What is a Software Bus?

- Middleware: sits between operating system and application software
- Much like a hardware bus, but in software :-)
- Allows flexible exchange of information between distributed components
- Synonyms: Object Bus, Information Bus, Event Service, Publish/Subscribe
- Unlike: Message Queues, TP Monitors, ORBs
Publish/Subscribe Model

Publishers

Subscribers

Channels

Events

Application Areas (Classic)

Intranet Software Bus

subscribe ("AAPL");
subscribe ("SUN");
publish ("AAPL", 29.2);
publish("AAPL", 29.3);
publish ("SUN", 43.0);
publish("SUN", 42.7);

publish ("AAPL", 29.3);
subscribe ("AAPL");
subscribe ("SUN");

subscribe ("AAPL");
Application Areas (Today)

- Electronic Business, system integration, stand-alone applications: e.g., Lotus eSuite™

In-Application Software Bus (E.g., InfoBus™)

Application Areas (Future)

- Interconnect appliances, board computers, ...

- Infrared, Satellite, Paging, GPS-Data, ...

Ubiquitous Software Bus
Comparison with CORBA, RMI (1)

- **CORBA, RMI**
  - Point-to-Point communication links
  - Synchronous request/reply model (information “pull”)
  - Interfaces and object references
  - Predefined data types (CORBA-2)

- **Software Bus**
  - Group communication
  - Asynchronous delivery of events (information “push”)
  - No interfaces, spontaneous networking much like JINI
  - User-defined data objects and format specifications

Comparison with CORBA, RMI (2)

- **CORBA, RMI**
  - ✔ Supports “classic” client/server model (2 Tier, N-Tier)
  - ✔ Multiple programming languages, access to legacy code
  - ✔ Well accepted standard
  - ✗ Stubs are “wired” into applications ☩ tight coupling of components ☩ limited extensibility
  - ✗ Few, hardwired “qualities of service”

- **Software Bus**
  - ✔ Systems that distribute events to many receivers almost in real-time
  - ✔ Systems that can be extended at run-time
  - ✔ Components are loosely coupled (see InfoBus)
  - ✗ Weak typing ☩ typing bugs are discovered at run-time
  - ✗ No clear standards (except OMG Event Service)
“Hub-And-Spoke” Architecture

- Publishers
- Hub
- Subscribers
- TCP

“Multicast Bus” Architecture

- A fully distributed and scalable architecture
- No “hub” processes required
- Fault-tolerance
- Events are transmitted by IP multicast

send (datagram, 226.1.1.1)  recv (226.1.1.1)
### Code Example

Uses the iBus Software Bus by SoftWired

```java
// Declare a communication channel:
ChannelURL url = new ChannelURL("ibus:///quotes/bluechips");
Channel blueChipsUSA = new Channel(url);

// Declare a publisher of information:
Publisher p = new Publisher(blueChipsUSA);

// Publish a java.lang.String object:
p.publish("SUNW 47 3/8 +1/16");

// Declare a communication channel:
ChannelURL url = new ChannelURL("ibus:///quotes/bluechips");
Channel blueChipsUSA = new Channel(url);

// Declare a subscriber:
Subscriber s = new Subscriber(blueChipsUSA);

// Register a listener (JavaBeans model):
s.addPublishListener(consumer);
```

### Products

- **OrbixEvents, OrbixTalk** ([www.iona.com](http://www.iona.com))
- **VisiBroker Events** ([www.inprise.com](http://www.inprise.com))
- **TIBCO** ([www.tibco.com](http://www.tibco.com))
- **Vitria Velociti** ([www.vitria.com](http://www.vitria.com))
- **Talarian SmartSockets** ([www.talarian.com](http://www.talarian.com))
- **Java iBus** ([www.softwired-inc.com/ibus](http://www.softwired-inc.com/ibus))
Summary

- Publish/Subscribe versus request/reply
- Enables spontaneous networking of components and applications
- Ensures loose coupling of components
- Has been applied to (real-time) financial information systems
- Is being applied to E-Business, system integration, and more
- Growing importance according to analysts
- No clear standards yet, many offerings, certain confusion