Uma Introdução ao Data Distribution Service - DDS

Markus Endler
Rafael Vasconcelos
Lincoln David Silva

LAC
Laboratory for Advanced Collaboration

Pontifícia Universidade Católica do Rio de Janeiro

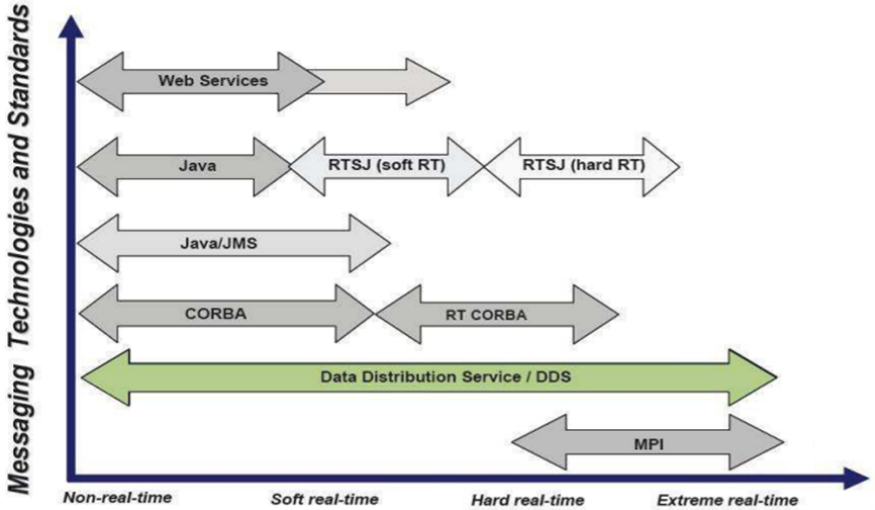
O que é DDS?

- É uma especificação da OMG que provê uma interface para desenvolvimento de aplicações distribuídas de tempo real
- Um modelo para comunicação assíncrona independente de plataforma e linguagem de programação
  - Windows, Linux, Android, Solaris,...
  - C, C++, Java, C#,...
- Interoperável com outros produtos DDS
- É essencialmente uma arquitetura P2P

O que é DDS?

- Objetivo de prover comunicação robusta e dinâmica para sistemas distribuídos e de tempo real
- Baseado no conceito de Publish/Subscribe de tempo real (RTPS – Real Time Publish Subscribe)
- Utiliza conceito de Data-Centric Publish-Subscribe
- Pub/Sub garante desacoplamento:
  - Temporal
  - Espacial e de referência

Comparação com outras tecnologias de comunicação



Aplicações

- US DoD - United States Department of Defense
  - Sistema de combate naval
  - Sistema de defesa
- EuroControl
  - Controle de tráfego aéreo
- Boing, Bell Helicopter, Sikorsky, US DoD, LAC
  - Net-Ready Applications for Rotorcraft

Aplicações

In this conceptual representation of a data-centric model, applications read and write data without requiring any information on the location of the data, transport protocol, operating system, and so on.

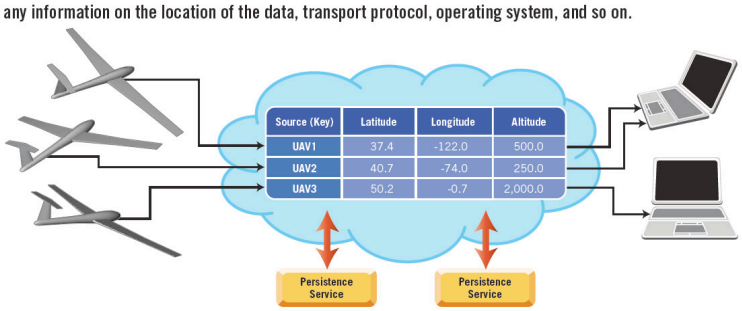
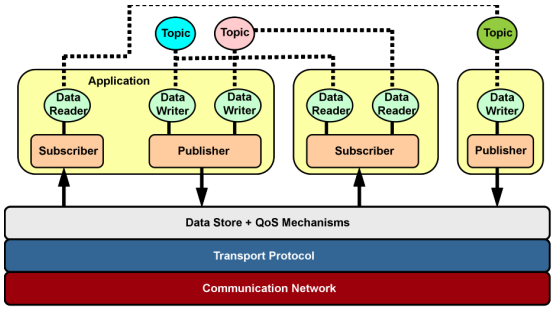


Figure 3

Arquitetura do DDS



- DataReaders e DataWriters se comunicam através de Tópicos
- Tópico possui {Nome, Tipo e ParâmetrosQoS}
- Data serão transferidos somente quando itens publicados por um DataWriter casam com a assinatura feita por um DataReader no mesmo tópico.

DDS in a nutshell

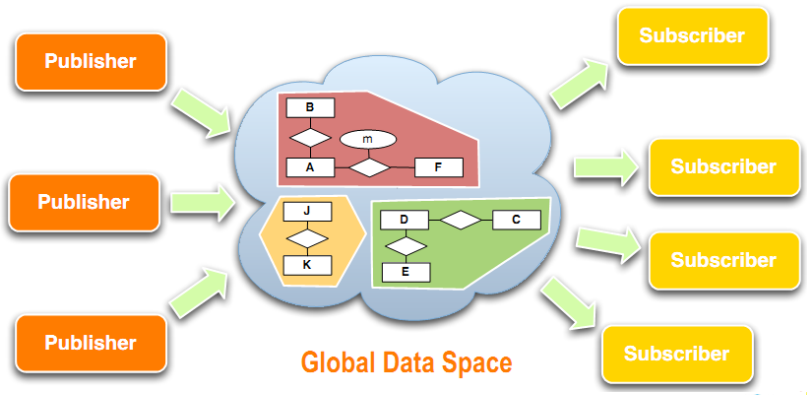
- DDS
  - Como todo Pub/Sub suporta comunicação one-to-one, one-to-many e many-to-many
  - Mensagens representam objetos
  - Arquitetura flexível e adaptável com suporte a descobrimento automático (de publicadores & assinantes)
  - Baixo overhead – visando sistemas de alta performance
  - Escalabilidade e uso eficiente da largura de banda
  - Grande gama de parâmetros de QoS: Ownership, History, Reliability, …
  - Middleware realiza cache de dados – Data Reader Cache
  - Conceito de Global Data Space e Domain

Politicas de QoS suportadas

USER_DATA	TIME_BASED_FILTER
TOPIC_DATA	PARTITION
GROUP_DATA	RELIABILITY
DURABILITY	TRANSPORT_PRIORITY
DURABILITY_SERVICE	LIFESPAN
PRESENTATION	DESTINATION_ORDER
DEADLINE	HISTORY
LATENCY_BUDGET	RESOURCE_LIMITS
OWNERSHIP	ENTITY_FACTORY
OWNERSHIP_STRENGTH	WRITER_DATA_LIFECYCLE
LIVELINESS	READER_DATA_LIFECYCLE

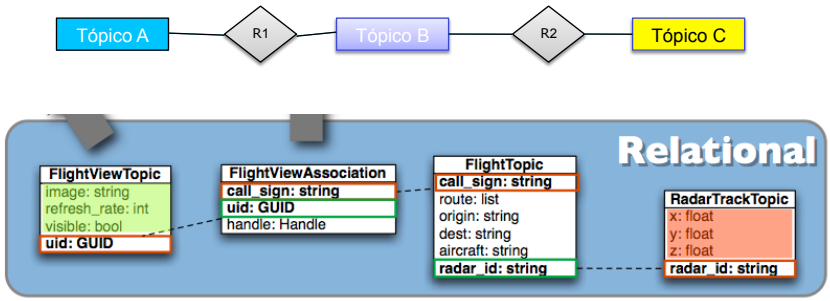
Global Data Space

- Domain é um subdivisão virtual do Global Data Space
- Partição é uma parte de um Domain (ajuda a isolar e otimizar a comunicação)



Modelo Centrado nos Dados

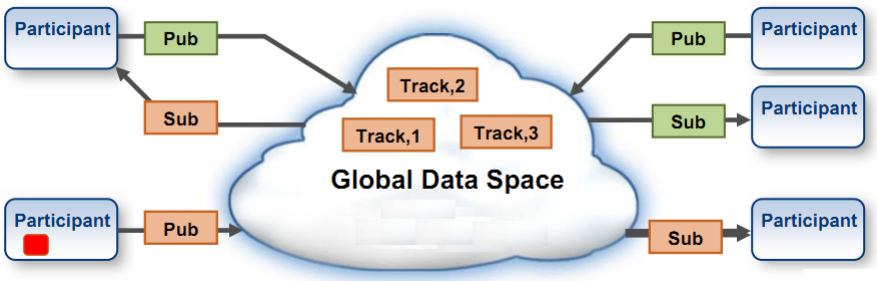
- Os Topicos são descritos em um modelo relacional (Diagramas Entidade-Relacionamento)



- Existe uma ferramenta que gera os DataWriters e DataReaders correspondentes

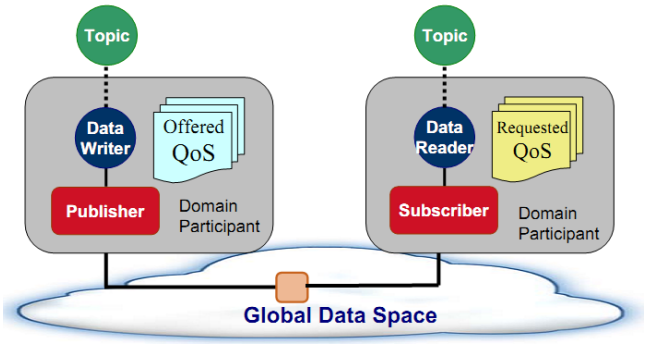
Funcionamento

- Subscrições são desacopladas das publicações
- Contratos estabelecidos por QoS
  - Possibilidade de ter várias configurações
- Descobrimento e configuração automáticos

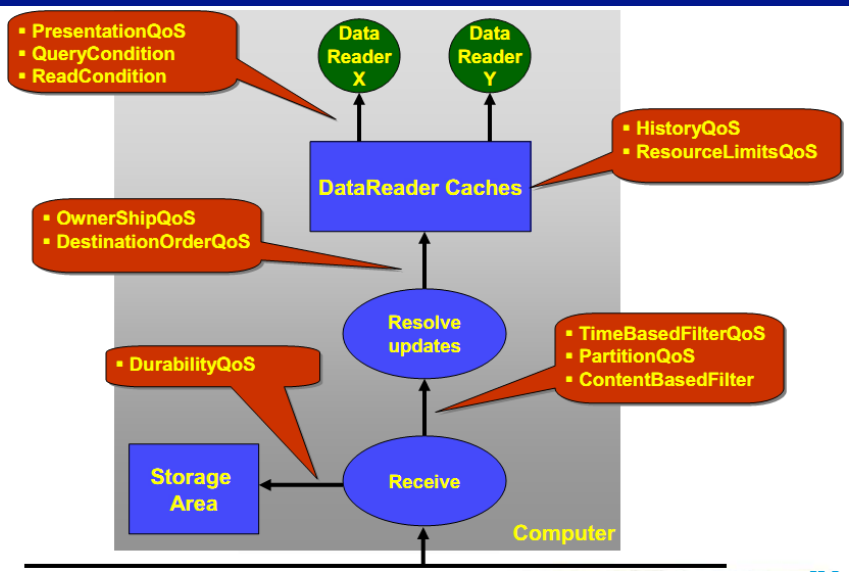


Funcionamento

- **DomainParticipant** – Permite aplicação entrar em um domínio
- **Publisher/Subscriber** – Envia/recebe objetos
- **DataWriter/Reader** – Envia/recebe um tópico específico
- **Topic** – Associa nome único e tipo de dado

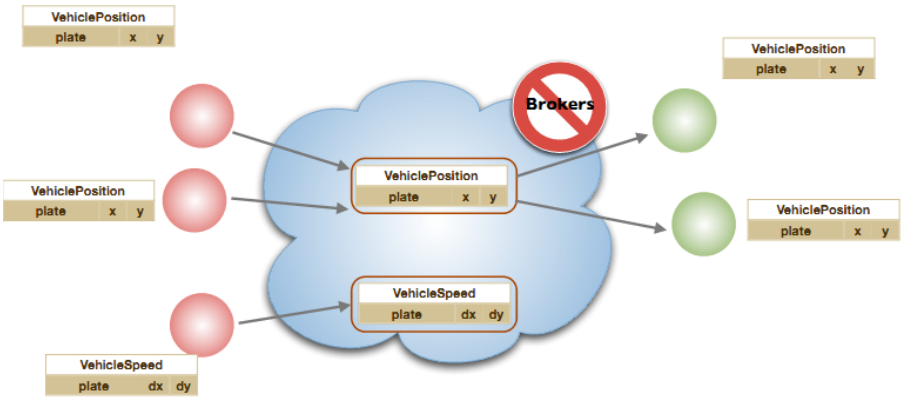


Politicas de QoS e Cache

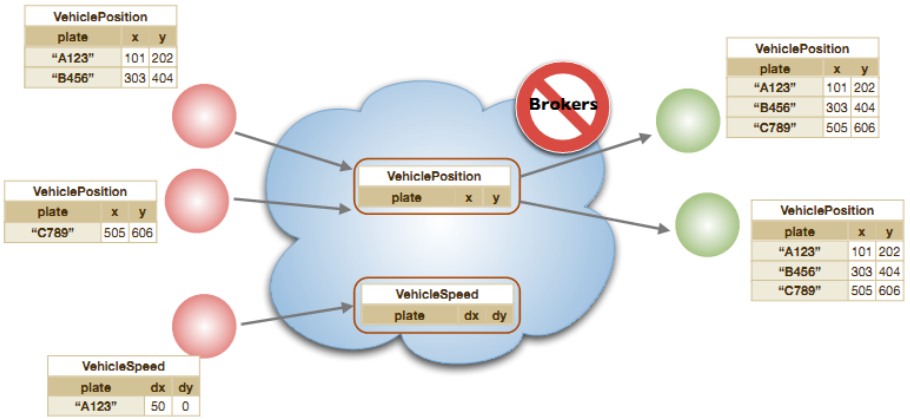


Funcionamento

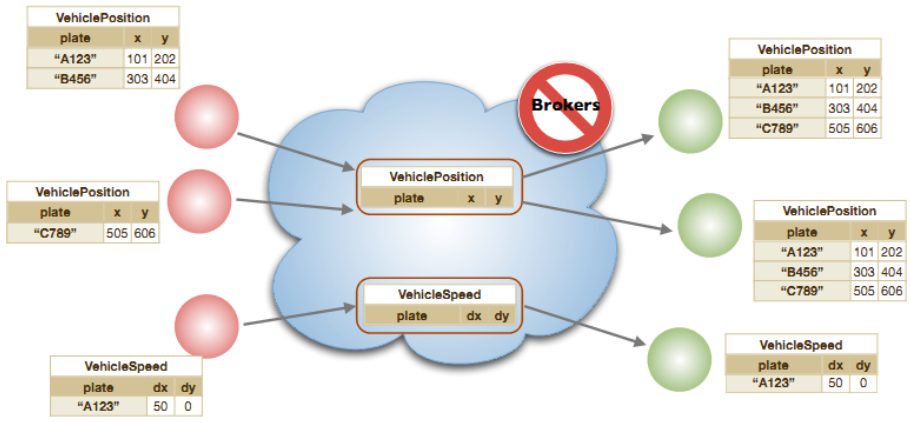
Funcionamento



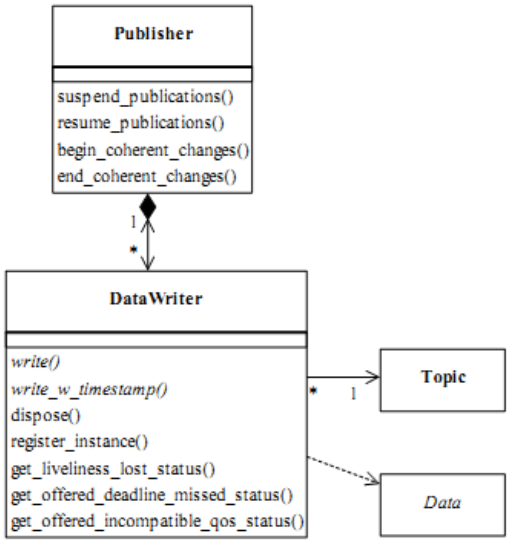
Funcionamento



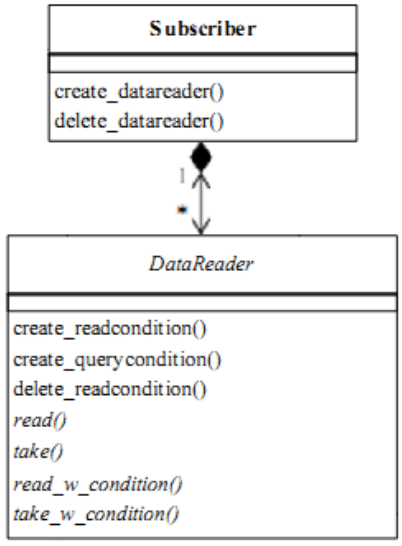
Funcionamento



Interface do Publisher

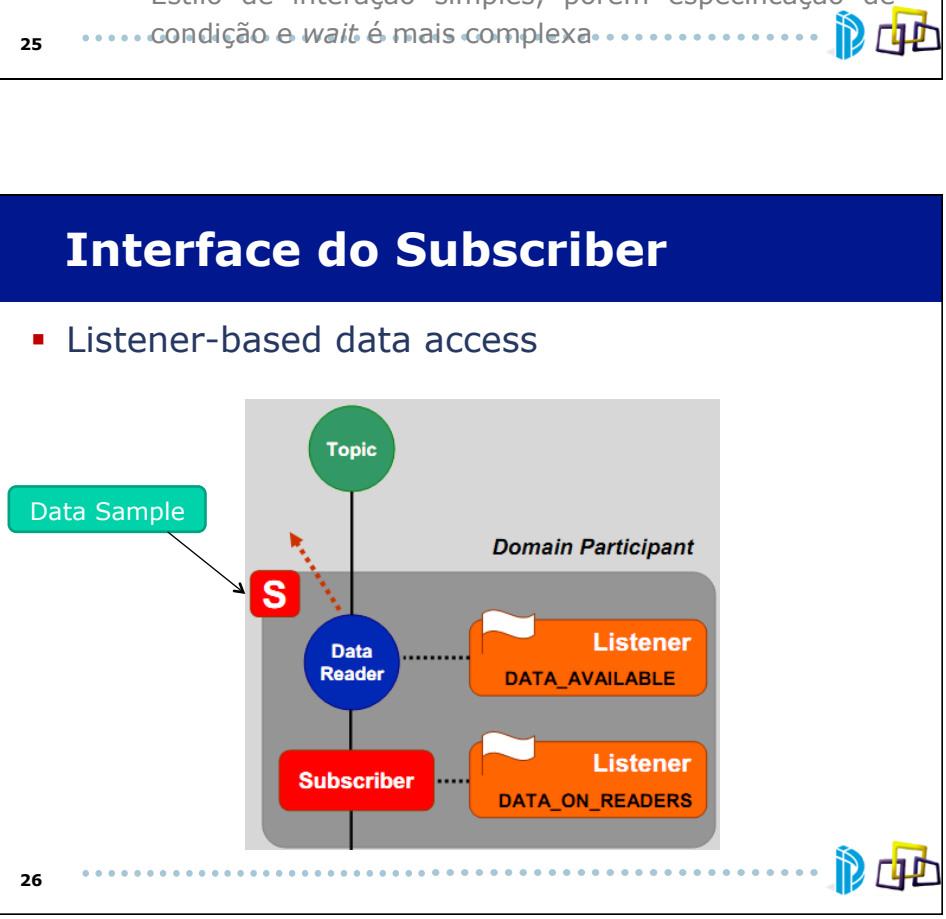


Interface do Subscriber



Interface do Subscriber

- Existem 2 estilos de interação distintos:
  - Listener-based data access
    - Participante é notificado por um *listener*
    - Middleware assincronamente informa o participante através dos métodos no *listener*
    - Simples e eficiente
  - Wait-based data access
    - Provê um conjunto de condições que as threads do participante podem usar para ficarem bloqueadas enquanto não há mudança
    - Semelhante ao uso de ***select*** em sockets
    - Estilo de interação simples, porém especificação de condição e *wait* é mais complexa

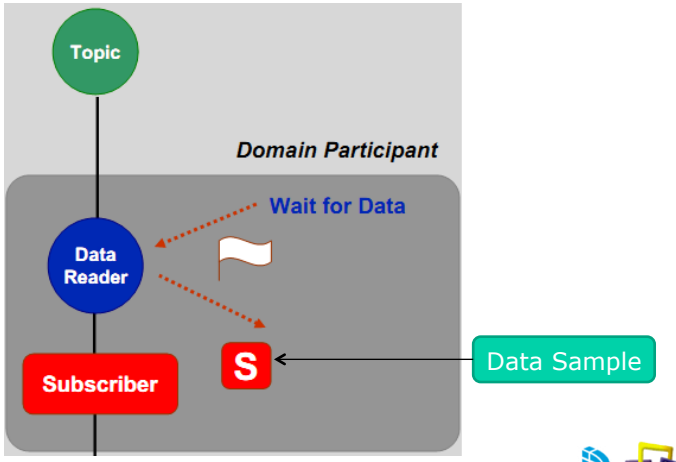


Interface do Subscriber

- Listener-based data access

Interface do Subscriber

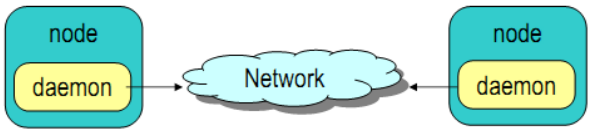
- Wait-based data access



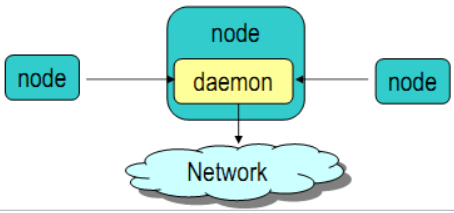
Arquiteturas DDS

- Decentralizada

- Federada

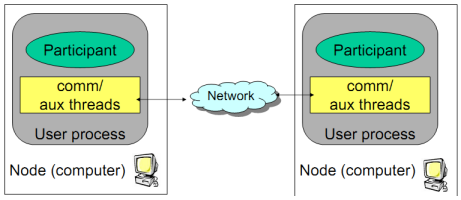


- Centralizada



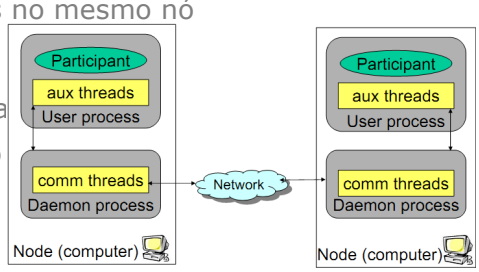
Arquiteturas DDS

- Decentralizada
  - Prós
    - Aplicação independente
    - Menor latência e jitter
    - Menos locais de configuração
    - Menos um ponto de falha
  - Contras
    - Aplicação mais complexa
    - Detalhes de configuração manipulados na aplicação
    - Dificuldade em fazer buffer de dados enviados entre aplicações no mesmo nó



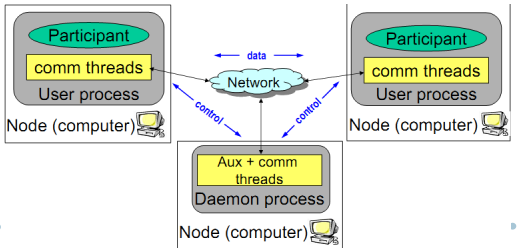
Arquiteturas DDS

- Federada
  - Prós
    - Maior número de participantes no mesmo nó
    - Priorização de mensagens entre aplicações
    - Simplifica a configuração da aplicação
    - Cache entre aplicações no mesmo nó
  - Contras
    - Mais um ponto de falha
    - Mais uma configuração
    - Maior latência e jitter



Arquiteturas DDS

- Centralizada
  - Prós
    - Simplicidade de configuração e implementação
  - Contras
    - Ponto de falha central
    - Potencial gargalo
    - Problema com escalabilidade



Principais produtos DDS

- Arquitetura Descentralizada
  - CoreDX
  - OpenDDS
  - RTI
- Arquitetura Federada
  - OpenSplice

Resumindo...

- Define uma arquitetura Data-Centric Publish-Subscribe para conectar participantes anônimos
  - Desacoplamento de espaço, tempo, fluxo, plataforma, multiplicidade
- Menor complexidade no desenvolvimento de aplicações distribuídas
- Possibilita tolerância a falhas, descobrimento dinâmico, subscrição com filtro, escalabilidade, comunicação em tempo real, processamento de eventos...
- Além de ser interoperável com outras soluções DDS

Referências

- OMG, Data Distribution Service for Real-Time Systems Specification, www.omg.org/docs/formal/04-12-02.pdf.
- A. Corsaro, Getting Started with DDS, PrismTech.
- G. Pardo-Castellote, OMG Data-Distribution Service: Architectural Overview, Proc. of 23th IEEE Int. Conference on Distributed Computing Systems Workshops, pp. 200-206, 2003
- G. Pardo-Castellote, B. Farabaugh, R. Warren. An Introduction to DDS and Data-Centric Communications, RTI.
- M. Xiong *et al.* Evaluating the Performance of Publish/Subscribe Platforms for Information Management in Distributed Real-time and Embedded Systems, Vanderbilt University.
- M. Xiong *et al.* Evaluating the Performance of Pub/Sub Platforms for Tactical Information Management, Proc. of SPIE, International Society for Optical Engineering,
- http://portals.omg.org/dds/presentations