Adaptive Middleware
For
Challenged Networks

Marc Born, Tom Ritter, Rudolf Schreiner
Fraunhofer FOKUS
ObjectSecurity Ltd.
**Adaptive Middleware for Challenged Networks**

**Fraunhofer FOKUS Germany**

Approved for public release, distribution unlimited.

**Security Classification of:**

<table>
<thead>
<tr>
<th>a. REPORT</th>
<th>b. ABSTRACT</th>
<th>c. THIS PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>unclassified</td>
<td>unclassified</td>
<td>unclassified</td>
</tr>
</tbody>
</table>
Agenda

- Issues of classical network layering
- Needed: secure component middleware
- Secure Distributed Middleware Project
  - Enhanced CORBA Component Model (CCM)
  - OpenPMF Policy Management Framework implementation
  - Qedo CCM implementation
- Conclusion
Classical Layering Issues

In real-world systems, layered protocol stacks have many issues:

- Functionality mixed up in different layers
- Loss of functionality
- Too tight coupling for replaceability
- Too loose coupling for adaptivity
- Security issues
- This leads to messy protocol stacks and obscure protocols (WAP, TCP/IP over ATM)
What do we really need?

- Consider networking from an application point of view
- Programmers mainly need some standard high level communications patterns:
  - Synchronous invocations (Request/Replay)
  - Asynchronous events
  - Streams
- QoS requirements need to be defined and fulfilled
- Low level “plumbing” is of little interest to application programmer
Component Middleware

- Component-based middleware offers a solution based on two layers:
  - Component implements business functionality
  - Container provides adaptive infrastructure transparent to component
    - Communications
    - Services

- Issue: COTS middleware does not meet all requirements of complex (military) systems

- Goal: Development of a secure, flexible and adaptive middleware based on the CORBA Components Model (CCM)
Secure Distributed Middleware Project

- Based on CORBA Components Model (CCM)
  - Improves object-oriented programming model
  - Development of independent modules: Components
  - Application development by assembling components
  - Supports asynchronous and synchronous communications
- Adapting CCM to the requirements of complex C4I applications
- Main extensions
  - Flexible container to implement services
  - Support for Quality of Service
  - Streams
  - Policy management framework esp. for security
- Future: Additional low level protocols
CCM Containermodell

Client

Home

Component

Container

callbacks

Policy Evaluator

QoS Provider

Network Protocols
Container Provides Network Abstraction

- Container handles all communications and abstracts from low level protocols:
  - Protocols transparently replaceable
- Container provides high level API to components for:
  - Addressing
  - Connections
  - Synchronous invocations (request, reply)
  - Asynchronous communications (events)
  - Streams
Container Provides Adaptivity

- Container manages and implements all non-functional aspects (QoS, security)
- Adaptivity by
  - Policies (QoS, security)
  - Scripts (automatic reconfiguration)
  - Environment-specific containers possible
- Enforcement/implementation using “Flexible Container”
  - Context interfaces
  - Interception points
  - Future: Pluggable protocols
    - Integration of SPREAD (multicast protocol) ongoing
    - Changing communication protocols online
OpenPMF Policy Management Framework

- Generic framework for policy specification, storage, enforcement:
  - Policy model defined using MOF
  - Policy Repository
  - Policy Definition Language (mechanism and platform independent)
  - Mappings to specific platforms

- Clear separation of functional and non functional aspects

- Currently used for CCM and CORBA security
  - Supports different security models (DAC, RBAC, MAC), information filtering and delegation

- Future: Support for other policy types, e.g. QoS, and automatic reconfiguration
Container as Runtime Environment

Container as flexible runtime environment also provides:

- Life cycle management
- Connection topology
- Well-defined interfaces for component implementation
- Flexible services (naming, events, transaction, persistence...)
- Standardized and uniform service configuration
Qedo CCM Implementation

- Based on MICO CORBA ORB with enhanced security support
  - CSIv2 protocol & SL3 API
  - ATLAS authorisation token server
- Enhanced CCM implementation in C++
- Extensions:
  - Component level interceptors
  - Streams support
- OpenPMF integration
- Currently used for prototypes of C4I applications
Qedo CCM Tool Chain

Qedo contains an extended CCM tool chain:

- Based on Meta Object Facilities (MOF)
- Model Driven Architecture (MDA) integration
- IDL/CIDL generators
- Assembly and packaging
- Testing (component based and application based)
- Deployment (even in large and heterogeneous environments)
- Administration and monitoring
Conclusion

- CCM abstracts from network infrastructure
- Two layer architecture
  - Container provides infrastructure and adaptivity
  - Component implements business logic
- Enhanced CCM provides an advanced framework for developing and operating of complex distributed applications on top of a wide range of (wireless) protocols
- OpenPMF as sophisticated security architecture
- Most promising middleware for C4I applications
Contact

- Marc Borc: born@fokus.fraunhofer.de
- Tom Ritter: ritter@fokus.fraunhofer.de
- Rudolf Schreiner: ras@objectsecurity.com

- Qedo: http://qedo.berlios.de
- OpenPMF: http://www.openpmf.org