PrismTech
Productivity Tools & Middleware

DDS TUTORIAL

For OMG RTE workshop July 2007

V3

Hans van 't Hag, DDS product Manager
hans.vanthag@prismtech.com

Agenda

- **Data-centric Foundation**
  - Into the mood
  - Net-centric Future: ‘the data is the network’
  - The ‘information-centric approach’
- **Driving the Standard**
  - Naval Combat Systems (CMS) example
  - The OMG DDS specification

<< 10 min. break >>

- **DDS ‘By Example’**
  - DDS profiles
  - Corba Integration
- **Concept Demo**
  - DDS Chatroom ‘concept’ demo

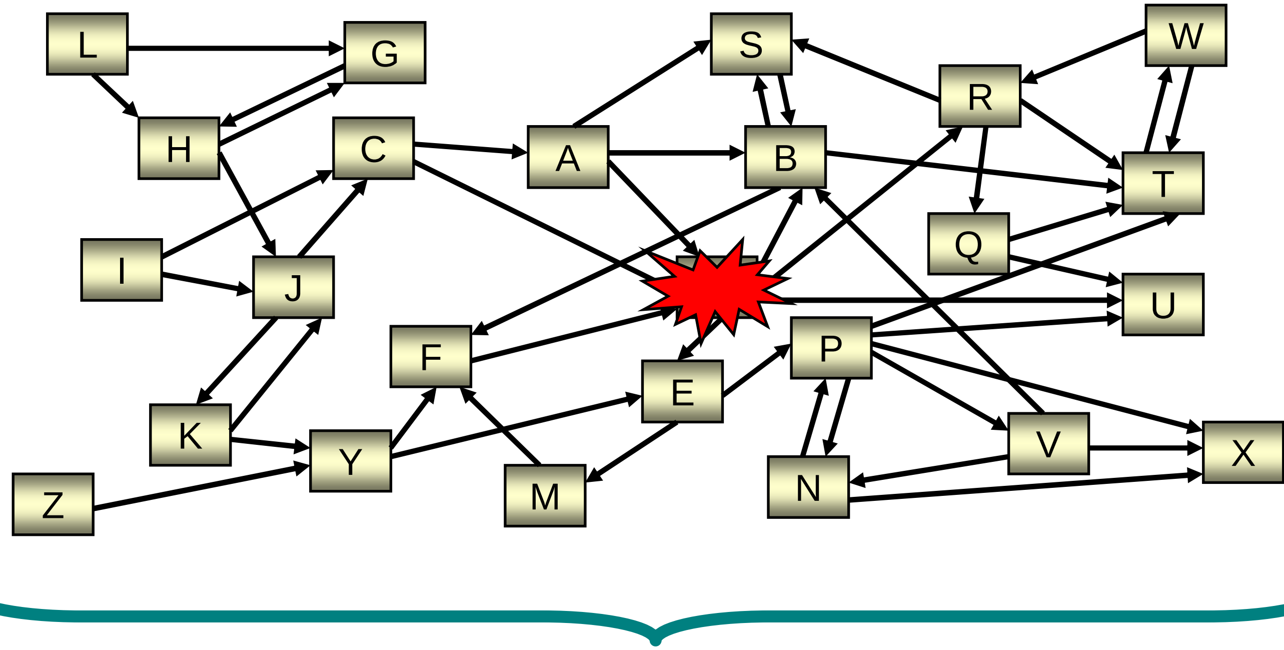
Into the mood (1): 'huh ?...'

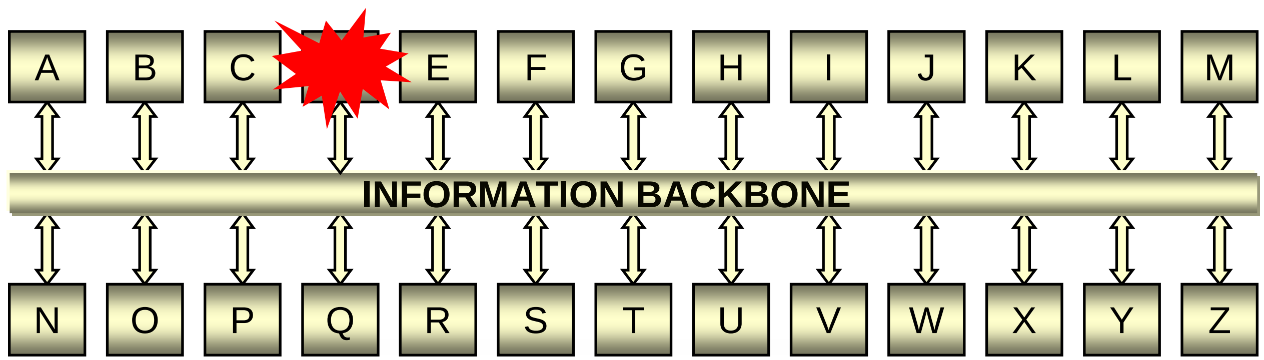
Into the mood (2): *'The Problem…'*

Into the mood (3): *'A scalable solution?...'*

PrismTech
Productivity Tools & Middleware

Into the mood (4): *'…Client/server vs. Pub-sub: A mind-shift…'*



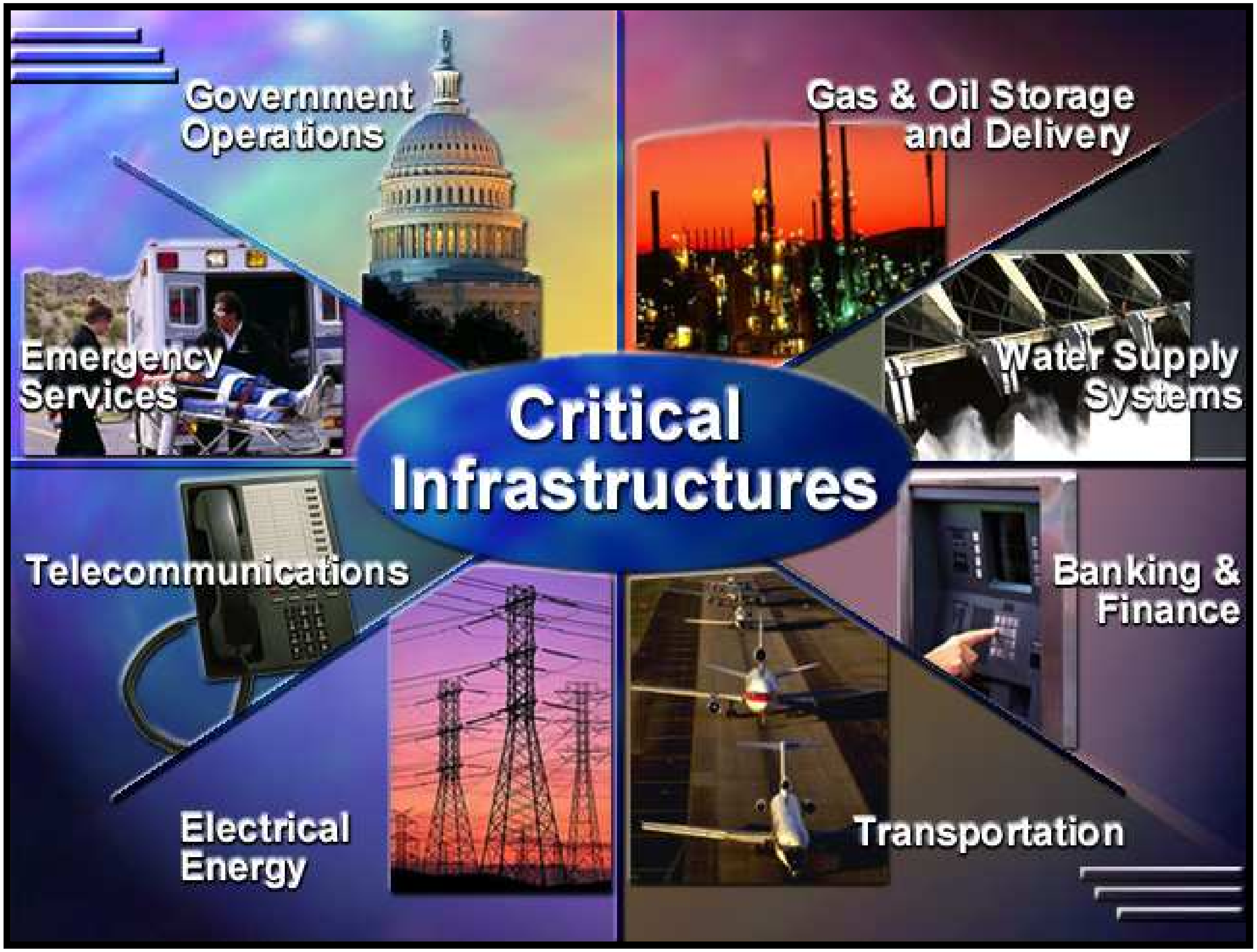


PRISMTECH
Productivity Tools & Middleware

PrismTech
Productivity Tools & Middleware

Net-centric Future:
'The Data is the Network '

© ARMS (Doug Schmidt): R&D Challenge

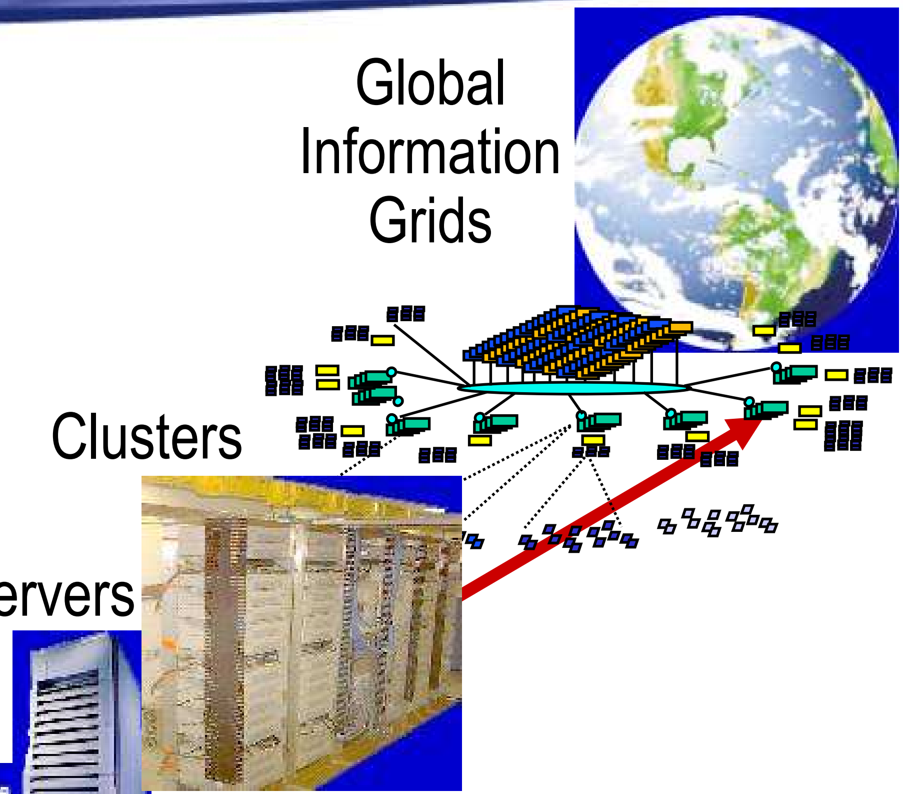


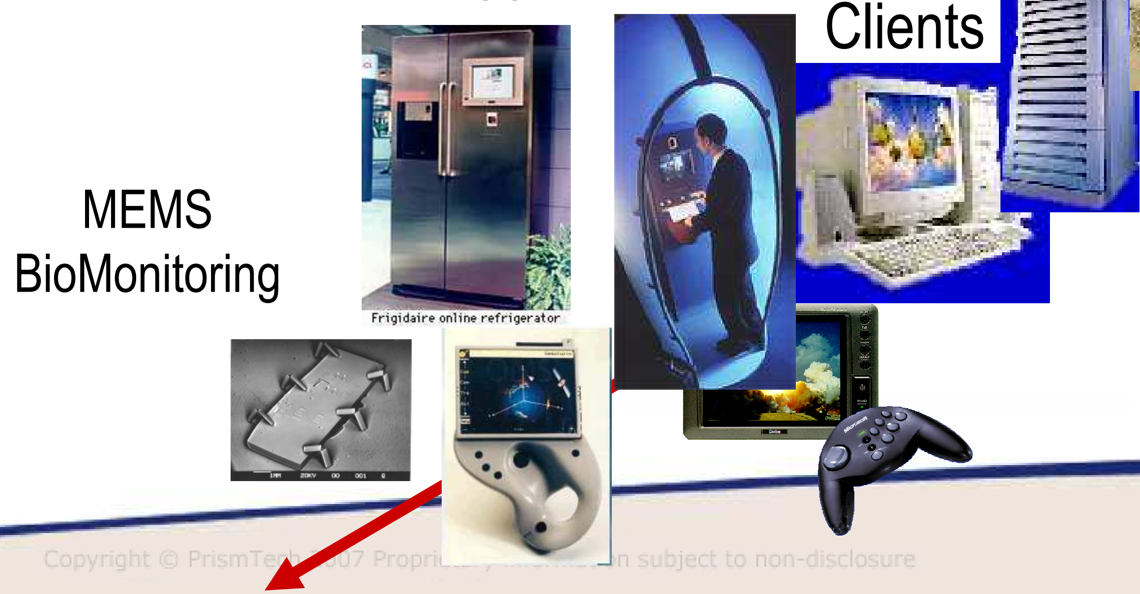
PRISMTECH
Productivity Tools & Middleware

© ARMS (Doug Schmidt): R&D Challenge

Develop DRE Systems Technologies to Protect Critical Infrastructure

The “soft underbelly” of commercial, military, & infrastructure DRE systems depend increasingly on **information technology**, making attacks both attractive & lethally effective





R&D Challenges: Create
- efficient,
- scalable,
- reliable,
- secure,
- & predictable

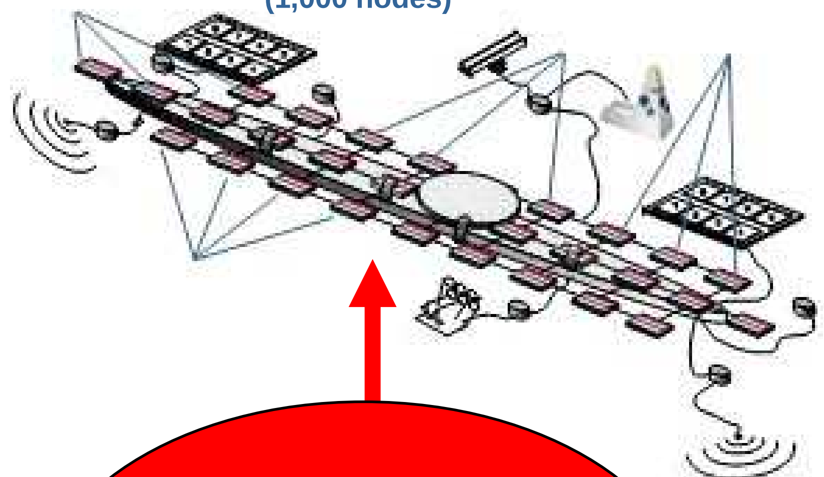
DRE system technologies from nano- to tera-scale

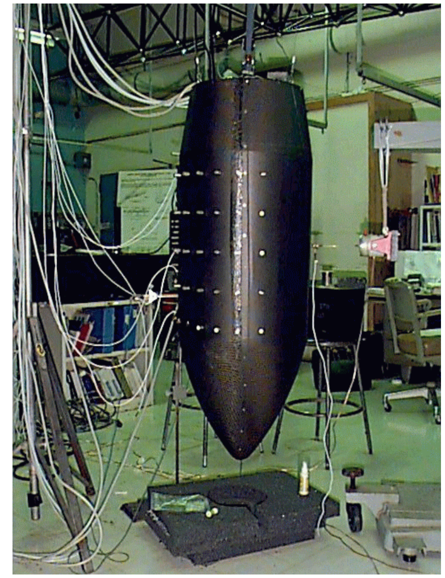
© ARMS (Doug Schmidt):
R&D Challenge: Current challenges and limitations = “Scalability” !

Emerging Trends

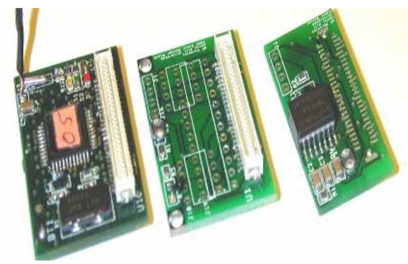
- Large-scale DRE system requirements are increasingly more
  - *Dynamic*
  - *Diverse*
  - *Demanding*

Total Ship Computing Environments: Global dynamic resource management (1,000 nodes)



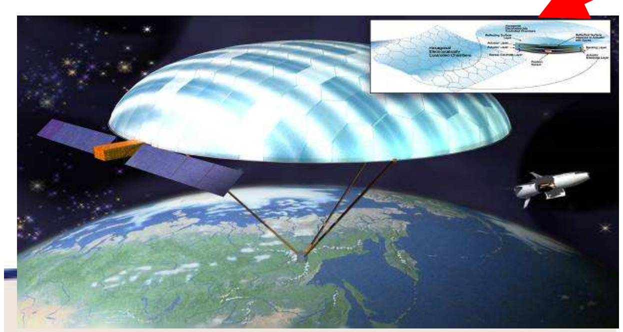


Distributed Active Control: Vibration Damping on Delta-4 Rocket Payload Fairing (1,000 nodes)

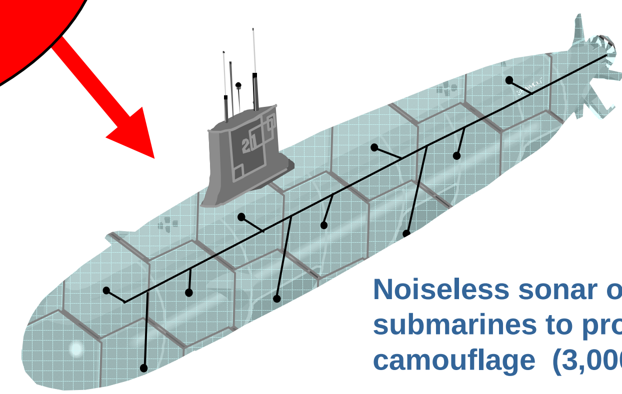


Distributed Network of embedded sensor motes for environment monitoring, tracking, & surveillance (10,000 nodes)

1000 – 1,000,000 node fusion of physical & information systems



Gossamer Space Antenna (1,000,000 nodes)



Noiseless sonar on submarines to provide camouflage (3,000 nodes)

Net-centricity: *The Data Is The Network™* - I

Leverage the Power of Information

NET-CENTRICITY:

People, processes, and technology working together to enable timely and trusted:

- **ACCESS** to information
- **SHARING** of information
- **COLLABORATION** among those who need it

Can Only Be Done on The Net!

Connecting People With Information 4

The Move to Net-Centricity

Current		**Net-Centric**
Information stovepipes	→	Shared information
"Welded" interfaces	→	Unconstrained
Predetermined needs	→	Unanticipated users
Fixed display formats	→	User-defined info and formats
Need to know	→	Need to share; right to know
Rigid		**Agile**

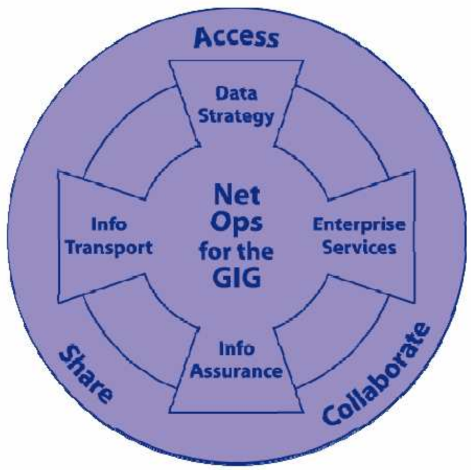
Connecting People With Information 14
01NOV05/0053

© DoD Chief Information Officer, Network and Information Integration, Nov'05

Net-centricity: *The Data Is The Network™* - III

CIO/NII
Enabling Net-Centric Operations

Net-Centric Framework



- **Data Strategy:**
  - *How to "**share**" the data*
- **Enterprise Services:**
  - *How to "**access**" the data*
- **Information Transport:**
  - *How to "**move**" the data*
- **Network Operations:**
  - *How to **"operate and defend"** the GIG*
- **Information Assurance:**
  - *How to keep it all "**dependable**"*

Data: Discoverable, Accessible, Understandable

Connecting People With Information 8
01NOV05/0050

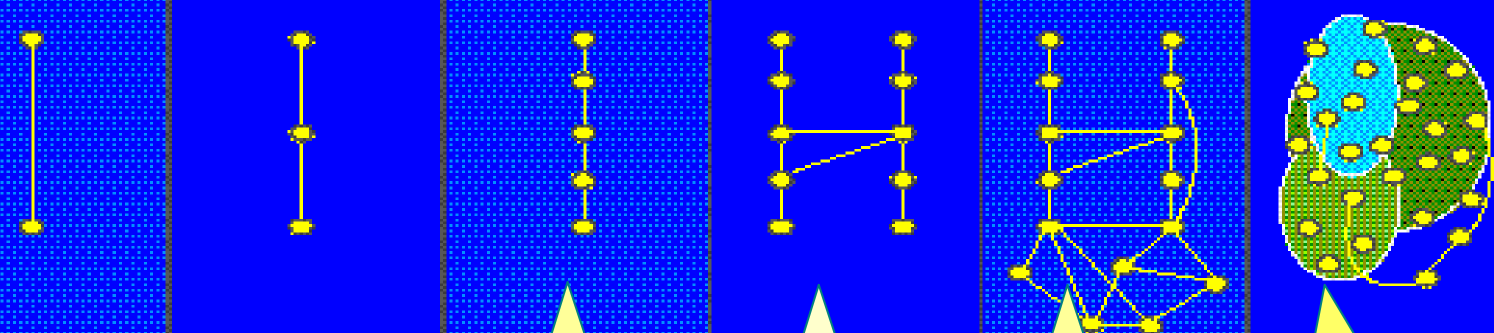
PRISMTECH
Productivity Tools & Middleware

Productivity Tools & Middleware

The Information-centric Approach

TRENDS (© SUN) 14

	Client-Server	3/N-Tier	Net Apps	Net Services	Next	After that
Catch Phrase	The Network is the computer	Objects	Legacy to the Web	The Computer Is the Network	Network of embedded things	Network of Things
System / Collections / Components						
Scale	100s	1000s	1000000s	100000000s	1000000000s	10000000000s
When/Peak	1984/1987	1990/1993	1996/1999	2001/2003	1998/2004	2004/2007
Leaf Protocol(s)	X	X	+HTTP (+JVM)	+[illegible]	[illegible]	Unknown
Directory(s)	NIS, NIS+	+ CDS	+ LDAP (*)	+UDDI	+ Jini	+ ?
Session	RPC, XDR	+CORBA	+CORBA, RMI	+ SOAP, XML	+ RMI/Jini	+ ?
Schematic						



“Information - Backbone” (DDS)

Corba

Web

Java

Information Grids

Dr. Richard Soley (OMG Chairman & CEO):

"The DCPS publish/subscribe model stands as a natural complement to the object-centric client/server model provided by CORBA"

(Consumer Electronics, September 13, 2004)

Doc Allen (OMG Co-Chair & Mitre):

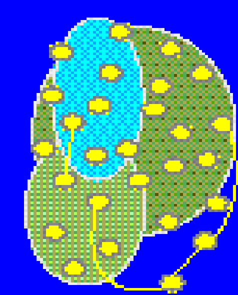
"DDS clearly has the potential to become 'THE' dominant (real-time) standard in the Net-centric environment"

(OMG-RTESS plenary meeting June'04)

Dave Sharp (Boeing FCS chief-architect):

"The (Army) SOSCOE environment will be based on MDA/UML and OMG-DDS"

(OMG sponsor-presentation, June'04)

	After that
Catch Phrase	Network of Things
System Collections Components	
Scale	10000000000s
When/Peak	2004/2007
Leaf Protocol(s)	Unknown
Directory(s)	+ ?
Session	+ ?
Schematic	

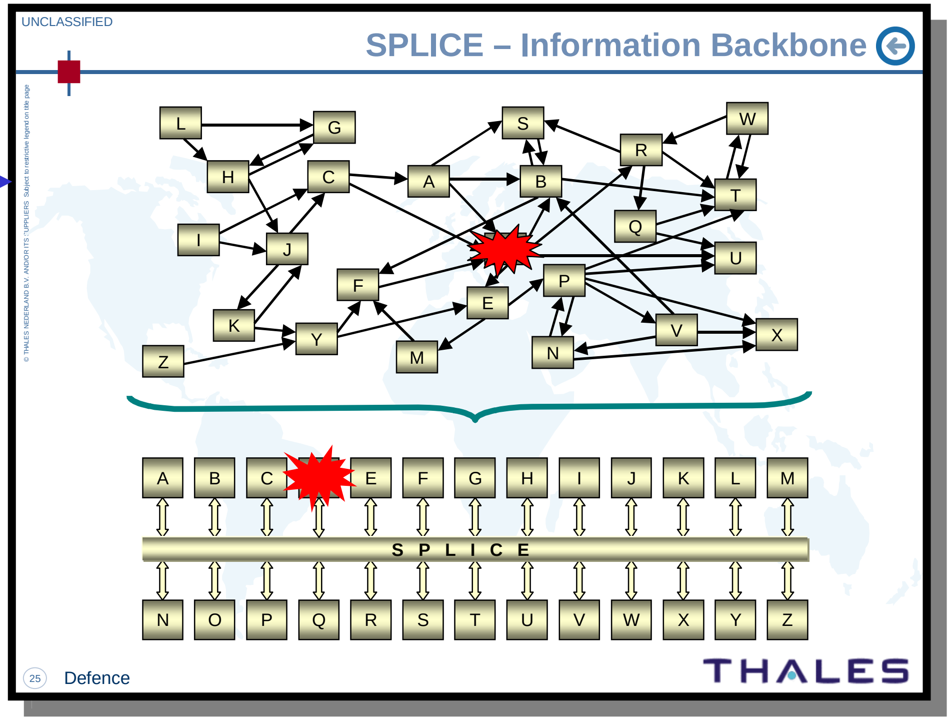
Information Grids

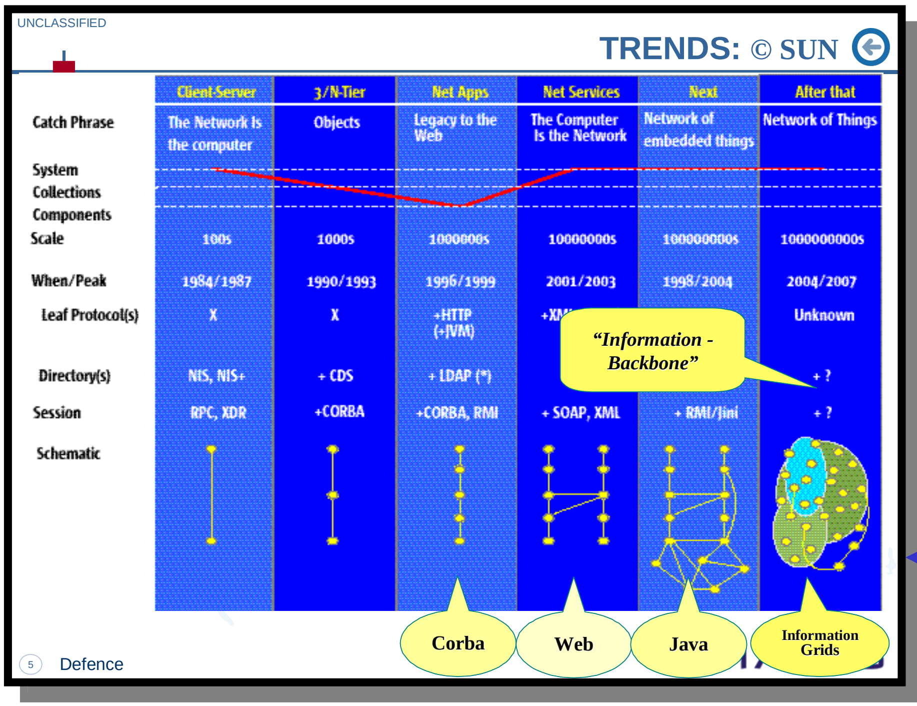
Java

BACKGROUND & TRENDS

The Idea: Reduced complexity

- pub/sub already patented in 1987
- information backbone
- "Right info, Right place, Right time"





Towards an 'information-centric' world

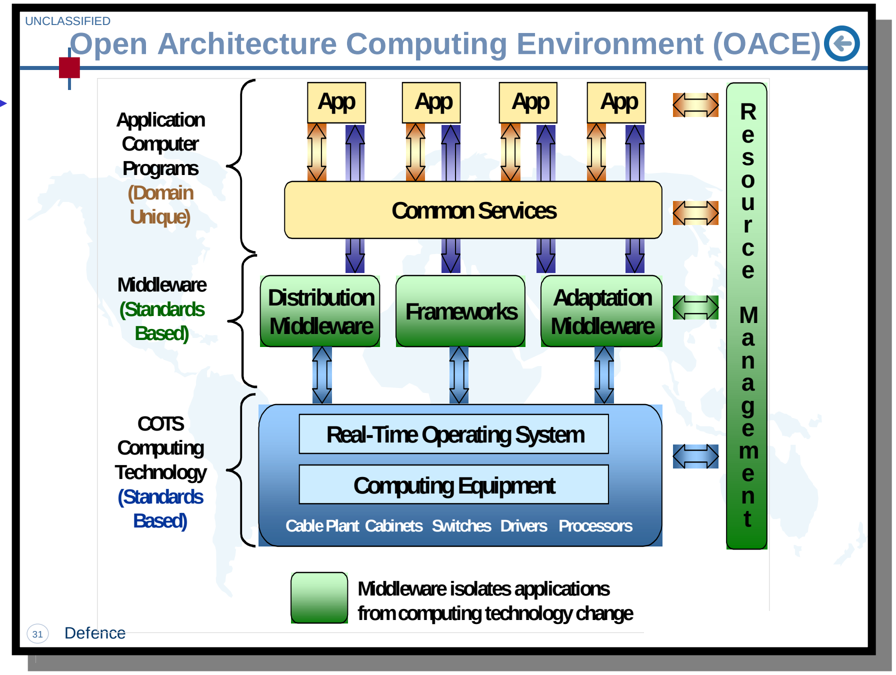
- Loosely coupled components
- Dynamic systems
- Traditional architectures don't suffice

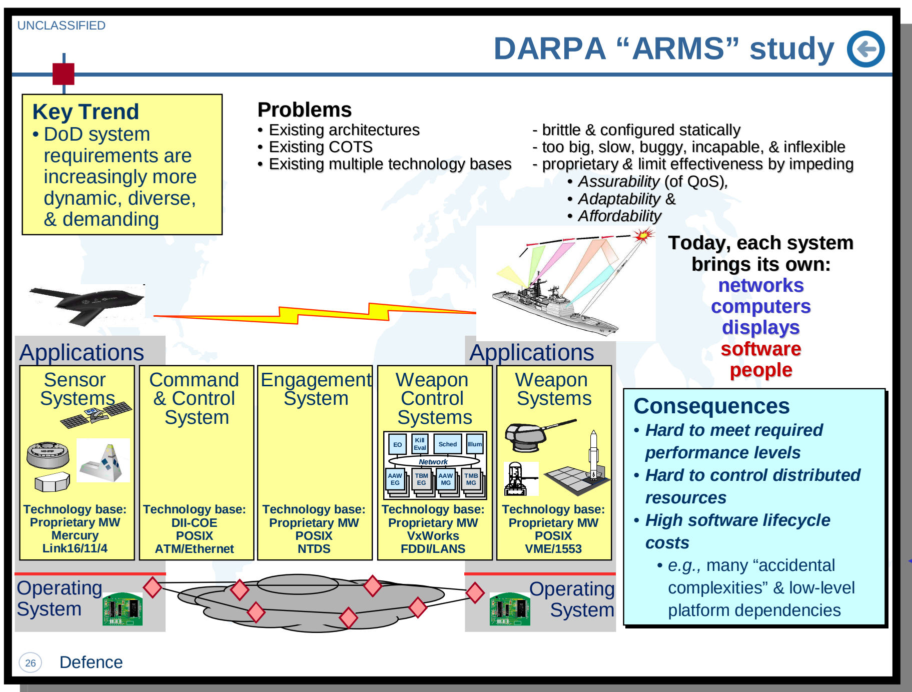
ARCHITECTURE TRENDS: *DDS, A Major Specification in Line* with the Network Centric Warfare Paradigm

DDS mandated for US - OACE

- DDS is key for success in NAVY OA
- Evaluated since 2004

"NDDS/SPLICE provide good performance and scalability as a publish-subscribe middleware for combat system applications"
(NSWC-DD, OA Technical assessment 07/2004)





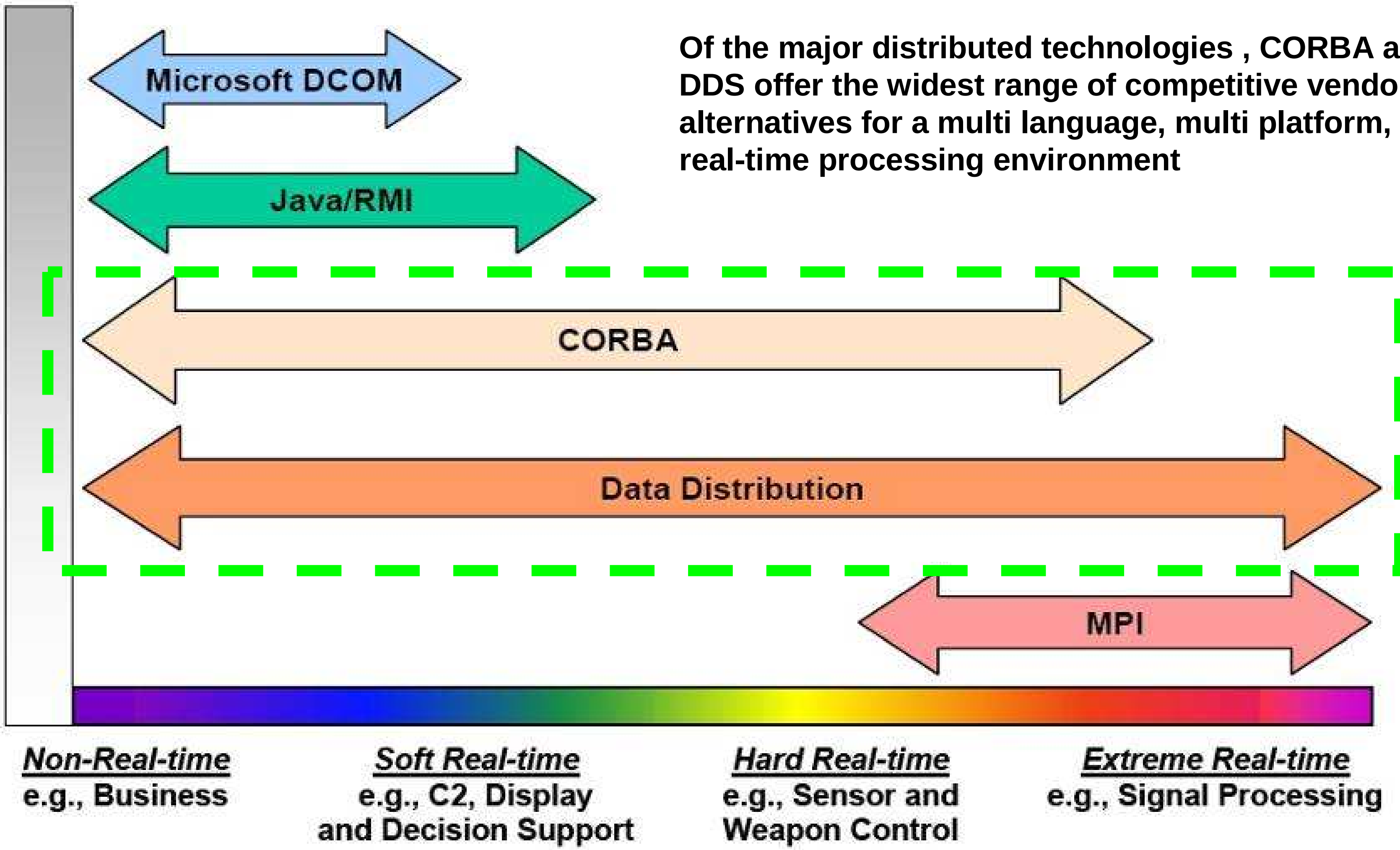
"The finalization and availability of the DDS specification really is a tremendous achievement that addresses a significant need in both government and civilian sectors"

(Dr. Richard Soley OMG Chairman, Consumer Electronics, September 13, 2004)

Recognized potential

- DARPA recognizes DDS importance
- Dynamic Resource Management potential

Distribution Middleware Requirements for OACE

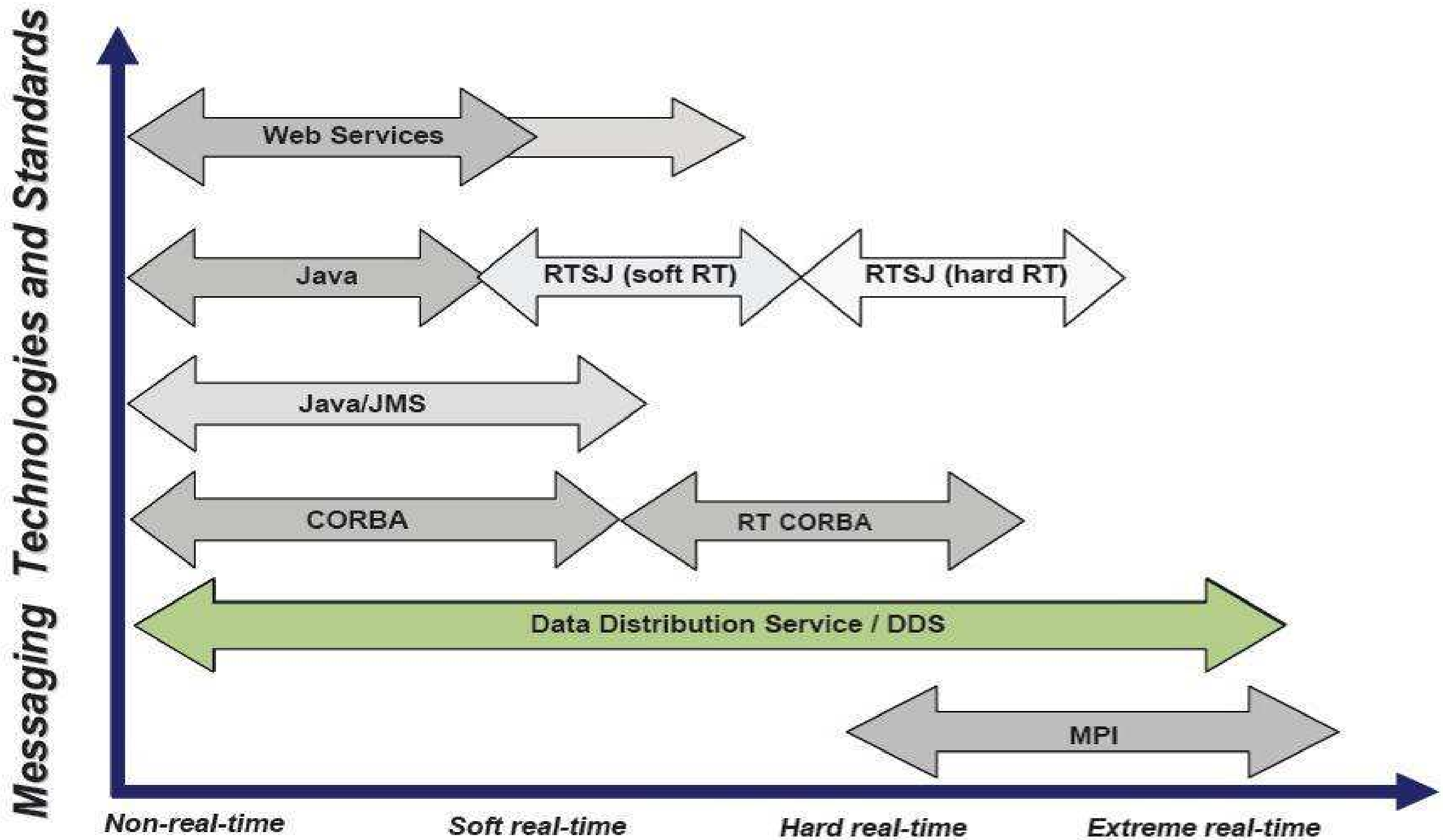


Of the major distributed technologies , CORBA and DDS offer the widest range of competitive vendor alternatives for a multi language, multi platform, real-time processing environment

Support for Real-time QoS is essential

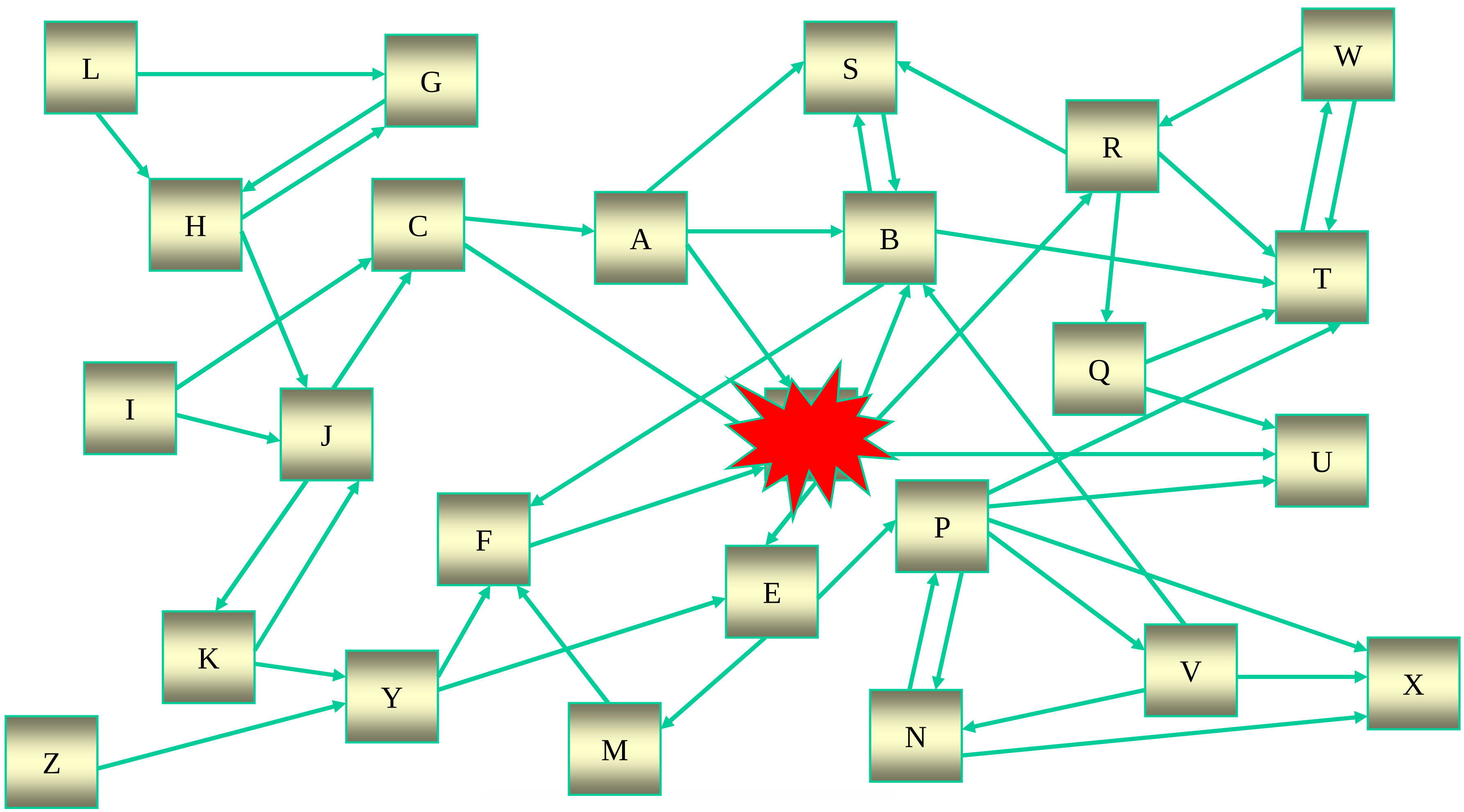
DDS APPLICABILITY (from US OACE documentation)

Data-Distribution and Real-Time

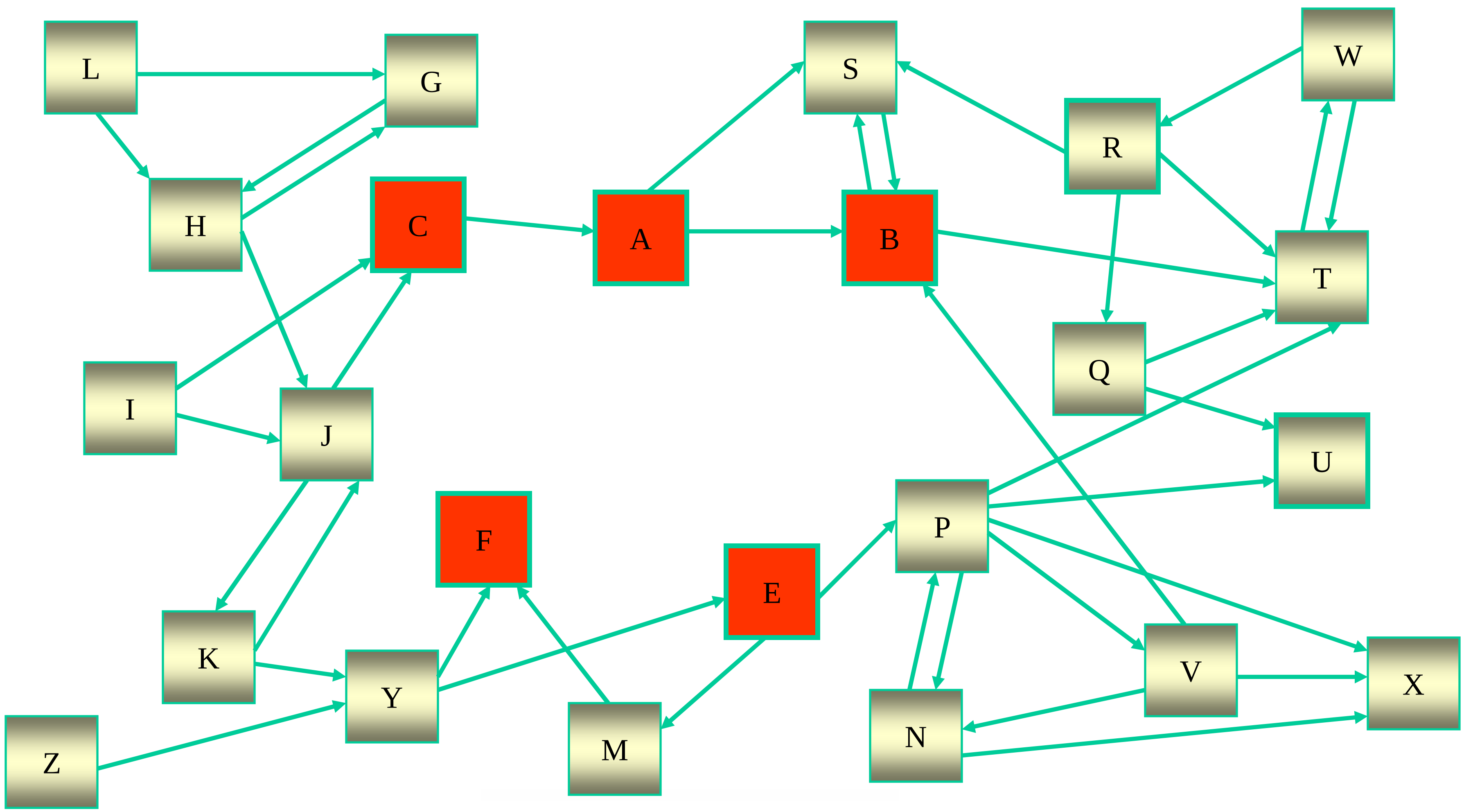


PRISMTECH
Productivity Tools & Middleware

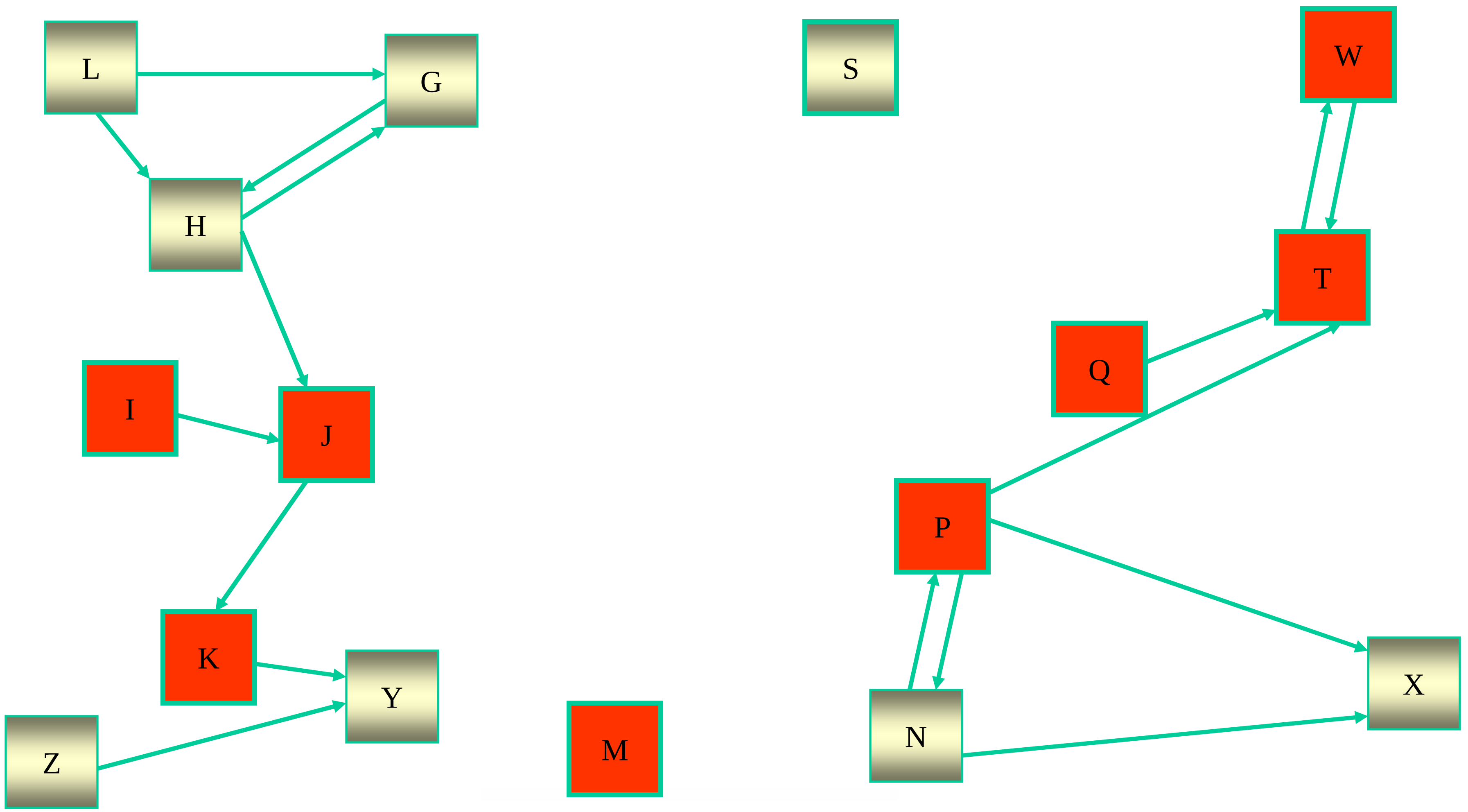
Traditional Architectures: fault propagation



Traditional Architectures: fault propagation

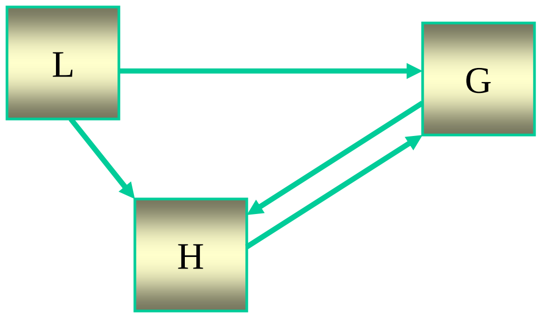


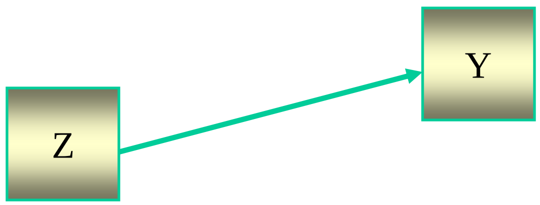
Traditional Architectures: fault propagation



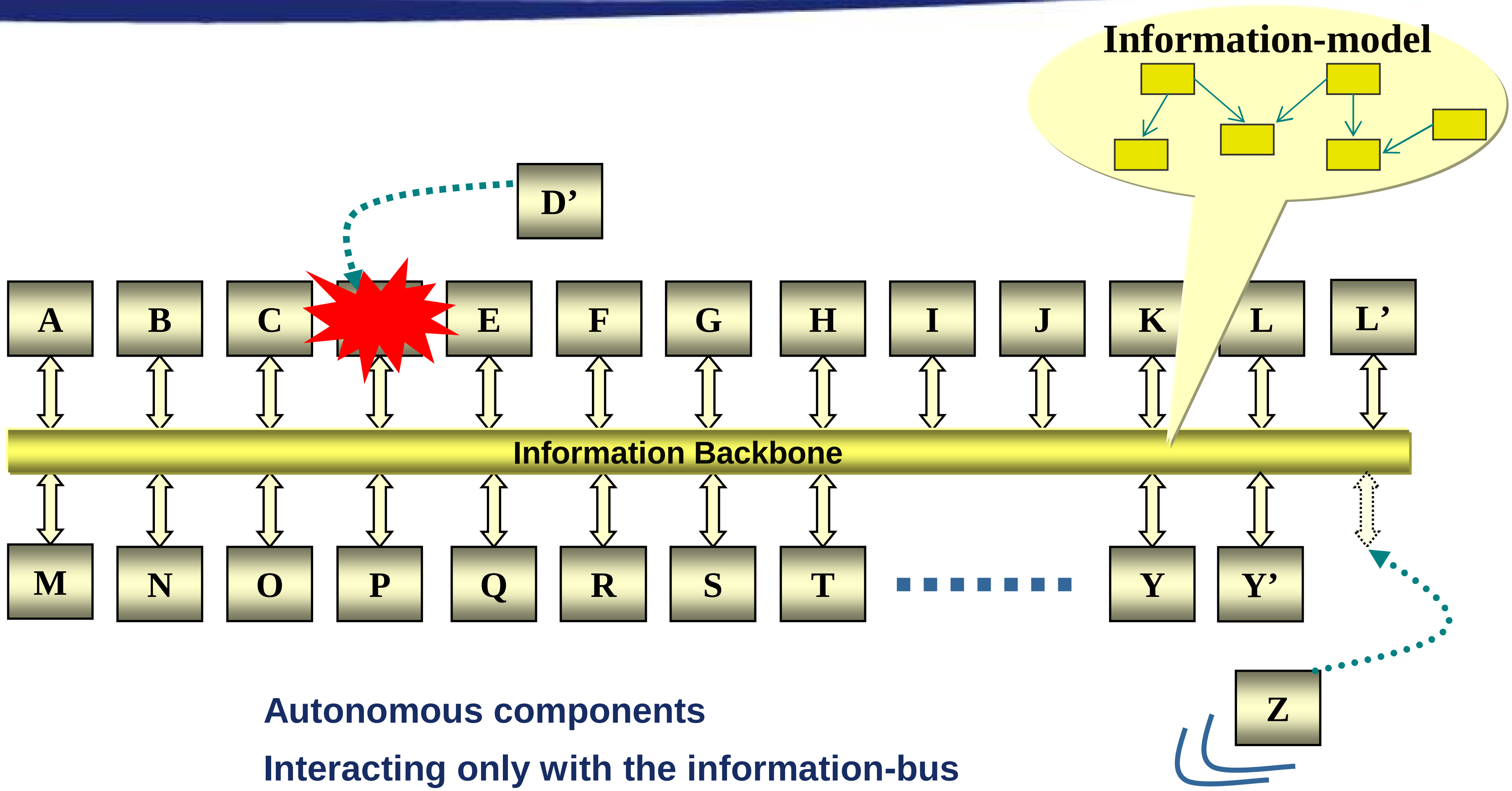
PRISMTECH
Productivity Tools & Middleware

Traditional Architectures: fault propagation





DDS: AN INFORMATION-CENTRIC ARCHITECTURE



Autonomous components

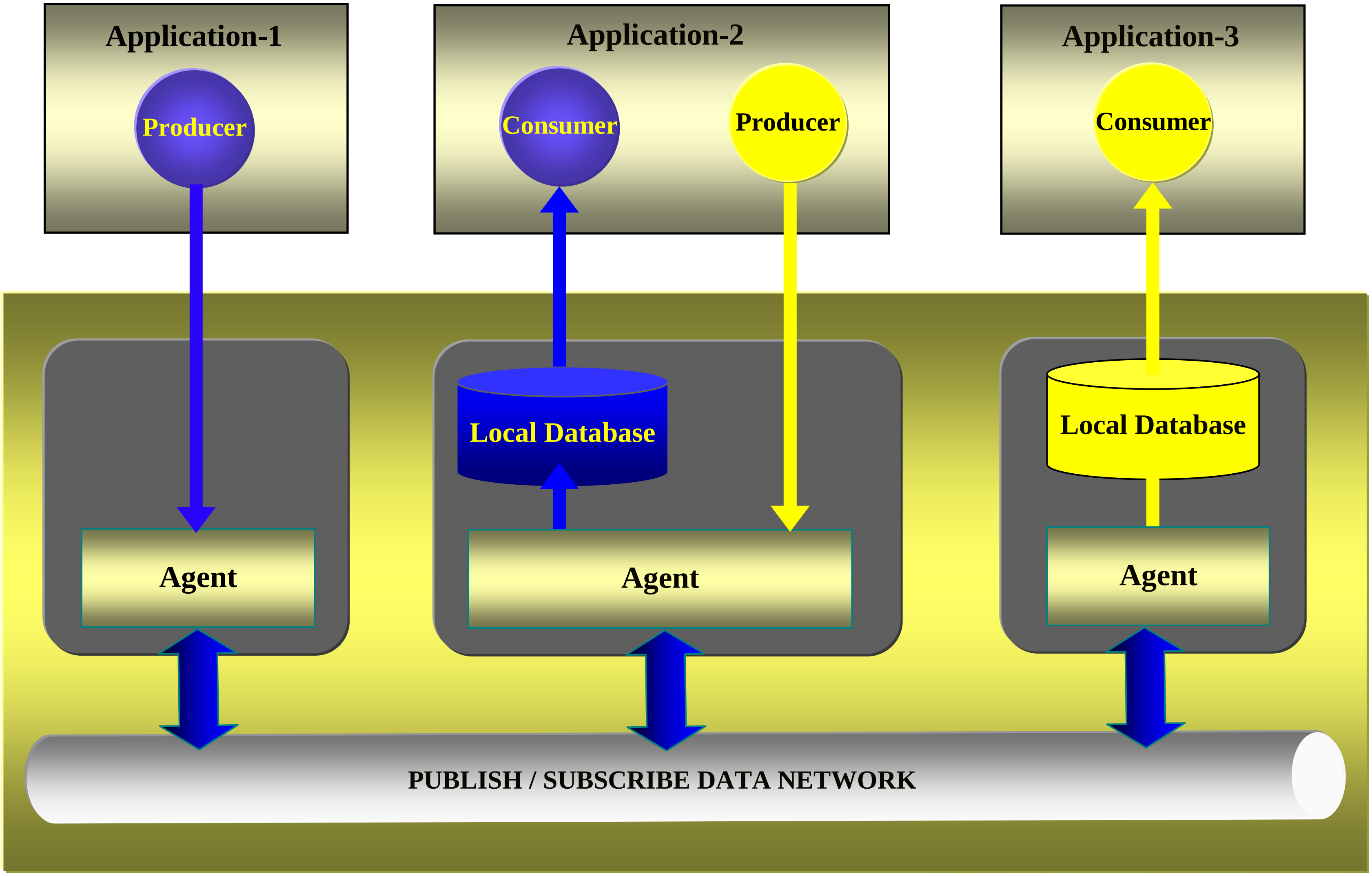
Interacting only with the information-bus

Spontaneous: Z , Self-healing: D’

Redundant & Replicated: L’ , Y’

QOS-driven Data Distribution Service (reliability, urgency, importance): DDS

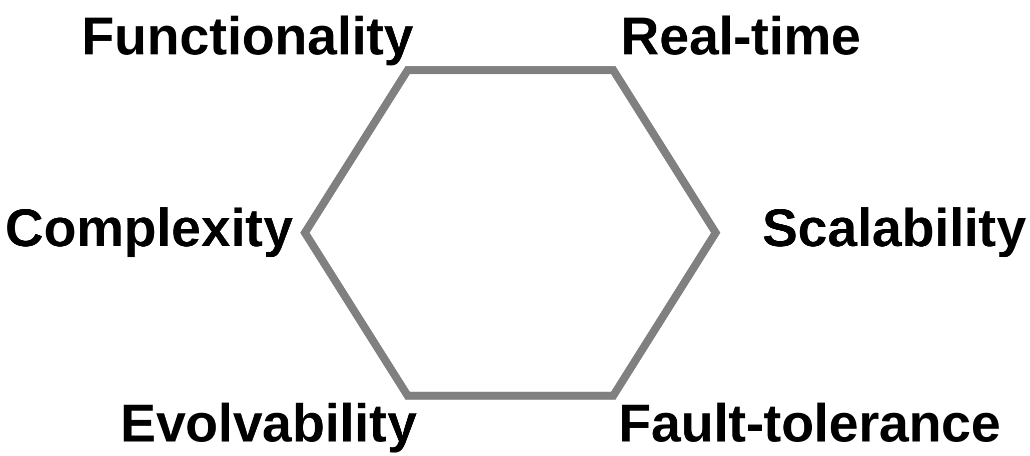
INFORMATION BACKBONE: “Under the hood”



PRISMTECH
Productivity Tools & Middleware

Pub/Sub: ‘Keep it simple stupid’

Distributed Real-time Systems Engineering



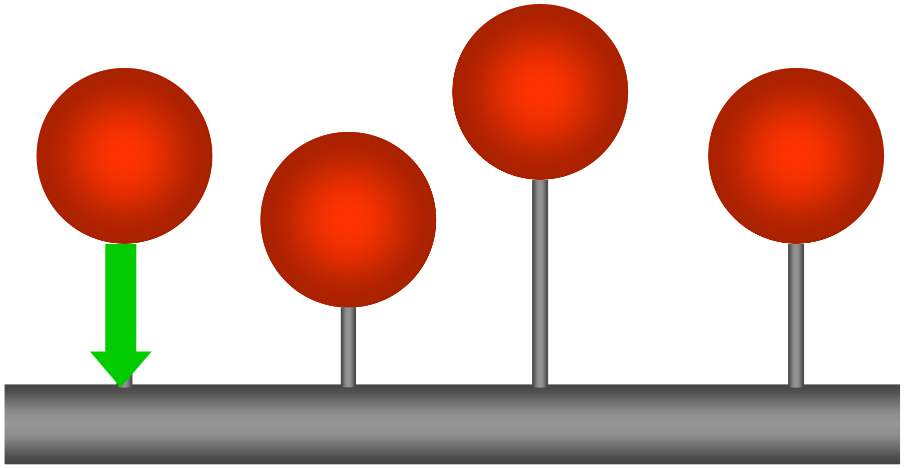
Many different types of requirements

OMG-DDS fundamentals

- **Design principles:**
  - Minimize dependencies between components
  - Share stable properties
- **Focus on:**
  - Autonomous component behavior
  - Common information model
- **Middleware delivers:**
  - *“The right information at the right place at the right time”*

Architectural Requirements: Lifecycle focus

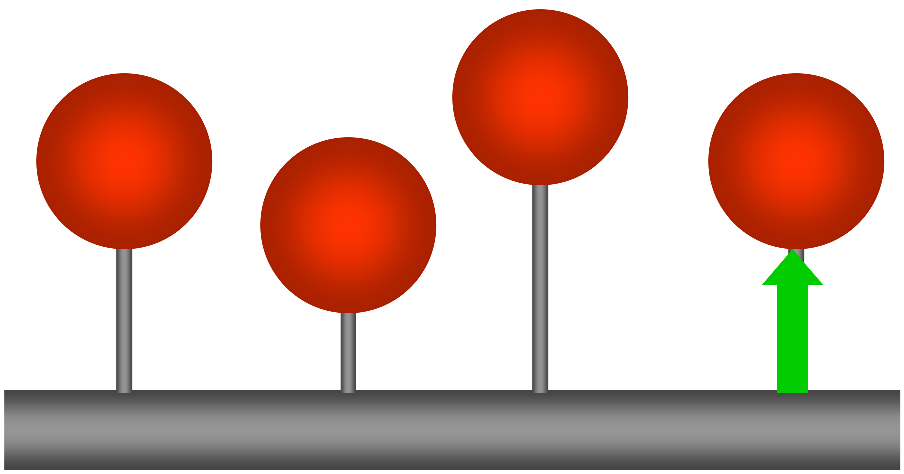
- System design
  - provide a stable basis to operate upon by applications
  - enhance component autonomy
  - allow transparent and global QoS assurance
- System development
  - reduce complexity and enhance re-usability
  - provide shared/guaranteed properties
  - small learning effort and flat learning curve
- System integration
  - support effortless component integration
  - provide easy monitor & control
  - shift ratio between design and integration effort
- System deployment
  - guaranty QoS for reliability, latency and persistency
  - allow runtime migration of applications
  - allow applications to join the system at any time
- System maintenance & evolution
  - allow runtime replacement and evolutionary upgrading
  - support for logging & replay of information
  - provide future-proof, re-usable, robust and scalable system



Decoupling in space and time

Built-in capabilities:

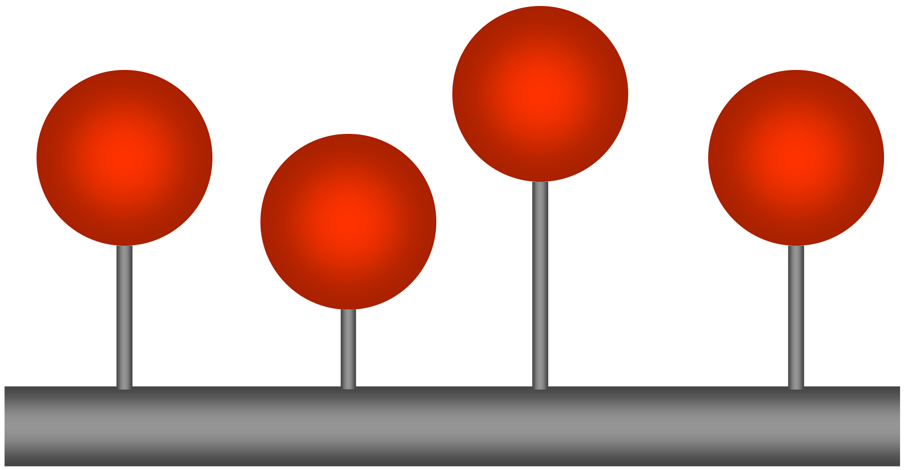
- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- information partitioning



Decoupling in space and time

Built-in capabilities:

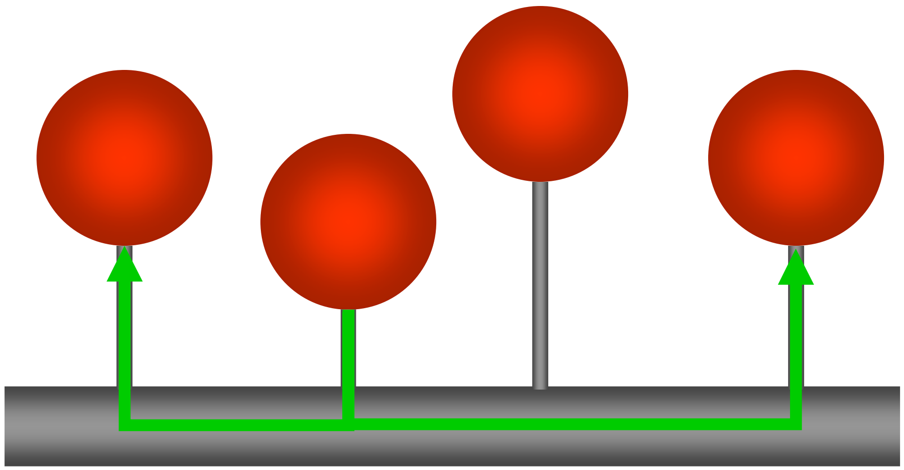
- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- information partitioning



Decoupling in space and time

Built-in capabilities:

- ➢ P/S data distribution
- ➢ relational data access
- ➢ data persistence
- ➢ dynamic (re-) configuration
- ➢ quality of service
- ➢ fault-tolerance
- ➢ information partitioning



Data is dynamically forwarded to all subscribed components

Built-in capabilities:

- **P/S data distribution**
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- information partitioning

Data Definition

- Applications subscribe to **TOPICS**
- Each **TOPIC** has an associated **name** and data **type**:
  - Data (type) definition in IDL
  - '**key**' fields for unique identification
  - more recent instances overwrite existing instances with same key value (keeping into account the 'history-depth' QoS setting of a subscriber)
- Example (IDL types):

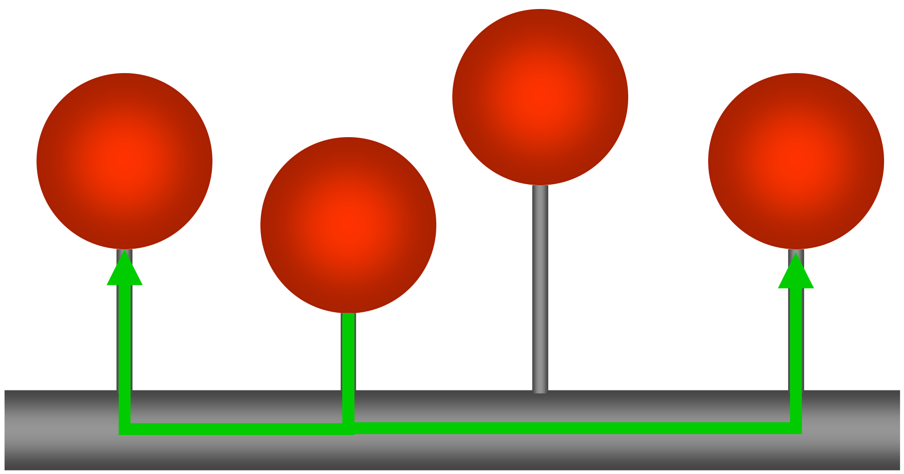
```
Struct PointTrack {
  long  source;        // key
  long  trackId;       // key

  Position pos;
}
```

```
Struct TrackState {
  long  source;        // key
  long  trackId;       // key

  long environment;
  long identity;
}
```


PRISMTECH
Productivity Tools & Middleware

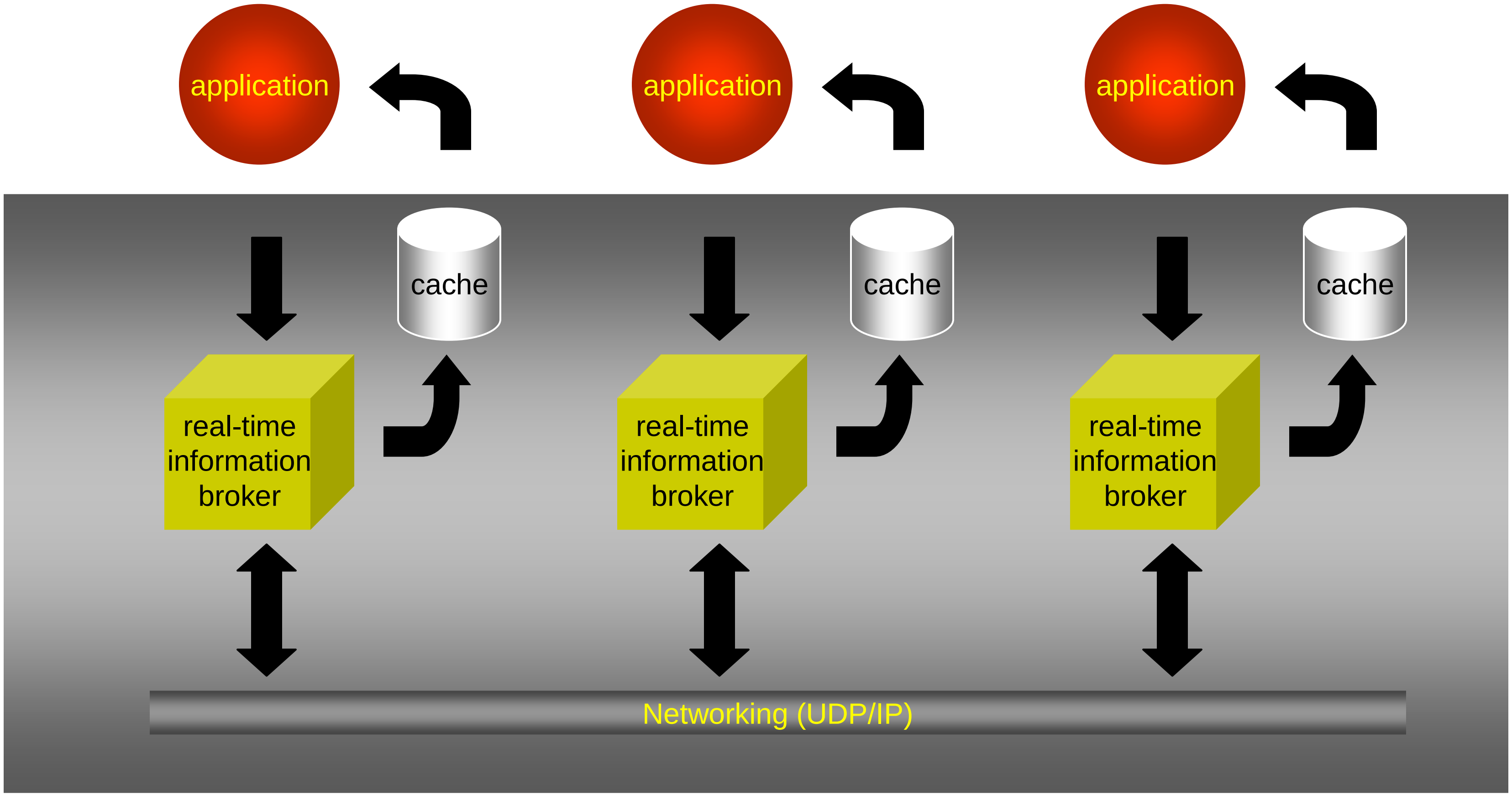


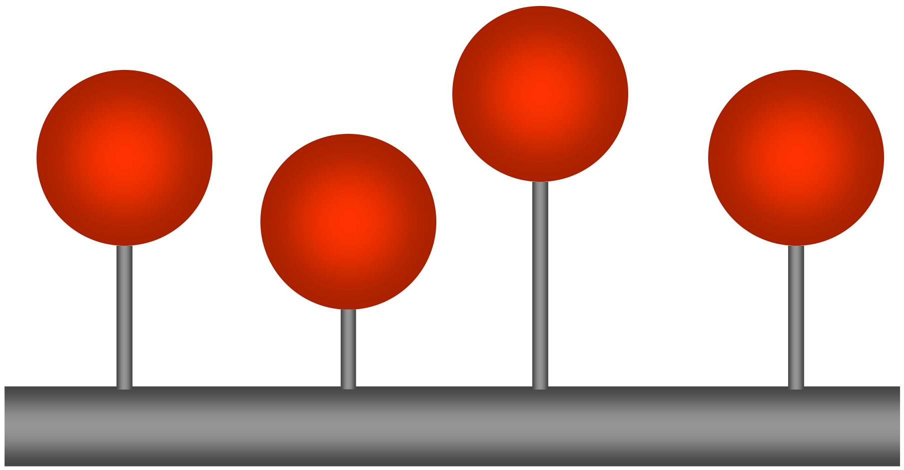
Data is dynamically forwarded to all subscribed processes

Built-in capabilities:

- **P/S data distribution**
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- information partitioning

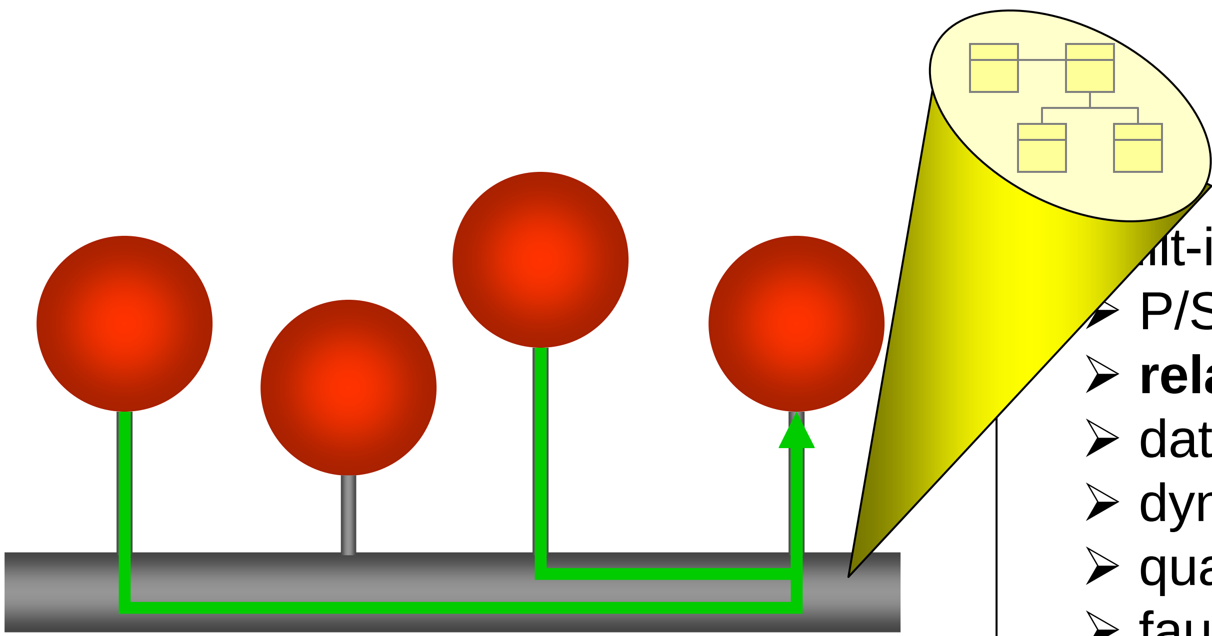
Realization





Built-in capabilities:

- **P/S data distribution**
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- information partitioning

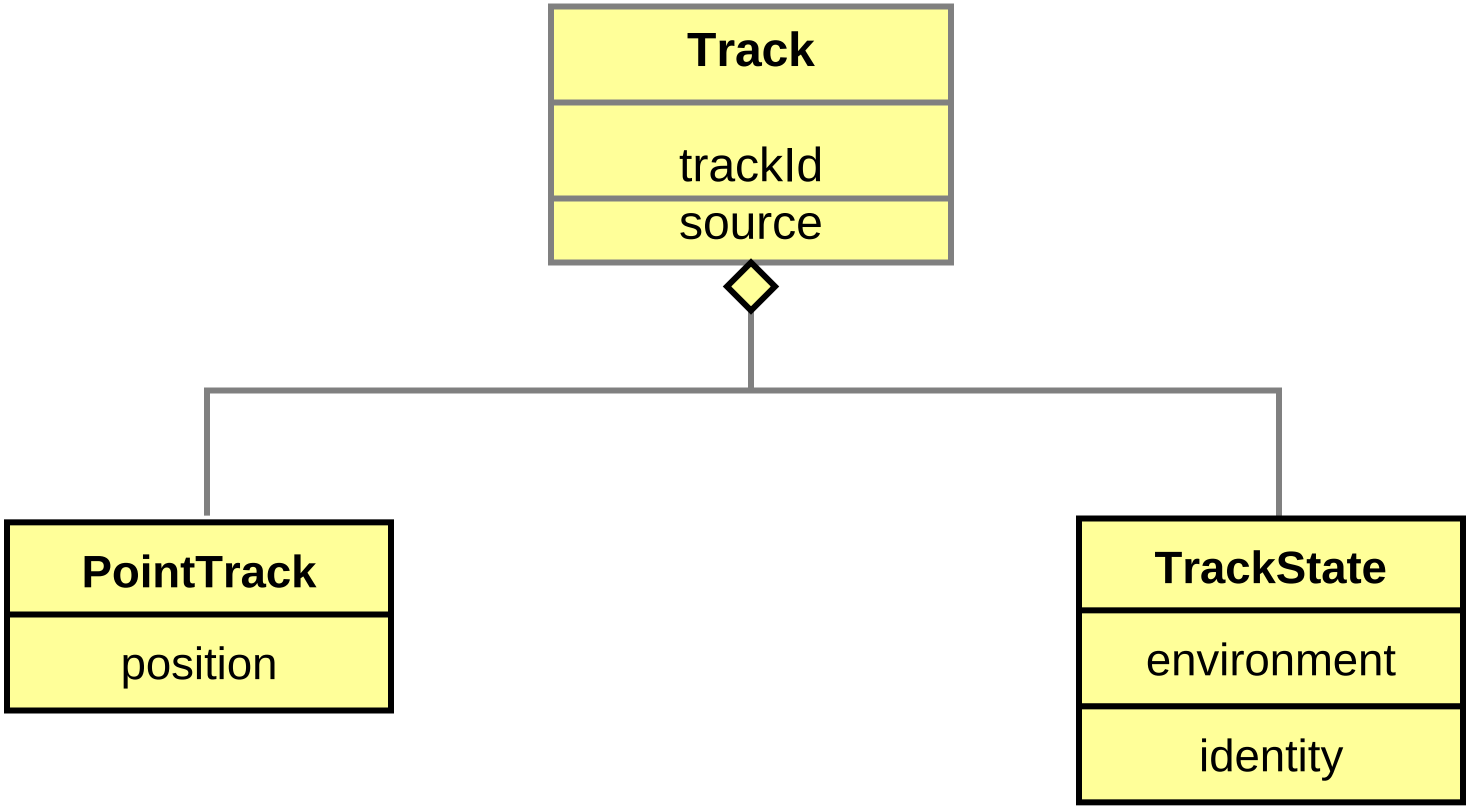


ilt-in capabilities:

- ➢ P/S data distribution
- ➢ **relational data access**
- ➢ data persistence
- ➢ dynamic (re-) configuration
- ➢ quality of service
- ➢ fault-tolerance
- ➢ information partitioning

Content-based subscription through relational views

Relational Object Models



Content-based Subscription

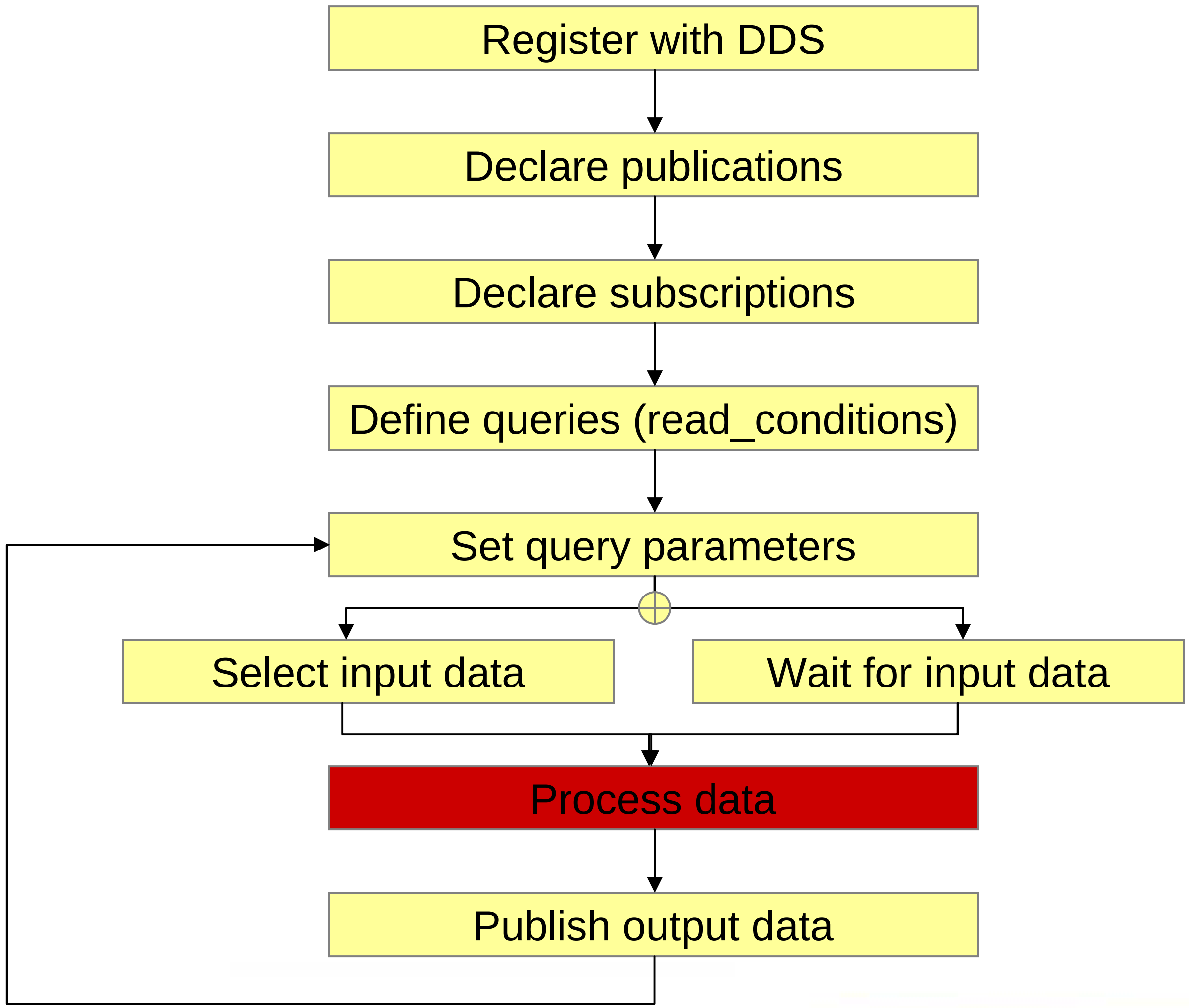
- Data filtering in DDS, e.g. using SQL:

```
select * from TrackPosition
where position.range < 10000
```

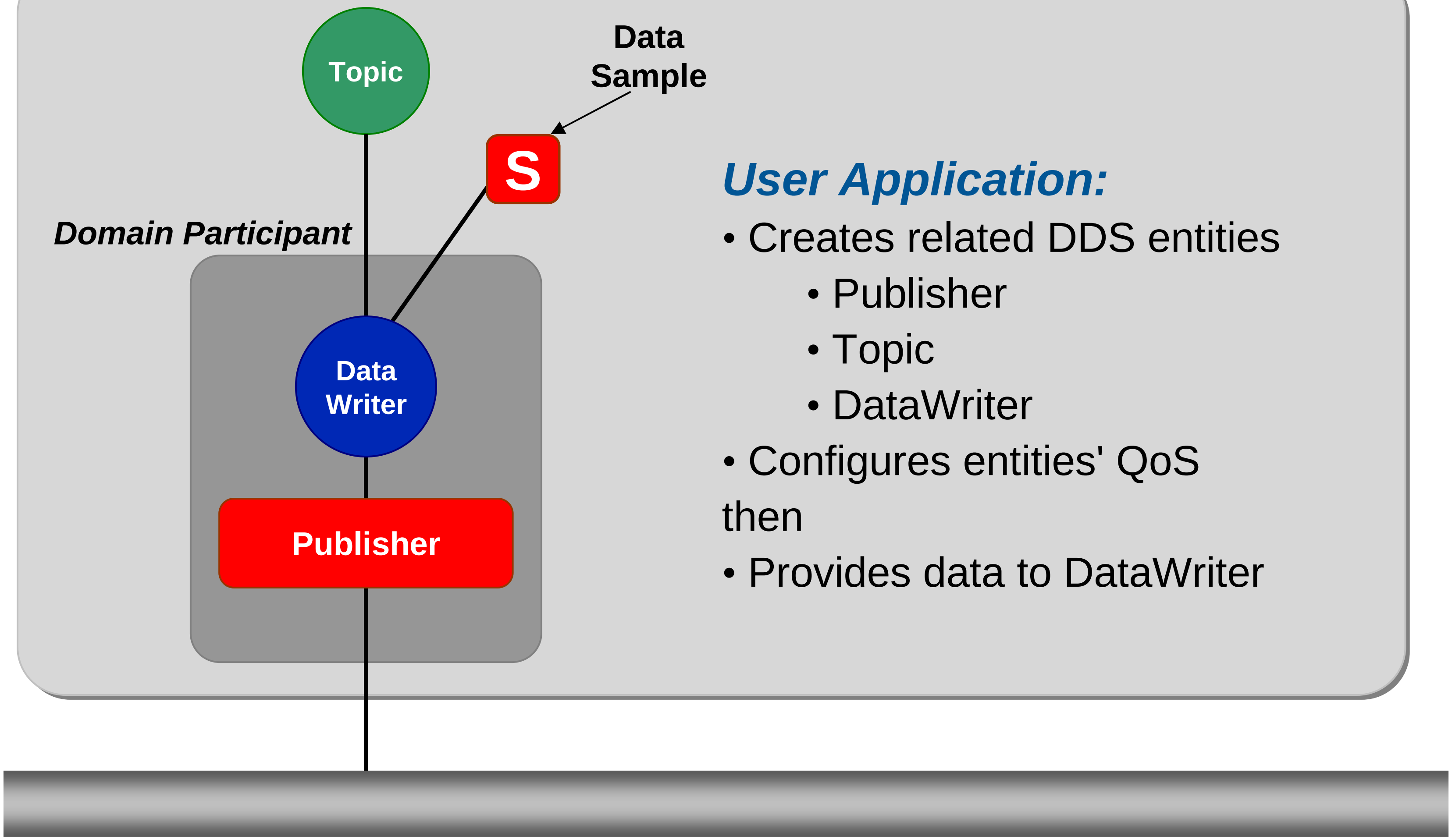
- Aggregation and projection in DDS, e.g. using SQL:

```
select position, environment
from PointTrack NATURAL JOIN TrackState
where position.range < 10000
      and identity = hostile
```

Typical Process Behavior



DDS Publication



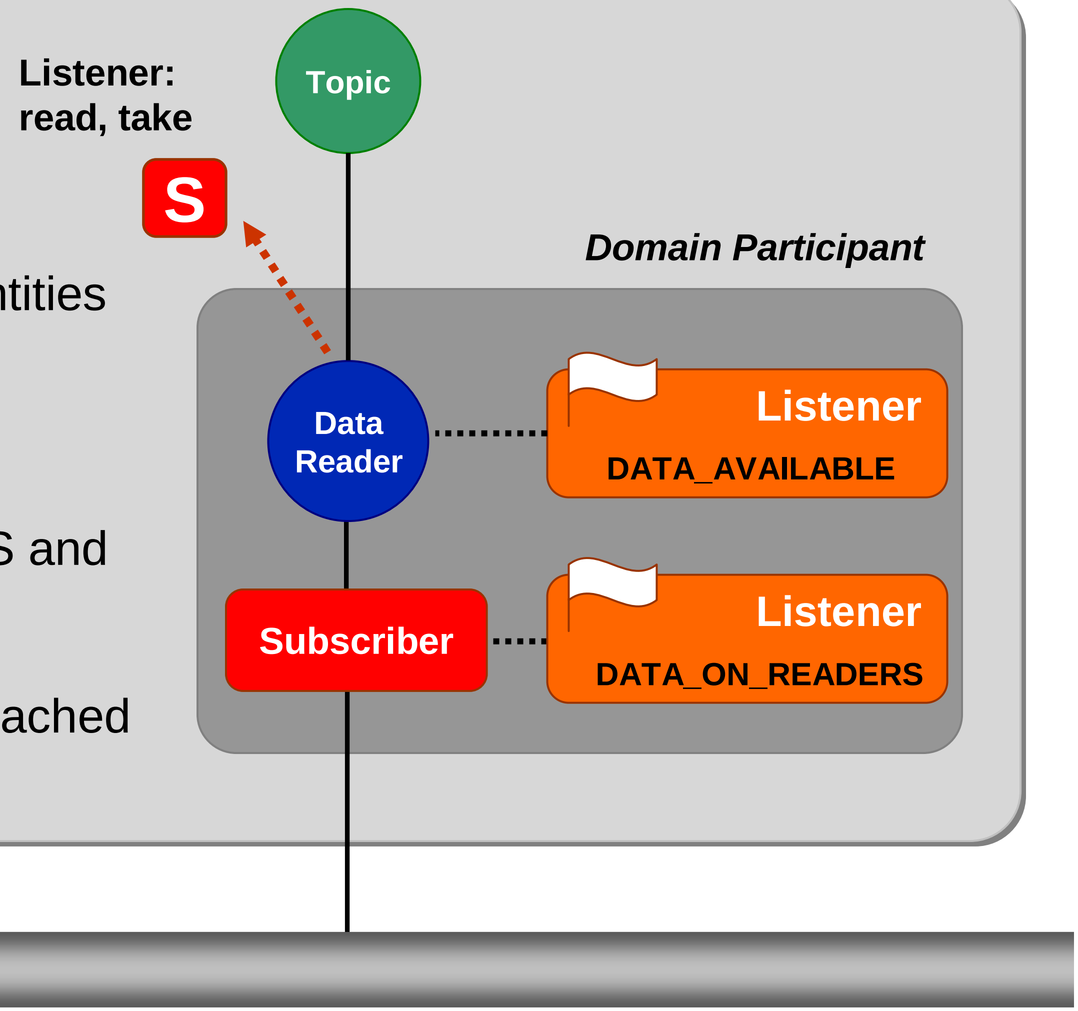
User Application:

- Creates related DDS entities
  - Publisher
  - Topic
  - DataWriter
- Configures entities' QoS then
- Provides data to DataWriter

DDS Subscription with Listeners

User Application:

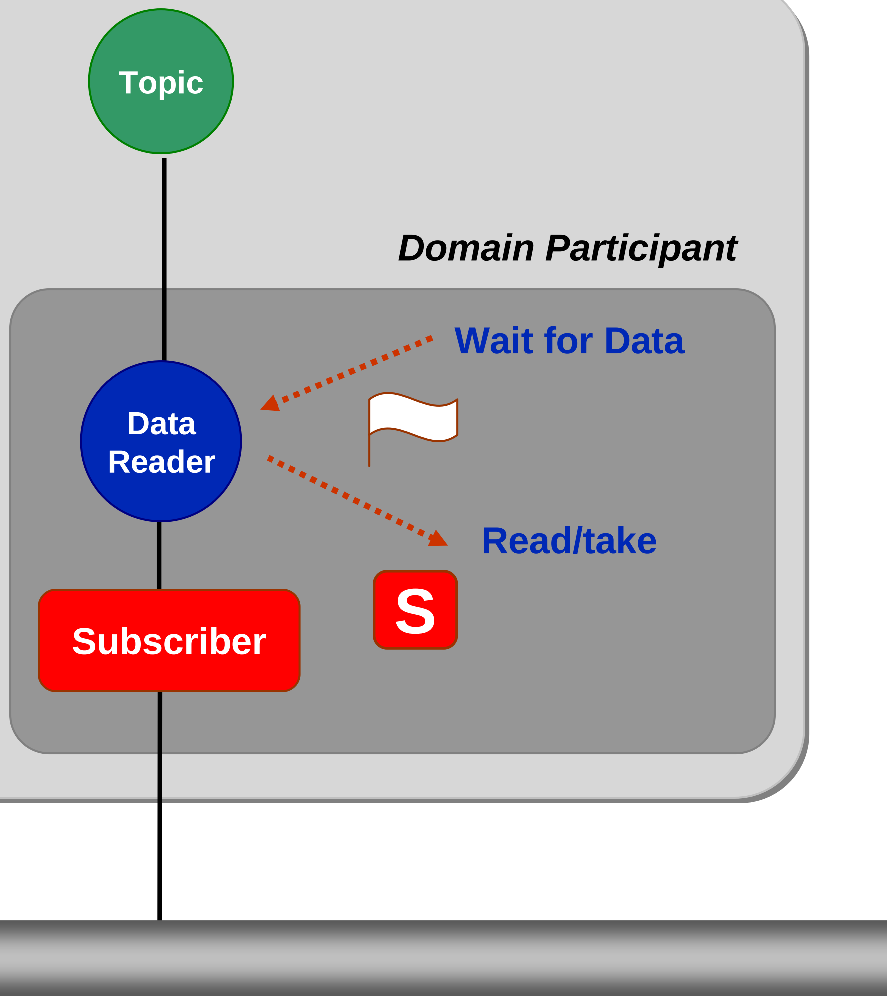
- Creates related DDS entities
  - Subscriber
  - Topic
  - DataReader
- Configures entities' QoS and attach listeners
- Receives Data from DataReader through attached listeners

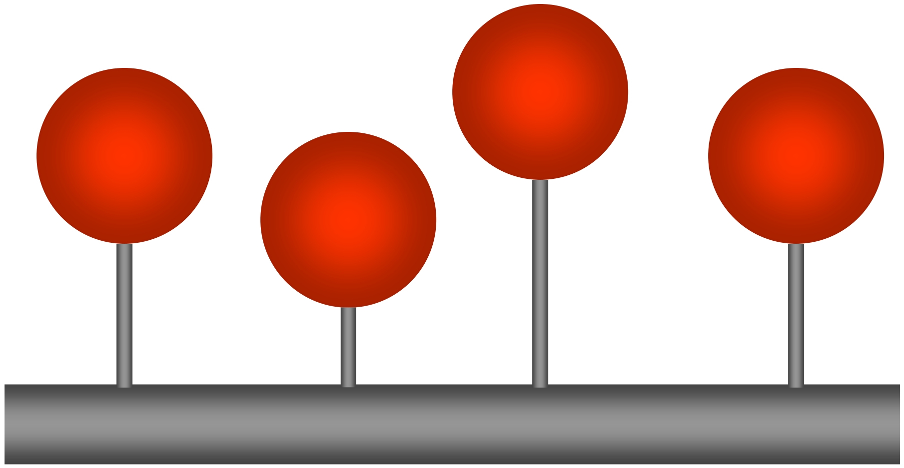


DDS Subscription with Wait-Set

User Application:

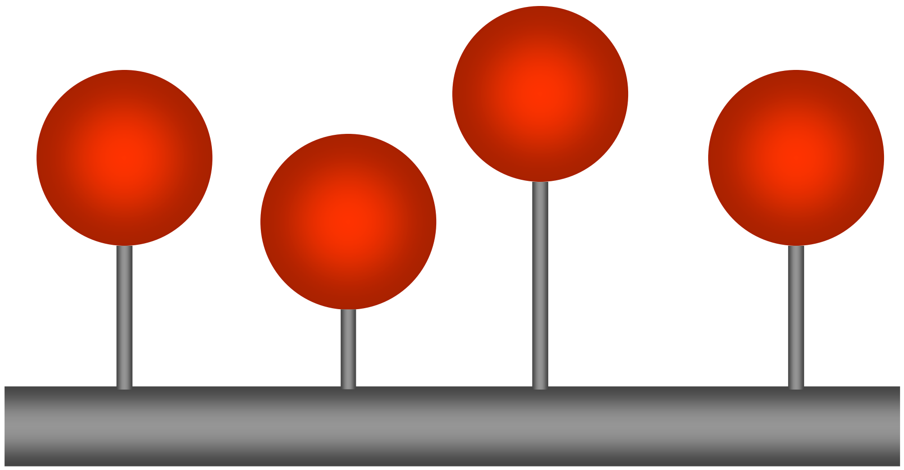
- Creates related DDS entities
  - Subscriber
  - Topic
  - DataReader
- Configures entities' QoS
- Creates a Condition and attaches it to a WaitSet
- Waits on the WaitSet until data arrive, then picks it on the DataReader





Built-in capabilities:

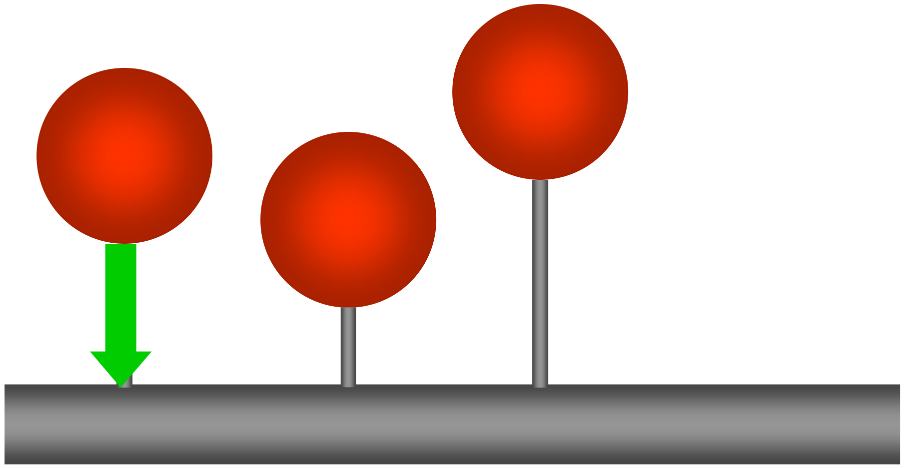
- P/S data distribution
- **relational data access**
- data persistence
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- information partitioning



Persistent data remains available for later access

Built-in capabilities:

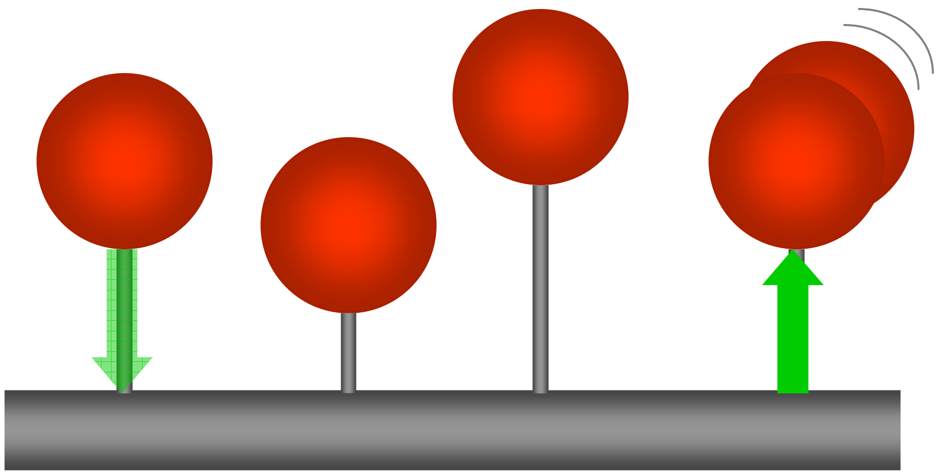
- P/S data distribution
- relational data access
- **data persistence**
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- information partitioning



Persistent data remains available for later access

Built-in capabilities:

- P/S data distribution
- relational data access
- **data persistence**
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- information partitioning



Persistent data remains available for later access

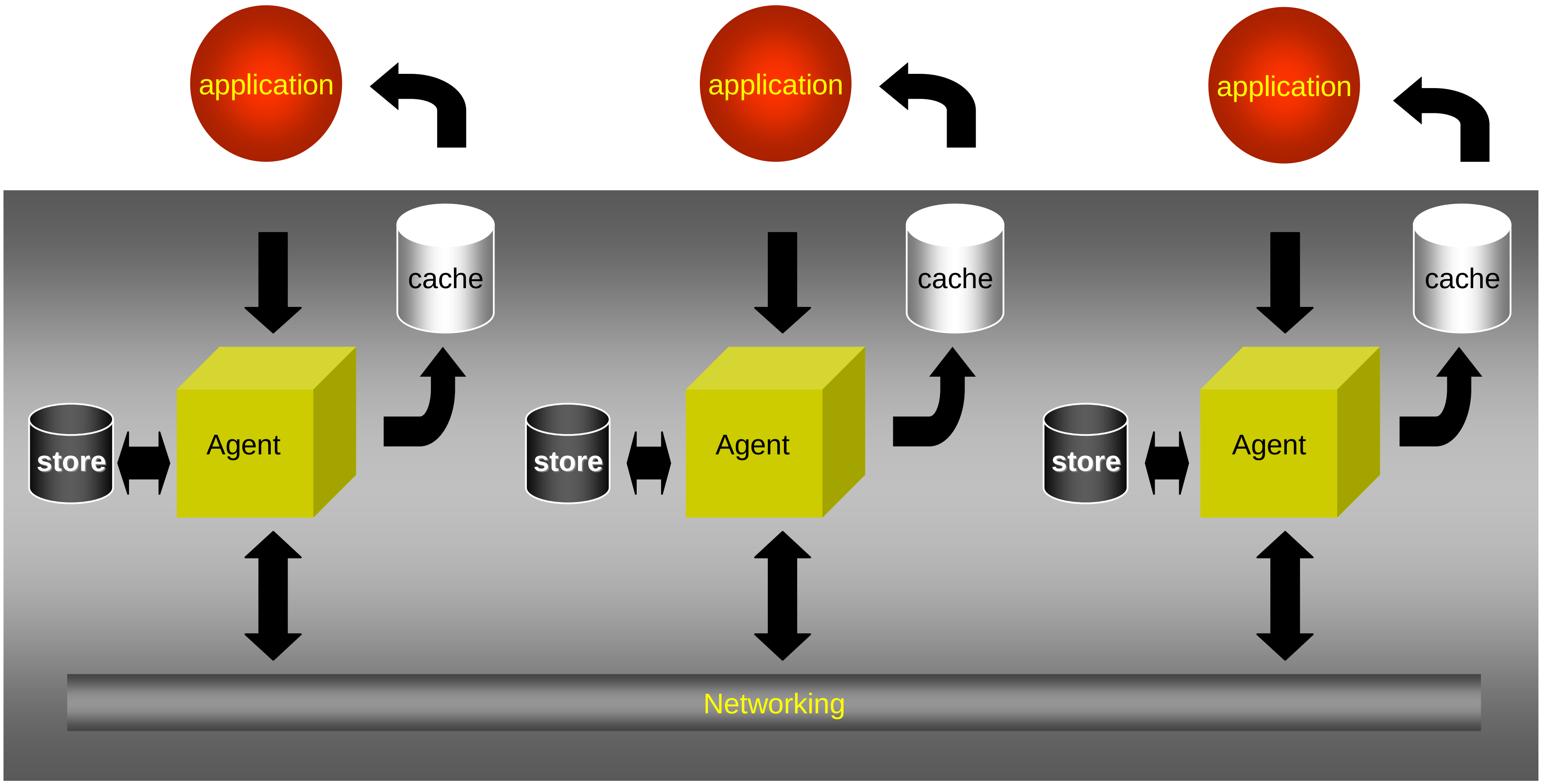
Built-in capabilities:

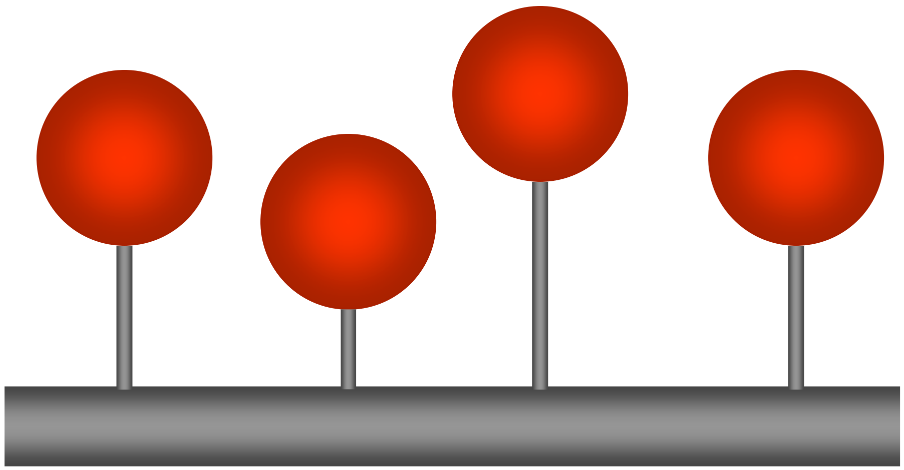
- P/S data distribution
- relational data access
- **data persistence**
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- information partitioning

Data Persistency

- **Volatile** data:
  - no copies outside process space
  - typically ***measurement*** related data
- **Transient** data:
  - copies are kept on more than one host
  - outlives process
  - typically ***state*** related data
- **Persistent** data:
  - same as transient data, but additionally stored on disk
  - outlives system
  - typically ***configuration*** data

Self-Healing: Fault-tolerant persistence

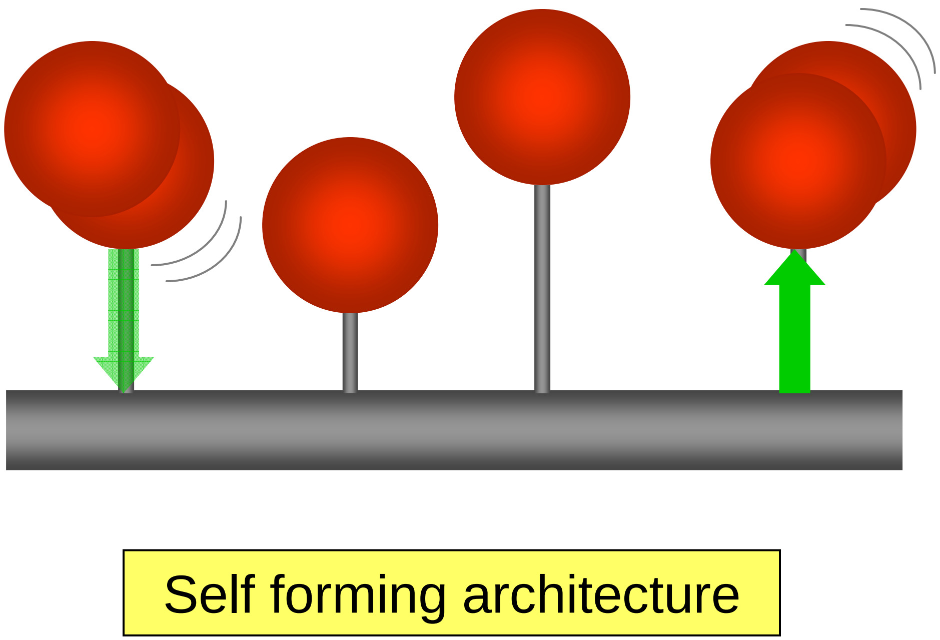




Self forming architecture

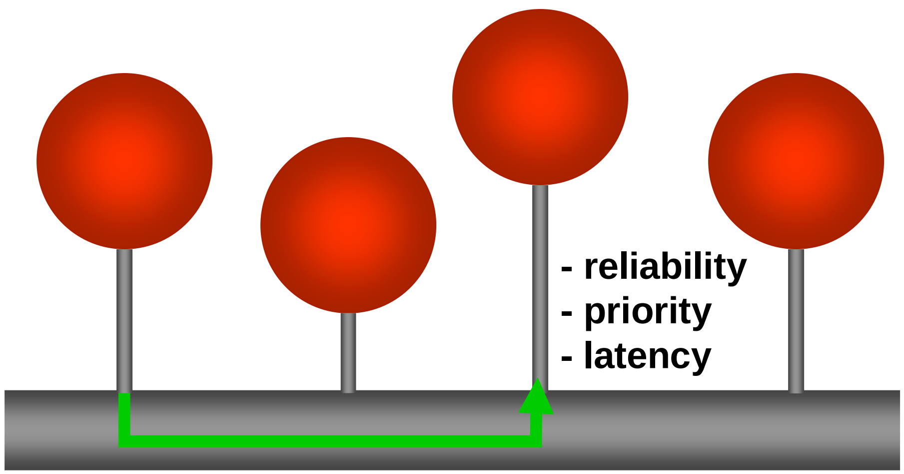
Built-in capabilities:

- P/S data distribution
- relational data access
- data persistence
- **dynamic (re-)configuration**
- quality of service
- fault-tolerance
- information partitioning



Built-in capabilities:

- P/S data distribution
- relational data access
- data persistence
- **dynamic (re-) configuration**
- quality of service
- fault-tolerance
- information partitioning



QoS is attributed to data
(statically or dynamically)

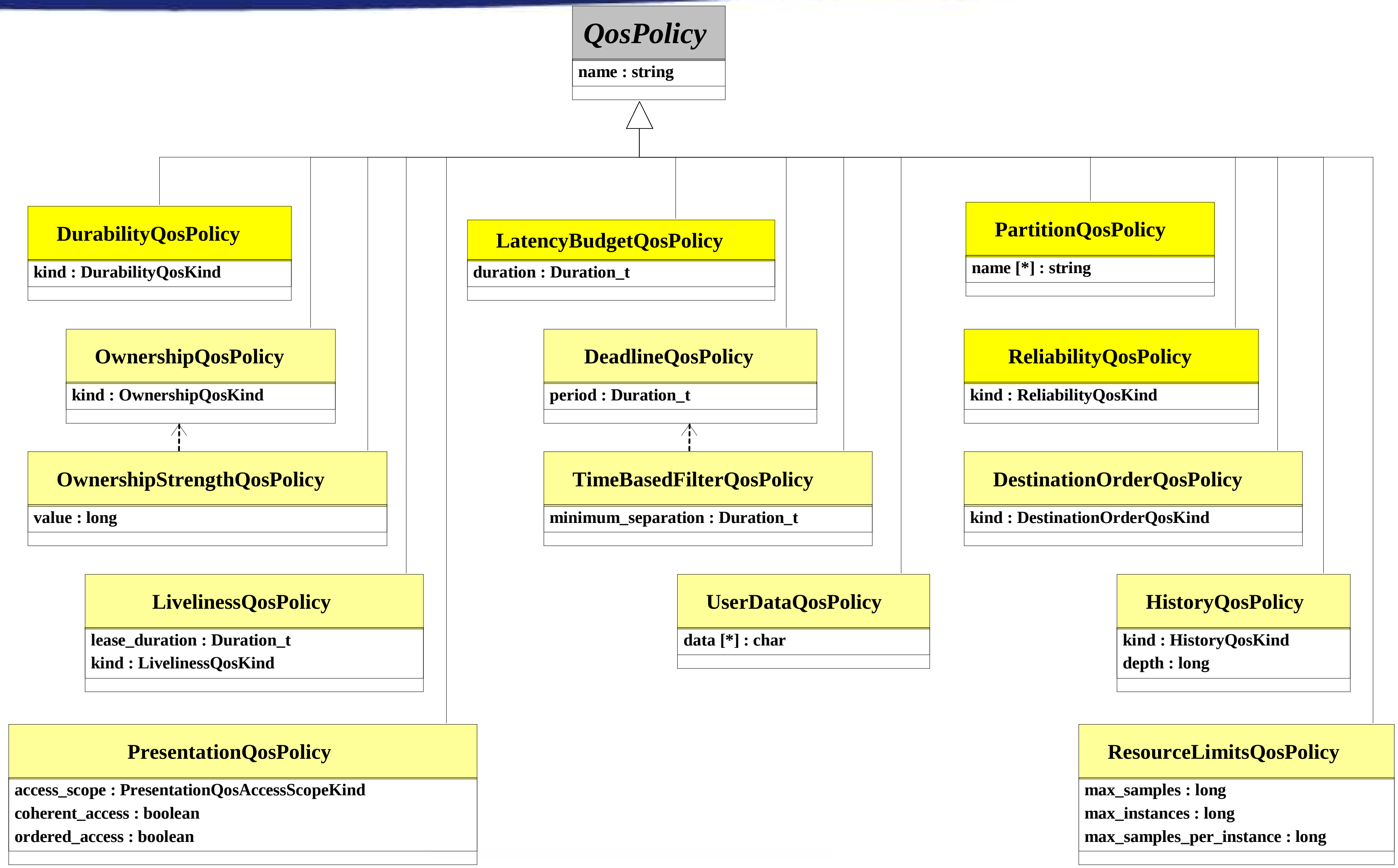
Built-in capabilities:

- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- **quality of service**
- fault-tolerance
- information partitioning

Data Delivery

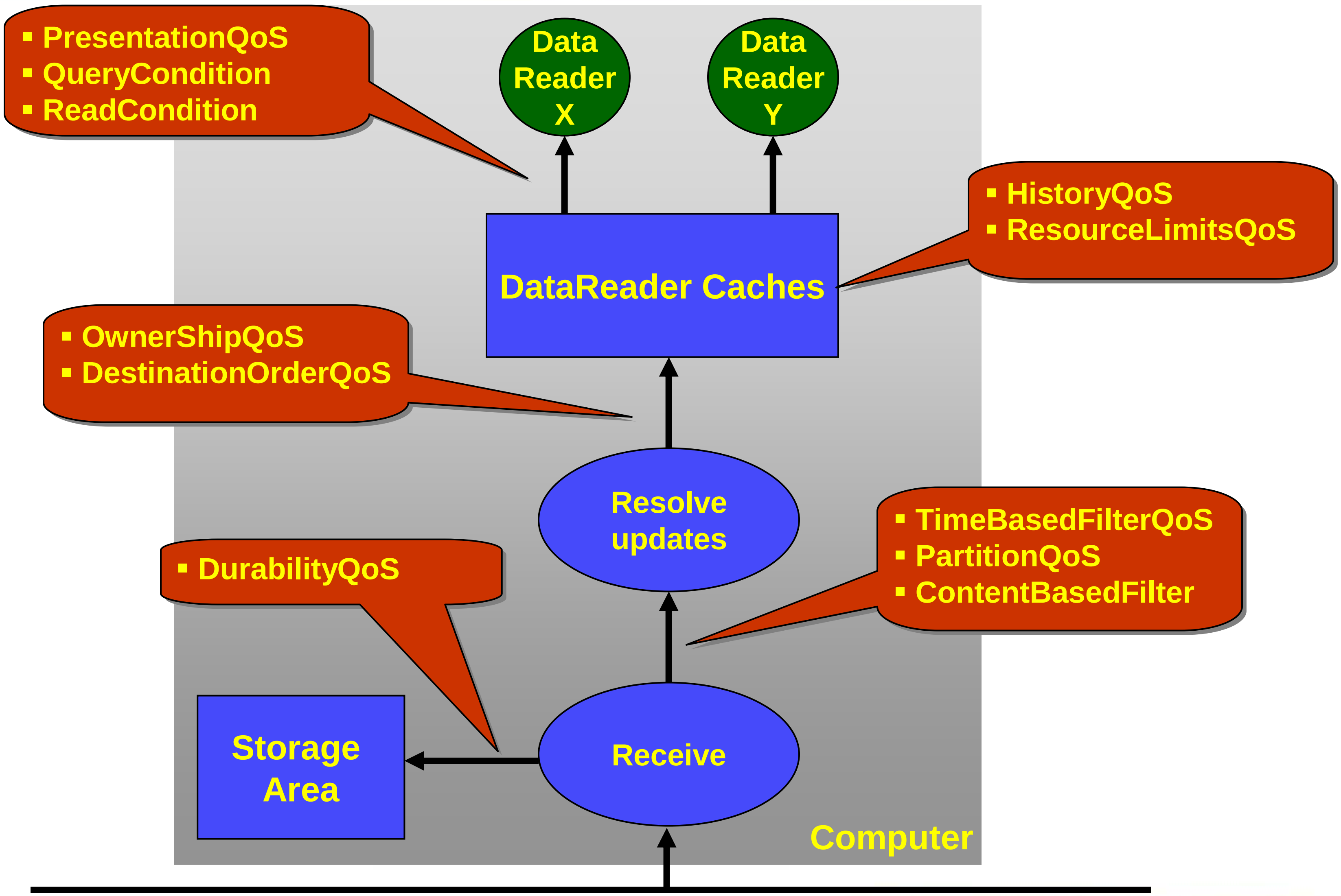
- **Reliability**:
  - **“Best effort”**
    - *No Guaranteed delivery i.e. no retransmissions when data ‘gets lost’*
    - *Typical for **periodic measurement** related data*
  - **“Reliable”**
    - *Guaranteed delivery by automatic re-transmissions of lost data*
    - *Typical for **non-periodic and important** published information*
- **Latency**:
  - **“Latency Budget”**
    - *Specifies ‘how fast’ data should be delivered i.e. its ‘**urgency**’*
    - *Allows middleware to balance between high-volume & low-latency*
- **Priority**:
  - **“Transport_Priority”**
    - *Specifies the priority of the published information i.e. its ‘**importance**’*
    - *Allows middleware to pre-empt low-priority data with high-priority data*

Overview: DDS QoS policies



PRISMTECH
Productivity Tools & Middleware

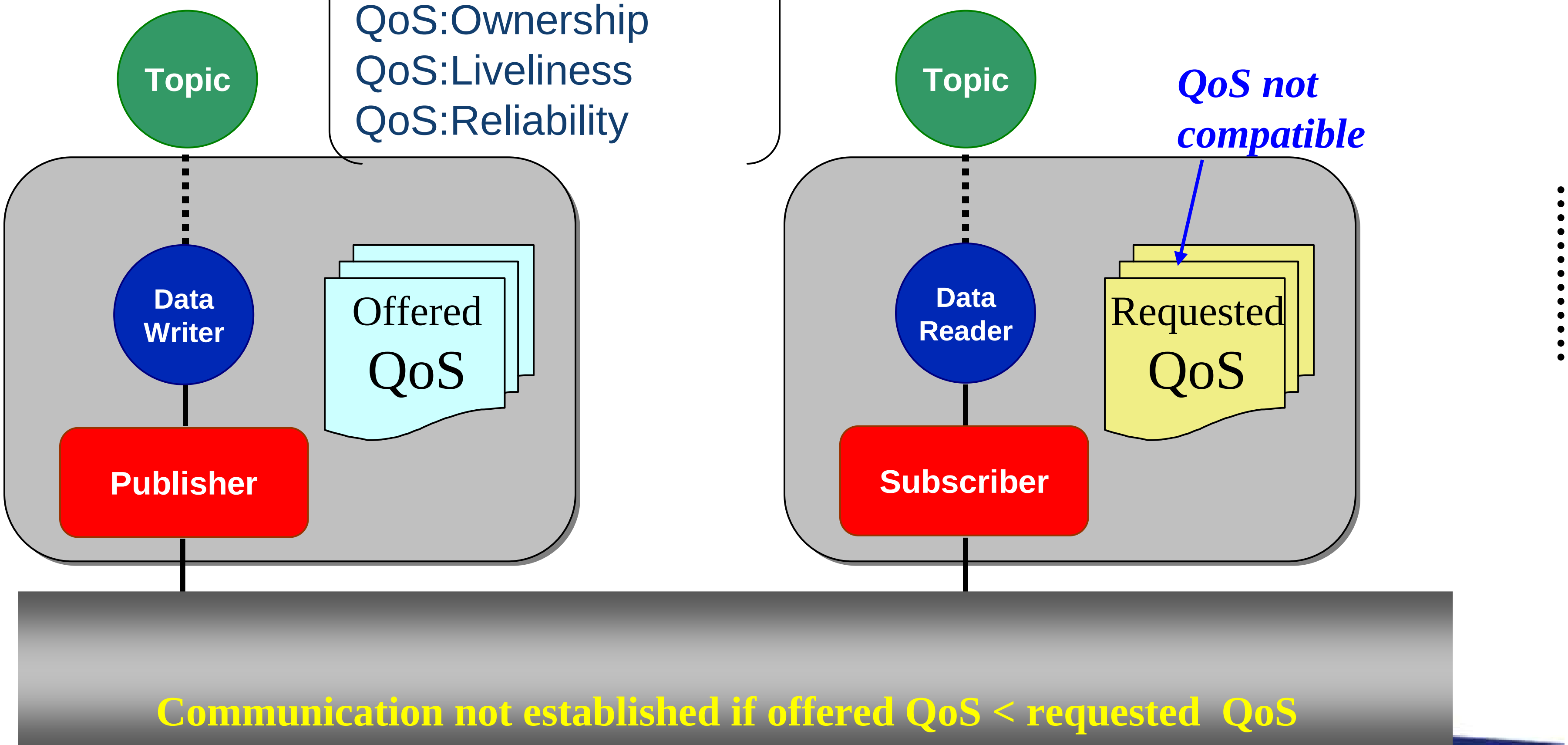
QoS Policies – Conceptual View

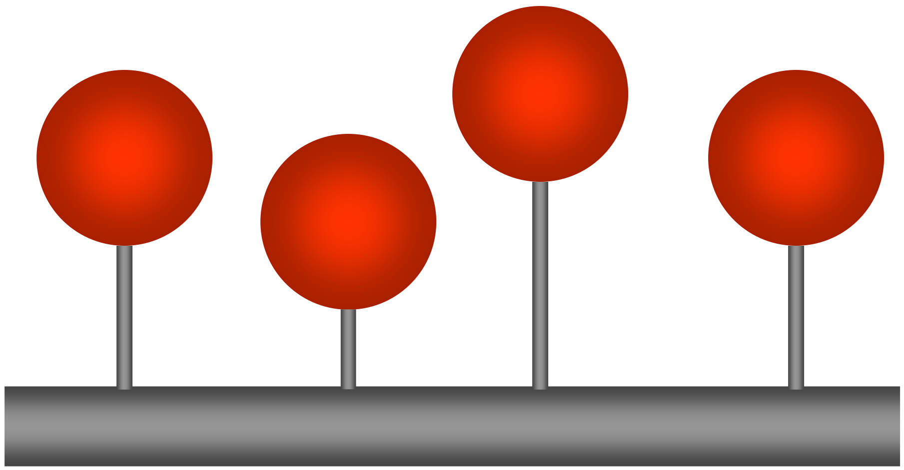


QoS Contract "Request / Offered"

QoS:Durability
QoS:Presentation
QoS:Deadline
QoS:Latency_Budget
QoS:Ownership
QoS:Liveliness
QoS:Reliability

QoS Request / Offered:
Ensure that compatible QoS parameters are set.

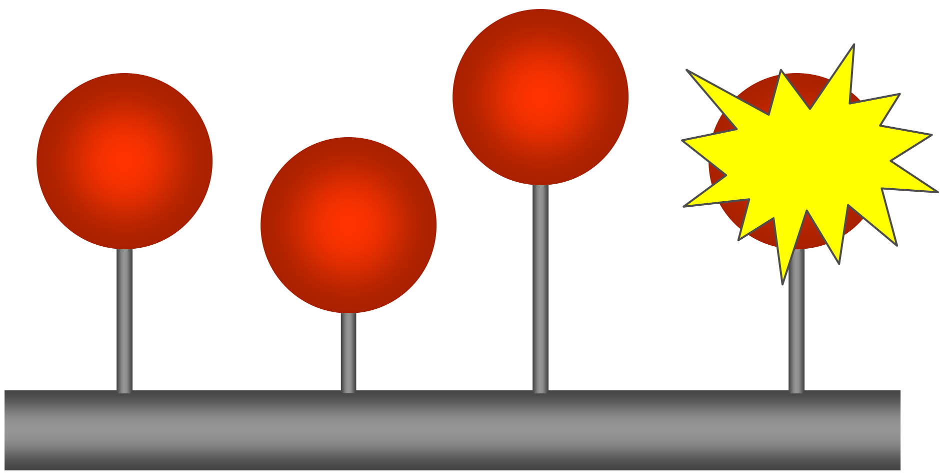




Self-healing software architecture

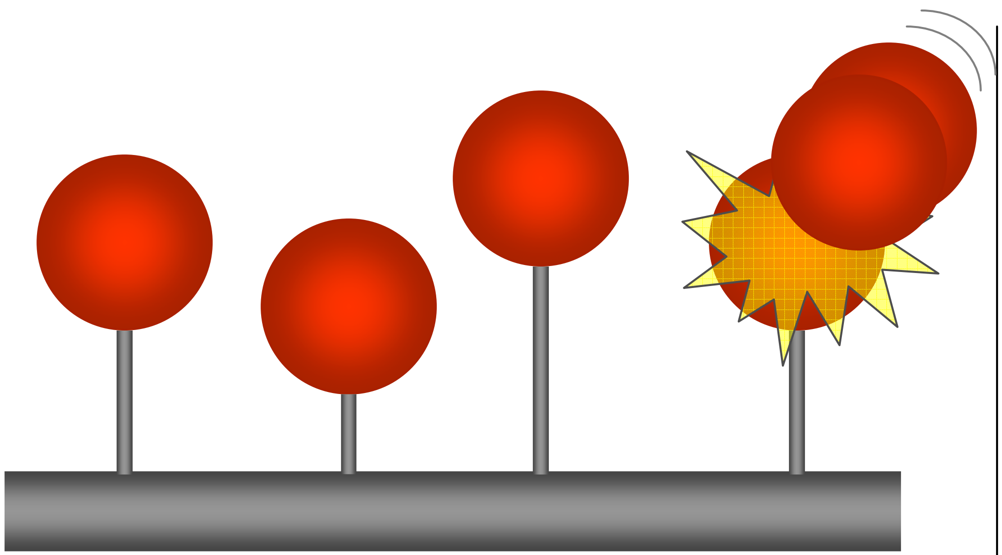
Built-in capabilities:

- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- **fault-tolerance**
- information partitioning



Passive process replication

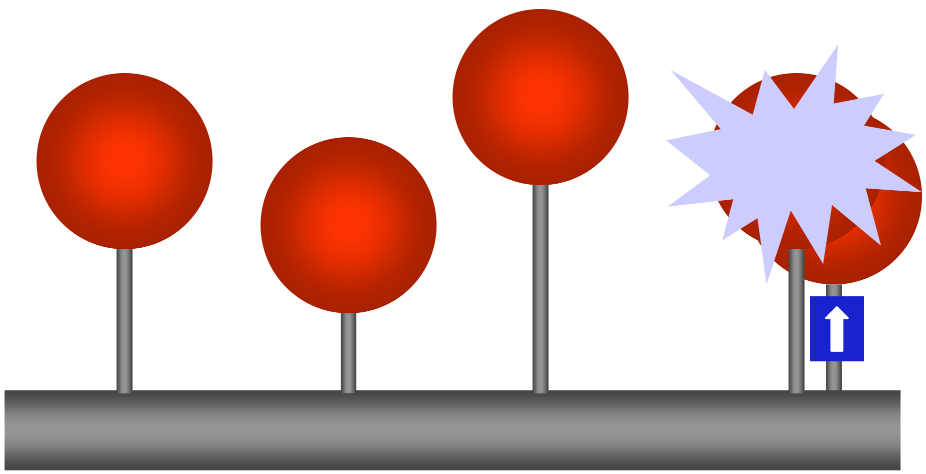
Built-in capabilities:
- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- **fault-tolerance**
- information partitioning



Passive process replication

Built-in capabilities:

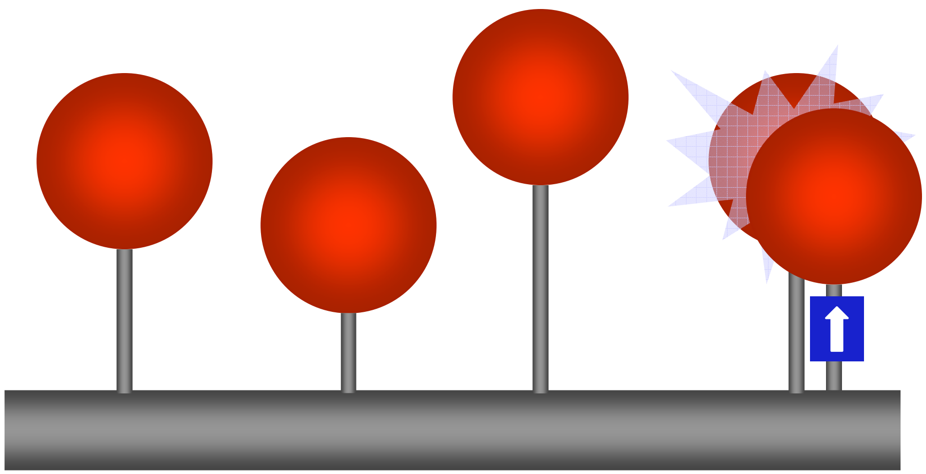
- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- **fault-tolerance**
- information partitioning



Semi-active process replication

Built-in capabilities:

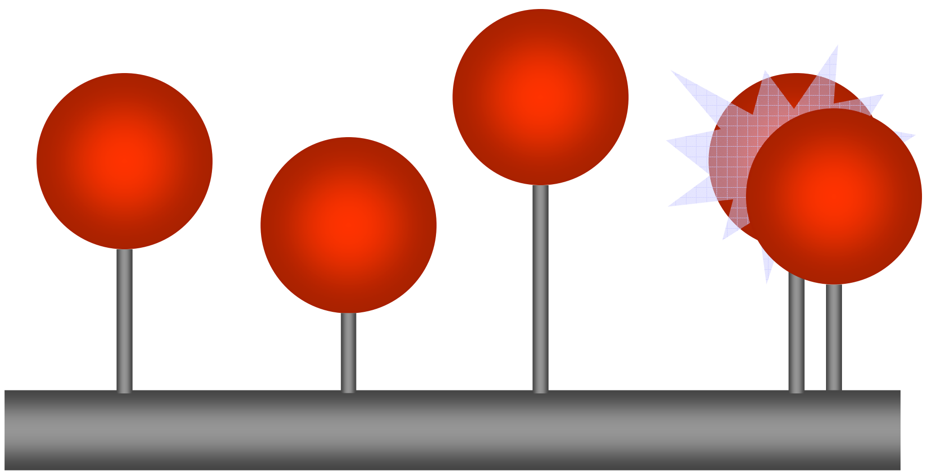
- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- **fault-tolerance**
- information partitioning



Semi-active process replication

Built-in capabilities:

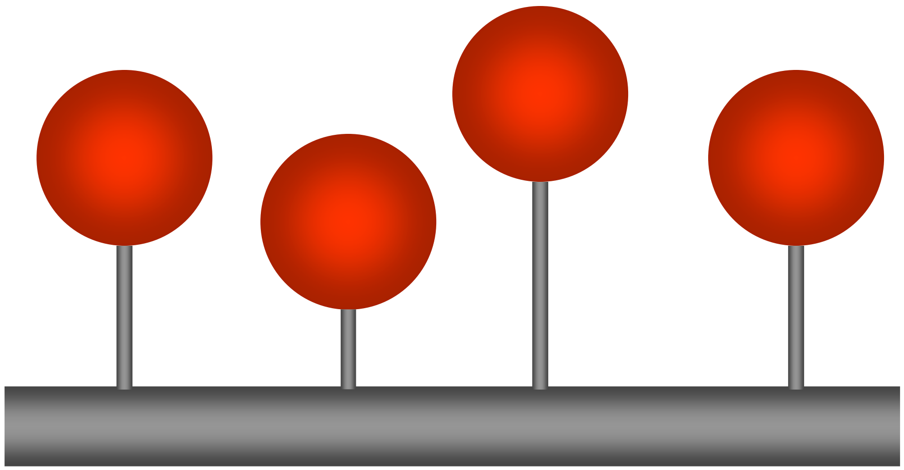
- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- **fault-tolerance**
- information partitioning



Semi-active process replication

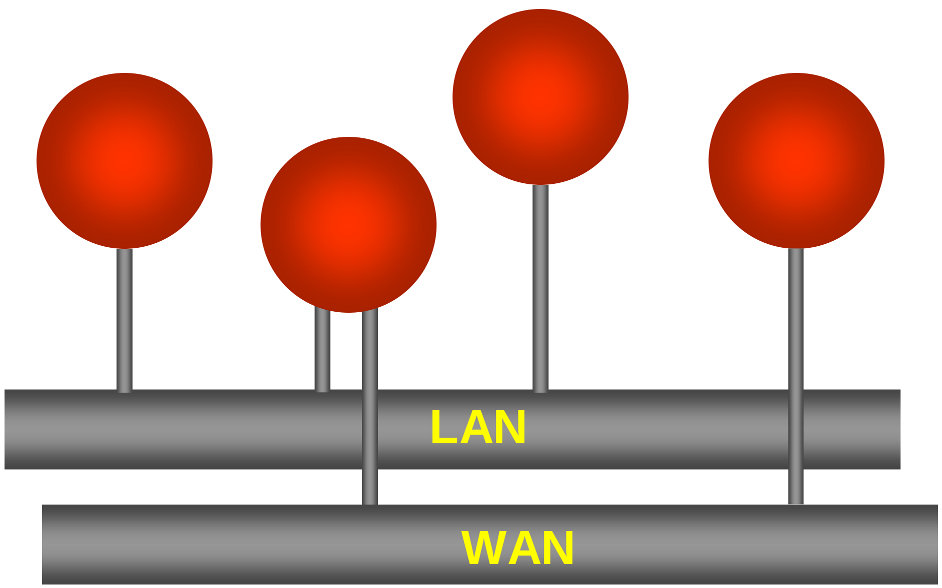
Built-in capabilities:

- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- **fault-tolerance**
- information partitioning



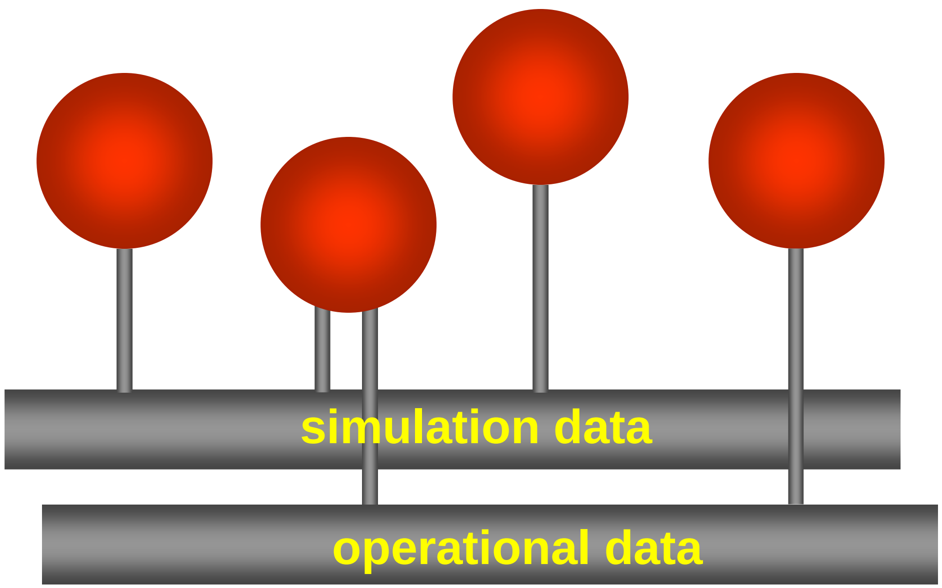
Built-in capabilities:

- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- **information partitioning**



Built-in capabilities:

- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- **information partitioning**

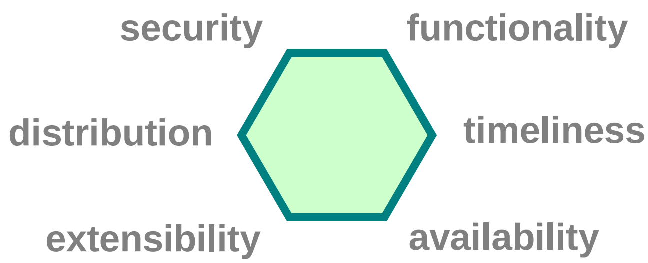


Built-in capabilities:

- P/S data distribution
- relational data access
- data persistence
- dynamic (re-) configuration
- quality of service
- fault-tolerance
- **information partitioning**

Summary: The Problem

- **Problem:** ***engineering (-cost) of distributed systems***
  - too complex
  - not reactive
  - not future-proof
  - not fault tolerant



- **Because** ***'multi-dimensional engineering' is needed:***

- **What about** ***the current 'state-of-the-art'?***
  - architectures: client/server, message-passing, DBMS
  - most efforts fall short in a number of dimensions:
  - typically:
    - *limited RT performance* *(high-volume & low-latency balance)*
    - *exploding complexity* *(dependencies in many dimensions)*
    - *costly evolution* *(impact of changes & extensions)*

Summary: An Information-Centric Approach

- **Towards a solution:**
  - make development effort more **simple**
  - develop **less**
  - develop solutions **only once**

- **How:**
  - minimize component dependencies ***('simple')***
  - maximize component autonomy ***('re-use')***
  - normalize component interactions ***('only once')***

- **The clue:**
  - ***share*** the stable properties, ***localize*** the unstable ones
  - ***information*** is what matters most, not how it is processed
  - properly modeled data is ***stable***, processing often is not
  - so ***focus on data*** first and then on the processing of it

Summary: the DDS concept

Data-centric architecture:

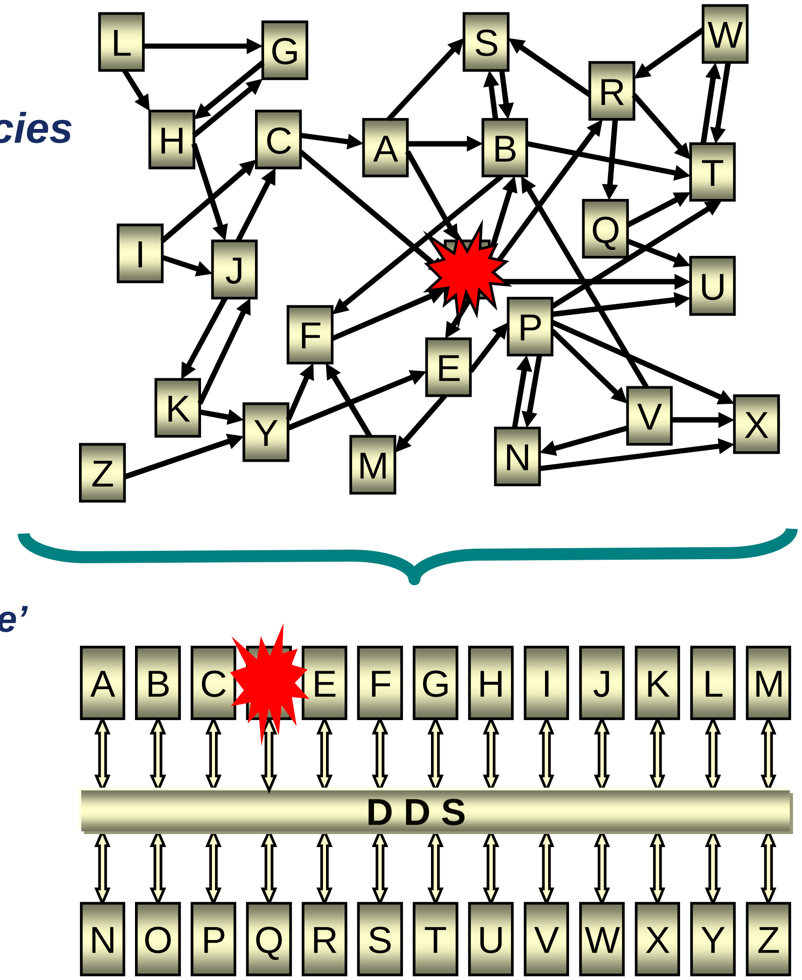
- ***autonomous*** components with ***minimal dependencies***
- ***separation*** of function and interaction
- architecture that ***focuses*** on ***data***

DDS realization:

- middleware that offers a ***normalized environment***
- designed once, ***guarantees*** system properties
- delivers ***'the right data at the right place at the right time'***

What about the processing?

- ***standard*** operating platform (HW, OS)
- ***standard*** communication facilities
- ***standard*** programming languages
- ***standard*** development tools



SUMMARY: Requirements & Realization

Requirement:	Realized by:
System design	
provide a stable basis to operate upon by applications	*- shared Information Model*
enhance component autonomy	*- state-based information-centric system*
allow transparent and global QoS assurance	*- Information classification (QoS topic-defaults)*
System development	
reduce complexity and enhance re-usability	*- minimized component dependencies*
provide shared/guaranteed properties	*- standardized (DDS-) interaction-environment*
small learning effort and flat learning curve	*- intuitive concept, simple/powerful features*
System integration	
support effortless component integration	*- maximized component autonomy*
provide easy monitor & control	*- globally accessible information (data+metadata)*
shift ratio between design and integration effort	*- focus on info-model & decoupled applications*
System deployment	
guaranty QoS for reliability, latency and persistency	*- realtime "DDS" information backbone*
allow runtime migration of applications	*- global & FT availability of transient state data*
allow applications to join the system at any time	*- dynamic discovery and data persistence*
System maintenance & evolution	
allow runtime replacement and evolutionary upgrading	*- de-coupled & autonomous components*
support for logging & replay of information	*- global availability of all (time-stamped) data*
provide future-proof, re-usable, robust and scalable system	*- highly adaptive associative data-model*

PRISMTECH
Productivity Tools & Middleware

.. 15 minutes break ..

PRISMTECH
Productivity Tools & Middleware

Combat Systems Example...

... *A requirements-driving domain*

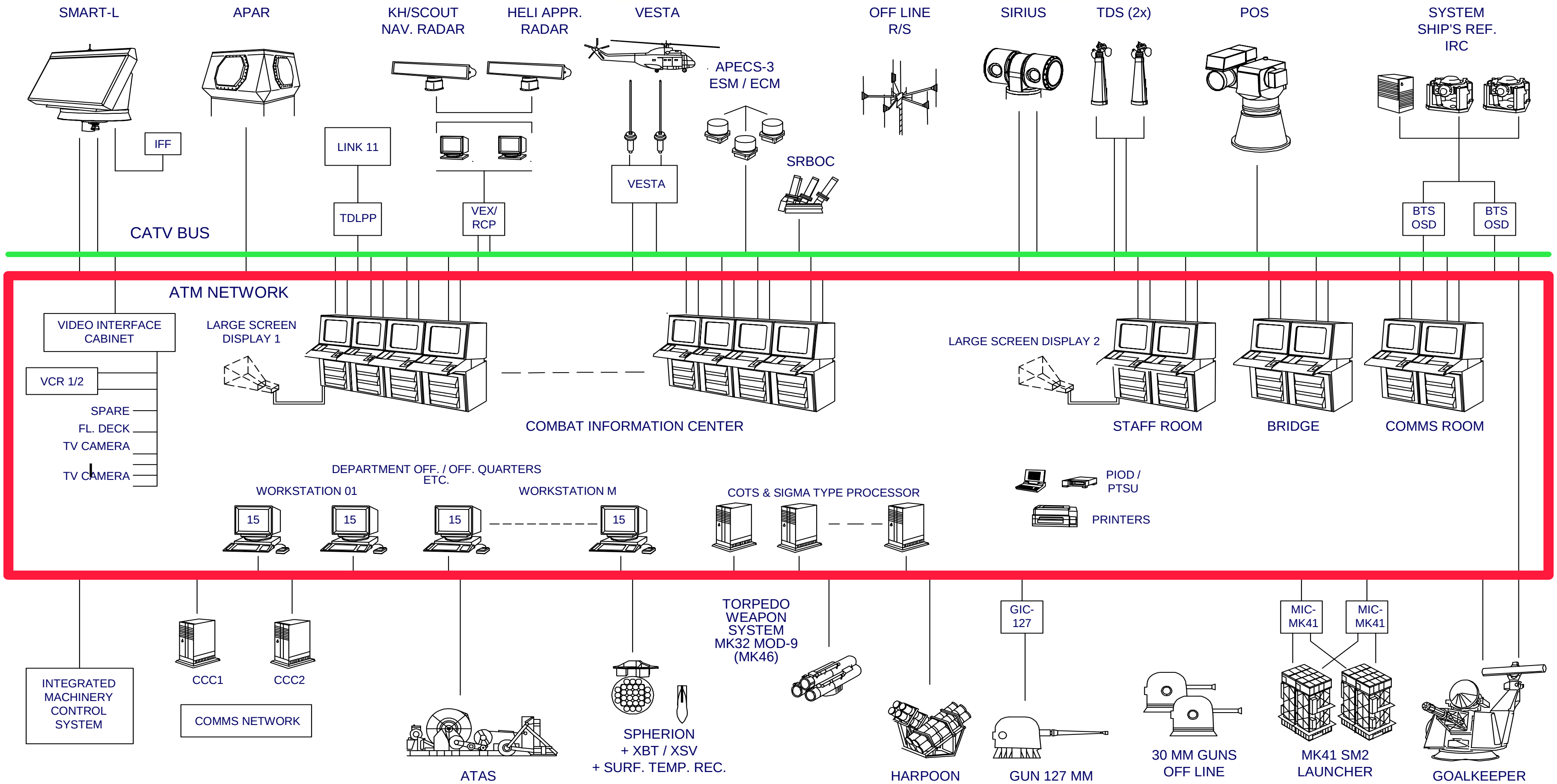
DDS usage in Naval Combat Systems

CHARACTERISTICS

Many different customers:	***>**15 Navies world-wide use DDS pub/sub*
Many different ships/missions:	***>** 22 Ships classes (from FPB's up to Destroyers)*
Large-scale & mission-critical:	*>150 CPU's, >2200 applications, >4.000 tracks/sec*
Real-time and Fault-tolerant:	*Battle-damage resistant, deterministic, reliable*

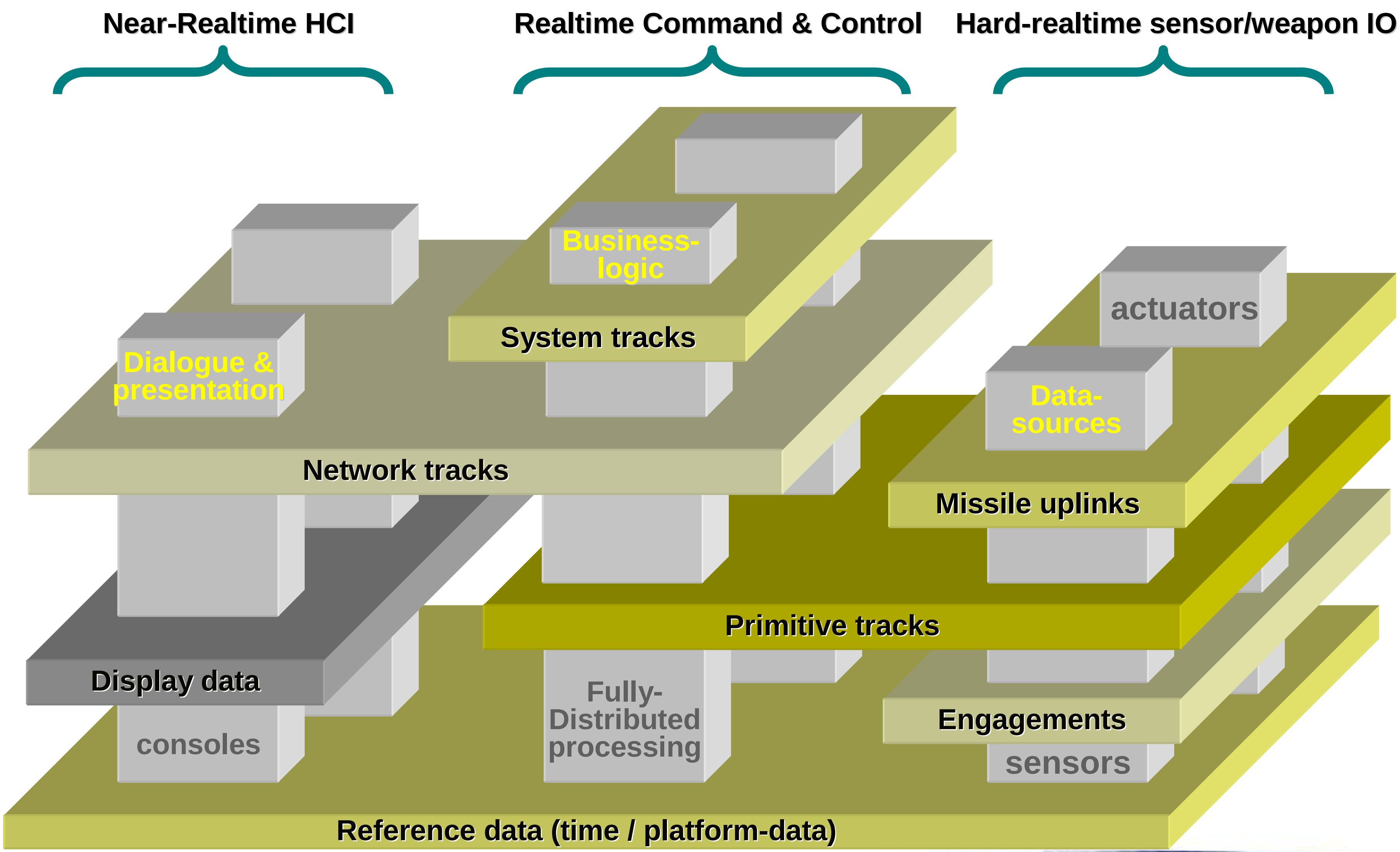
PrismTech
Productivity Tools & Middleware

Example: Frigate-size environment



- **Data-traffic:** *>4.000 publications per second over the system-data bus*
- **Programs:** *2.200 programs allocated over 150 processors*
- **Data flows:** *urgent & non-urgent data (latency), important & less-important data (priority)*

DDS: 'Self-forming 3-tier data-planes'



PRISMTECH
Productivity Tools & Middleware

the OMG DDS Specificiation

The DDS Specification

- Data Distribution Service for Real-Time Systems
  - Adopted in June 2003, Finalized in April 2004
  - Joint submission
    www.omg.org/technology/documents/formal/data_distribution.htm
  - Specifies the API required for a Data-Centric Publish-Subscribe communication environment for real-time distributed systems

Dr. Richard Soley, OMG Chairman and CEO:

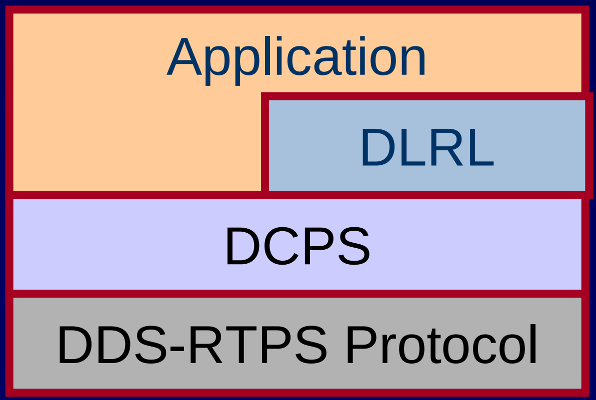
"The DCPS publish/subscribe model stands as a natural complement to the object-centric client-server model provided by CORBA."

"The finalization and availability of the DDS specification really is a tremendous achievement that addresses a significant need in both government and civilian sectors."

-- in *Consumer Electronics*, September 13, 2004

DDS Structure

- DCPS: Data Centric Publish/Subscribe
  - **Purpose: QoS-Driven Distributed Data Management**
- DLRL: Data Local Reconstruction Layer (optional)
  - **Purpose: An OO Model to access data as local objects**
- Related Specification: DDS-RTPS
  - **Purpose: provide (network-)interoperability between multiple DDS vendors**



DDS Features

Object Oriented information view

- *Local object model extending the distributed DCPS data model*
- *Manages relationships and supports native language constructs*

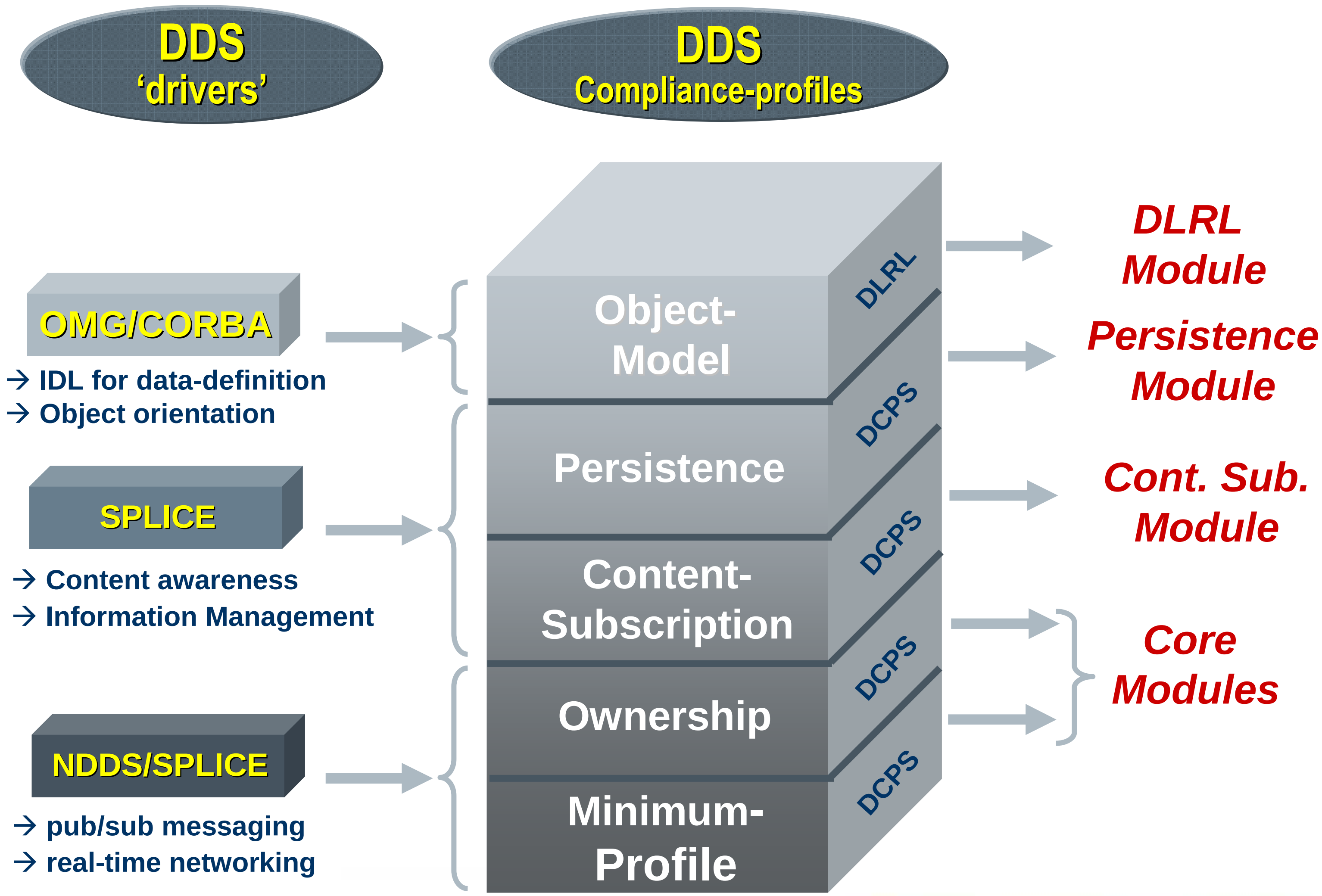
Distributed QoS-driven information management

- *Fault tolerance and global persistence of selected data*
- *Guaranteed data availability supports application fault-tolerance*
- *Content-aware filtering and dynamic queries:*
  - *reduces application complexity*
  - *improves system performance*

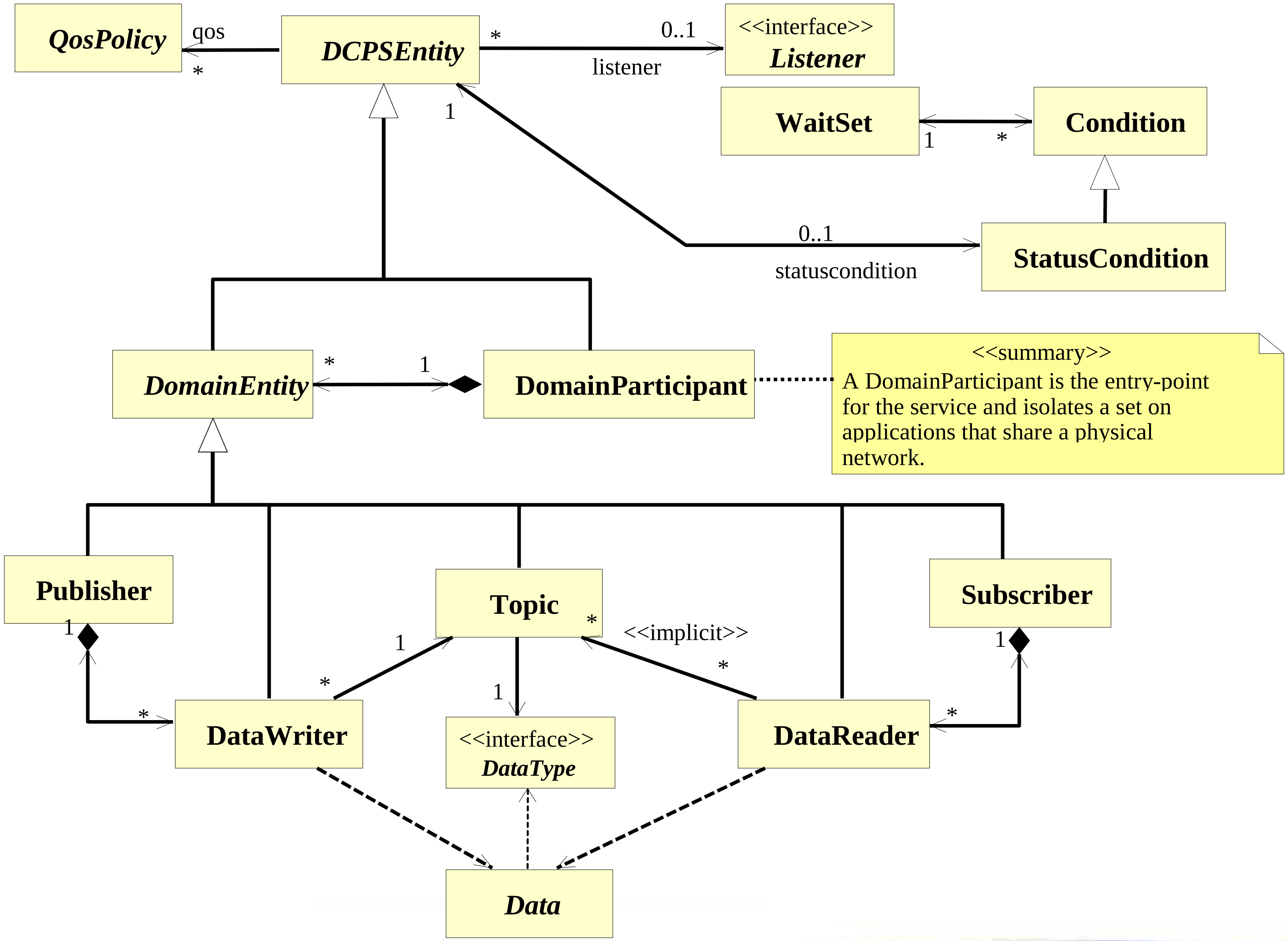
Real-time publish/subscribe messaging:

- *Asynchronous 'one-to-many' real-time data communication*
- *Dynamic data flow based on 'current interest' (pub/sub)*
- *Platform independent data model (IDL)*
- *Strongly typed interfaces for multiple languages*
- *Information Ownership management for replicated publishers*

Contributions to the OMG-DDS specification



Overall DCPS Model

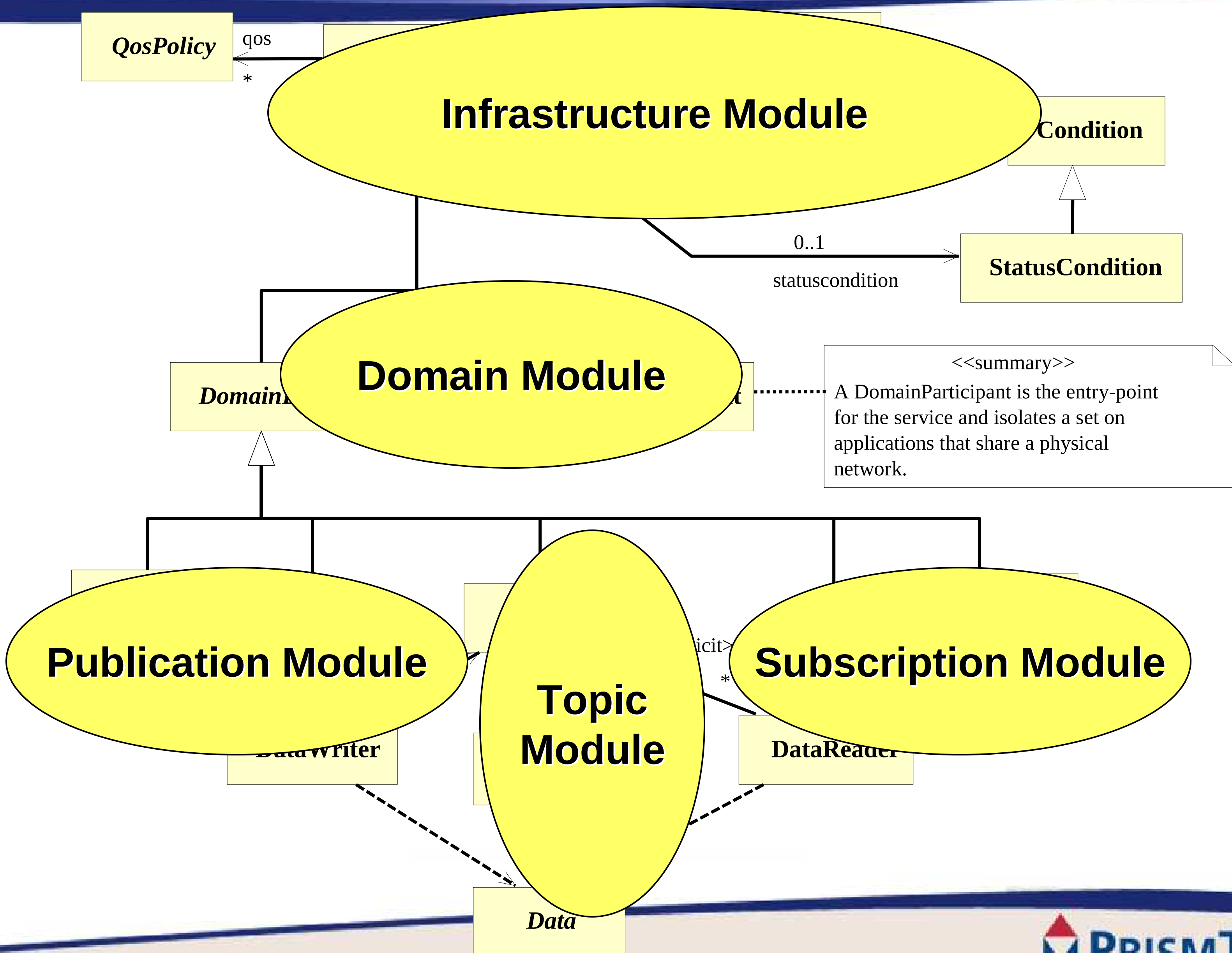


PRISMTECH
Productivity Tools & Middleware

Overall DCPS Model

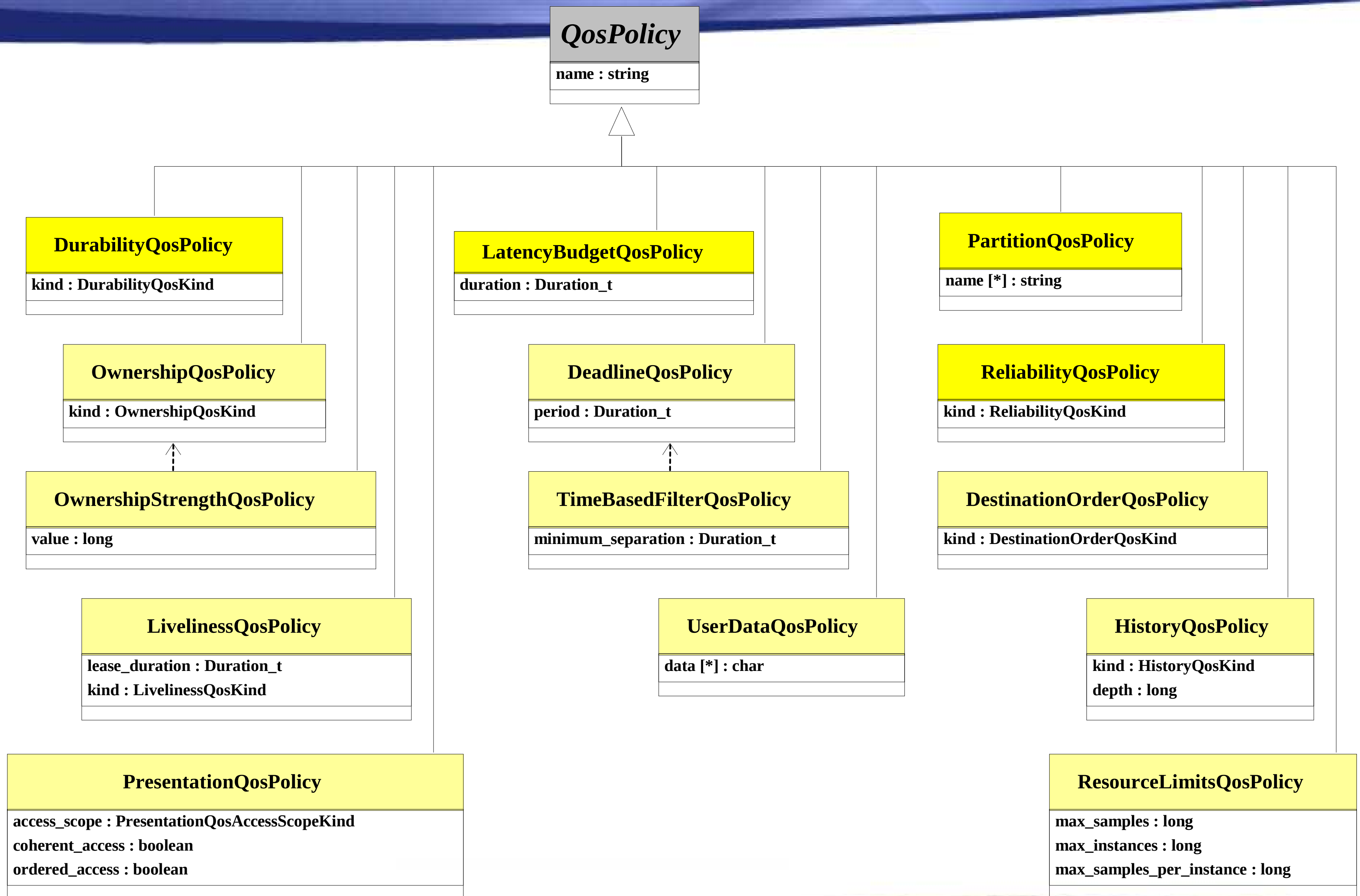
DCPS Entity	Description
Publisher	Responsible for data distribution taking into account the applicable QoS-policies
DataWriter	Holds the data and enables modifications
Subscriber	Responsible for receiving data taking into account the applicable QoS policies
DataReader	Holds the data and provides access to the data
Topic	Associates a name with a data type
Content filtered topic	*Expresses interest in content-filtered data*
MultiTopic	*Expresses interest in aggregated (& filtered) data*

DCPS Architecture



PrismTech Productivity Tools & Middleware

QoS Policies

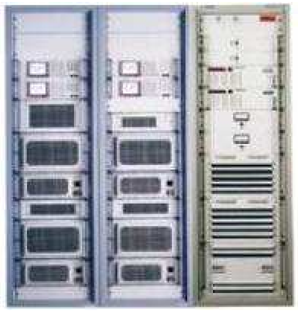


PrismTech
Productivity Tools & Middleware

PRISMTECH
Productivity Tools & Middleware

DDS by Example

DDS™ by Example: "*building a mini CMS, …*"



SENSOR PROCESS

- Optical sensor
- Scans the environment
- Produces 'Tracks'
- Position of 'objects'
- Reports '***pointTrack'***

CLASSIFICATION PROCESS

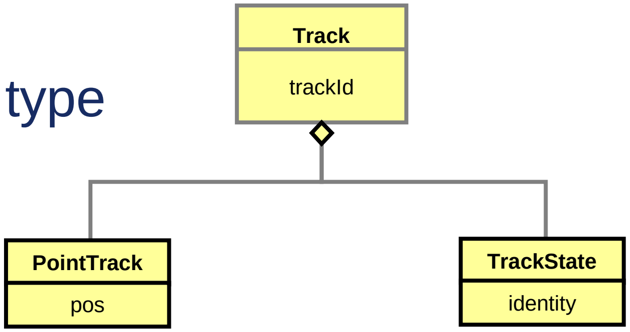
- Classifies tracks
- Determines their identity
- Analyses the trajectories
- Determines hostility
- Reports '***trackState'***

DISPLAY PROCESS

- Displays track info
- Both position & identity
- Raises alerts
- Requires '***pointTrack'***
- Requires '***trackState'***

DATA DEFINITION

- Information modeled as "**TOPICS**"
- Each **TOPIC** has an associated name and data type
  - Data-definition in IDL
  - 'Key' fields for unique identification
  - Relational Data Model (keys)



- Our example:

*Topic **"PointTrack"***

```
Struct PointTrackType {
  long  trackId;        // key

  Position pos;
}
```

*Topic **"TrackState"***

```
Struct TrackStateType {
  long  trackId;        // key

  Id identity;
}
```

DDS™ by Example: “*the Minimum Profile …”*

Sensor

PointTrack Publisher

Topic

PointTrack {

long trackId;

Position pos;

}

***Key*: trackId**

***QoS*: best-effort, volatile**

PointTrack Topics

Display

PointTrack Subscriber

OMG-DDS Real-time Information Backbone

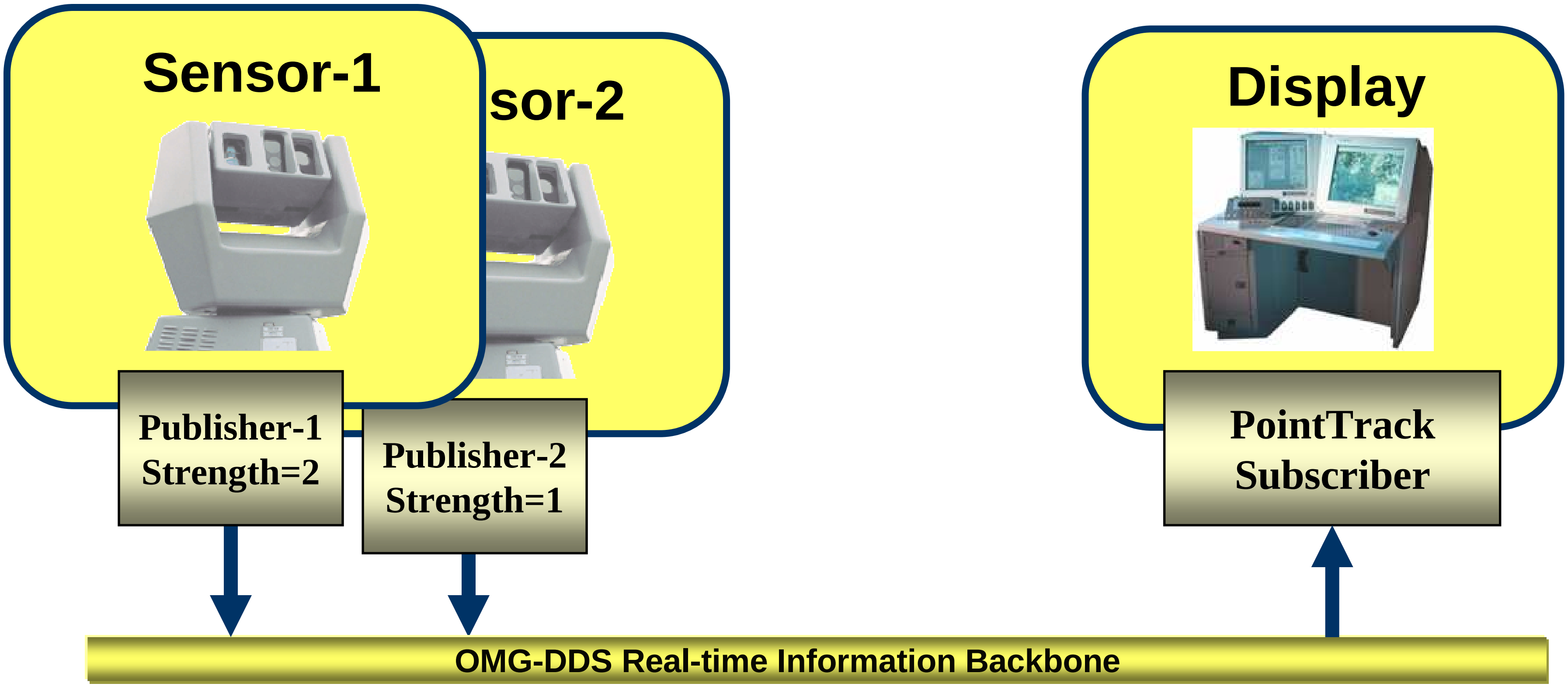
Characteristics

- Basic publish/subscribe data distribution
- Topics (types) specified in IDL
- QoS regarding: reliability, urgency, priority, etc.

Features / Advantages

- Autonomous & loosely coupled applications
- Pub/Sub & QoS driven communication
- Strong-typed interfaces
- Smart networking based on priority & latency budget

DDS™ by Example: “*The Ownership Profile …*”



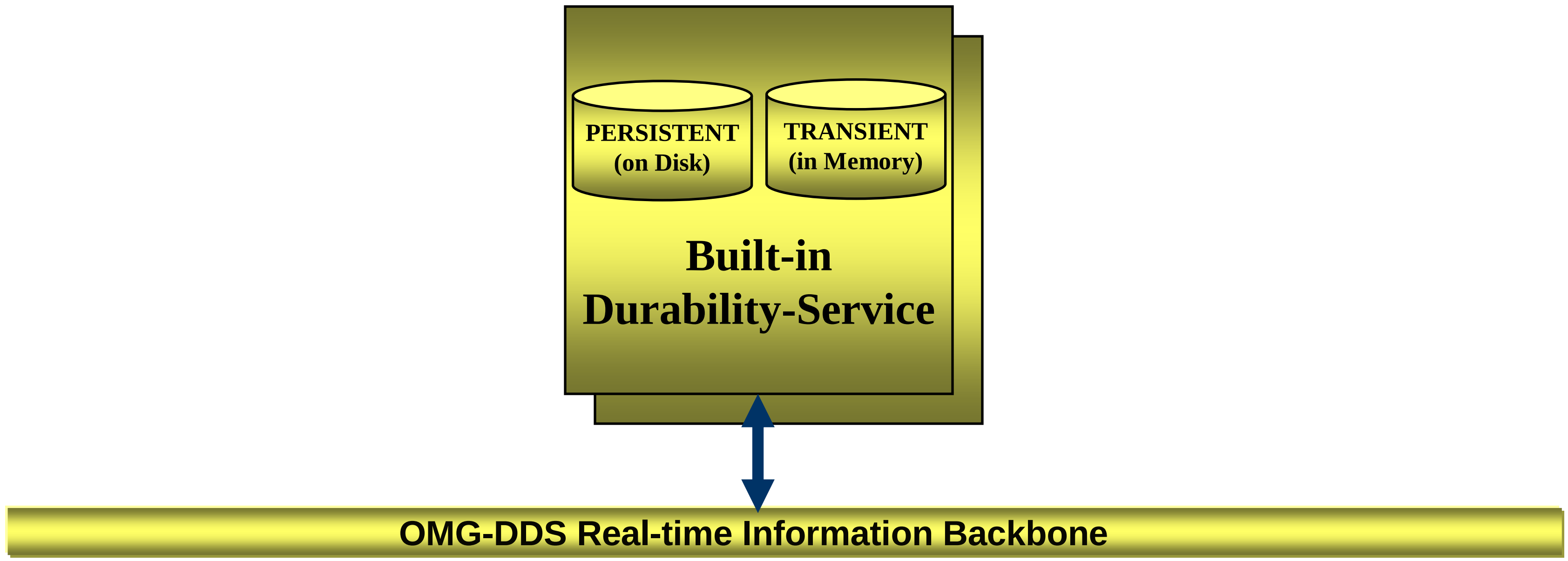
Characteristics

- Replicated publishers of data (with own ‘strength)
- Only highest-strength will be received
- On failure, next highest-strength will ‘take-over’

Features / Advantages

- Fault-tolerance by replication
- Notes:
  - Requires a lot of resources
  - Quality must be expressible as an ‘integer’

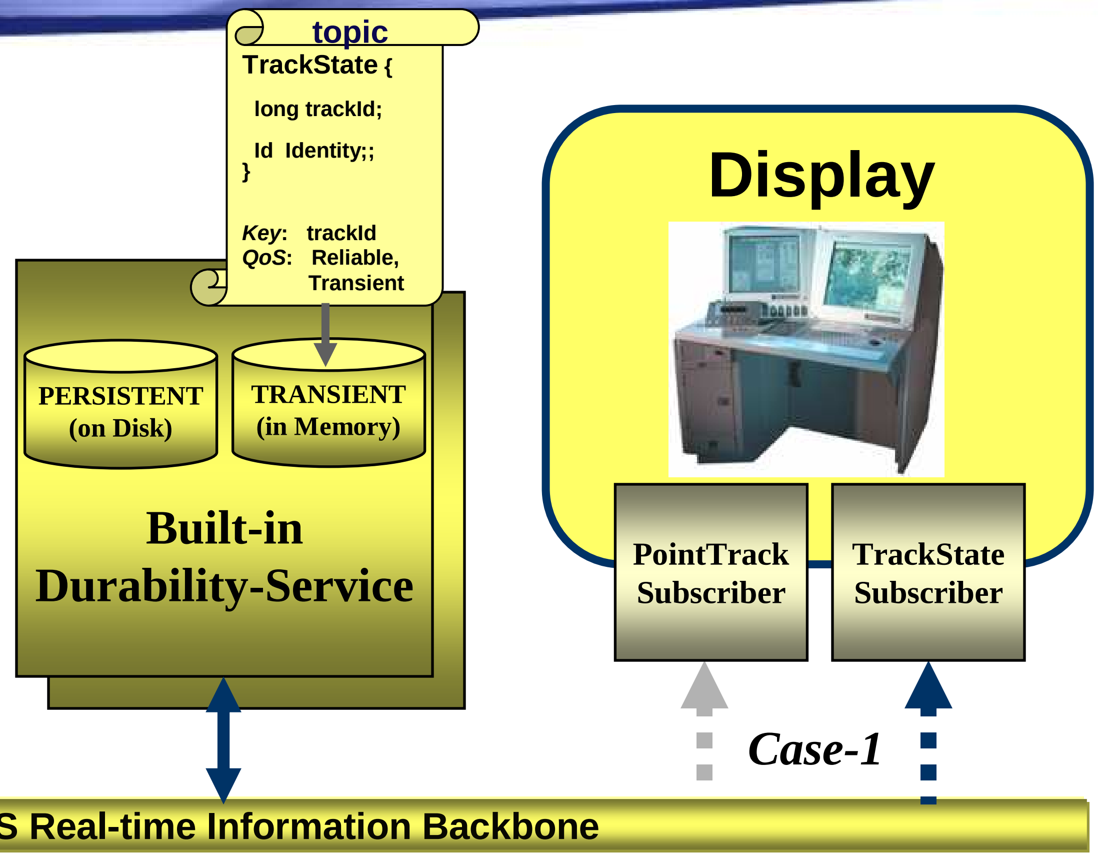
DDS™ by Example: "*The Persistence Profile …*"



Characteristics

- Built-in persistence for non-volatile data
  - State preservation for transient publishers
  - Settings persistence surviving system downtime
- Replicated durability service for maximal fault-tolerance

DDS™ by Example: *“The Persistence Profile …”*



Characteristics

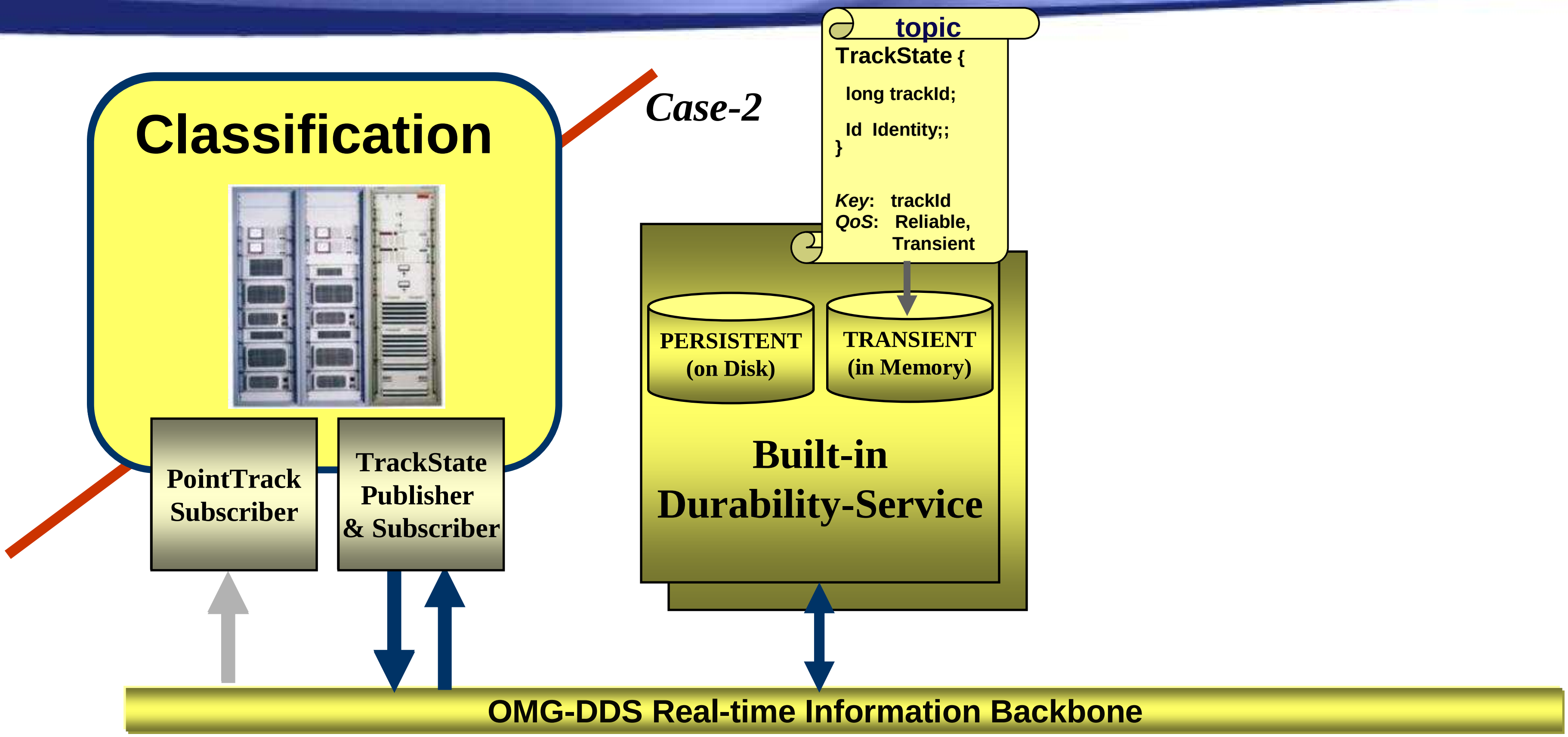
- Built-in persistence for non-volatile data
  - State preservation for transient publishers
  - Settings persistence surviving system downtime
- Replicated durability service for maximal fault-tolerance

Features / Advantages

- **Case-1**: late-joining of Display process
  - Previously produced TrackStates readily available
- **Case-2**: restart of failed Classification process
  - Internal state (already classified tracks) regained

PRISMTECH
Productivity Tools & Middleware

DDS™ by Example: *"The Persistence Profile …"*



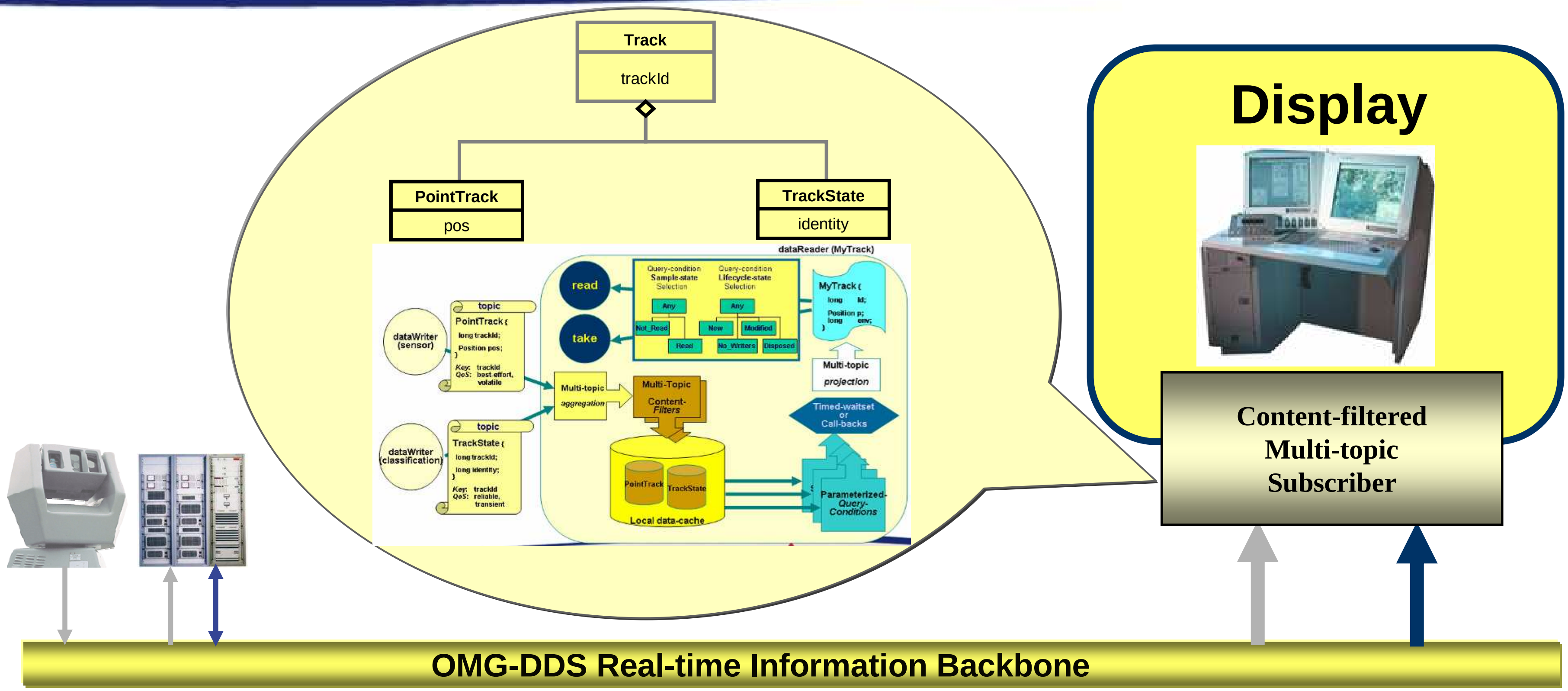
Characteristics

- Built-in persistence for non-volatile data
  - State preservation for transient publishers
  - Settings persistence surviving system downtime
- Replicated durability service for maximal fault-tolerance

Features / Advantages

- **Case-1**: late-joining of Display process
  - Previously produced TrackStates readily available
- **Case-2**: restart of failed Classification process
  - Internal state (already classified tracks) regained

DDS™ by Example: “*The Content Subscription Profile …*”



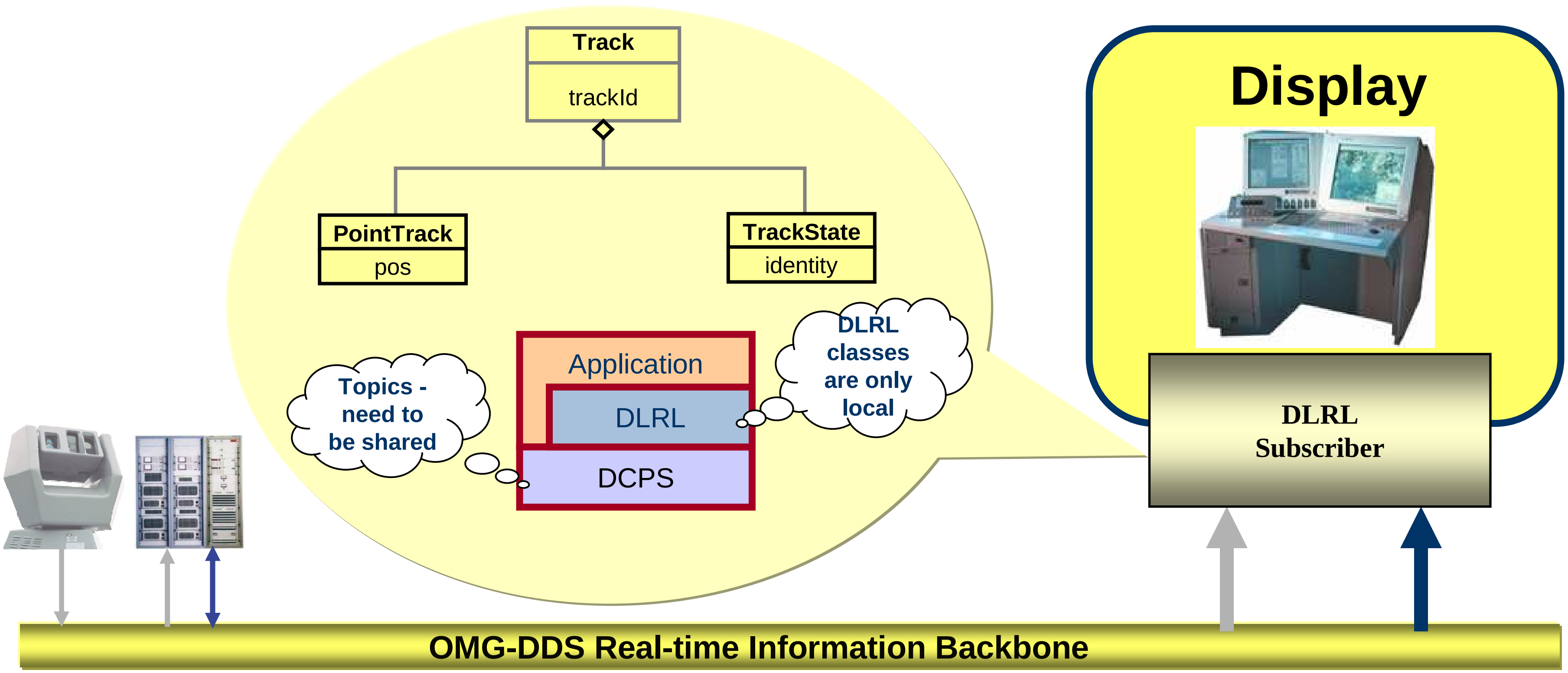
Characteristics

- Adds ‘content awareness’
  - Content-filtered Topics & query-conditions
- Supports ‘compound interest’
  - Multi-topics (combine/filter/re-arrange topics)

Features / Advantages

- Reduced application complexity
  - SQL-like querying and reconstitution of related data
- Improved system performance
  - Only receive/process what is of interest

DDS™ by Example: “*The DLRL Profile …*”



Characteristics

- **Local** Object Oriented Data-Access Layer
- Supports ‘OO’ features:
  - Inheritance, aggregation, composition
- Uses DCPS to distribute **state** by ‘mapped topics’

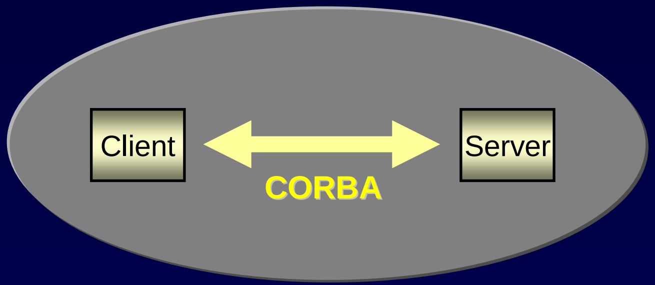
Features / Advantages

- Ease of Management of (related) data
  - Object oriented ‘graphs of objects’ (value-types)
- Supports ‘native language constructs’ (I.e. navigation)
  - Automatic ‘change-management’ of objects

CORBA & DDS integration

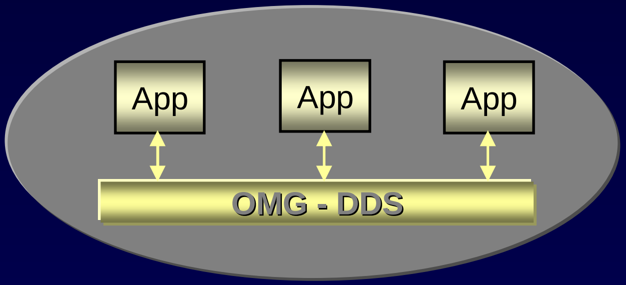
CORBA & DDS

Server Behavior



CORBA

Data Availability

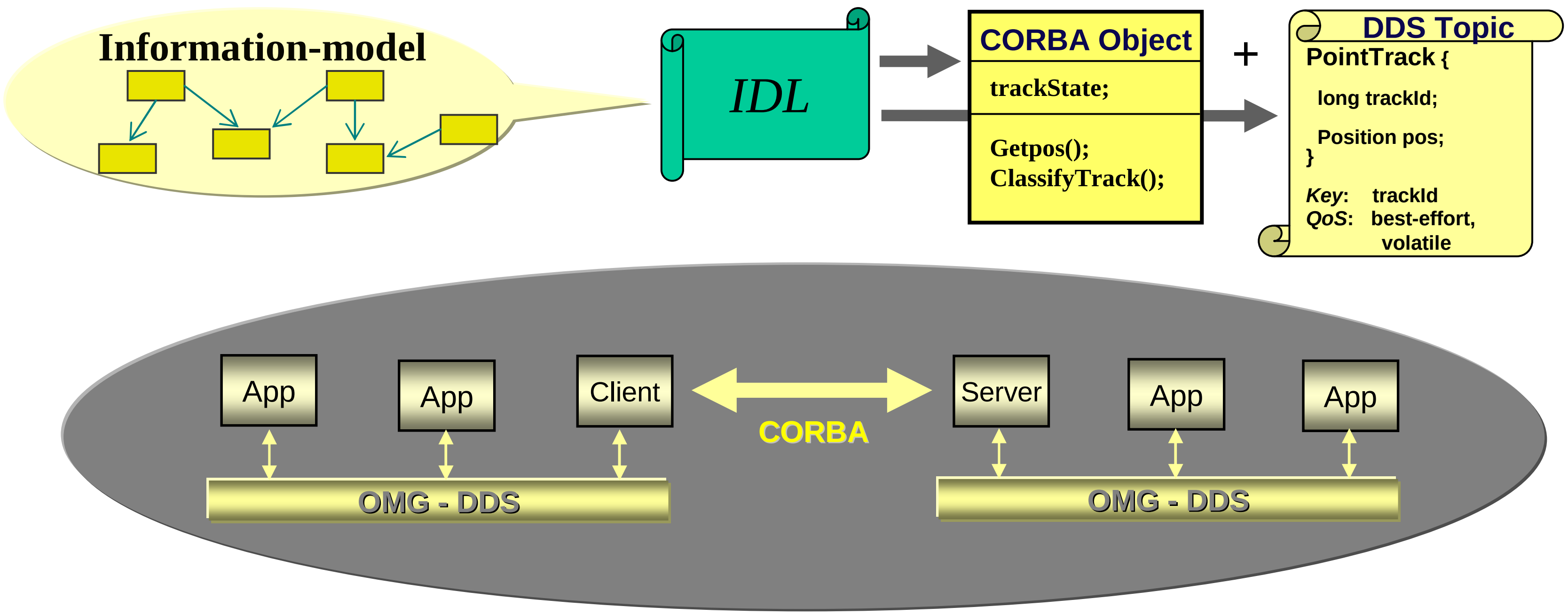


DDS

- Distributed *objects*
  - **Model: Client/Server**
  - **Remote Method Invocation**
  - **Reliable Communication**
- Purpose:
  - **Distributed Processing**
  - **Synchronous Transactions**

- Distributed *data*
  - **Model: Publish/Subscribe**
  - **Distributed Information Access**
  - **Fine-grained QoS**
- Purpose:
  - **Real-time data distribution**
  - **Fault tolerant Info Management**

CORBA & DDS DDS: Combining Strength



Characteristics & Benefits

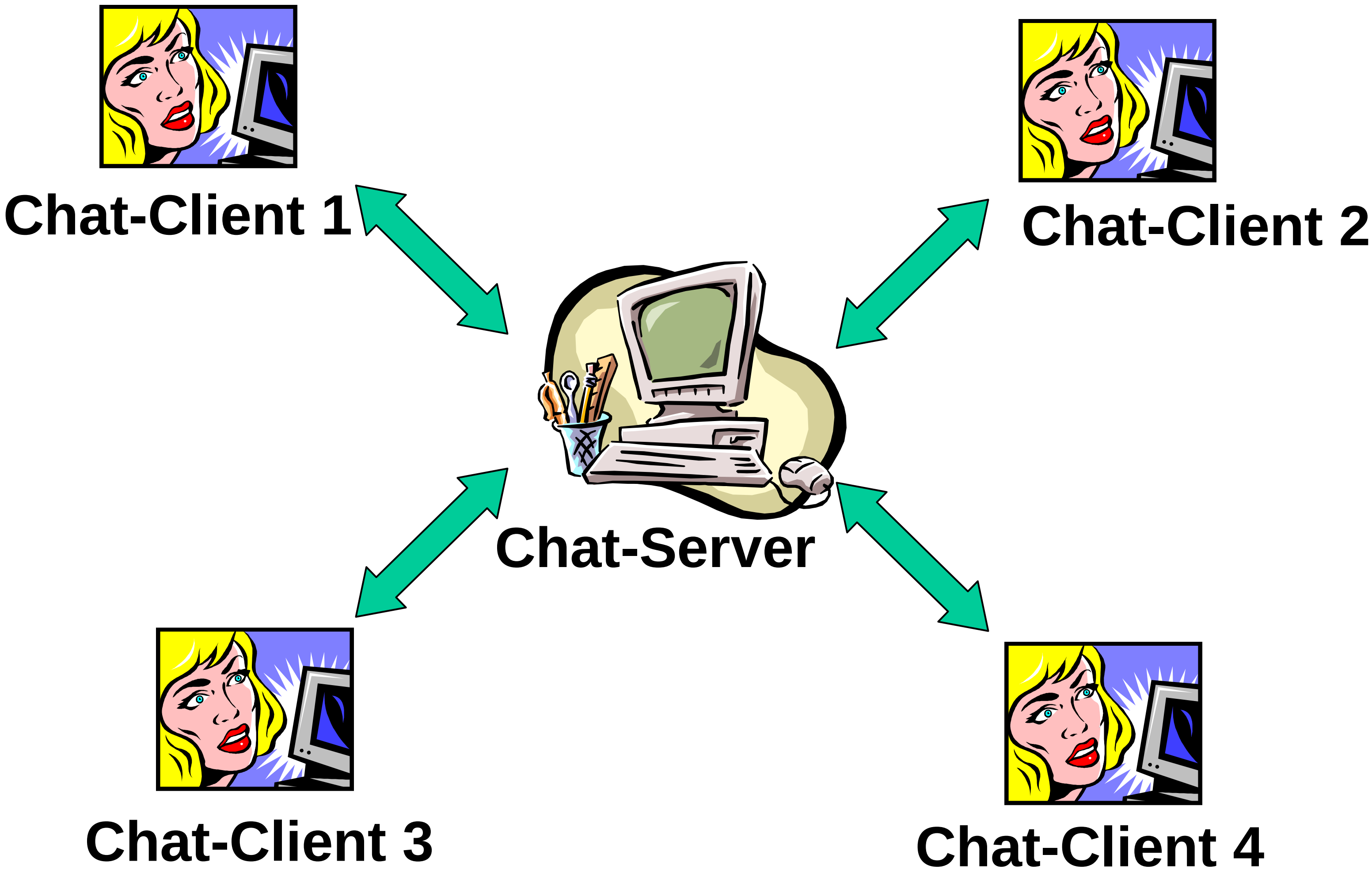
- **Corba/DDS Common Definition language: IDL**
  - Type definition for CORBA-interfaces & DDS topics
  - Code generation : Type-generation as well as generated (typed-)interfaces
- **Potential seamless Runtime Cooperation**
  - Shared types allow direct passing-on of RPC-obtained information into DDS-topics
  - Autonomous runtime-systems (ORB and DDS libraries)

PrismTech

Productivity Tools & Middleware

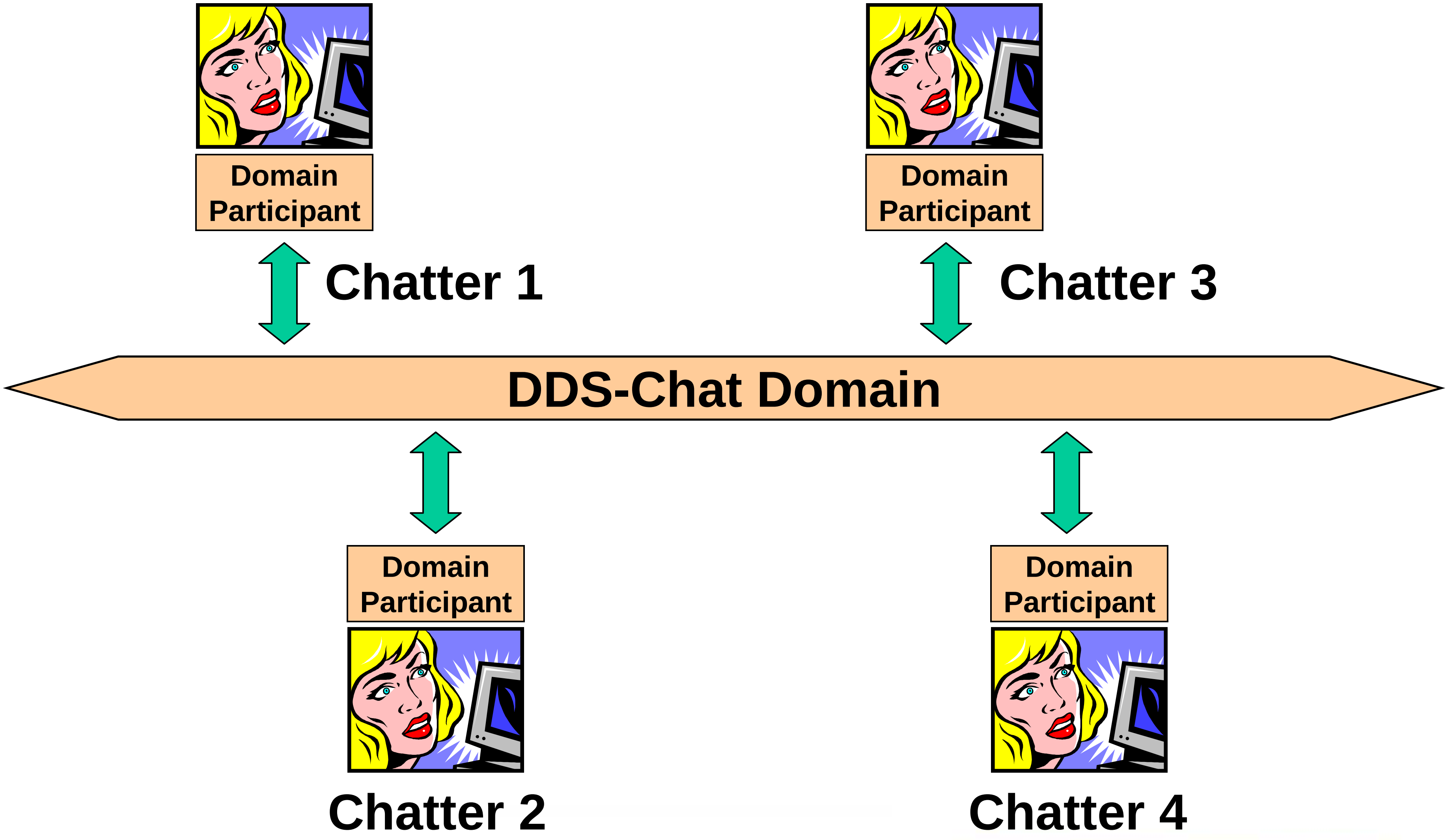
CHATROOM EXAMPLE

Exercise – Traditional Chat architecture



PrismTech
Productivity Tools & Middleware

Exercise – DDS based Chat architecture



Chatroom example: Sequence of events

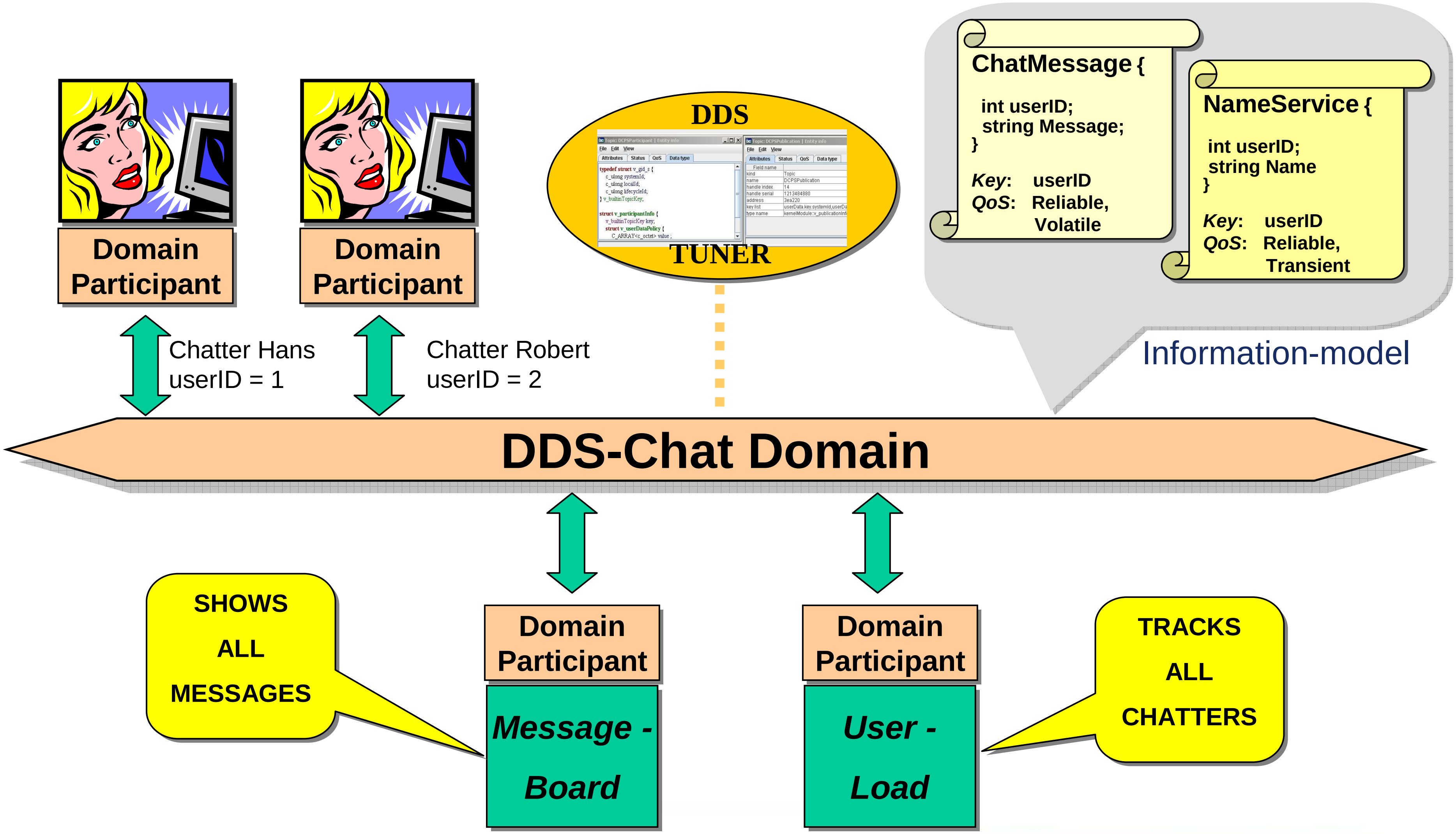
Typical sequence of events on a **traditional** Chat-application:

- **Connect** to the Chat-Server.
- **Transmit** your identity.
- **Download** the identities of the other chatters.
- **Receive** chat messages from other users.
  - These messages are **forwarded** to you by the server.
- **Write** your own chat-messages.
  - These messages are **forwarded** by the server to all the other users.

Typical sequence of events on a **DDS-based** Chat-application:

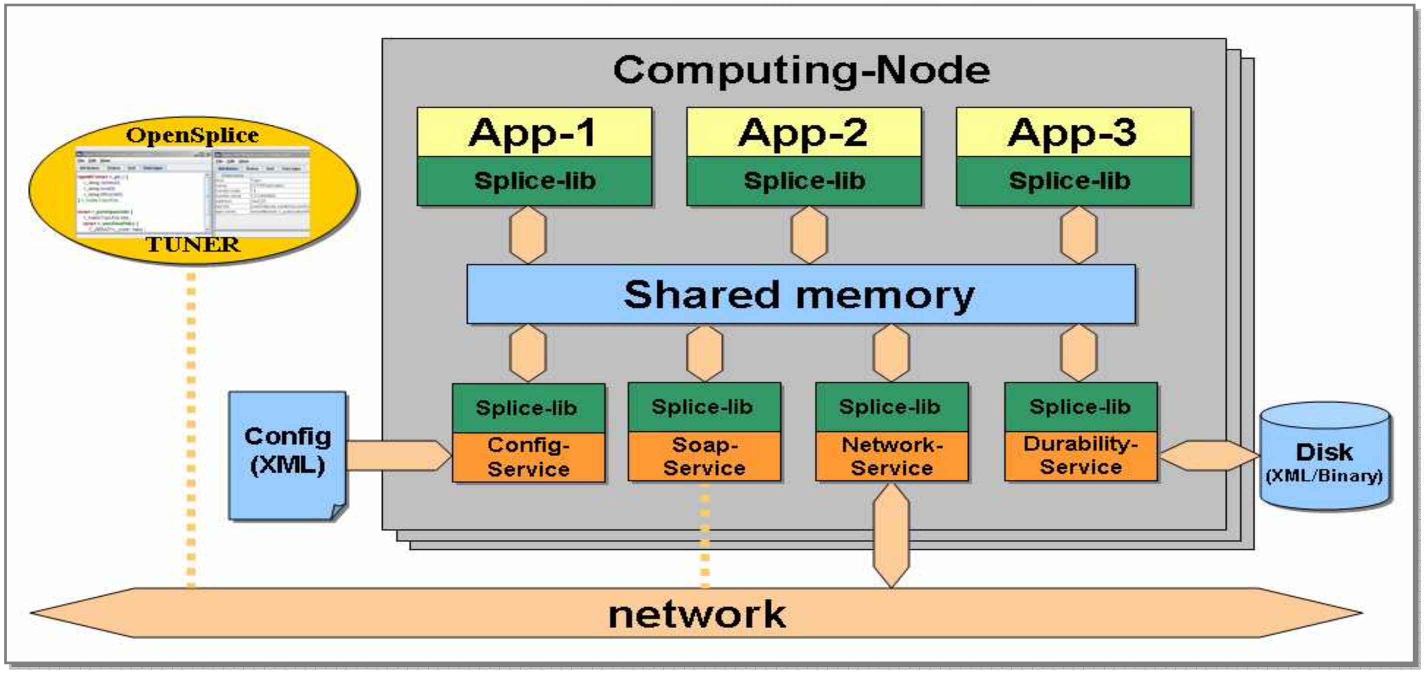
- **Participate** in the Chat-domain.
- **Publish** your identity.
- **Subscribe** yourself to the identities of all other chatters
- **Subscribe** yourself to all chat-messages in the Chat-domain.
  - All messages are delivered to you **directly** by their respective writers.
- **Publish** your own chat-messages.
  - Your messages are **directly** delivered to all the other interested users.

DEMO: DDS-based Chatroom



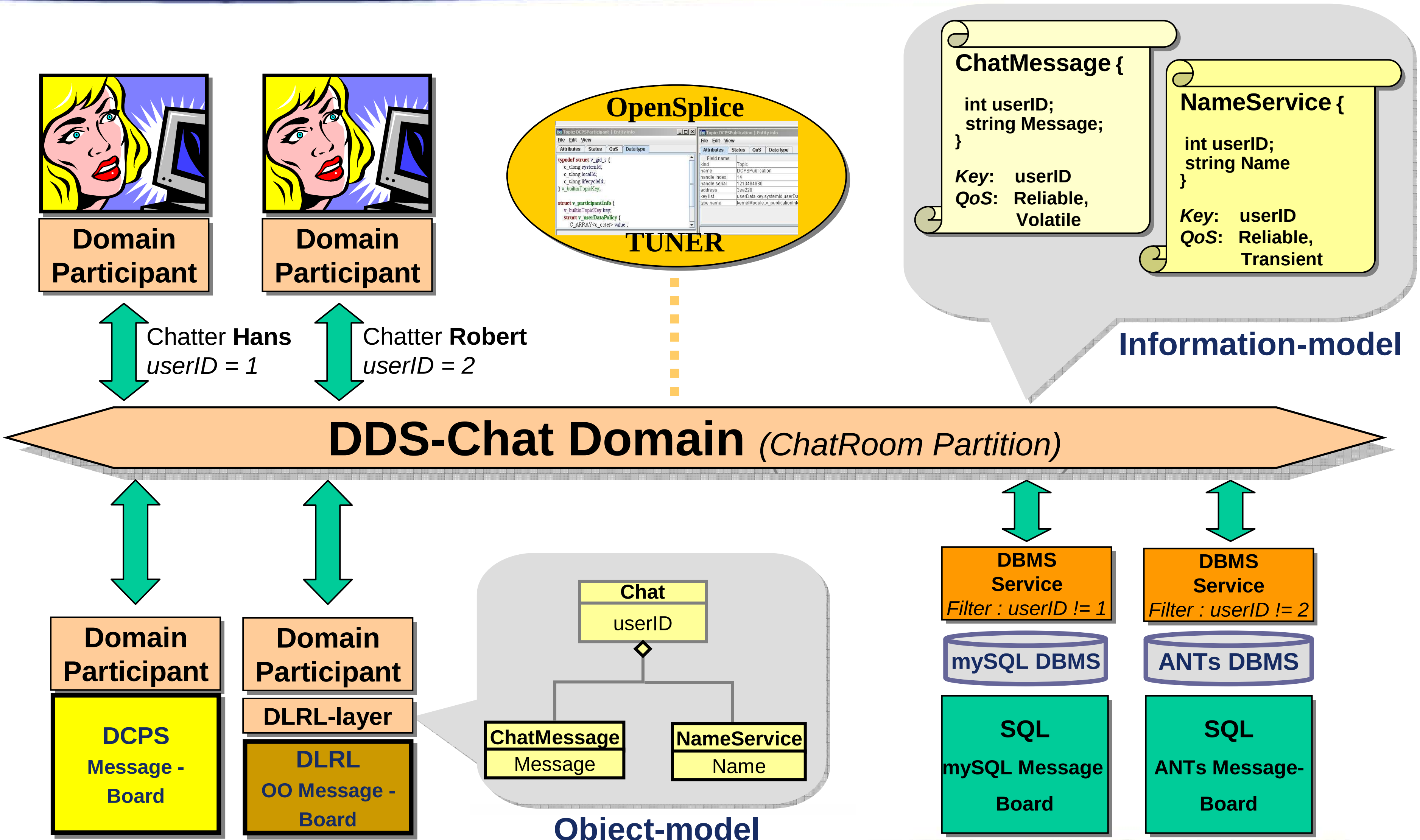
PrismTech
Productivity Tools & Middleware

Demo DDS architecture: “OpenSplice” example



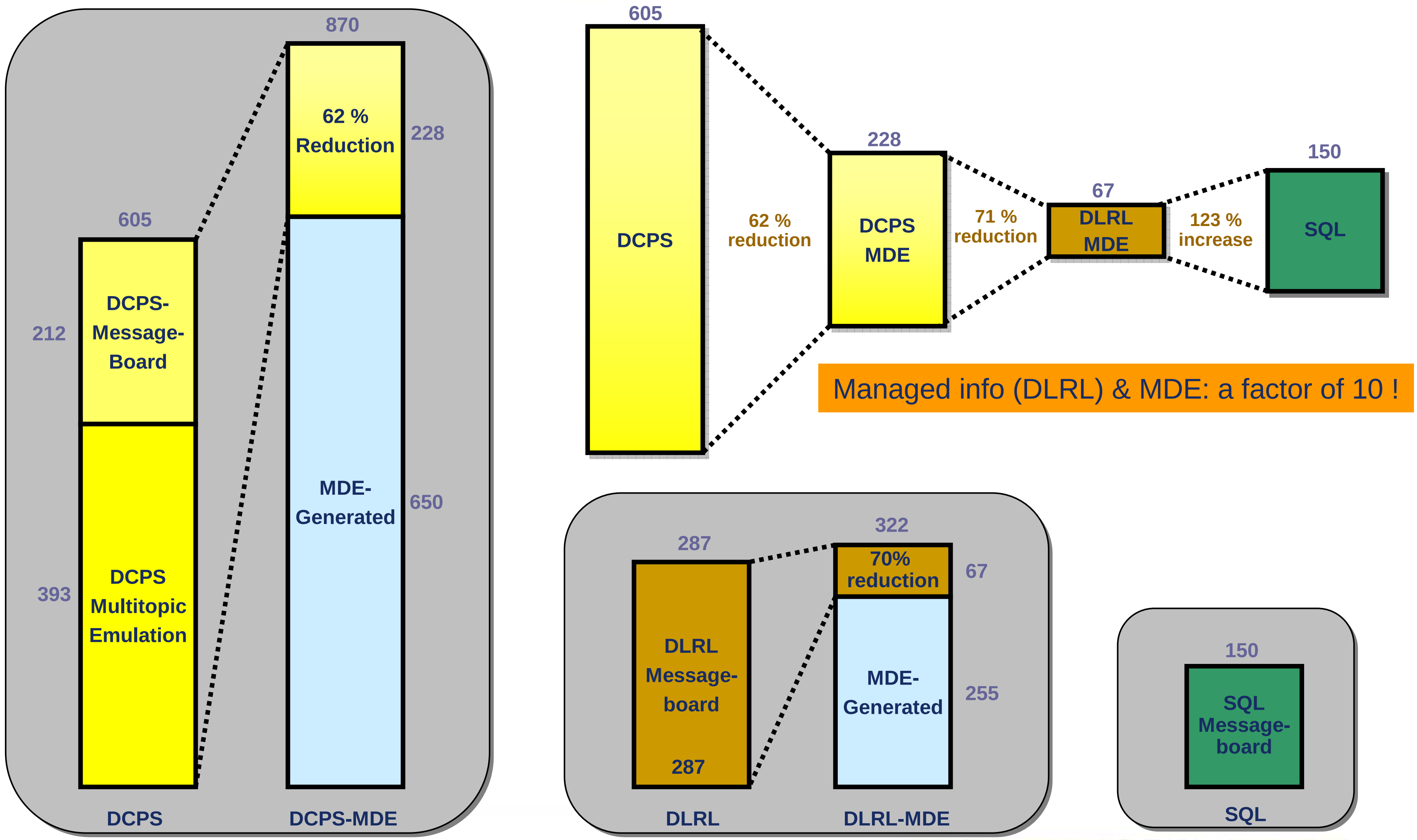
- **Choices: *Scalability & Efficiency***
  - **Single shared library for applications & services** *(code-footprint)*
  - **Ring-fenced shared memory segment** *(single copy regardless of nr. of applications)*
  - **Data urgency driven network-packing** *(Latency_budget QOS drives packing per channel)*
- **Choices: *Determinism & Safety***
  - **Pre-emptive network-scheduler** *(priority pre-emptive network-threads per priority-band with traffic-shaping)*
  - **Data importance based network-channel selection** *(Transport_Priority QoS of actual data)*
  - **Partition based multicast-group selection** *(dynamic mapping of logical DDS partitions)*
  - **Managed critical network-resource** *(limited impact/damage of faulty-applications)*

DEMO: DDS-based Chatroom: DCPS, DLRL .. and even SQL



PRISMTECH Productivity Tools & Middleware

MDE and DLRL: "*Messageboard* example", Reducing Complexity !!



Managed info (DLRL) & MDE: a factor of 10 !

PRISMTECH
Productivity Tools & Middleware

PrismTech

Productivity Tools & Middleware

CHATROOM LIVE DEMO

PRISMTECH
Productivity Tools & Middleware

Q&A