THINK

iBus

SEE

iBus

HEAR

iBus

EXPERIENCE

iBus

Connecting the WORLD

iBus

Enterprise Messaging Middleware for Business Critical Systems

Business Aspects (1)

➢ **Customers are Demanding;**

- **Access** from anywhere, any time from any device. **Mobile**, **Wireless** is the key.
- **Alerts** if important events occur. “Push“ instead of “pull“ is key to success in the market.
- **Immediate closing**. Batch is “out“, **live, real-time** interaction with direct access to back-end systems is “in“.
- **Up-to-date** information. **Real-time** information push instead of polling.

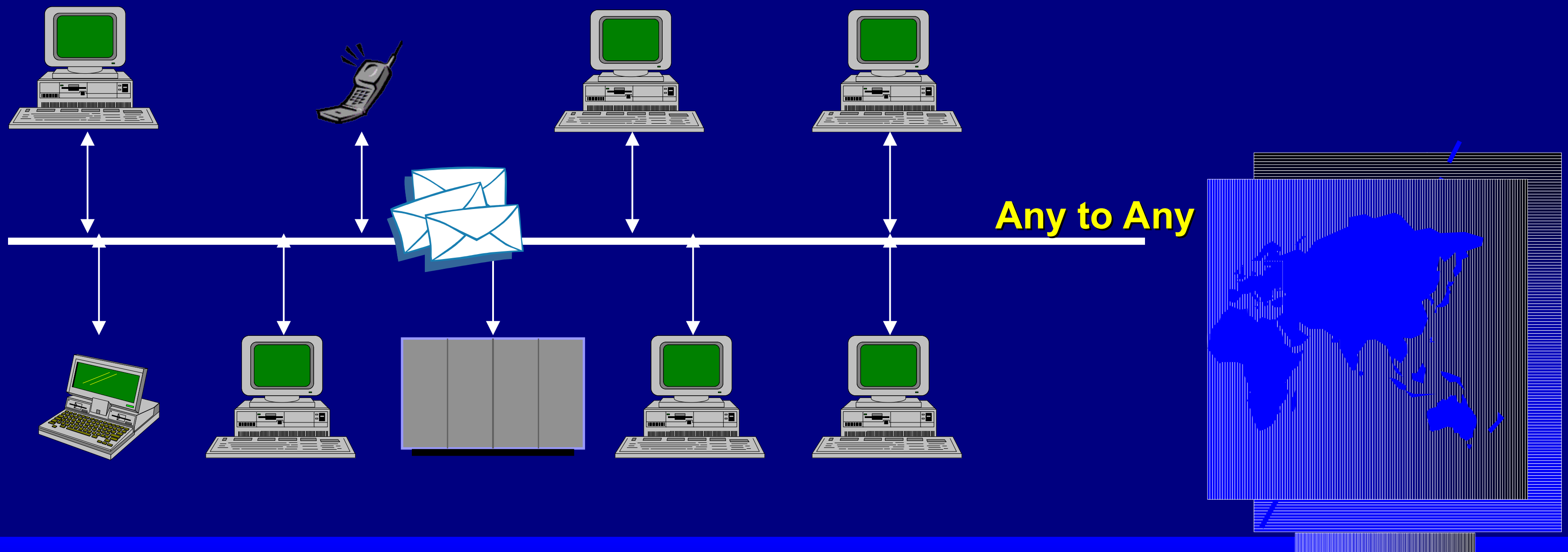
Business Aspects (2)

- **Information systems are becoming more complex**
  - New and old applications must be **integrated** (there is no such thing as “clean-room“!)
  - Business **requirements change** during development
  - Target platforms become more and more **heterogeneous**
  - (Ms&As, systems must run on customer‘s and supplier‘s machines)
  - The number of **interacting parties** explodes

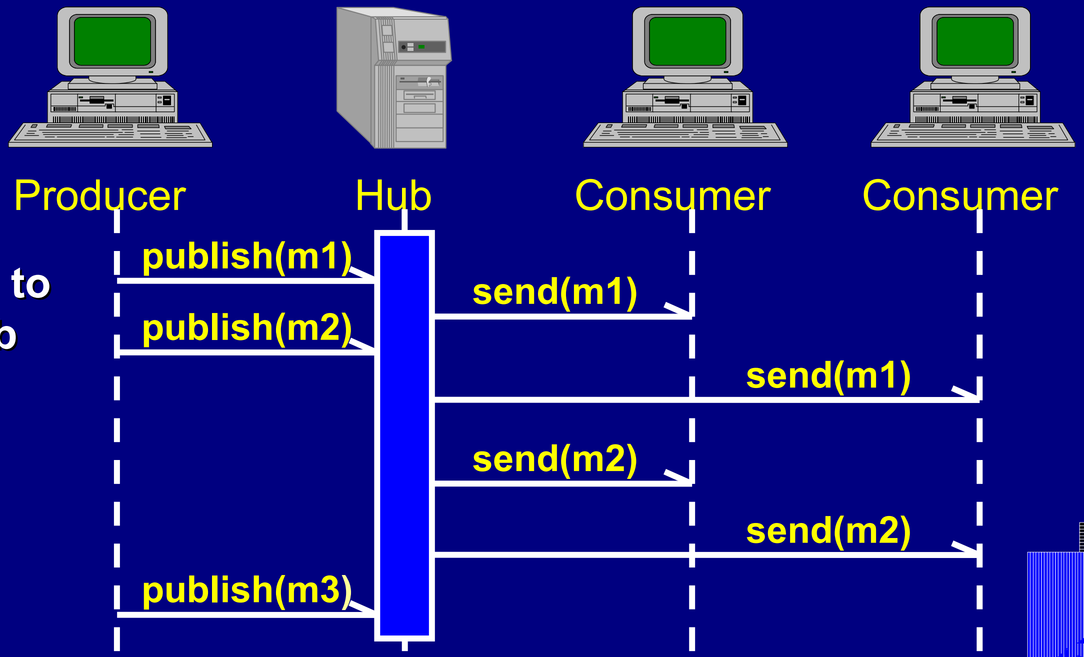
- **Reduce complexity - iBus Connecting the WORLD**

What is Messaging?

- Messaging is a communication model, in which loosely coupled components exchange self-describing messages.
- Logical View of publish/subscribe messaging



Publish / Subscribe



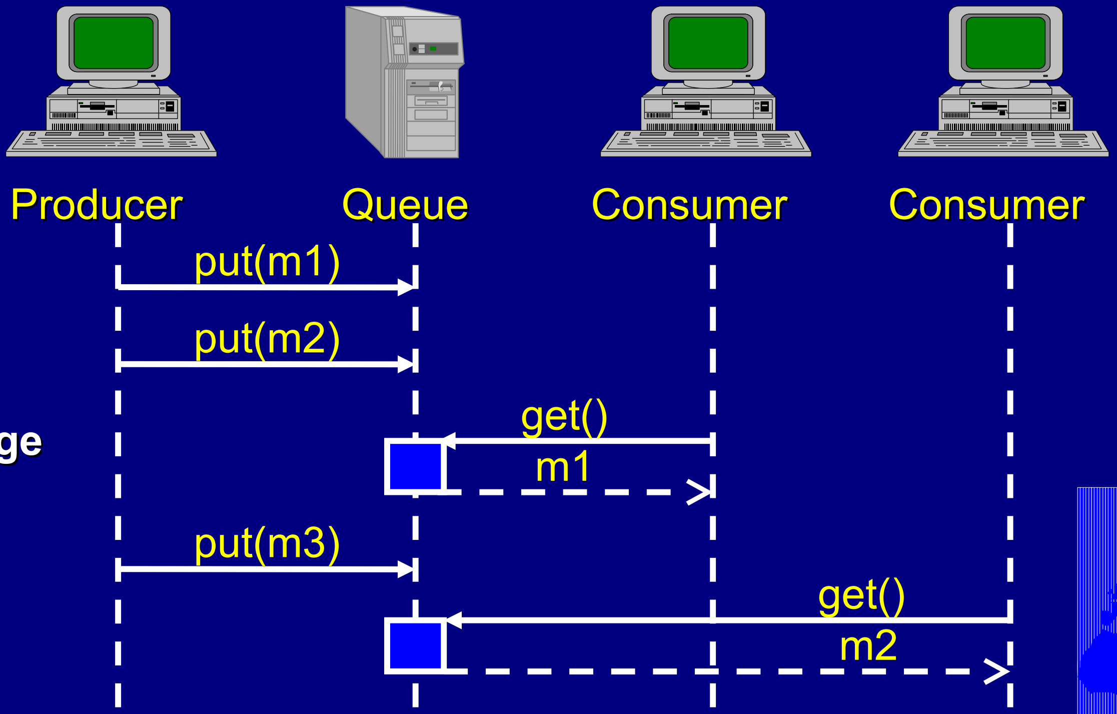
Pass a message to the message hub

Forward the message to all subscribed consumers

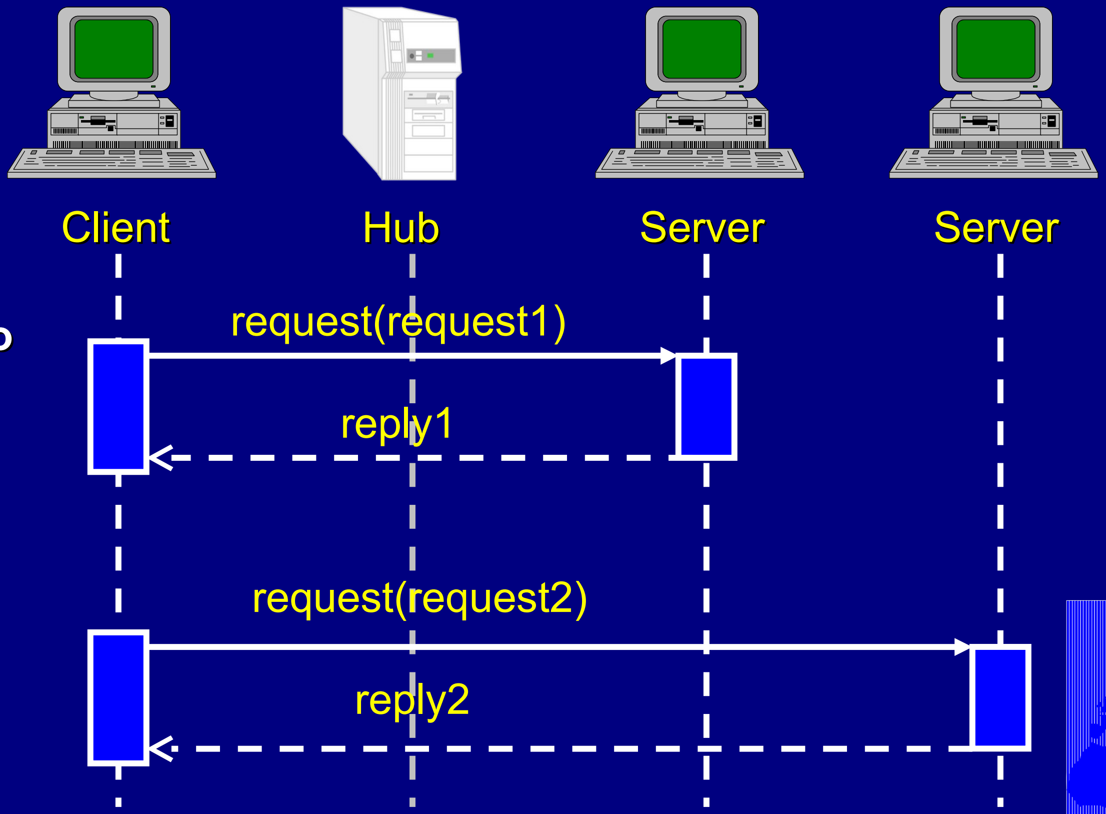
Message-Queueing

Put message in queue

Consume message



Compare to CORBA / RMI



Invoke request to server and wait for reply

The Java Message Service • JMS

- **First and only standard in the MOM area**
- **Two messaging models**
  - Point-to-Point
  - Publish/Subscribe
- **5 Message types** *(several suited for **XML**)*
- **Qualities-of-service**
  - Volatile Messages *(reliable and best-effort)*
  - Persistent Messages
  - Transactions

JMS Producer

Initialize JMS

- ☞ topic = IBusJMSContext.getTopic(”quotes");
- ☞ session = IBusJMSContext.getTopicSession(...);
- ☞ publisher = session.createPublisher(topic);

Compose the message

- ☞ message = session.createTextMessage(...);

Publish the message

- ☞ publisher.publish(message);

JMS Consumer

Initialize JMS

```
☞ topic        = IBusJMSContext.getTopic(”quotes");
☞ session      = IBusJMSContext.getTopicSession(...);
☞ subscriber   = session.createSubscriber(topic);
```

Setup consumer

```
☞ consumer     = new MyConsumer();
☞ subscriber.setMessageListener(consumer);
```

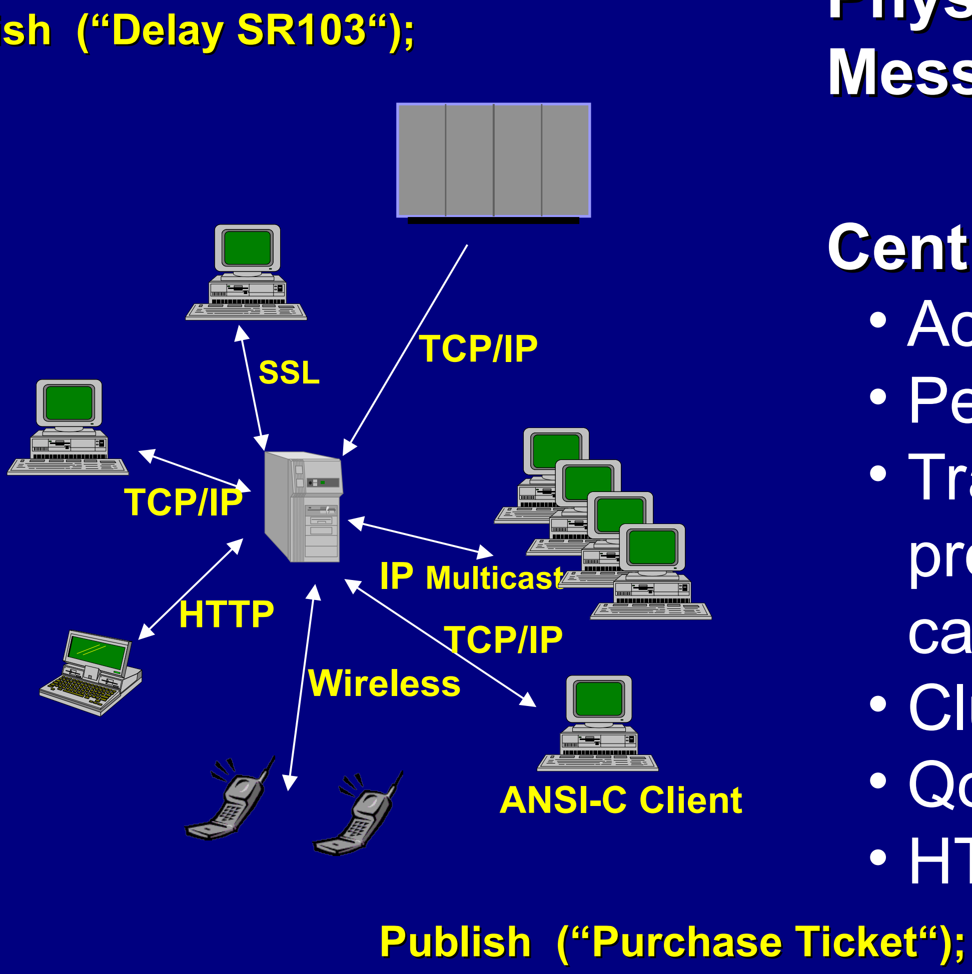
Declaration of message handler

```
☞ void onMessage(Message message);
```

The SoftWired iBus

- "Pure Java" JMS implementation
- ANSI-C API available
- Focus on lightweight and speed
- Industry's only finetunable Quality of Service (QoS)
- Industry's only protocol-bridging JMS product
  - Today: messaging via TCP/IP, SSL, UDP, IP Multicast, HTTP
  - Developing: **Messaging via Wireless Protocols** (WAP, SMS, GPRS)

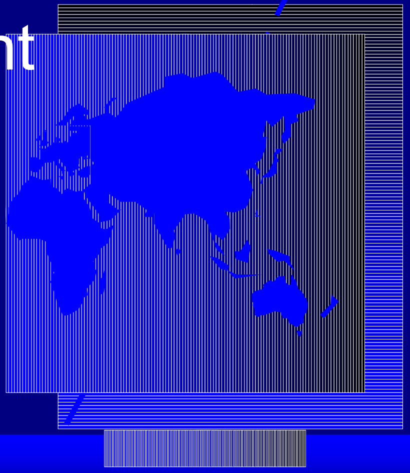
iBus//MessageServer



Physical view of the Messaging Infrastructure

Central Server Architecture:

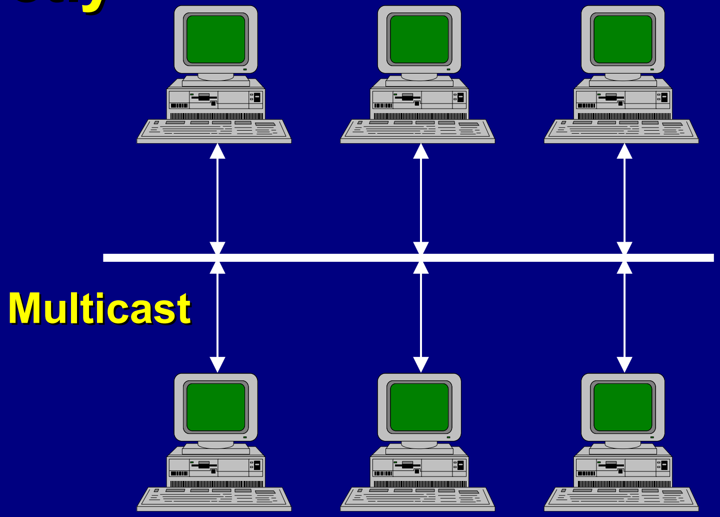
- Access control possible
- Persistent messages
- Transport protocols of producers and consumers can be different
- Clustering
- Qos
- HTTP / SSL



iBus//MessageBus

- **Components communicate directly**

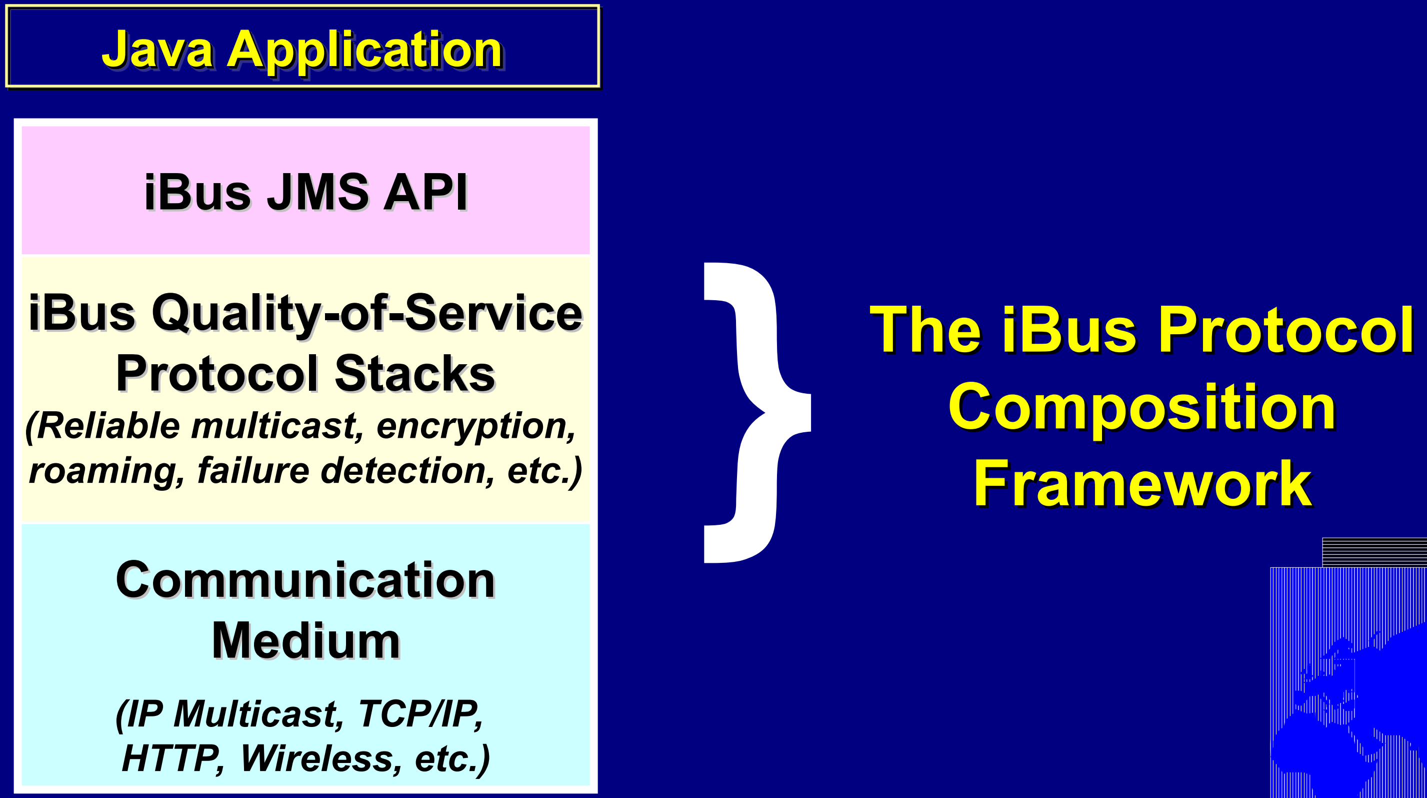
- Deploys **multicast networks** (**IP multicast, satellite,** etc.)
  - Easy to embed (library)
  - High-speed, reliable group coordination features
  - Inherent scalability
  - Inherent fault tolerance
  - Components need to agree on transport protocol
  - **Zero Maintenance Zero Administration**



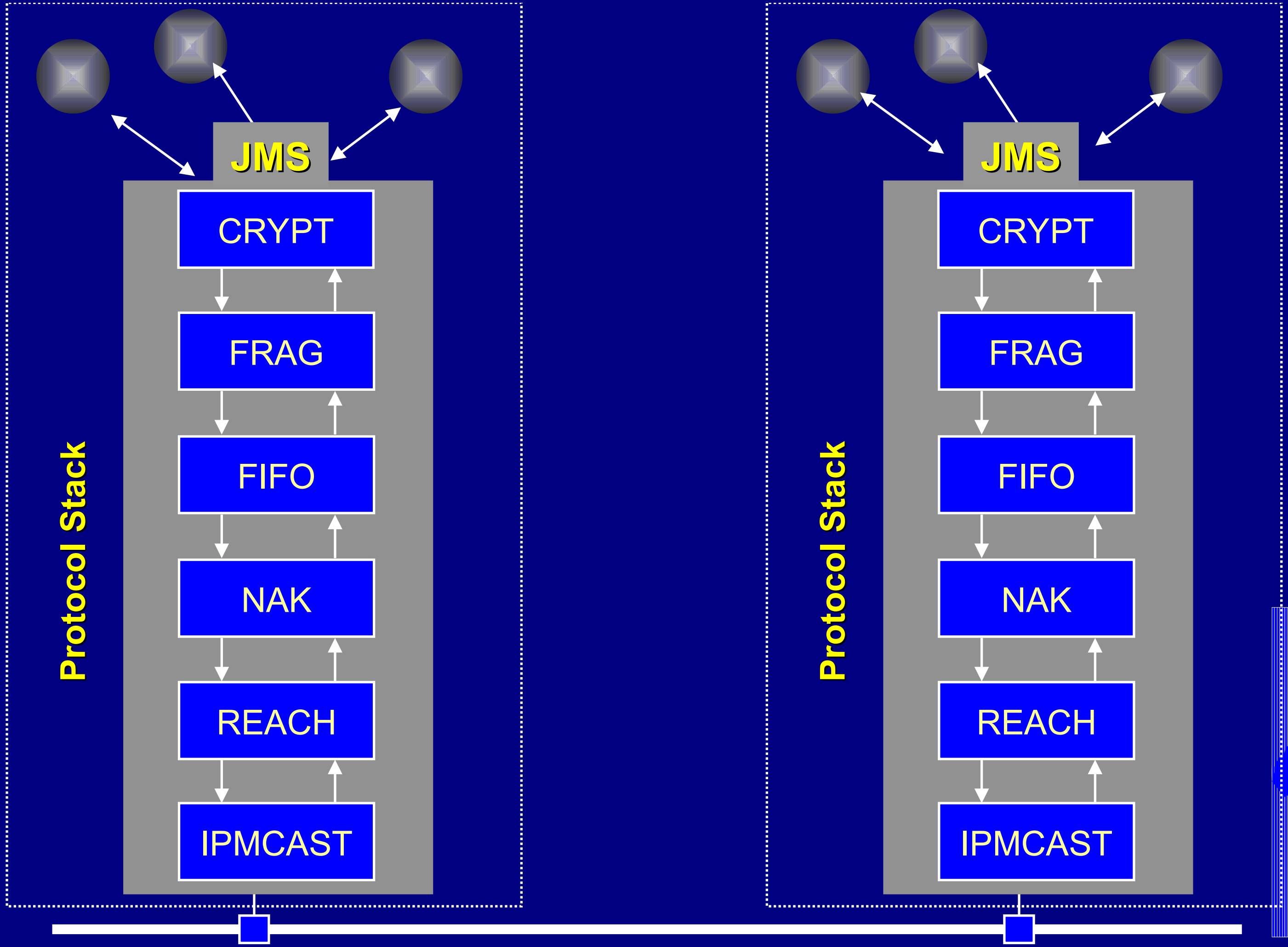
Choose your Quality of Service

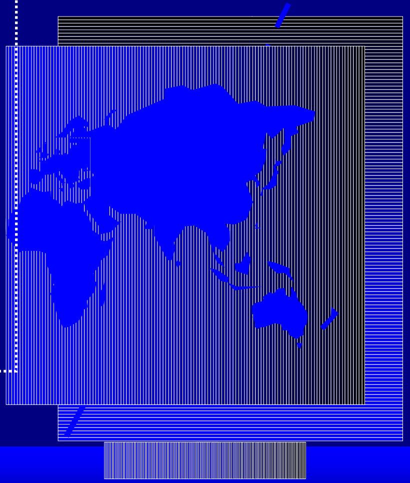
- **"Guaranteed" Message Delivery =**
  Very Secure High Latency and Less Throughput
  - Accounting, Games involving Money etc.
- **Volatile over TCP, SSL, HTTP =**
  Low Latency High Throughput, but Less Secure
  - Life Games, Many Participants, Occasional Packet Loss Tolerable
- **Forward Error Correction, UDP, Multicast =**
  Highest Throughput (constant scaleability)
  - Realtime Games, Private User Groups

The Versatility of iBus


Java Application
iBus JMS API
iBus Quality-of-Service Protocol Stacks
(Reliable multicast, encryption, roaming, failure detection, etc.)
Communication Medium
(IP Multicast, TCP/IP, HTTP, Wireless, etc.)
The iBus Protocol Composition Framework

The iBus Protocol Stack





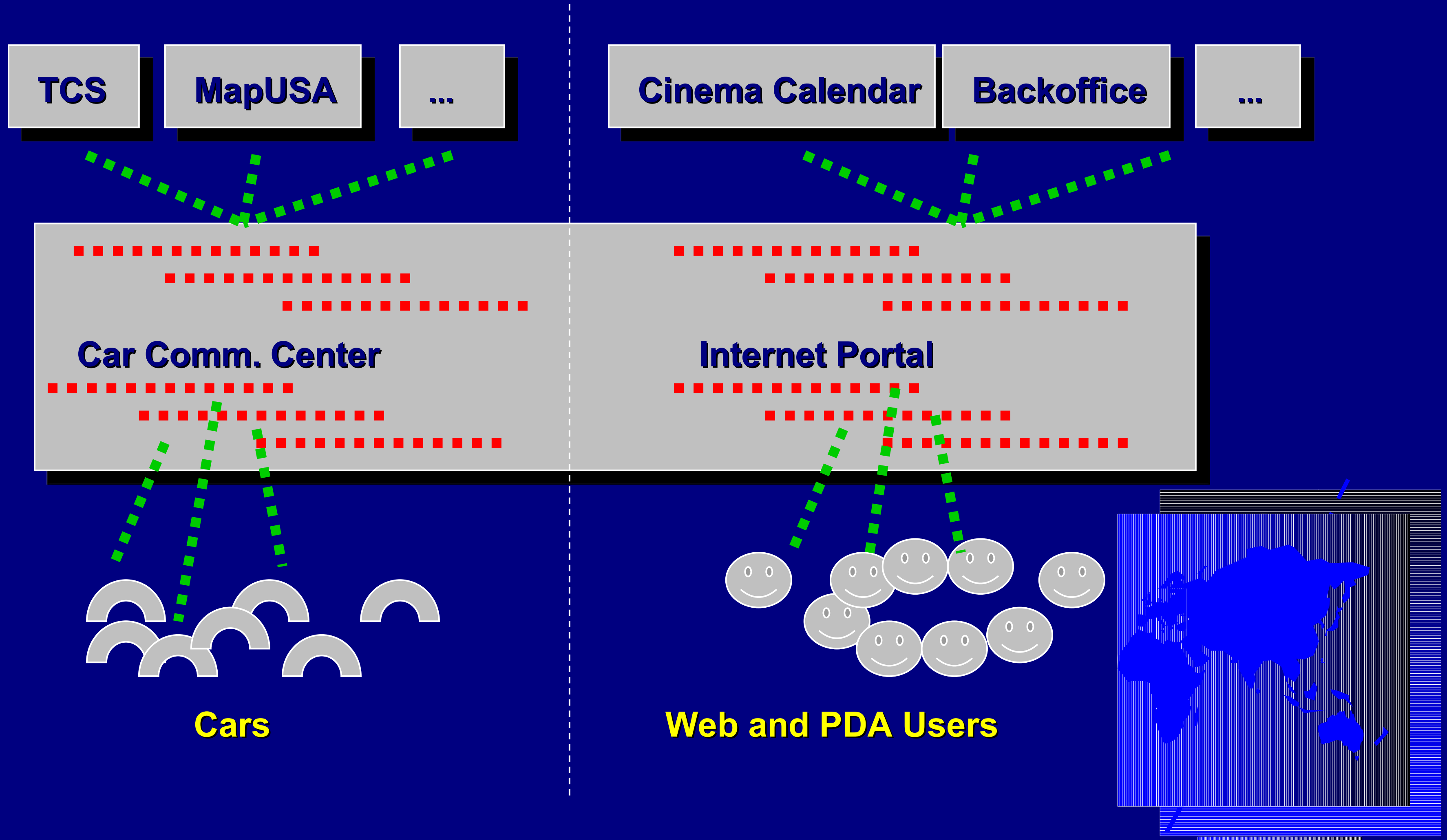
The iBus Product Line

- **Core Products**
  - iBus//MessageBus: **Zero maintenance** architecture
  - iBus//MessageServer: Server based architecture
- **Add-Ons**
  - iBus//ANSI-C
  - iBus//RealTime: **IP Multicast** for the message server
  - iBus//Web: **HTTP**(S) Transport Protocol
  - iBus Modules for **Wireless** Protocols
- **Infos & Download**
  - ☝ www.JavaMessaging.com/ibus

iBus for Portals: An Analogy

- The next generation Telematics Platform
- **Dozens** of *Service Providers*
- **Millions** of *Cars*
- In-between: A huge "Switchboard", driven by **iBus//MessageBus**
- Communication from/to Cars and Service Providers: **iBus//MessageServer**

The Analogy - explained



[Click for next slide]

iBus

Connecting the WORLD