IBM Research

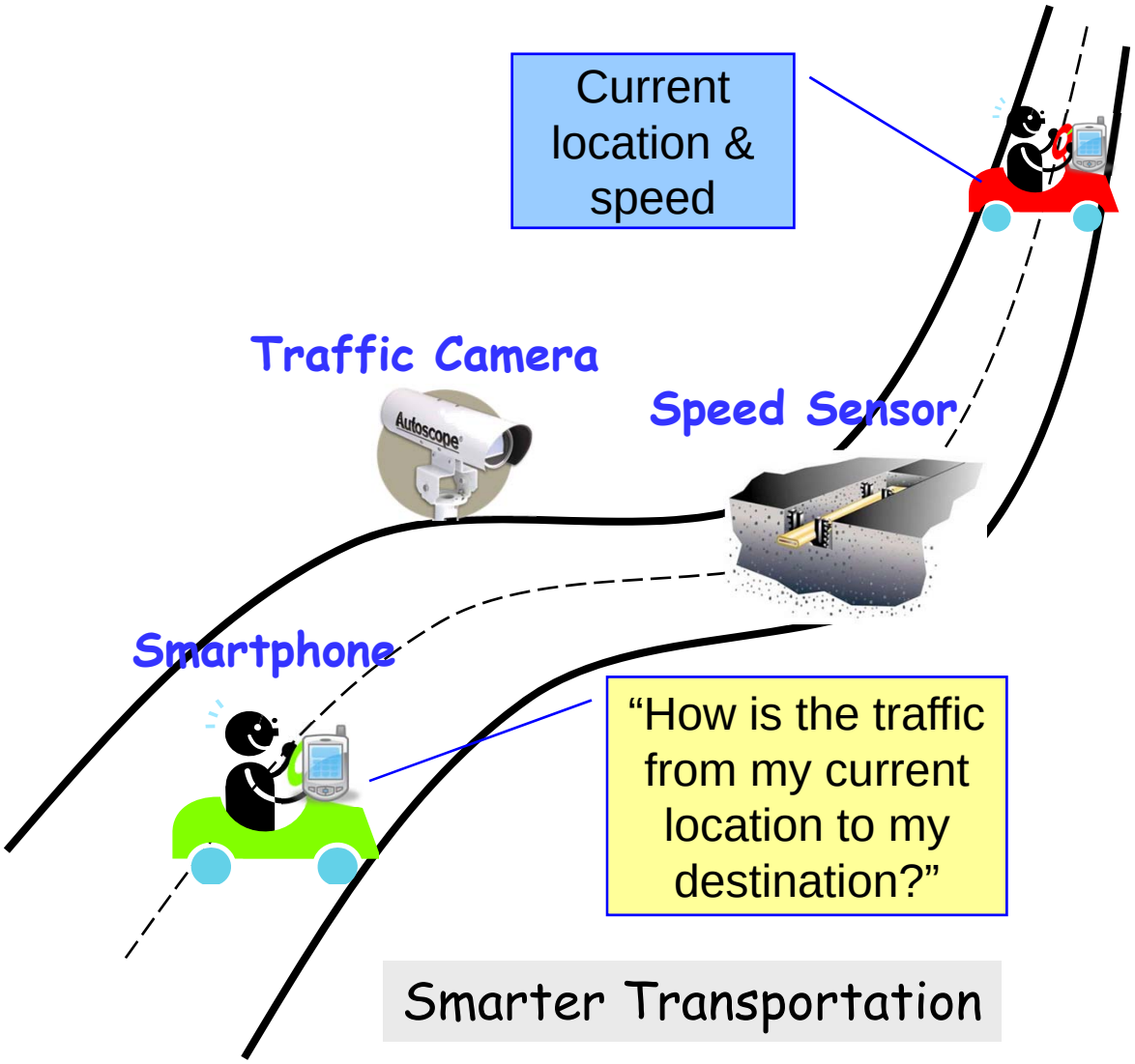
BlueDove Pub/sub

–A Scalable and Elastic Publish/Subscribe Service

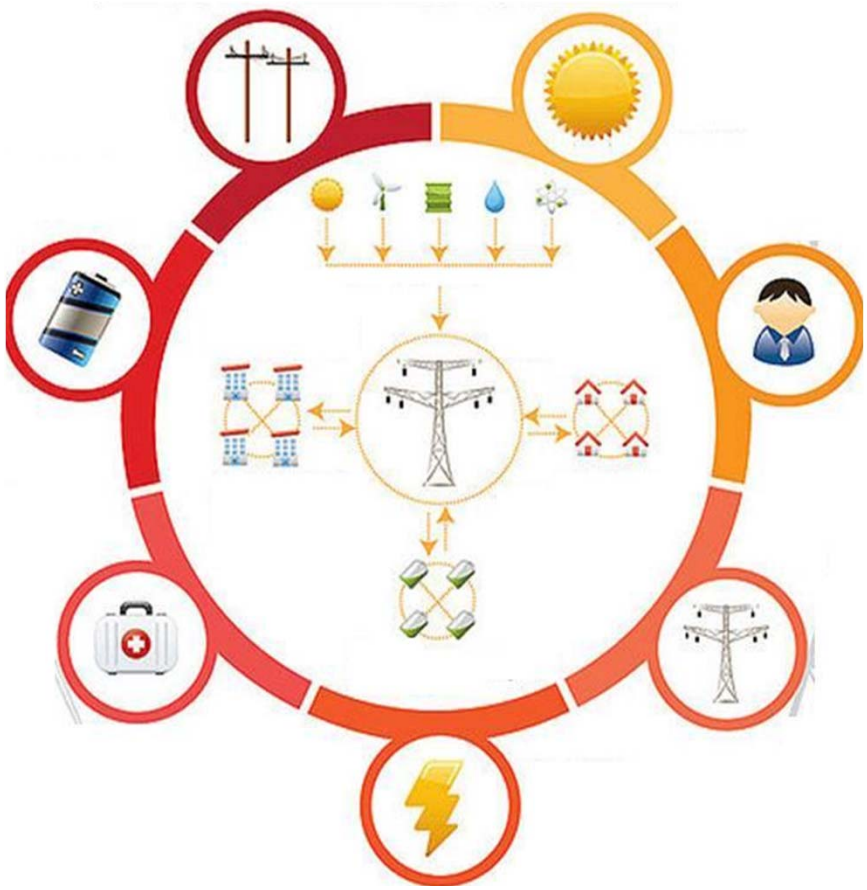
Fan Ye
IBM T. J. Watson Research Center

In collaboration with Ming Li, Minkyong Kim, Han Chen, Hui Lei

Emerging Smarter Planet Applications



Smarter Transportation



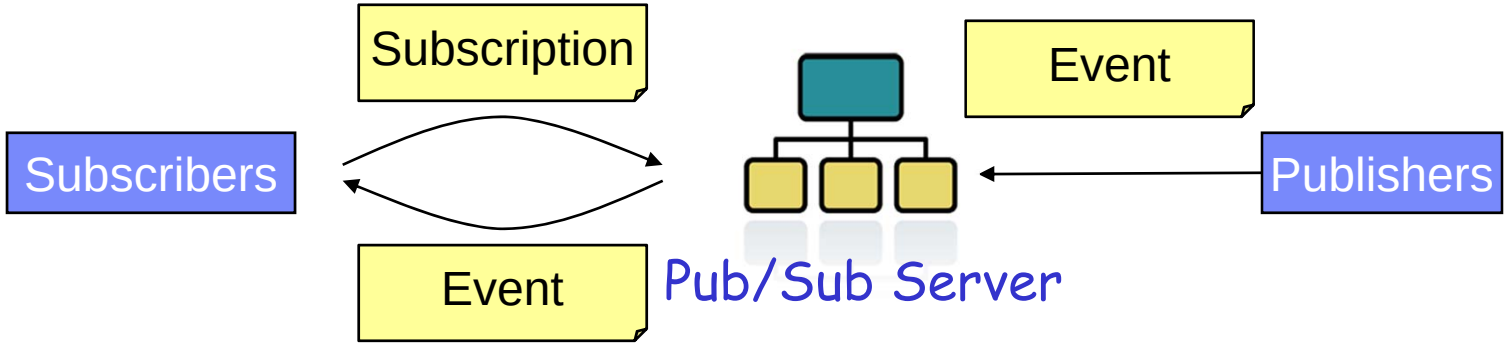
Smarter Grid

Requires highly efficient messaging service

Challenges to Messaging Service

- **Large number of clients, high information rate—Scalability**
- **Dynamic work load—Elasticity**
- **Personalized information—Real time fine-grained filtering**
- **Non-interrupt service—High availability**

General Publish/Subscribe Messaging Model

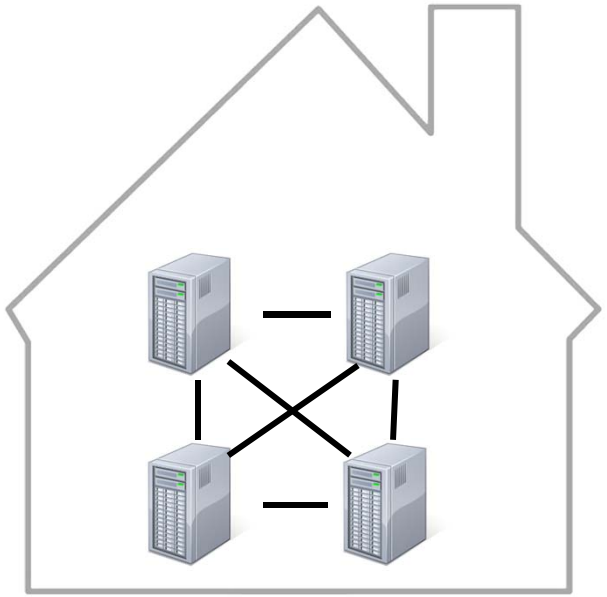


- Store subscriptions
- Match events to subscriptions
- Push events to subscribers

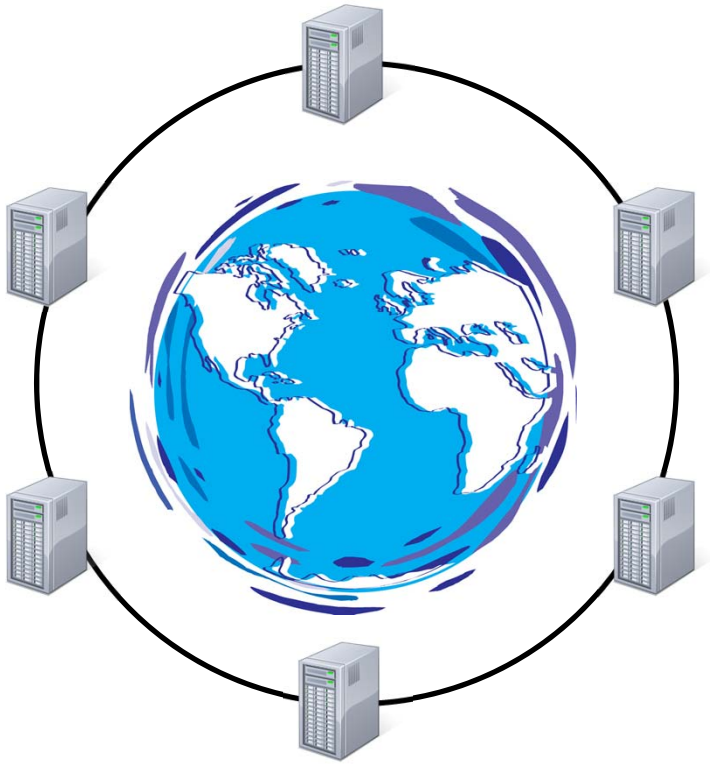
- **Decoupling of information producer/consumer**
- **Fine-grained event filtering**

Existing Pub/Sub Systems

- **Centralized enterprise pub/sub**
  - Multiple fully connected, dedicated cluster servers

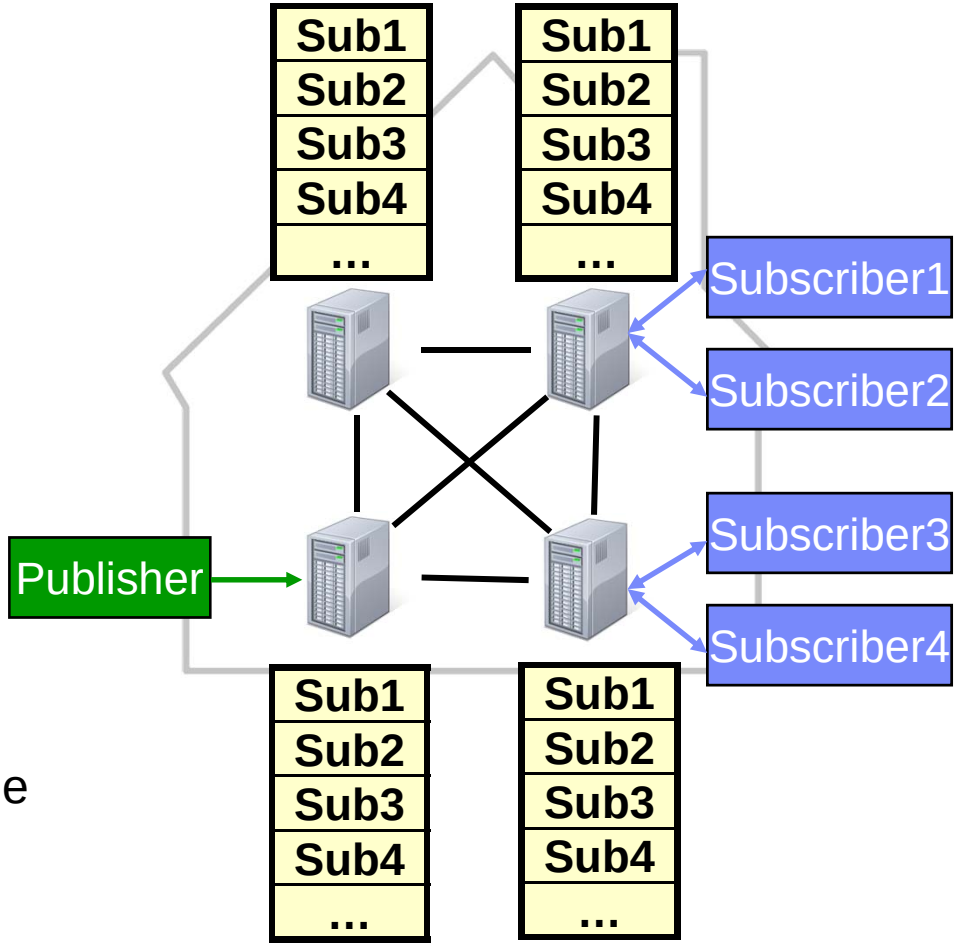


- **Peer-to-peer public pub/sub**
  - Large number of partially-connected, global-distributed machines
  - Meghdoot, Pastrystring, Hermes



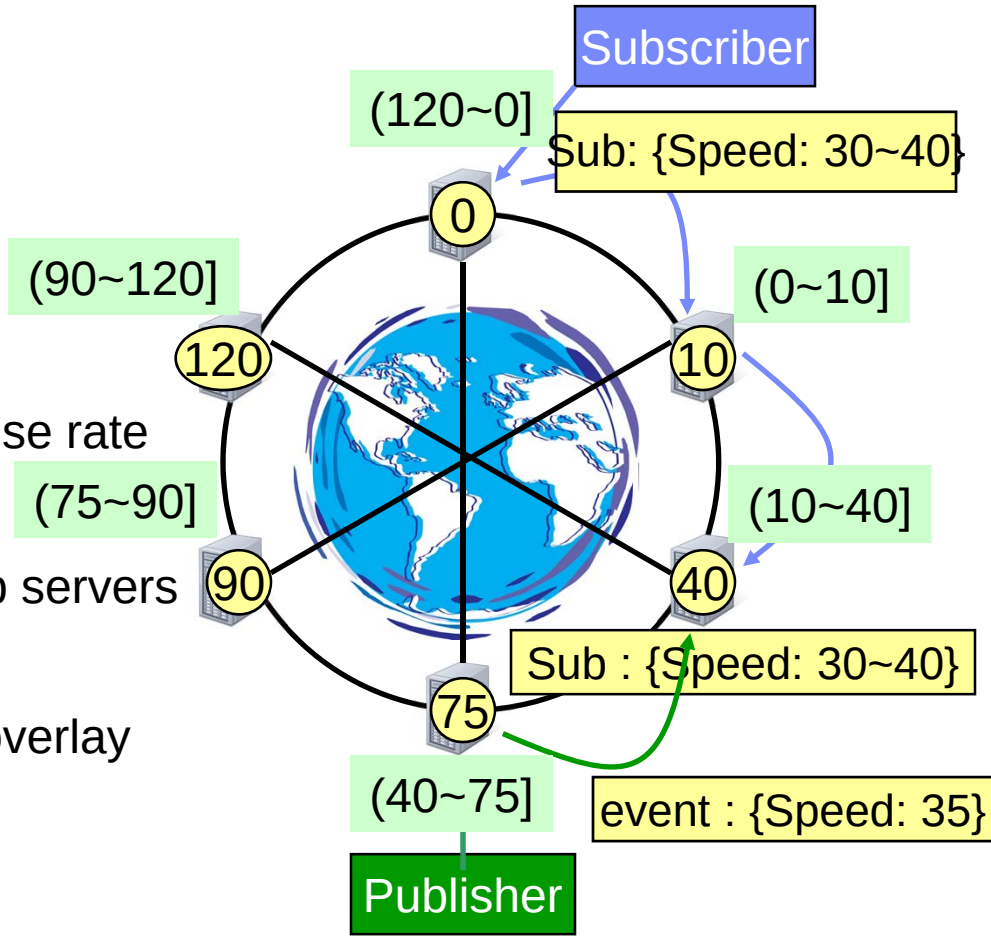
Centralized Enterprise Pub/Sub

- **Support enterprise applications**
  - Relatively static clients
  - Predictable work load
  - Strict requirement on response time
- **Design**
  - Centralized and relatively static servers
  - Fully connected topology
  - Subscriptions are replicated to all servers
- **Challenges**
  - Hard to provision for highly dynamic work load
  - Heavy administration when system scale is large

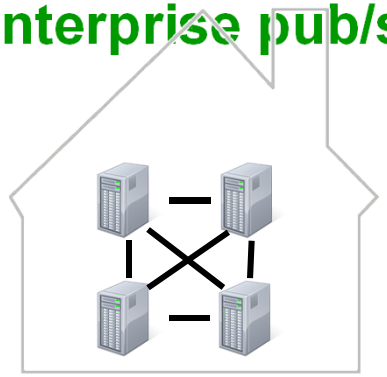


Peer-to-Peer Public Pub/Sub

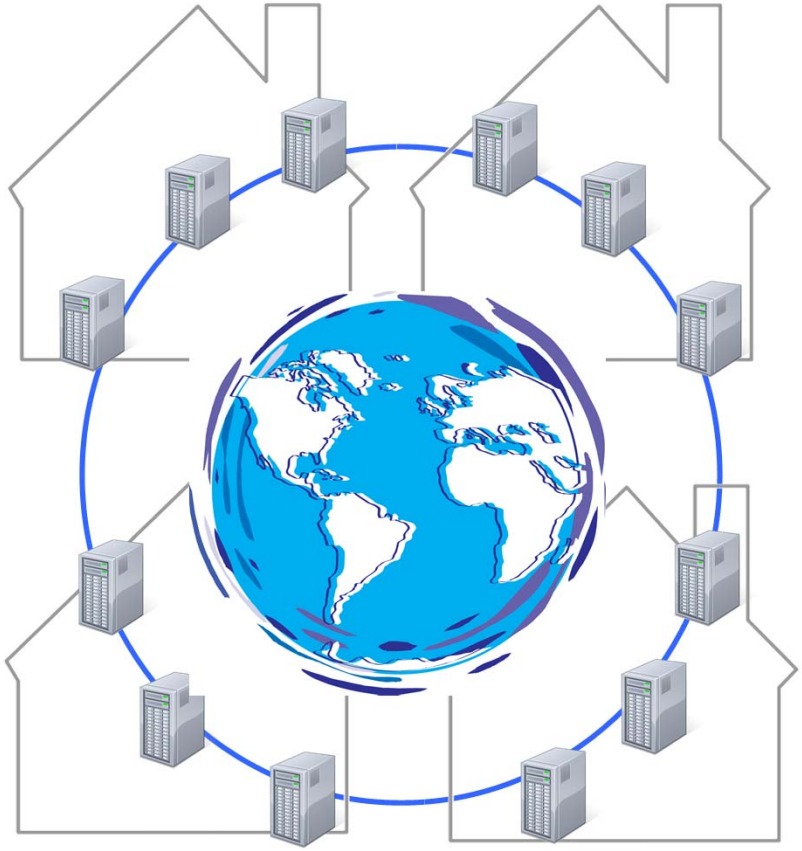
- **Support public applications**
  - Clients join or leave frequently
  - Highly dynamic work load
  - Flexible requirements on response time and false rate
- **Architecture**
  - Publishers and subscribers also act as pub/sub servers
  - Servers form a self-maintained logical overlay
  - Subscriptions/publications are mapped to the overlay
  - Routing are done in multi-hop manner
- **Challenges**
  - Bottlenecks due to skewed distribution
  - Performance issues due to dynamics and heterogeneity of servers
  - Large network delay



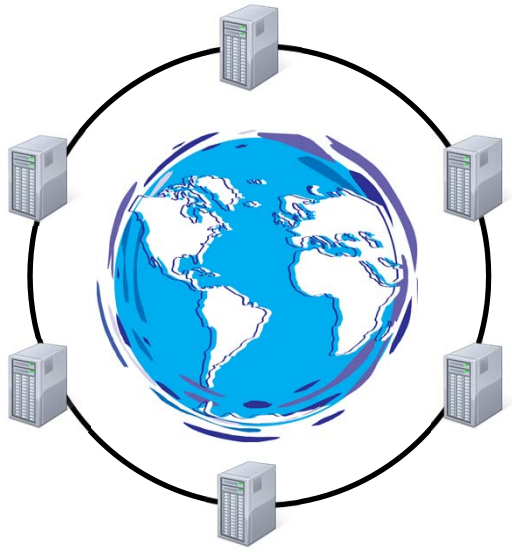
- **Centralized enterprise pub/sub**



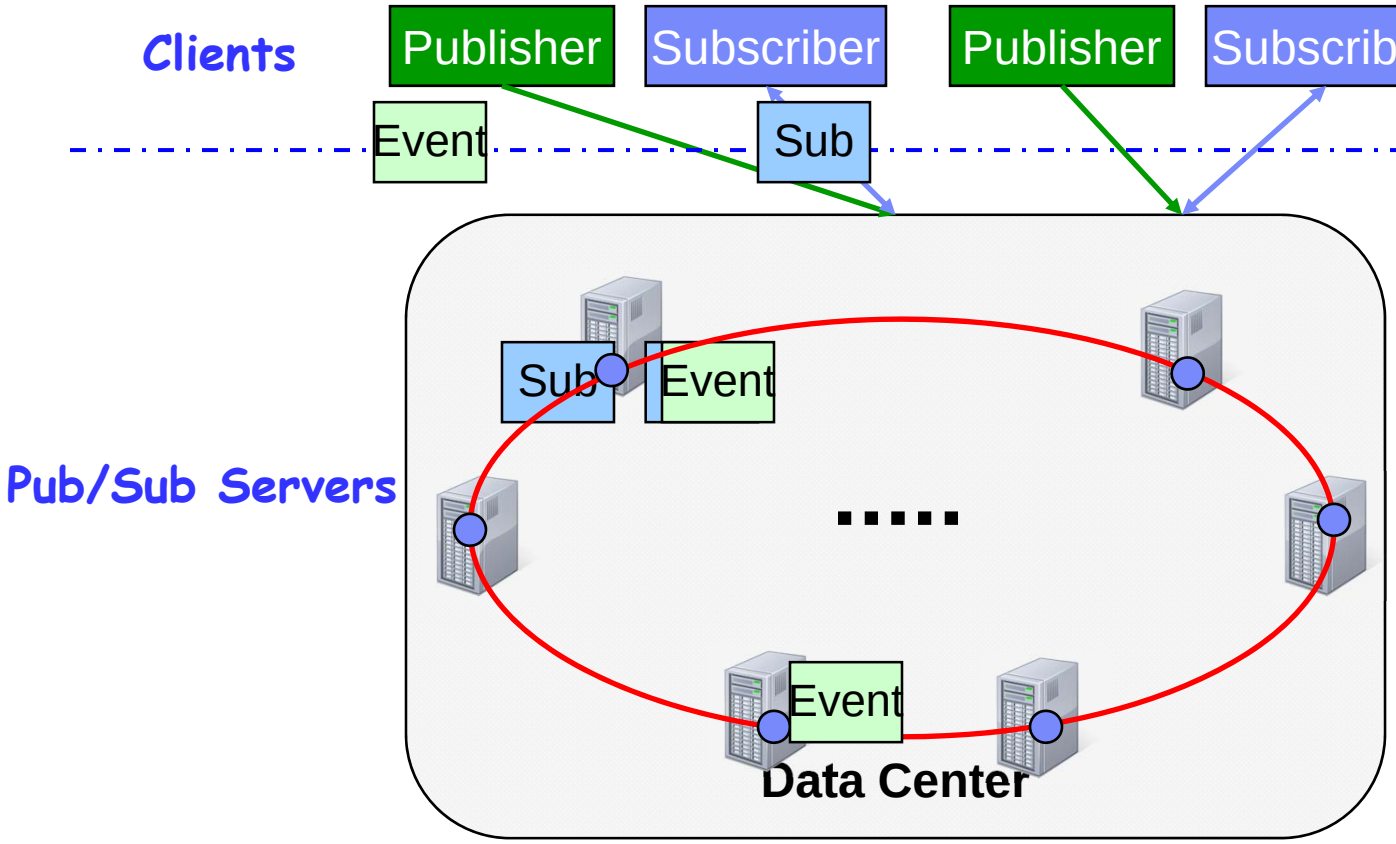
BlueDove Pub/Sub Cloud



- **Peer-to-peer public pub/sub**



BlueDove Pub/Sub Cloud Architecture

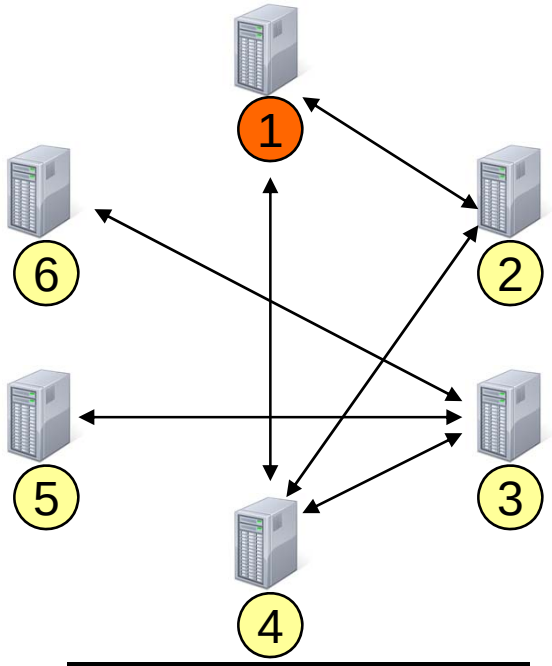


- Performance-oriented Event Forwarding
- Multi-dimension Subscription Assignment
- Gossip-based Single-hop Overlay

Gossip-based Single-hop Overlay

- **Each server maintains global view of the overlay**
  - The contact information of each other server
  - Enables one-hop forwarding
- **Overlay changes are propagated by gossip**
  - Each server periodically exchanges info with a few random servers
  - Any state change is propagated to the whole network in log(N) rounds
- **Advantages**
  - Auto, low-overhead maintenance
  - Low network delay

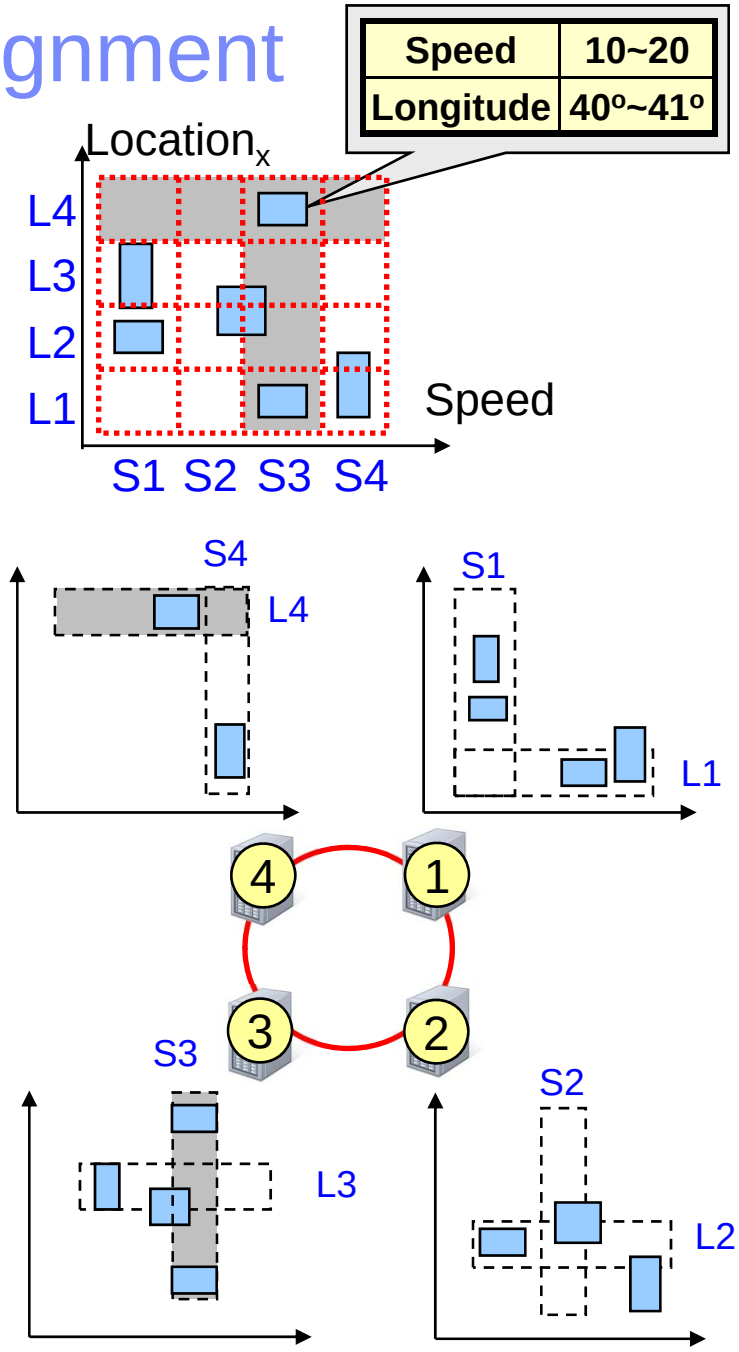
Overlay Table	
ID	IP
2	192.168.1.2
3	192.168.1.3
4	192.168.1.4



Overlay Table	
ID	IP
1	192.168.1.1
2	192.168.1.2
3	192.168.1.3

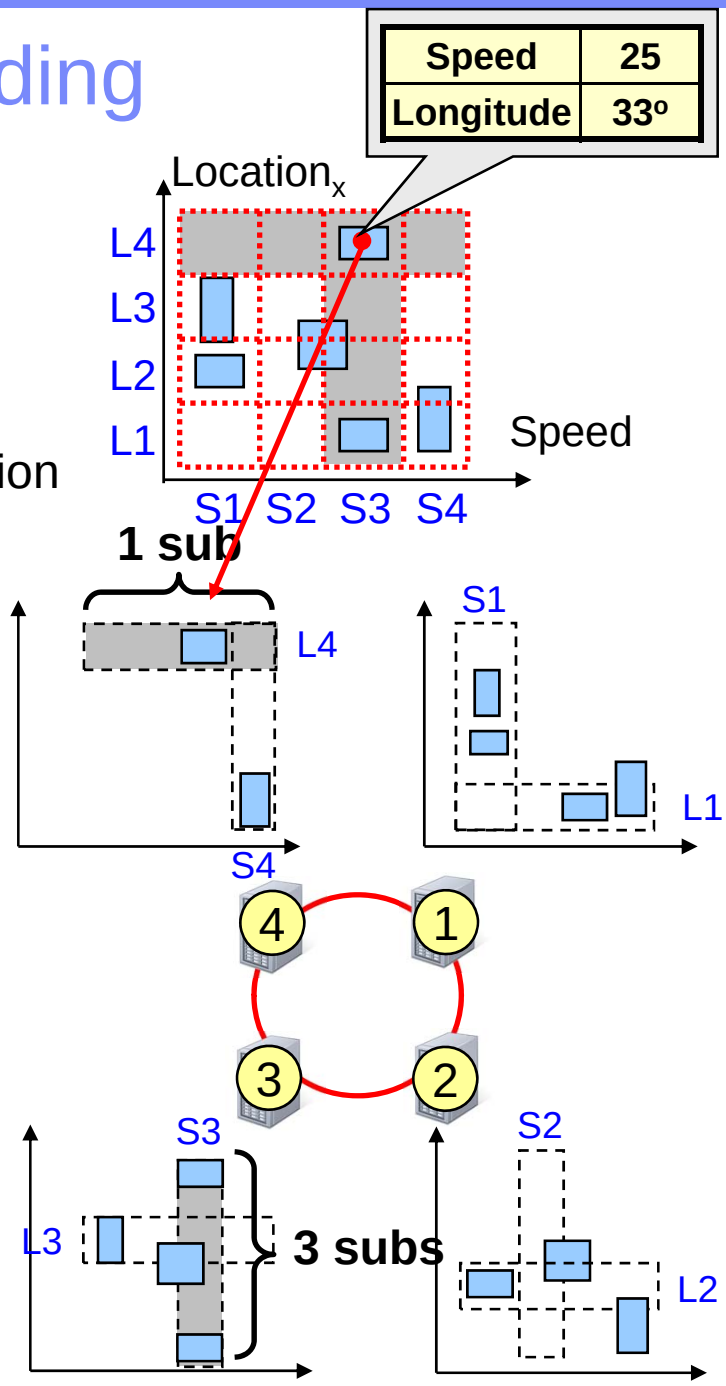
Multi-dimensional Subscription Assignment

- **Multiple dimensions are mapped to the overlay simultaneously**
  - Each dimension is partitioned into sections
  - Each server maintains a section of each dimension
  - Partition information is propagated by gossip and known by all servers
- **Each subscription is mapped in all K dimensions and stored by K servers simultaneously**
  - Each server maintains a separate index for subscriptions assigned on each dimension
- **Advantages**
  - Redundancy of servers
  - Allows us to exploit the skewness in distribution of subscriptions

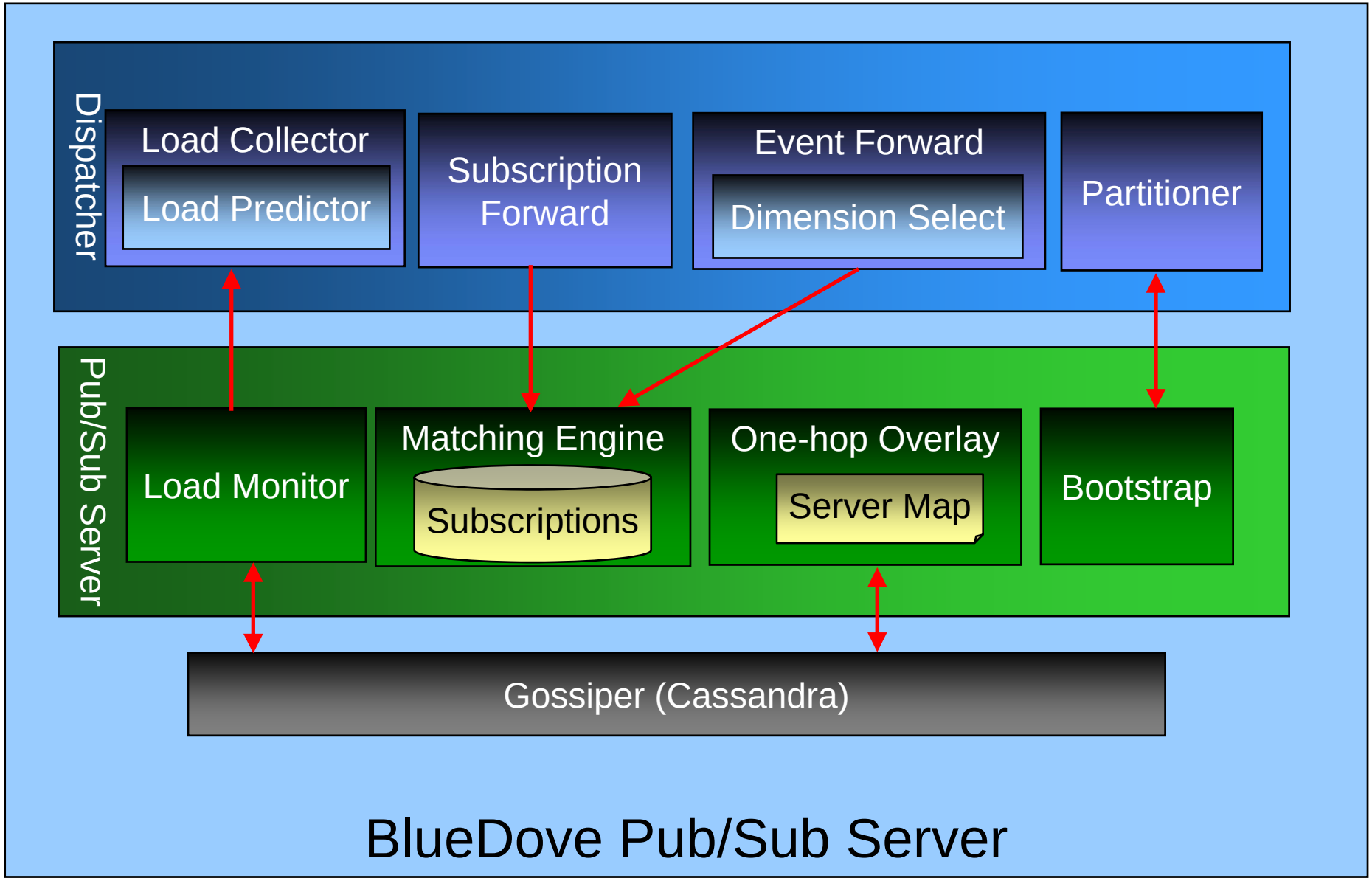


Performance Oriented Event Forwarding

- **Select the most efficient dimension to match events**
  - Servers exchange per dimension load information through gossip
    - Different load metris: #subscriptions, average response time, cpu load, …
  - Each event has K candidate servers, one for each dimension
  - Select the server with the best performance metric
- **Policy 1: minimum number of subscriptions**
  - Does not work well in heterogeneous environment
- **Policy 2: minimum average response time**
  - May not adapt fast enough when server load changes quickly
- **Policy 3: minimum predicted response time**
  - *Match_Time * ($QLen_{old}$ + T * (Event_Rate - Match_Rate))*
  - Adapts to heterogeneous environment and fast changing server load
- **Advantage**
  - Exploits the skewness of subscriptions
  - Load balancing



Implementation

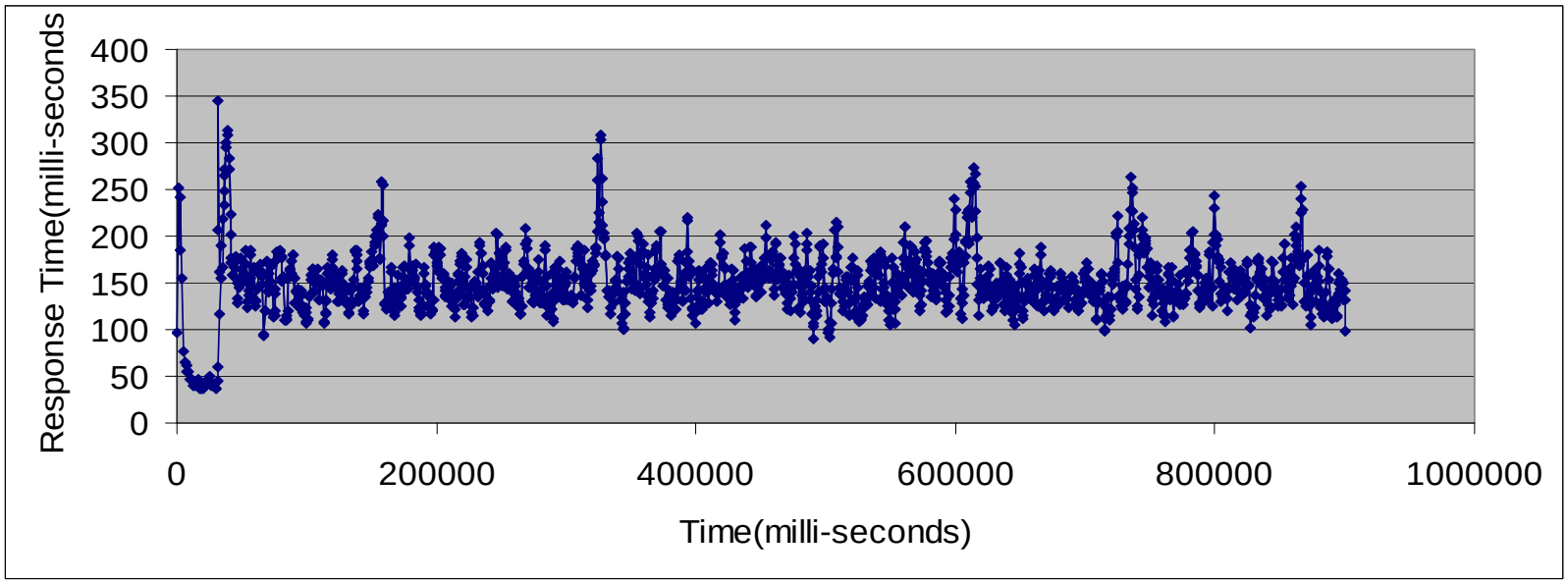


Testbed and Experiment Setup

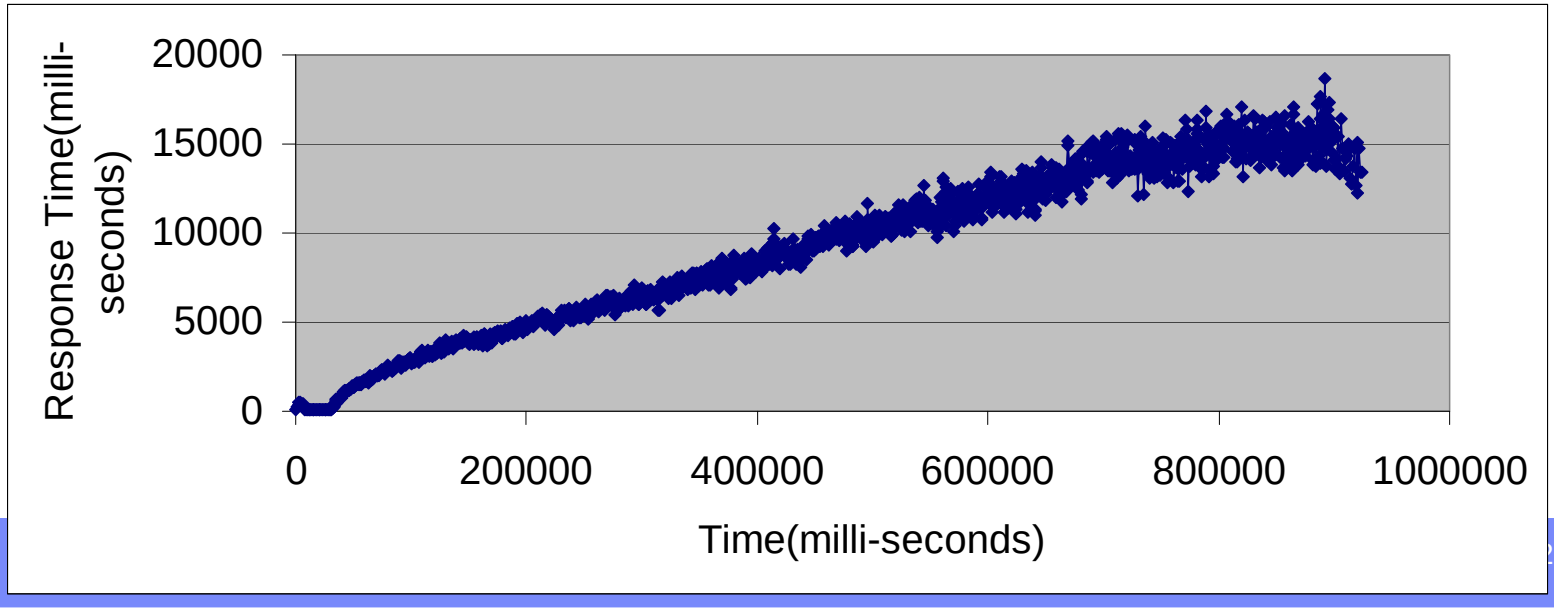
- **Testbed**
  - 25 RC2 machines
    - 20 pub/sub servers; 3 dispatchers; 2 to simulate clients
- **Simulators simulate publishers and subscribers**
  - Each subscription or publication has four attributes
  - Each attribute in subscriptions represents a range of 200
    - The center of the range follow a Normal distribution cropped of range size 1000, standard deviation of 250, mean at 125, 375, 625, 875 for dimension 1-4
  - Each attribute in events represents a point in [0,1000]
    - The point follows a uniform random distribution
- **Metrics**
  - Response time: the time period from when a publication is generated to when it is received by corresponding subscribers
  - Saturation event rate: the highest event rate that the system can handle without publication queue overflow

Response time behavior to event rate

- **Response time series before saturation: 40K subscriptions, 20 servers, 100K event/sec**

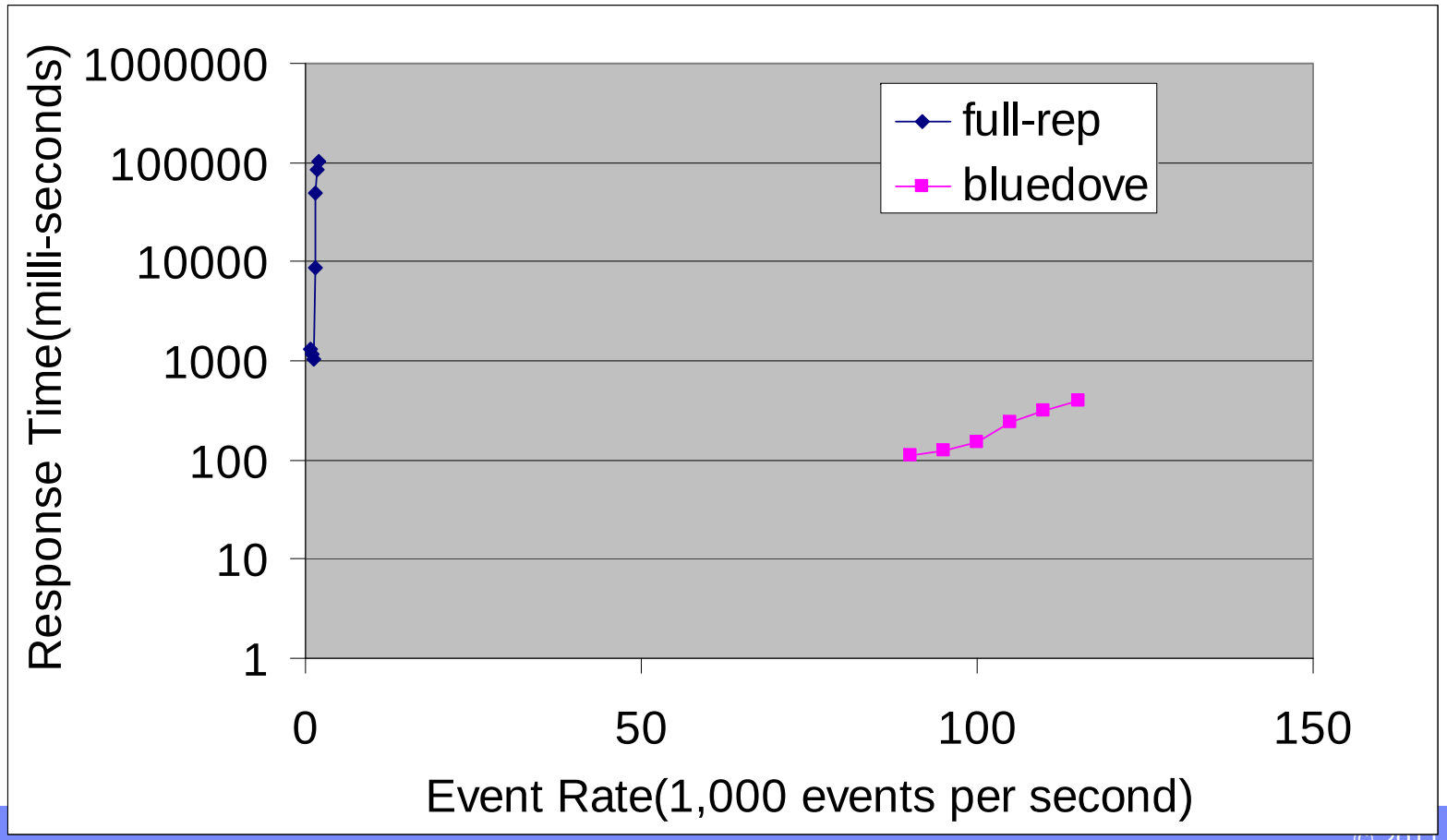


- **Response time series after saturation: 40K subscriptions, 20 servers, 150K event/sec**



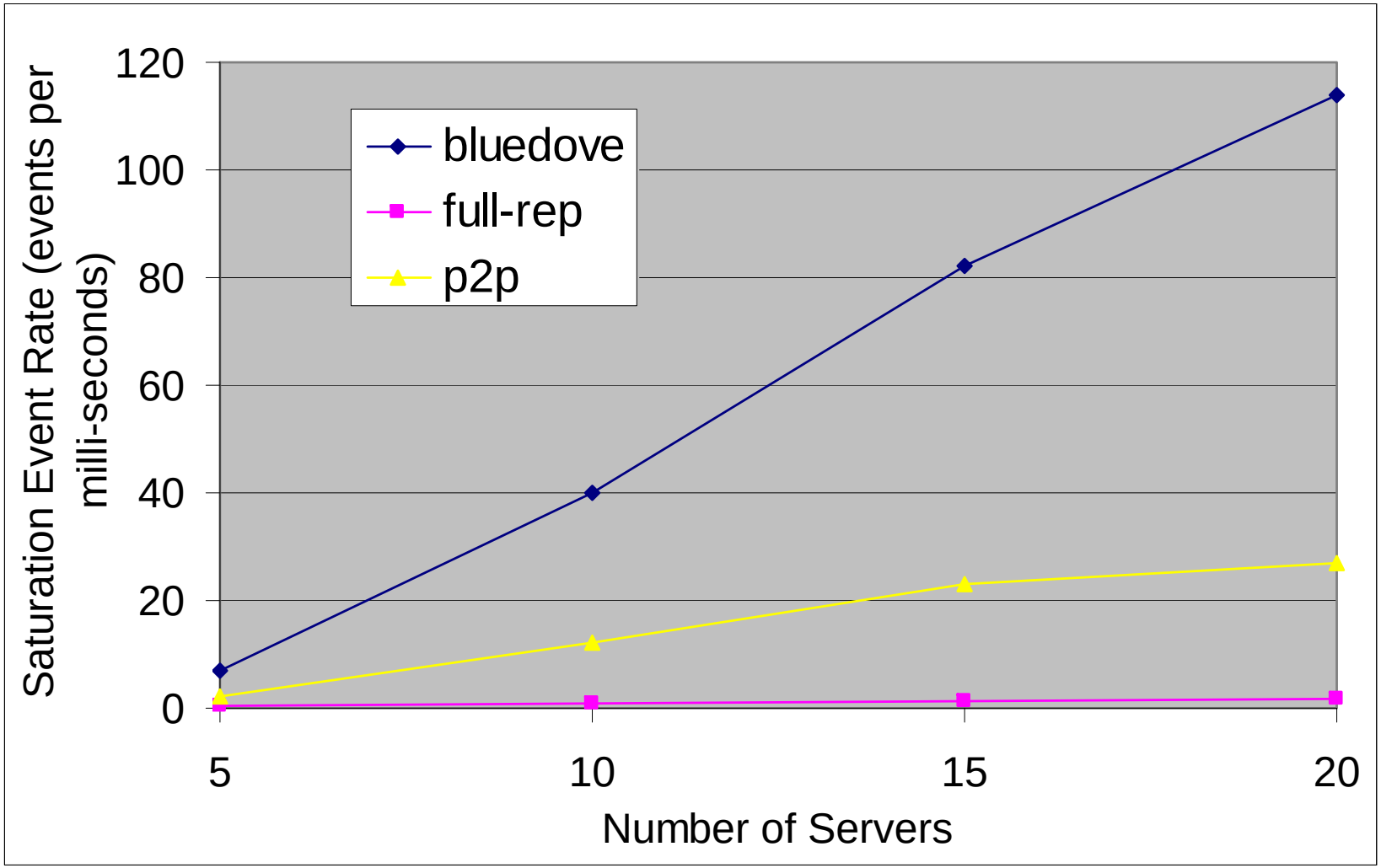
Impact of event rate to average response time

- **Compare BlueDove to Full-replication scheme**
  - 20 servers
  - 40,000 subscriptions



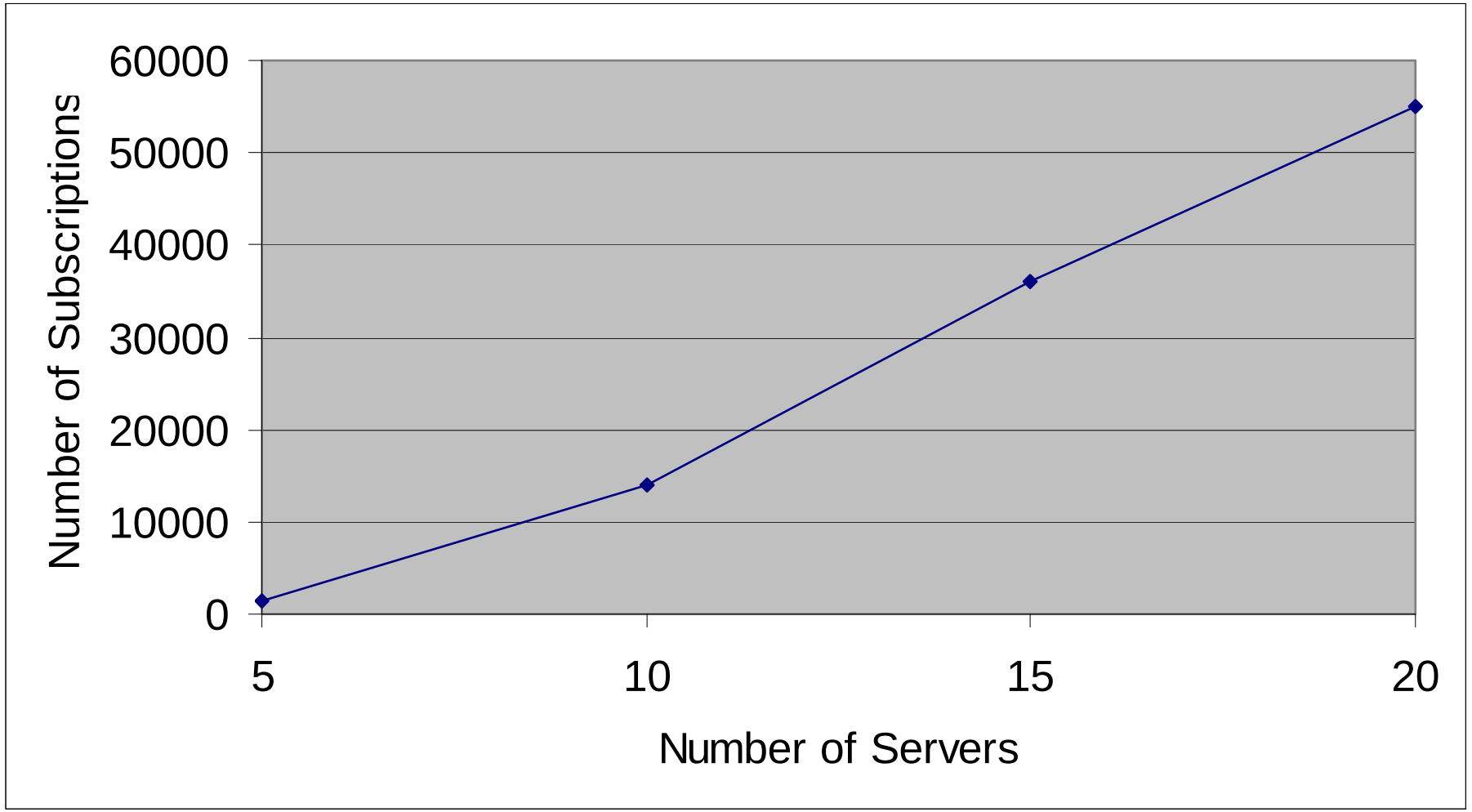
Scalability to event rate

- **Fix number of subscriptions (40K), increase number of servers**



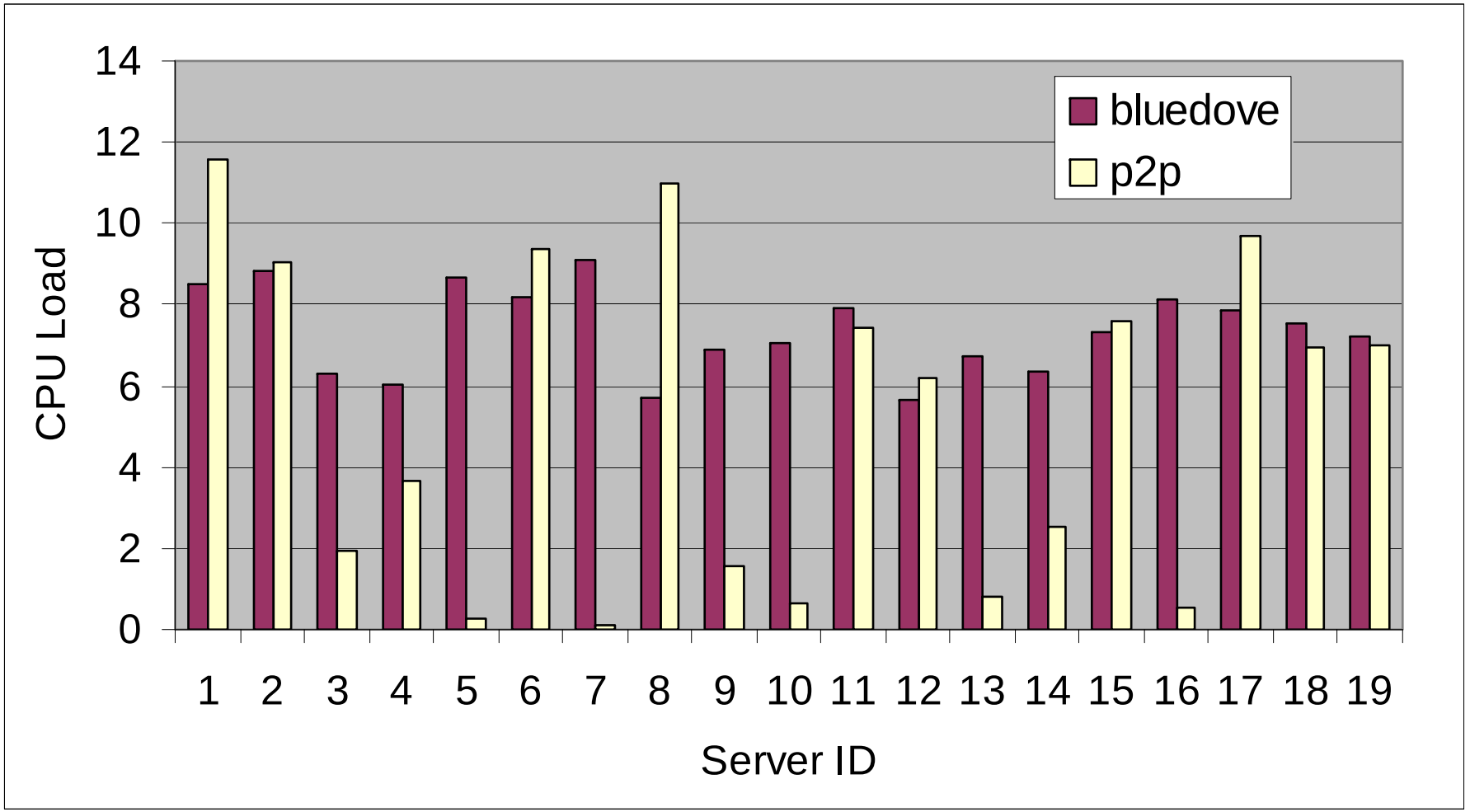
Scalability to number of subscriptions

- **Fix event rate; increase number of servers**



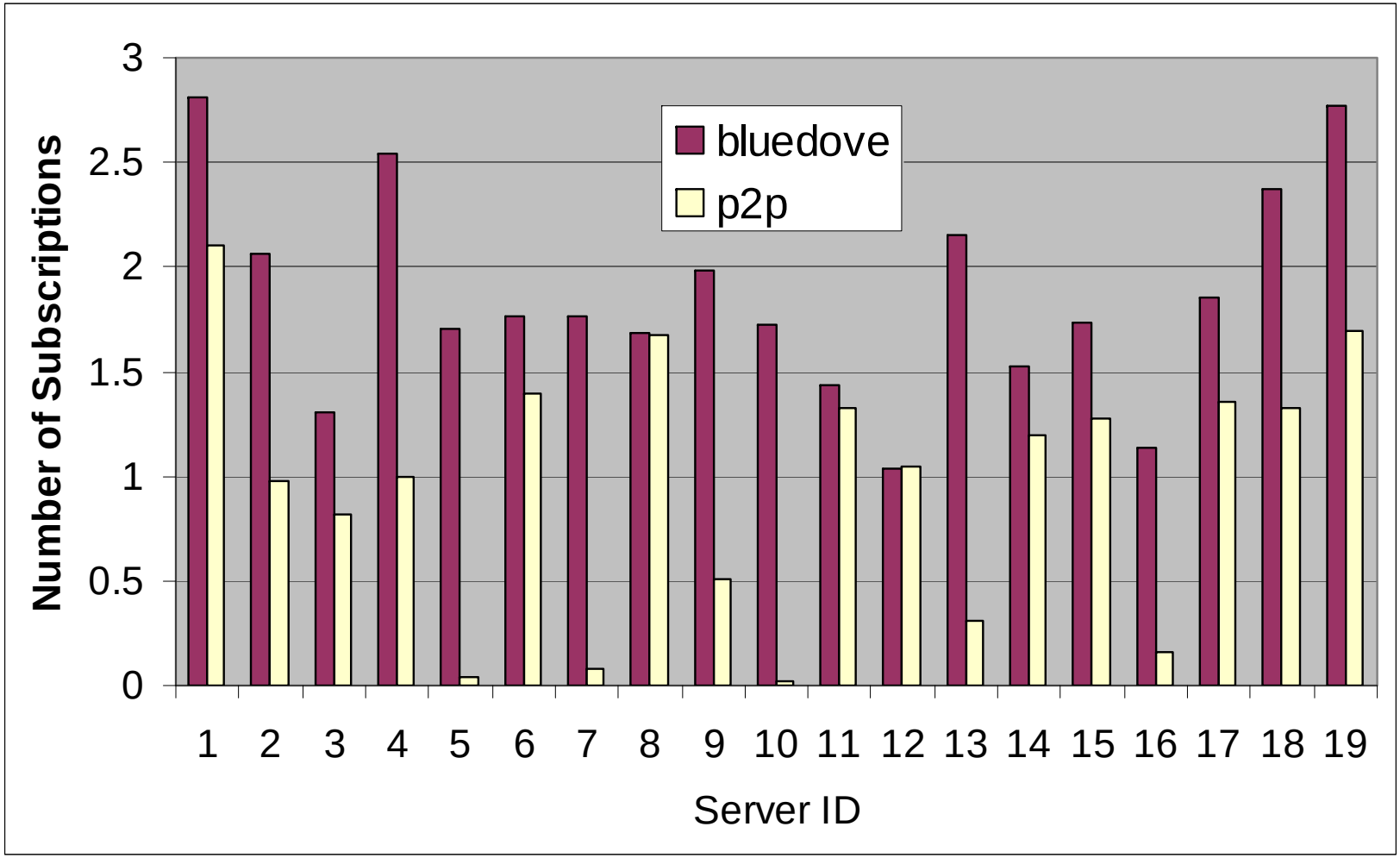
CPU load distribution

- **CPU load across different servers**
  - 20 servers; 40,000 subscriptions; 100,000 events per second for BlueDove, 27,000 events per second for p2p scheme



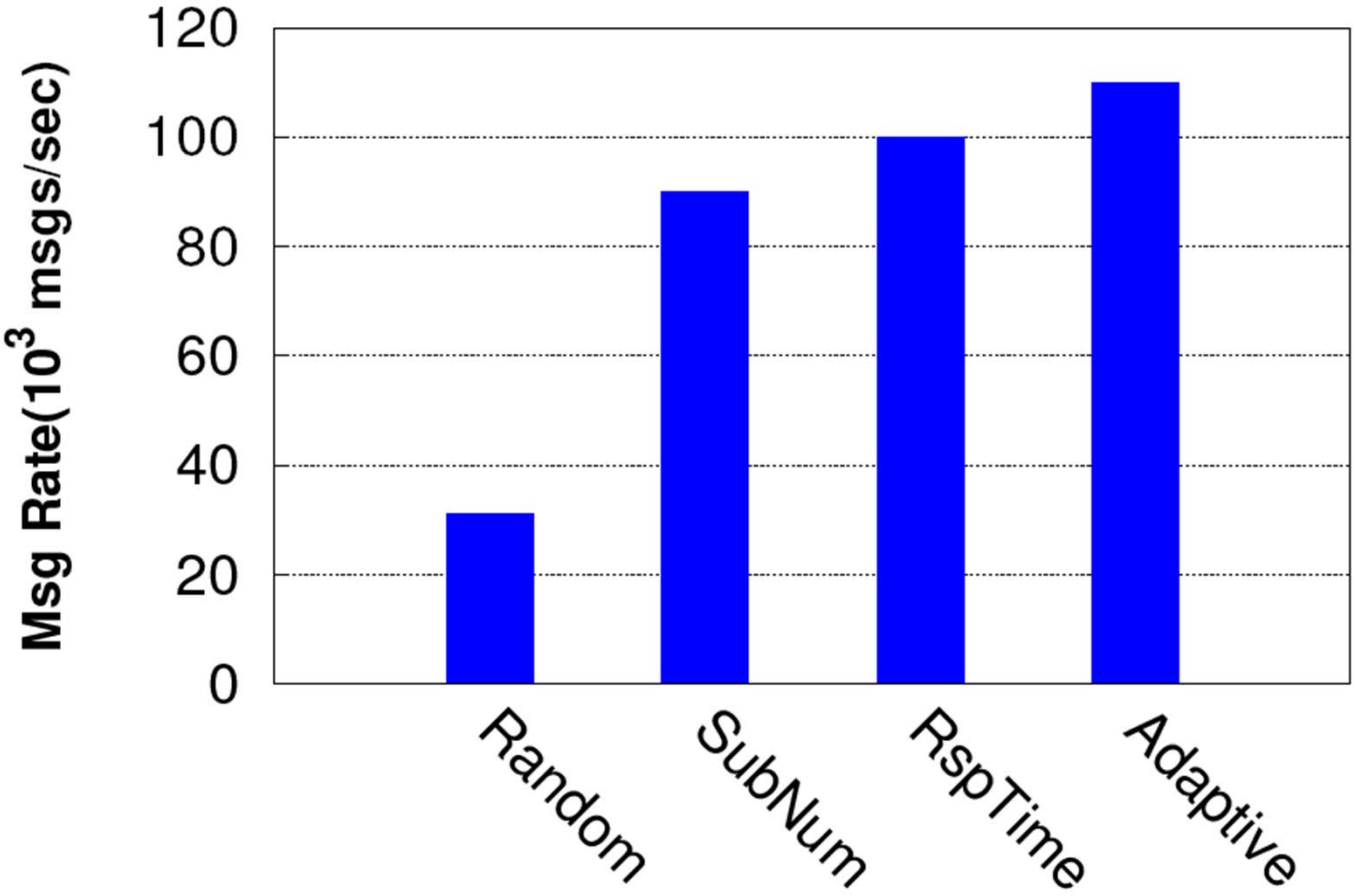
Matching load distribution

- **Number of events (10^9) matched on different servers**
  - 20 servers; 40,000 subscriptions; 100,000 events per second for BlueDove, 27,000 events per second for p2p scheme



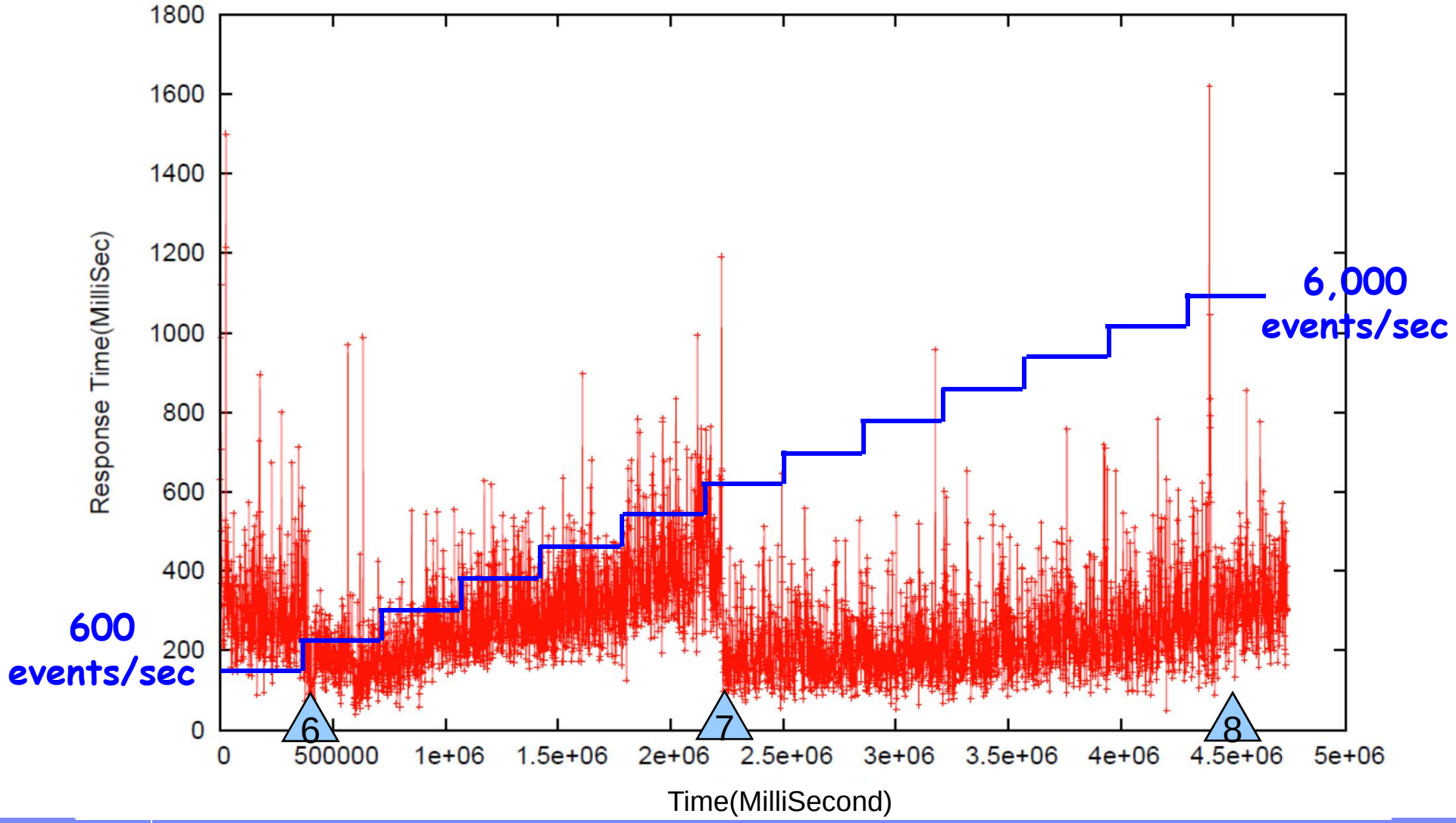
Different Event Dispatching Policies

- **Compare random, subscription-number-based, response-time-based and prediction-based policies**



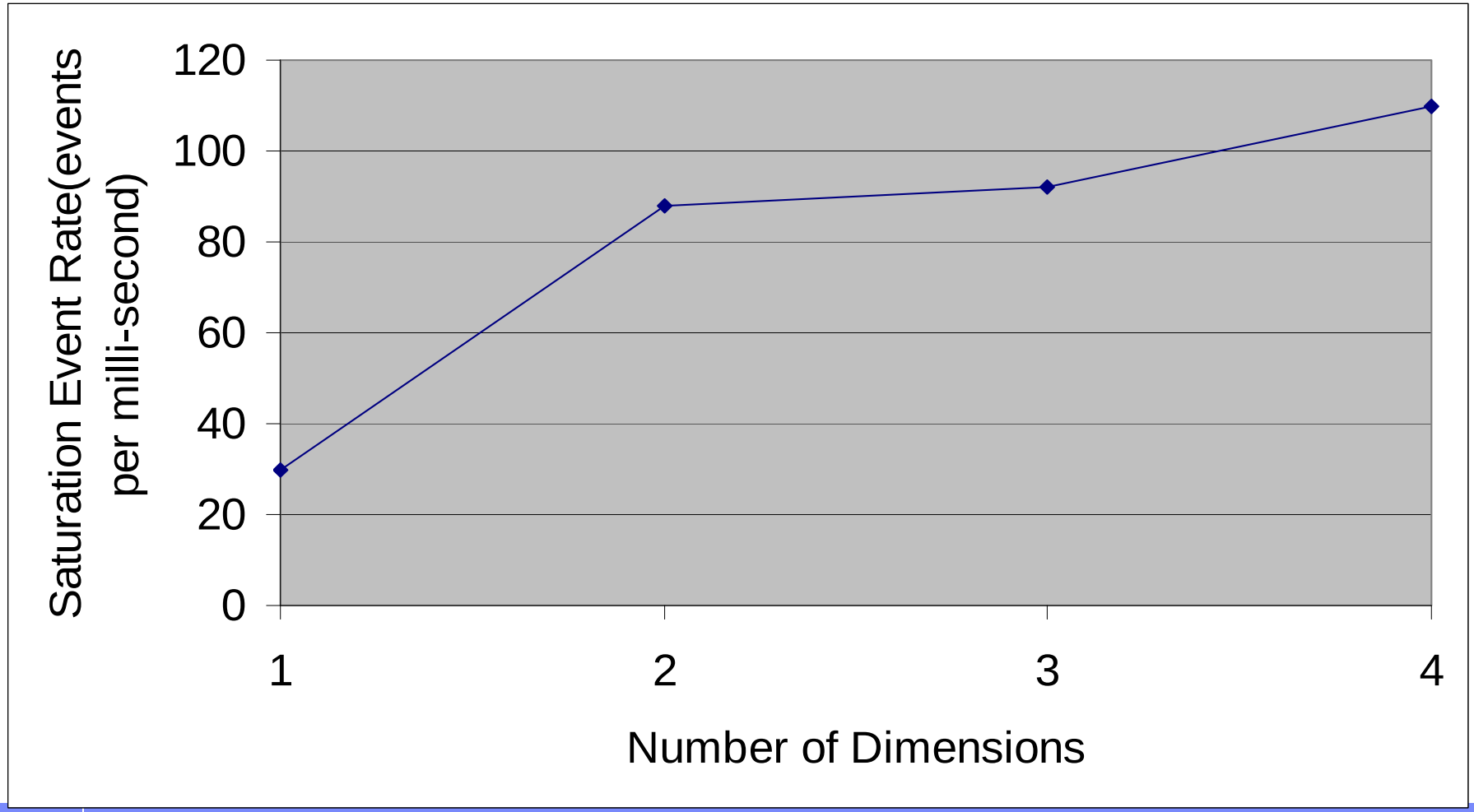
Elasticity upon increasing load

- **Event rate increases continuously. Initially there are 5 nodes**



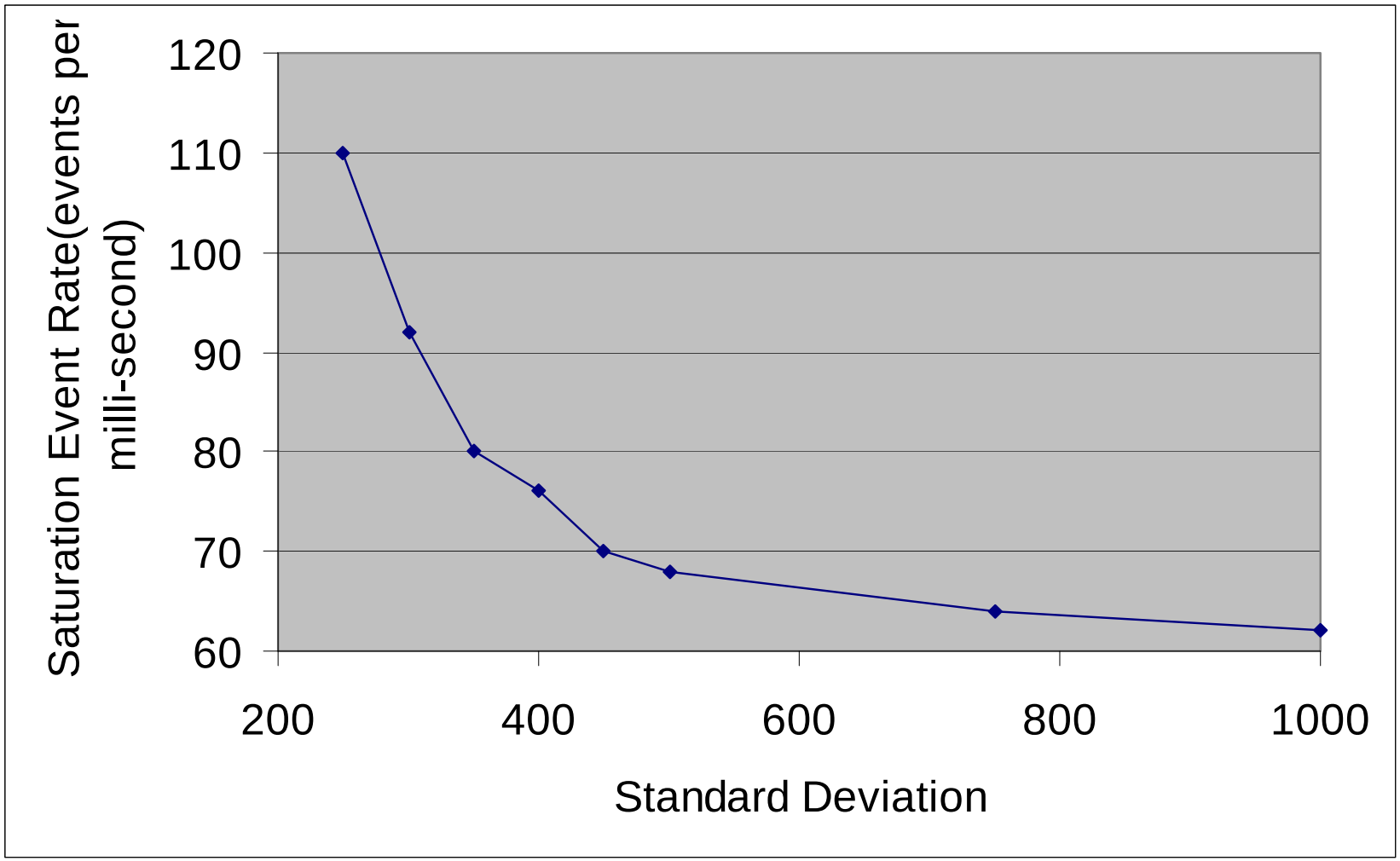
Impact of the number of partition dimensions

- **Fix the number of subscriptions and change the number of partition dimensions**
  - 20 servers; 40,000 subscriptions



Impact of the skewness of the distribution of subscriptions

- **Change the standard deviation of subscription distribution**
  - 20 servers; 40,000 subscriptions



Summary

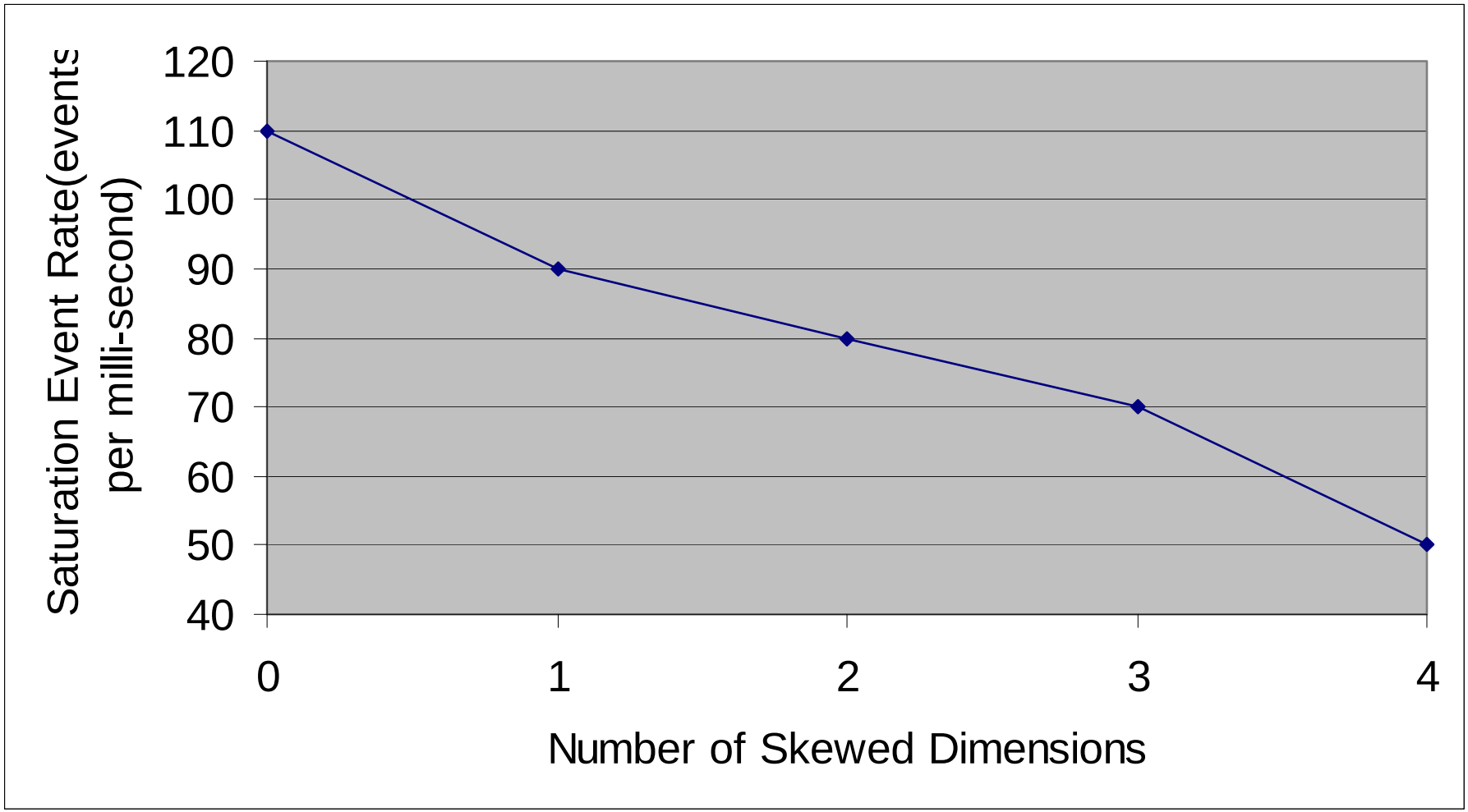
- **A cloud based publish/subscribe service provides an answer to the challenge of interconnecting numerous information producers and consumers in the Smarter Planet**
- **BlueDove exploits the skewness in data distribution and difference across multiple dimensions to achieve high performance**
- **Evaluation has shown its linear capacity scaling and agile adaptation to changes in workload**

Thank you!

Questions?

Impact of the skewness of the distribution of events

- **Change the number of event dimensions that are skewed in the same way as their corresponding subscription dimensions**
  - 20 servers; 40,000 subscriptions



Future Work

- **Explicit load balancing**
  - Move subscriptions from overloaded nodes to less-loaded ones
- **Integration with BlueDove queue**
  - BlueDove queue takes the responsibility of pushing publications to subscribers
- **Persistence**
  - “outsource” data storage to a cloud-based storage (e.g. Cassandra)

Cross-data-center replication of subscriptions

- **Implicit replication**
  - Each subscription is already replicated to multiple servers
  - With high probability at least two servers are in different data centers
- **Explicit replication**
  - If all replicas are in the same data center, then explicitly make a copy in another data center