Распределенные информационные системы

Технология DDS

Вопросы

- Общие положения
  - Стандарты
  - Общие особенности
  - Области применения
- Архитектура
  - Основные подходы
  - Парадигма «подписка-публикация»
  - Особенности архитектуры (DCPS)

Стандарты

Общие особенности

Области применения

ОБЩИЕ ПОЛОЖЕНИЯ

Источники и составные части

- The Data Distribution Service for Real-Time Systems (DDS) – стандарт OMG для middleware
- Доступные стандарты:
  - Data Distribution Service for Real-time Systems (DDS). 2007 Version 1.2
  - The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification (DDSI). Version 2.1
  - DDS for Lightweight CCM
  - DDS-Java: Java 5 Language PSM for DDS
    - PSM – Platform specific mapping
  - DDS-PSM-Cxx: ISO/IEC C++ 2003 Language DDS PSM, включает
    - API
    - QoS
  - Extensible and Dynamic Topic Types for DDS (DDS-XTypes)
  - UML Profile for Data Distribution

Особенности

- Декларируемые
  - Масштабируемость
  - Работа в реальном времени
  - Надежность
  - Высокая производительность
- Существенные и практически значимые
  - Интероперабельность
    - На уровне разных платформ (ОС + «железо»)
    - Разные языки программирования
    - Единый транспортный протокол → на уровне разных производителей DDS
  - Поддержка парадигмы «подписка-публикация»

Области применения: общее – 1/3

- Приложения Big data:
  - financial trading
  - Управление воздушным движением
  - smart grid management
- Приложения на мобильных устройствах (типа планшетов и смартфонов)
  - Военные приложения
- Приложения на встроенных устройствах:
  - Транспортные системы и транспорт
  - Медицина
- DDS предполагается использовать как базовую технологию для реализации IoT

Области применения: война – 2/3

Boeing
AWACS

E2C Hawkeye

Boeing
Future Combat
Systems

Raytheon
SSDS

Lockheed
AEGIS

Insitu
Unmanned
Air Vehicles

Области применения: мир – 3/3

WiTronix
Train and vehicle
Tracking

Schneider Electric
Industrial Automation

Tokyo Japan
Traffic Control

Kuka
Robotics

EU and US
Air Traffic
Management

Varian
Medical Instruments

Основные подходы

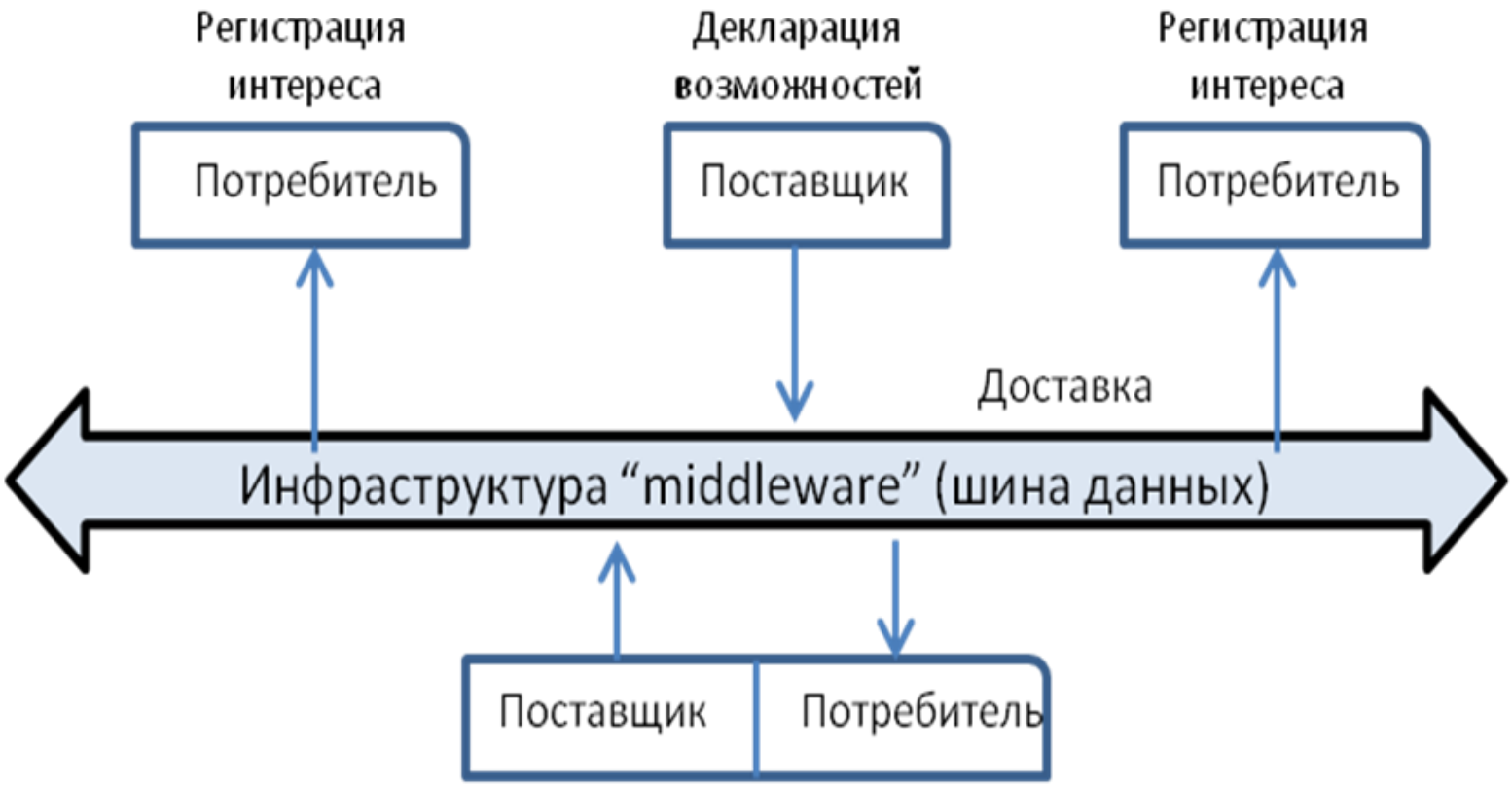
Особенности архитектуры (DCPS)

АРХИТЕКТУРА

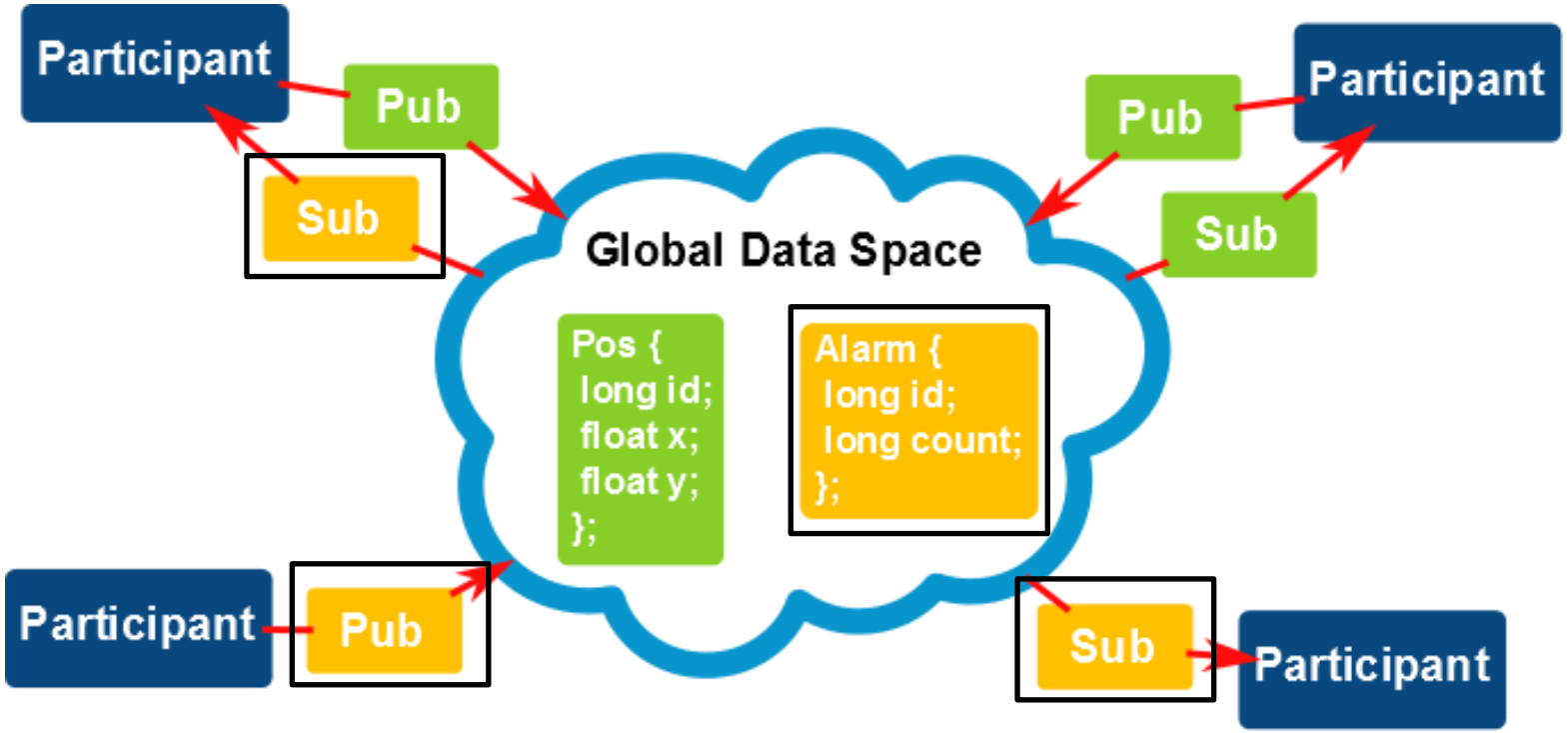
Основные подходы

- Поддержка парадигмы «подписка-публикация»
- Концепция «Global Data Space»
  - Циркулирующие данные представляют характеристики некоего виртуального пространства
- Дублирование источников данных

DDS – 1/2


Регистрация интереса
Декларация возможностей
Регистрация интереса
Потребитель
Поставщик
Потребитель
Доставка
Инфраструктура “middleware” (шина данных)
Поставщик
Потребитель

DDS – 2/2


Participant
Pub
Sub
Pub
Participant
Sub
Global Data Space
Pos {
long id;
float x;
float y;
};
Alarm {
long id;
long count;
};
Participant
Pub
Sub
Participant

Особенности архитектуры

- Размещение компонентов
  - Автоматическое обнаружение обеспечивает независимость от топологии сети
- Избыточность
  - Можно ввести дублирующих поставщиков и потребителей данных с указанием приоритетов и «передачей эстафеты»
- Время
  - Прием данных не обязательно синхронен с передачей. Есть возможность получения данных, относящихся ко времени, когда подписчик еще даже не был включен в сеть
- Платформа
  - Представление данных автоматически платформо-независимо («железо», ОС, язык программирования)
- Реализация
  - Сетевой протокол является стандартом

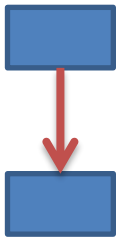
Парадигмы обмена данными в РИС

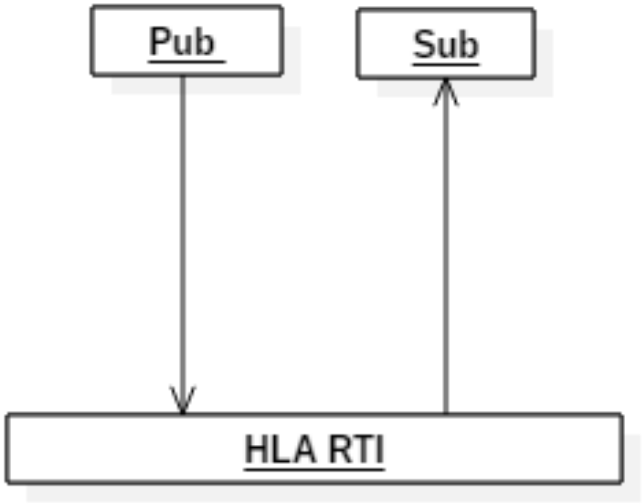
- Point-to-point
  - Каждый компонент «знает» кому нужные его данные и отправляет (или адресует их непосредственно)
- Client-server
  - Клиент обращается к серверу за нужными ему данными
- Publish-subscribe
  - Компонент, порождающий данные, сообщает об этом middleware (публикует данные), затем отправляет их в middleware не заботясь о наличии и расположении компонентов, нуждающихся в этих данных
  - Компонент, нуждающийся в данных, сообщает об этом middleware (подписывается на данные). При наличии публикатора данные данного типа начинают поступать

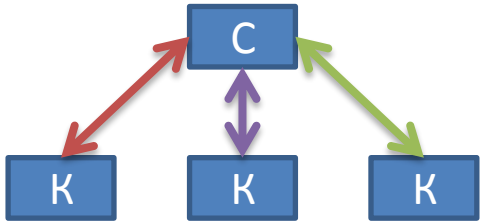
Pub-sub vs. Point-to-Point

Обмен данными 1:1
один публикатор и один подписчик

Повторяет семантику архитектуры «точка-точка»



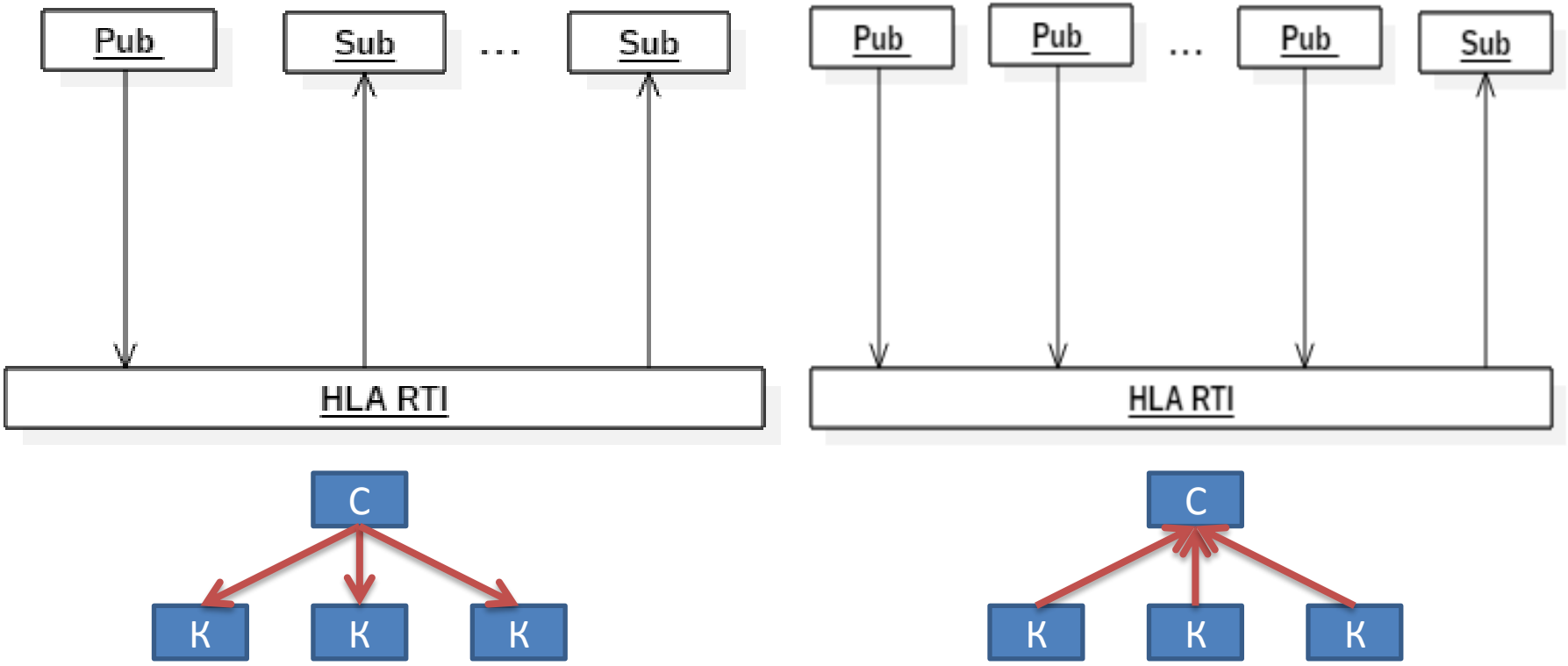




Pub-Sub vs. Client-Server

Обмен данными 1:N, M:1

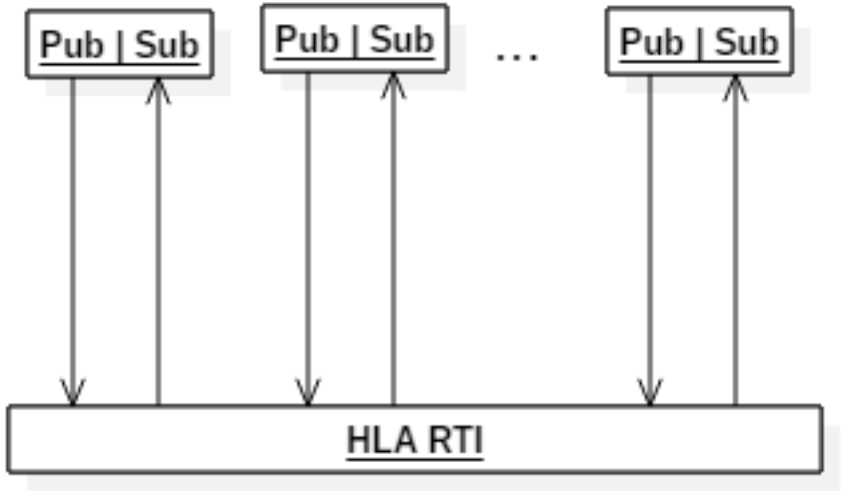
Семантика архитектуры «клиент-сервер»



Pub-Sub vs. others (?)

Обмен данными M:N
много публикаторов и много подписчиков

Прямых аналогов нет (непрямой – multicast группы)



Message-centric architecture

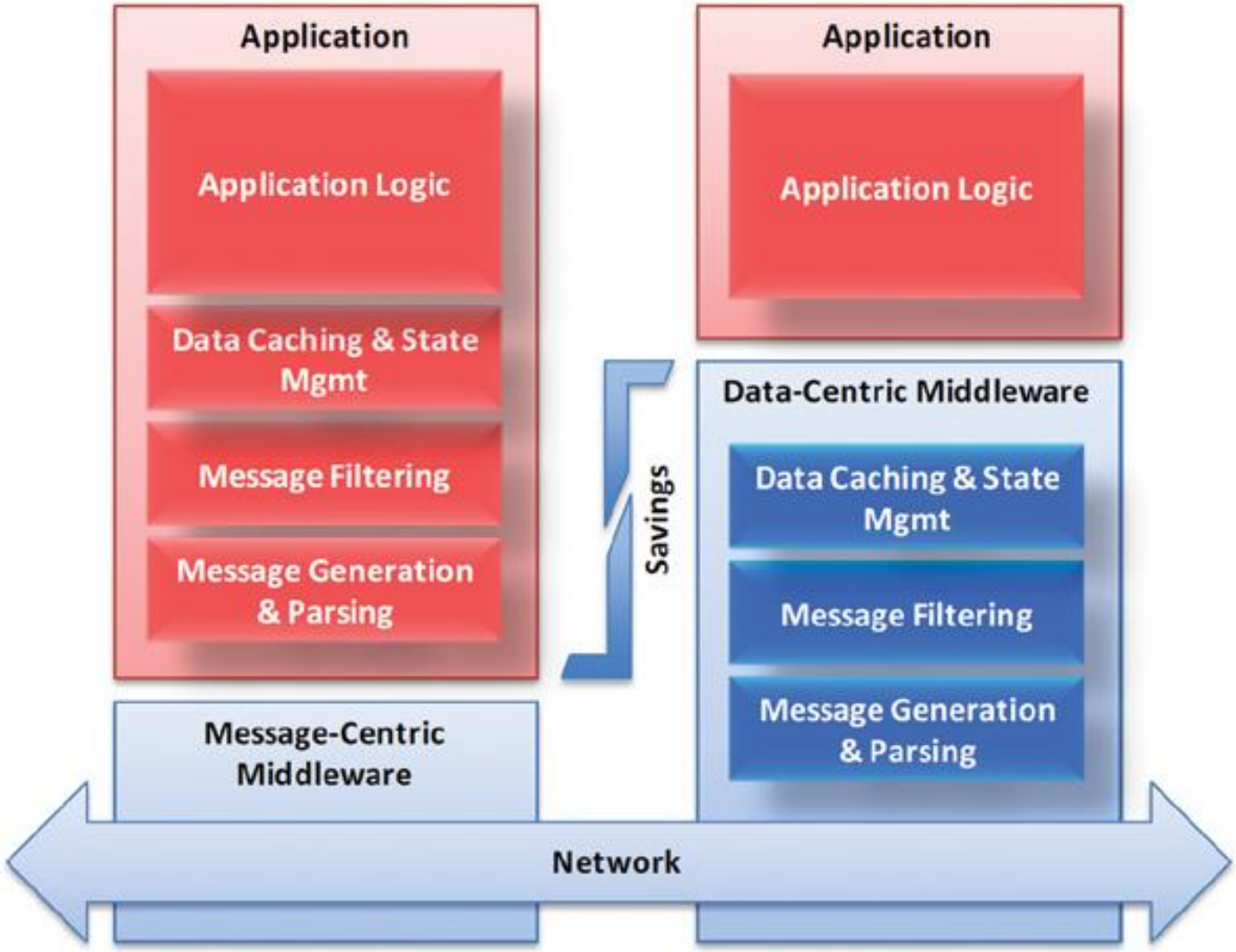
- Используется парадигма Publish-subscribe
- Middleware
  - Позволяет обмениваться сообщениями
  - Передает данные как наборы байт, не заботясь о содержимом
  - Не хранит состояние системы и/или данные
- Контроль синтаксической и семантической целостности информации возлагается на разработчика конечного приложения
- Наиболее известные реализации подхода:
  - JMS API
  - Протокол AMPQ
  - HLA RTI

Data-centric architecture

- Используется парадигма Publish-subscribe
- Middleware
  - Позволяет обмениваться сообщениями
  - Хранит состояние системы и данные
  - Передаваемые данные проверяются на синтаксическую [и семантическую?] целостность
- Разработчики создают ПО, которое читает и обновляет данные в глобальном виртуальном пространстве
- Наиболее известные реализации подхода:
  - DDS API
  - Протокол RTPS (DDSI)

«Эволюция middleware»

MCPS vs. DCPS – 1/2



MCPC vs. DCPS – 2/2

MCPS

- Middleware не «понимает» ваши данные
- Разработчику приходится реализовывать проверку синтаксической правильности и формировать/разбирать сообщения
- Можно выиграть на времени передачи

DCPS

- Middleware «понимает» ваши данные
- Разработчик «экономит» на проверке синтаксической правильности и формировании/разборе сообщений
- Постоянный контроль в отлаженных приложениях избыточен и поглощает ресурсы