Raytheon

NAVSEA WARFARE CENTERS DAHLGREN

Analysis of Passive CORBA Fault Tolerance Options for Real-Time Applications

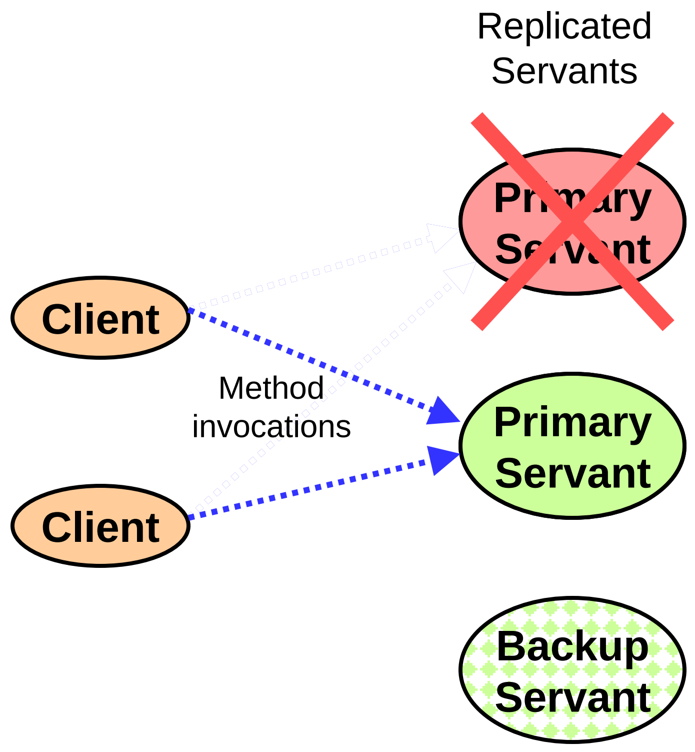
Robert A. Kukura, Raytheon IDS
Paul V. Werme, NSWCDD

Raytheon

PASSIVE CORBA FAULT TOLERANCE

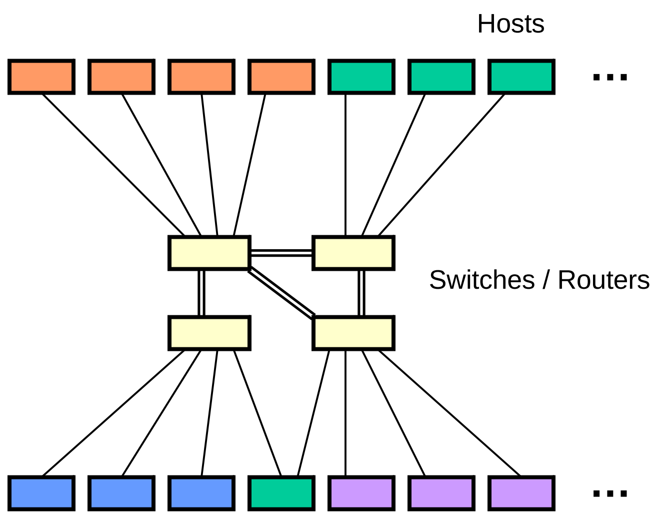
NAVSEA WARFARE CENTERS DAHLGREN

- All clients send method invocations only to the primary
- All clients have same view of which servant is primary
- On failure of the primary, a backup servant is promoted to primary
- All clients begin sending method invocations to new primary



COMPUTING ENVIRONMENT CONTEXT

- Distributed systems
  - TCP/IP-based network infrastructure
  - Potentially multiple hops via switches / routers
- Heterogeneous hardware and operating systems
- Mix of applications developed in different languages
- Application use of multiple middleware technologies
  - often within the same process
- Mix of hard, soft, and non-real-time processing
- Application use of multiple ORBs
  - occasionally within the same process.
- Existing fault management (fault detection, fault isolation, fault recovery) infrastructure capabilities
  - Primarily process-level failover



- Complex Non-CORBA-centric Distributed Computing Environment

GOALS AND CONSTRAINTS

- Provide a passive CORBA FT approach
  - Focus on object group connection management and fail-over
  - Leverage external application and object life-cycle control and state mgmt
  - Leverage external application and system-level fault detection and recovery
- Work with existing CORBA ORBs
  - Use of standard ORB extension mechanisms
- Provide Client-side object interfaces via a single Client-side stub for each replicated object group
- Maximize transparency of the FT approach at the Application Level (making it look like "normal" CORBA)
  - During initial resolution of the primary server object address
  - During failover to a new primary server object
  - During ORB/POA instantiation and initialization
  - During Server Object creation and Client Stub instantiation
- Use IORs for addressing
- Use standards-based mechanisms for:
  - routing of client requests to the current primary server object
  - re-routing of client requests to a new primary server object in the event of primary server failures detected during method invocation

GOALS AND CONSTRAINTS (cont.)

- Allow the use of out-of-band non-CORBA mechanisms for internal dissemination of object group addressing, status, and role information
- Avoid single points of failure
- Minimize complexity of the CORBA FT implementation
- Minimize run-time latency for normal communications and FT cases
- Use of “Directory Service” usage patterns for storing / retrieval of IORs
- Additional Assessment Criteria:
  - Standards compliance
  - Support for Integration with non-FT aware CORBA clients
  - Scalability of the solution
  - Extensibility for supporting additional CORBA or Application FT models

- The goals and constraints serve as the basis for the analysis of the option space

Raytheon

WHY NOT FT CORBA?

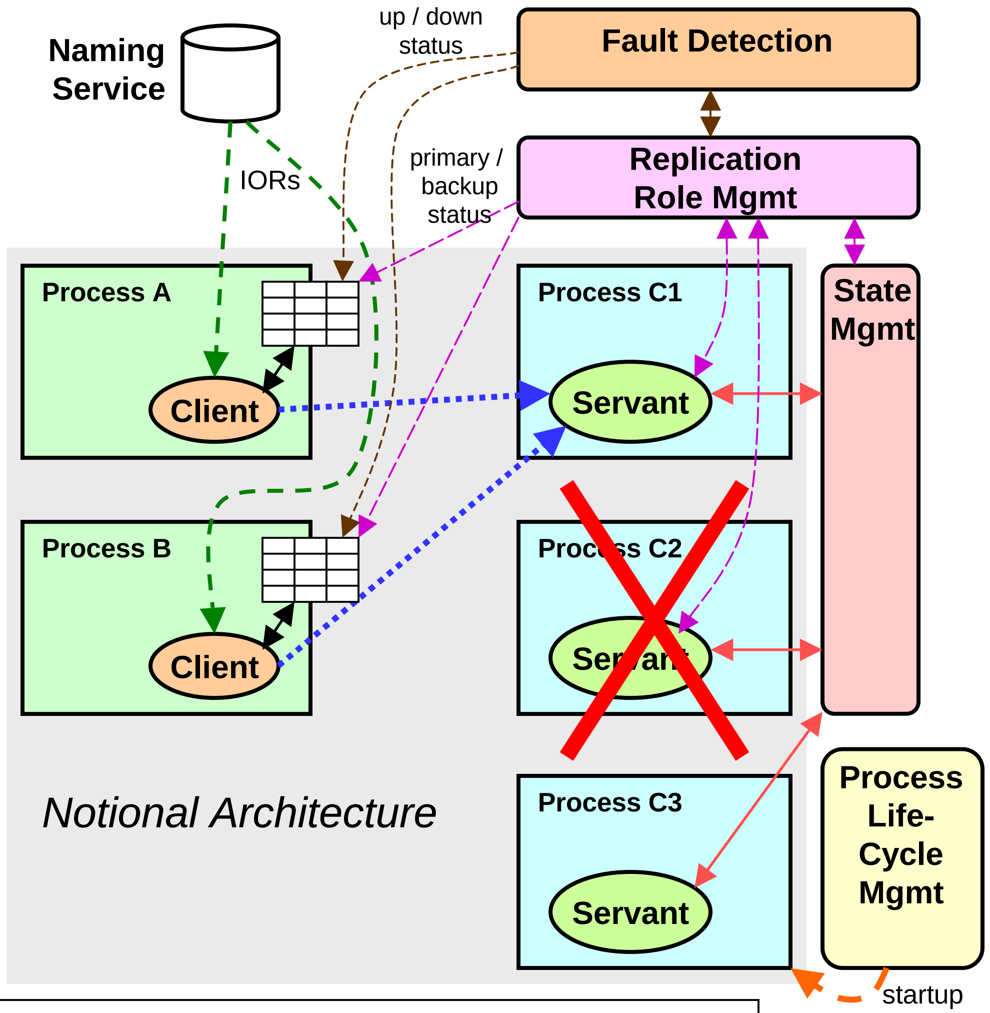
NAVSEA WARFARE CENTERS DAHLGREN

- **Fault Tolerant CORBA published as a chapter of CORBA 2.5 in Sept, 2001**
  - **has not been widely implemented or applied**
- **Issues:** *
  - **Overly Complex**
    - applications can often obtain sufficient reliability through simpler and more transparent standard interoperable mechanisms
    - Spec is too feature rich for typical application developers and ORB vendors
    - Potential vendors have been deterred by the difficulty of implementing the complete set of features needed to claim conformance to the specification
  - **Unsuitable for Real-Time Systems**
    - FT CORBA's interoperable passive replication protocol extensions do not adequately bound recovery time for many RT systems
    - No interoperable protocol for active replication
    - The mechanisms and constraints required to achieve strong replica consistency are difficult to achieve in real-time and are often not needed
  - **Too Fine Grained**
    - Fault detection and Replication management is at the object level rather than at the process level or the POA and ORB-provided endpoint level.
  - **Difficult System Integration**
    - Imposes a CORBA-centric view and requires a CORBA-centric support infrastructure

* Problem statement (Section 6.1) from the OMG LW FT for Distributed RT Systems RFP (realtime/2006-06-06)

EXTERNAL ARCHITECTURAL COMPONENTS

- Directory Service
- Replica Group Membership
  - Mapping of Group References to their associated Direct Object References
- Application State Management
- Server Replica Role Management
  - determination of primary and backup statuses
  - handshaking required to promote backup to primary
- Network, OS, Process, and System-Level Fault Detection
- Process Life-Cycle Management
  - startup, shutdown, and restart of replicated servers



- **CORBA FT capabilities should be viewed as an integrated part of a broader application and system fault management context**

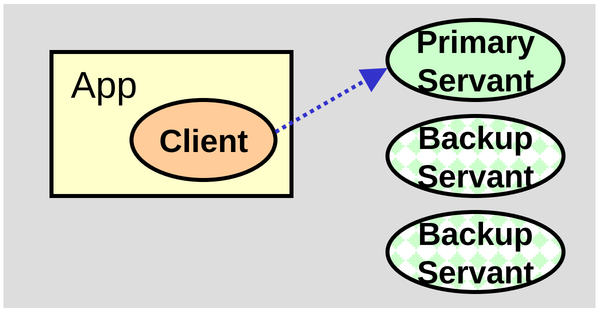
TAXONOMY OF CORBA FT APPROACHES

- Mechanisms constructed on top of ORB client binding
  - CORBA-based frameworks (e.g., smart proxies)
  - Application-specific constructs
- Modifications to ORB/GIOP behavior
  - IOGRs (FT CORBA)
  - Vendor-specific extensions
- Transport protocols under GIOP
  - Extensible Transport Framework (ETF) or proprietary APIs
  - Group communication protocols
- Location forwarding
  - External forwarding agent
  - Forwarding within client or server Request Interceptors
  - ORB internal (e.g., transport-level) forwarding

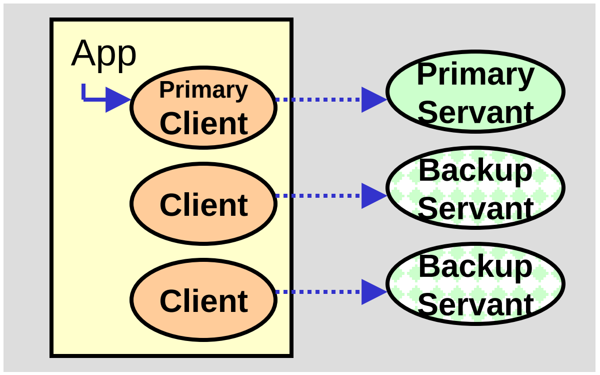
MECHANISMS ON TOP OF ORB CLIENT BINDING

Application Management of Member References:

1. **Instantiate a client stub only for the primary reference**
   - **On failure:**
     - retrieve the new primary reference from the Directory Service and/or use OOB mechanisms or policy to determine the new primary reference
     - instantiate a new client stub with the new reference



2. **Instantiate a client stub for each member reference**
   - **Application-level coordination of which client stub represents the primary**
   - **Polling of Directory Services and/or use of OOB mechanisms to:**
     - find and instantiate stubs for new member references
     - Determine change of primary



Issues:

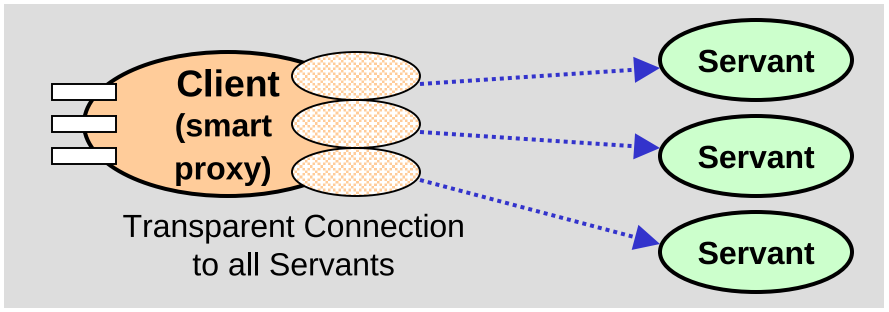
- **Difficult to achieve RT performance due to dependence on TCP and CORBA timeouts and non-RT Directory access**
  - Use of non-CORBA mechanisms for fault detection and low-latency propagation of new primary reference information can improve failover times
- **No transparency**
  - All coordination and management handled at the application level

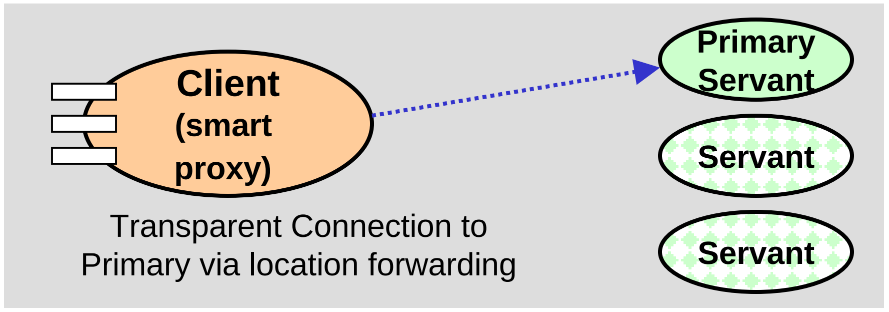
MECHANISMS ON TOP OF ORB CLIENT BINDING

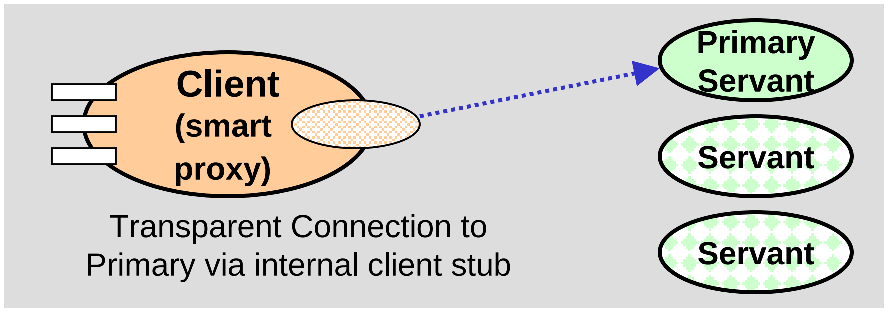
Smart Proxies:

- Allows abstracting the underlying mechanisms and constructs behind a single client-side stub
  - Can be used to implement passive, semi-active, and active fault tolerance approaches
  - Logic to retrieve and manage member references, connections, and failover can be embedded within proxy
- Potential for additional marshaling / unmarshaling penalties depending on proxy implementation
- Non-IOR semantics for accessing local proxy interfaces
- No standards
- Significant variation in vendor approaches and capabilities

Examples:







MODS TO ORB/GIOP BEHAVIOR & TRANSPORT PROTOCOLS

- **Modifications to ORB / GIOP behavior**
  - **Example: IOGRs (Interoperable Object Group References)**
    - **Standards-based**
      - Most ORBs have not implemented all IOGR capabilities
  - **Vendor-specific extensions**
    - **Could limit interoperability with other ORBs**
    - **Could limit portability to other ORBs**
- **Transport protocols under GIOP**
  - **Multicast Transport Protocol**
    - **no standard for Reliable Multicast transport**
    - **development of server-side coordination infrastructure is required**
  - **Unicast Transport Protocol**
    - **Sequential or Concurrent**
  - **Extensible Transport Framework (ETF) or vendor-proprietary pluggable transport APIs**
    - **Group Communication approaches have potential**
    - **Transports to provide a priori reliable connection endpoints**
  - **FT mechanisms at the Transport Level are possible**
    - **Issue of limited context at transport level for determining proper action(s) or group membership information**

Promising options for passive, active, and semi-active capabilities

LOCATION FORWARDING

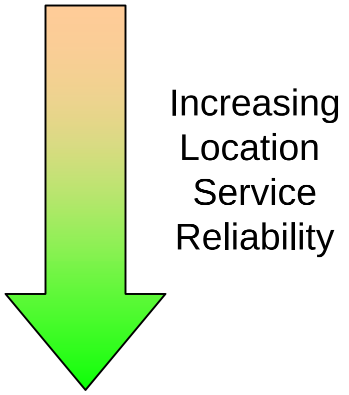
- Location Forwarding is a fundamental mechanism provided by the CORBA interoperability architecture
  - The target of a Request or LocateRequest message can respond with a Reply or LocateReply message containing a new object reference that the client should contact for the current and subsequent invocations.
  - APIs for generating location forwards are provided in the Portable Object Adapter and in Portable Interceptors

- Mechanisms / Tools Supporting Location Forwarding Approaches:

– Location Forwarding:	• GIOP Location Forwarding • Portable Interceptors ForwardRequest Exception
– IOR Construction:	• IOR Interceptors • Object Reference Template (ORT)
– Object Reference Validation and Control Path Hooks:	• ClientRequestInterceptor • ServerRequestInterceptor • Extensible Transport Framework (ETF)
– Reliable Addressing:	• Extensible Transport Framework (ETF)

LOCATION FORWARDING

- General Approach
  - Location forwarding is used to direct the client ORB to the current primary group member
  - ClientRequestInterceptors can be used to prevent invocations on failed group members
  - On failure of a forwarded connection, the client ORB reverts back to the original object reference
    - Must be reliable!
  - IORInterceptors and Object Reference Templates (ORT) are used to transparently construct object group references.
- Approaches:
  - Traditional location service
  - Location service accessed via Request Interceptors
  - Replicated Location service
  - Local Location service instances
  - Location service within application process
  - Location service within client ORB transport



Raytheon

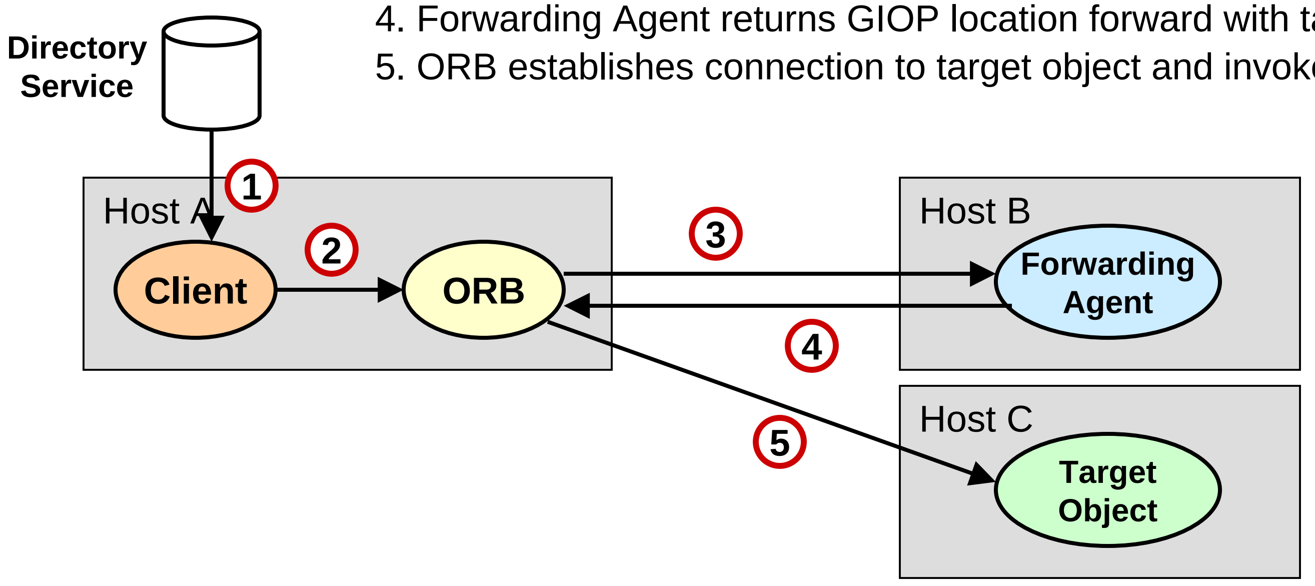
TRADITIONAL LOCATION SERVICE

NAVSEA WARFARE CENTERS DAHLGREN

- **Contact a proxy service which will redirect the client invocation to the desired target service**

Initial method invocation:

1. Discover IOR of proxy (Forwarding Agent)
2. Client invokes request using the proxy object reference
3. ORB establishes connection to proxy object and invokes the request
4. Forwarding Agent returns GIOP location forward with target object reference
5. ORB establishes connection to target object and invokes request

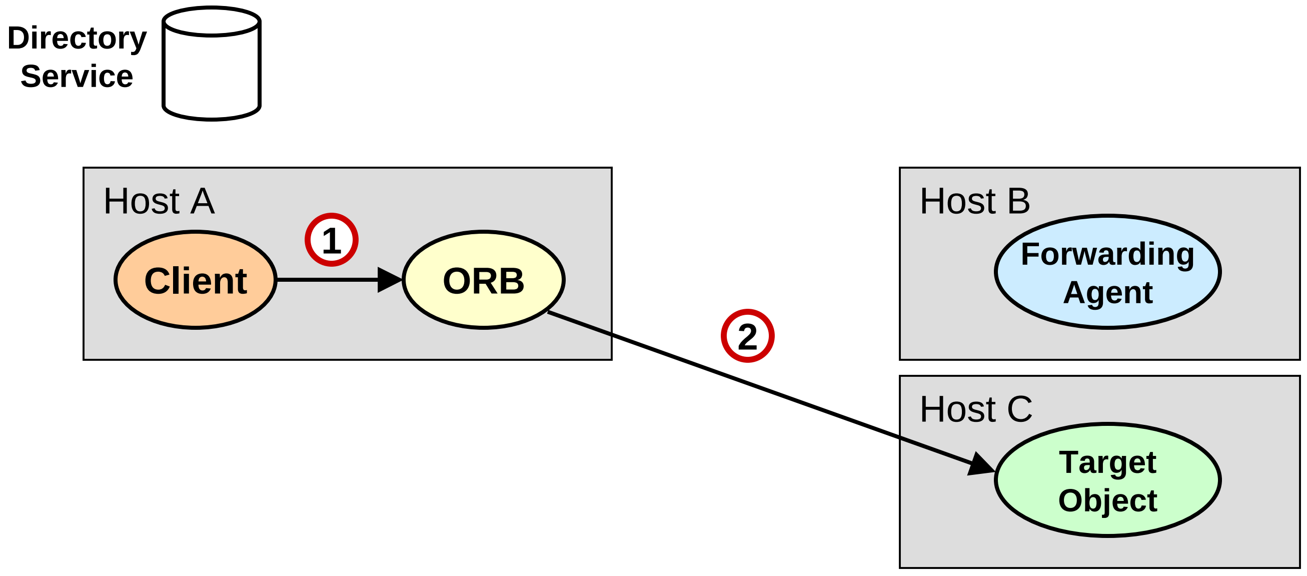


TRADITIONAL LOCATION SERVICE

- **Contact a proxy service which will redirect the client invocation to the desired target service**

Subsequent method invocations:

1. Client invokes request
2. ORB invokes request on target object



Raytheon

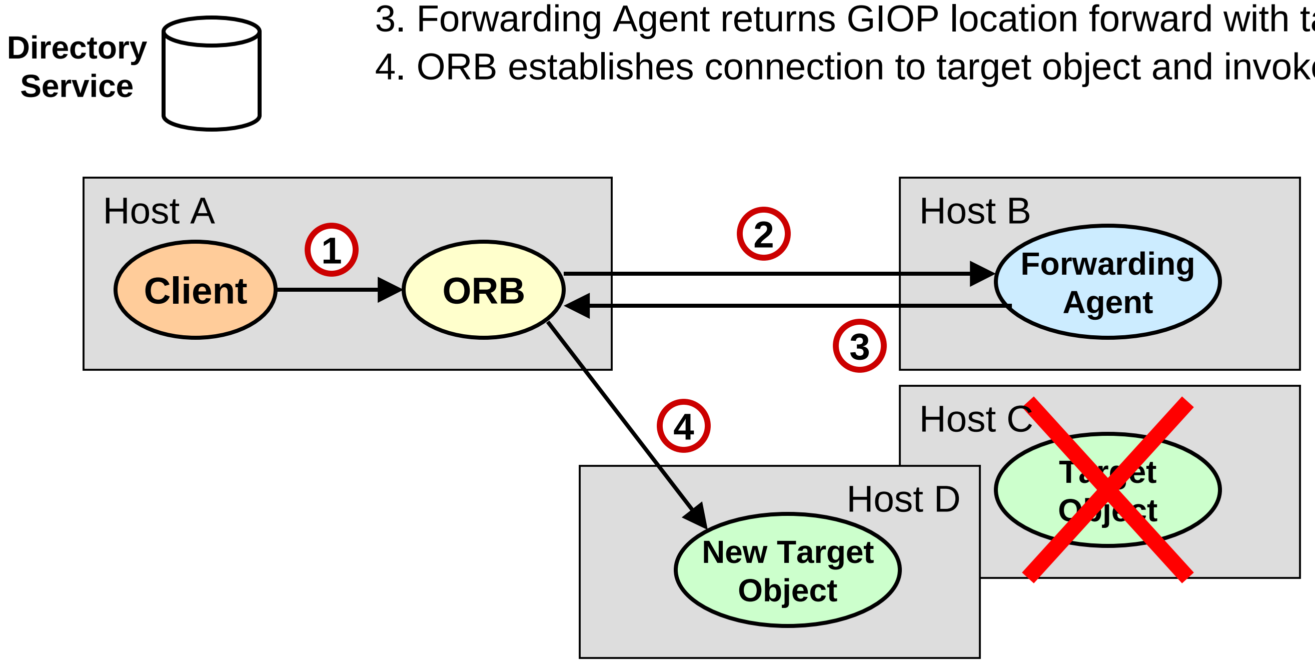
TRADITIONAL LOCATION SERVICE

NAVSEA WARFARE CENTERS DAHLGREN

- **Contact a proxy service which will redirect the client invocation to the desired target service**

Failure of Target Object:

1. Client invokes request
2. ORB connects to proxy object using the original object reference and invokes the request
3. Forwarding Agent returns GIOP location forward with target object reference
4. ORB establishes connection to target object and invokes request

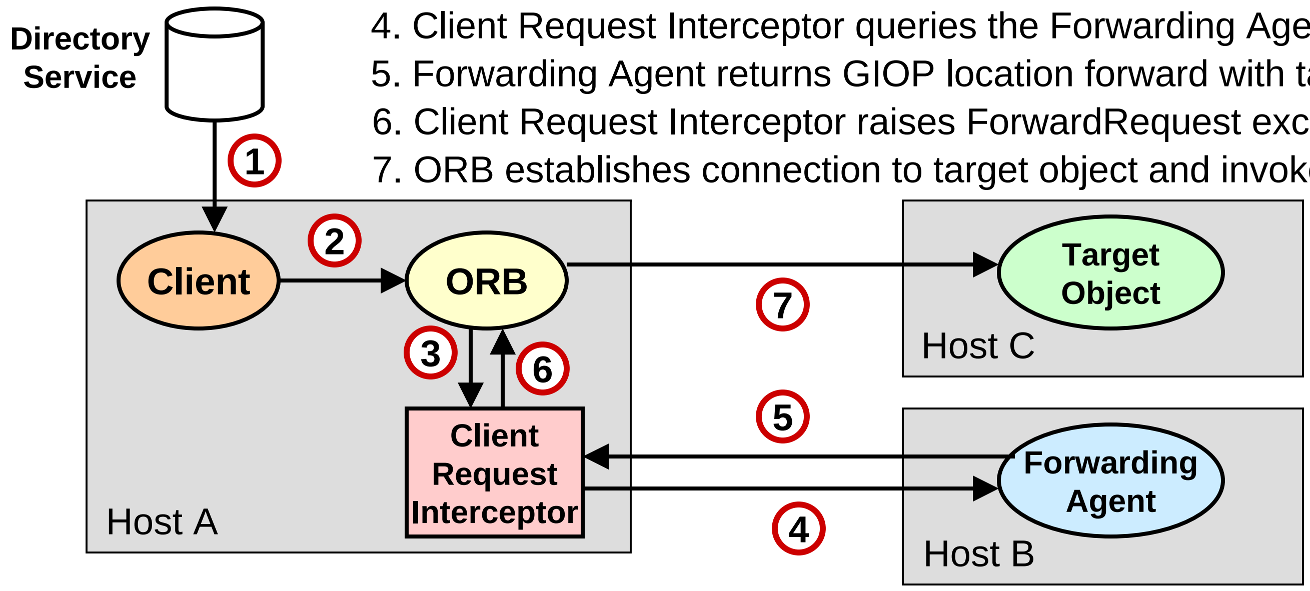


LOCATION SERVICE ACCESSED VIA REQUEST INTERCEPTORS

- Client Request Interceptors are used to:
  - Check if the current target is still valid
    - Assumption is that external infrastructure components will update status info
  - If valid, invoke the request on the current target (primary)
  - If not valid, invoke the request on the Location Service in order to be forwarded to the new primary

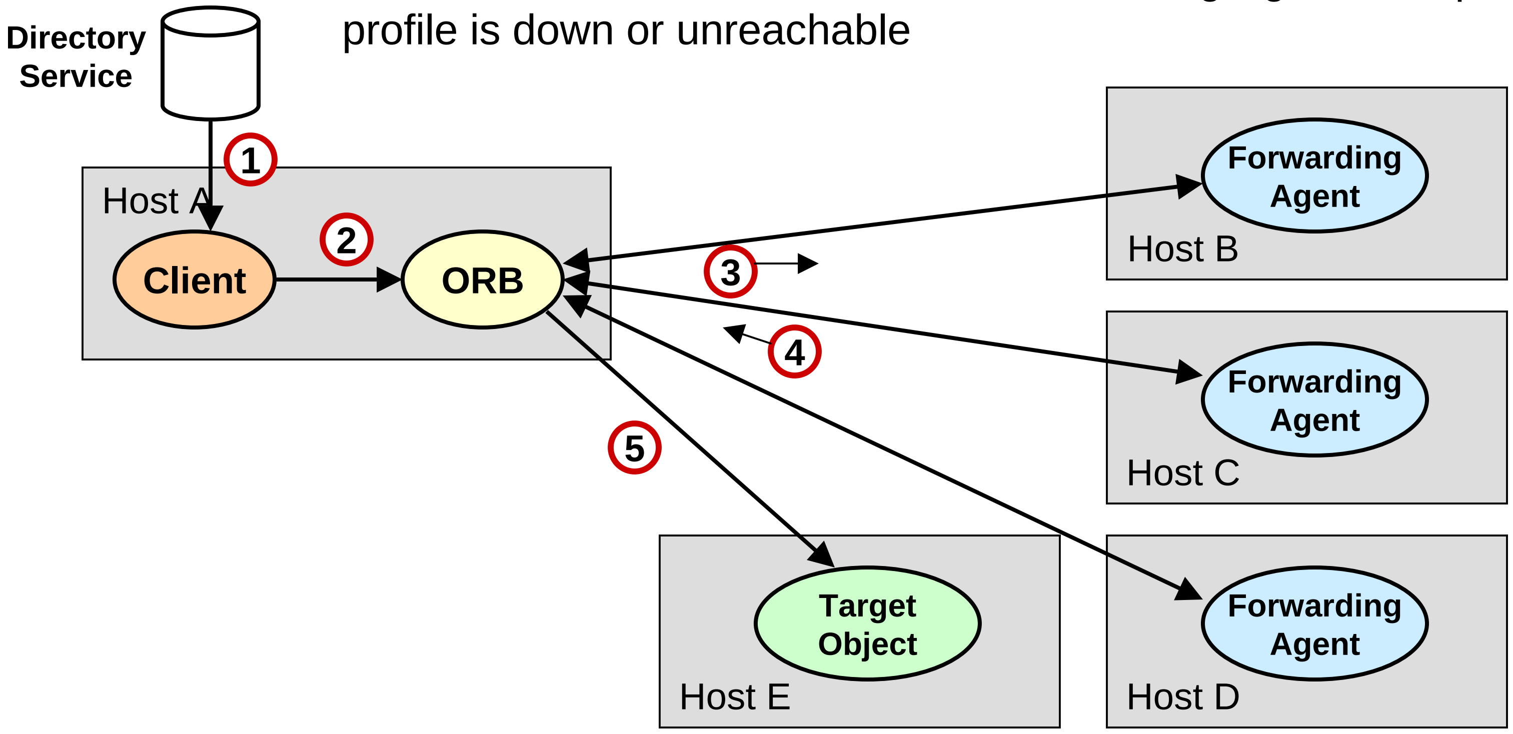
Initial method invocation:

1. Discover IOR of proxy (Forwarding Agent)
2. Client invokes request using the proxy object reference
3. Client Request Interceptor is called and there is no valid target
4. Client Request Interceptor queries the Forwarding Agent
5. Forwarding Agent returns GIOP location forward with target object reference
6. Client Request Interceptor raises ForwardRequest exception to the ORB
7. ORB establishes connection to target object and invokes request



REPLICATED LOCATION SERVICE

- **Replication of the Location Service can provide increased reliability**
  - **The approaches can use either the mechanisms shown for the Traditional Location Service or the Location Service via the Request Interceptors**
  - **Options:**
    - **Multicast IOR to address a set of Forwarding Agents**
    - **IOR with multiple IIOP profiles for each of the set of Forwarding Agents**
      - Determinism can be an issue if the Forwarding Agent for a particular profile is down or unreachable

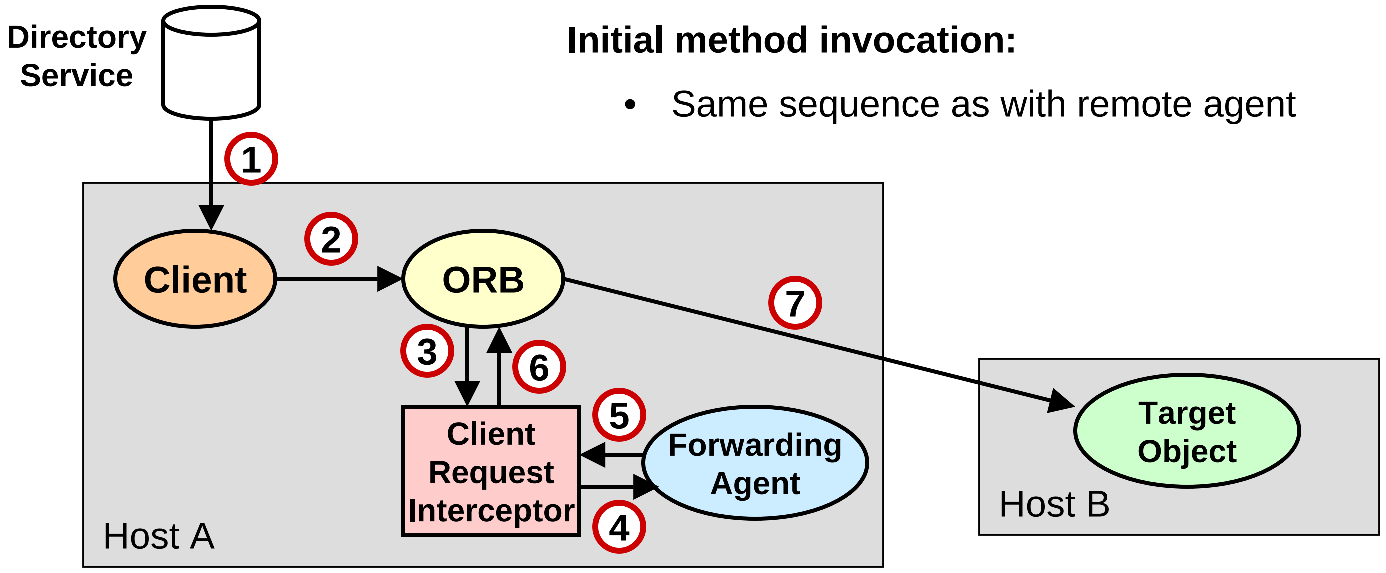


Raytheon

LOCAL LOCATION SERVICE INSTANCES

NAVSEA WARFARE CENTERS DAHLGREN

- **Co-location of the Location Service on the same host as the client**
  - **Increases reliability**
    - **Fails when Location Service on local box is down**
  - **Location Service Proxy IOR:**
    - **Can be generated within the client to contain the local IP address**
      - The IOR will not be usable by components external to the host
    - **Can use the IP Loopback address**
      - The IIOP IOR will be common across hosts
      - Will only be valid on hosts where a Location Service resides



Initial method invocation:

- Same sequence as with remote agent

Raytheon

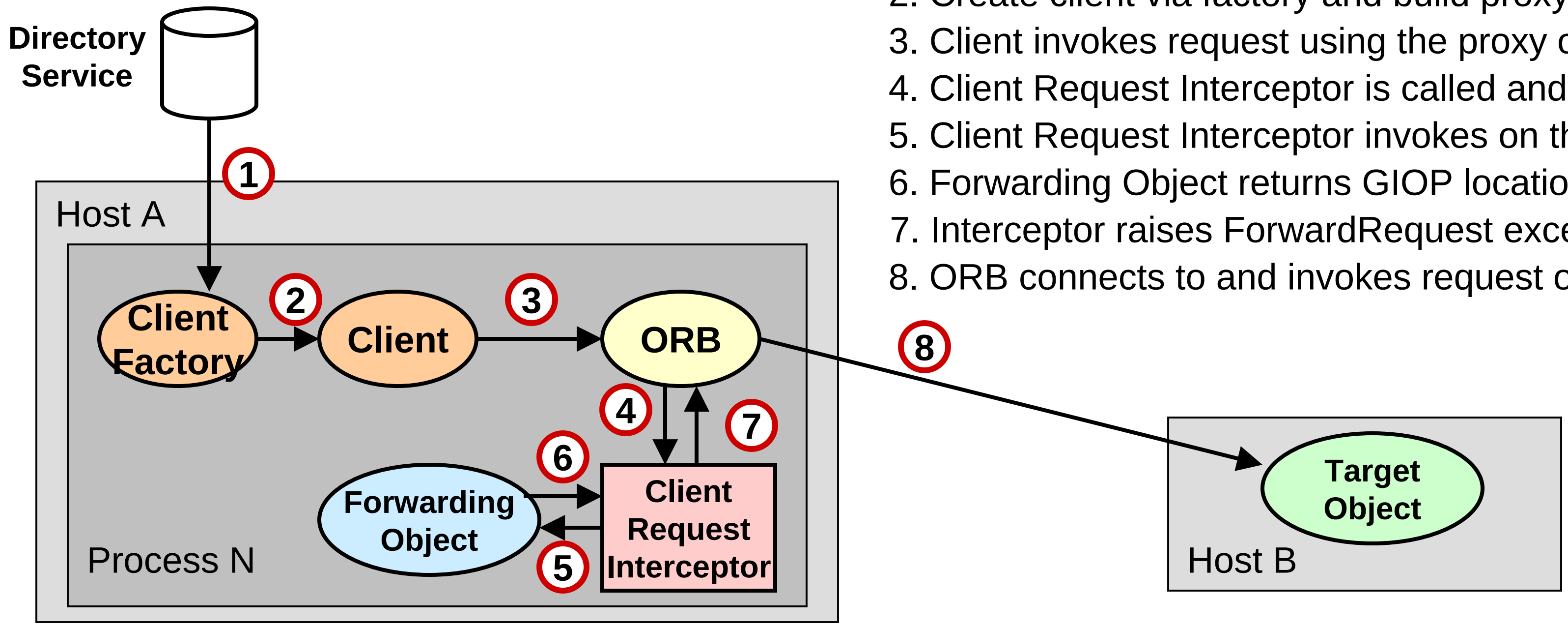
LOCATION SERVICE WITHIN APPLICATION PROCESS

NAVSEA WARFARE CENTERS DAHLGREN

- **Location Service resides within process address space**
  - **Accessed via IP address within process**
  - **Object reference containing local IP address is created on the client-side**
    - **Object reference will not be valid external to the process**
    - **Object reference is reliable while the process is running and there are no socket problems**

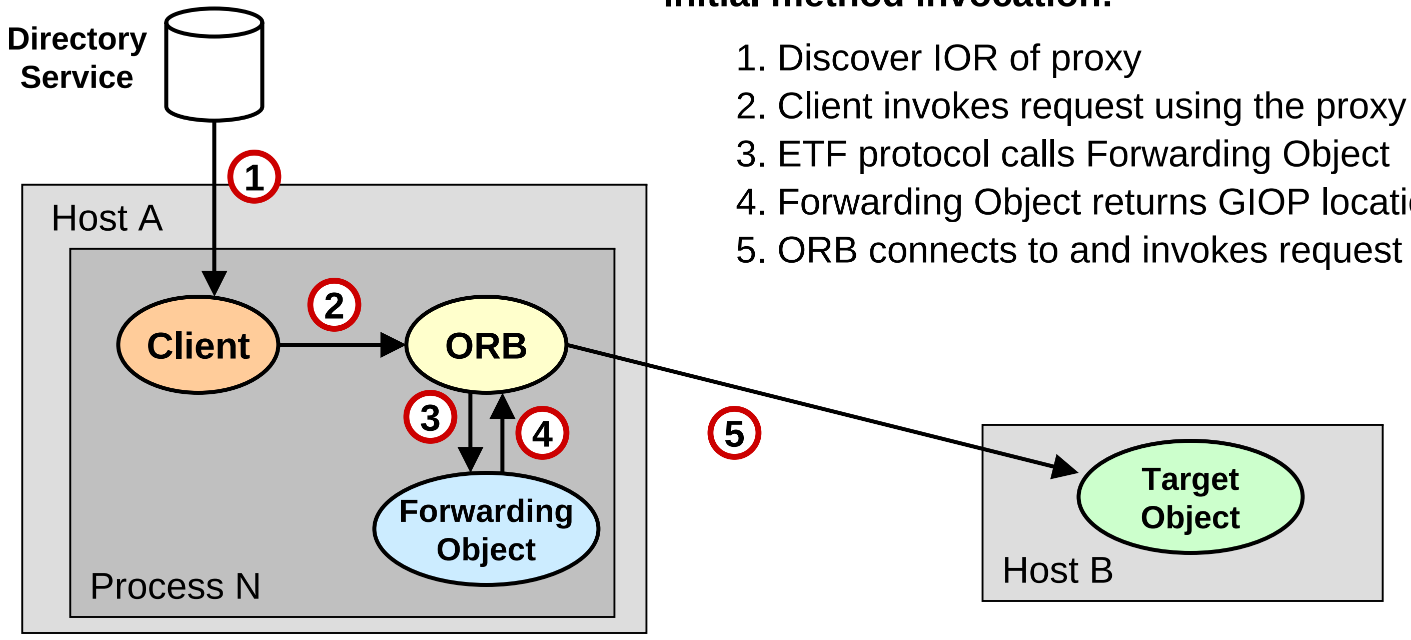
Initial method invocation:

1. Discover IOR of proxy (Forwarding Object)
2. Create client via factory and build proxy IOR
3. Client invokes request using the proxy object reference
4. Client Request Interceptor is called and target not valid
5. Client Request Interceptor invokes on the proxy
6. Forwarding Object returns GIOP location forward
7. Interceptor raises ForwardRequest exception
8. ORB connects to and invokes request on target object



LOCATION SERVICE WITHIN CLIENT ORB TRANSPORT

- Location Service resides within ORB Transport
  - ORB Transport uses dummy ETF protocol that is always connected
  - Object reference contains ETF "connected" profile
    - Can include any string as the object_id (i.e., can be group id)
    - Object reference will be valid external to the process for any ORBs that support the ETF "connected" protocol
    - Object reference is reliable while the process is running



Initial method invocation:

1. Discover IOR of proxy
2. Client invokes request using the proxy object reference
3. ETF protocol calls Forwarding Object
4. Forwarding Object returns GIOP location forward
5. ORB connects to and invokes request on target object

PASSIVE CORBA FT TECHNOLOGY CONSTRAINTS

- Reliable object references
- Distribution of group membership and role changes
- Determination of server group identification information
- Where to create group references
- Relaxation of "At-Most-Once" semantics
- Sources of non-determinism

RELIABLE OBJECT REFERENCES

- **Reliable Object References**
  - **Never want the ORB to attempt to connect to something that does not exist**
    - **TCP/IP connect timeout is too long**
  - **Gaining initial control of the communication path (i.e., something that can be reliably connected to) so that interceptors can be invoked or location forwarding can be done**
    - **Specification does not state whether a connection to the target object is established prior to or after the request interceptor is invoked**
    - **Many ORBs do establish a connection before the interceptor is called**
  - **Request Interceptors are not called for GIOP LocateRequest messages which are used by some ORBs to determine/verify the direct address of a server object**

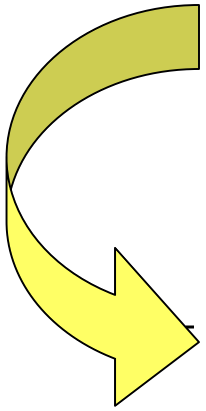
$\}$ The object reference needs to resolve to a known live address

Raytheon

RELIABLE OBJECT REFERENCES

NAVSEA WARFARE CENTERS DAHLGREN

- **Reliable Object References**
  - **Order of Preference / Reliability:**
    1. **Resolves always**
    2. **Resolves to connection within local address space**
    3. **Resolves to connection on local host**
    4. **Resolves to connection on external host(s)**
       a. resolves to connections on multiple hosts
       b. resolves to connection on one of a potential set of hosts
       c. resolves to connection on one host



  - **Options Explored:**
    1. **ETF "always connected" transport protocol**
    2. **IIOP address within the application (e.g., within a library)**
    3. **IIOP address of separate process using local host or loopback address**
    4. **Address of process on remote host**
       a. multicast transport address
       b. IOR with multiple IIOP profiles
       c. IIOP address

DISTRIBUTION OF MEMBERSHIP AND ROLE CHANGES

- **Distribution of Group Membership and Role Changes**
  - **What about IOGRs?**
    - **Interoperable Object Group References allow the specification of multiple IORs for a group and an indication of which is the primary**
    - **Management and distribution of group updates within IOGRs is difficult, especially under real-time constraints**
    - **More importantly, IOGRs do not provide mechanisms needed for effective real-time control of passive group connections**
      - specifying order of preference beyond just a hint as to which member is the primary
        - » **the hint is always wrong immediately after a failure!**
      - specifying which references are valid and invalid
      - hooks into the ORB to break and reestablish connections when the IOGR is updated
  - **Need alternative approach for real-time management and distribution of Group membership and role changes**
    - **out-of-band updates distributed to interested CORBA clients and servers**
    - **sufficient information to know status of and allow forwarding to primary**

DETERMINATION OF GROUP ID

- **Determination of server group identification information on the server-side and client-side (i.e., information needed in order to lookup information on the set of server instances)**
  - **limited context available within IOR interceptors**
    - **cannot add object-specific info (i.e., ObjectGroupID) to an IOR**
  - **creation of object keys is ORB-specific**
    - **An object_key created by an ORB encodes the POA’s Fully Qualified POA Name (FQPN) and object_id**
      - can only be interpreted by the same ORB implementation
  - **ETF "connected" protocol can allow the use of the ETF profile object_id field**

CONSTRUCTION OF GROUP REFERENCES

- **Where to create group references**
  1. **No explicit group references – use direct references as group references**
     - **Difficult to achieve since there are no simple hash algorithms for comparing object reference**
       – Multiple encodings can exist for the same object reference
  2. **server-side**
     - **Issue of how to publish both the group reference and the direct reference via standard CORBA mechanism**
     - **Current IIOP IOR formats require the same ORB in order to construct and decipher the object reference**
  3. **client-side**
     - **Removes transparency on the client side**
     - **Created references are local to the client process**
  4. **External**
     - **Object references created by third parties**
     - **Useful for IOR formats that are persistent and not tied to a specific ORB**
       – E.g., ETF "connected" object reference containing group_id string

RELAXATION OF "AT-MOST-ONCE" SEMANTICS

- Potential need to select “at-most-once”, “at-least-once”, or "exactly once" semantics where the completion result is MAYBE
  - when failure occurs after invocation but prior to response.

SOURCES OF NON-DETERMINISM

- **timeouts on connection attempts when remote address is unavailable**
- **timeouts on failed method invocations**
  - **no mechanism to “interrupt” a method invocation upon notification of failure**
- **server-side transition from backup to primary:**
  - **potential blocking on server-side if connections are established during transition**
  - **potential repeated reconnect attempts to failed primary if connections are not allowed until transition is completed**
  - **potential blocking within client-side middleware between notification of failure and indication of successful transition**
- **reconnection penalty at method invocation time when failover to a new primary has occurred**
  - **no mechanism to break and reestablish a connection except at method invocation**
  - **validate_connection() or _non_existent() CORBA pseudo-operations could be used to attempt to establish/verify connections to one or more servers**
    - **however, ORB behavior concerning maintaining and reusing connections is undefined**
- **reinstantiation of client stub due to unrecoverable errors or new group IOR**
  - **potential blocking time when discovering state from external components**

LW FT FOR DISTRIBUTED RT SYSTEMS RFP

- Lightweight Fault Tolerance for Distributed Real-time Systems RFP
  - released by the OMG RTESS PTF in June 06
  - initial submissions due in Nov 06
- Requires Platform Specific Models (PSM) for CORBA and RT CORBA
- Architectural approach is consistent with our assumptions
  - division of responsibilities between applications, FT-enabled middleware, external fault management components
- RFP scope is broader than just passive replication
  - covers active, semi-active, and passive capabilities

SUMMARY / RECOMMENDATIONS

- **Several promising RT Passive FT CORBA approaches identified**
- **Identified several areas where standards work would be beneficial**
  - **New standards**
  - **Clarification/modification of existing standards**
    - **Allow for additional points of control**
    - **Provide additional access to application-specific context**

- **Go forth, respond to the Lightweight Fault Tolerance for Distributed Real-Time Systems RFP, and develop viable solutions!!!**

ACKNOWLEDGEMENTS

- We would like to express our thanks to the following people for their valuable insights and assistance in support of this analysis:
  - Graham Dooley (Raytheon IDS)
  - John K. Black (Raytheon IDS)
  - John Robert (CMU SEI)
  - Gair Brown (NSWCDD)