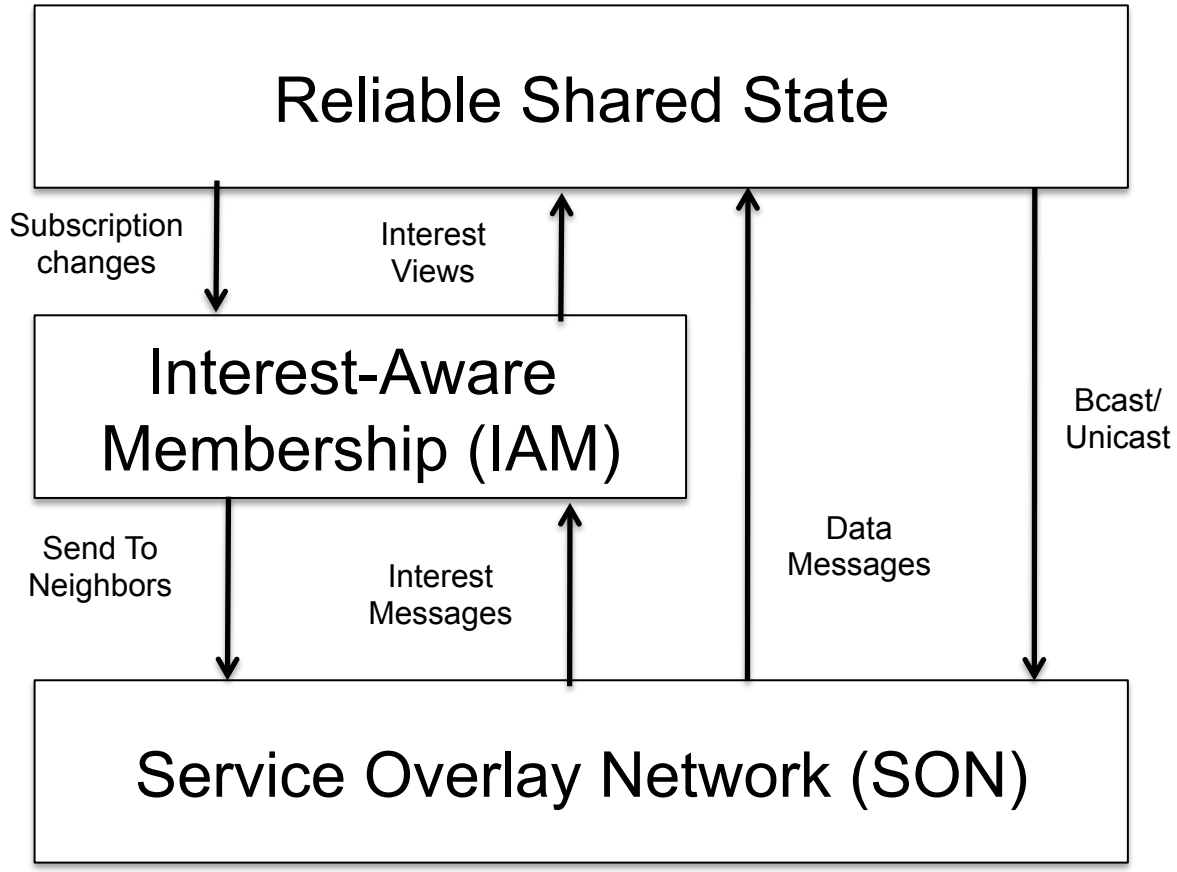
- Adequate latency, scalable runtime costs
  - Throughput is less of an issue (mgmt load is fixed)
- Low management/configuration overhead
- Robustness and resiliency
- Scalability in the presence of large number of processes and topics
  - 2883 topics in a system of 127 processes
  - Initial target ~1000 processes

# Approach

- Leverage Service Overlay Network (SON)
  - Semi-structured P2P overlay
  - Already in the product
  - Self-*, resilient
  - Supports peer membership and broadcast

The research question:
Can BB semantics be efficiently supported on top of a P2P overlay technology?

# Architecture



# Reliable Shared State Maintenance

- Fully decentralized
- Update propagation
  - Optimized for bimodal topic popularity
  - Overlay broadcast or iterative unicast over direct TCP connections

$$|Subscribers(T)| <> Threshold(N)$$

  - Message coalescing and compression to reduce costs

# Reliable Shared State Maintenance

- Reliability
  - Periodic refresh of the latest written value (on a long cycle) if not overwritten
  - State transfer to new/reconnecting subscribers
- Ordering
  - Simple timestamp-based mechanism
  - Individual records are garbage collected based on views, aging, and process incarnations (epochs)

# Experimental Study

- Studied CPU utilization, CPU cost per unit of work, latency
  - Unit of work: (write, matching subscription) pair
- Workloads were captured from the real product runs and replayed on a standalone BB to isolate CPU costs
- Studied topologies: 75, 147, 215, and 287 processes
  - Run on up to 4 machines, 16 cores/machine
- Focus on the cruise phase: light application load, stable connectivity/subscriptions
  - ~10 minutes

# Experimental study

| topology | core-ms/pair | CPU % | lat. $< 1$ sec |
|---|---|---|---|
| $2 \times 35 + 5$ | 1.3 | 0.47% | 100% |
| $3 \times 47 + 6$ | 2.1 | 0.84% | 99.998% |
| $4 \times 52 + 7$ | 2.9 | 1.22% | 99.9% |
| $4 \times 70 + 7$ | 5.2 | 2.92% | 94.7% |

Table 1: CPU Cost, Latency Distribution

# Impact of Refreshes and Broadcast Dissemination

| topology | core-ms/pair | CPU % |
|---|---|---|
| $2 \times 35 + 5$ | 0.78 | 0.27% |
| $3 \times 47 + 6$ | 0.74 | 0.30% |
| $4 \times 52 + 7$ | 0.90 | 0.38% |
| $4 \times 70 + 7$ | **1** | **0.57%** |

Table 2: CPU cost without refreshes **and without flooding**

# Lessons Learned

- Communication cost is the major factor affecting scalability of an overlay based implementation
- Efficient mechanisms for update reliability and propagation are needed
  - Anti-entropy is emerging as a promising approach

# More Lessons

- Flow control and overload protection are important even under low update rates
- Message compression is advantageous when applied to packets > size of the Ethernet frame
- Testing and debugging is a huge challenge

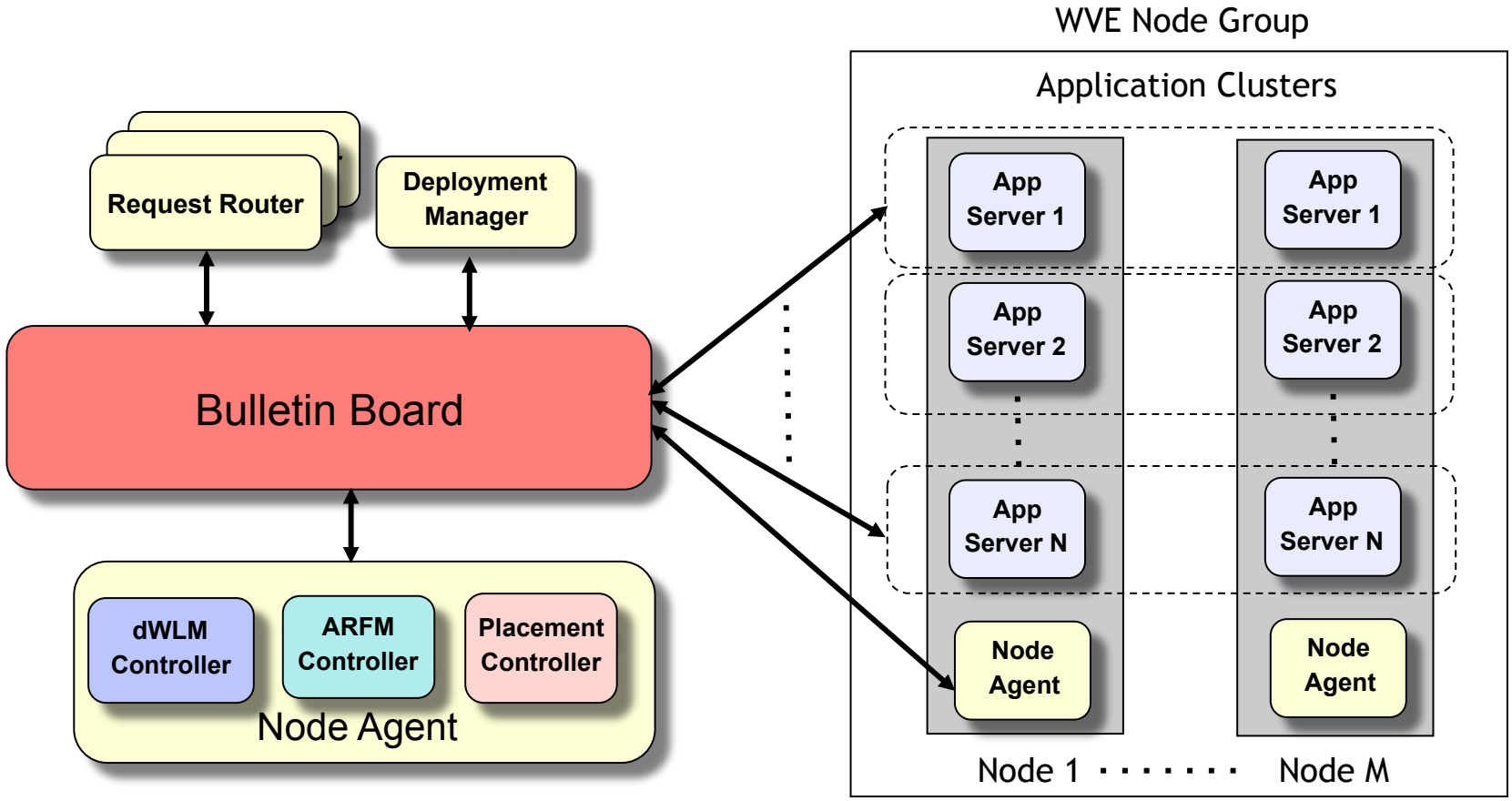
# Ongoing and Future Work

- Generic gossip anti-entropy layer on top of the overlay
  - IAM and Broadcast
- Large topologies
  - Hierarchical decomposition
- Flow control and overload protection
- Miscellaneous  efficiency improvements
  - Custom serialization, etc…

# Thank You!

# What is Bulletin Board?

- Communication substrate for sharing control and monitoring data among management controllers, agents, and application servers



# Non-Functional Requirements

- Performance: adequate for supporting management functionality at moderate scales
  - Low overhead, timeliness, scalability
  - Throughput is less important (mgmt load is fixed)
- Simplicity: management, configuration, implementation
- Robustness: dealing with high rates of dynamic changes in an autonomous fashion
  - Failures, network partitions, wedged processes, dynamic replacement of servers, growth of system

# Typical Workloads

- Process and communication failures, flaky processes are common
- Moderate rates of churn
- Large numbers of processes
  - Initial target ~1000
- Large numbers of topics, large subscription sizes
  - 2883 topics in a system of 127 processes
  - 100s subscriptions, 10s written topics per process
- Bimodal topic popularity

# Our Solution

- Peer-to-peer overlay network for basic connectivity maintenance
  - Self-organization, self-management in the presence of dynamic changes
- Can P2P overlays be leveraged to support scalable management of a shared state?
  - No prior work known to us

# The IAM Implementation

- Scalability in the number of topics, subscriptions
  - Each process WVE subscribes to 100s of topics
- Current implementation: anti-entropy of the Proc → Interest map along overlay edges
- Use of topic hashes instead of strings
- Still leaves room for scalability improvements

# Existing Approaches

**Shared state maintenance on top of a group-oriented messaging/membership services:**

- Virtually Synchronous group communication
  - ✔ Convenient programming model, strict consistency
  - ✘ Performance/stability problems at large scales
- Pub/Sub Bus: carefully configured backbone of message brokers
  - ✔ High-end QoS and performance guarantees
  - ✘ High admin/configuration overhead in dynamic systems
- IP Multicast
  - ✔ Low runtime costs (due to NIC offload)
  - ✘ Robustness, security problems in the presence of large number of groups

# Existing Approaches (contd.)

**Probabilistic shared state maintenance**

- ✔ Scalable and robust
- ✘ Lack sufficient determinism to meet the latency and reliability needs of an enterprise system

# The BB Service Model

**Write/Sub: Pub/Sub, Shared Memory, Group Membership**

- Pub/sub:
  - Communication through topics
  - Asynchronous notifications
- Group membership
  - Failures/stops/disconnects are indicated by exclusion from the snapshot
- Shared memory
  - Each write overrides the previously written value
  - Single writer per (topic, process)
  - Notification is a snapshot of the topic state