Middleware Requirements for Remote Monitoring and Control

Frank Olken, H.-Arno Jacobsen, Chuck McParland
Lawrence Berkeley National Laboratory,
Berkeley, CA 94720
{olken,hajacobsen,cpmcparland}@lbl.gov

1.0 Introduction

We discuss middleware requirements for remote monitoring and control applications. These requirements reflect our experiences with the RBMO Project (remote building monitoring and operations) of commercial buildings HVAC systems (heating, ventilating, and air conditioning) and our participation in standards efforts related to control of large electric utility grids. Similar requirements exist for process control, and communications network monitoring and control applications.

The RBMO project is developing software systems to support remote monitoring and control of multiple buildings with heterogeneous legacy Energy Management Control Systems. across the Internet, using CORBA protocols. More details on the RBMO project can be found in Olken (Olken et al. 96) or on the RBMO web page. We have also recently become involved in some electric utility standards work, e.g., EPRI's UCA (Utility Communication Architecture and the newly formed OMG SIG on utility applications. The requirements described below reflect our experiences with these two application areas. We discuss only requirements which we deem (1) not largely and sufficiently supported by existing midddleware platforms and (2) mainly incurred by the remote monitoring and operation character of our system.

Key requirements we identified for this class of application:

- event notification
- fine grained time synchronization
- security
- naming and directory services
- mediation services
- unit and time conversion
- distributed configuration management
- dynamic substitution of components
- debugging support for distributed systems
- real-time ORBs
- performance enhancements in general

We have classified these requirements into the more general categories: basic services, security, distributed system management, and performance. The remainder of the paper is organized as follows. Sections 2 to 7 discuss in greater detail the above listed requirements for remote operation of devices in distributed systems. Section 3 relates these requirements to existing middleware platforms.

2.0 Basic Services

2.1 Event Notification Services

Central to any remote monitoring and control system is some mechanism for the remote sites to notify the monitoring sites of "interesting" events. Such event notifications need to be asynchronous, persistent, and (often) multicast. Ideally, events should be typed (so that certain processes can listen for specific types of events).

Event notification services address the need to propagate notification of significant occurrences to relevant system components. Depending on system functional requirements, this notification may be considered unreliable (e.g. periodic monitoring value updates) or may be required to be reliable (e.g. equipment status changes). In a distributed environment where system components can fail or become unreachable, the technical requirements for implementing these services can be substantial.

Although low level event delivery protocols are defined in CORBA, the semantics of describing and specifying higher level event service behavior remains (thus far) vendor-specific. Event service reliability and delivery persistence capabilities are not yet consistently defined and implemented. It is therefore important to identify and describe a significant subset of high level event service functionality. These event service functions should be sufficiently well described to allow multiple implementations that exhibit interoperable behavior. The ultimate goal will be to have a well understood set of high level event services available to any distributed control application in much the same way as TCP/IP naming services now are.

2.2 Data Push Architectures

Data push architectures are becoming an increasingly important method for distributing data to a large and variable set of remote applications. When combined with publish/subscribe techniques, the push model allows multiple applications to subscribe to data published by a single platform. Subscribing applications can be local to the publisher (i.e. located on it's LAN) or at some remote Internet site. Since the network topology between the publisher and subscriber can be determined, the publisher can use the most appropriate network protocol (i.e. unicast, multicast, unicast tunnel to multicast daemon, etc.).

Unfortunately, there are some conceptual problems in matching this model to the pure, synchronous, remote object method invocation model found in most ORBs. Ideally, control systems designers will find it most convenient to map individual data source and control objects onto single software objects. Readout of a data source by a remote application consists of a remote method invocation on the data source's software object. Unfortunately, data push architectures perform object method invocations at their own instigation and push data out at their own pace. Furthermore, for efficiency sake, data collected from a variety of objects are collected together and pushed out as a single message. Since the semantics that "tie" software object methods and data together becomes lost, these data collections end up being treated in various special, ad hoc ways.

As data push architectures grow in popularity, it will become increasingly important to bridge the gap between these two models. One method would be to formalize the data publishing protocol by making specific references to ORB interface repository entries. These references would "tie" elements of the data message to specific method invocations on specific objects. Upon reception, elements of the data message would unmarshalled in accordance with interface repository information.

2.3 Time Synchronization

Electric Power Grid control applications have stringent requirements for time synchronization across the distributed system (e.g., the entire Western United States). Phase differences among generator are key

performance indicators for electric power systems. Assuming 60 Hz power generation (16.7 ms/cycle) 1 degree phase difference amounts to 46 microseconds. To reliably measure this we will need time synchronization of about 10 microseconds. Recent practice (in the electric power industry and some network monitoring applications) is to use GPS satellite time bases for time synchronization.

3.0 Security and Privacy

Remote monitoring and operation of devices over the Internet requires strict security measures for several reasons: (1) to protect the monitored data from being stolen, corrupted and intentionally falsified; (2) to protect the device from malicious use (abuse) by impersonators; (3) to protect the device from unauthorized use; (4) commercial or national security concerns may require steps to preserve the privacy of monitoring data.

Most of our attention has focused on authentication concerns. Unauthorized operation of HVAC or electric power grid devices involves potential damage to expensive equipment and to the security of the electric power grid. It is therefore imperative to prevent unauthorized operation of this equipment.

To enforce these security requirements the following middleware functionality is needed:

- *Server authentication:* i.e., building site authentication; ensures the client application that it is truly operating on the intended site.
- *Client authentication:* i.e., building operator authentication; ensures building EMC system that an authorized client/operator is operating the equipment.
- *Integrity:* Non-corruption of data transferred. This prevents both malicious and false operation.
- *Confidentiality:* Data items transferred are encrypted. This prevents both malicious and false operation, as well as eavesdropping .
- *Secure invocation* of methods from client application to building EMCs, routed (i.e., delegated) through the central DBMS for logging purposes and to gather 'evidence' of who initiated an invocation when.
- Non-repudiation of control operations to guarantee liability.

Whereas the first four items are covered by SSL (Secure Socket Layer), delegation (needed for item 5), policies and roles (needed for 5 and 6) are not covered by these protocols. SSL is a simple and widely available security mechanism, developed particularly for web applications.

Secure invocation based on policies and roles, delegation of credentials and non-repudiation are part of more elaborate security mechanisms. The OMG CORBA Security Service specification, for example, identifies techniques to achieve such functionality. Vendor implementations of these services are unfortunately lagging behind, often not fully standard conform implemented, and modified through proprietary solutions.

Note that current practice typically employs private (physically separate) networks for building automation and power grid operations. If (to reduce cost) shared communications networks (e.g., Internet) are to be employed then security services will have to be available at the device controller level.

Privacy considerations may arise in building monitoring data, inasmuch as some of this data could disclose information about unusual levels on activity by a firm or agency, e.g., merger preparations or war planning. Detailed information on electricity consumption of competitors may also be useful for competitive intelligence. Conventional encryption services should suffice for these concerns.

4.0 Semantic Heterogeneity

4.1 Naming and Directory Services

Effective configuration description and control is the bane of distributed systems. Distributed systems are constructed, in part, by binding together objects distributed over many platforms. Without a mechanism for describing these bindings, the resulting system lacks flexibility and it's complexity remains hidden to all but the original implementors. Naming services provide a way to bind meaningful human names to object instances.

These problems take on greater importance in real-time control environments where systems must accommodate equipment replacement and/or upgrade cycles as well as "live" system reconfiguration. Stable and ubiquitous naming and directory services are a key tool in the design of systems that will accommodate these changes. Examples include the CORBA naming services and LDAP Directory services.

The function of these services goes well beyond the nuts and bolts issues of publishing specific object names and attributes. An effective distributed naming and directory service can expose, through the use of multiple hierarchical views, objects aggregated by function, location, control sub-system, etc. In addition, these services can provide a single, ubiquitous repository for object meta information. In the case of control systems, this can include information about sensors and instrumentation used in generating object method values, resolution of object name aliases due to differing local and supervisory object namespaces, and provide automated mechanisms for locating physical and/or logical control system components.

4.2 Mediation Services

While standards efforts offer some hope in the long run for reconciling semantic heterogeneity among various network devices and controllers (HVAC or electric power grid), the existence of many incompatible legacy systems requires that some facilities for mapping between the global schema and local schemas be provided. At present, e.g., in RBMO (and other systems) this is often done in an *ad hoc* fashion with hand-crafted mapping tables. Such facilities are variously referred to as mediators (partial mappings, late binding) or schema integration tools (nearly complete mappings, early binding). Such tools will require access to metadata (e.g. DB or object schemas). Examples of related work include OMG Object Interface Repository, and Meta Object Facilities.

4.2 Unit Conversion Services

We have found it necessary to provide measurement unit conversion capabilities for our RBMO project. Our problems arise in part from heterogeneous use communities employing variously English or metric units (SI). However, even among users of metric units there are often idiosyncratic units conventions in particular disciplines (e.g., the use of Angstroms vs. nanometers). Units information either needs to accompany measurements, or being readily attainable via metadata queries. The ubiquity of these issues (in both industrial measurement and commercial trading) and their invariance suggests the provision of standard services. Note that unit conversions may be either: invariant (true equivalences), or material and state dependent. The latter type of conversions typically involve material properties, e.g., density, and conversions between mass and volume.

5.0 System Management

5.1 Static vs. Dynamic Schemas

Traditional database designs specify complete detailed schemas statically. Such static schemas facilitate

systematic design, query optimization, etc. However, for many monitoring and control applications (shipped as shrink wrap software), it is desirable to be able to alter the schema (e.g., to accommodate new types of equipment) without rebuilding the DB or recompiling the code. Examples of such approaches include: frame-based knowledge representation systems, and the dynamic invocation interface of CORBA, together with interface repositories. Because such dynamic schema capabilities are typically significantly slower than static interfaces, some groups have built hybrid schemas - examples include the International Alliance for Interoperability (building data model) which provide both static schemas (e.g., for geometry) and dynamic schemas (frames) for extensibility. Conceivably, on-the-fly compilation techniques might provide a reasonable performance compromise.

5.2 Configuration management and component distribution

The remote entity to be monitored and controlled, may often be located at great distances from the central system development and installation site, and it may not always be possible to travel to the remote site to install, upgrade and maintain the system. It is therefore essential to provide support for static and dynamic software distribution over the network onto heterogeneous target platforms.

With static software distribution we mean the distribution of patches, upgrades, new versions, and entire releases, as well as their automatic installation. For intranets and homogeneous environments this problem has already been addressed and tools are available to support the developer with the tasks of configuration management and version control; automatic installation, however, is still an ad hoc procedure not globally supported. Moreover, the more general issue of wide-area network software distribution aiming at highly heterogeneous environments is still an open question. Additionally, critical sites will often be secured by firewalls which opens up the question of secure software distribution over the Internet supported by middleware services.

Unless the software has been designed for runtime maintenance, it is often not feasible to halt a system in operation to exchange components, since continuous operation is required (e.g., for security reasons, due to continuous data acquisition.) With dynamic software distribution we mean the substitution of software components, modules and objects at runtime and design to permit such use.

5.2 Message logging, monitoring and debugging support

In the development process of a distributed application it becomes often necessary to monitor and debug the evolving system. Monitoring serves mainly to pinpoint performance bottlenecks and fine tune the system. In later stages monitoring may also be applied to build network aware applications, i.e., applications which adapt their communication pattern to available resources.

Little support is available that aids the developer monitor and debug the distributed system under development. Only, recently have tools been announced for specific middleware implementations (OrbixManager [Iona 97] and Object/ Observer, [Black and White 97]). A standards effort aiming to specify a monitoring service (including a set of performance metrics) and debugging hooks in the middleware platform would greatly benefit the system developer. Within the OMG Utility SIG effort is underway to design a logging facility for events to better captured causality relationships of the components involved.

6.0 Performance Issues

Our experience with existing middleware systems (i.e., CORBA) was that the facilities were too slow to permit the modeling of individual measurement points as separate objects with separate operations. It proved necessary to aggregate many of these measurement points together into named collections and to

query the groups of points. This dramatically reduced the number of CORBA invocations (messages) and improved performance. We conclude that we need both faster ORBs and automatic (or convenient semi-automatic) tools for facilitating such aggregations.

7.0 Discussion and conclusion

In this section we summarize our experience by comparing the capabilities of existing middleware platforms with respect to the issues raised in the previous section. These platforms are: CORBA, DCOM, MMS (Manufacturing Message Specification), Java/RMI, DCE. The following two table constitute the comparison.

Requirement/ Platform	Event Notific-ation	Data Push	Time Sync.	Security	Naming/ Directory	Unit conversion
CORBA	Event Service, limited event typing	no	not yet	Security service (roles, policies, delegation)	yes	no
DCOM	?	no	?	comparable to DCE	yes	no
MMS	Event service with typed events	no explicit support	no	no (primarily used in LANs)	use of other's services possible	no
Java/RMI	supported; EventClass	supported; JavaBeans via events	no	Java's security model	use of other's services possible	no
DCE	No Event Service available	no explicit support	Time service & synchronization	Security service (ACLs, delegation)	yes	no

Requirement/ Platform	Dynamic Schemata	Config. Manag.	Logging & Debugging	Real time capabilities
CORBA	Reflexive sys. arch.; IR; dyn. invocation	no	vendor specific support	in progress
DCOM	yes	yes (versioning only)	?	?
MMS	no	no	in progress	goal: real-time operation & control
Java/RMI	JavaBeans: Discovery & Registration	no	in progress	no

DCE	no	Distributed System Management specifications (in progress)	RFPs issued	in progress

Bibliography

- **T. Cross, Instrumenting an Application with OrbixManager , CORBA Management workshop, Dublin, Ireland, September, 1997.**
- **C. White, Observing Traffic on the CORBA Bus - The Object/Observer.doc.zip(7.5k), CORBA Management workshop, Dublin, Ireland, September, 1997.**
- **Olken, F., McParland, C., Piette, M.A., Sartor, D., Selkowitiz, S., Remote Building Monitoring and Control, Proceedings of the ACEEE Summer Study Conference, Asilomar, CA, August 25-31, 1996. vol. 4, pages 285-296**

Contact author Frank Olken at Lawrence Berkeley National Laboratory. Email: olken@lbl.gov Last updated: November 24, 1997