ECMS06
2006, Bonn, Germany

THROUGH THE EVOLUTION OF PUBLISH/SUBSCRIBE SYSTEMS – PART I

Emiliano Casalicchio (casalicchio@ing.uniroma2.it)

Federico Morabito (federico.morabito@telecomitalia.it)

Outline

- Part I: Base concept on publish/subscribe system architecture
- Part II: An application case
- Part III: Modeling and Performance evaluation of publish/subscribe systems

Outline: Part I

- The pub/sub interaction paradigm
  - How it works
  - Applications
- Basic concepts
  - Architecture
  - Subscription scheme
  - Events routing
  - Events matching

Outline: Part II

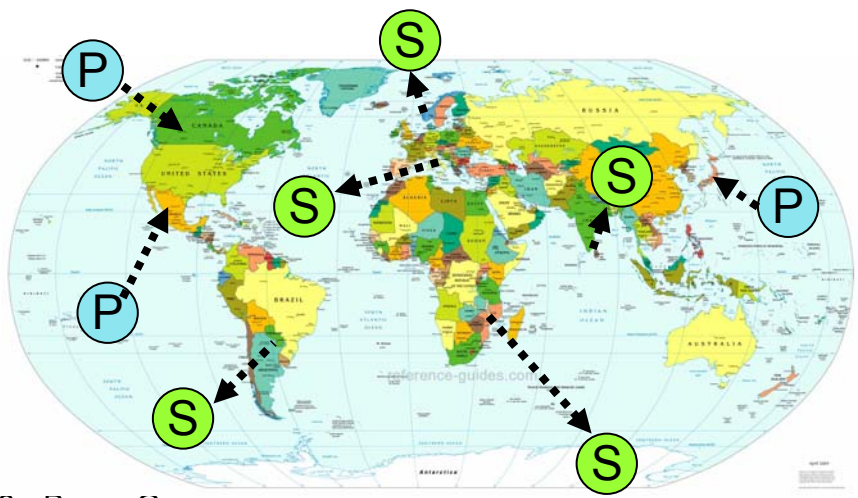
- An application case
  - The global monitoring infrastructure for critical systems

Outline: Part III

- Modeling and Performance evaluation of pub/sub systems
  - General consideration
  - Challenges
- Modeling techniques and analysis methods:
  - Overlay infrastructure
  - Event routing
  - Event Matching
- Workload characterization
  - Overlay infrastructure
  - Event routing
  - Event Matching

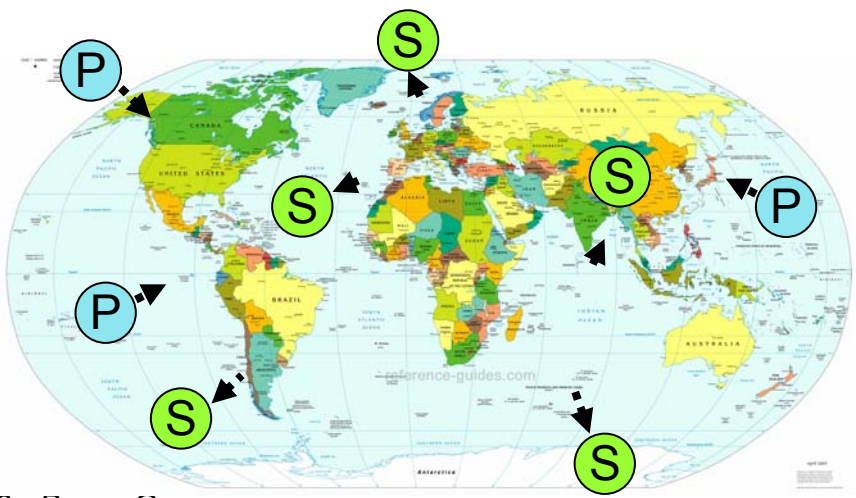
What's a Publish/Subscribe (Pub/Sub) system?

- A system based on the pub/sub interaction paradigm
  - Asynchronous communication paradigm
- Publishers generate events
  - that is, they publish data on the basis of some rules or triggers
- Subscribers register their preferences on some events, or pattern of events



Mains Pub/Sub interaction paradigm characteristics

- Message based, asynchronous and anonymous communication
- Participants are decoupled
  - in space: no need to be connected or even know each other,
  - in flow: no need to be synchronized,
  - in time: no need to be up at the same time

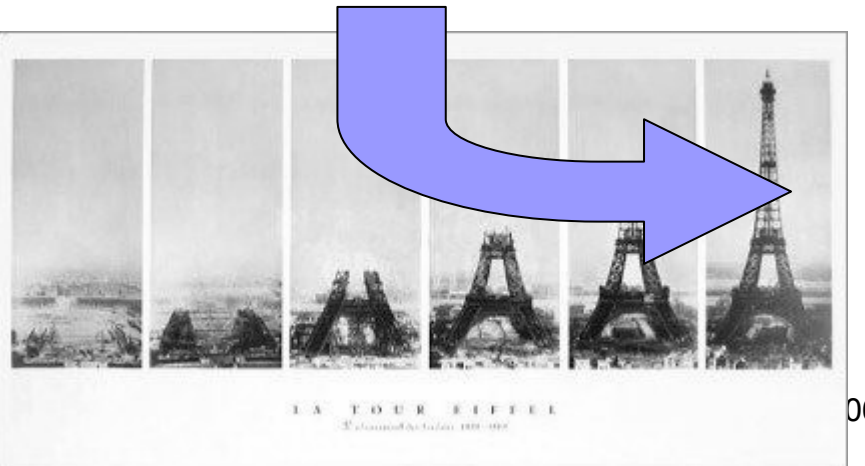


Examples of Pub/Sub systems

- Pub/sub interaction paradigm is a good solution for building distributed applications, over large scale, highly dynamic networks
  - News delivery (dissemination of multimedia content)
  - Stock quoting, Auction bids dissemination
  - On line games
  - Management of large scale systems (services and resources)
  - Monitoring and Control of critical systems

Challenges

- Pub/sub as Message Oriented Middleware (MOM) for asynchronous communication [LVV06]in
  - web-based and distributed systems [FCMB06]
  - High performance computing (cluster-grid environment) [ESB06]



- Automatic reconfiguration of the topology of the distributed dispatching infrastructure [PCM03]
- Reconfigurable pub/sub notification service [FCFB05]
- Define a global M&S framework to discover dependencies and correlations
- Support for management and customization, heterogeneity, security [FZBK04], reliability [BBCF04]
- Support for QoS provisioning (producer and consumer are decupled and unaware)
- Support for mobility [FGKZ03]

The Pub/Sub system actors

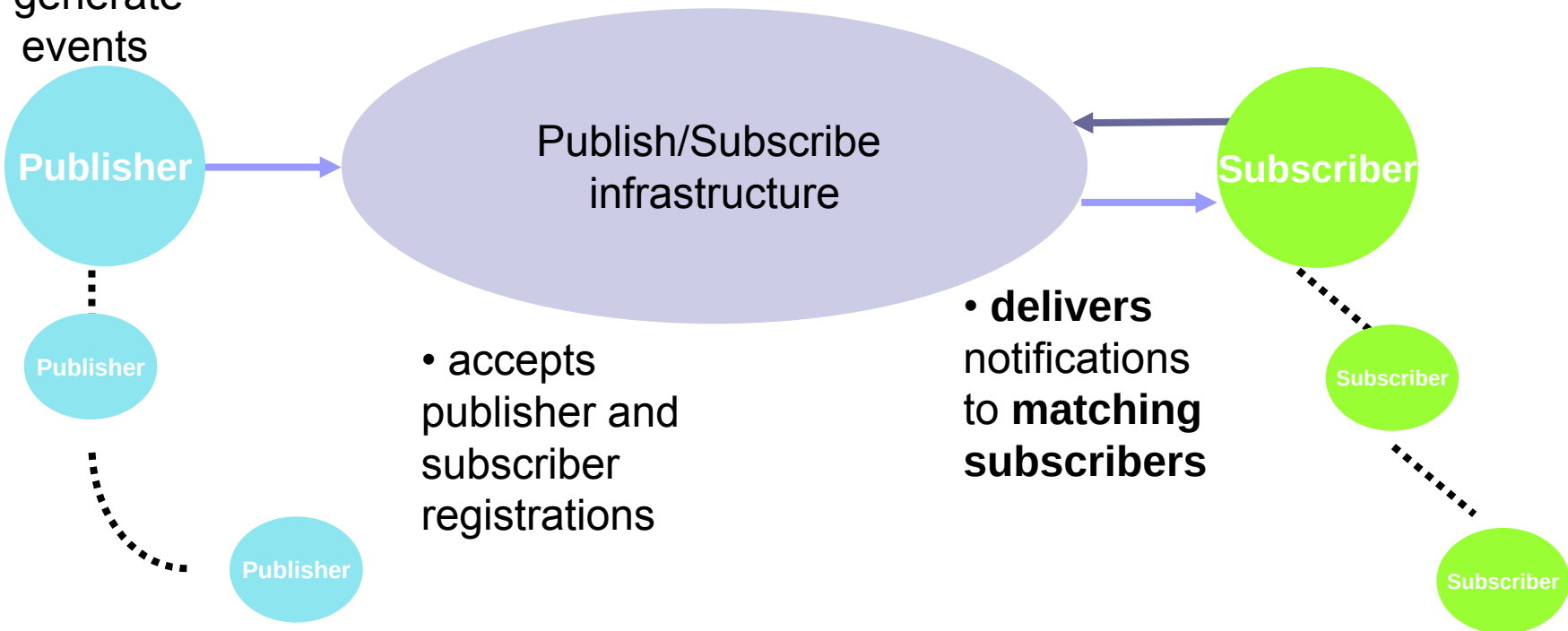
- Subscribers have the ability
  - to express their interest in an event, or a pattern of events,
  - they are subsequently notified of any event, generated by a publisher, which matches their registered interest.
- Publisher generates events
  - An event is asynchronously propagated to all subscribers that registered interest in that given event.
- Pub/sub infrastructure allow the communication process

The Pub/Sub system actors

- determines which subscription **match** with the **event**

- register subscriptions
- express their interest in events or pattern of events

- generate events



- accepts publisher and subscriber registrations

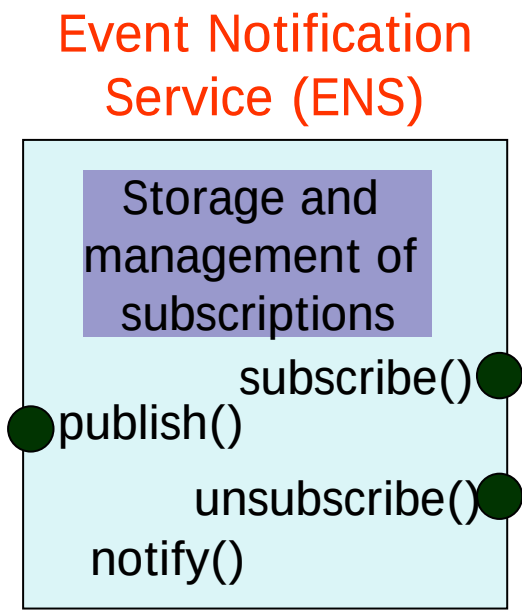
- **delivers** notifications to **matching subscribers**

Terminology

- An event is represented by a data structure called event notification, or simply notification
- Publisher are nodes that generate event notifications
- Subscribers are nodes that consume event notifications
  - They express their interest in an event, or a pattern of events, by subscribe themselves to the class of events they are interested.
- The Event Notification Service (neutral mediator between pubs and subs)
  - accepts subscriptions from the subscribers.
  - dispatches the notification to all the subscribers that expressed their interests for an occurring event or combination of events.

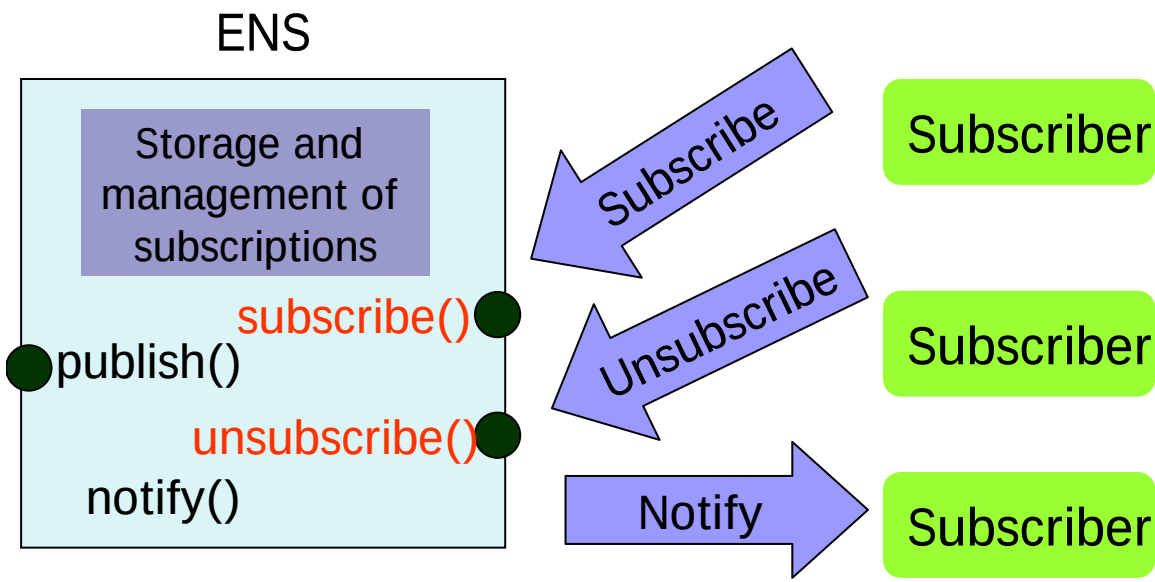
The basic interaction scheme

- Event notification service provides
  - storage and management for subscriptions
  - efficient delivery of events
  - communication primitives



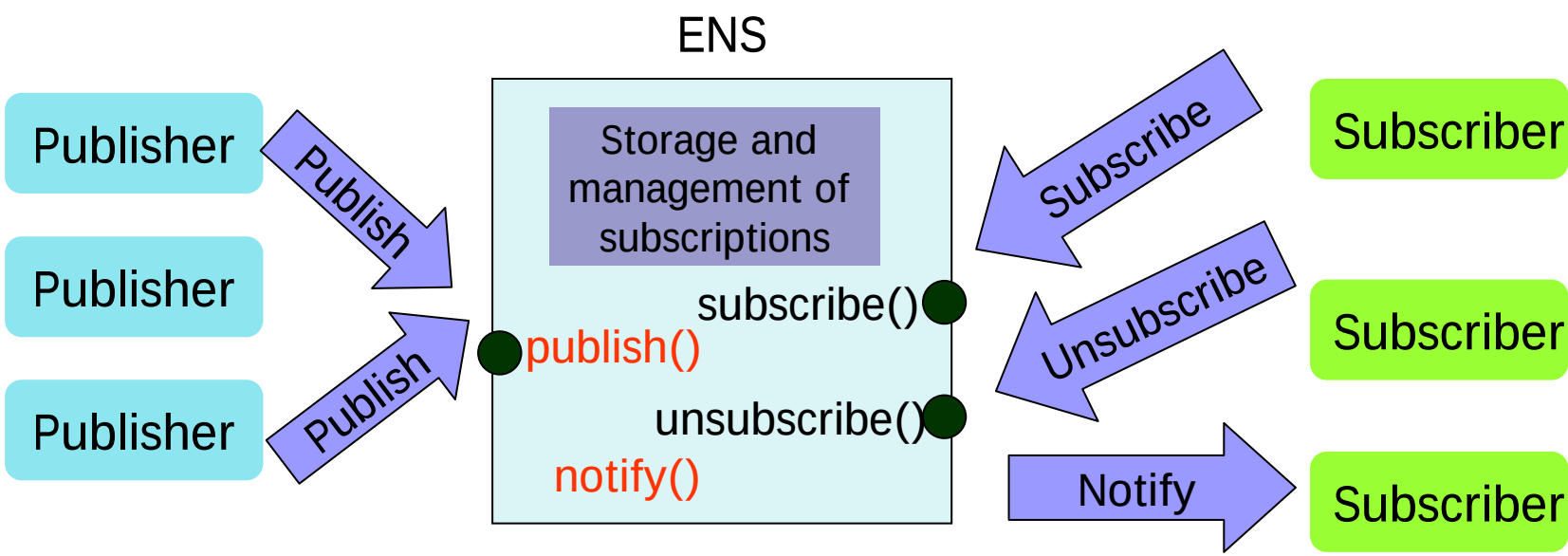
The basic interaction scheme

- **subscribe()**: invoked by subscribers to register their interest in events
  - This subscription information remains stored in the event service and is not forwarded to publishers.
- **unsubscribe():** invoked by subscribers to terminate a subscription.



The basic interaction scheme

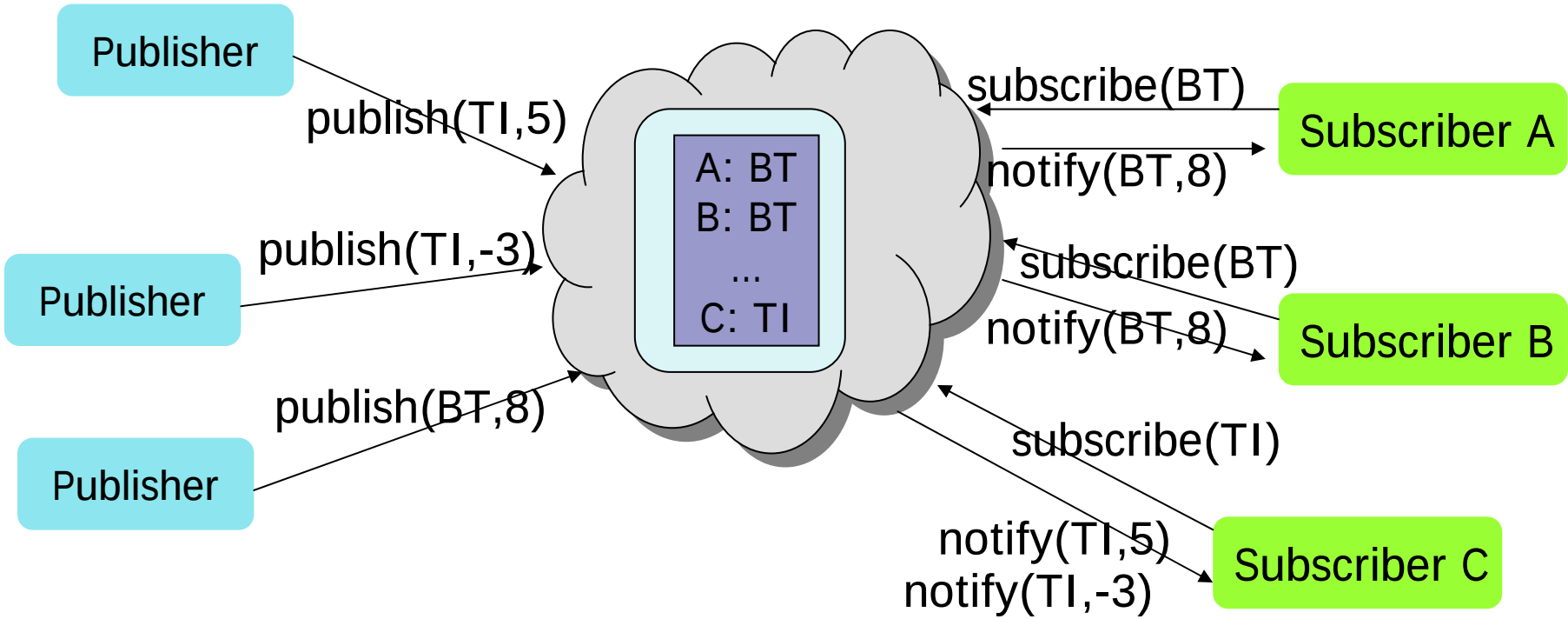
- **publish():** invoked by publishers to generate an event
- **notify():** invoked by the ENS to propagates the event to all relevant subscribers
- Subscribers will receive an event for every event conforming to its interest



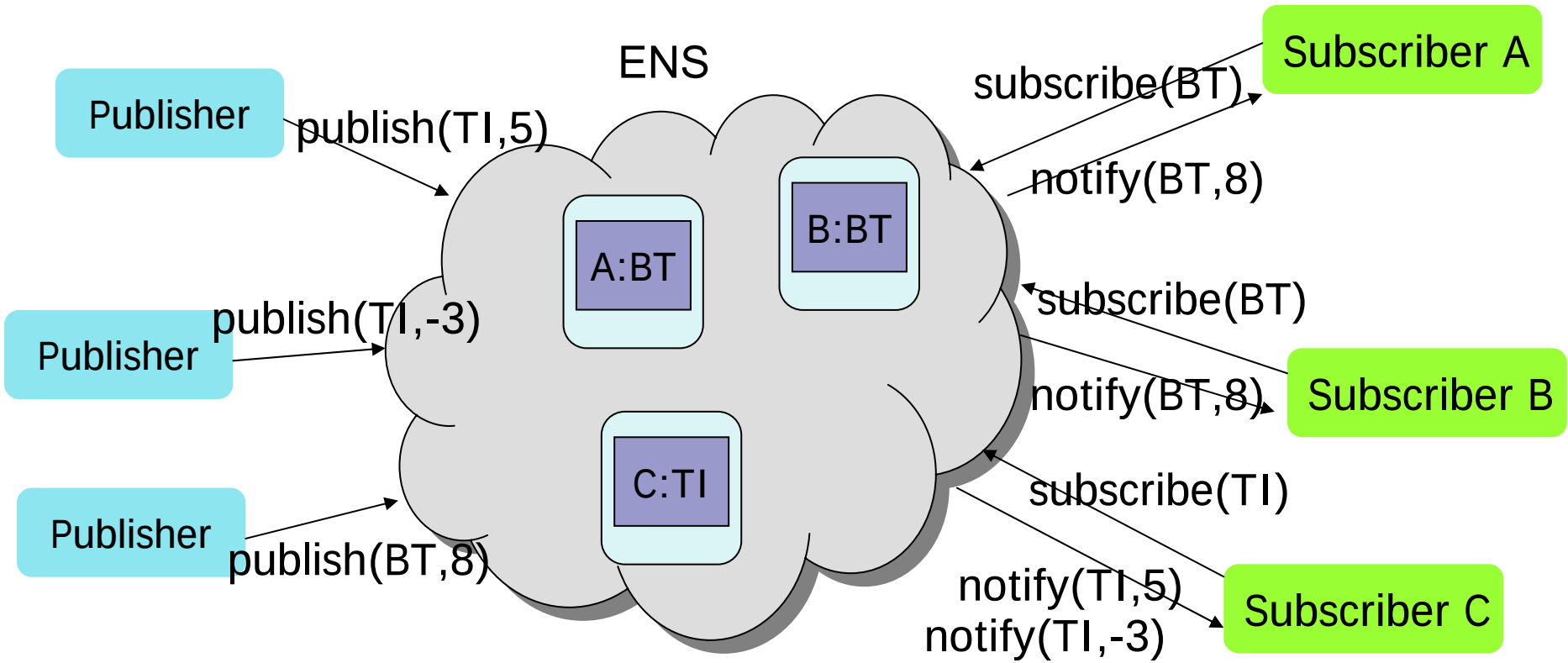
Event Notification Service Architecture

- Centralized
- Distributed

Centralized Event Notification Service



Distributed Event Notification Service

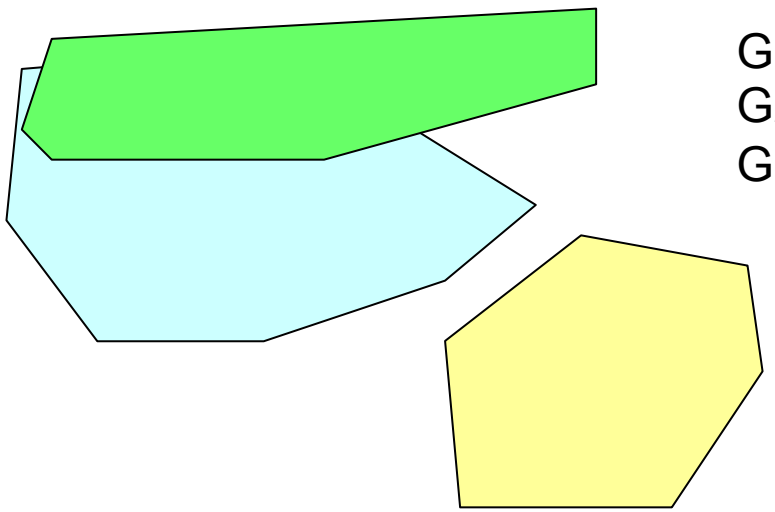


The subscription scheme

- The set of rules to specify the events of interest
  - Subscribers are usually interested in particular events or event patterns, and not in all events.
- Subscription scheme are characterized on the basis of their expressiveness
  - **Topic-based subscription**
  - **Content-based subscription**
  - **Type-based subscription**

Topic-based subscription scheme

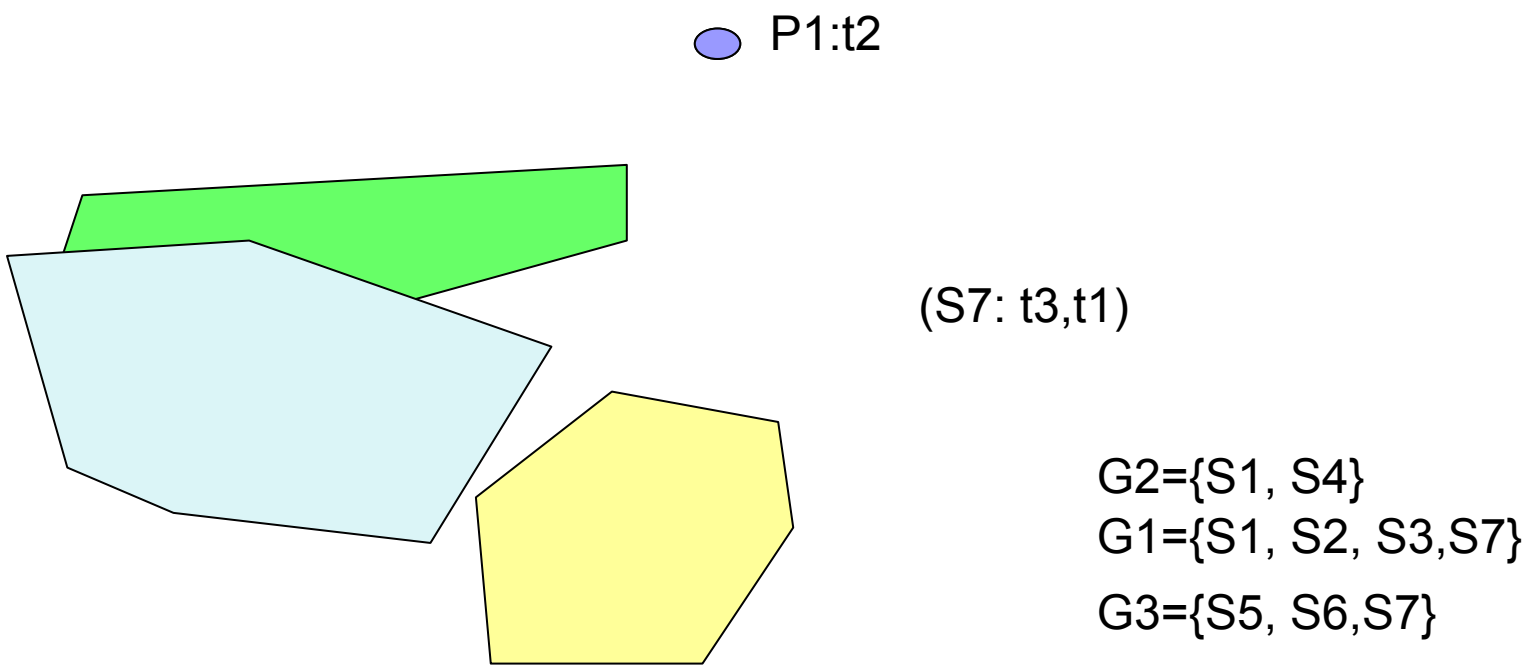
- Events are classified according to a predefined set of topics T={t1, t2, …, tn}
  - The Event space is partitioned according to the topics of interested
  - Topics are identified by keywords
  - Topics are strongly similar to the notion of groups.



G1={S1, S2, S3}
G2={S1, S4}
G3={S5, S6}

Topic-based subscription scheme

- Subscribing to a topic ti can be viewed as becoming member of a group G(ti).
- Publishing an event e on topic ti results into broadcasting e among the members of G(ti)

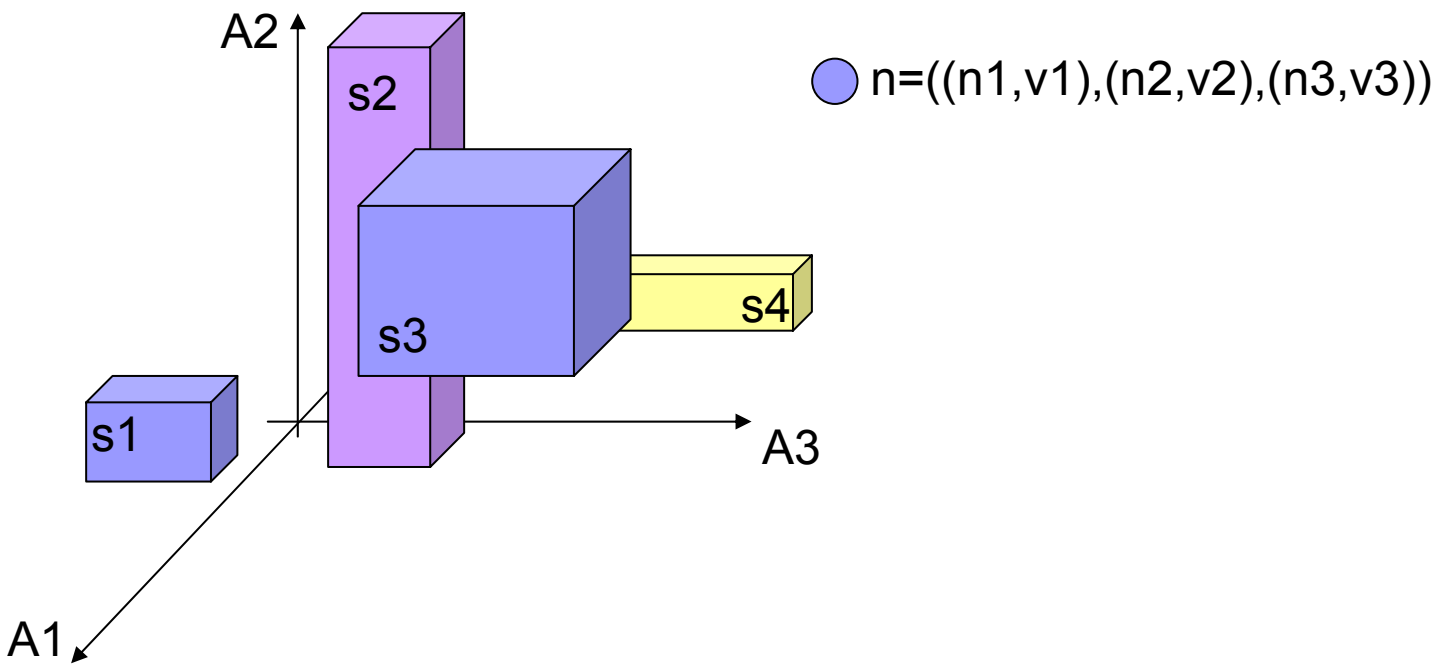


Content-based subscription scheme

- Events are classified according to the properties of the events themselves (e.g. their values)
- Subscribers express their interest by specifying selection predicated (filters), i.e. constraints over the content of notifications they want to receive
  - The filters define constraints, usually in the form of name-value pairs of properties and basic comparison operators (=, <, , >,
  - Predicates are defined through a specific language (e.g. SQL).
- Publishers just publish notification of event as a set of attribute/value pairs
  - Notification of events are matched to subscriptions on the basis of their content

Content-based Data model (I)

- A notification is a message that contains information about an event that has occurred.
  - A notification n is a set of attributes {A1, A2, …, An} where each A_i is a name value pair (ni, vi) with name ni and value vi



Content-based Data model (II)

- A filter F is a stateless boolean function that is applied to a notification, i.e. $F(n) \rightarrow \{true, false\}$
  - A notification matches F if $F(n) = true$.
- A filter consisting of a single atomic predicate is a simple filter or constraint.
- Filters that are derived from simple filters by combining them with boolean operators are compound filters
- Filters are used to define subscriptions and advertisements
  - A subscription S is a filter that is issued by a consumer to indicate its interest to receive future notifications.
  - Advertisements are filters that are issued by producers to indicate their intention to publish future notifications.

Content-based Data model (III)

- Subscription S:

  Alert-type="intrusion" AND
  severity=3 AND
  device-type="web-server" AND
  ((time= "0:00-10:00" AND days="Tuesday, Wednesday") OR
  (time="11:00-22:00" AND days="Monday"))

- Notification A:

  Alert-type="intrusion"
  severity=3
  device-type="web-server"
  time= "5:00"
  days="Tuesday"

- Notification B:

  Alert-type="intrusion"
  severity=3
  device-type="web-server"
  time= "12:00"
  days="Tuesday"

- **Notification Matching: F(A)=true F(B)=false**

Type-based subscription scheme

- Events are objects belonging to a specific type, which can thus encapsulate
  - attributes and
  - methods.
- The declaration of a desired type is the main discriminating attribute.
- This schema is suitable to distribute applications or executable objects

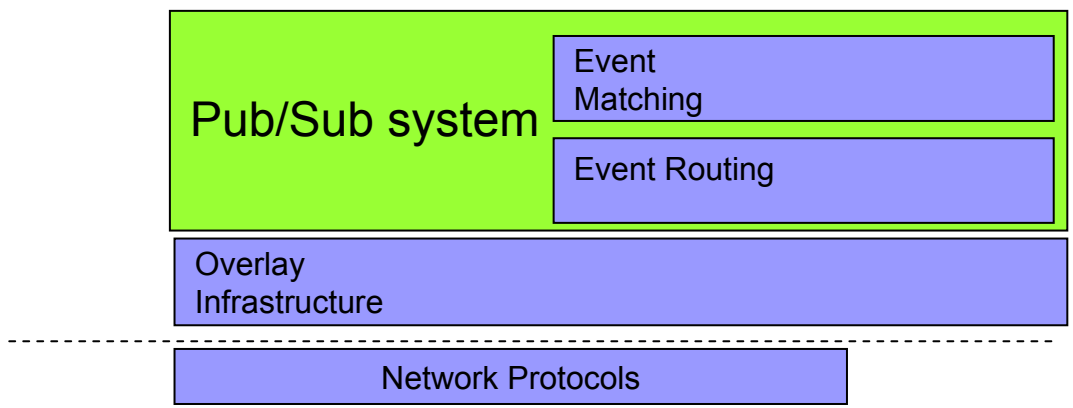
Topic-based vs Content-based subscription scheme

- Topic-based
  - Efficient implementations (multicast research area)
  - Limited expressiveness to the subscribers
- Content-based
  - More powerful in terms of flexibility and expressiveness
  - It supports rich subscription languages
  - Difficult to implement

Scalability: Ability to provide an event routing service across a whole wide-area network.
Expressiveness: Ability do with the type of information that can be attached and propagated with notifications as well as the ability of the event service to aggregate and filter events as precisely as possible on behalf of applications

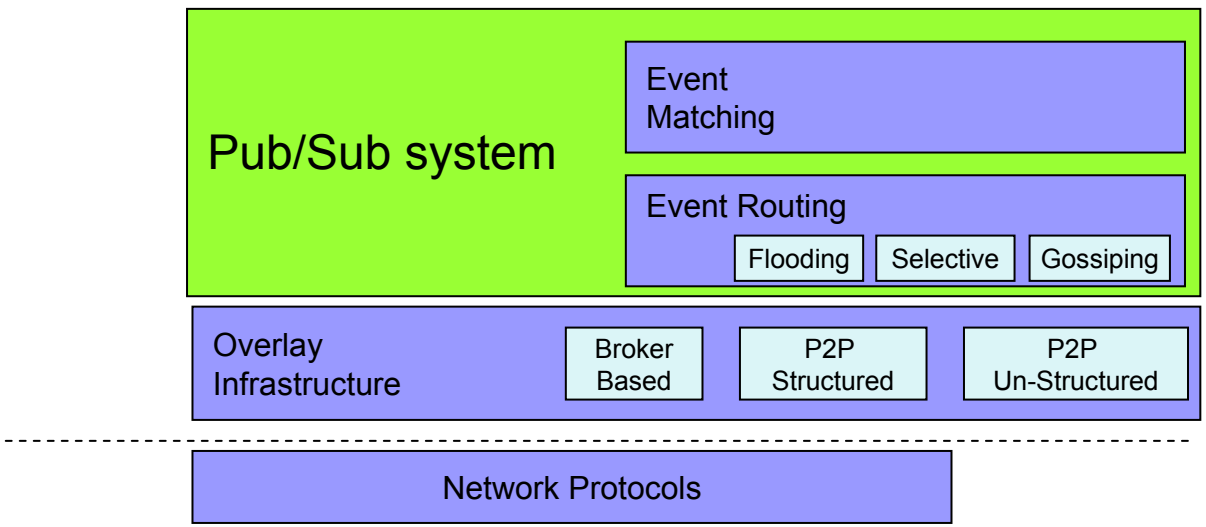
A multilayer architecture

- A pub/sub system is implemented on top of an application level overlay network
- Nodes composing the overlay
  - are arranged into an interconnection topology and
  - make use of a overlay routing protocol.
- The overlay topology and routing protocol have a heavy impact on the pub/sub system architecture
- The application level overlay network rely on a TCP/IP (or equivalent) communication network



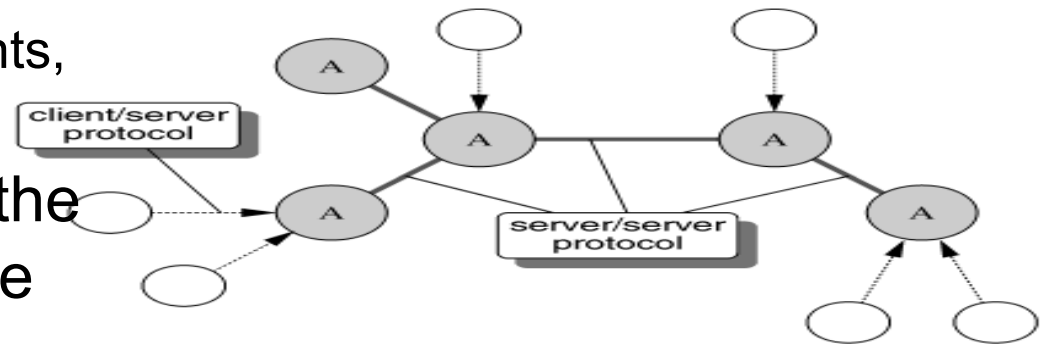
A multilayer architecture

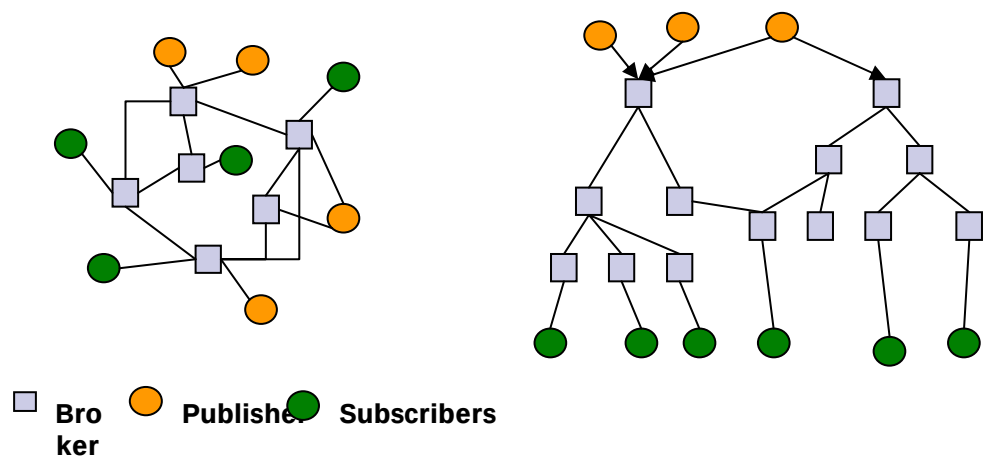
- The overlay topology (and overlay routing protocols) are classified as
  - Overlay of Brokers (broker-based)
  - P2P structured overlay network
  - P2P unstructured overlay network



Broker-based overlay

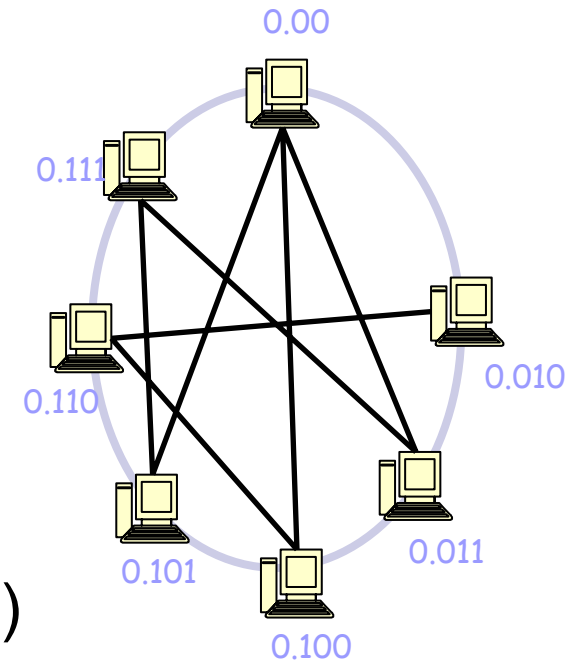
- Symmetric communication among brokers, to exchage
  - subscriptions, advertisements, and notifications
- Publisher/Subscriber join the system connecting with the brokers
- Static configuration with respect to nodes dynamics
  - Centralized Management
  - Hierarchical or flat topology





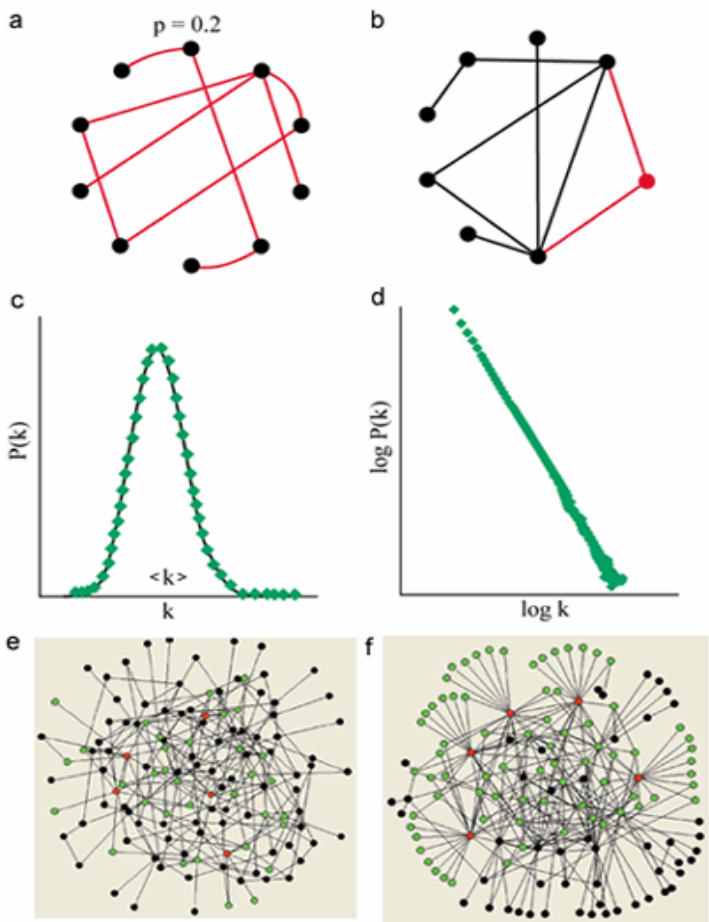
P2P structured overlay network

- Self-organizing application-level network composed of P2P nodes
  - nodes are organized as a structured graph
  - Each node is represented on a virtual key space by a unique node ID, forming a Distributed Hash Table (DHT)
- appropriate for dynamic systems
  - joining/leaving nodes
  - Self management property (i.e. automatic reconfiguration in case of nodes failures

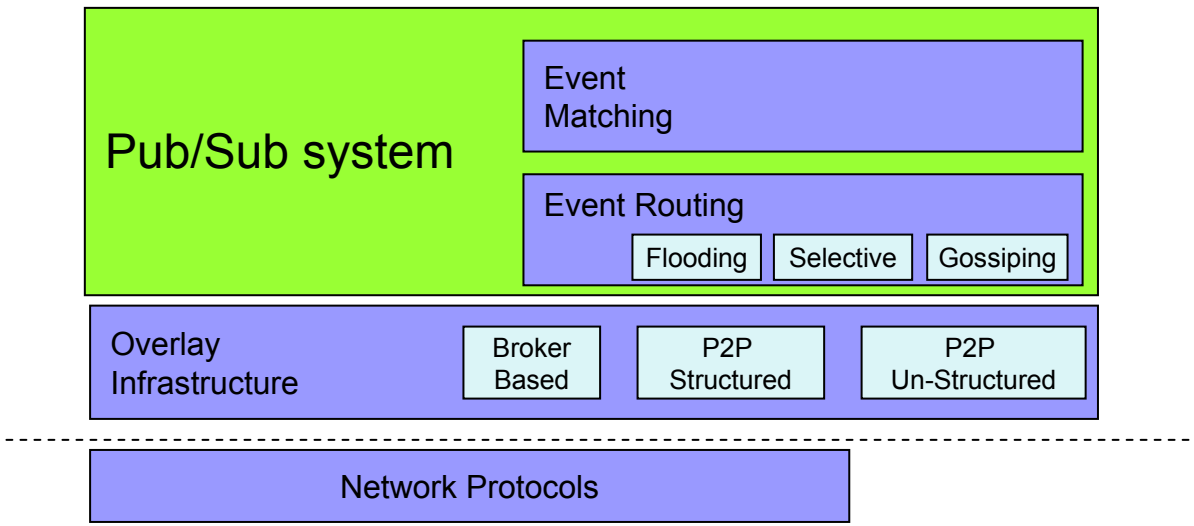


P2P unstructured overlay network

- Nodes are organized into random graphs
  - low diameter
  - high connectivity
- Good structure for
  - broadcast communication
  - system characterized by very high dynamicity
- Gossip based protocols
  - to maintain connectivity
  - to disseminate information



A multilayer architecture

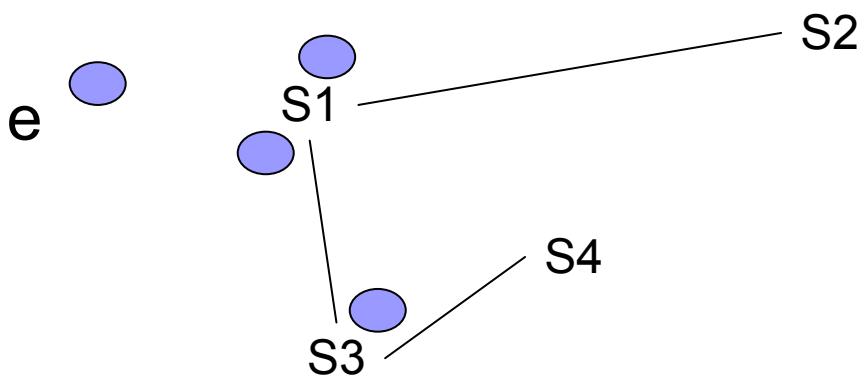


Event routing

- Responsible to deliver an event to the subscribers really interested on it (and that issued a subscription before the event publication)
- Solutions are classified as
  - Flooding algorithms
  - Selective algorithms
    - Filtering-based routing
    - Rendezvous-based routing
  - Gossip based algorithms

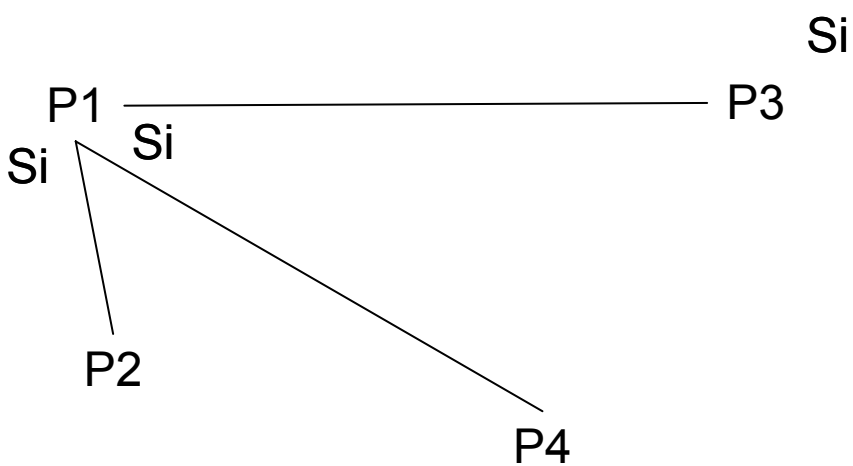
Flooding algorithms

- Event flooding
  - Each event is diffused within the network
  - The matching process happens at the interested subscriber site
  - Small memory overhead
  - Large communication overhead (generated for event diffusion)
  - Not scalable in terms of message overhead



Flooding algorithms

- Subscription flooding
  - Each subscription is diffused within the network
  - The matching process happens at the publishers site and then, the event is sent to interested subscribers
  - Small communication overhead generated
  - Un-pratical for dynamic systems



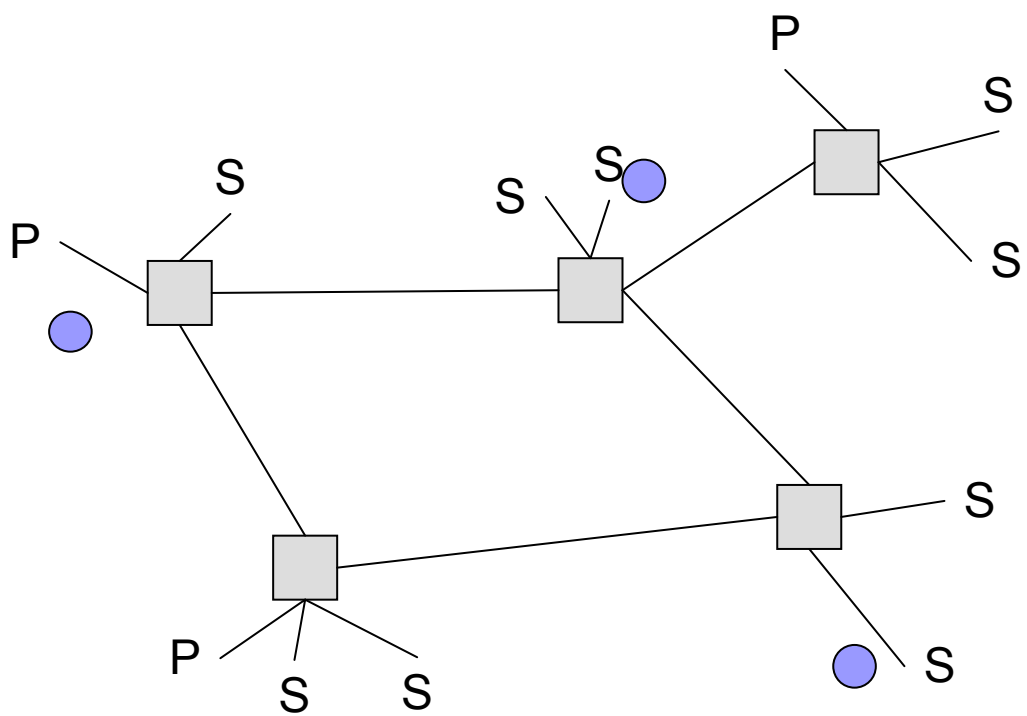
Selective algorithms

- Filtering-based routing
- Rendezvous-based routing

Filtering-based routing

- Events forwarding based on diffusion paths
  - Diffusion path permit to forward events to broker nodes leading to the interested subscribers
- Data structures are maintained on the routing nodes
  - Routing table contains the neighbors and associated set of subscriptions reachable
  - Subscription list contains the subscriptions maintained by the node
  - Need to maintain the consistency trough specific algorithms
- Broker-based architecture is more appropriate
  - Acyclic topology avoiding duplicate message diffusion (both subscriptions and events)
- Reduction of the message overhead during the event flooding phases
- Not appropriate in very dynamic configurations

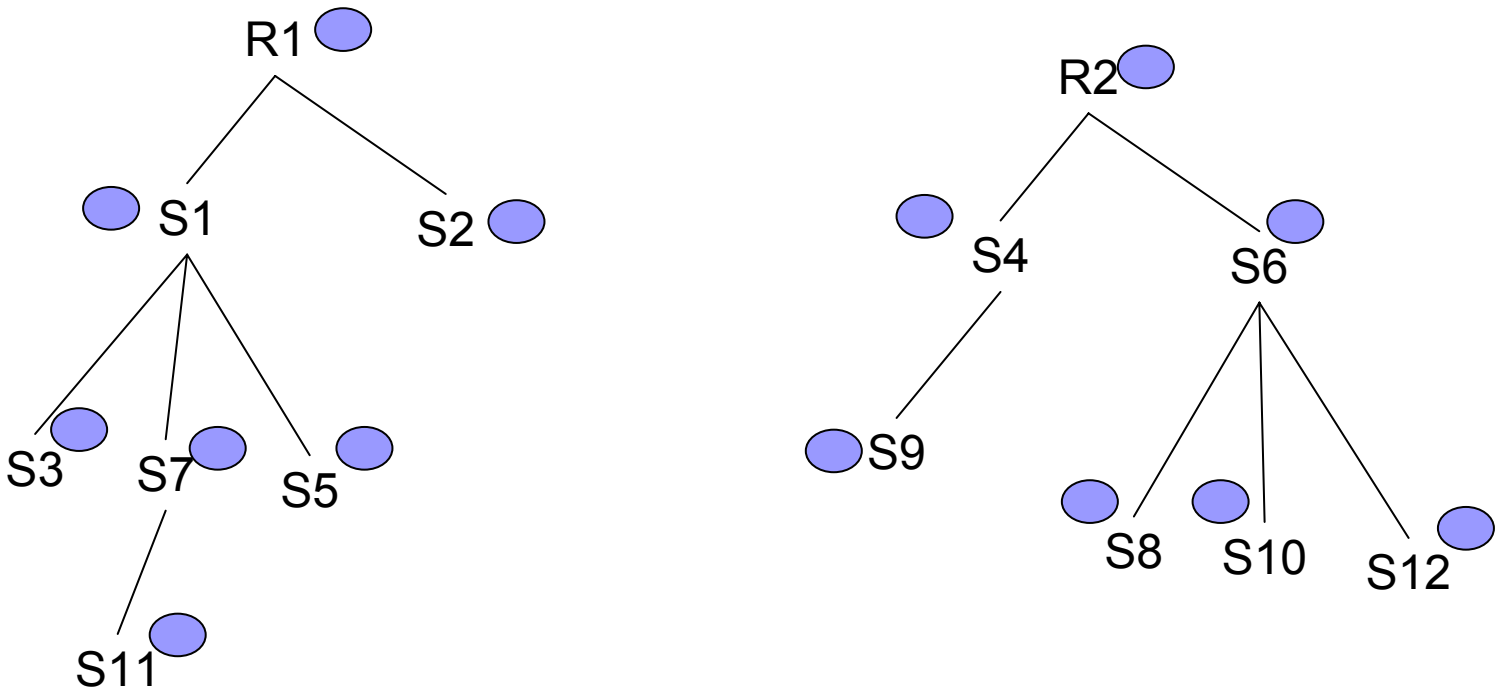
Filtering-based routing



Rendezvous-based routing

- The subscriptions and the events are associated to some nodes (rendez-vous nodes), responsible to manage them
- The rendez-vous nodes are responsible of storing subscriptions, matching events and forwarding to corresponding subscribers
  - The events are sent to the nodes responsible to disseminate the event to the appropriate subscribers.
  - These responsible nodes disseminate the event to the susbscribers
- Each rendez-vous node is a root of diffusion tree
- Structured overlay networks can be used, exploiting the logical structure of the overlay network representation and guarantying good dynamicity

Rendezvous-based routing



Gossip based algorithms (I)

- The dissemination of information happens as a diffusion of epidemic:
  - To broadcast a message, a node sends the message to a randomly selected subset of nodes.
  - Each node that receives the message, sends the message to a randomly selected subset of nodes
  - The size of the subset is the fan out
  - Every node might gossip the same information only a limited number of times (repetition) and/or the same information might be forwarded only a limited number of times (hops)

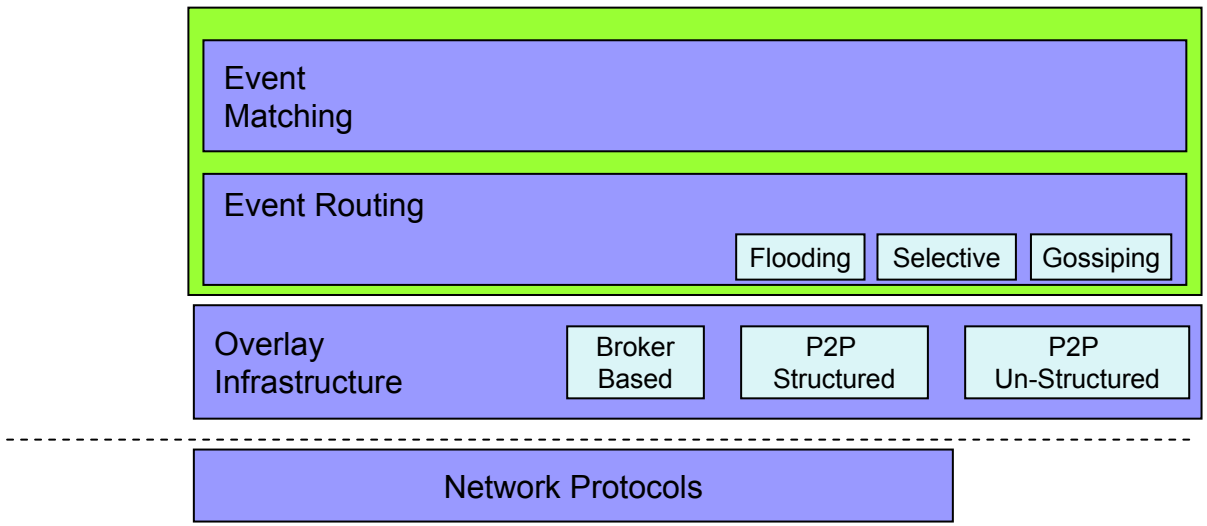
Gossip-based algorithms: properties

- The decentralization is the key concept
  - High scalability
  - Simple communication protocol based on decentralized broadcast algorithm
- Resource management
  - No need to store locally routing data structure
  - Message redundancy and overhead
  - Message buffering: Mechanism to remove message from buffers
  - Membership mechanism should maintain some important topology properties (low diameter, high connectivity)

Selective-based vs Gossip-based

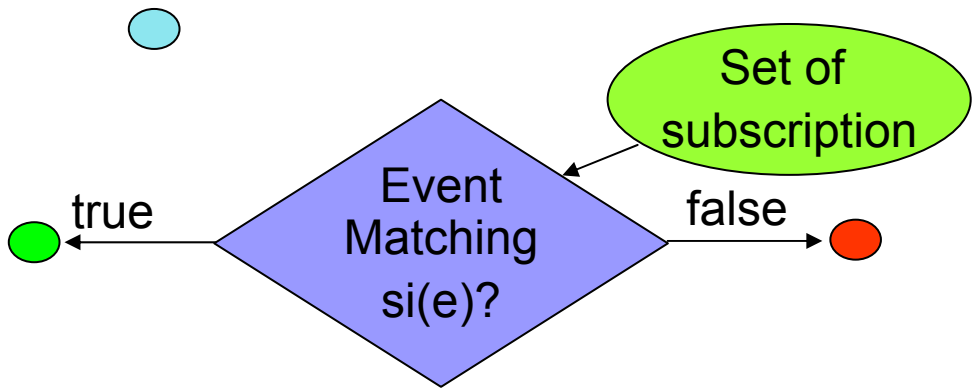
- Selective-based approaches
  - Deterministic solutions
  - Topology reconfiguration causes inefficiency → best effort, no guarantees during the reconfiguration time delay
- Gossip-based approaches
  - Probabilistic solutions
  - More appropriate for system characterized by high churn

A multilayer architecture



Event Matching

- Given an event *e* and a subscription *s* (a boolean predicate on events)
  - *e* matchs *s iff s(e)=true*
- determine the set of subscription s that are matched by e

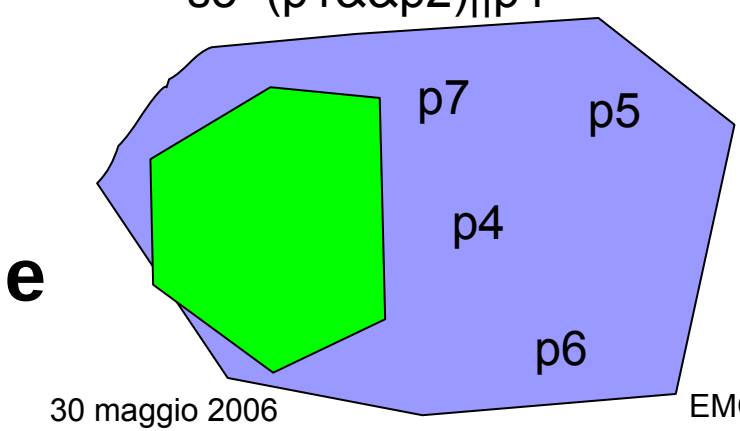


- Two approach
  - Predicate indexing based algorithms
  - Testing network based algorithms

Event Matching: Predicate indexing based algorithms

- The first phase determines all the predicates (in all subscriptions) that are satisfied by the event.
- The second phase finds all the subscriptions that are matched by the event based on the results of the first phase.

s1=p1&&(p2||p4)
s2=p1&&p4
s3=(p1&&p2)||p4



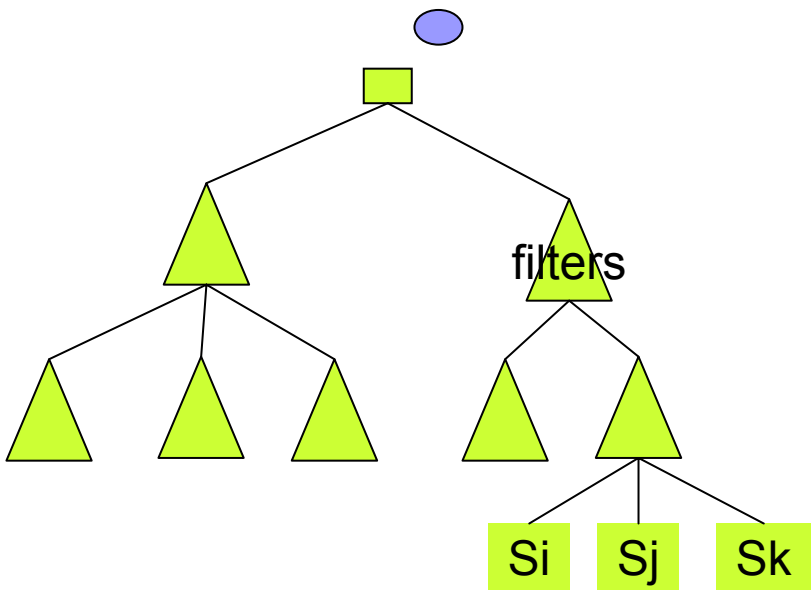
s1=p1&&(p2||p4)=TRUE

s2=p1&&p4=FALSE

s3=(p1&&p2)||p4=TRUE

Event Matching: Testing network based algorithms

- pre-processes the set of subscriptions into a matching tree.
  - Events enter the tree at the root node and are filtered through by intermediate nodes.
  - An event that passes all intermediate testing nodes reaches a leaf node where a reference to a matching subscription is stored.



Three examples of real systems

- Scribe: topic-based / p2p structured overlay
- Siena: topic/content-based / broker-based
- Gossip based

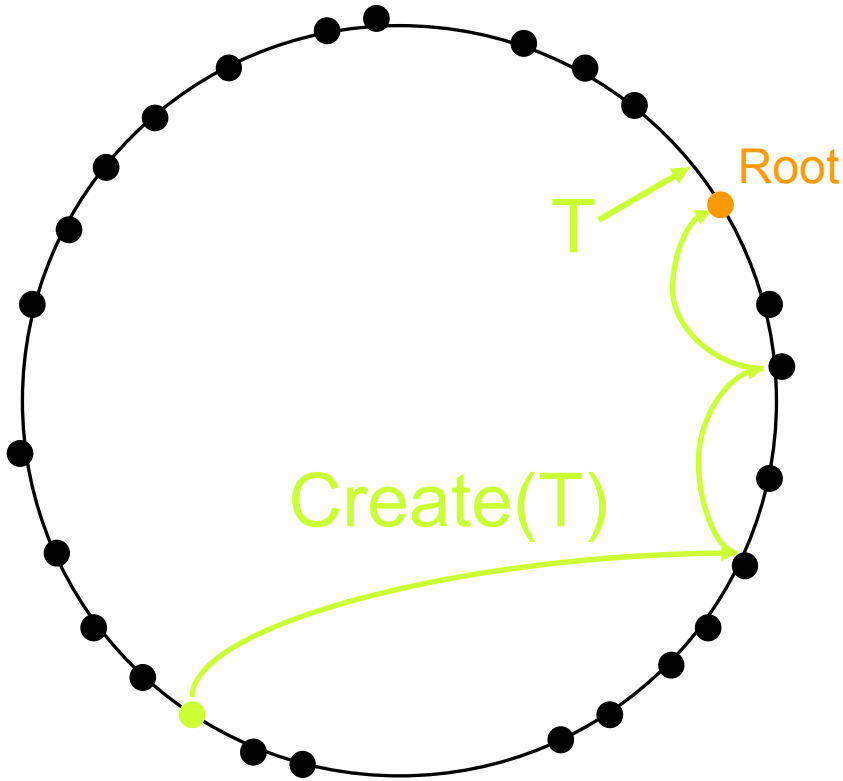
Scribe

- Large scale, decentralized event notification infrastructure (based on Pastry)
- Topic-based
  - Builds a multicast tree formed by joining the Pastry routes from each subscriber to a rendez-vous point associated with a topic
- Any Scribe node
  - can create a topic,
  - can become subscribers

Scribe: Implementation

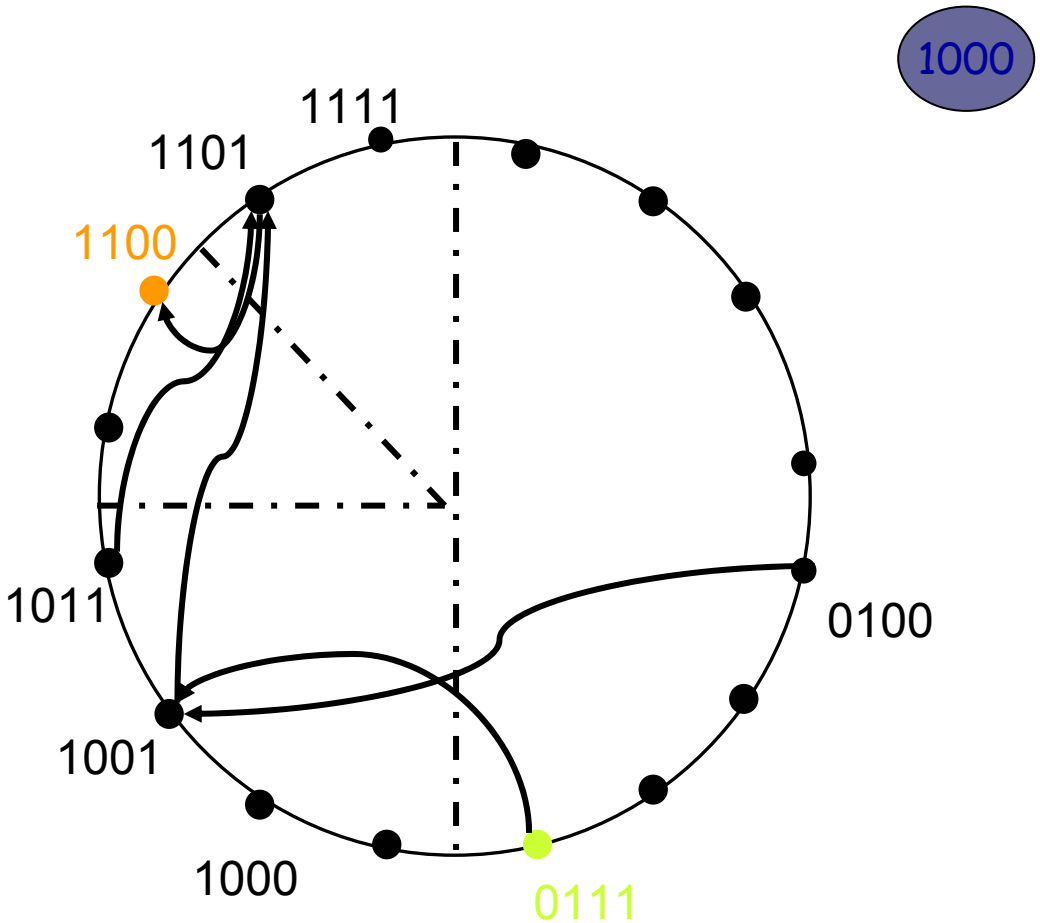
- *Topic* hashed to a key
- Construct a multicast tree based on the Pastry network
  - Have the (Pastry) node with the closest Id to the topic key be the root (rendez vous point for the topic)
  - Tree is formed by joining the routes from each subscriber to root
    - Interior nodes (forwarders) maintain a children table
- Pass event notification down through the tree
  - Each parent forwards event to it's children
  - Avoids over stressing network links close to the topic node

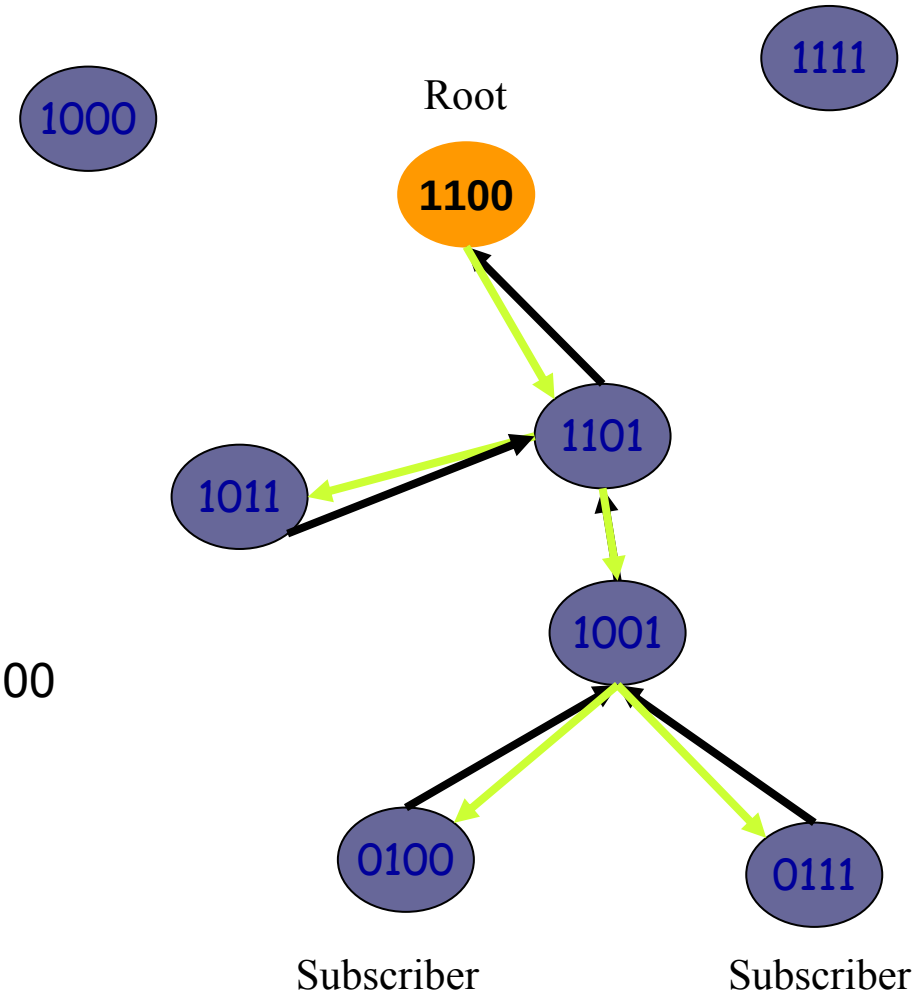
Scribe: Topic creation



- Each topic is assigned a topic-Id
- Root of the multicast tree= node with nodeId numerically closest
- Create(topic): route through Pastry to the topic-Id

Scribe: subscribing

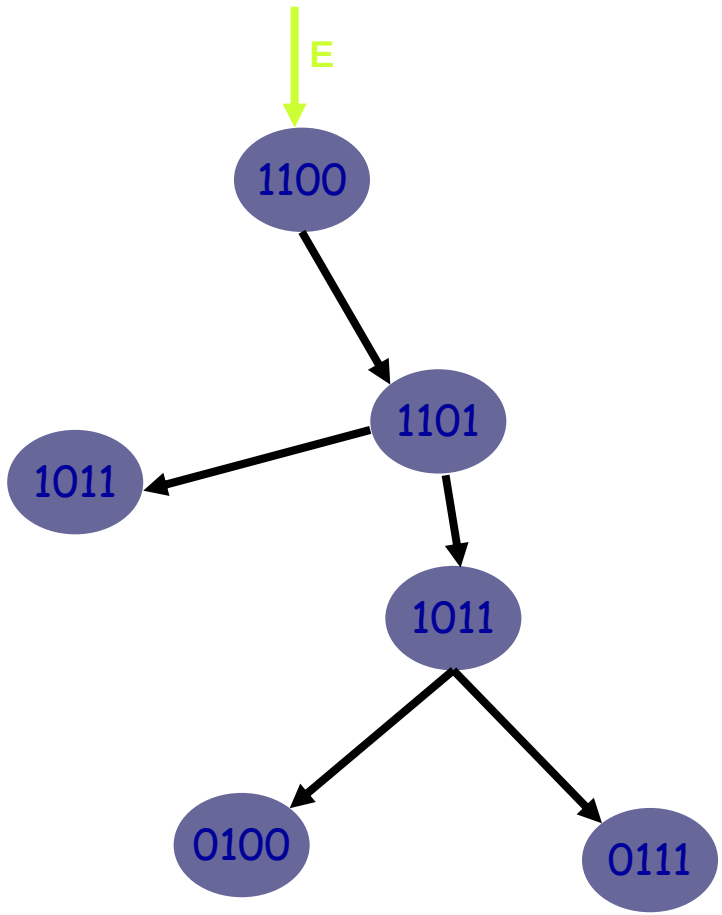




Scribe: event dissemination

Publish(topic, event)

- Route through the Pastry network using the topicId as the destination
- Dissemination along the multicast tree starting from the root



Scribe: Repairing the multicast tree (I)

- Periodically each non-leaf node send heartbeats to childrens
  - Events serve as implicit heartbeats
- Upon detection of failure of a parent, children send SUBSCRIBE (topicID)
  - Pastry routes message to a new parent
  - Example
    - node 1001 detects failure of 1101
    - Uses Pastry to route a SUBSCRIBE msg to root through alternative route (e.g. 1101)

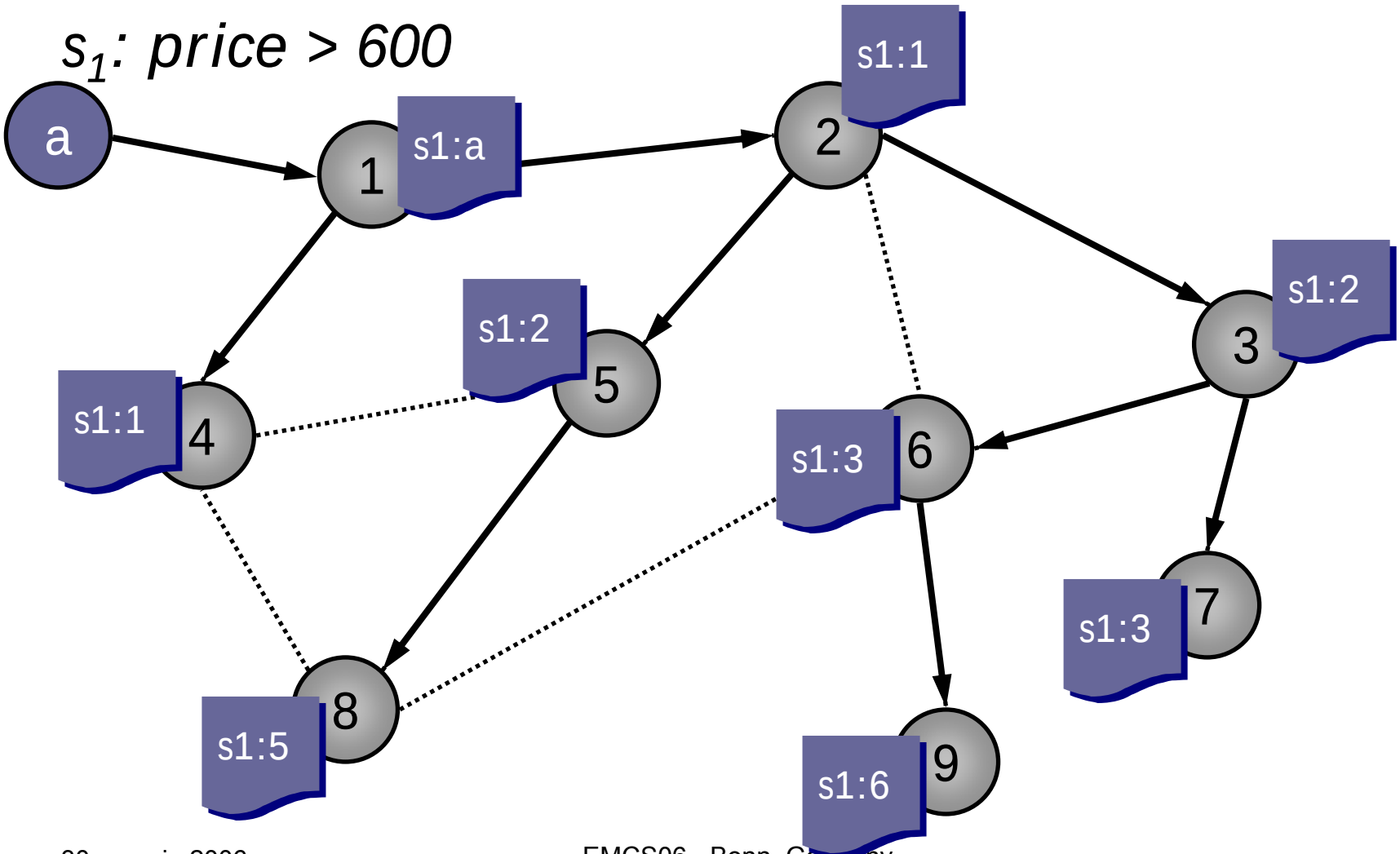
Scribe: Repairing the multicast tree (II)

- Failure of rendez-vous nodes is tolerated through replication of state information on k nearest nodes
- If route fails, children subscribe again through Pastry
  - Children exchange heartbeats with root
- Pastry routes to new root
  - Live node with closer nodeId to topicId takes over the role of rendez-vous point

SIENA Filtering-Based Routing

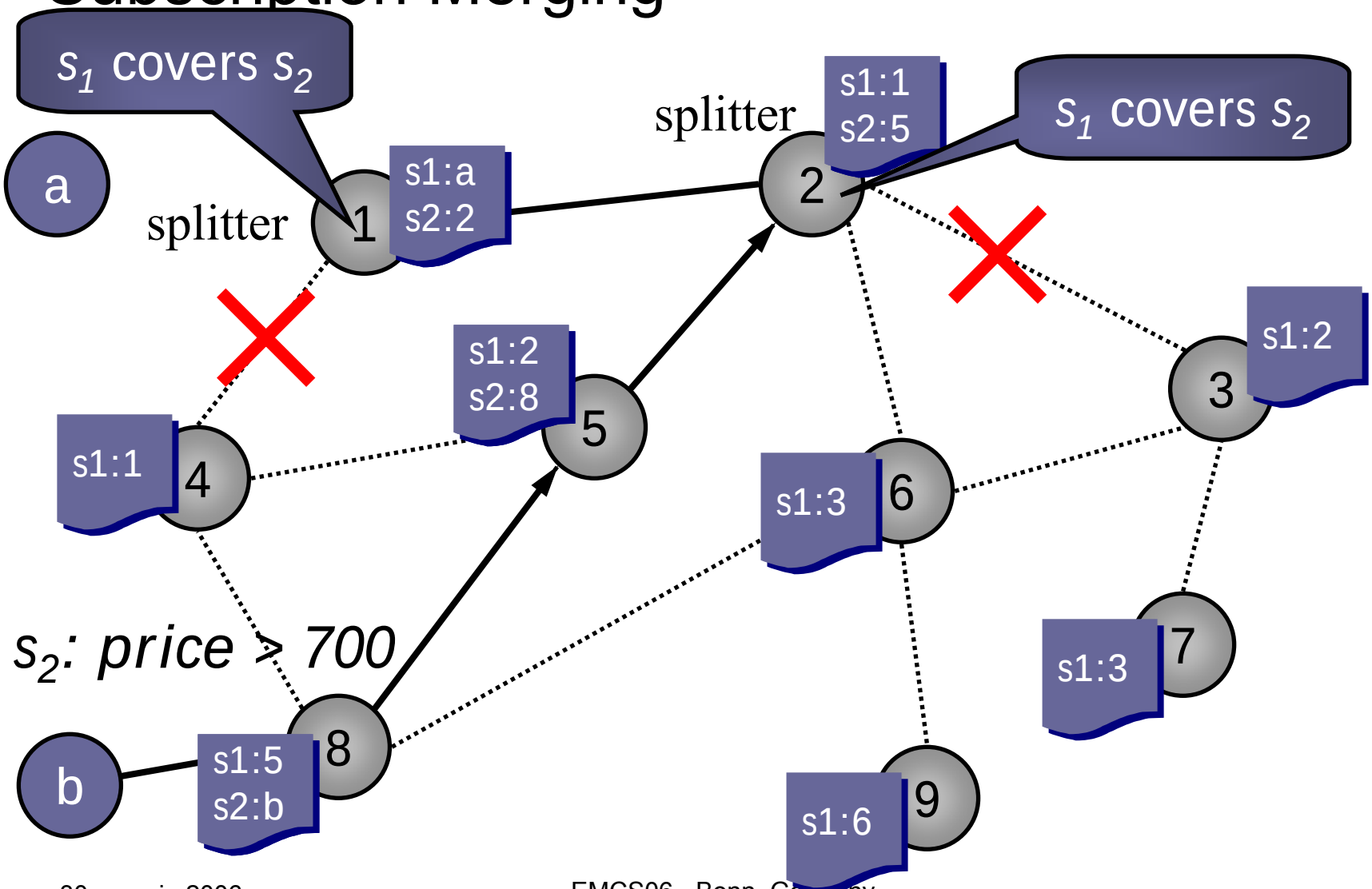
Subscription Forwarding

s_1: price > 600



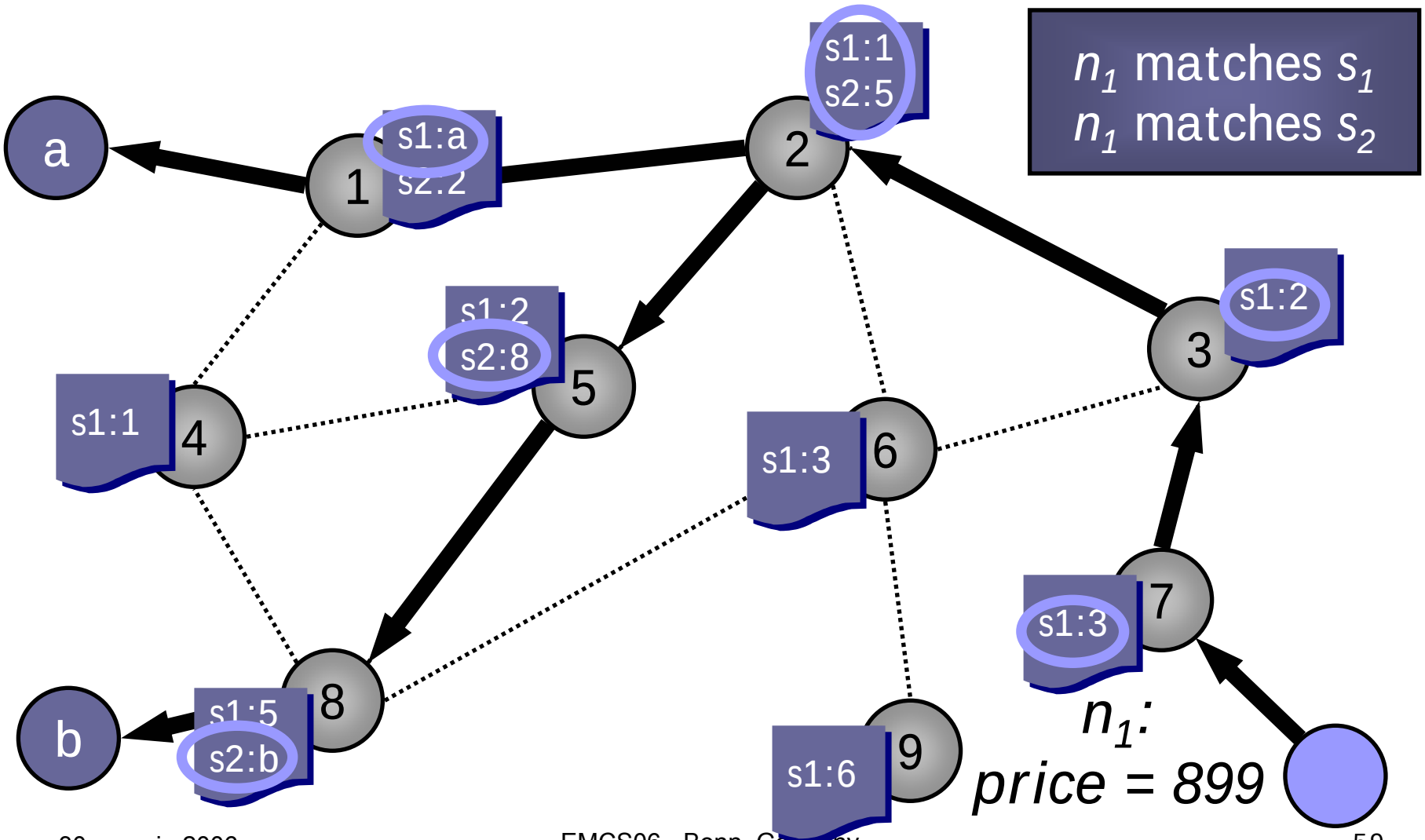
SIENA Filtering-Based Routing

Subscription Merging



SIENA Filtering-Based Routing

Notification Delivery



Gossip-based solution

- Semi-probabilistic content-based publish-subscribe, P. Costa, G.P. Picco
- Connectivity configuration of nodes can change freely and frequently
- Minimal assumption on the underlying overlay network→ connectivity of a random graph
- Semi-probabilistic Content-based routing strategy
  - Deterministic info about subscriptions in the vicinity of a dispatcher→ accurate information for routing events
  - Probabilistic routing decisions in the portion of network where the info is not available

Gossip-based solution: Subscription propagation

- Each dispatcher knows a limited portion of the interests of other dispatchers (subscription horizon φ: number of hops a subscription is propagated away from the subscriber)
- Propagation as a subscription forwarding,
  - Number of consecutive forwarding is φ
- Subscription table
  - Subscription and related link
  - Distance of the subscribers ([0, φ])

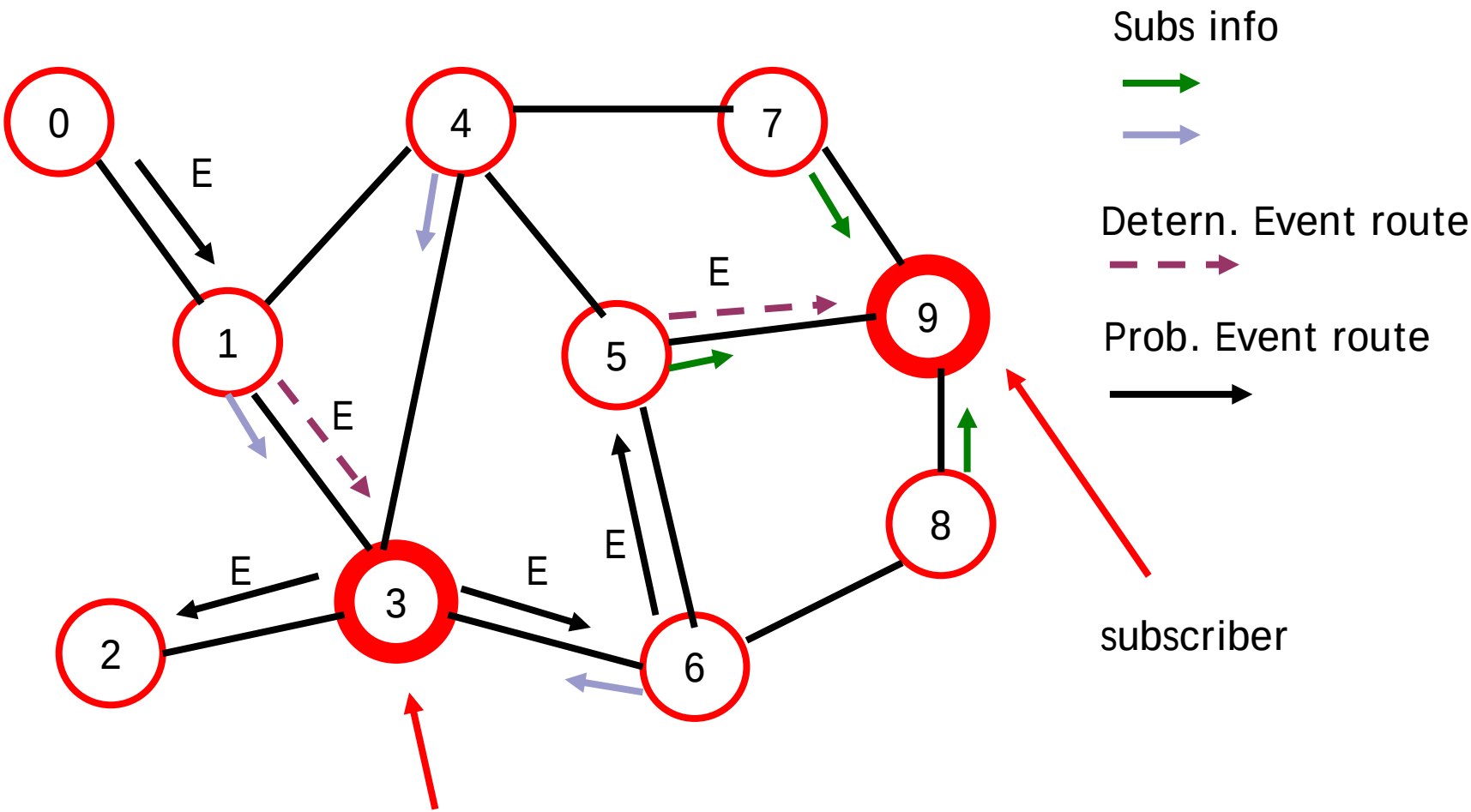
Gossip-based solution: event propagation

- Probabilistic and deterministic decision
- An event is forwarded along a subset of the outcome links from the dispatcher node
  - The subset cardinality is $f=\tau(l-1)$
- When a node receives an event, the subscription table is inspected
- <u>Deterministic event routing:</u> If a match exists, the event is routed along the link associated to the subscriptions (prioritizing according to φ)
  - This step is iterated until the number f is reached
- <u>Probabilistic event routing</u>: if the number of matching is not sufficient to reach f, the event is forwarded to a subset of randomly chosen nodes, as needed to reach f

Gossip-based solution: Example

Φ=1

τ=0.5



Gossip-based membership protocol

- Approximate a random graph trough View Shuffling
- Each node is equipped with a local view of the system and periodically makes this 3 steps protocol
  - Builds a list L_1 contains the local views of *f* random chosen nodes
  - Builds a list L_2 containing the nodes requesting its view
  - Creates a new local view by choosing *k* distinct nodes from L_1 and L_2, using probability *w* to select from L_2 rather than L_1

Bibliography

- “The Many Faces of Publish/Subscribe”, P.Th. Eugster, P.A. Felber, R. Guerraoui A.-M. Kermarrec
- “Design and Evaluation of a Wide-Area Event Notification Service”, A.CARZANIGA, D. ROSENBLUM, A. L. WOLF
- “Epidemic Algorithms for Reliable Content-Based Publish-Subscribe: An Evaluation, Paolo Costa”, Matteo Migliavacca, Gian Pietro Picco, Gianpaolo Cugola
- “Semi-probabilistic Content-Based Publish-Subscribe”, Paolo Costa and Gian Pietro Picco
- “Generic Constraints for Content-Based Publish/Subscribe”, Gero Mühl
- “SCRIBE: The design of a large-scale event notification infrastructure”, Antony Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel
- “Peer-to-peer membership management for gossip-based protocols”, Ayalvadi J. Ganesh , Anne-Marie Kermarrec and Laurent Massouli
- “From Epidemics to Distributed Computing”, P.T. Eugster, R. Guerraoui, A.-M. Kermarrec, L. Massouli