Building Adaptive and Agile Applications Using Intrusion Detection and Response

Joseph P. Loyall, Partha P. Pal, and Richard E. Schantz

{jloyall, ppal, schantz}@bbn.com

BBN Technologies

Franklin Webber

franklin.webber@computer.org

http://www.dist-systems.bbn.com/tech/QuO

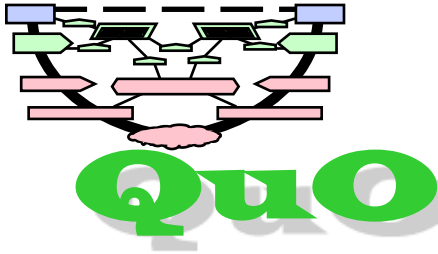
Outline

Motivation & Context

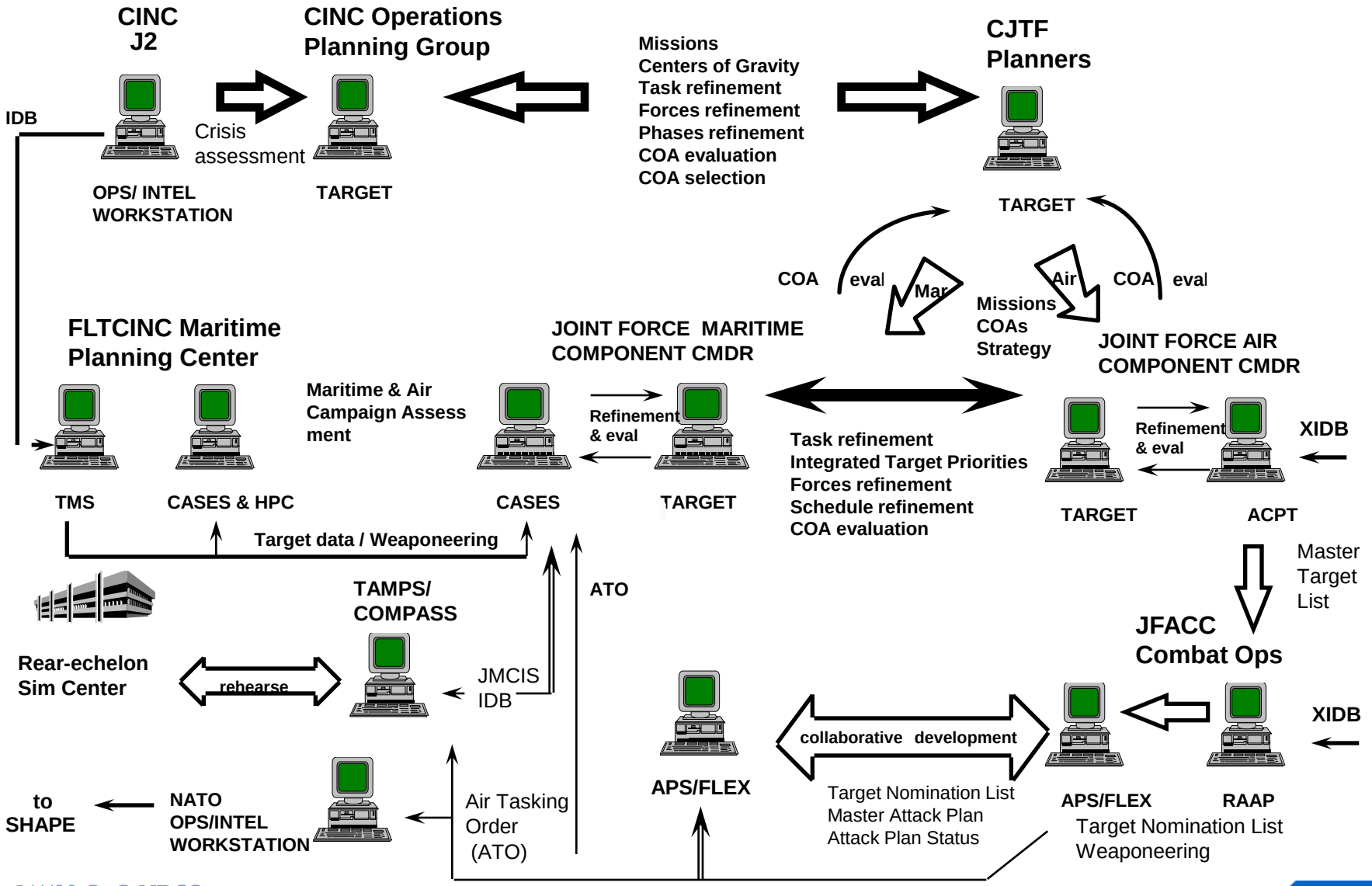
Overview of QuO technology

Building adaptive, intrusion-aware applications using QuO

Conclusions and Issues



As applications become more distributed and complex, their needs for adaptability, security, and survivability increase



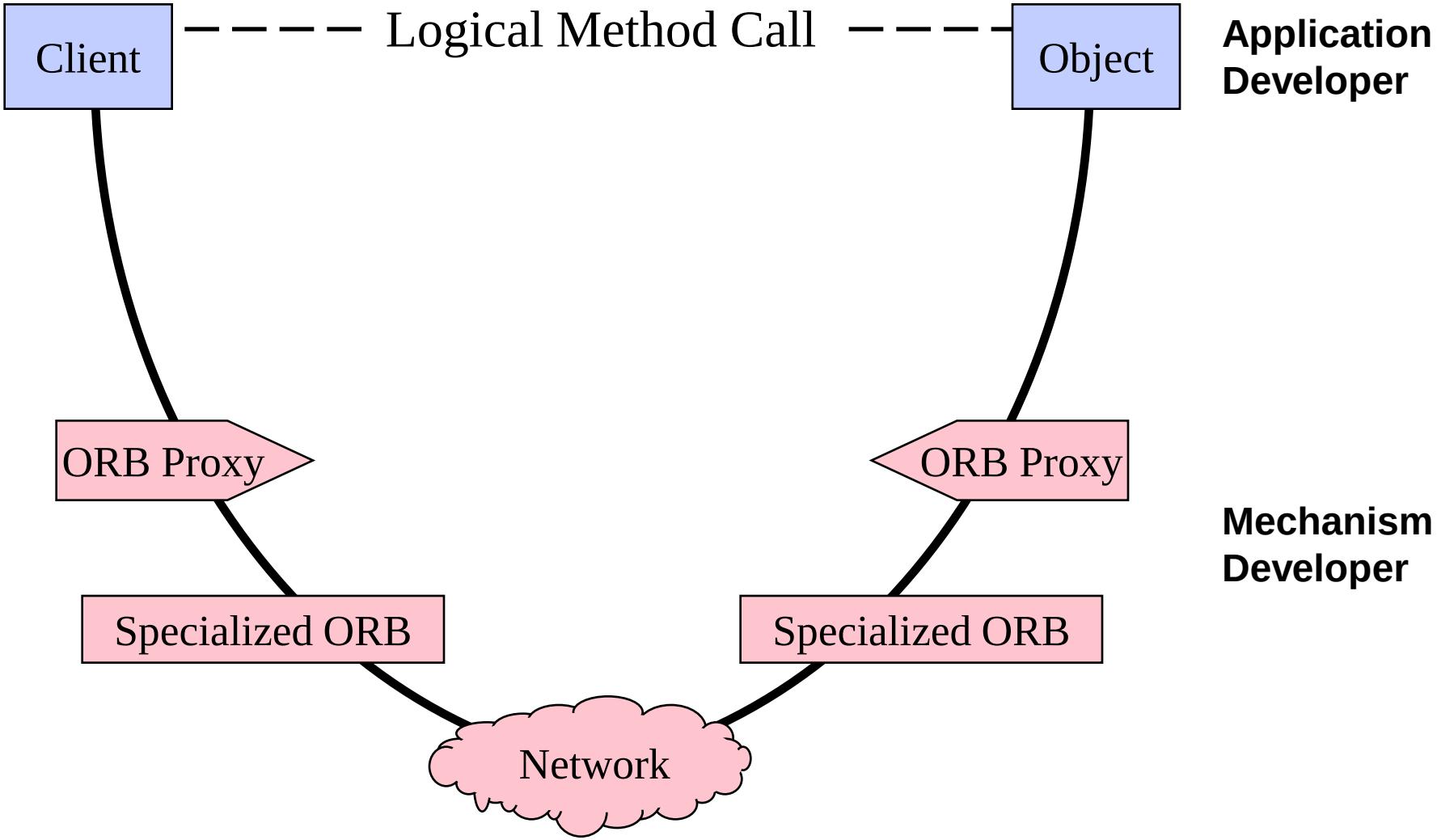
Large scale information systems are vulnerable to attack

- Increasingly so because of their distributed, networked nature
- Distributed object systems and wide-area networks offer increased chances of failure or attack
- Increasing reliance on COTS increases possibility of attacks and ease of propagating attacks
- Homogeneity in architectures, platforms, OSs, components, algorithms increase vulnerability to attack

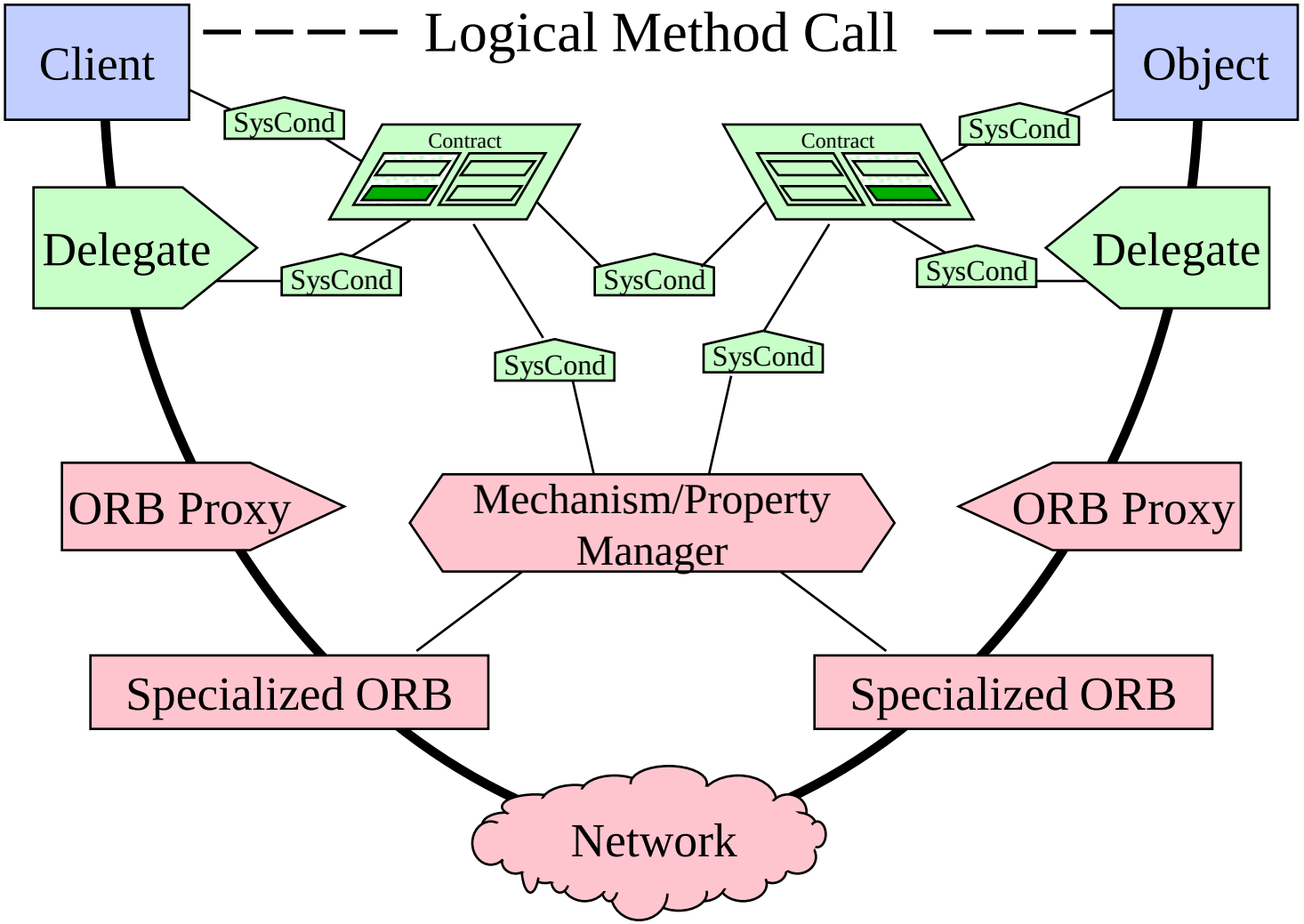
We are building upon research that we're conducting in adaptive distributed systems

- Developing technology to support specification, control, and measurement of QoS in distributed systems, including security, survivability, dependability, performance, and resource management
- Addressing the following problems of critical, networked applications:
  - No effective way to control behavior of critical applications in today's highly networked environment
  - Gap between low-level mechanisms that control resources and high-level strategies appropriate for critical applications
  - Researchers generally working on one piece of the solution, e.g.,
    - Focus on one critical QoS property: security, real time behavior, fault tolerance, ...
    - Focus on implications of one time epoch
    - Functional integration has taken precedence over system properties
- End goal is to support building integrated QoS adaptive applications (with varying requirements on granularity of changing behavior):
  - adaptable: change at runtime while application/service running
  - reconfigurable: change at runtime while application/service halted
  - evolvable: change at development time

Simplified DOC (CORBA) Runtime Components



QuO adds specification, measurement, and adaptation into the distributed object model



Application Developer

QuO Developer

Mechanism Developer

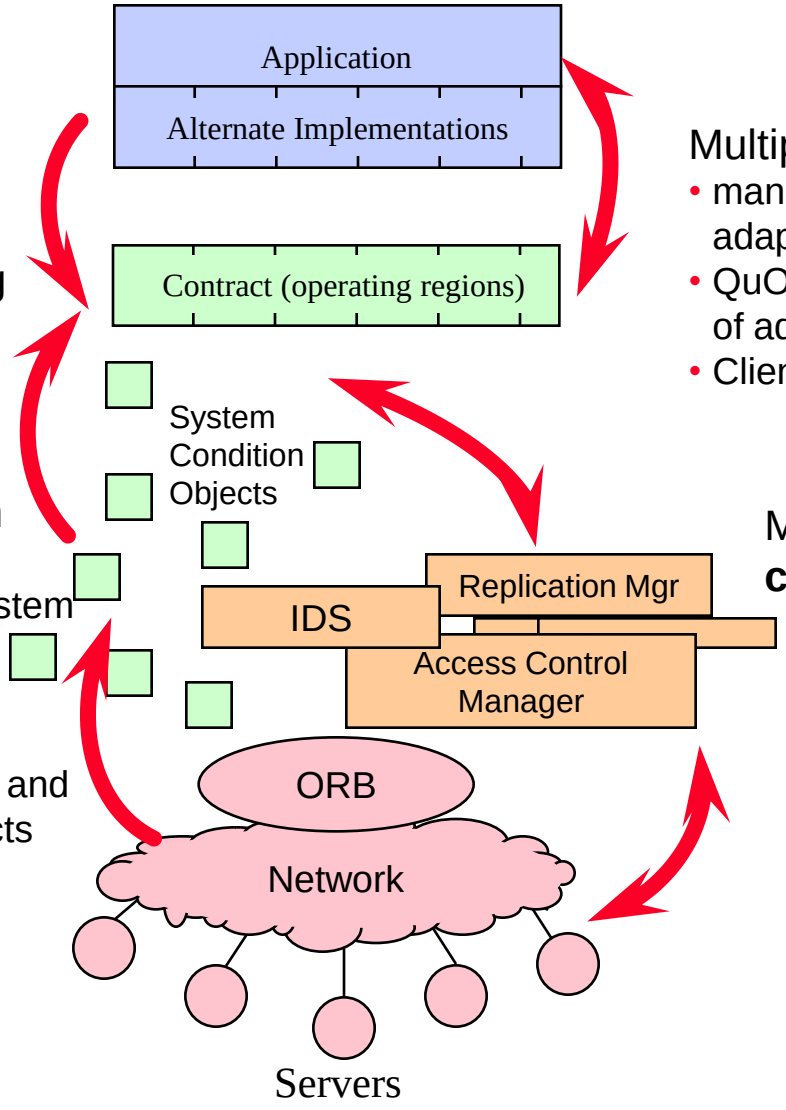
Client **Network** **Server**

A QuO application contains additional components (from traditional DOC applications)

- **Contracts** summarize the possible states of QoS in the system and behavior to trigger when QoS changes
  - Regions can be nested, representing different epochs at which QoS information becomes available, e.g., *negotiated regions* represent the levels of service a client expects to receive and a server expects to provide, while *reality regions* represent observed levels of service
  - Regions are defined by *predicates* over system condition objects
  - *Transitions* specify behavior to trigger when the active regions change
- **System condition objects** are used to measure and control QoS
  - Provide interfaces to system resources, client and object expectations, mechanisms, managers, and specialized ORB functions
  - Changes in system condition objects observed by contracts can cause region transitions
  - Methods on system condition objects can be used to access QoS controls provided by resources, mechanisms, managers, and ORBs
- **Delegates** implement the QoS specific adaptivity behavior
  - Upon method call/return, delegate can check the current contract state and choose behavior based upon the current state of QoS
  - For example, delegate can choose between alternate methods, alternate remote object bindings, perform local processing of data, or simply pass the method call or return through

QuO applications specify, control, monitor, and adapt to QoS in the system

Specification of operating regions, alternate implementations, and adaptation strategies using QuO's QDL



Multiple layers of **adaptation**

- managers and mechanisms can adapt to changes in the system
- QuO contracts provide another layer of adaptation
- Client and user can also adapt

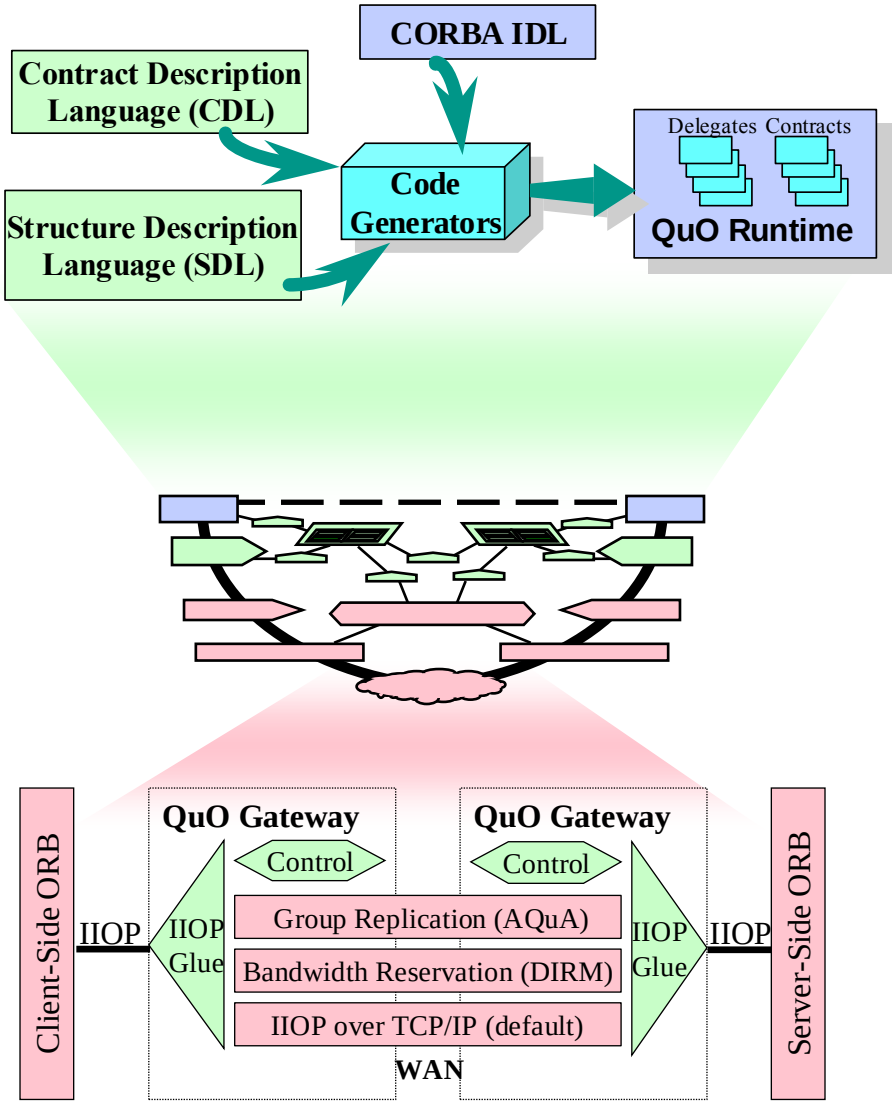
System condition objects **monitor** QoS in the system

- system condition objects recognize changes in the system and notify the contracts that observe them
- QuO contracts notify client programs, users, managers, and other system condition objects through transition behavior

Mechanisms and managers **control** QoS in the system

- a layer below QuO that provides ORB-level services, such as managed communication, replication, or security
- contracts and delegates interface to these services through system condition objects

The **QuO Toolkit** provides tools for building adaptive applications



- Quality Description Languages (QDL)
  - Contract description language, adaptive behavior description language, connector setup language
  - Code generators that generate Java and C++ code for contracts, delegates, creation, and initialization
- System Condition Objects, implemented as CORBA objects
- QuO Runtime Kernel
  - Contract evaluator
  - Factory object which instantiates contract and system condition objects
- Instrumentation library
- QuO gateway
  - Insertion of special purpose transport layers and adaptation below the ORB

Benefits of QuO adaptable software to IA, survivability, and security

- Development of intrusion- and security-aware applications
  - Can recognize application-level patterns of usage and attack to aid IDSs and security managers, e.g., denials of service, corrupted data
  - Probes in applications can gather information that traditional IDSs and security systems (which treat applications as black boxes) don't have
- Development of survivable applications
  - Applications that can adapt to changes in their environment to continue execution in the face of potential attacks
- Integration and interfacing of IDSs at the application level
  - Receive alerts from complementary IDSs
  - Dynamically engage IDSs to increase coverage and security
- Integration and interfacing of IDSs and security managers with other resource managers
  - Some managers, such as dependability managers and resource managers, provide services useful for security, survivability, and intrusion detection

An example survivable application using QuO as an adaptive integration framework

- Intrusion detection systems (IDSs)
  - Tripwire, a COTS file system integrity checker (Purdue/Tripwire Security)
  - A simple directory access checker
- Access controls enforcing a global policy in each CORBA ORB
  - Object Oriented Domain Type Enforcement (OO-DTE) (TIS Labs at NAI)
- Dependability property manager
  - AQuA dependabilitysystem (DARPA/ITO Quorum), uses Proteus replication manager (UIUC), Maestro/Ensemble group communication (Cornell)
  - Provides replication, group communication, fault tolerance
  - Restarts replicas when they fail, votes to mask value faults, balances load across available hosts
  - Added a notification and control API to notify a system condition object when faults occur and provide parameters to Proteus to control replica placement
- All of these were integrated using QuO sysconds and contracts

Application-level intrusion awareness illustrated by the example system

- Application encodes its normal, expected behavior (in a contract); QuO middleware recognizes when application is operating outside normal ranges
- Recognizes alerts triggered by Tripwire and directory access checker
- System condition object records faults (value, replica, and host) reported by Proteus and looks for patterns that could represent attacks
  - failures on a particular host
  - failures of a particular replica or implementation; CORBA supports multiple implementations of an interface, in different languages and for different platforms
  - failures on a particular platform, e.g., OS, hardware, language, COTS component

Adaptation and control illustrated by the example system

- Access control provided by OO-DTE
  - IDS alerts trigger automatic changes in security policy
- QuO middleware provides parameters to dependability manager to control fault tolerance policy
  - When patterns of failure are detected, QuO requests that the dependability manager

    Avoid a host where repeated failures have occurred

    Kill replicas that have repeatedly provided incorrect values
  - When patterns of failure on a particular OS or in a particular language are recognized

    QuO reconfigures to replace objects with others implemented in different language, running on different platform, or using different COTS components
- Application code enters different operating mode

QuO technologies are equally applicable to overall intrusion detection and correction defenses

- Our infrastructure for adaptable control is applicable to both the "overall defense of a system" view and the "survivable application" view
- Same tools
  - Contracts and regions
  - System conditions, probes, and instrumentation
  - Encoding of adaptive behavior
- Different point of view
  - Encoding of policies – ID&R, security
  - Tradeoffs between systems and applications
- Including QuO technologies in both the overall defense system and in the applications facilitates interoperation

QuO integrated property managers and mechanisms for survivability

- Suspicious delay would trigger detailed QoS instrumentation and interaction with **intrusion detection systems** for identification, isolation and possible change in policy
- Adapt in response to host/replica (**dependability**) crashes & value faults
  - Use of Proteus Dependability Manager to obtain fault notification and to reconfigure replicas in case of intrusions
- In case of a certain kind of intrusion, **reserve bandwidth** to ensure survivability
  - Use of DIRM style instrumentation and management
- **Real-time** components adapt to intrusions and heightened security
  - As more cycles are used for security and ID, policies adapt to schedule the most critical operations
  - As recovery mechanisms kick in or more resources become available, scheduling policies adapt to utilize them

Adaptive software raises several interesting issues for IA and security

- Tradeoff between critical application survivability and system security
  - System security policy manager has ultimate authority over security issues
  - Critical applications that are adapting to survive must take priority over other applications
- What security holes will adaptable frameworks, applications, and systems introduce
  - Can adaptation be exploited?
  - Will adaptation, measurement, and control APIs be exploitable?
- If good guys can adapt, then so can bad guys
  - If our applications are adapting to make it more difficult to compromise them and to increase their chances for survival;
  - Can attackers adapt making it more difficult to detect them, defend against them, and recover from their attacks?

Conclusions

- We have developed middleware that supports adaptable software that can measure and control changing system properties
  - Supports software that has critical resource needs and can be deployed in unpredictable WAN environments
  - Instrumentation, triggers and effects can be defined and employed in an application in a structured manner as opposed to ad-hoc exception handling
  - Enables the combination and trade-off of multiple QoS dimensions, e.g., dependability and security
- This is useful for survivability and security
  - Supports applications that can reconfigure to avoid and recover from attacks
  - Supports application-level interfacing of IDSs, security managers, and other property managers
  - Enables applications to provide information, such as DOS information, useful for IDSs and security managers

Where to find more information

- QuO

  http://www.dist-systems.bbn.com/tech/QuO

- To get the QuO Toolkit v2.1 software, send e-mail to

  jloyall@bbn.com