Java- and CORBA-Based Network Management

Mika Leppinen, Pekka Pulkkinen, and Aapo Rautiainen
Nokia Research Center

Nokia developed the Distributed Computing Platform prototype to support distributed telecommunications services. In the process, engineers learned several lessons about Java and CORBA integration.

Integrated, *distributed* management of heterogeneous networks and services is becoming crucial to telecommunications. Such new management systems must often use off-the-shelf components and leverage previous investments in legacy management applications. Yet despite implementation difficulties, distributing management applications provides scalability as well as cost and performance benefits.

Much of the telecommunications industry uses a network architecture based on CMIP (Common Management Information Protocol)[1] to manage equipment networks and services. In yet another arena—the Internet—the SNMP (Simple Network Management Protocol)[2] has gained widespread acceptance. Both of these management frameworks will continue to coexist far into the future. Thus, to provide distributed network management, the telecommunications industry must accommodate both.

NOKIA DCP PROTOTYPE

A standard telecommunications platform must provide many types of services. Such a platform should support distributed-object computing, network management protocols, Java,[3] and Web technologies. Nokia has incorporated these features into the Distributed Computing Platform prototype[4,5] to create, manage, and invoke distributed telecom services.

CORBA based

We based our system on the Common Object Request Broker Architecture,[6] which addresses these

0018-9162/97/$10.00 © 1997 IEEE

distribution, interface, and integration problems. CORBA also provides a set of distributed services such as uniform resource naming, event handling, and service trading. The Object Management Group,[7] however, designed CORBA to provide an architecture for distributed-object computing, not network management. To manage network resources, you need managed-object models and protocols, which is what we added to DCP. In developing DCP, we learned many lessons about integrating Java and CORBA (see the "Lessons Learned: Java and CORBA Integration" sidebar).

CORBA does provide an Interface Description Language to facilitate both service development and service integration. IDL is a neutral, portable specification language, which compilers can map to other languages (C++ and Java) to provide executable code. We can use this code to integrate different management protocols and systems.

Prototype description

DCP, illustrated in Figure 1, implements the components of all services as objects. These objects fall into three categories defined by the entities for which they provide communication paths.

- *CMIP distributed-objects* communicate to applications via managed-object IDL interfaces. A CMIP-string package passes protocol data units between CMIP systems encapsulated as objects. CORBA naming and trading services facilitate communication among distributed objects.
- *SNMP-based systems* interface to DCP via the CMIP-SNMP gateway.
- *Users* access the network from Web browsers, sometimes through CGI gateways, server extensions, and Java or HTTP daemons.

IDL interfaces. GDMO (Guidelines for the Definition of Managed Objects) and ASN.1 (Abstract Syntax Notation One) are standards that provide a common transfer syntax so that different applications can exchange data structures. Converting them to IDL makes it possible to integrate CMIP and SNMP systems using CORBA. We wrote GDMO/ASN.1-to-IDL compilers to provide a systematic way to produce IDL definitions for GDMO models and ASN.1 definitions. The compilers use a repository for storing knowledge about

- GDMO objects, using management knowledge metadata object definitions, and
- IDL definitions, stored in the CORBA interface repository.

CMIP-string package. All messages between CMIP systems can be encoded in an ASCII string representation called a CMIP-string PDU (protocol data unit). PDUs are special messages that pass necessary information between protocol layers. A DCP package builds, parses, and passes CMIP-string PDUs using CORBA, transferring protocol information using the approach outlined by W. Allen.[8]

The application programming interface allows use of any syntax defined in ASN.1. However, all integer and real values must be converted to/from the string format, a distinct disadvantage. CMIP-string PDUs can also become very large.

Naming and trading services. The naming service is the principal mechanism used by objects in the DCP

Lessons Learned: Java and CORBA Integration

- Java is "programmer friendly"—it provides automatic garbage collection and quick compilation—so it is attractive for prototype development. We found Java to be a more attractive language with which to develop CORBA applications than C++ because it simplifies memory handling, has an exception mechanism that works well, and does not require porting.
- Unfortunately, today's commercial development tools still leave a lot to be desired. For example, only one of the Java development tools we tested supported the Java layout models, and many tools were too slow to use on real applications.
- A Java-based object request broker is an attractive concept because it combines advantages provided by CORBA and Java. CORBA provides scalable distributed services and is especially suitable for legacy application integration. The portability of Java source code and bytecode make it suitable for distributed computing. Iona's OrbixWeb object request broker provides robust support for integrating Java and CORBA.
- A browser such as Netscape Navigator uses a very restricted security policy. An applet can connect via a socket only to the machine from which it was loaded. This thus restricts the CORBA daemon and servers to locations within the same machine from which the applet was loaded. Otherwise, a proxy must be used.
- Applets must pass object references to each other. Naming and trading services allow clients to locate object services based either on name or service properties. In an environment where different object request brokers communicate, object services are invoked through interoperable object references (IOR). String IORs are long and unreadable, so cutting and pasting them is impractical. Presenting IORs as icons would be one solution. Future application should include drag-and-drop support for IORs.

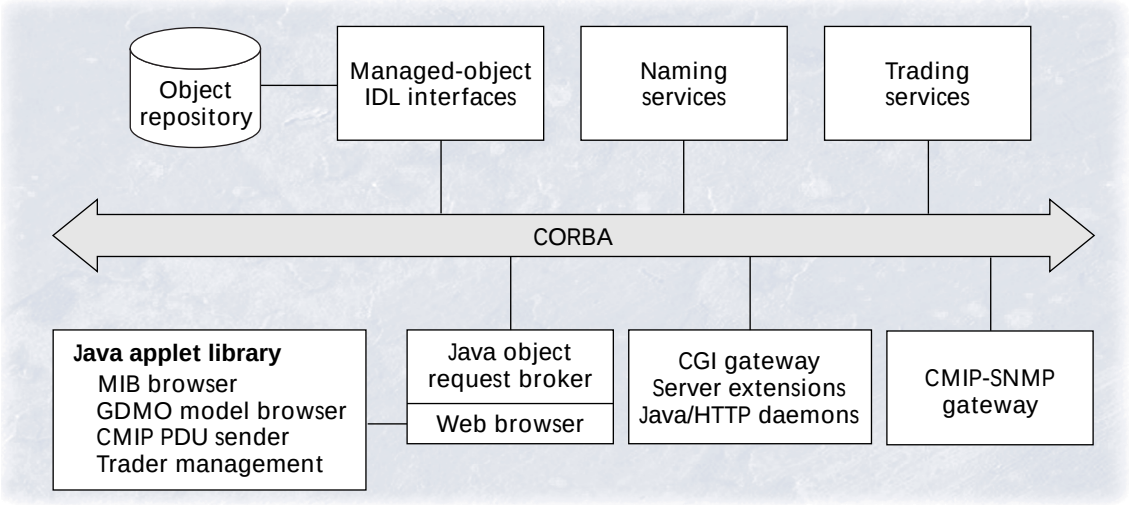

Figure 1. Nokia Distributed Computing Platform prototype.

prototype to locate other objects. It defines operations via which a client can reference an object by a name. Our prototype contains an implementation of a specially designed X.500 names library that allows conversion of X.500 names to CORBA names.

The trading service provides another mechanism for dynamically finding distributed services. Service providers can advertise service offers to a trader. Potential service users (importers) can search information on available services and their accessibility. An importer may use the standard constraint language to restrict the properties of service offers. After a successful match, the importer can interact with the service provider. The next phase of DCP prototype will use the trading service in the intelligent network service management.

CMIP-SNMP gateway. This gateway enables management of SNMP devices by using the CMIS[9] (Common Management Information Services) standard. The gateway supports the sending and receiving of CMIP and SNMP protocol and timer events, and it dispatches methods to managed objects residing in the gateway.

The gateway's managed objects communicate with Internet agents by using the SNMP protocol to carry out the operations invoked through their IDL interfaces. The gateway can handle several simultaneous managers and SNMP agents.

By mapping them to SNMP messages, the gateway emulates CMIS services. CMIP messages are received and sent as string PDUs over CORBA. Because we use a stateless approach, the gateway does not store the Internet management information base (MIB) data. (MIBs are databases maintained by each object to record its network status.) Instead, for each CMIS-based service request it receives, the gateway generates one or more SNMP requests to the Internet agent to achieve the function of the CMIS-based request.

User access. Web browsers, many of which will be Java-enabled clients, communicate to DCP via CORBA object request brokers. We have developed some special features to facilitate communications between DCP and Java clients.

JAVA CLIENT COMMUNICATION

We implemented these Java clients (applets) so they communicate to DCP using *proxy communication*. Basing them on CORBA binds them to a specific proxy manager server. The proxy manager implements an IDL interface that provides two operations: sendTask and pollTask. A client sends a PDU to the manager through sendTask. Since the applets are CORBA-based clients and sent tasks are processed asynchronously, the manager cannot notify the client when a task is completed. Therefore the proxy manager provides pollTask to poll the state of submitted tasks. If a task is completed, the manager will return the results.

An autonomous thread processes the reception of results. The package uses an observer/observable model, in which the applet sending a task through the communication package registers itself as an observer who wants notification of task completion.

In addition, three reusable Java packages provide Web access to the platform:

- *Java StringPdus.* Because communication within the platform is via string PDUs, we ported the string PDU package (originally written in C++) to Java to enable applets to access the platform components.
- *LayoutTree.* Because we can easily present most information that needs to be handled as a tree, we created a generic tree package with a layout algorithm. The package uses the model/view approach to separate the actual tree object from the canvas visualizing the tree.
- *ProxyCommunication.* Due to current Web browser security restrictions, the communication between applets and CORBA servers is unidirectional—from the applet to the CORBA server. We wrapped the server-to-client communication into one Java package to provide the previously

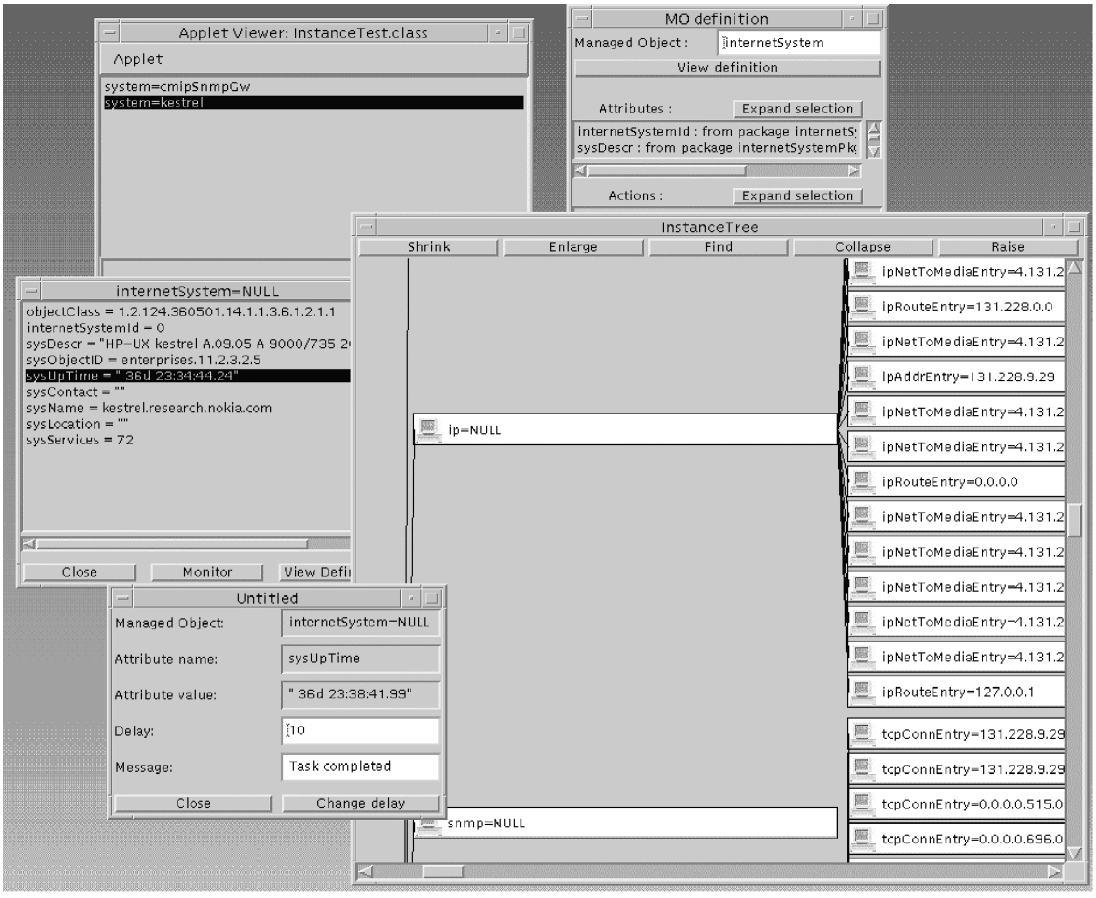

Figure 2. MIB browser applet.

discussed proxy communication methods for sending PDUs.

JAVA APPLETS

To provide object-oriented access to network management services, we wrote several applets for DCP.

MIB browser

This applet provides a graphical view of the selected MIB. At the beginning, the applet shows a list of system objects. A user selects one system object, and the applet sends a scoped-Get operation with the selected system object as a base managed-object instance. (A scoped-Get returns a set of objects, as defined by the scope.) The applet receives the resulting PDU and, based on distinguished names, builds a MIB tree.

Figure 2 shows the MIB browser applet. The instance tree window (lower right) shows a part of the selected system object's MIB and provides functionality for shrinking and enlarging the tree, finding a specific node, and expanding and collapsing subtrees. When a user selects a MIB object, a window (far left) displays the object's attributes. Figure 2 shows those for the internetSystem object. From the attribute window, the user can monitor some specific attribute or view the GDMO definition for the selected object provided by the GDMO-to-IDL compiler. A user can set a monitor for some attribute (shown for sysUpTime in Figure 2). The attribute value is polled periodically, with a user-defined frequency. From the attribute window, users can monitor a specific attribute (lower left) or view the GDMO definition (upper right) for the selected object provided by the GDMO-to-IDL compiler.

GDMO model browser

GDMO model metadata information is stored in an object-oriented database and accessed through the GDMO-to-IDL compiler. The GDMO model browser applet provides an inheritance-tree-based user interface to the model.

When the applet starts, it connects to the GDMO-to-IDL compiler and asks for inheritance information for the GDMO metadata. The compiler returns the inheritance information, which the applet parses, and the inheritance tree window opens. The operations provided by the inheritance tree window are the same as for the instance tree window. When a user selects a node, the applet opens a managed-object information window showing the attributes, actions, and notifications for the selected managed object.

CMIP PDU sender

For administrative tasks, such as managed-object creation and MIB-attribute value setting, the platform provides an applet that can send raw CMIP PDUs to managed objects. Raw CMIP PDUs are those for which users must write all fields of the PDU (in the

ASCII-string PDU format). The applet is also capable of receiving notifications from managed objects.

The applet also provides an operation that sends stored PDUs. In this way, users can write commonly used PDUs to files and load them (instead of rewriting every time).

Trader management

We can use applets to add new service type descriptions and service offers. The service type properties are limited to string, long, double, Boolean, or sequences of these types. The applet handles Iona Orbix object references in strings. Property values are parsed and converted to form the trading-service acceptances. Users can search services by evaluating constraint and preference expressions against the service type properties. If the applet finds matches, it lists the corresponding object reference strings, and the user may cut and paste them for later use.

In the first phase of DCP prototype development, we emphasized proof of concept—that it was possible to prototype the integration of the Web, Java applets, and CORBA objects with legacy network management systems. Java and HTTP provide remote access to network management functionality from any Web browser.

During the project it became evident that even though current commercial object request brokers are not optimized, CORBA technology is mature enough to serve as a base for telecom software platform development.

In our next phase, we will consider such issues as security, fault tolerance, multithreading, and support of real-time processing. On the Java side, we will continue to work with Java ORBs and also investigate the use of mobile terminals in network management. We will also investigate pure Java-based concepts such as Java remote method invocation (RMI) and serialization. RMI seems to be a good solution for distribution in homogeneous, Java-virtual-machine-based environments. Other interesting Java issues include the use of Java Beans and Java Management application programming interfaces. ❖

Acknowledgments

We thank James Reilly for his comments and the Nokia DCP prototype project team: Sakari Rahkila, Susanne Stenberg, Petri Nuuttila, Jyri Virtanen, and Wei Chiang.

References

1. *ISO 9596, Information Technology—Open Systems Interconnection—Common Management Information Protocol*, Int'l Organization for Standardization, Geneva, 1991.
2. J. Case et al., *RFC 1157, A Simple Network Management Protocol (SNMP)*, http://sunsite.auc.dk/RFC/rfc/rfc1157.htm, May 1990.
3. *The Java Language: A White Paper*, http://www.javasoft.com/, 1995.
4. S. Rahkila and S. Stenberg, "Experiences on Building a Distributed Computing Platform Prototype for Telecom Network and Service Management," *Proc. IFIP/IEEE Int'l Symp. Integrated Network Management*, IEEE Press, Piscataway, N.J., 1997.
5. S. Rahkila and S. Stenberg, "Experiences on Integration of Network Management and a Distributed Computing Platform," *Proc. 30th Hawaii Int'l Conf. System Sciences*, IEEE CS Press, Los Alamitos, Calif., 1997, pp. 140-149.
6. *The Common Object Request Broker: Architecture and Specification, Revision 2.0*, Object Management Group, Framingham, Mass., July 1995.
7. *Object Management Architecture Guide, Revision 2.0*, 2nd edition, Object Management Group, Framingham, Mass., Sept. 1992.
8. W. Allen, "An Alternate API for Representing ANS.1 Values," *CMIP Run!*, IBM Corp., 1994.
9. *ISO 9595, Information Technology—Open Systems Interconnection—Common Management Information Service Definition*, Int'l Organization for Standardization, Geneva, 1991.

Mika Leppinen is a project manager at Nokia Research Center. His research interests include distributed systems, fault tolerance, computer security, and design patterns. Leppinen received a master's degree in computer science from the Helsinki University of Technology.

Pekka Pulkkinen is a research engineer at Nokia Research Center. He received a master's degree in computer science from the University of Joensuu.

Aapo Rautiainen is a research engineer at Nokia Research Center. His research interests include distributed systems, fault tolerance, group communication, and formal methods. Rautiainen received a master's degree in computer science from the Helsinki University of Technology.

Contact the authors at Nokia Research Center, Distributed Computing Platforms, PO Box 422, FIN-00045 Nokia Group, Finland; {mika.leppinen, pekka.pulkkinen, aapo.rautiainen}@research.nokia.com.

Even though current ORBS are not optimized, CORBA is mature enough to serve as a base for telecom software development.