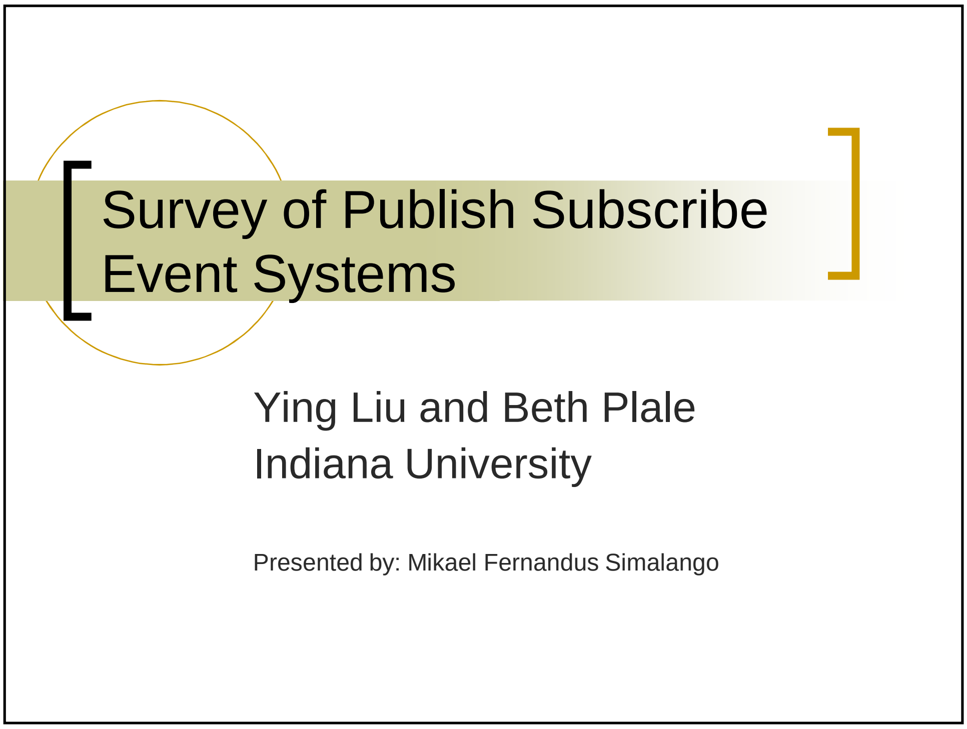
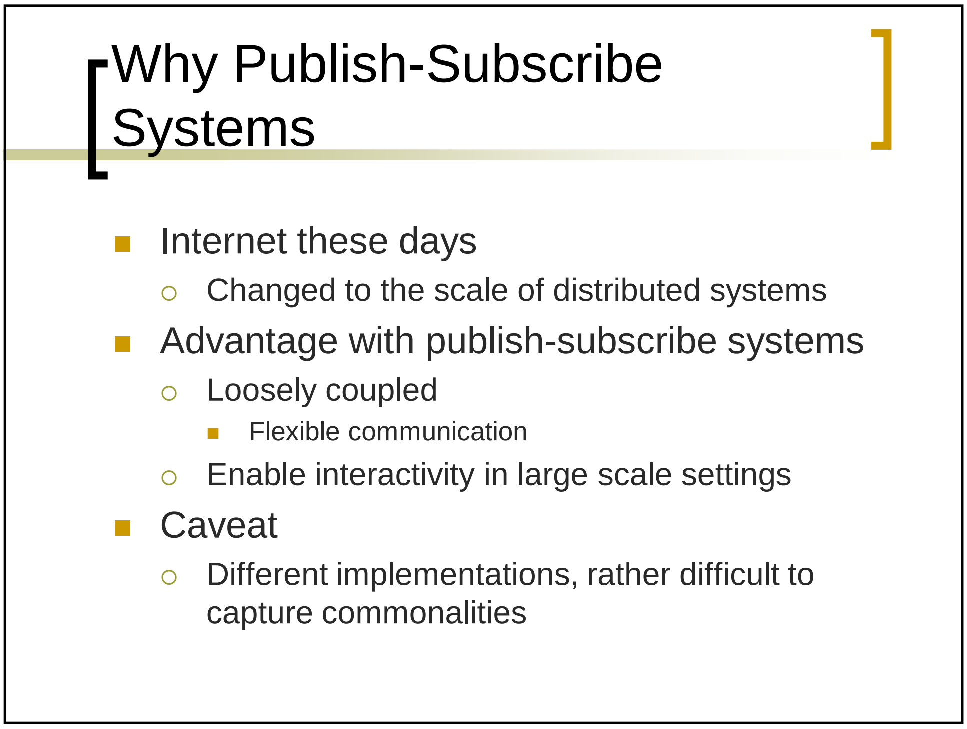


Survey of Publish Subscribe Event Systems

Ying Liu and Beth Plale
Indiana University

Presented by: Mikael Fernandus Simalango



Why Publish-Subscribe Systems

- Internet these days
  - Changed to the scale of distributed systems
- Advantage with publish-subscribe systems
  - Loosely coupled
    - Flexible communication
  - Enable interactivity in large scale settings
- Caveat
  - Different implementations, rather difficult to capture commonalities

Interaction in Publish-Subscribe Systems

- Push-based
  - Messages are automatically broadcast to subscribers
  - Provides tight consistency
  - Stores minimal data
- Pull-based
  - Can be more responsive to user needs [?]
    - No further explanation, I tend to disagree
- Combination of both to achieve better scalability

Taxonomy of Publish-Subscribe Systems

- Subject-based vs Content-based
- System architectures
- Matching algorithms
- Multicast algorithms
- Reliability and Security

Subject-based vs Content-based

- Subject-based:
  - A message belongs to groups, channels, or topics
  - Users are subscribed to groups / channels / topics and receive messages associated with the groups
- Content-based:
  - Message delivery based on a query or predicate issued by subscriber
  - No need to learn / get info of all available groups
  - But it burdens underlying system to match message for subscription

System Architecture

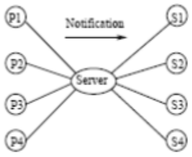
- Publish-subscribe system can be categorized into two general models:
  - Client-server model
    - A component serves as an event server or an event client
    - Event server receives events, store if necessary, and forward
    - Event server communicates with other event server to provide better scalability
    - Event client acts as publisher, subscriber, or both
  - Peer-to-peer model
    - Nodes are equal. Each can act as a publisher, subscriber, root of a multicast tree, internal node of a multicast tree or any combination thereof

Client-Server Model

- General topologies
  - Star topology (centralized server)
  - Hierarchical topology
  - Ring topology
  - Irregular polygon topology

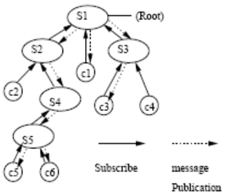
Client-Server Model (cont'd)

- Star topology
  - Relies on a single event server to broker between publishers and subscribers
  - Does not scale well



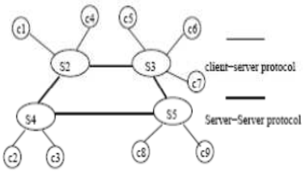
Client-Server Model (cont'd)

- Hierarchical topology
  - Hierarchical relationship between event servers
  - Clients can be either publishers or subscribers
  - Parent server only forwards message to its subtree
  - Provide better scalability



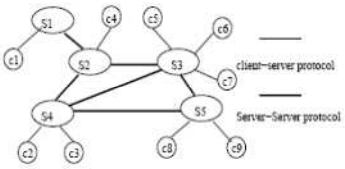
Client-Server Model (cont'd)

- Ring topology
  - Servers exist in peer-to-peer relationship with one another, forming a ring
  - Communication between servers is via bidirectional communication protocol for exchanging subscriptions and notifications



Client-Server Model (cont'd)

- Irregular polygon topology
  - Similar with ring topology
  - However servers are connecting each other in irregular polygon form instead of a ring



Matching Algorithm

- Subject-based system
  - Simplistic way by looking up topic ID for the message and then determine subscribers ID
  - Bayeux -> hashed-suffix mesh algorithm to locate subscribers and route messages across large network
  - Echo -> establishes direct connection between a publisher and subscriber when the subscriber subscribes to the channel

Matching Algorithm (cont’d)

- Content-based system
  - Matching tree algorithm
    - More general term at higher level
    - More refined/specific term at lower level
    - Subscription through tree traversal from up to bottom

Multicast Algorithms

- When the system scales, it needs more efficient event distribution
- Which is often in form of software-based multicast
  - Multicast: broadcasting message from one broker to subscribers that are associated with that broker
- The algorithm varies on different systems

Reliability and Security

- Reliability: delivery of the message should be reliable
- Security: secure delivery of message
  - Key aspects:
    - Authentication -> establishment of identity of originator of an action
    - Confidentiality -> ability to keep others away from accessing messages
    - Integrity -> requirement of keeping the message in its original form
    - Accountability -> ensuring only proper credential is responsible for sending the message

Surveyed Pub-Sub Systems

- Gryphon
  - Content-based using matching tree algorithm
  - Client-server model
  - Link matching algorithm for multicast
  - Broker organization protocol for fault tolerant delivery
  - Targeted toward the distribution of large volumes of data in real-time to thousands of clients in a large public network

Surveyed Pub-Sub Systems (cont'd)

- Scribe
  - Subject-based with numeric keys and node IDs for matching algorithm
  - Peer-to-peer model
  - Group-based algorithm form multicast
  - Best-effort delivery

Surveyed Pub-Sub Systems (cont'd)

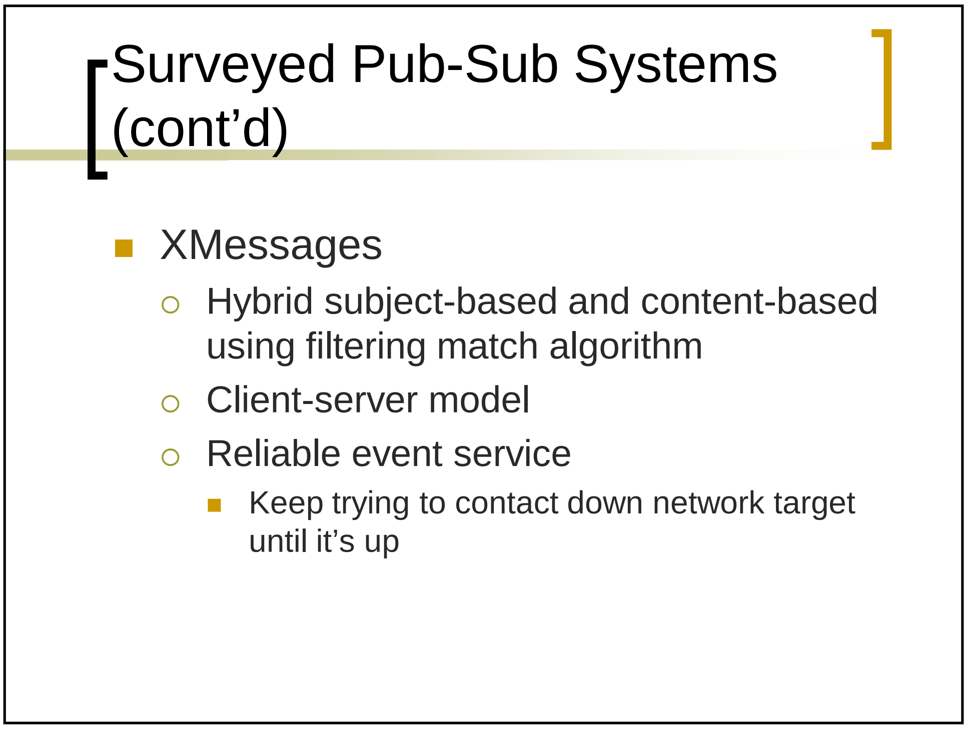
- Bayeux
  - Based on Tapestry
  - Subject-based with hashed-suffix mesh algorithm
  - Peer-to-peer model
  - Tolerating failures in routers and network links

Surveyed Pub-Sub Systems (cont’d)

- Siena
  - Content-based with Binary Decision Diagram algorithm
  - Client-server model
  - Hierarchical server topology

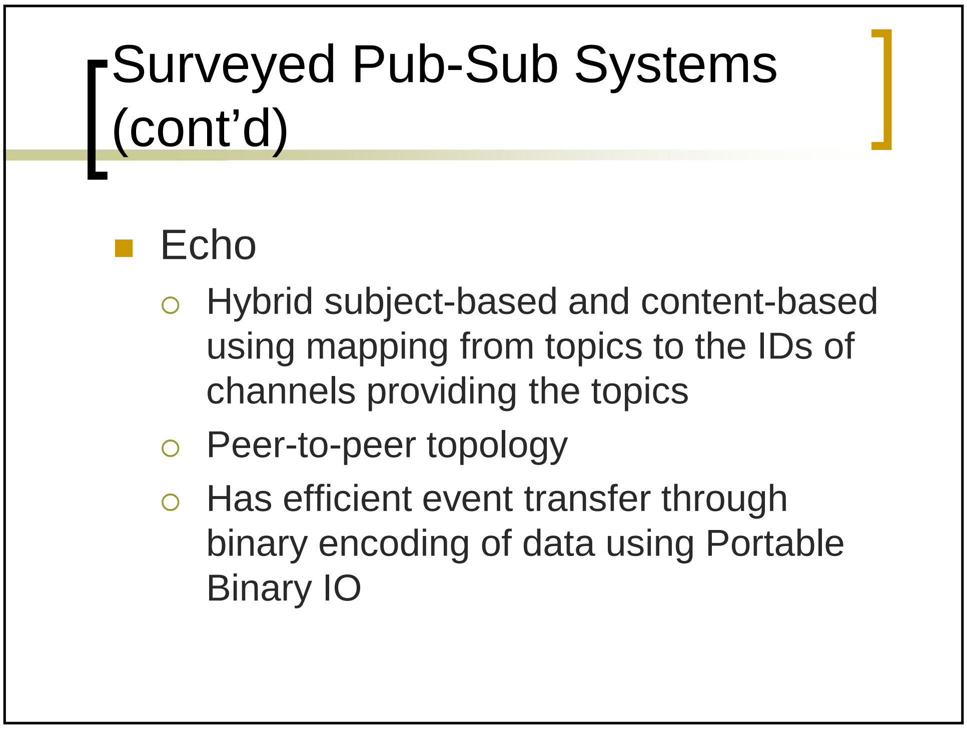
Surveyed Pub-Sub Systems (cont’d)

- NaradaBrokering
  - Content-based using matching tree from the content of subscriptions or SQL 92 based on JMS or XML attribute-value pairs for topic subscriptions
  - Client-server model (hierarchical topology)
  - Routing through shortest path computations



Surveyed Pub-Sub Systems (cont'd)

- XMessages
  - Hybrid subject-based and content-based using filtering match algorithm
  - Client-server model
  - Reliable event service
    - Keep trying to contact down network target until it's up



Surveyed Pub-Sub Systems (cont'd)

- Echo
  - Hybrid subject-based and content-based using mapping from topics to the IDs of channels providing the topics
  - Peer-to-peer topology
  - Has efficient event transfer through binary encoding of data using Portable Binary IO

Surveyed Pub-Sub Systems (cont'd)

- JMS
  - Java API that allows applications to create send, receive, and understand messages
  - Defines interfaces that can be used by different Message-Oriented Middleware (MOM) vendors
  - Messaging models: publish-subscribe and point-to-point queuing
  - Can support client-server and peer-to-peer model
  - Can support durable subscriptions
    - Temporarily unsubscribe but later receive messages when it resubscribes

Conclusion

- This survey categorizes publish-subscribe system by observing their taxonomy: subject based or content-based, system architecture, matching algorithm, multicasting algorithm, reliability, and security