CS514: Intermediate Course in Computer Systems

Lecture 4: Sept. 10, 2003

"Introduction to Message Oriented Middleware (MOM)"

Overview of Lecture

CS514

Introduction to Message Oriented Middleware (MOM)

- MOM versus OOM (Object Oriented Middleware)
- Goals of MOM
- Categories of MOM
  - Explicitly addressed versus publish/subscribe
  - One-to-many versus many-to-many
  - Guarantees
- Examples of MOMs
  - Email and SIP
  - Newsgroups
  - Message Bus (TIBCO)
  - Content routed (Gryphon)

MOM versus OOM

CS514

- Message-oriented versus Object-oriented
- Seems to be a popular distinction to make
- Perhaps a strict definition:
  - OOM manufactures and passes around objects with method invocations, etc…
    - Corba, DCE, Java RMI, Microsoft DCOM
  - MOM passes around un-typed messages
    - IBM MQSeries, Lotus Notes, Sun JMS, TIBCO, even email and net news!
- But this definition probably doesn't get at the important distinction

MOM versus OOM

CS514

Probably more important….

- MOMs are historically asynchronous, whereas OOMs are historically synchronous
  - Perhaps because OOMs evolved from OO languages
    - In the same way that RPC evolved from procedural languages
- Related to this, MOMs accommodates one-way message passing in addition to query/reply
  - i.e. a broader range of applications

Synch versus asynch

CS514

- It is easier to make asynchronous perform like synchronous than vice versa
- Synch implies blocking, expectation of a reply
  - Which in turn implies a certain style of programming, one that requires an answer before progress can be made
  - Going asynch requires rethinking the whole flow of logic
- Asynch (non-blocking) is a more general programming style
  - If you make the reply fast, you can always choose to block…

MOM vs. OOM distinction is perhaps silly

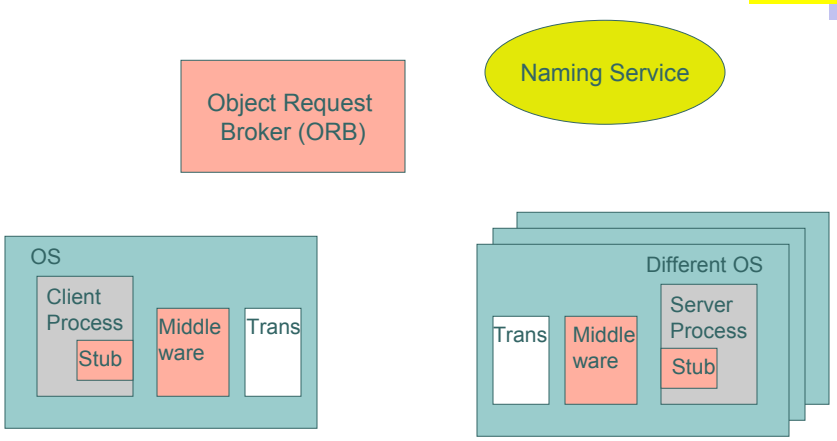
CS514

- Ultimately, OOM is a way to distribute an OO program, whereas MOM is a communications abstraction
  - Though both camps are trying to encompass the other
- The only thing the two terms have in common is “middleware”
- But what is “middleware”?
  - A very vague and ill-defined term

(We are only presenting these terms because industry throws them around)

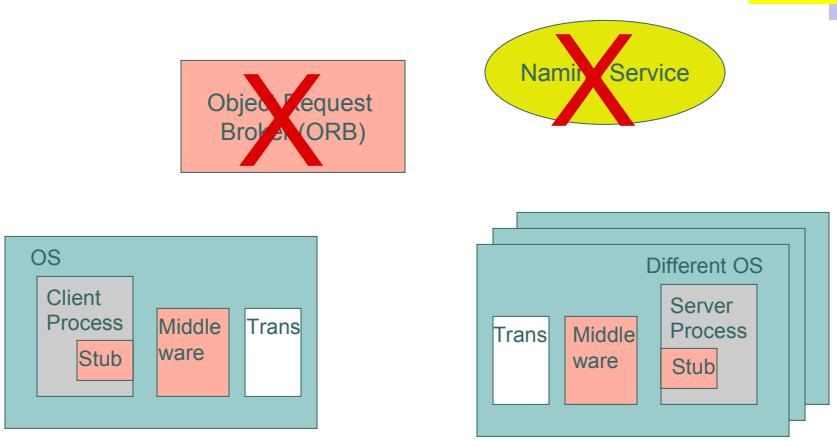
Remember this picture? (Full-featured RPC)

CS514



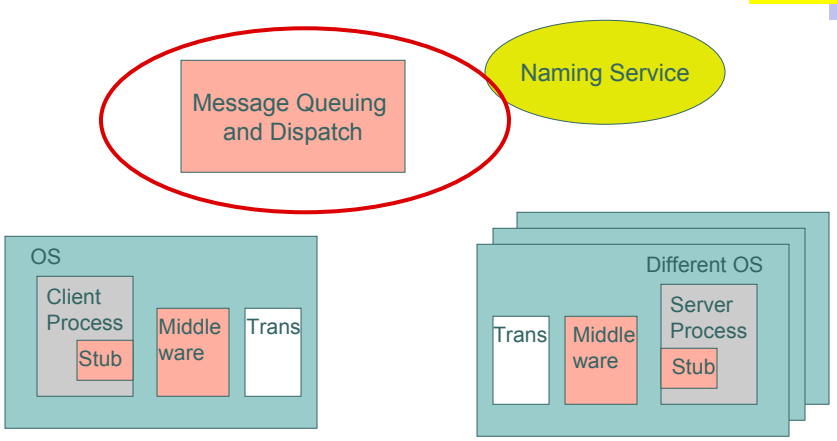
This stuff is ultimately optional

CS514



But not in a MOM system…

CS514



Various goals of MOMs

CS514

- Of course integration of different system types
  - The one thing all “middleware” has in common!
- Delivery (persistence) and ordering guarantees
  - Eventually message will arrive in the right order
  - Prioritization
  - Causal ordering (i.e. knowing that A’s message 142 came after B’s message 217)
- Flexible addressing
  - “Function”-based as well as ID-based
  - Aids in system evolution
  - Point-to-point and one-to-many and many-to-many (event) communications models
- Increased system throughput

Example MOM Applications

CS514

- Person-to-person messaging (email)
- Groupware applications
  - Planning, document sharing and editing, scheduling
- Database access
- Event notification (publish/subscribe)
- Workflows
- Many others

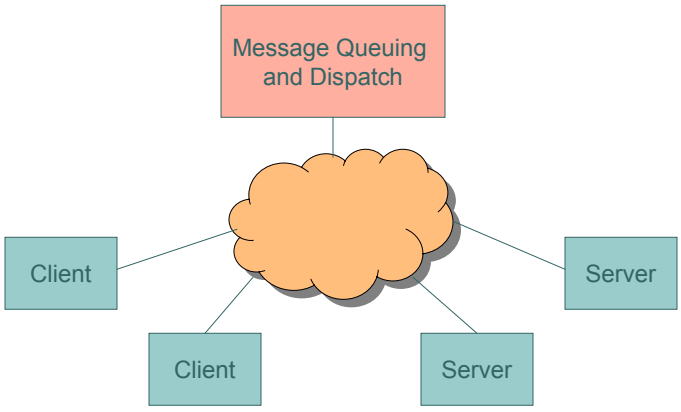
Various MOMs

CS514

- Hub-and-Spoke
- Email
- Usenet (NNTP News Groups)
- SIP (Session Initiation Protocol)
- Message Bus (TIBCO)
- Content-based subscription (Gryphon)

Simple Hub-and-Spoke Model

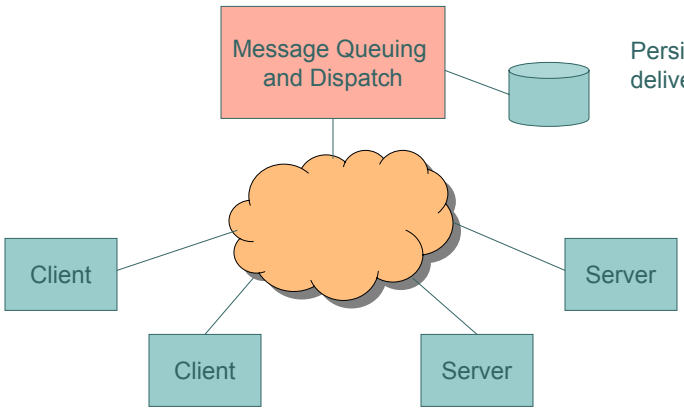
CS514



Some issues (client/server/dispatch configuration, user auth, user identification, flow control, message order), but obviously operation is fairly simple

Simple Hub-and-Spoke Model

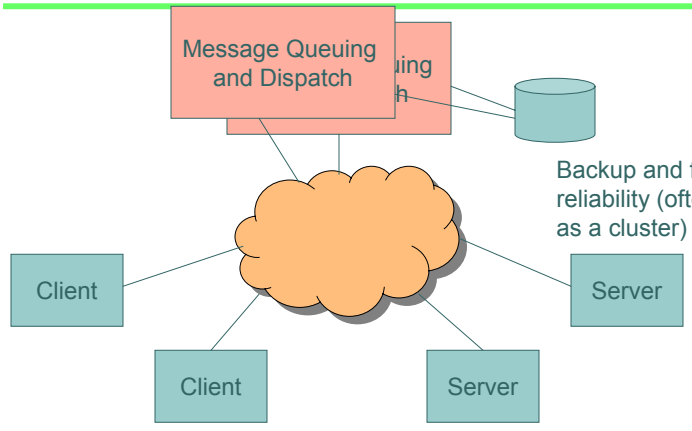
CS514



Persistent storage for delivery guarantees

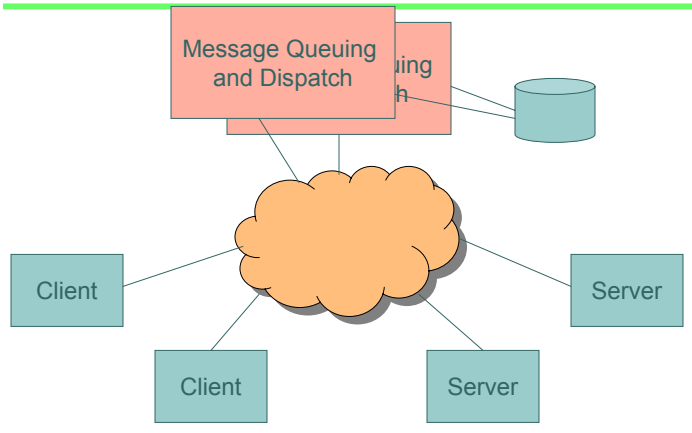
Simple Hub-and-Spoke Model

CS514



Obvious scaling limitations

CS514



Though you can go pretty far with this model (hundreds of “spokes”)

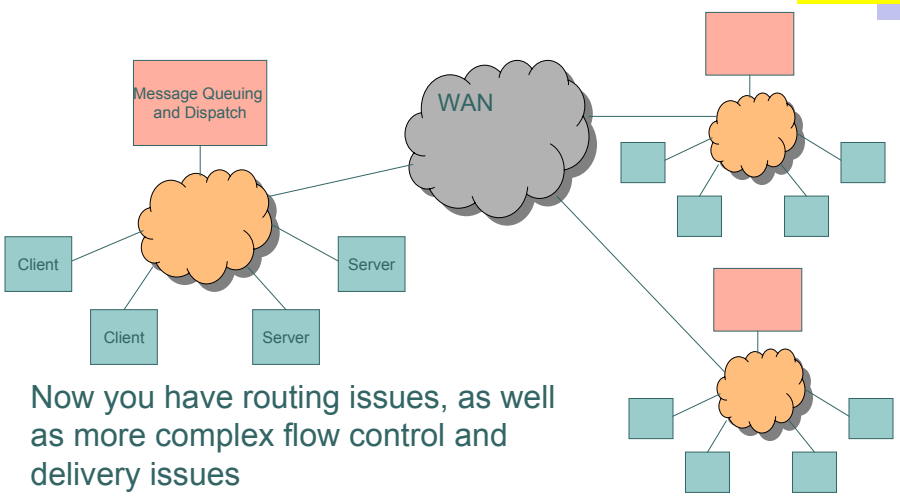
Lots and lots of products

CS514

- JMS (Java Messaging Service)
  - Sun ONE Middleware server
- Websphere MQ (MQSeries)
  - Has JMS interface
- MSMQ (Microsoft Messaging Queue)
- BEA Systems
- ObjectWeb (Open Source) JORAM (JMS)
- Fiorano
  - JMS interface
- Sonic Software (Sonic MQ)
- etc….

Network of Hub-and-Spoke

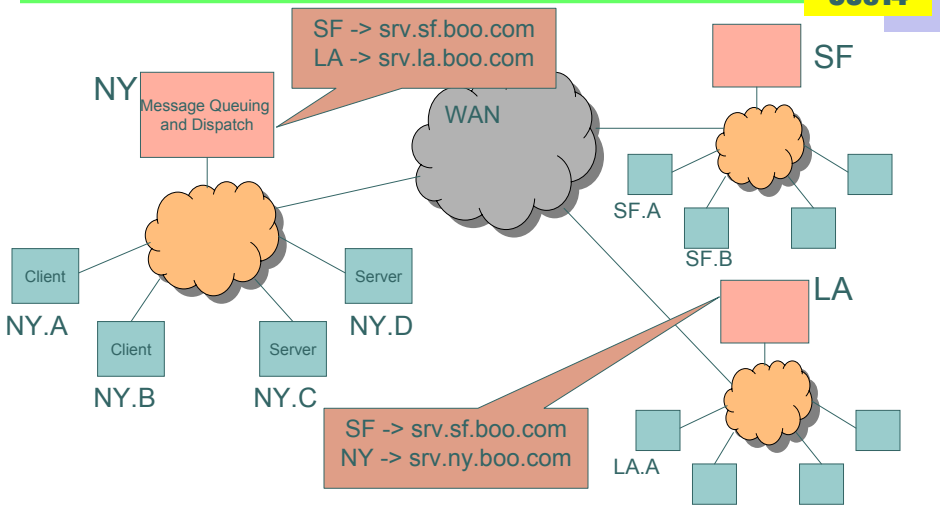
CS514



Now you have routing issues, as well as more complex flow control and delivery issues

Typical routing is hierarchical and static

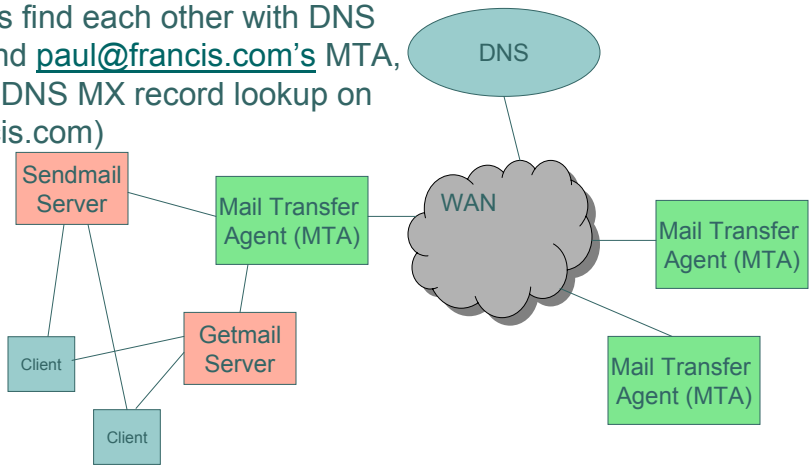
CS514



Email

CS514

MTAs find each other with DNS (to find paul@francis.com's MTA, do a DNS MX record lookup on francis.com)



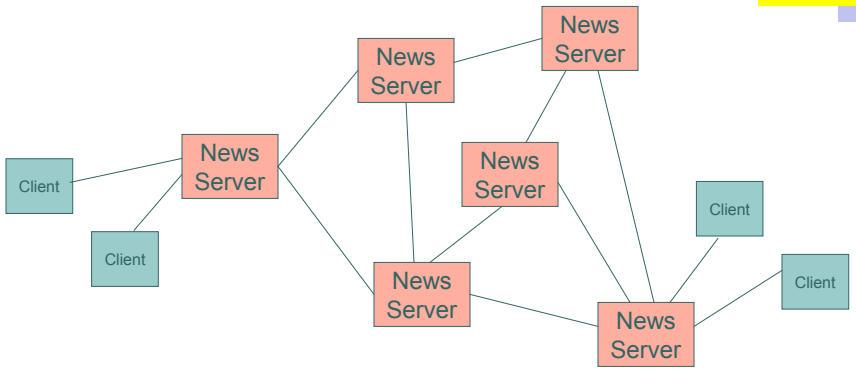
Email

CS514

- List of MTAs prevents routing loops
- No guaranteed delivery
- Limited causal message ordering
- Addressing
  - Point-to-point (explicit, with address list aliasing)
  - One-to-many or many-to-many
    - "Topic" or "Channel" publish/subscribe semantics through various add-on list management tools
    - But delivery mechanism is a list of destination addresses (possibly with local exploders)

Usenet (News Groups, NNTP)

CS514



Messages are flooded to all news servers (with duplicate suppression)

Usenet (News Groups, NNTP)

CS514

- Publish/Subscribe semantics (rec.arts.origami)
  - Many-to-many only
  - Broadcasts all topics/channels
- No delivery guarantees (by a long stretch!)
- Some causal ordering
- Note primary motivation for design was to save disk space on clients!
  - Creation of groups used to be tightly controlled
  - I think Moore's law has bypassed Usenet!

SIP (Session Initiation Protocol)

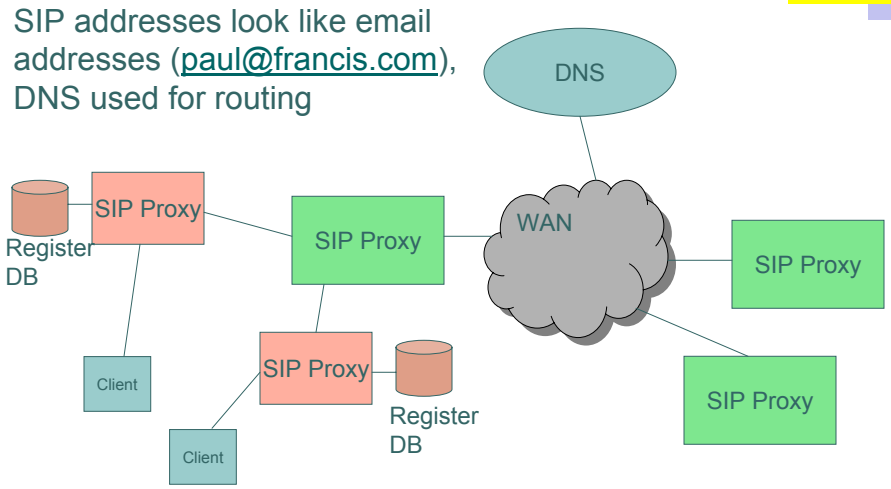
CS514

- Originally a Voice/Video over IP signaling protocol
  - Joe wants to talk to Sue over the Internet, needs to:
    - Signal that fact (ring her phone)
    - Negotiate what voice coding format to use, etc.
- Expanded to include presence and messaging (called "SIMPLE")
  - Microsoft is behind this

SIP Architecture

CS514

SIP addresses look like email addresses (paul@francis.com), DNS used for routing



SIP versus email

CS514

Email is async with storage in the middle	SIP is (mainly) sync with a stateless middle (delivery semantics E2E)
Email users contact server when they want to get/send messages.	SIP users register their location(s) to the “proxy”, are continuously reachable by the proxy
Email has MIME encoding	SIP has MIME encoding
Email has user@dns-domain addresses	SIP has user@dns-domain addresses

I think SIP is important

CS514

- It hasn't (quite) reached critical mass
- It hasn't seeped into the consciousness of the middleware community
- But it is powerful in fundamental ways (like email and http are)
  - HTTP: Client contact any (up) server instantly
  - Email: Peer contacts any peer with delay
  - SIP: Peer contacts any (up) peer instantly

Message Bus (TIBCO)

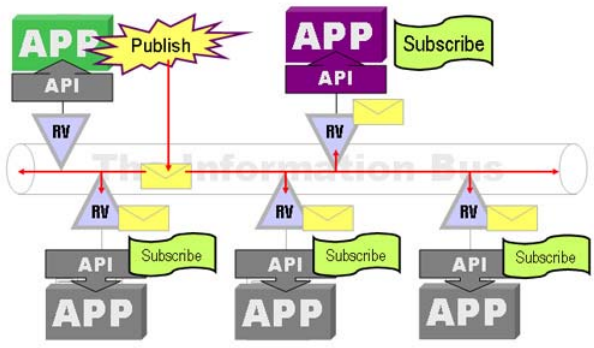
CS514

- Most famous for publish/subscribe event service
  - But also has point-to-point
  - Usenet is publish/subscribe, but not event
  - Topic based subscriptions, but somewhat more general than Usenet
    - Can do wild-carding at each name level
    - People.*.Schwarzenegger matches both people.actors.Schwarzenegger and people.politicians.Schwarzenegger

Images borrowed from Chris Ré's CS614 talk

TIBCO Architecture

CS514



TIBCO Architecture

CS514

- Makes heavy use of LAN broadcast
- Every node listens to all messages
  - But only passes up those to which it has subscribed
  - Scaling limitation, but ok for many cases
- Runs “reliable” protocol over UDP
  - Sequence number per publisher per topic
  - Periodically broadcast update message with last transmitted sequence number
  - If subscriber hasn’t seen it, requests retransmission
  - But eventually publisher deletes message

Subject-based addressing

CS514

- Key concept is that publishers and subscribers don't need to know about each other explicitly
  - Makes it easy to add and remove boxes and applications
- Publish: destination address is conceptually "everyone interested in this topic"

Subject-based addressing flexibility

CS514

- Use to discover services or specific nodes:
  - Printer p1 in systems lab subscribes to following:
    - printers.syslab.p1
  - To find printer p1, publish:
    - printers.*.p1
  - To find any printer in syslab, publish:
    - printers.syslab.*
  - Either way, p1 will receive message and reply directly (point-to-point)
    - Message contains a "call-back" command

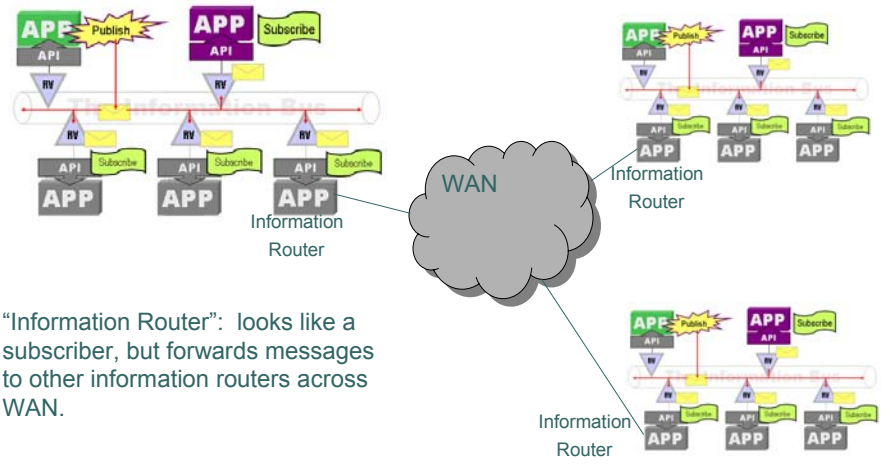
Subject-based Addressing abstraction not perfect

CS514

- For “guaranteed” message delivery, publisher requires subscriber acks
  - Must know all subscribers explicitly
  - Will periodically republish message until all acks received
  - Gives up eventually
    - Network partitions will result in failed delivery, though at least publisher will know it

WAN architecture: required because no WAN multicast

CS514



“Information Router”: looks like a subscriber, but forwards messages to other information routers across WAN.

TIBCO WAN Issues

CS514

- To avoid broadcast of all messages to all sites:
  - Information router must know local subscriptions, tell other information routers
  - No longer “silent subscribe”
    - (perhaps never was?)
- Reliability/guarantees harder
  - Information router crash causes partition
- Ultimately, the “Information Bus” is not as transparent and simple as it appears
  - And later we’ll find that reliable multicast is hard to scale

Content-based subscription

CS514

- Don’t define topics, rather subscribe based on contents of message
  - “all messages with Schwarzenegger in body”
- Or values of predicates
  - Stock ticks with ATT > 30
- Like a relational database turned on its head
  - Match entry against many queries!

Approach

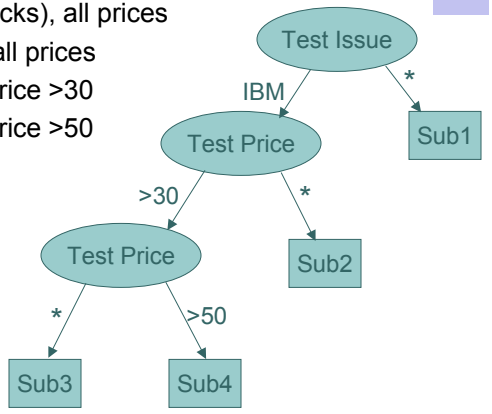
CS514

- Broker collects subscriptions
- Broker receives published messages with attributes and values, and matches them against subscriptions
- Hard part:
  - Doing this scalably
- Basic idea:
  - Build a tree data structure from subscriptions, walk tree with published messages

Simple Example:

CS514

- Sub1: All issues (stocks), all prices
- Sub2: Issue = IBM, all prices
- Sub3: Issue = IBM, price >30
- Sub4: Issue = IBM, price >50



See IBM Gryphon project for more details