



Powering Netcentricity

Leveraging DDS to Provide Real-Time Fault Tolerant CORBA

Date: 15th July 2008

Bob Kukura, PrismTech Solutions Americas

Jaiganesh Balasubramanian, Vanderbilt University

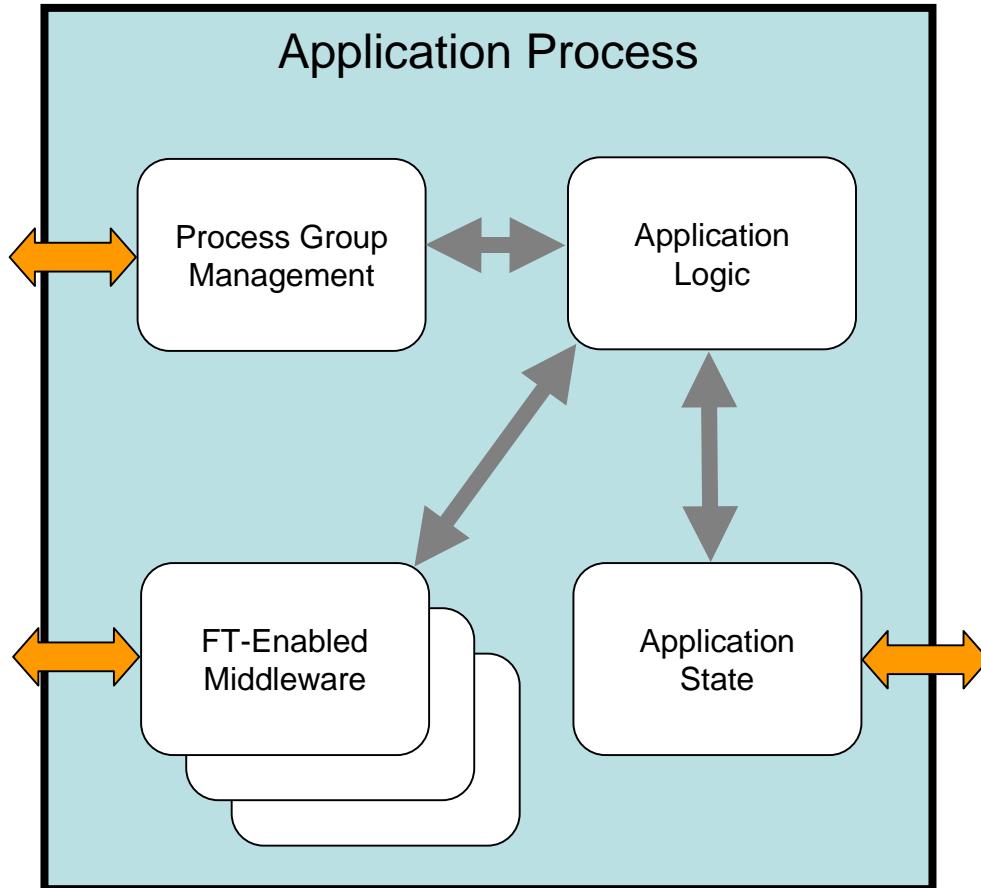
Nanbor Wang, Tech-X Corporation

- ▶ **Fault Tolerant CORBA**
 - ▶ Chapter 23 of CORBA 3.0.3 (formal/04-03-01)
 - ▶ Comprehensive
 - ▶ Complete FT platform
 - ▶ Lots of options
 - ▶ Not Real-Time
 - ▶ Not widely implemented
- ▶ **Lightweight Fault Tolerance for Distributed Real-Time Systems RFP**
 - ▶ RFP (realtime/06-06-06) in progress
 - ▶ Submissions from OIS, PrismTech, Raytheon, RTI, THALES
 - ▶ “Final” revised submissions due November 10, 2008
 - ▶ Focus on composing fault tolerant systems
 - ▶ FT-enabled distributed computing middleware, possibly multiple types
 - ▶ Process group management
 - ▶ Application state management
 - ▶ Keep it simple
 - ▶ Don't assume strictly deterministic behavior
 - ▶ Relax “exactly-once” semantics at middleware level
 - ▶ Manage at process granularity



Lightweight Fault Tolerant Application Architecture

3



- ▶ Application Logic
 - ▶ Business logic
 - ▶ Infrastructure Integration
- ▶ Application State
 - ▶ Initialization
 - ▶ Replication and synchronization
- ▶ Process Group Management
 - ▶ Identity and membership
 - ▶ Startup and shutdown
 - ▶ Replication role
 - ▶ Fault detection and recovery
- ▶ FT-Enabled Middleware
 - ▶ Communication between components
 - ▶ Fault detection
 - ▶ Fault isolation
 - ▶ Fault recovery



Copyright PrismTech 2008



- ▶ **CORBA RT FT Challenge**
 - ▶ Real-time CORBA is proven technology
 - ▶ Approaches to fault-tolerance are well understood
 - ▶ Solid foundation needed to build RT FT CORBA
- ▶ **DDS Capabilities**
 - ▶ **Decentralized Group Communication**
 - ▶ No single point of failure
 - ▶ Peer-to-Peer
 - ▶ Multicast
 - ▶ **QoS Policies**
 - ▶ Durability - Topic data can be made persistent
 - ▶ Reliability - Reliable or Best-effort messaging
 - ▶ Ownership / Ownership Strength - Semantics for multiple writers of same instance
 - ▶ Real-time - Priorities, latency budgets, deadlines, etc.
- ▶ **DDS Applicability to LwFT4DRTS**
 - ▶ Application State Management
 - ▶ Process Group Management
 - ▶ **CORBA Fault Tolerance**
 - ▶ Active replication
 - ▶ Passive replication



- ▶ LwFT4DRTS RFP does not require responses to address application state management
 - ▶ Applications can use any mechanism, or none
 - ▶ No guarantee that state is tightly synchronized between replicas and with invocation semantics
- ▶ DDS provides functionality, reliability and QoS to manage CORBA servers' replicated state
 - ▶ Services in DDS-based systems are often replicated
 - ▶ DDS topic keys and query APIs allow state instances to be retrieved as needed, or subscribed to
 - ▶ State can be redundantly persistent (on disk), available at system startup and to late-joining applications
 - ▶ Support for active and passive replication of services



- ▶ Durability
 - ▶ PERSISTENT – stored on disk across shutdown
 - ▶ TRANSIENT – stored in memory external to writer
- ▶ Presentation
 - ▶ Controls scope for ordering and coherency
- ▶ Ownership and Ownership Strength
 - ▶ SHARED – useful with passive replication, where only one replica should be updating the state at a time
 - ▶ EXCLUSIVE – useful with active replication, so that only updates from publisher with highest strength are seen
- ▶ Reliability
 - ▶ RELIABLE – If consistency is needed among replicas
- ▶ Writer Data Lifecycle
 - ▶ `autodispose_unregistered_instances` – set to FALSE



- ▶ Traditional node manager hierarchy
 - ▶ Use DDS to replicate state between node managers
 - ▶ Use DDS for communication between node managers and applications
- ▶ Peer-to-peer approaches?
 - ▶ Need consistent decision that a primary replica has failed and on which replica replaces it
 - ▶ Select primary replica based on Ownership Strength
- ▶ Fault detection
 - ▶ Liveliness policy provides various options
 - ▶ Can be used hierarchically or peer-to-peer
- ▶ Application Management and System Monitoring (AMSM) for CMS Specification (Beta 2 dtc/08-03-08)
 - ▶ Optional Fault Tolerance Management Profile
 - ▶ DDS PSM
 - ▶ Could be extended to handle fault detection and recovery



- ▶ Goal - Leverage DDS's reliable one-to-many messaging to actively replicate CORBA servers
- ▶ Prototyped DDS transport as TAO pluggable protocol
 - ▶ Initially focused on non-replicated messaging
 - ▶ First built connection-oriented stream implementing ACE_IPC_SAP abstraction
 - ▶ ACE_DDS_Addr, ACE_DDS_Acceptor, ACE_DDS_Connector, ACE_DDS_Stream classes
 - ▶ Use DDS Event Pattern to distribute messages
 - ▶ Keys determine destinations
 - ▶ Then built TAO pluggable protocol
 - ▶ TAO_DDSIOP_Profile, TAO_DDSIOP_Acceptor, TAO_DDSIOP_Connector, and TAO_DDSIOP_Transport classes
 - ▶ Intended to implement active replication using Ownership Strength to select from multiple replies received in client



```
module DDSIPC
{
  typedef string DestinationId;
  typedef unsigned long ConnectionId;
  typedef sequence<octet> Buffer;

  struct Connect
  {
    DestinationId server;
    DestinationId client;
    ConnectionId connection;
  };
  #pragma keylist Connect server

  struct Accept
  {
    DestinationId client;
    ConnectionId connection;
  };
  #pragma keylist Accept client connection

  struct Transfer
  {
    DestinationId client;
    ConnectionId connection;
    Buffer data;
  };
  #pragma keylist Transfer client connection
};
```

▶ Connect Topic

- ▶ Message sent from client to server group
- ▶ Single topic with server's DestinationId as key
- ▶ ACE_DDS_Connector publishes
- ▶ ACE_DDS_Acceptor subscribes with ContentFilteredTopic

▶ Accept Topic

- ▶ Message sent from each server group member to client
- ▶ Single topic with client's DestinationId and ConnectionId as keys
- ▶ ACE_DDS_Acceptor publishes
- ▶ ACE_DDS_Connector subscribes with ContentFilteredTopic

▶ Transfer Topic

- ▶ Transfer connection data between client and server
- ▶ Separate topic for each direction
- ▶ ACE_DDS_Stream publishes
- ▶ Peer ACE_DDS_Stream subscribes with ContentFilteredTopic



- ▶ IOR Profile contents
 - ▶ Domain ID – Identifies DDS domain for client to communicate with server
 - ▶ Topic Name Base – Append “_Connect”, “_Accept”, “_Request”, and “_Reply” to form topic names
 - ▶ Destination ID – Identifies server endpoint within domain and topic namespace
 - ▶ Object Key
- ▶ Future extensions
 - ▶ DDS QoS policies



- ▶ DDS Transport works fine for non-replicated CORBA server
 - ▶ Possible advantages over IIOp for real-time systems
 - ▶ Prototype's IPC_SAP integration with ACE Reactor via Unix pipe likely bottleneck
- ▶ DDS Topic definitions are relatively static
 - ▶ Initially wanted separate topic for each connection or destination
 - ▶ Topic cleanup not defined, so used keys and content filtering
 - ▶ Partition QoS can be used to control network data flow
- ▶ Stream abstraction is not a good fit for active replication
 - ▶ Reply streams from replicated server are not identical unless system has strictly deterministic behavior
 - ▶ Should send entire GIOP Request message, or maybe message fragment, as DDS sample
 - ▶ Should decouple reply from request
 - ▶ Need request IDs scoped globally rather than by connection



- ▶ Build on GIOP Location Forwarding
 - ▶ Similar to CORBA Locator / Implementation Repository
 - ▶ Group IORs point at Forwarding Agent
 - ▶ Forwarding Agent responds to GIOP Request or LocateRequest message with IOR of current active group member
 - ▶ Client uses active group member until it fails
- ▶ DDS Applicability
 - ▶ For FT, need highly available Forwarding Agent
 - ▶ For RT, need to bound recovery time
 - ▶ DDS can provide high availability and bounded recovery time



- ▶ **Several Possible Approaches**
 - ▶ Redundant traditional Forwarding Agents
 - ▶ Possibly as process on each client node with “localhost” IIOIP address
 - ▶ Replicate group management state via DDS
 - ▶ See previous slides on Application State Management
 - ▶ Fully decentralized Forwarding Agent and Group Management
 - ▶ Co-located in-process with all CORBA clients and servers
 - ▶ All group management state distributed via DDS
- ▶ **Prototyped fully decentralized approach**
 - ▶ Used TAO’s Pluggable Protocols framework and Portable Interceptors to implement Location Forward Fault Tolerance (LFFT) framework
 - ▶ Used OpenSplice to implement DDS Location Forwarding (DDSLF) provider within LFFT framework



- ▶ LWFT4DRTS RFP asks for “FT-enabled middleware” interfaces rather than specific solutions
 - ▶ LFFT Framework serves as intermediary between ORB and abstract LFFT provider
 - ▶ LFFT provider implementation based on DDS or any other mechanism
- ▶ LFFT Assumptions
 - ▶ Location Forward based fault tolerance
 - ▶ Forwarding Agent co-located in client application process
 - ▶ Registration of member IORs co-located in server application process
- ▶ LFFT Framework interfaces could be implemented in any ORB using standard and/or proprietary ORB hooks
 - ▶ Portable Interceptors
 - ▶ Object Reference Template
 - ▶ Extensible Transport Framework (possible RFC)
 - ▶ Implementation Repository hooks



CORBA Passive Replication - LFFT IOR Profile and Component

15

- ▶ Object Group references contain single LFFT IOR Profile
 - ▶ Object Key identifies Object Group
 - ▶ process_group_id – identifies server process group
 - ▶ adapter_name – POA name hierarchy
 - ▶ object_id – object identity within POA
 - ▶ LFFT Profile interpreted by client-side-only LFFT transport
 - ▶ No addressing information used in simple prototype
 - ▶ Might let LFFT Provider supply information such as DDS domain ID
- ▶ Object Group Member reference contain normal IOR Profiles such as IIOP
 - ▶ LFFT adds Component to all member reference Profiles
 - ▶ process_group_id – identifies target's process group
 - ▶ process_id – identifies specific target's process
 - ▶ LFFT component interpreted by ClientRequestInterceptor
 - ▶ Check that target's process is the current active member of its process_group
 - ▶ If not, forwards client back to group reference



Copyright PrismTech 2008



CORBA Passive Replication - LFFT Provider Interface

16

```
class LFFT_Provider
{
public:
    virtual void
    register_member (CORBA::ORB_ptr orb,
                    const char* adapter_name,
                    const CORBA::OctetSeq& object_id,
                    CORBA::Object_ptr member) = 0;

    virtual CORBA::Object_ptr
    get_primary_member (CORBA::ORB_ptr orb,
                       const char* process_group_id,
                       const char* adapter_name,
                       const CORBA::OctetSeq& object_id) = 0;

    virtual bool
    is_primary () = 0;

    virtual bool
    is_primary (const char* process_group_id,
               const char* process_id) = 0;
};
```

- ▶ **register_member** operation
 - ▶ Object Adapter in server registers IOR of Object Group member
 - ▶ Called from LFFT_ObjectReferenceFactory's make_object operation in prototype
- ▶ **get_primary_member** operation
 - ▶ Forwarding Agent in client retrieves IOR of current primary Object group member
 - ▶ Called from LFFT_Transport's send_request operation in prototype
- ▶ **is_primary** operations
 - ▶ Determines whether process is current primary member of group
 - ▶ In prototype client, called from LFFT_ClientRequestInterceptor's send_request operation with process_group_id and process_id obtained from current target's IOR Component
 - ▶ In prototype server, called from LFFT_ServerRequestInterceptor's receive_request_service_context operation to check current process role

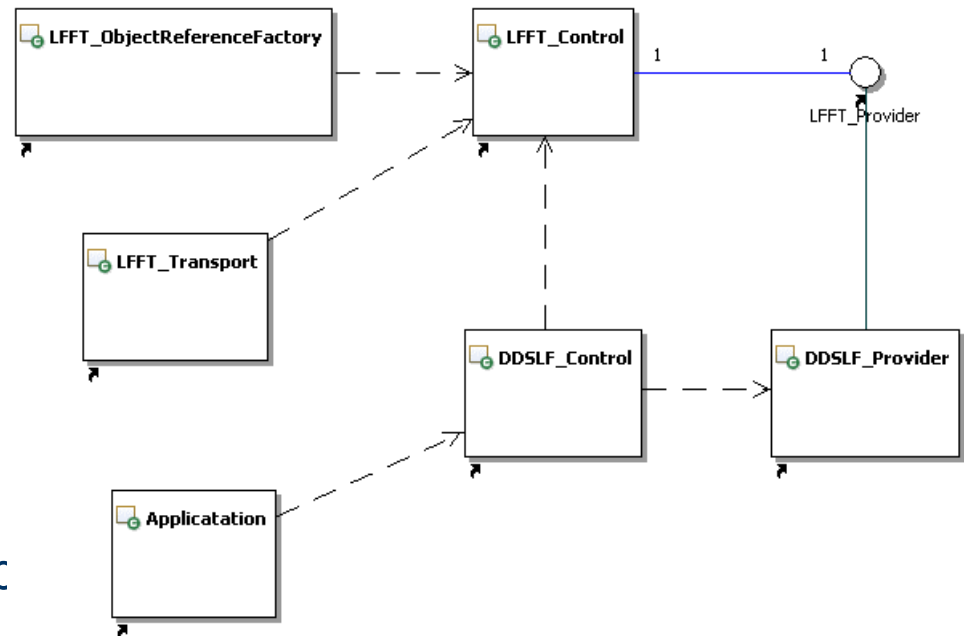


▶ DDSLF_Provider class

- ▶ Implements abstract LFFT_Provider interface
- ▶ Reads and writes DDS topics
 - ▶ DDS State Pattern

▶ DDSLF_Control class

- ▶ API used by application logic and/or process group management
- ▶ Delegates to DDSLF_Provider and LFFT_Control



```
class DDSLF_Control
{
public:
    explicit
    DDSLF_Control (const char* domain_id = "");

    ~DDSLF_Control ();

    void
    init_client ();

    void
    init_server (const char* process_group_id,
                const char* process_id = 0);

    void
    become_primary ();
};
```

- ▶ Constructor
 - ▶ Specifies DDS domain
- ▶ `init_client` operation
 - ▶ Initializes without being member of server Process Group
 - ▶ Supports client role only
- ▶ `init_server` operation
 - ▶ Initializes specifying identity within Process Group
 - ▶ If not specified, used host and PID for `process_id`
 - ▶ Supports client and server roles
- ▶ `become_primary` operation
 - ▶ Transitions from backup to primary replication role
 - ▶ Application responsible for integration with Process Group Management



```
module DDSLF
{
  struct GroupMember
  {
    string process_group_id;
    string adapter_name;
    string object_id;
    string member_process_id;
    string member_ior;
  };
  #pragma keylist GroupMember process_group_id \
    adapter_name object_id member_process_id

  struct ProcessGroup
  {
    string process_group_id;
    string primary_process_id;
  };
  #pragma keylist ProcessGroup process_group_id
};
```

- ▶ **GroupMember Topic**
 - ▶ Each instance represents state of one replica of one Object Group
 - ▶ Published by CORBA server
 - ▶ Queried by Forwarding Agent in CORBA client
- ▶ **ProcessGroup Topic**
 - ▶ Each instance represents state of one Process Group
 - ▶ Published by CORBA server when promoted to primary replica
 - ▶ Queried by CORBA client to verify that target is current primary replica
 - ▶ Queried by CORBA server to verify that it is current primary replica



- ▶ Process and Object Group Management state can easily be represented as DDS topics
- ▶ Querying state from DDS when needed is extremely simple
 - ▶ Need to analyze performance
 - ▶ Cache ProcessGroup state if needed
- ▶ Co-locating Forwarding Agent and Group Management functionality with applications results in highly dependable and scalable system
- ▶ Interfaces between FT-enabled middleware, specific FT solution providers, and Process Group Management facilities need to be carefully architected



▶ Summary

- ▶ DDS is a natural fit anywhere highly available real-time data distribution is needed
- ▶ DDS in practice co-exists peacefully with CORBA at application and infrastructure levels
- ▶ Real-time systems that need both data distribution and distributed object invocations should consider leveraging DDS to support CORBA services
- ▶ Using DDS to manage and distribute application state is well understood
- ▶ DDS can support Process Group Management systems in various ways
- ▶ Using DDS as a GIOP message transport for active replication looks promising but needs further investigation
- ▶ LwFT4DRTS RFP's concept of composing systems from FT-Enabled middleware and external FT infrastructure seems viable
- ▶ Joint LwFT4DRTS revised submission should incorporate experience gained from this effort

▶ Contacts

- ▶ Bob Kukura - robert.kukura@prismtechusa.com
- ▶ Jai Balasubramanian - jai@dre.vanderbilt.edu
- ▶ Nanbor Wang - nanbor@txcorp.com

