Communication Paradigms

IN5020 – Distributed Systems

lecturer: Roman Vitenberg

UiO : University of Oslo

What is a communication paradigm?

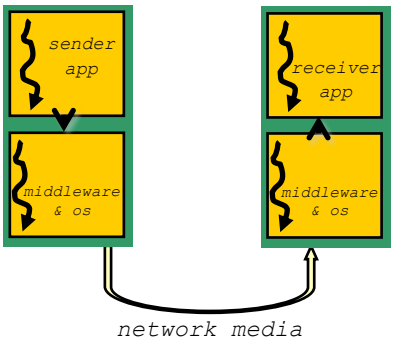
- It is a set of communication primitives
  - Network adapters allow us to send data using MAC-layer primitives
  - Transport layer endows us with sockets
  - How could we raise the abstraction further?
- Study of a communication paradigm:
  - Properties
  - Target applications
  - Underlying implementation concepts

Communication properties

- Addressing scheme and space decoupling
  - Underlying protocol addresses (IP) – no decoupling
  - Logical aliases – partial decoupling
    - DNS and NAT translation, service names, email aliases
  - Content-based addressing – full decoupling
    - Interactions are declarative
- Persistence level
  - Fully persistent
  - Fully transient
  - Intermediate

Communication properties

- Synchrony
  - Fully synchronous
  - Fully asynchronous
  - Intermediate
    - middleware-level sync
    - man-in-the-middle
    - others
- Time decoupling



(some) Communication paradigms

- Remote procedure call
  - Object-based (CORBA, Java RMI, DCOM)
  - Earlier data-based (DCE, Sun RPC)
- Message-oriented communication
- Stream-oriented communication
- Software-based distributed shared memory (DSM)

(some) Message-oriented communication paradigms

- Raw socket programming
- Message-passing interface (MPI)
- Message-oriented middleware (MOM)
- Publish-subscribe communication

Raw socket programming

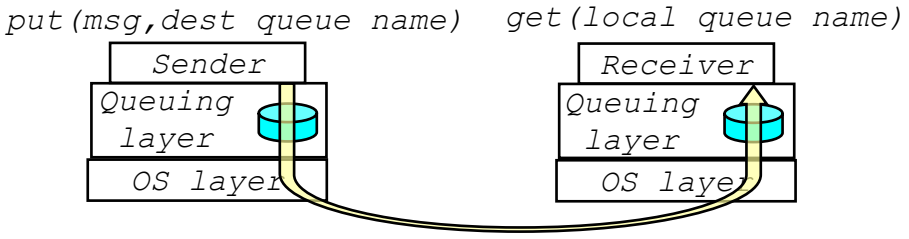
- Addressing scheme: IP addresses
- No time decoupling
- Transient
- Mainly used for building higher-level abstractions

Message-programming interface (MPI)

- Addressing scheme
  - A group of nodes assigned logical addresses
- Not designed to cope with failures
- Transient without time decoupling
- Data-oriented (advanced data manipulation)
  - Basic API: MPI_send, MPI_recv
  - Data-oriented API: MPI_scatter, MPI_gather
- Use: parallel computation in fast networks

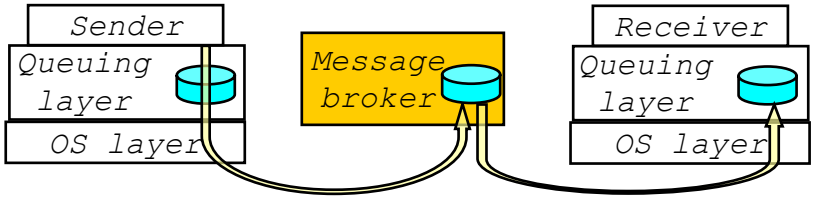
Message-oriented middleware (MOM)

- Addressing scheme: logical queue name
- Persistent
- Full time decoupling



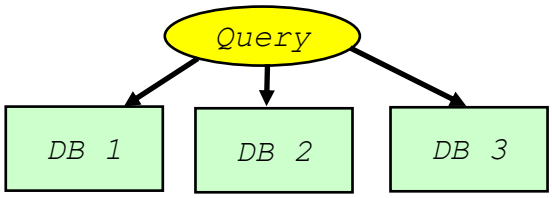
Routing in MOM

- Handles queue name to address translation
  - Hierarchical names: {queue manager, internal id}
- Message brokers perform inter-domain routing with format conversion

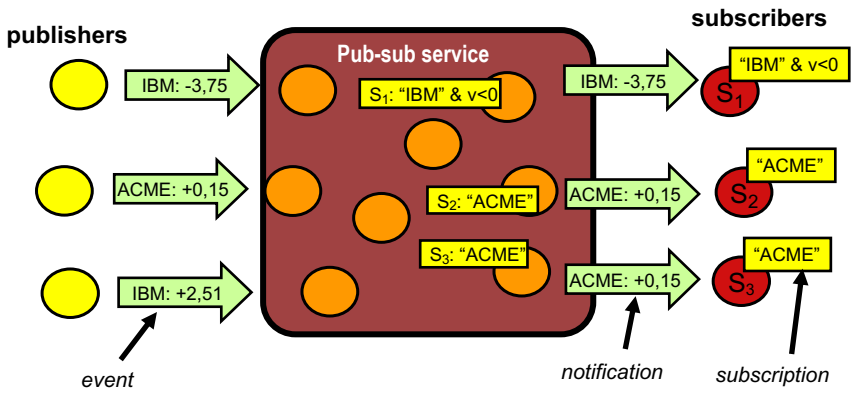


MOM applications & implementations

- Implementations: VMware RabbitMQ, IBM MQ, Oracle AQ
- The E-mail application
- Workflow and other collaborative apps
- Federated information systems



Publish-subscribe communication



- Publishers: objects of interest or observers

Pub-sub properties

- Addressing scheme: through content
- May be persistent or transient
- If persistent, may provide time decoupling
- The pub/sub matching service is frequently implemented by a number of dedicated servers
  - Called message brokers
  - Might be within a datacenter of across datacenters
  - A subscriber connects to one of the brokers
  - This broker becomes responsible for matching and delivery of notifications to that subscriber

Pub-sub applications

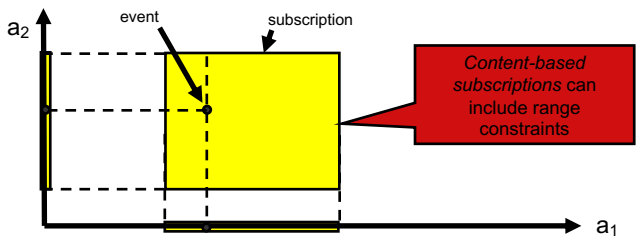
- Event-based business processes
- News distribution
  - The research-originated Gryphon system was part of the Web infrastructure serving the Olympic games in 2000
  - More recently: RSS and RSS aggregators
- Delivery of financial data
  - Many stock exchanges around the world
- Intrusion detection and other applications of distributed data mining
- Online games
- Social notifications from Spotify
- Many others …

Subscription semantics

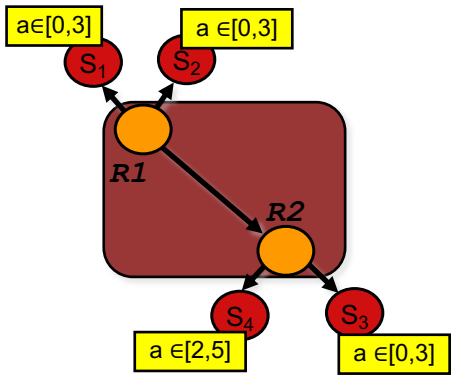
- Topic-based pub-sub:
  - *publish(topic t), subscribe(topic t)*
  - The topic namespace may be hierarchical
  - Wildcards: *subscribe("nasdaq.stockvalue.a*")*
- Type-based pub-sub
  - Generalization of topic hierarchy
  - Uses the fact that events of the same type have the same structure (fields)

Subscription semantics

- Content-based pub-sub
  - Universally known list of event attributes
  - Event represented as a set of attribute values
    - A point in the multi-dimensional event space
  - Subscription is a cuboid in the event space



Content-based routing



The routing table of R1

Interface	Filter
To node 1	a ∈[0,3]
To node 2	a ∈[0,3]
Toward R2	a ∈[0,5]

Communication paradigms (summary)

Abstraction	*Space decoupling*	*Time decoupling*	*Persistence*
Raw sockets	no/partial	no	no
RPC	no/partial	no	no
MOM	partial	yes	yes
Pub-sub	full	possible	possible

Multicast and its effect on communication abstraction

- Can appear as an element in many paradigms or be considered as a paradigm by itself
- Makes complicated communication abstractions even more complicated
  - Addressing scheme becomes even more important
    - Stronger case for space decoupling
  - Reliability issues become more involved
  - Message orderings
  - Atomicity

Not transparent for apps, Affects the paradigm

The challenges of supporting multicast communication

- No standardized transport protocols to rely upon
  - What about IP-multicast?
    - Not always available
    - Historical trend: shift of the solutions from the network to application level
- Different approaches
  - Emulate multicast by unicast
  - Overlay-based multicast
  - Epidemic or gossip-based dissemination
  - Result in different paradigms

Overlay-based multicast

- Organize the destination nodes in a logical application-level network graph (*overlay*)
- Disseminate messages using overlay links
- Monitor links and nodes: failures, link quality, communication load
- Incrementally reconstruct upon joins, leaves, overload, link and node failures

Overlay-based multicast (the underlying principles)

- It is possible to achieve both good scalability and low latency at the same time
  - Logarithmic or better fan-out for scalability
  - Short routing paths (logarithmic # of hops)
- The small-world phenomenon
  - Overlay topology induced by the physical one
    - (e.g., a rectangular grid of sensors)
  - Adding a single link from each node to a random destination node is enough to create short routing paths

Multicast overlay types

- Multicast tree
  - The most efficient dissemination
  - Simple routing scheme (flooding)
  - The load is distributed non-evenly
  - Highly vulnerable to failures
- Other overlays (regular hypercube, regular random graph, rectangular grid)
  - Better load distribution & resilience to failures
  - More complicated routing scheme

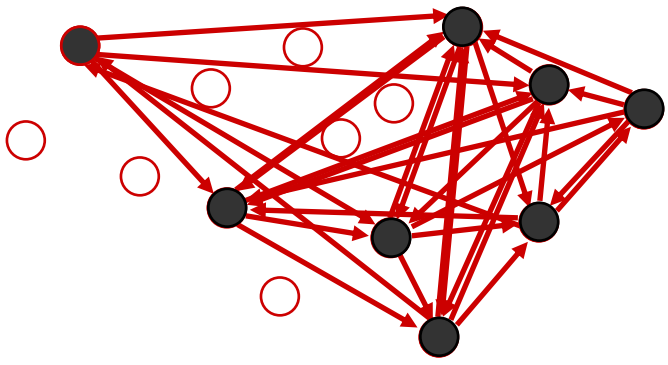
Epidemic dissemination

- Observe how fast epidemics propagate in the absence of treatment
- Use the same principles for the positive purpose of message dissemination
- Infected, susceptible, and removed nodes
- Based on membership: every node maintains a (possibly partial) membership of other nodes it can communicate with

Epidemic Dissemination (Push)

- The protocol is parameterized by *infection period t* and *fan-out f*:
  - When a node becomes infected, it executes *t* rounds and then becomes removed
  - At each round, it sends the message to *f* random nodes from its membership list
- Global round *k*: every node has executed at least *k* rounds and at least one node has executed exactly *k* rounds

Push Epidemic Dissemination Example *(t=2, f=2)*



Epidemic Dissemination (Pull)

- Each susceptible node executes an unlimited number of rounds until it becomes infected
- At each round, it contacts *f* random nodes from its membership list, checks if one of them is infected, and pulls the message
- After *t* rounds, it becomes removed
- Can be combined with push dissemination to form a push-pull approach

Epidemic dissemination (properties)

- Fault-tolerance: no need to detect message losses due to link and node failures, no message retransmissions
- Probabilistic atomicity (bimodal behavior): depending on *t* and *f*, the message is likely to be delivered
  - either to almost all nodes
  - or to a negligible portion of nodes
- The propagation is reasonably fast: if it reaches almost all nodes, it does so in *O(log N)* global rounds on average

Push vs pull gossiping

- Push approach:
  - Fast & efficient when few nodes are infected
  - When just a few nodes are susceptible
    - Takes a long time to reach susceptible nodes
    - A lot of unnecessary messages are sent
- Pull approach:
  - Fast & efficient when most nodes are infected
  - Wasteful and slow if few nodes are infected

Push vs pull gossiping

- Push-pull approach:
  - Fast propagation to all nodes
  - Wasteful whatever portion of nodes is infected
- Rumor spreading:
  - Push-based
  - Non-constant # of rounds: whenever a node pushes to an already infected node, it becomes removed with probability *p*
  - Communication-efficient but slower dissemination

Membership properties

- Membership list of size *L*
  - Infeasibility of full membership in large-scale systems
  - Fundamental tradeoff: smaller membership list scales better but may limit dissemination
    - Risk of partitioning the set of nodes
- Uniformity: partial lists are uniform samples
- Adaptivity: ideally, *L* should be adapted to *N*
  - Nodes may have difficulty of estimating *N*
- Bootstrapping: membership initialization

Applications of gossiping

- Characterization
  - Scalable and low maintenance
  - Fast but not the fastest
  - Robust but no hard guarantees
- Failure detection
- Data aggregation
- Resource discovery and monitoring
  - Access to replicated web pages
- Update propagation for data caching
- Experimental: content search, file sharing

Comparison: overlay- vs gossip-based multicast

- Overlay-based multicast
  - Efficient propagation
  - 100% delivery guarantee in the absence of churn
  - Costly and complex reconfiguration upon churn
- Gossip-based multicast
  - Many unnecessary messages may be sent
  - May not reach 100% of nodes even in a completely stable environment
  - Very resilient to all kind of churn

Reading material

- TvS Sections 4.1.2, 4.3, 4.5, 13.4.1
- Coulouris et al. Sections 6.1, 6.3 and 6.4
- For PhD students only
  - "The Many Faces of Publish/Subscribe" by Eugster, Felber, Guerraoui, Kermarrec
    - Can be found in the teaching plan on the web
  - "Epidemic Information Dissemination in Distributed Systems" by Eugster, Guerraoui, Kermarrec, Massoulie