Future of CORBA for Distributed Real-time & Embedded Systems

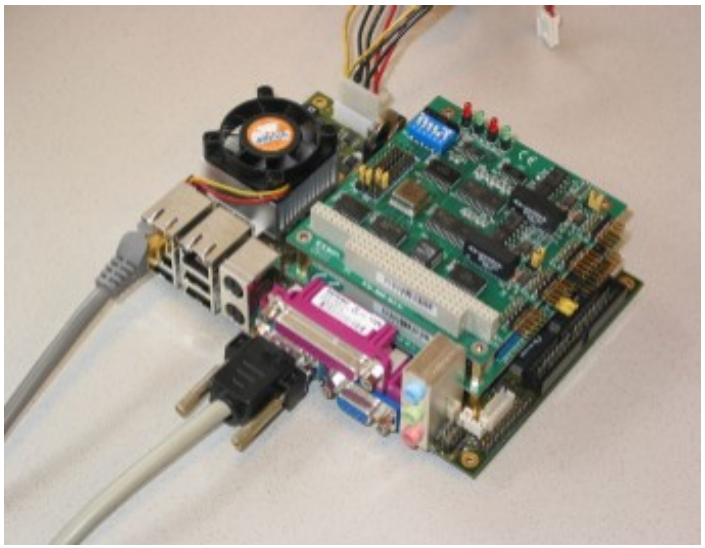
Thursday, October 18, 2007, ICALEPCS

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt/

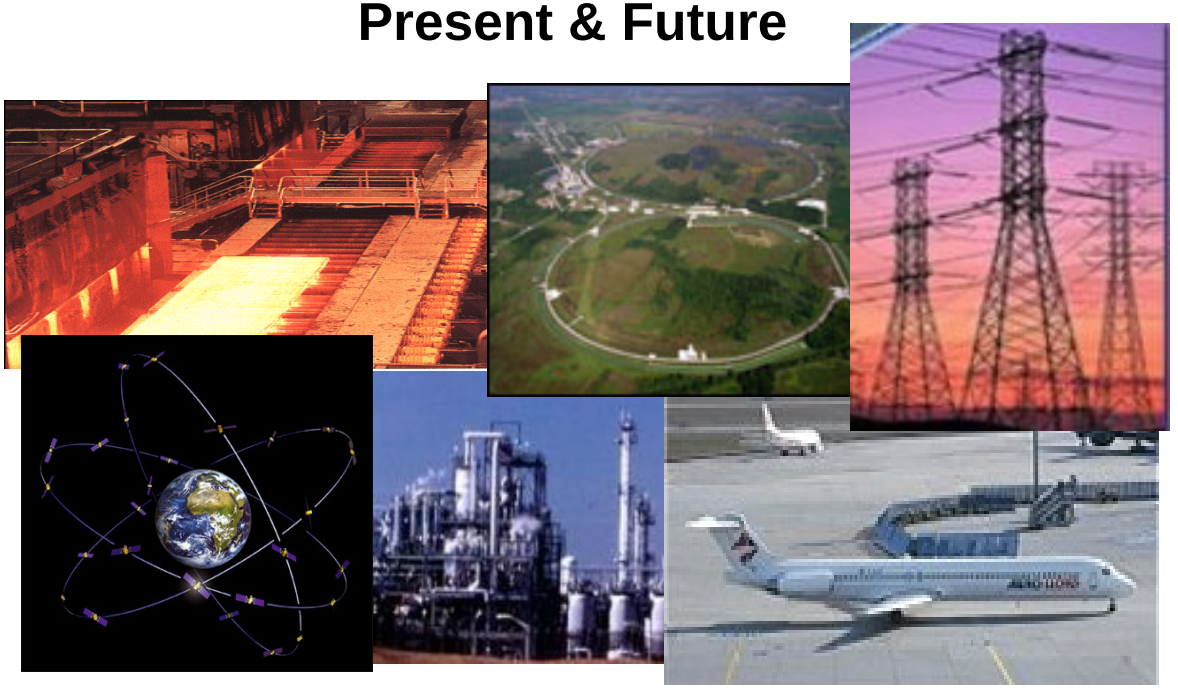
Institute for Software Integrated Systems
Vanderbilt University Nashville, Tennessee

Distributed Real-time & Embedded (DRE) Systems

The Past



Present & Future



Stand-alone real-time & embedded systems

- Stringent quality of service (QoS) demands
  - e.g., latency, jitter, footprint
- Resource constrained

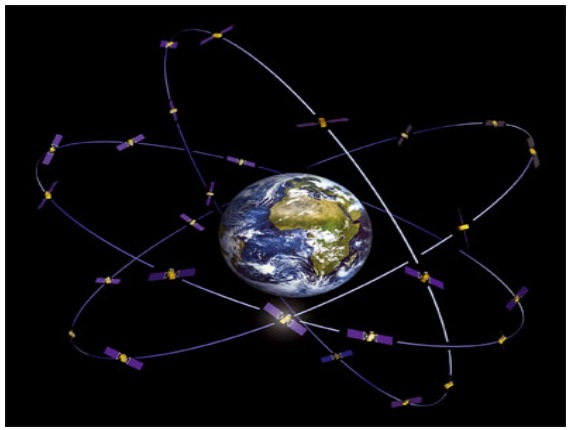
Enterprise distributed real-time & embedded (DRE) systems

- Network-centric "systems of systems"
- Stringent simultaneous QoS demands
  - e.g., dependability, security, scalability, etc.
- Dynamic context

This talk focuses on technologies for enhancing DRE system QoS, productivity, & quality

Diverse Mission-Critical DRE System Characteristics

SCADA & C2



Transport Management

Air Traffic Control

These systems have characteristics of enterprise & real-time embedded systems

- Typically heterogeneous & complex, requiring support for:
  - Different hardware platforms
  - Software written in different programming languages
  - Highly distributed net-centric environment(s)
- Need to assure efficient, predictable, & scalable end-to-end QoS
- Need dynamic reconfiguration to support varying workloads over operational lifecycle of system
- Need to be affordable to reduce initial system acquisition costs & recurring upgrade & evolution costs

Challenge: Selecting Middleware for DRE Systems

- Develop software entirely in-house using proprietary solutions

- Develop software using domain-specific, community-based technologies

MPI

PVM
Parallel Virtual Machine

- Develop software using latest commercial-off-the-shelf (COTS) technologies

Microsoft Windows

W3C WORLD WIDE WEB consortium

Microsoft .net

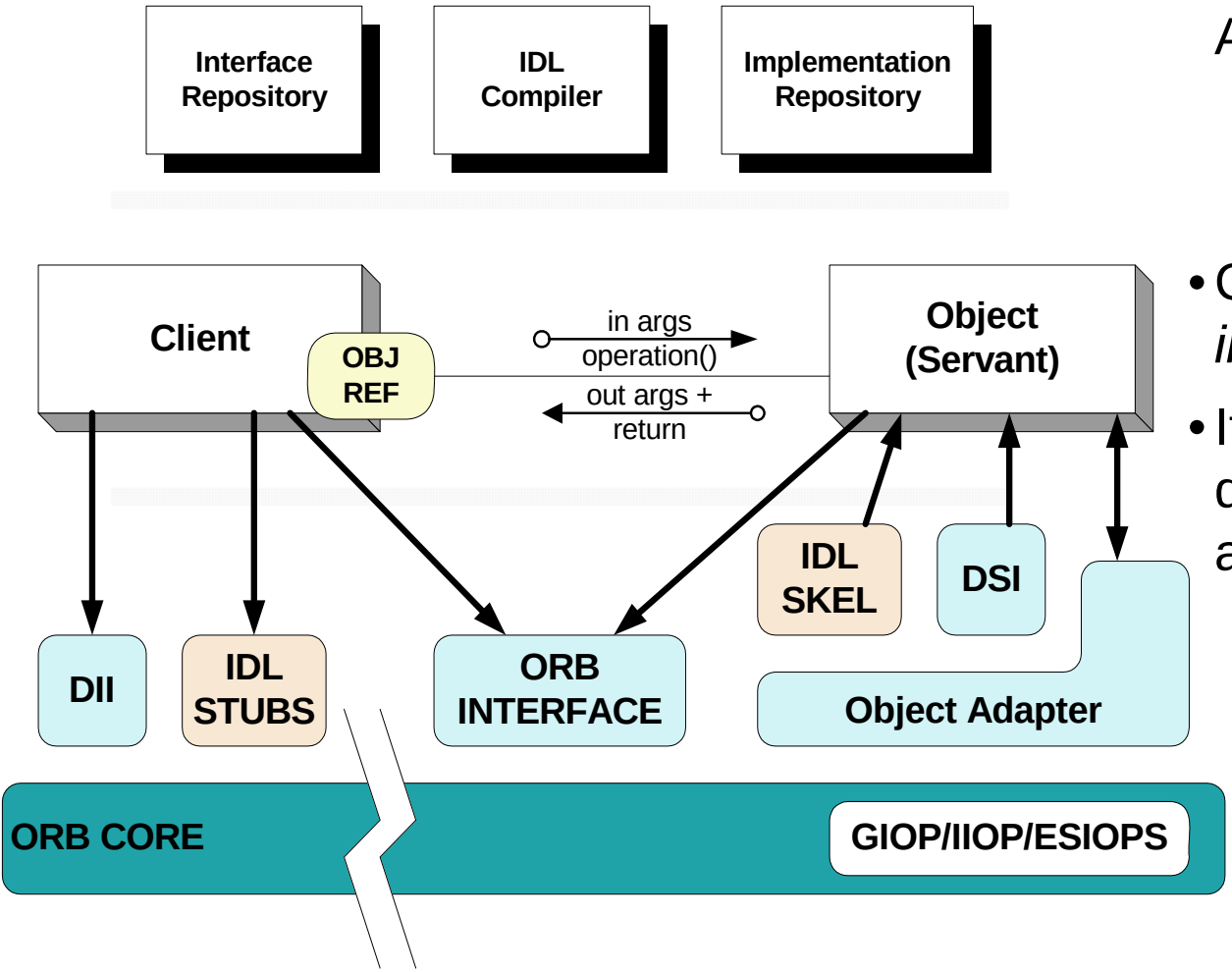
- Develop software using mature standards-based technologies

OMG OBJECT MANAGEMENT GROUP

CORBA

DDS

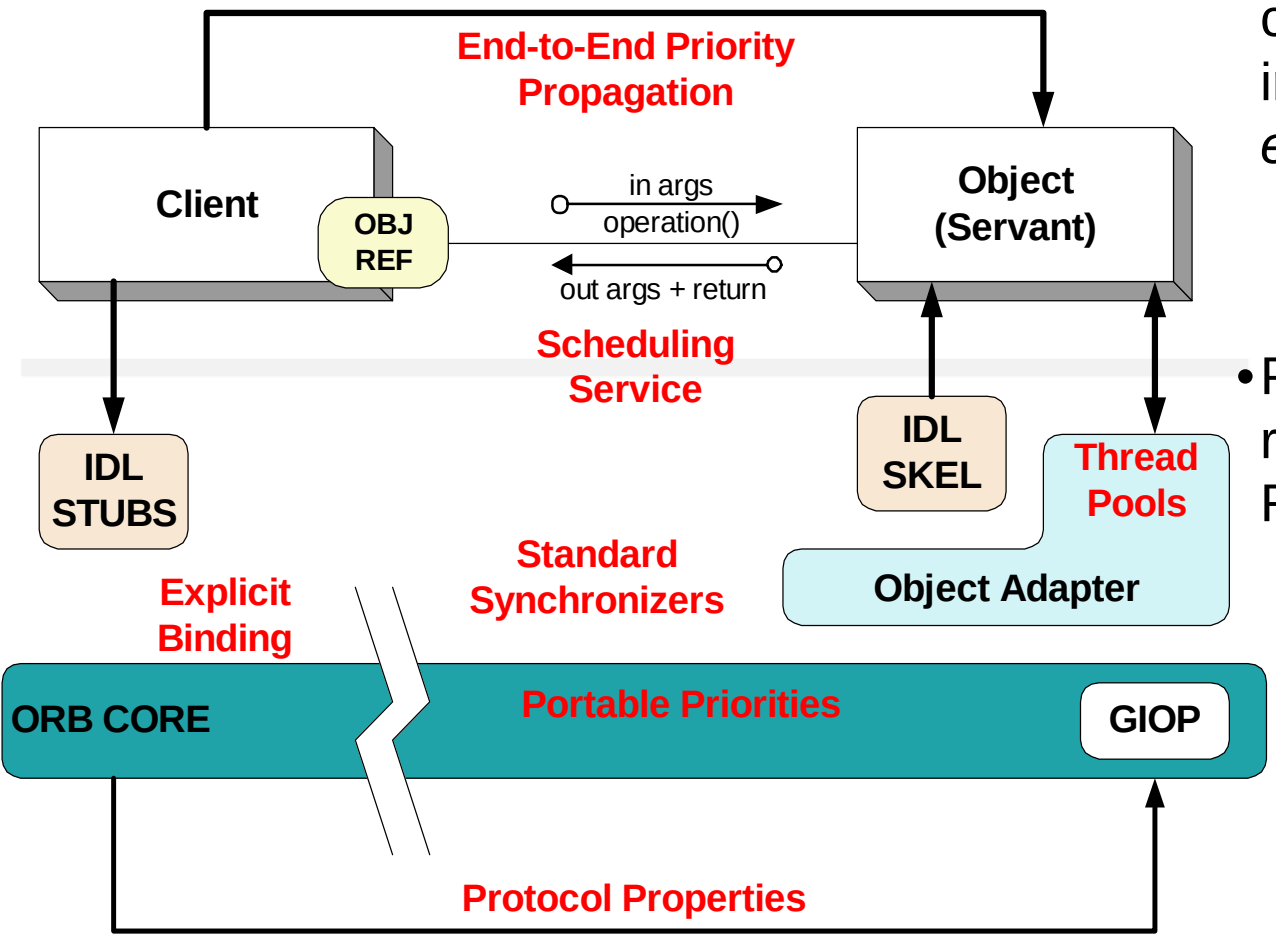
Overview of CORBA



- Common Object Request Broker Architecture (CORBA)
  - A family of specifications
  - OMG is the standards body
- CORBA defines *interfaces*, not *implementations*
- It simplifies development of distributed applications by automating/encapsulating
  - Object location
  - Connection & memory mgmt.
  - Parameter (de)marshaling
  - Event & request demultiplexing
  - Error handling & fault tolerance
  - Object/server activation
  - Concurrency
  - Security

- CORBA shields applications from hetero-geneous platform *dependencies*
  - *e.g.,* languages, operating systems, networking protocols, hardware

Overview of Real-time CORBA



Real-time CORBA address *some* (but by no means all) important DRE system development challenges

- Real-time CORBA adds QoS control to regular CORBA to improve application *predictability*, *e.g.*,
  - –Bounding priority inversions &
  - –Managing resources end-to-end
- Policies & mechanisms for resource configuration/control in Real-time CORBA include:

1. Processor Resources

- Thread pools
- Priority models
- Portable priorities
- Synchronization

2. Communication Resources

- Protocol policies
- Explicit binding

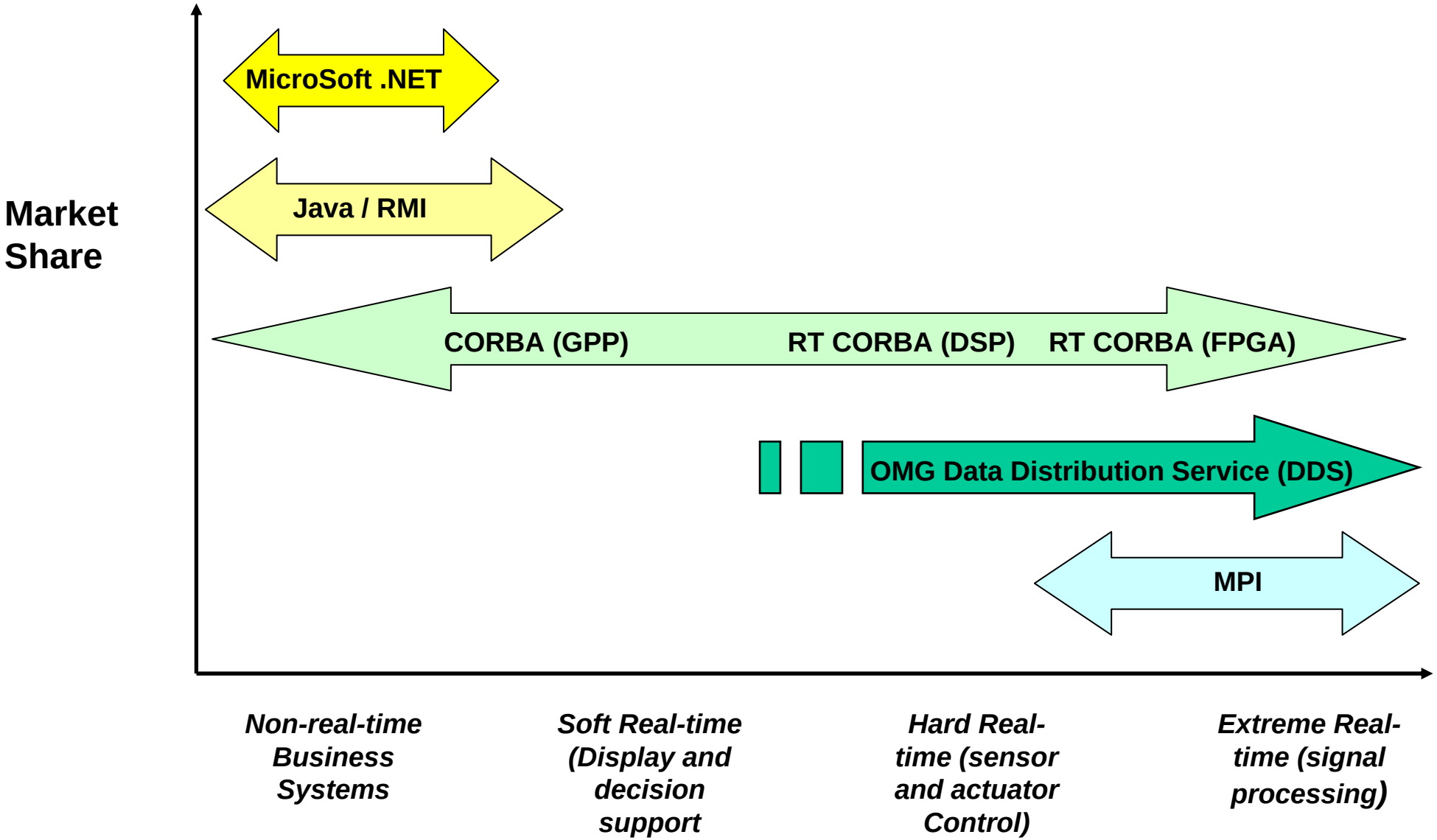
3. Memory Resources

- Request buffering

Why Use CORBA?

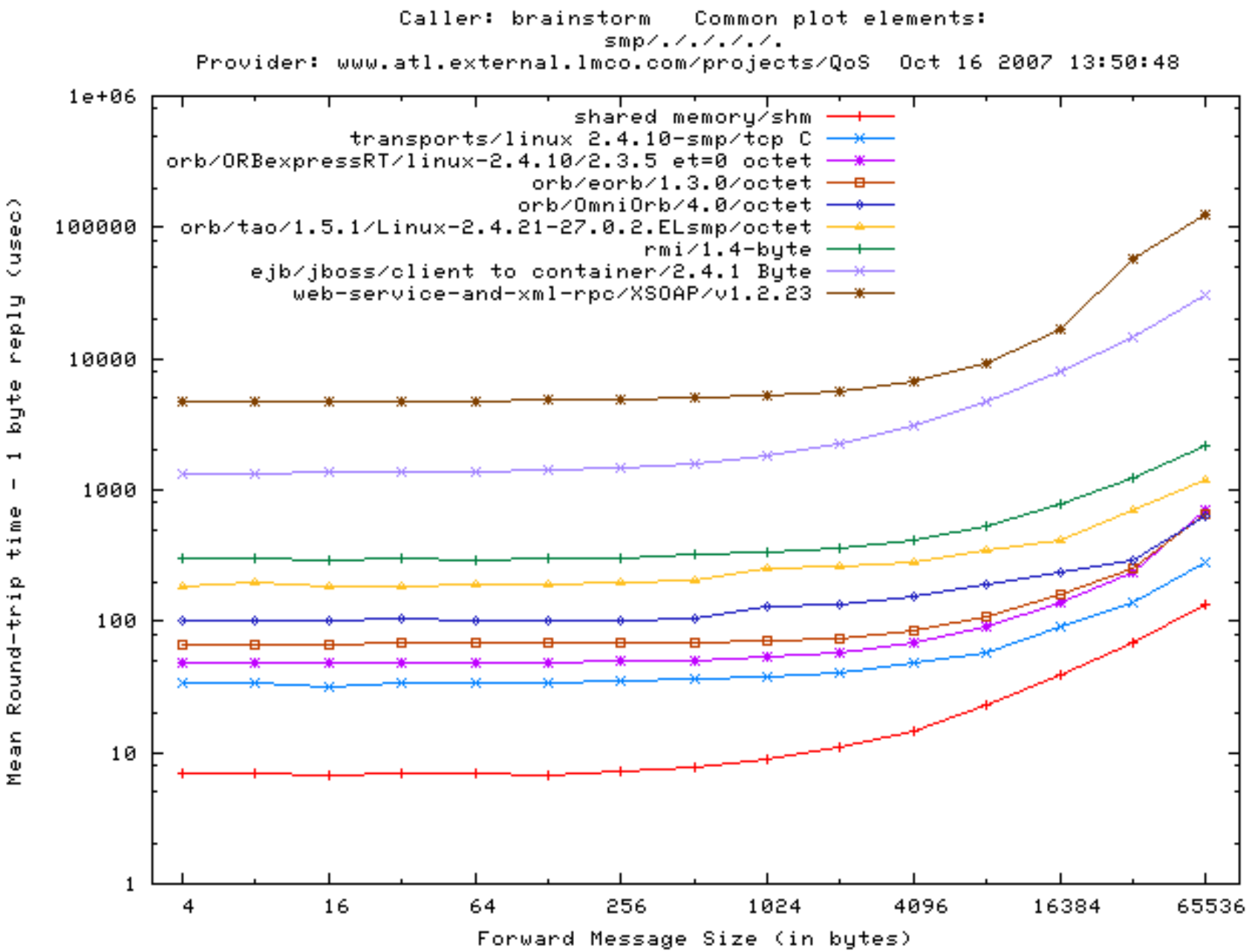
- After all people think CORBA is dead
- Why?
  - Associated with legacy systems
  - Mid 90's technology therefore must be obsolete
  - Perceived as "big & slow"
  - Not exciting to write about
  - They think it died of complexity
- Why not?
  - Inclusive technology
  - Committed, seasoned user base
  - Maturity has led to highly time/space optimized ORBs
  - What works is boring
  - It is solving increasingly complex issues

Span of Middleware Technologies for DRE Systems



Source: OACE Tech. & Stds. Sept. 2003

Alternate Technology Message Speeds



Transport characteristics eventually dominate large messages

Source:Gautam H. Thaker Lockheed Martin Advanced Technology Labs Camden, NJ

The Future of CORBA

- **Improvements in CORBA features & performance**
- **Extensions to the CORBA object model**
- **Complementary technologies**

The Future of CORBA

- **Improvements in CORBA features & performance**
- **Extensions to the CORBA object model**
- **Complementary technologies**

Fixing Problems with the CORBA C++ Mapping

1. Memory management is too complicated & easy to get wrong due to lots of rules to memorize, e.g.,
   - Storing strings within sequences & structs
   - Not handling the return reference from an operation, but passing it to another operation
   - Not setting length() of sequence properly
   - Not duplicating object references properly
   - Not using ServantBase_var properly
2. Lack of standard C++ classes makes CORBA look “old & lame” & causes extra work for programmers
   - e.g., it’s a lot of work to move the data back & forth between the standard C++ types you want to manipulate & the types you need to pass as parameters
3. A tremendous amount of code gets generated for the C++ mapping, leading to bloat & slow compilation
   - The size difference between the same essential set of functionality can be roughly ***on the order of 5:1***
   - e.g., for e*ORB C & C++ on Red Hat 9 Linux compiled with gcc 3.2 the C libec_poa.so is **29 kbytes** C++ vs libe_mpoa.so is **105 kbytes**

Top 10 Things to Fix in C++ Mapping

1. All memory should be self-managed
   - This includes CORBA::Object, sequences, strings, structures of all types, etc
2. Structs & unions should have useful constructors
3. Arrays should be implemented using std::vector<>
4. Fix valuetypes so they use consistent reference counting scheme
5. All types should offer exception-safe swap() operations
6. Use bool, wchar_t, wstring, std::string, std::vector, etc.
   - Do not introduce new types unless you must
7. Repeat number (1) until you reach (10)

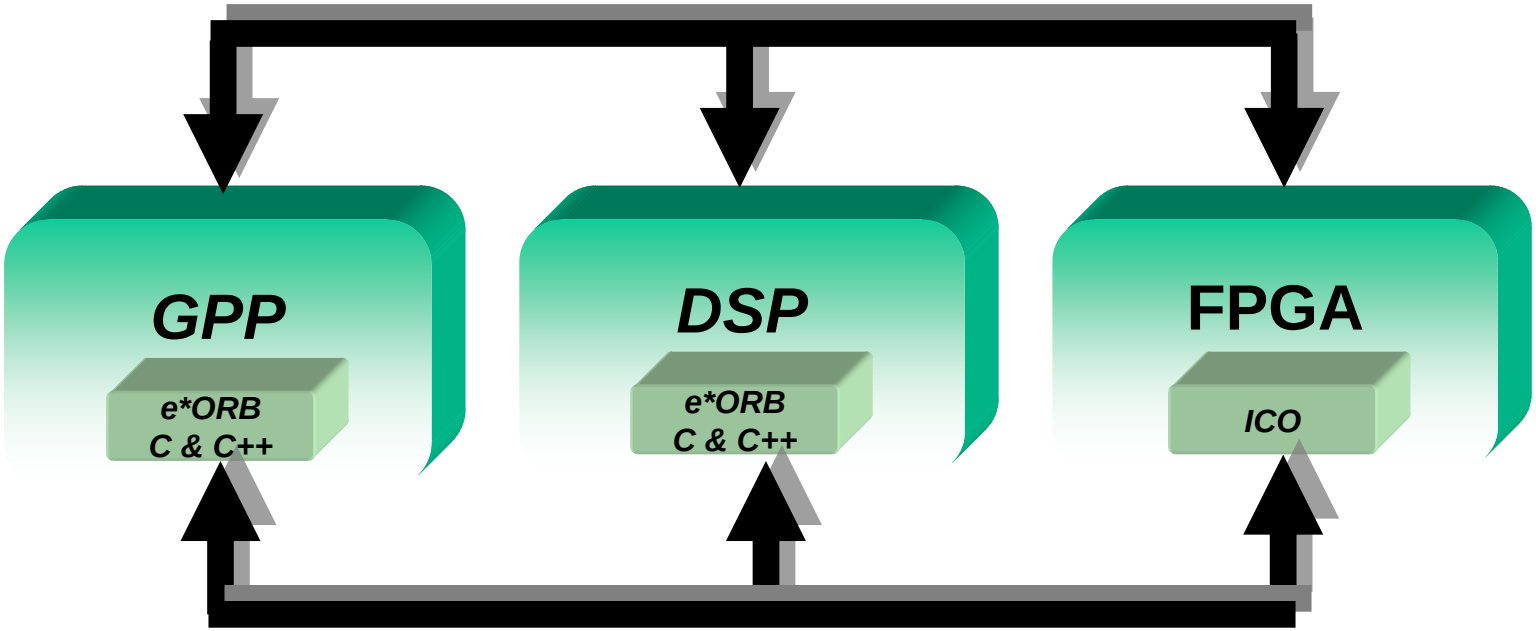
Many more suggestions in CUJ columns by Vinoski & Schmidt

- http://www.ddj.com/dept/cpp/184403757
- http://www.ddj.com/dept/cpp/184403765
- http://www.ddj.com/dept/cpp/184403778

Improvements in CORBA Performance

One benefit of CORBA being a mature standard is that it runs in multiple processor types, including GPP, DSP, & FPGA environments

GIOP Everywhere



Extensible Transport Framework

Key Advantages:

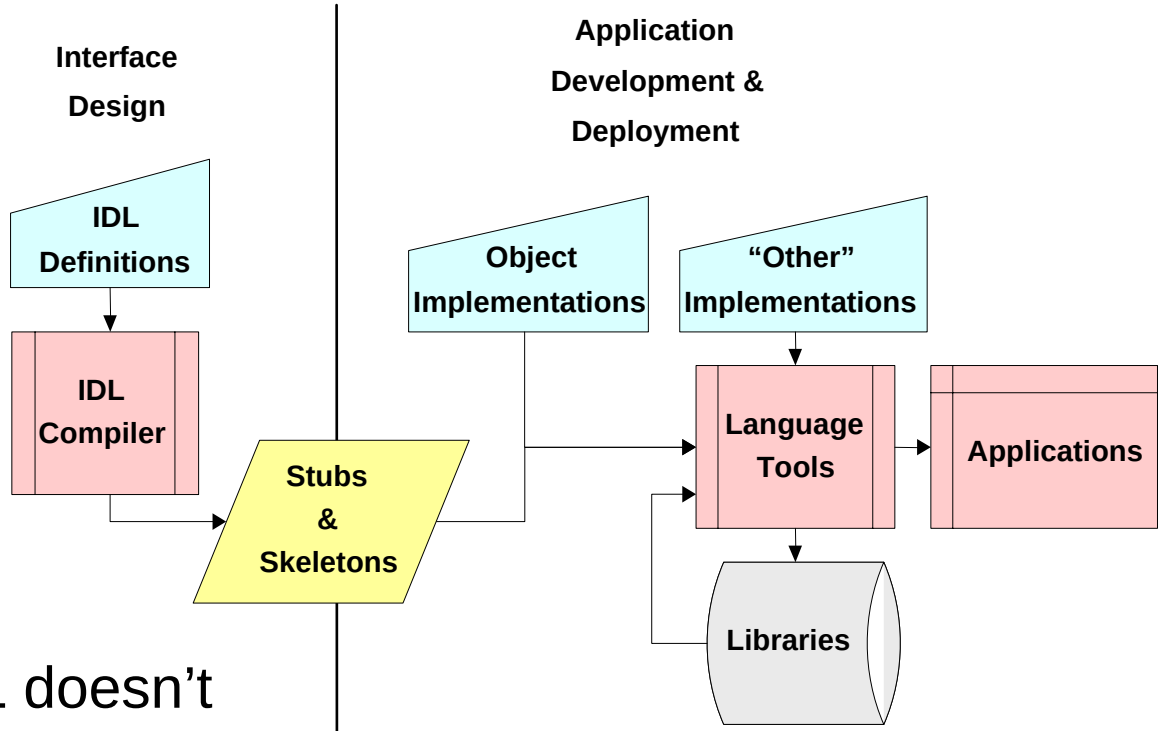
- CORBA message processing can be executed directly in H/W, which is 100x faster than in S/W
- Eliminates the need for S/W proxies/adapters on GPPs, which Reduces overhead/latency & increases throughput
- Supports direct access to application components running on H/W
- Supports vision of architectural consistency across all aspects of the application

The Future of CORBA

- Improvements in CORBA features & performance
- **Extensions to the CORBA object model**
- Complementary technologies

Drawbacks of CORBA Middleware

Distributed Object Computing (DOC) CORBA 2.x application development can be tedious



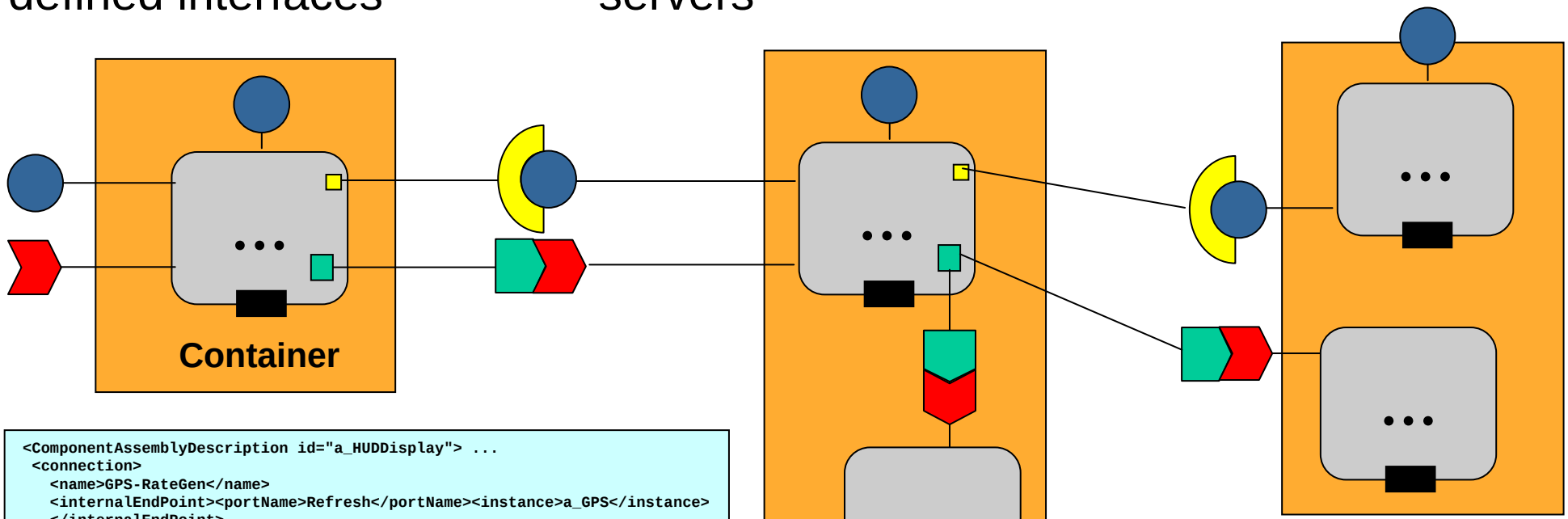
- DOC CORBA IDL doesn't specify how to group related interfaces to offer a *service family*
  - Such "bundling" must be done by developers via idioms & patterns

- DOC CORBA doesn't specify how configuration & deployment of objects should be done to create complete applications
  - Proprietary infrastructure & scripts are written by developers to facilitate this

DOC CORBA 2.x defines interfaces & policies, but *not* implementations

Solution: Component Middleware

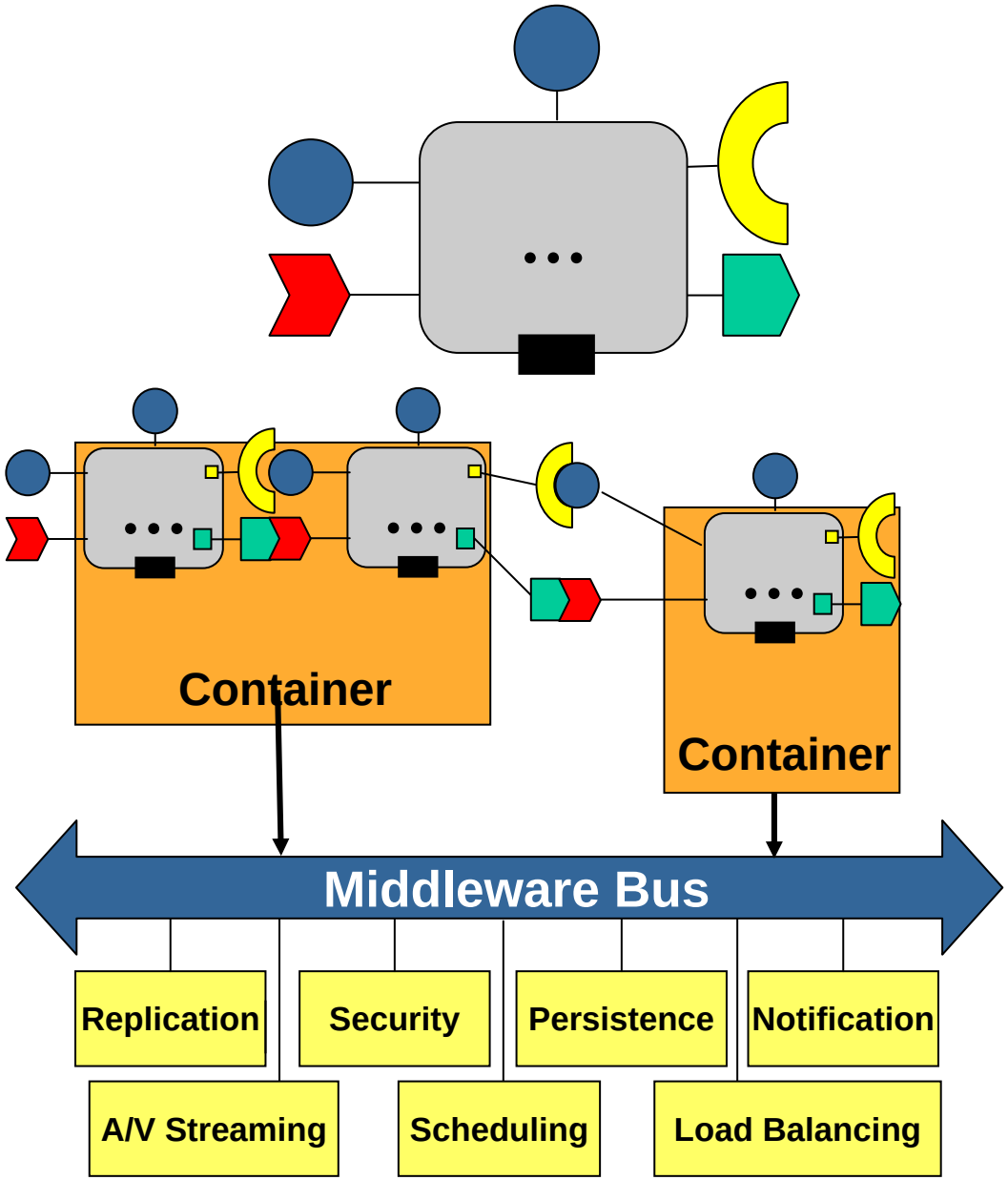
- Creates a standard "virtual boundary" around application component implementations that interact only via well-defined interfaces
- Define standard container mechanisms needed to execute components in generic component servers
- Specify the infrastructure needed to configure & deploy components throughout a distributed system




```
<ComponentAssemblyDescription id="a_HUDDisplay"> ...
 <connection>
   <name>GPS-RateGen</name>
   <internalEndPoint><portName>Refresh</portName><instance>a_GPS</instance>
   </internalEndPoint>
   <internalEndPoint>
      <portName>Pulse</portName><instance>a_RateGen</instance>
   </internalEndPoint>
  </connection>
  <connection>
    <name>NavDisplay-GPS</name>

<internalEndPoint><portName>Refresh</portName><instance>a_NavDisplay</insta
nce>
    </internalEndPoint>
    <internalEndPoint><portName>Ready</portName><instance>a_GPS</instance>
    </internalEndPoint>
  </connection> ...
</ComponentAssemblyDescription>
```

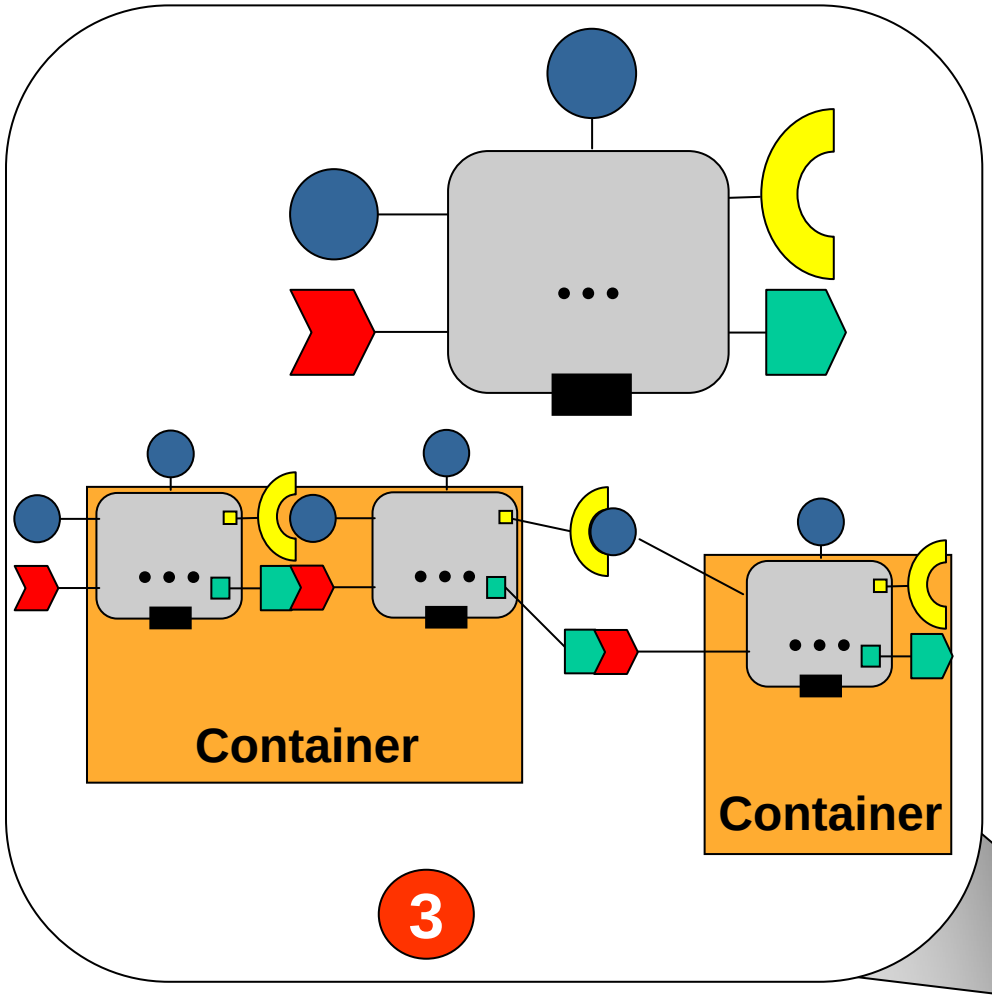
Overview of Lightweight CORBA Component Model

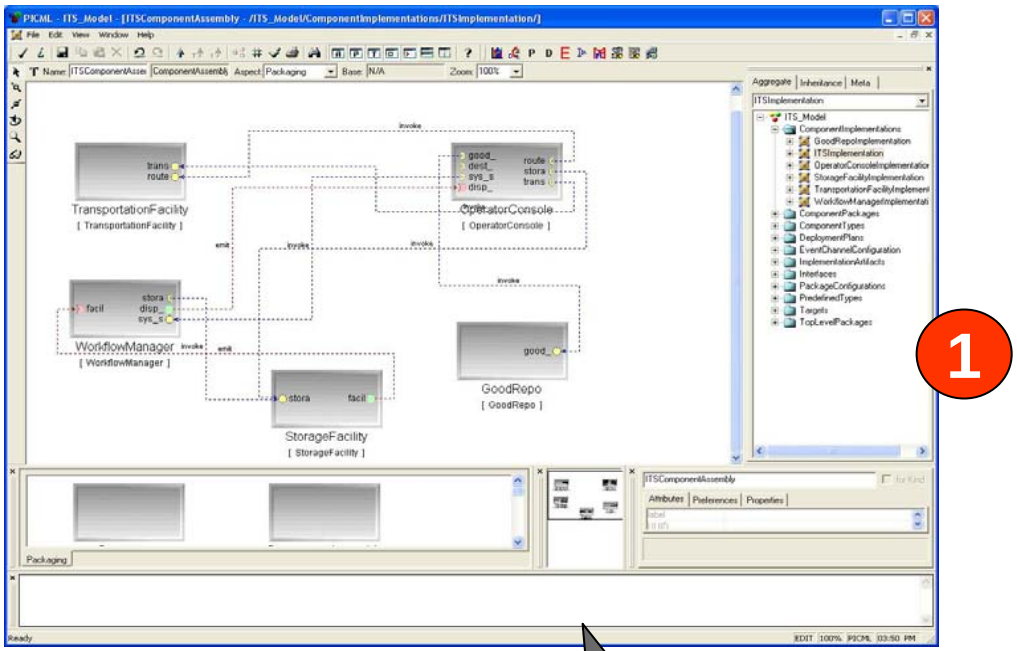


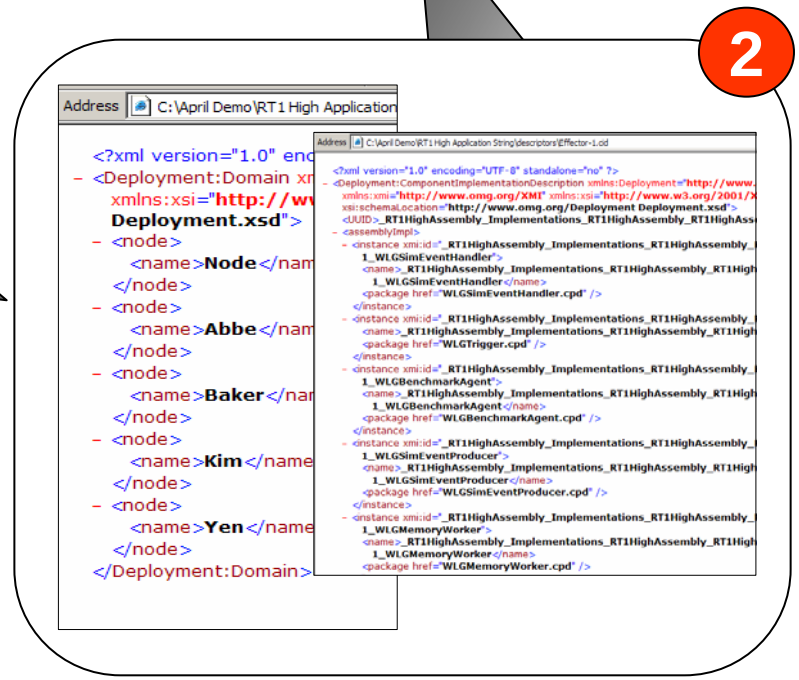
- *Components* encapsulate application "business" logic
- Components interact via *ports*
  - *Provided interfaces*, e.g.,facets
  - *Required connection points*, e.g., receptacles
  - *Event sinks & sources*
  - *Attributes*
- *Containers* provide execution environment for components with common operating requirements
- Components/containers can also
  - Communicate via a *middleware bus* and
  - Reuse *common middleware services*

Lightweight CCM defines interfaces & policies, & *some* implementations

Applying Model-Driven Engineering to Lightweight CCM





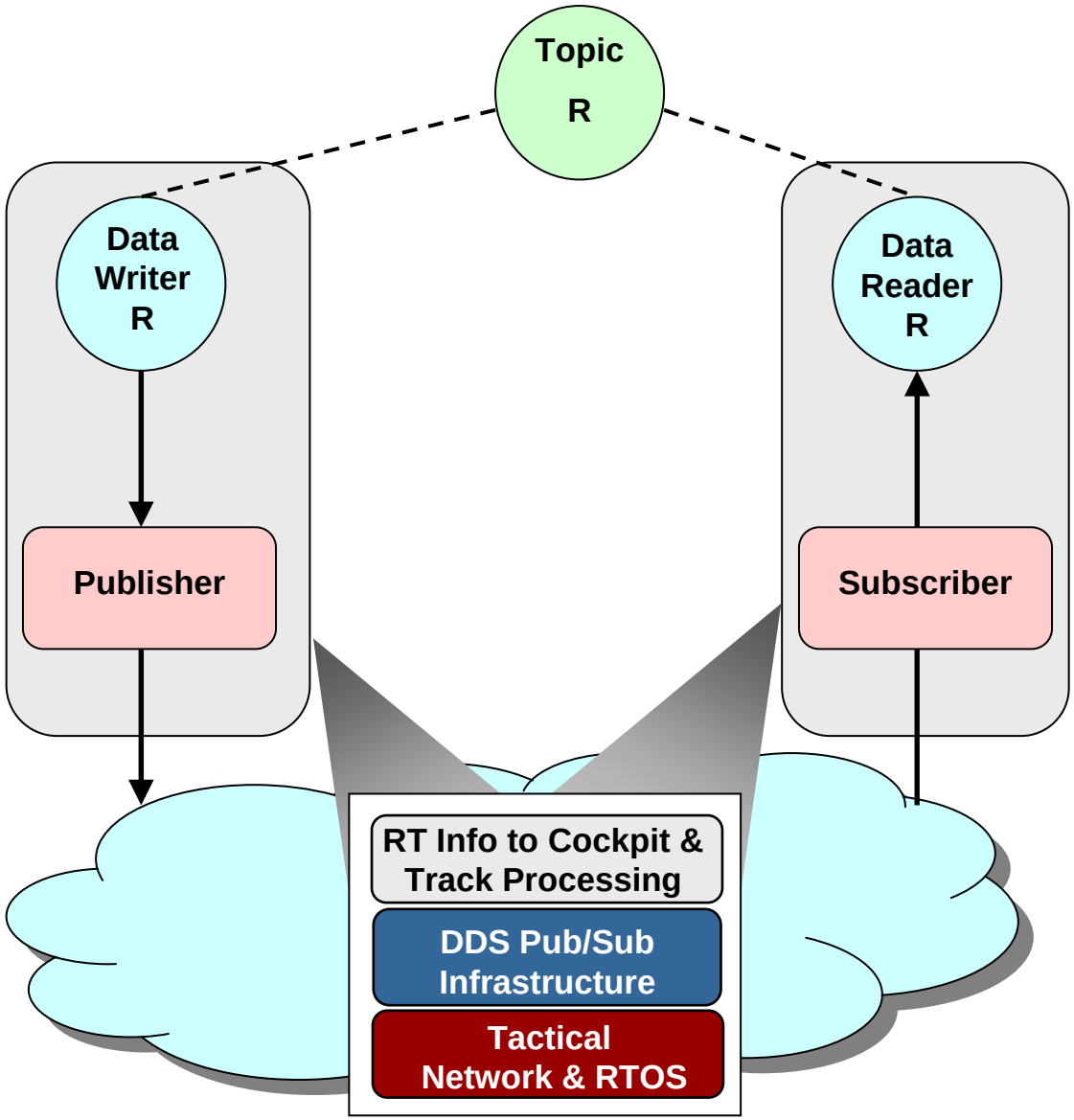


The Future of CORBA

- Improvements in CORBA features & performance
- Extensions to the CORBA object model
- **Complementary technologies**

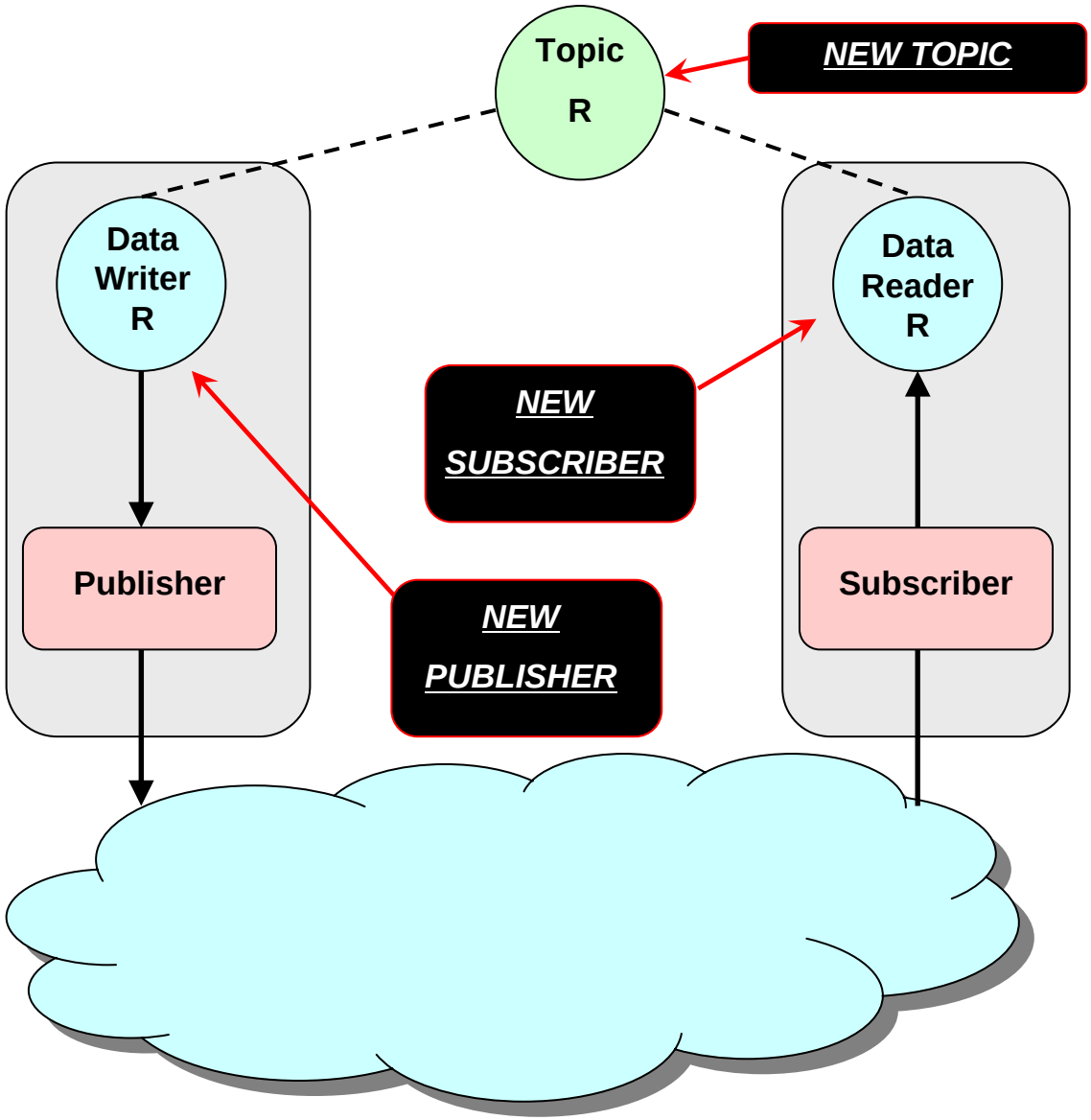
Overview of the Data Distribution Service (DDS)

- DDS is an highly efficient OMG pub/sub standard
  - e.g., fewer layers, less overhead



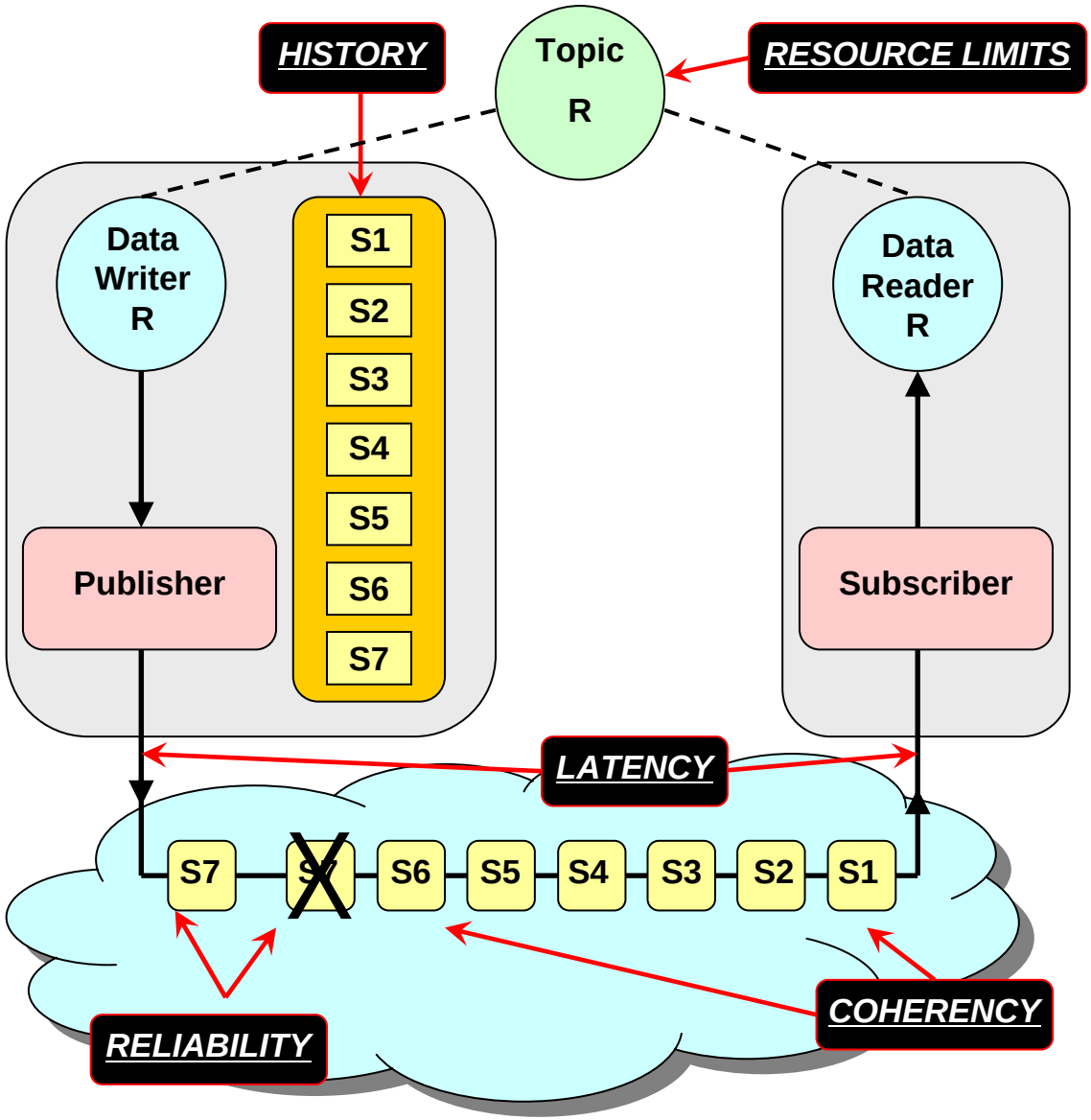
Overview of the Data Distribution Service (DDS)

- DDS is an highly efficient OMG pub/sub standard
  - e.g., fewer layers, less overhead
- DDS provides meta-events for detecting dynamic changes



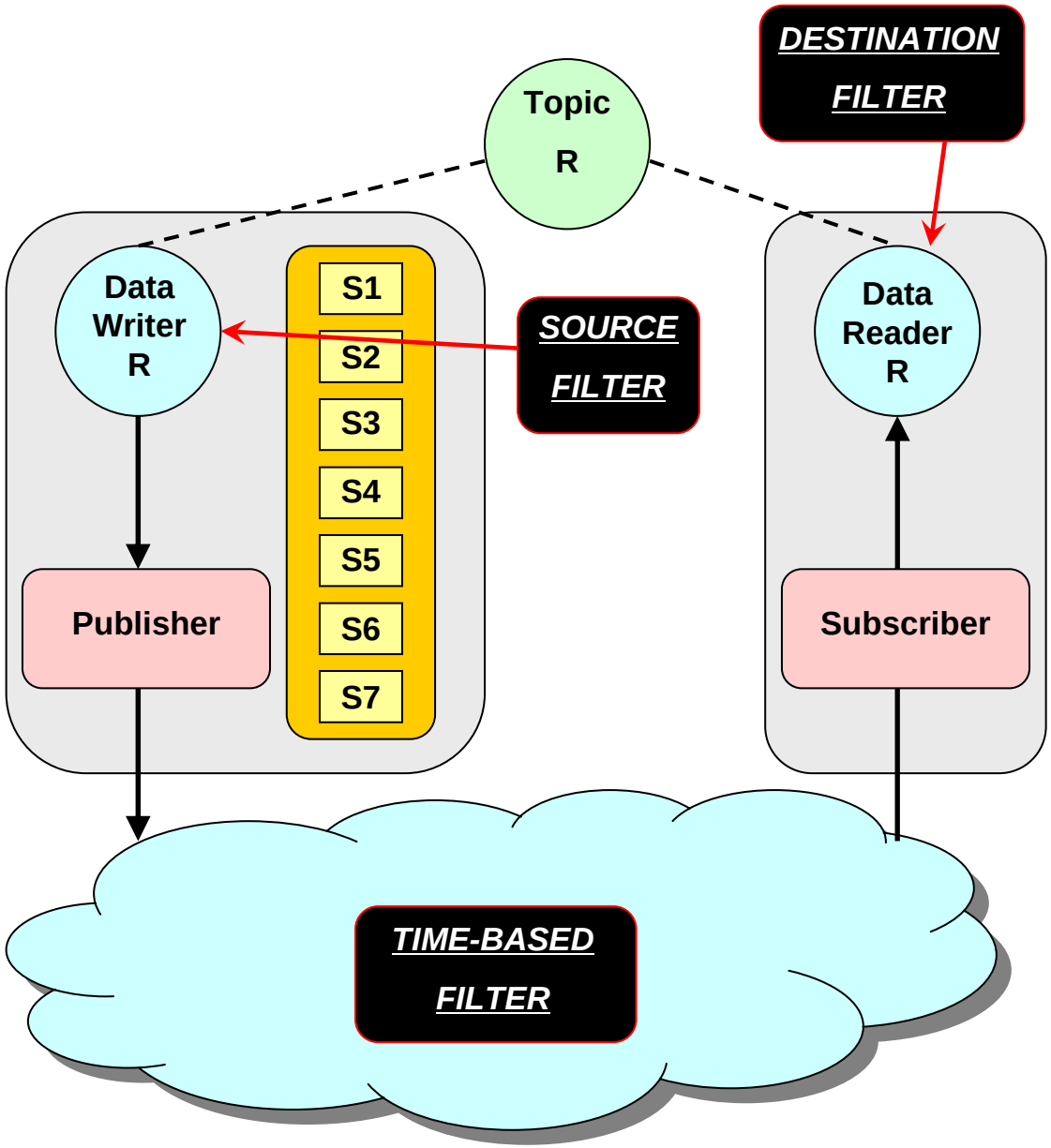
Overview of the Data Distribution Service (DDS)

- DDS is an highly efficient OMG pub/sub standard
  - e.g., fewer layers, less overhead
- DDS provides meta-events for detecting dynamic changes
- DDS provides policies for specifying many QoS requirements of tactical information management systems, e.g.,
  - Establish contracts that precisely specify a wide variety of QoS policies at multiple system layers



Overview of the Data Distribution Service (DDS)

- DDS is an highly efficient OMG pub/sub standard
  - e.g., fewer layers, less overhead
- DDS provides meta-events for detecting dynamic changes
- DDS provides policies for specifying many QoS requirements of tactical information management systems, e.g.,
  - Establish contracts that precisely specify a wide variety of QoS policies at multiple system layers
  - Move processing closer to data



Concluding Remarks

- Software industry is heavily driven by “fads”
  - i.e., “Teen-age boy band” syndrome

- CORBA is no longer the new kid on the block
  - In fact, it has a lot of facial hair, much of it gray ;-)

- With maturity comes certain virtues
  - High performance & integration with many platforms, languages, & technologies