**ECMS06**
***2006, Bonn, Germany***

# THROUGH THE EVOLUTION OF PUBLISH/SUBSCRIBE SYSTEMS – PART III

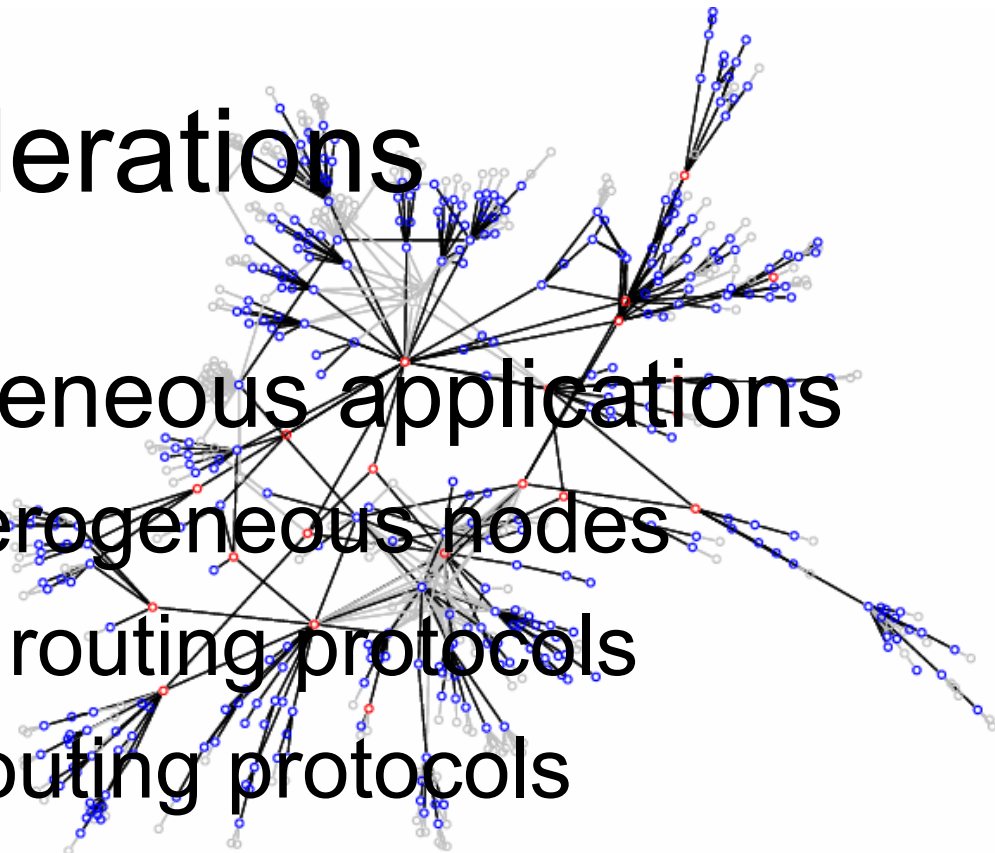
Emiliano Casalicchio (casalicchio@ing.uniroma2.it)

Federico Morabito (federico.morabito@telecomitalia.it)

# 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

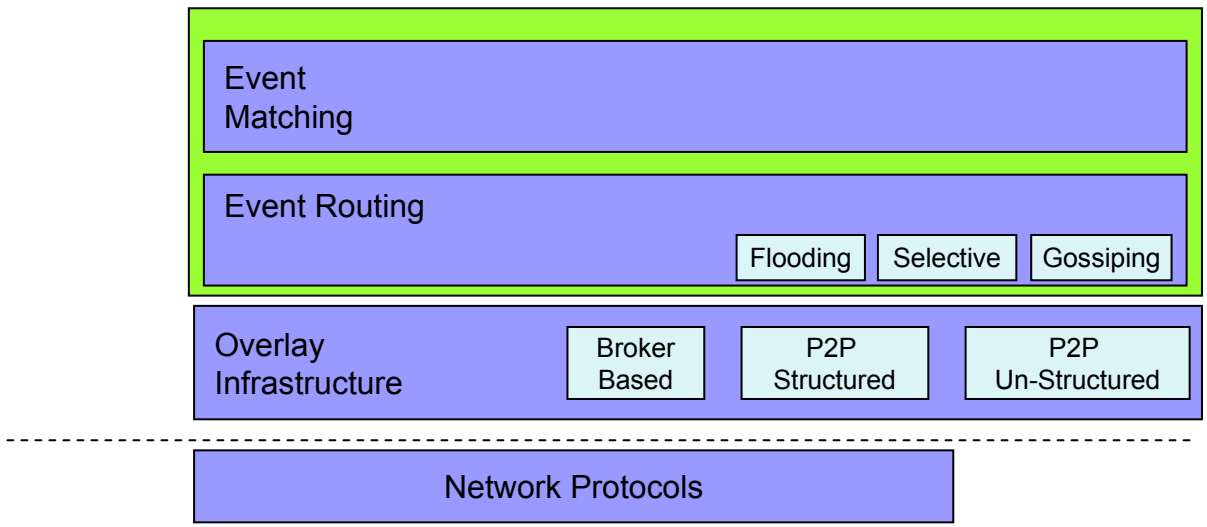
# General Considerations



- Complex heterogeneous applications
  - Thousand of heterogeneous nodes
  - Complex overlay routing protocols
  - Complex event routing protocols
  - Complex event matching algorithms

# General Considerations

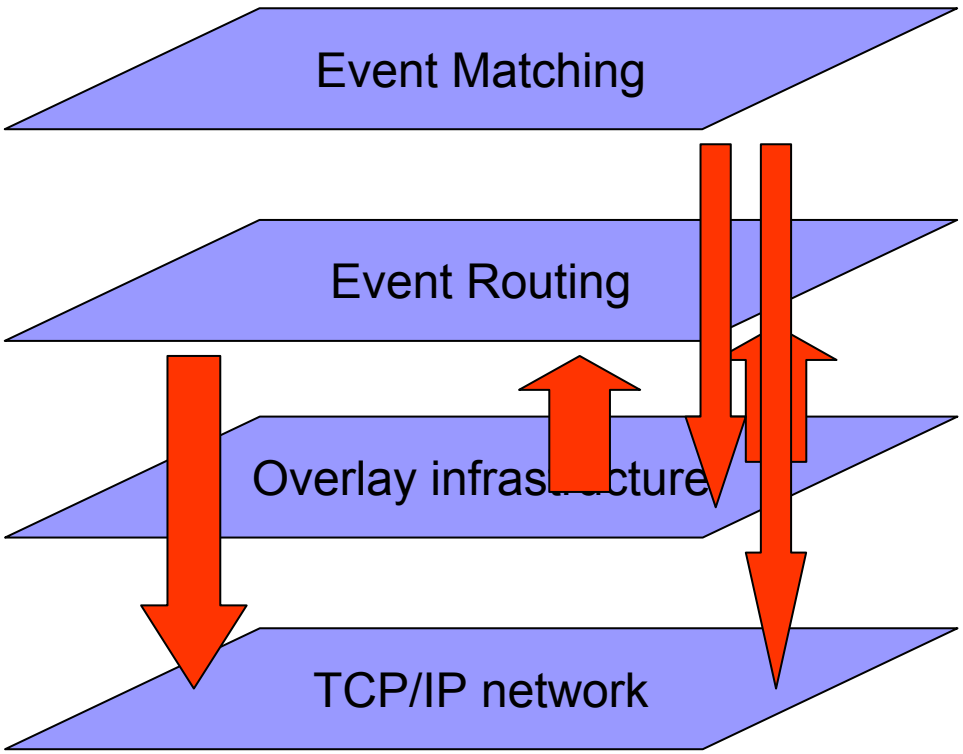
- A multilayer architecture




  - It requires a multilayer modeling approach

# General Consideration

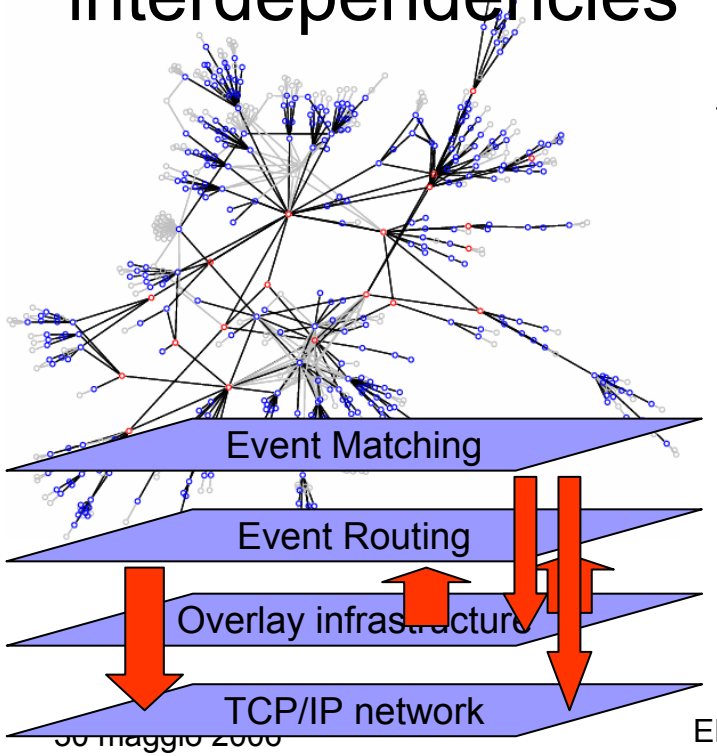
- System components at each layer are interdependent




- Examples
  - Forwarding and flooding have different performance if implemented on a broker based or acyclic p2p net
  - Data structure used in ER impact the performances of Communication network (in term of cost)
  - The overlay infrastructure dynamic has an impact on ER protocol scalability
  - Event Matching impact the unnecessary message traffic…
  - Etc…

# General Consideration

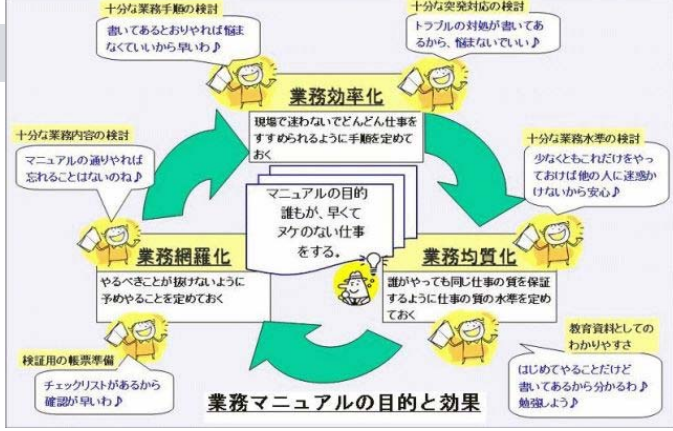
- Complex system
- Multilayer architecture
- Component interdependencies



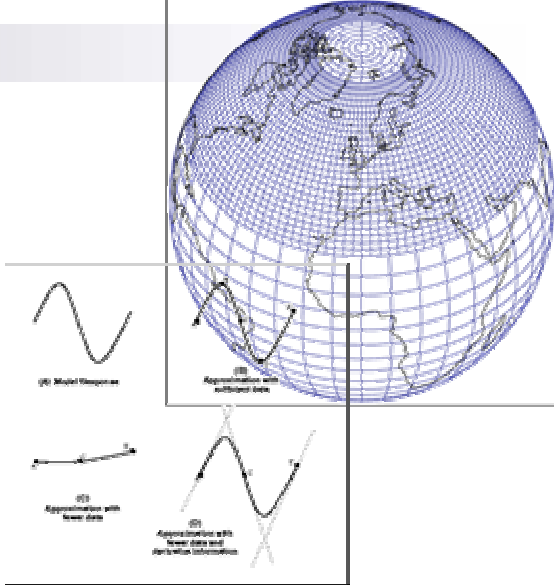

- different modeling techniques
- system level modeling
- simulation

# What to model

- Overlay infrastructure layer
  - node architecture
  - Network topology
  - routing protocols
- Event Routing layer
  - Protocol behavior,
  - Overlay topology
- Event Matching layer
  - Algorithm behavior and complexity
- Workload
  - Overlay routing
  - Event routing
  - Event matching



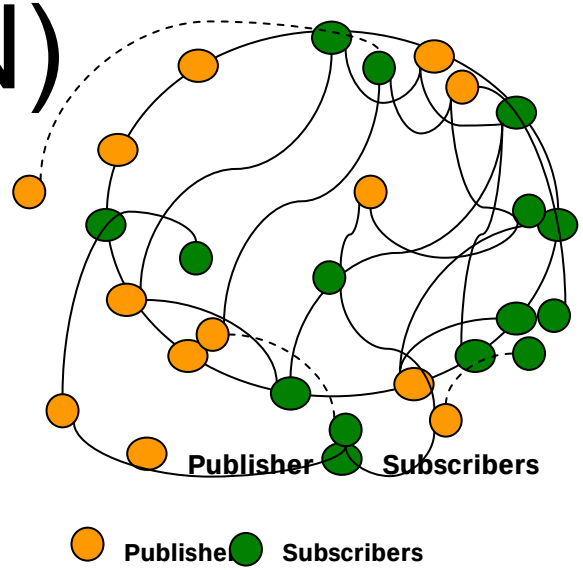
# How to model

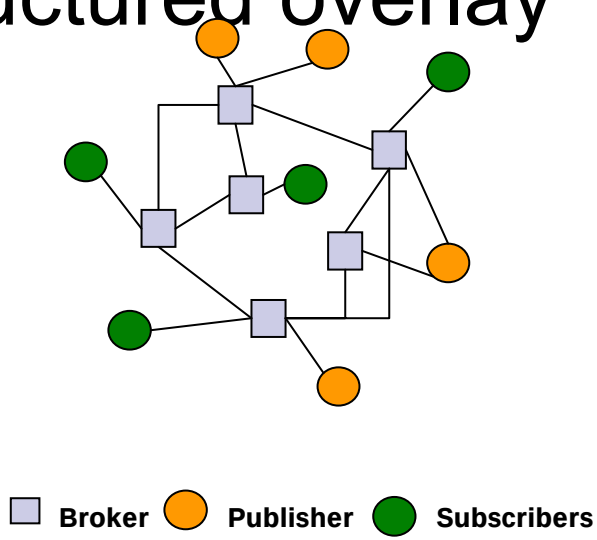


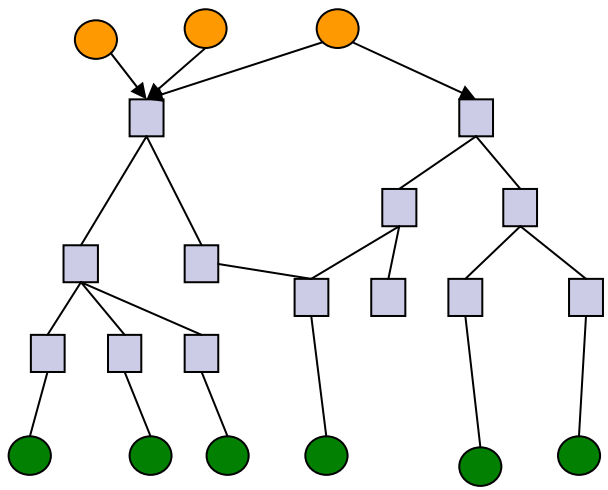
- Overlay Infrastructure
  - Operational analysis, QN
  - Discrete Event Simulation
- Event Routing
  - formal methods to verify the protocol correctness, robustness, etc… (model checking, PN, FSM)
  - Computational and Analytical models [BBTV05][BCTV01]
  - Discrete event simulation
- Event Matching
  - formal methods to verify the algorithms correctness, robustness, etc… (model checking, PN, FSA)
  - Computational model
  - Discrete event simulation

# Overlay Network (ON)

- Broker Overlay
  - Hierarchical
  - Flat
- p2p structured overlay
- p2p un-structured overlay



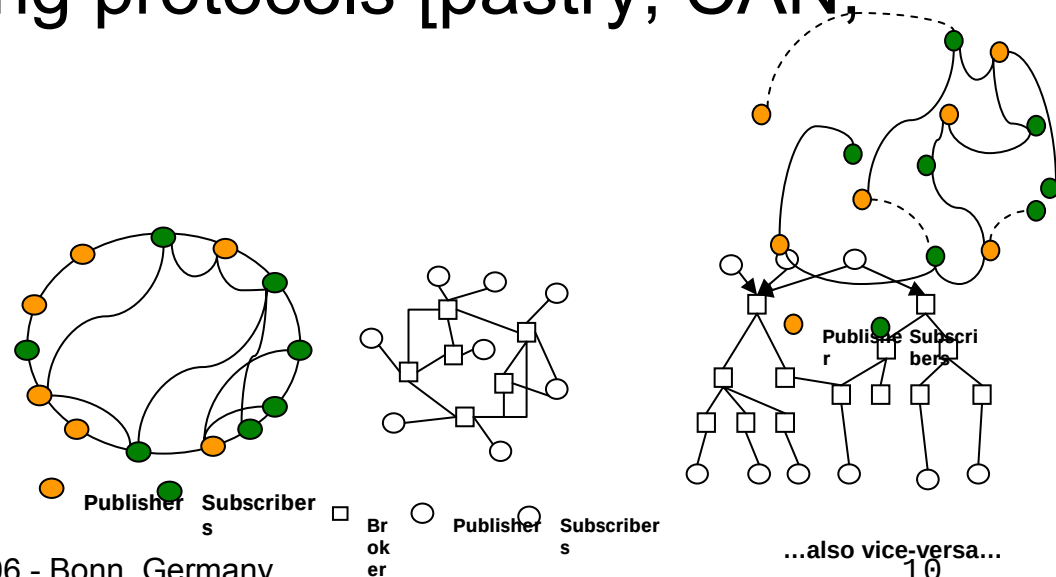



**...also vice-versa...**

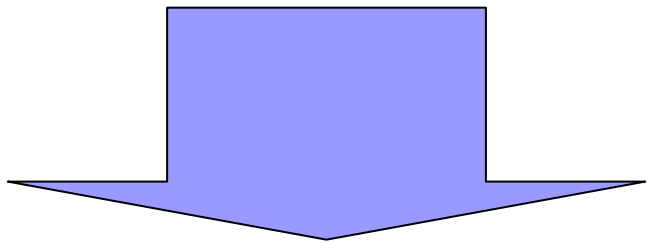
# ON: what to model

- Nodes (broker, subscriber, publisher)
- Overlay network
  - topology and
  - links (also wireless in case of unstructured overlay)
- Overlay network routing protocols [pastry, CAN, Chord, Tapestry]



# ON modeling: Issues

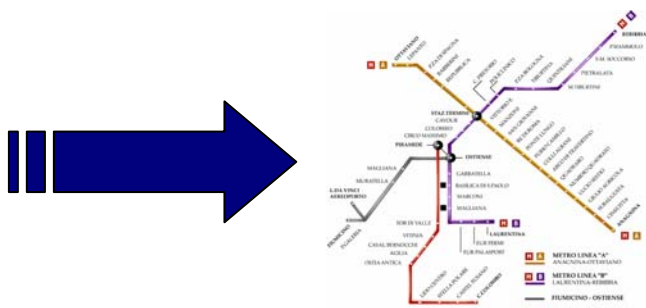
- Large number (~$10^6$-$10^{10}$) of components to model



- Large dimension of systems of equations to solve (for analytical models)
- High number of events to simulate
- Large number of parameters to tune
- Large number of dimension to explore

# ON modeling: Solutions

- Model simplification!!!!
  - System view rather then component view
  - Simplification hypothesis in
    - network modeling
    - routing protocols modeling
  - Simulation rather then analytical models



# Nodes model

broker, subscriber, publisher, root, etc…

- Discrete time queue model
- System level performance model
  - The nodes viewed as a black box
- Component level performance model
  - The node viewed as a network of components

# Node model: System level view

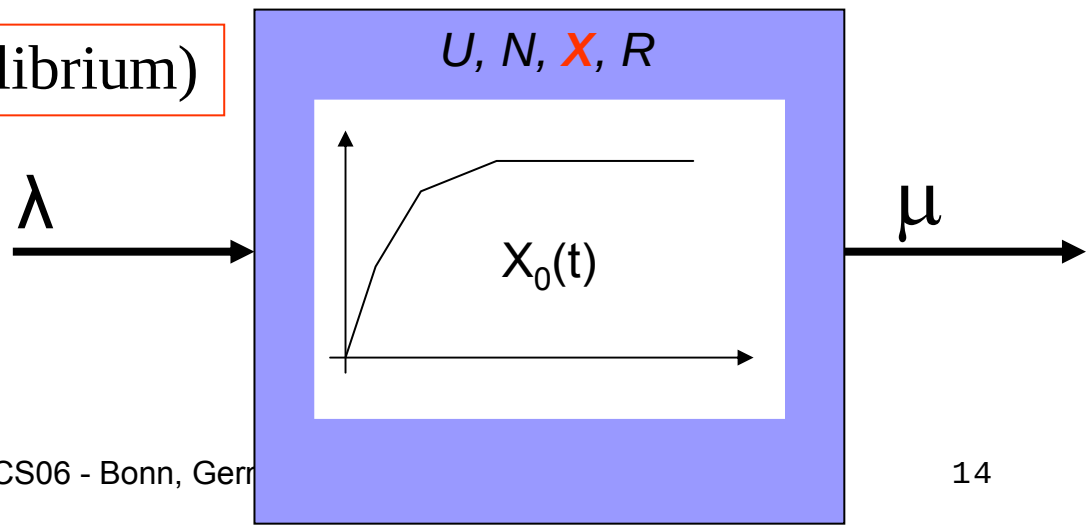
- The node is viewed as a black box
- We know only the system throughput $X_0(t)$
- e.g.
  - Infinite Population (homogeneous workload) vs Finite Population,
  - Infinite Queue vs Finite Queue
  - Fixed Service Rate vs Variable Service Rate
- Performance metrics:

$$\boxed{X = (\lambda / \mu) \times \mu = \lambda \text{ (at the equilibrium)}}$$

$$U = \lambda / \mu$$

$$N = U / (1 - U)$$

$$R = N / X$$



# Node model: Component level view

- The node viewed as a network of components (a queuing networks)

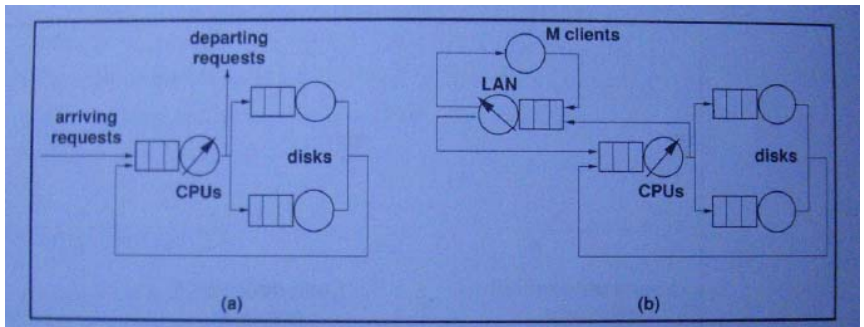



**Figure 9.2.** (a) Example of an open QN. (b) Example of a closed QN.

- The main node's components are modeled by
  - Load dependent
  - Load independent
  - Delay resources

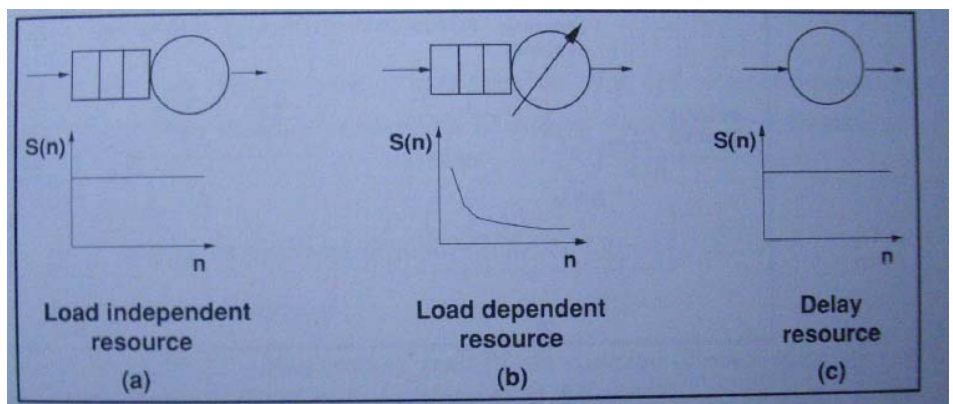



**Figure 9.1.** Types of resources in a queuing network.

# Node model: Component level view

- Open vs Closed model
- Single Class vs Multi Class model
- Infinite Queue vs Finite Queue
- Fixed Service Rate vs Variable Service Rate

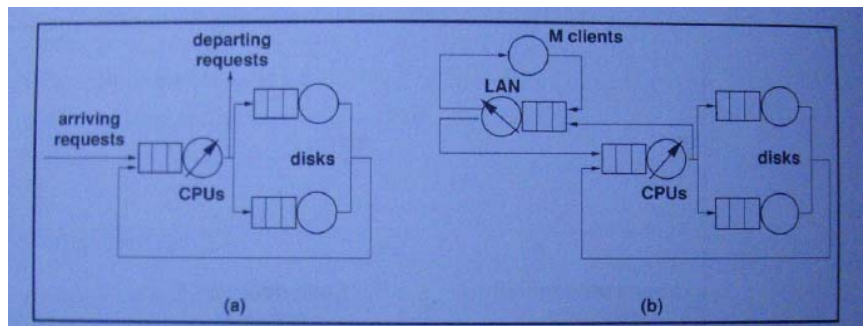
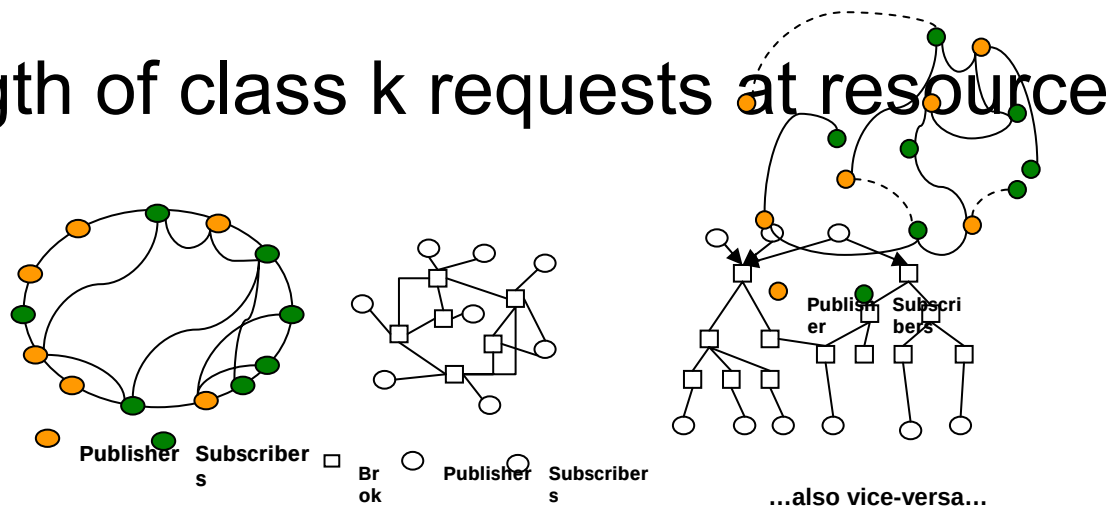



Figure 9.2. (a) Example of an open QN. (b) Example of a closed QN.

- Input parameters: $\lambda^k, K^k, V_i^k, S_i^k$

# Node model: Performance metrics

- U: node (CPU/Disk/Memory/NIC) utilization,
- $R_{i,k}$: Response time of class k requests at resource i
- $W_{i,k}$: Queuing time of class k requests at resource i
- $N_{i,k}$: queue length of class k requests at resource i



# Conclusions on node modeling

- Component level modeling is unfeasible
  - Due to the large number of components,
  - Due to the high number of input parameters

# Overlay Network model

- Graph based models
  - Global properties
- Link based models
  - Local properties

# ON: graph-based model

- The topology is a function of the goal of the overlay
  - Resilient overlay need to find a topology that is redundant respect to the underlining network
  - Ad hoc and sensors networks acquire their topology from the environment in witch the nodes operate

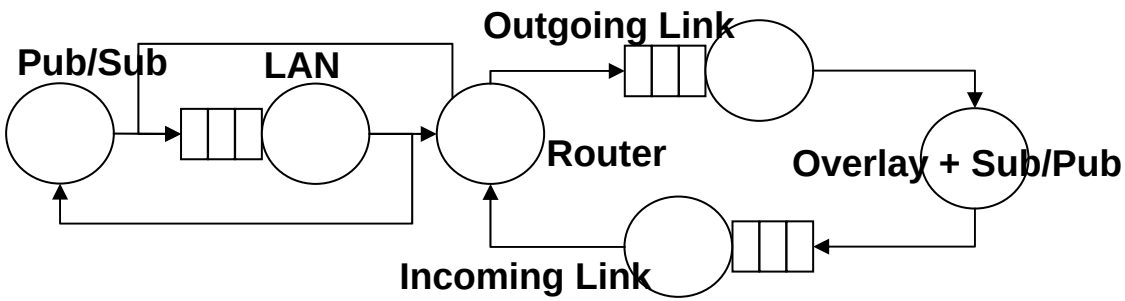
- Graph based models [F03]
  - Random graph (Erdos and Renyi) and (Watts and Strogatz)
  - Small-world networks (Barabasi and Albert)

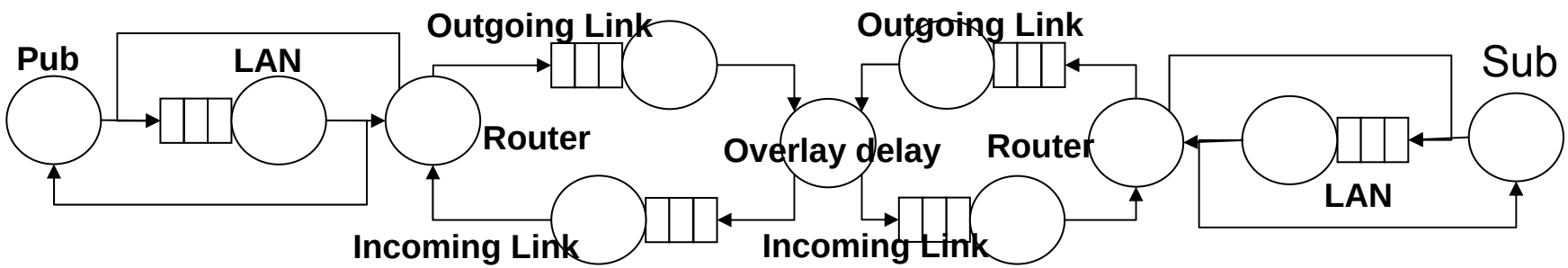
# ON: graph-based model

- Extraction of network properties such as:
  - *Curvature*, related to the reachable part of a network.
    - allow to study network topology properties on larger scale
    - E.g. why the node reachable in a given TTL do not increase when the TTL increase?
  - *Dimension*, related to the size of the network and then to the distance among nodes
    - allow to study network topology properties on small scales
    - E.g. what's the set of nodes reachable in H hops?
- Performance metrics used are: hops count, transmission delay, communication cost, etc…

# ON: link-based model

- End to end model of an overlay link
  - Modeling internet It's in general an open problem
  - E.g.: LAN-R-OL-NetDelay-IL-R-L
    - Parameters: LAN parameters, router parameters, O-IL parameters, InternetDelayRTT and InternetDataRate [Mena02]



# ON: link-based model - performance metrics

- Overlay network links (also wireless in case of unstructured overlay)
  - Throughput,
  - Delay
  - Packet losses

# Conclusion on Overlay Network modeling

- Macroscopic model acceptable
  - Graph based modeling
- Microscopic model unfeasible
  - Network link model

# Overlay Network Routing (ONR) protocols: How to study

- The more used approach in literature is simulation
  - Implementation of the routing protocol
  - Execution of the algorithm on top of a network model
    - Considering network delay
    - Ignoring network delay
- Also prototyping is used:
  - Implementation of the routing protocol
  - Execution of the algorithm on top of a network of nodes
    - Emulated WAN (e.g. WANEmulator, Ohio Network Emulator, …)
    - Real WAN (e.g. PlanetLab)
- Formal methods (model checking, PN/CPN, FSM, …) to prove correctness

# ONR protocols: Simulation

- Considering Network delay only (but not node processing delay):
  - Hierarchical structure: links with different delay
  - Flat structure: all links are equals
- Ignoring Network and nodes processing delay
- No examples of simulators that consider node delays
  - Considered only in prototypes evaluation

# ONR protocols: Simulation

- Considering Network delay only (not node processing delay):
  - Distributed event driven simulation
  - Hierarchical structure: links with different delay[BCMN99]
  - Transit-stub Model[ZCB96]
  - Examples
    - [PCM03][Bayeux01][Scribe02] (OMNET++, NS2,…)

[BCMN99]

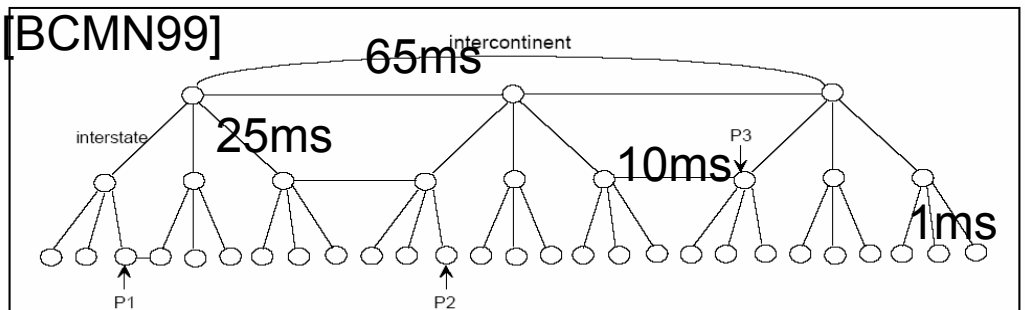



Figure 5: Simulated broker network topology

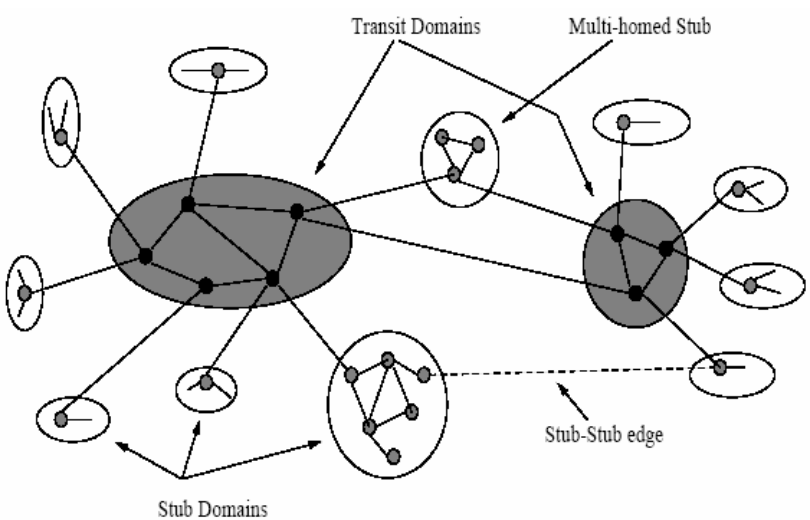



Figure 5: Example of Internet domain structure

[ZCB96], Transit-stub Model Topology

# ONR protocols: Simulation

- Ignoring Network and nodes processing delay:
  - Chord simulator [CHORD01] and p2psim[PASTRY01] simulators:
    - discrete time event driven simulator
    - Publish and subscribe arrival rate
    - Network nodes,
      - algorithm execution,
      - information insertion (notification)
      - Information search (subscription)

# ONR protocols: Prototyping

- Implementation of the routing protocol
- Execution on a controlled environment
  - WAN emulator: the network is emulated on a LAN (WANEmulator) or on a single node (Ohio WAN Emulator)
    - Tapestry [ZHSR04]
    - Pastry [RD01]
- Execution on a real environment
  - PlanetLab
    - Tapestry [ZHSR04]

# ONR protocols: Formal and computational models

- To evaluate the protocol correctness
- To evaluate the protocol complexity
- E.g.
  - Petri nets, FSM, predicate-based languages, model checking…

# ONR protocols: Performance metrics

- Depends on the final goal
  - Load balancing
    - N/K in dht-based overlay (N=num of nodes, K=number of keys)
    - B/M in broker-based overlay (B=num of brokers, M=num of pub/sub)
  - Reducing Path length
    - Number of hop
    - Locality
  - Maintain routing table consistency respect to simultaneous node failure
    - Failed lookup vs node failures
  - Routing table stabilization latency/overhead (after single or simultaneous join)
- Others metrics
  - Node processing overhead
  - Overlay routing overhead respect IP routing (in hops)
  - Routing table occupancy

# Event Routing (ER) Algorithms

- Different routing algorithm
  - Flooding
  - Selective
  - Gossiping
- Different behavior
- Different data structures

# Event Routing: What to model (I)

- Algorithm functionality (correctness):
  - does the protocol deliver messages to nodes that are interested in them?
- Traffic filtering:
  - does the protocol prevent unnecessary traffic?
  - How many unwanted messages a node receive?
  - How many wanted messages a node don't receive?
- Protocol scalability:
  - does the protocol produce a reasonable and stable amount of control traffic?
  - Does the protocol scale respect the number of participants
  - Does the protocol scale respect the number of events
  - Does the protocol scale respect the routing table size?

# Event Routing: What to model (II)

- Reconfiguration capability
  - Dose exist a reconfiguration mechanism respect to the churn? (changes on overlay topology)
  - How many events are lost or misrouted during a reconfiguration?
  - What's the cost of the reconfiguration?
- Algorithms complexity
  - How much memory space is consumed by the routing tables?
  - What's the filter forwarding overhead?
- Correlation with underlining network and protocols
  - How could be better used the underlining (overlay/internet) layer capability to increase the ER performance

# Event Routing: How to

- The more used approach are:
  - Simulation
    - considering the network delays [CRW01]
    - ignoring the network delays
  - Prototyping
    - WAN emulation
    - Real WAN [CRW01]

# ER: Simulation considering/ignoring network delays

- The implementation of the routing algorithm runs on top of nodes of an overlay
  - The underlining overlay routing protocol is not modeled
  - The use of a network simulator allow to consider network delays

# ER: performance metrics

- Algorithm functionality, expressiveness and filtering:
  - False negative, false positive, duplicated events and received advertising [CRW04]
- Protocol scalability:
  - Respect the network consumption [TAJ03] [BMVV05]
    - total cost, cost per service request, forwarding overhead, misrouted events
  - Respect the memory consumption [MFGB02]
    - routing table size
- Reconfiguration capability respect the overlay topology changes [PCM03]
  - Percentage of events delivered, percentage of events losts
  - Number of events misrouted, number of extra sub/unsub messages

# Events Matching Algorithms (EMA)

- Predicate index algorithms
  - Preprocessing phase: decomposition into elementary constraints
  - Matching phase: filters determination
- Testing network algorithms
  - Preprocessing phase: organization of filters constraints into data structure (tree or binary decision diagram)
  - Matching phase: exploration of the data structure to determine the matching

# EMA: what to study

- Algorithm behavior
  - Expressiveness
  - Correctness
  - Distributed solutions
- Complexity
  - Algorithm complexity
  - Theoretical hardness of the problem (proof)
  - Data structure complexity and space complexity
  - Distributed solutions
- Overhead on system nodes
  - Distributed solutions
  - Proof of non trivial performances

# EMA: How to study

- Formal methods
  - Expressiveness and correctness
  - Complexity
- Implementation & testing
  - Complexity
  - Overhead
- Simulation
  - Complexity
  - Distributed solutions

# EMA: Open issues

- Establish theoretical hardness of the problem
- Define algorithms that obtain non-trivial performance guarantee (must be proven)
- How to compare alternatives (performance metrics)
- How to study distributed solutions

# EMA: Formal methods

- Model Checking
- Computational complexity model
  - Turing Machine, Von Neumann Machine, etc...
  - Asymptotic notation

# EMA: Implementation

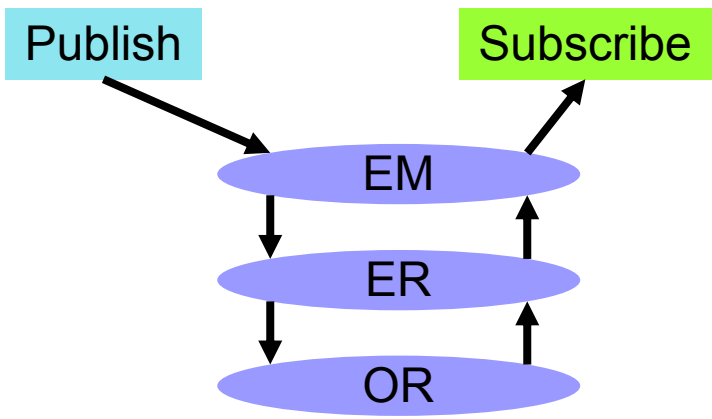
- Typically algorithm were implemented and executed on a machine
  - Stressed by a synthetic or analytical workload
  - Matching time measured as a wall time
  - Space complexity measured as memory consumption

# EMA: Simulation

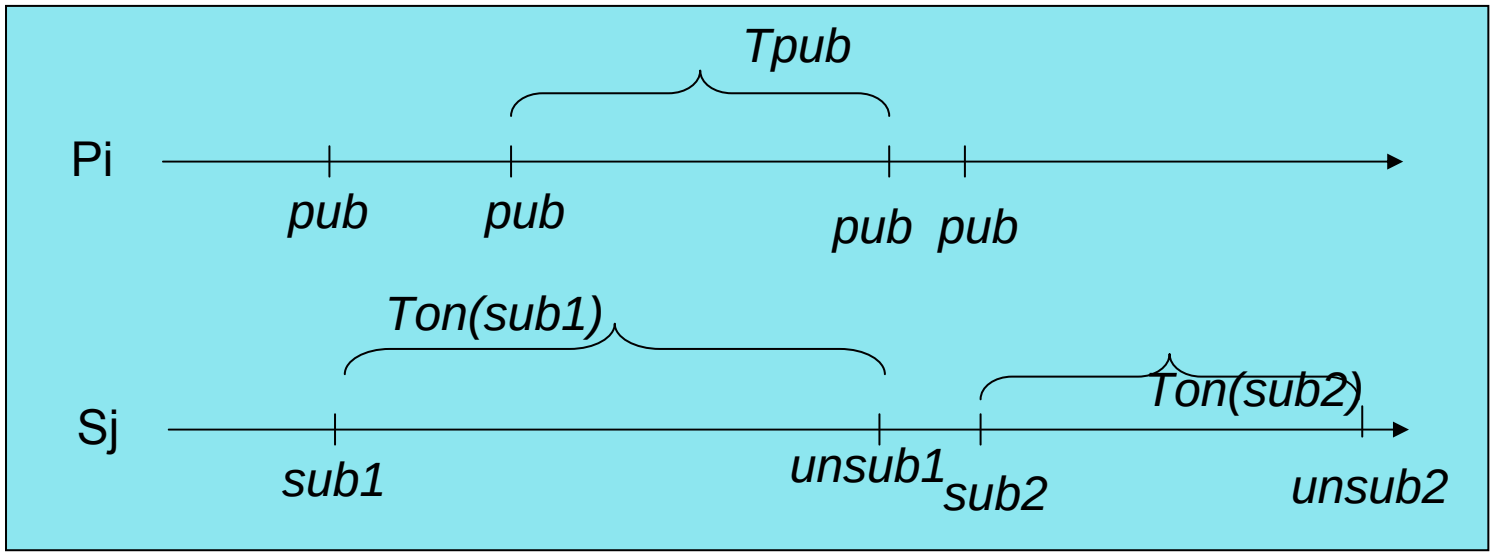
- Used to study impact on nodes and network
  - E.g. to study distributed solutions
- Algorithms are implemented and run on nodes of a network simulator

# Workload Characterization

- Request model
  - Publish/subscribe requests
  - Event Matching requests
  - Event Routing requests
  - Overlay Network routing requests

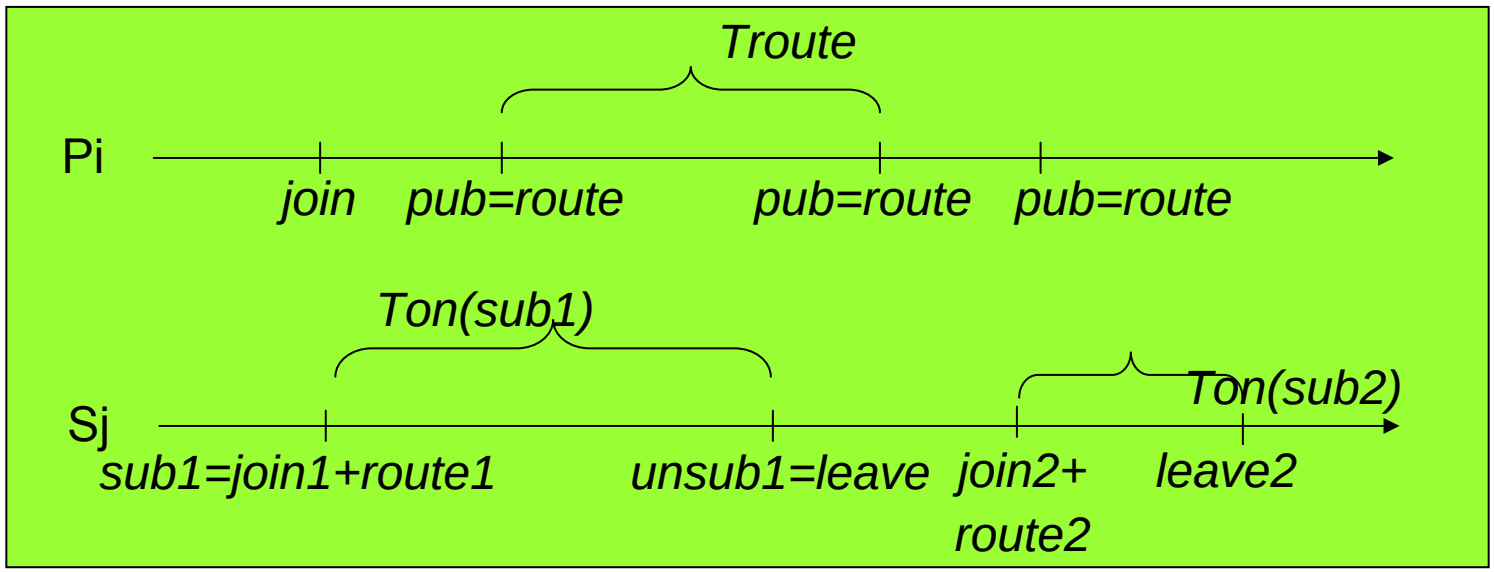


# Workload Characterization: pub/sub requests




- Each pub/sub action loads the Event Matching algorithm
  - Sub: data structure update
  - Pub: algorithm execution

# Workload Characterization: Event/Overlay routing requests



# Overlay Infrastructure Workload Characterization

- Request inter-arrival time
  - Routing requests
  - join/leave requests
  - Overlay management requests
- size of requests
- Node failure model

# Event routing Workload Characterization

- Event space model
- Subscription scheme
  - Type
  - Content
- Request inter-arrival time
  - Subscribe/unsubscribe
  - Publish

# Event Matching Workload Characterization

- Event space model
- Predicate space
- Subscription space

# Bibliography


- [TAJ03] D. Tam, R. Azimi, and H.-A. Jacobsen. Building Content-Based Publish/Subscribe systems with distributed hash tables. In Proc. of the 1st Intl. Workshop on Databases, Information Systems, and P2P Computing (DBISP2P), Berlin, Germany, Sept. 2003.
- [BCMN99] . Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R.E. Strom, and D.C. Sturman, "An Efficient Multicast Protocol for ContentBased Publish-Subscribe Systems," Proc. Int'l Conf. Distributed Computing System, 1999.
- [ZCB96] Ellen W. Zegura, Kenneth L. Calvert, Samrat Bhattacharjee: How to Model an Internetwork. IEEE INFOCOM 1996 San Francisco, CA, USA

# Thank you!

http://www.ce.uniroma2.it/~casali

casalicchio@ing.uniroma2.it