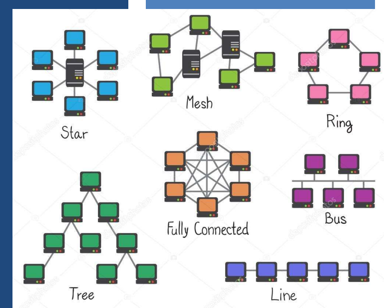
TCSS 558: APPLIED DISTRIBUTED COMPUTING



Distributed Systems: Architectures and Middleware

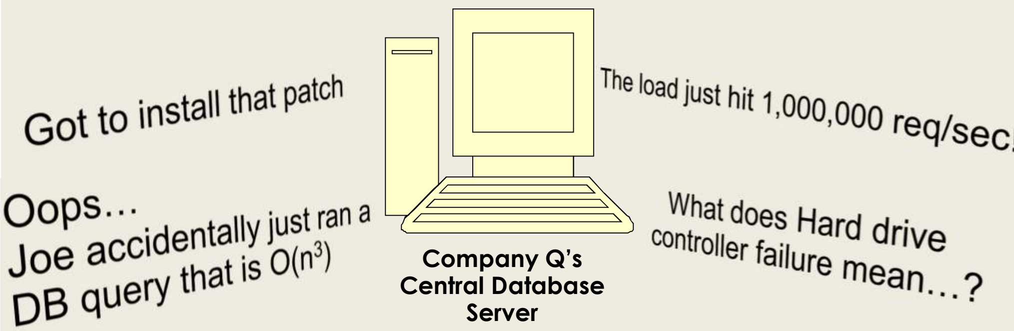
Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma

OBJECTIVES

- Feedback from 10/10
- Ch. 2 - Architectural styles
  - Event-based / publish & subscribe
- Class activity: architectural styles
- Middleware organization
- System architectures
  - Centralized: Single client, multi-tier
  - Decentralized peer-to-peer: structured, unstructured, hierarchical
  - Hybrid

FEEDBACK – 10/10

- What is the difference between a centralized vs. decentralized architectural style?
- Why is a centralized system less available? (fewer 9s)



Availability is measured in percentage uptime (e.g. 99.9%)

October 12, 2017	TCSS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma	L5.3

FEEDBACK - 2

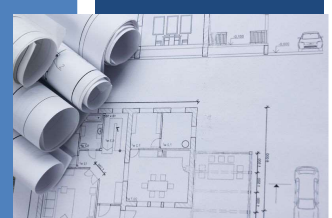
- Still confused about RMI…
  Could you please give a detailed example to show why and how an object should invoke the method of a remote object?
- The use cases for distributed objects will vary
- These are the same reasons we "distribute" the system
- Local CPU resources of a node may be insufficient to complete work in a timely manner → ***outsource the computation***
- Data required to complete the computation may be unavailable at local node → ***move the computation to the data***
  - *It may be too slow or expensive to move the data to the node*
- Local node may be unauthorized to directly access data required for computation → ***delegate to authorized host***

October 12, 2017	TCSS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma	L5.4

FEEDBACK - 3

- For assignment 0:
  After building the tomcat container, and using "docker images –a", my image has a name of <none>.
- Need to include the "-t" flag on docker build
- See "man docker-build" or "docker build --help"
- Can also include a version number:
- docker build -t <name>:<version> <path to Dockerfile>
- **Example:**

```
docker build –t tcss558test:version1 .
```

CH. 2: DISTRIBUTED SYSTEMS ARCHITECTURES

PUBLISH-SUBSCRIBE ARCHITECTURES

- Enables separation between processing and coordination
- Types of coordination:

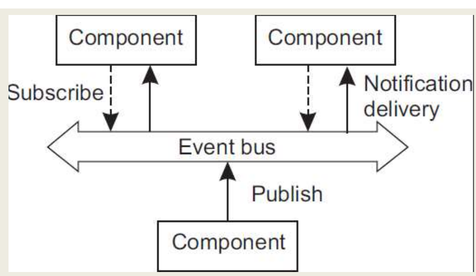
	Temporally coupled (at the same time)	Temporally decoupled (at different times)
Referentially coupled (*dependent on name*)	**Direct** Explicit synchronous service call	**Mailbox** Asynchronous by name (address)
Referentially decoupled (*name not required*)	**Event-based** Event notices published to shared bus, w/o addressing	**Shared data space** Processes write tuples to a shared data space

Not publish and subscribe

October 12, 2017	TCSS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma	L5.10

PUBLISH-SUBSCRIBE ARCHITECTURES - 2

- **Event-based coordination**
- Processes do not know about each other explicitly

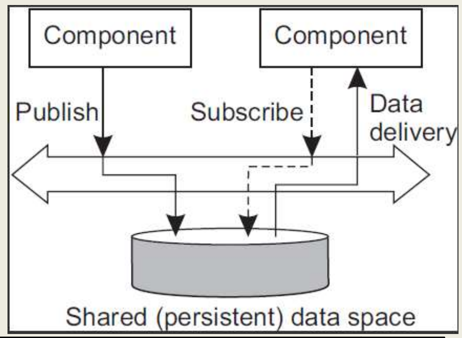


- Processes:
  - **Publish:** a notification describing an event
  - **Subscribe:** to receive notification of specific kinds of events
- Assumes subscriber is presently up (*temporally coupled*)

October 12, 2017	TCSS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma	L5.11

PUBLISH SUBSCRIBE ARCHITECTURES - 3

- **Shared data space**
- Full decoupling (name and time)
- Processes publish "tuples" to shared dataspace (publish)
- Processes provide search pattern to find tuples (subscribe)
- When tuples are added, subscribers are notified of matches
- **Key characteristic:** Processes have no explicit reference to each other



PUBLISH SUBSCRIBE ARCHITECTURES - 4

- Subscriber describes events interested in
- Complex descriptions are intensive to evaluate and fulfil
- **Middleware will:**
- Publish matching notification and data to subscribers
  - Common if middleware lacks storage
- Publish only matching notification
  - Common if middleware provides storage facility
  - Client must explicitly fetch data on their own
- Publish and subscribe systems are generally scalable
- **What would reduce the scalability of a publish-and-subscribe system?**

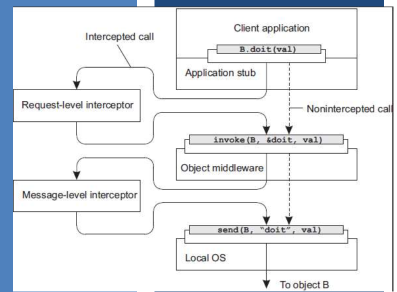
IN-CLASS ACTIVITY: DISTRIBUTED SYSTEMS ARCHITECTURES

L5.14

DISTRIBUTED SYSTEM GOALS TO CONSIDER

- Consider how the architectural change may impact:
- Availability
- Accessibility
- Responsiveness
- Scalability
- Openness
- Distribution transparency
- Supporting resource sharing
- Other factors…

October 10, 2017 | TCSS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma | L4.15



MIDDLEWARE ORGANIZATION

October 12, 2017 | TCSS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma | L10.16

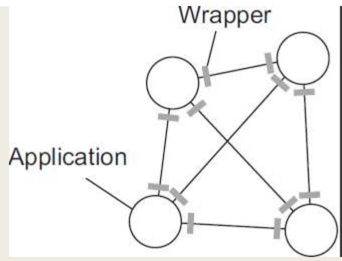
MIDDLEWARE: WRAPPERS

- **Wrappers (adapters)**
  - Special “frontend” components that provide interfaces to client
  - Interface wrappers transform client requests to “implementation” at the component-level
  - Provide modern services interfaces for legacy code/systems
  - Enable meeting all preconditions for legacy code to operate
  - Parameterization of functions, configuration of environment
- Contributes towards system openness
- **Example: Amazon S3**
- Client uses REST interface to GET/PUT/DELETE/POST data
- S3 adapts and hands off REST requests to system for fulfillment

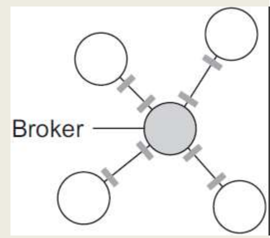
October 12, 2017 | TCSS558: Applied Distributed Computing [Fall 2017]
Institute of Technology, University of Washington - Tacoma | L5.17

MIDDLEWARE: WRAPPERS - 2

- Inter-application communication
  - Application provides unique interface for every application
- Scalability suffers
  - N applications → O(N^2) wrappers



- **Broker**
  - Provide a common intermediary
  - Broker knows how to communicate with every application
  - Applications only know how to communicate with the broker



October 12, 2017	TCSS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma	L5.18

MIDDLEWARE: INTERCEPTORS

- **Interceptor**
- Software construct, breaks flow of control, allows other application code to be executed
- Enables remote procedure calls (RPC), remote method invocation (RMI)
- Object A can call a method belonging to object B on a different machine than A.

October 12, 2017	TCSS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma	L5.19

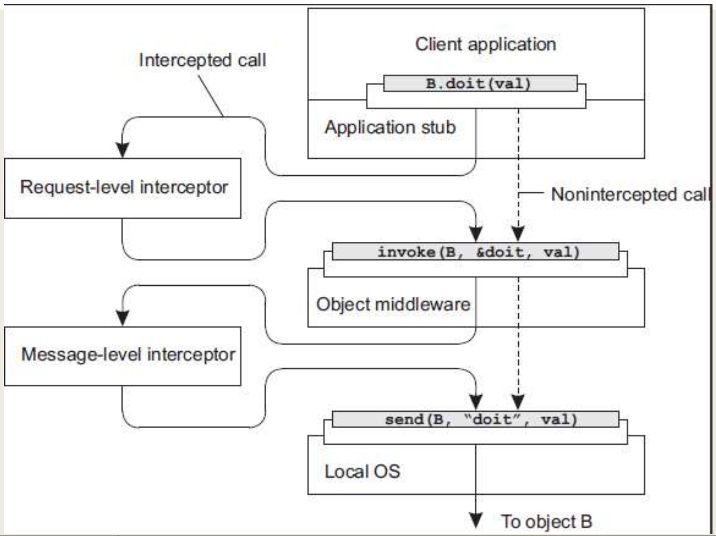
MIDDLEWARE INTERCEPTION - METHOD

- Local interface matching Object B is provided to Object A
- Object A calls method in this interface
- A's call is transformed into a "generic object invocation" by the middleware
- The "generic object invocation" is transformed into a **message** that is sent over Object A's network to Object B.
- Request-level interceptor automatically routes all calls to object replicas

MODIFIABLE MIDDLEWARE

- It should be possible to modify middleware without loss of availability
- Software components can be replaced at runtime
- Component-based design
  - Modifiability through composition
  - Systems may have static or dynamic configuration of components
  - Dynamic configuration requires ***late binding***
  - Components can be changed at runtime
- Component based software supports modifiability at runtime by enabling components to be swapped out.
- **Does a microservices architecture (e.g. AWS Lambda) support modifiability at runtime ?**

MIDDLEWARE: INTERCEPTORS - 2



QUESTIONS

EXTRA SLIDES

24