



An Adaptive Communication Mechanism for Heterogeneous Distributed Environments Using XML and Servlets

Presenter: Vincent Cheung

Supervisors: Prof. M R Lyu & Prof. K W Ng

19 June 2001

Motivation

- ✱ Integration of distributed systems to serve the increasingly demanding users.
- ✱ Many obstacles hinder the expansion of distributed systems.
- ✱ Firstly, firewalls block many communication protocols.
- ✱ Secondly, objects in heterogeneous distributed environments cannot communicate.
- ✱ The existing solutions for these two problems have deficiencies.
- ✱ Our objective is to develop a simple and generic method to tackle these two problems but get rid of those deficiencies in existing solutions.

Presentation Outline

- ★ Overview of XML and Servlets
- ★ Firewall issue in distributed systems
 - Causes of the problem and existing solutions
 - Our mechanism using XML and Servlets
 - How our mechanism supports callbacks
- ★ Our XML schema for communications
- ★ Communication in heterogeneous environments
 - Causes of the problem and existing solutions
 - Our mechanism using XML and Servlets
- ★ A query system using our mechanism
- ★ Evaluation of our mechanism
- ★ Conclusion



Overview of Related Technologies

Overview of XML

- ✦ XML - eXtensible Markup Language.
- ✦ Proposed by WWW Consortium, in 1998.
- ✦ To define a complete, platform-independent and system-independent environment for the authoring and delivery of information resources across the web.
- ✦ **Element** is the basic component, i.e., a piece of text bounded by matching tags.
- ✦ **Attributes** are something associated with elements.

XML Example

- ★ use elements & attributes to describe information

```
<database>
```

```
  <news>
```

```
    <date year = "2001" month = "2" day = "12"/>
```

```
    <title> Mayor hails rule of law</title>
```

```
    <subtitle>Visiting official says HK's traditions can be model for  
              island but rejects 'one country, two systems</subtitle>
```

```
    <reporter>Kong Lai-fan</reporter>
```

```
    <content>
```

```
    The tradition of rule of law in Hong Kong could act as a model for Taiwan,  
    visiting Taipei Mayor Ma Ying-jeou said last night. But Mr Ma, who will  
    become the most prominent Taiwanese figure to meet Chief Executive Tung  
    Chee-hwa when they hold talks tomorrow, said the concept of "one country,  
    two systems" was not suitable for Taiwan no matter how successful it was  
    in Hong Kong. </content>
```

```
  </news>
```

```
  <news>
```

```
    . . .
```

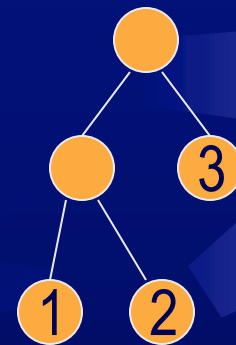
```
  </news>
```

```
</database>
```

XML – Flexible Structures

- ✦ XML is semistructure, which provide great flexibility to represent complex structures.
- ✦ Good tool to represent complicated data structures.
- ✦ For example, a nested tree to arbitrary depth

```
<node>  
  <node>  
    <node> 1 </node>  
    <node> 2 </node>  
  <node>  
    <node> 3 </node>  
</node>
```



Overview on Java Servlets

- ★ Servlets are the bodies of code that run inside request/response-oriented servers, and extend their functionality.
- ★ HTTP Servlets are typically capable of serving multiple clients concurrently and handling HTTP client requests.
- ★ Due to the growing popularity of Java, the trend today is to use servlets rather than CGI programs for new development.

Comparing CGI & Servlets

- ★ Java is platform independent, while CGI can be platform independent scripts (Perl scripts) or platform dependent (compiled C programs)
- ★ Servlet module would be loaded once when the first time it is invoked; stays loaded until the HTTP server task is shut down or restarted. CGI script is loaded every time it is invoked and unloaded when it has finished.
- ★ Many distributed environments, like CORBA, DCOM and Java RMI support Java. Hence, Servlet components can be naturally combined to those platforms.



Using XML and Servlets to Support CORBA Calls

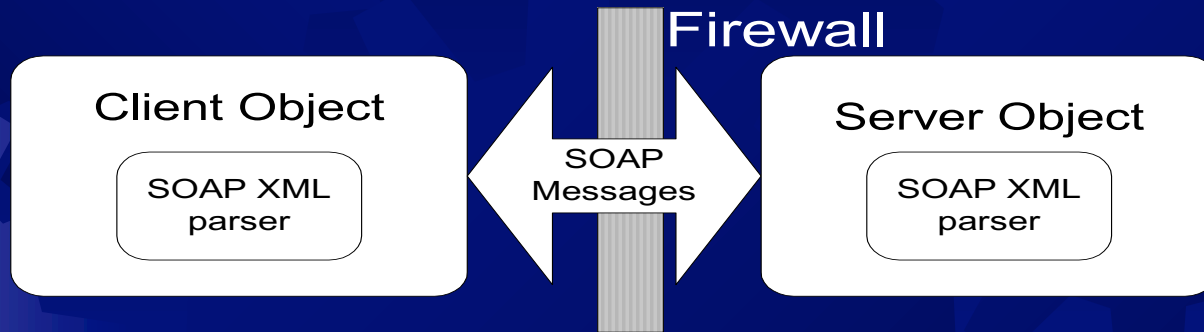
Firewalls – Obstacles in Distributed Systems

- ★ Firewalls use packet filtering in network layer to enforce certain security rules.
- ★ Firewalls can include elements that operate in application level, e.g. HTTP, FTP, Telnet.
- ★ But for those less common protocols, firewalls may not support, such as IIOP in CORBA.
- ★ In application level, message body of IIOP is encoded in Common Data Representation, so firewalls cannot decode it.

Existing Solutions to Firewall Problem

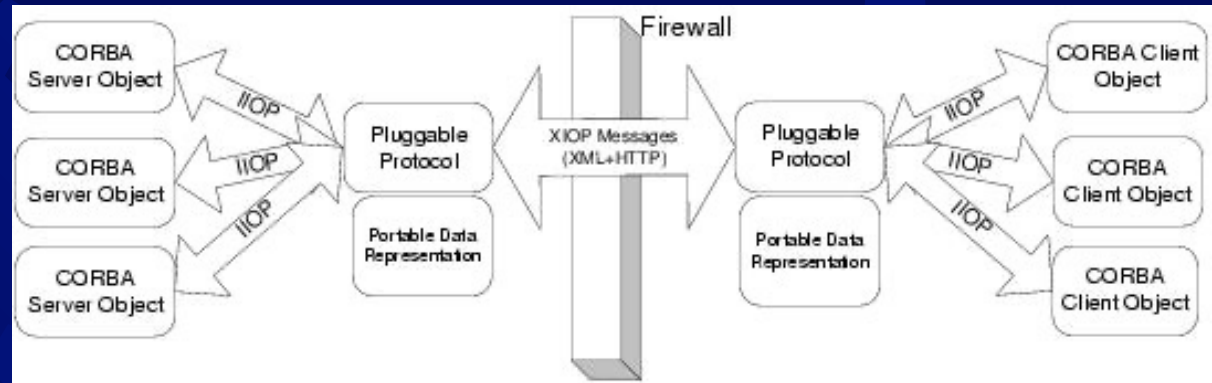
- ✦ Some firewalls are dedicated for CORBA IIOP, such as IONA Orbix Wonderwall and Visibroker Gatekeeper.
- ✦ They are vendor-dependent and proprietary, and cannot handle callbacks.
- ✦ Another approach is using HTTP for tunneling across firewalls. Using XML messages to describe the communication protocols and transmit them with HTTP.
- ✦ SOAP, XML-PRC and XIOP are using this approach.

SOAP & XML-RPC



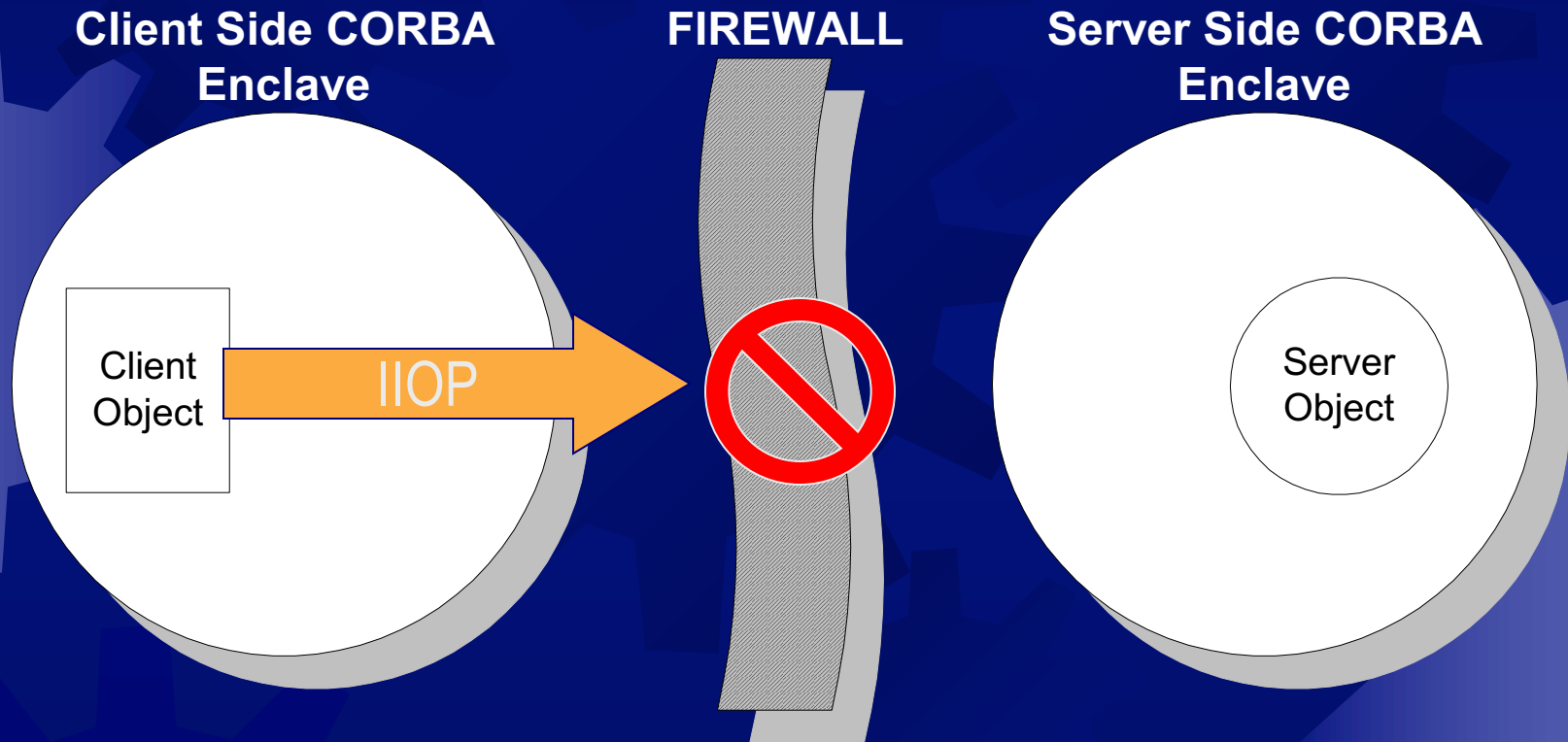
- ★ Simple Object Access Protocol (SOAP) is accepted by W3C as a standard in 1999.
- ★ Using XML based messages as communication protocols, hence having flexible semistructure to represent data.
- ★ Not designed for supporting existing protocols.
- ★ Not designed for bi-directional calls.
- ★ XML-RPC is a smaller subset of SOAP.

XIOP



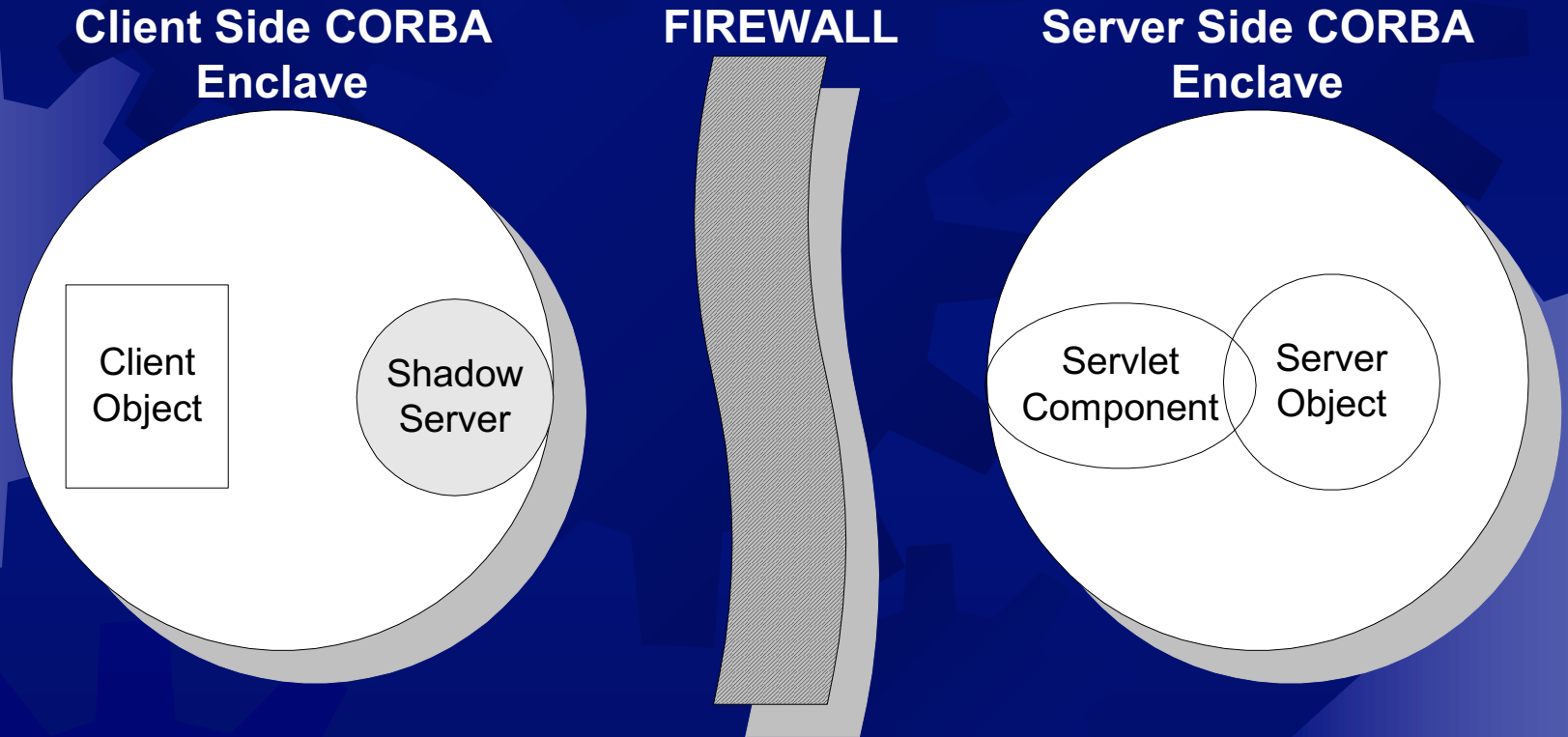
- ✦ XIOP is first introduced at April 2000.
- ✦ A substitute of CORBA IIOp in XML format. It is not an international standard.
- ✦ Fully compatible with CORBA systems.
- ✦ Requires pluggable protocol framework for conversion of IIOp to XML.
- ✦ Requires modifications to existing CORBA components.
- ✦ Not designed for bi-directional calls.
- ✦ XIOP schema contains low-level contents, which increase the complexity of the XML messages.

Our Proposed Mechanism



- ✦ Directly IIOP communication is not allowed.
- ✦ We put some add-on components in our systems for communications

Our Proposed Mechanism



- ✦ A Shadow Server in the client side.
- ✦ A Servlet Component in the server side.

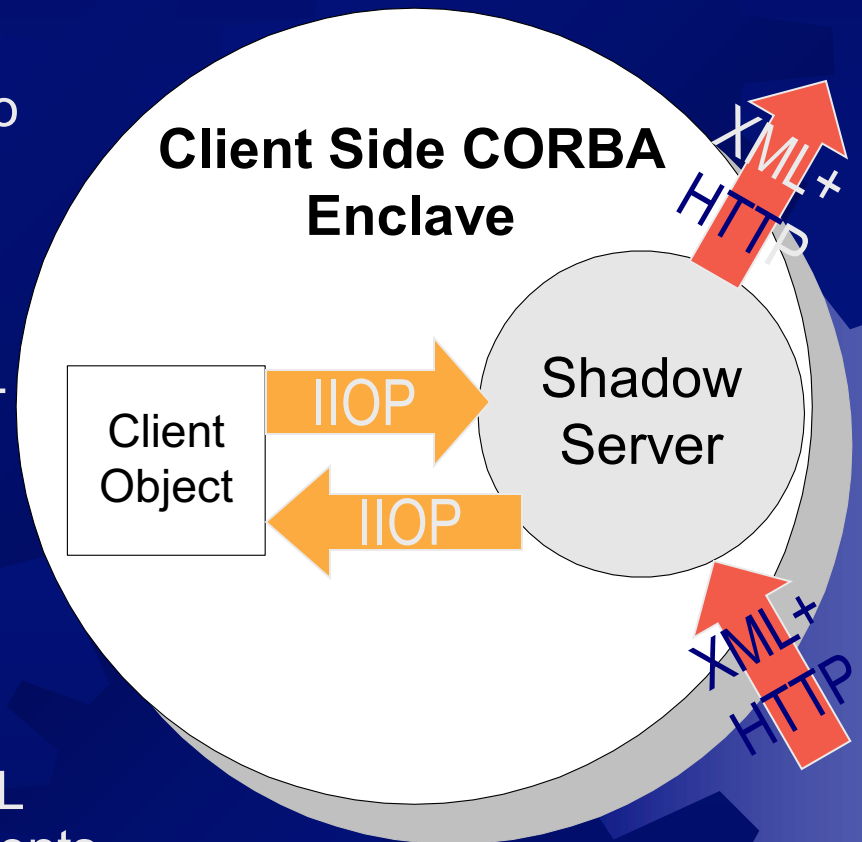
In Client Side

REQUEST.

1. Client object sends a request to Shadow Server
2. Shadow Server converts the IIOp request to XML format
3. Shadow Server sends the XML message to server side by HTTP

RESPONSE

1. Shadow Server receives XML-based response message
2. Shadow Server parses the XML message and extracts the contents
3. Shadow Server sends an IIOp response calls to client object



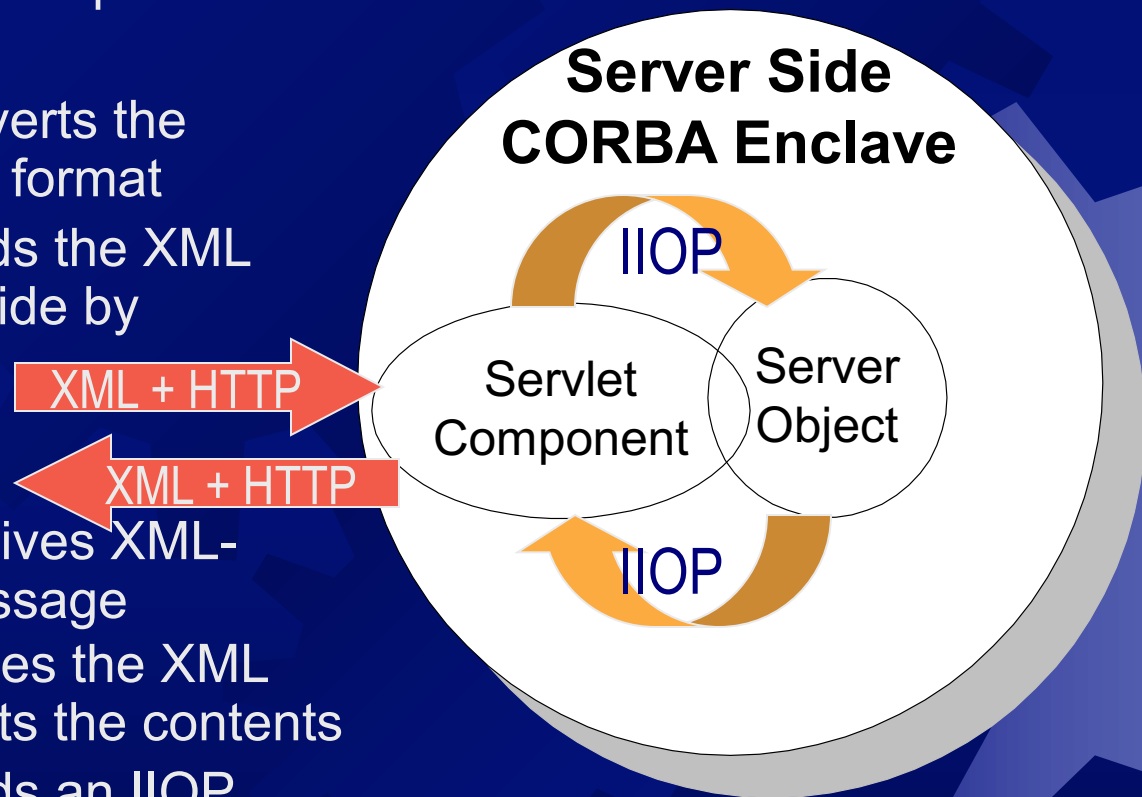
In Server Side

REQUEST.

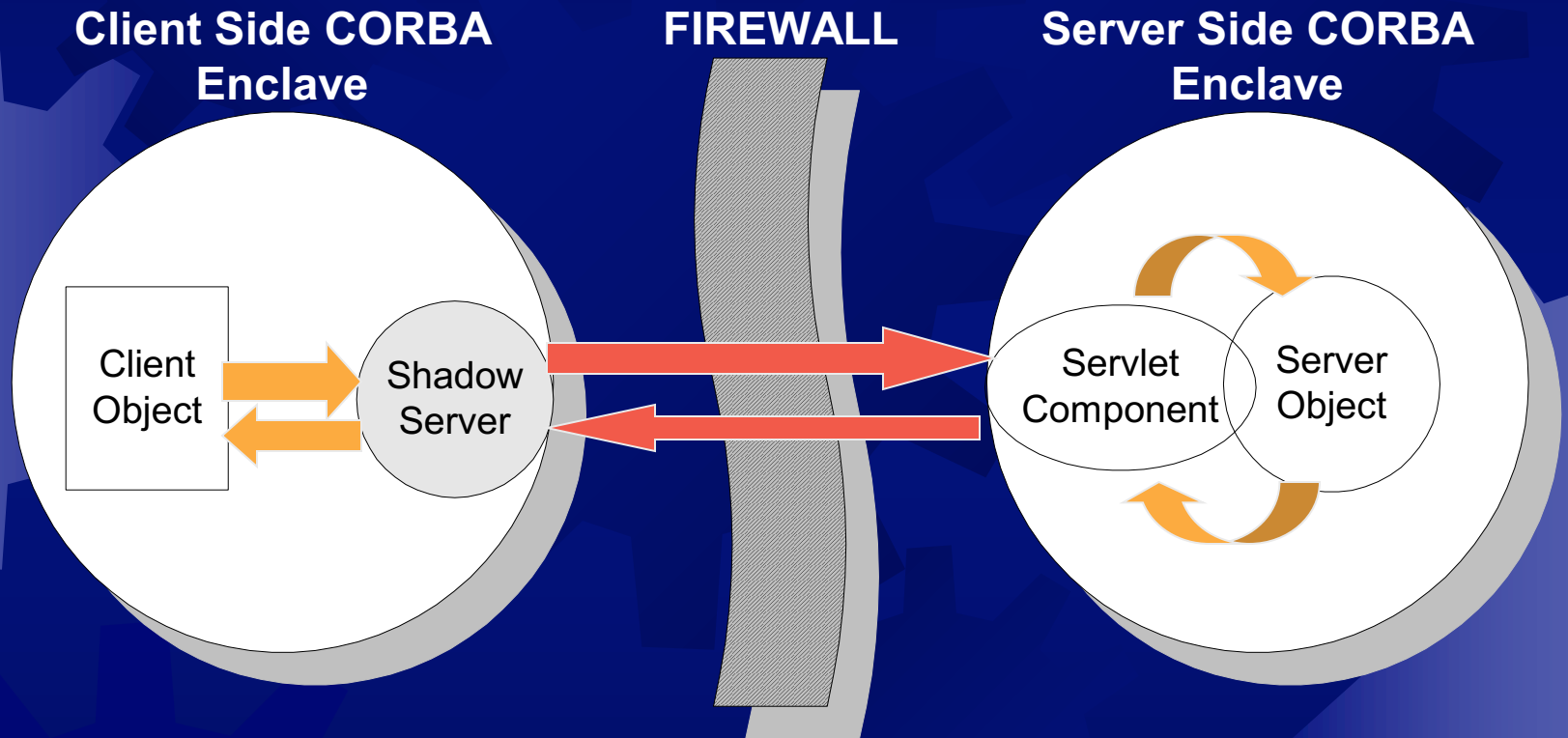
1. Client object sends a request to Shadow Server
2. Shadow Server converts the IIOp request to XML format
3. Shadow Server sends the XML message to server side by HTTP

RESPONSE

1. Shadow Server receives XML-based response message
2. Shadow Server parses the XML message and extracts the contents
3. Shadow Server sends an IIOp response calls to client object



Overall Procedures



- ✦ Client objects regards shadow object as the actual target server object.
- ✦ Shadow Server and Servlet Component are ordinary local CORBA objects.

Contents of XML messages

★ A sample method call:

```
<request>
  <Account type="interface">
    <deposit type="operation">
      <parameter ref="in" order="1">
        <float name="amount">23000.45</float>
      </parameter>
    </deposit>
  </Account>
</request>
```

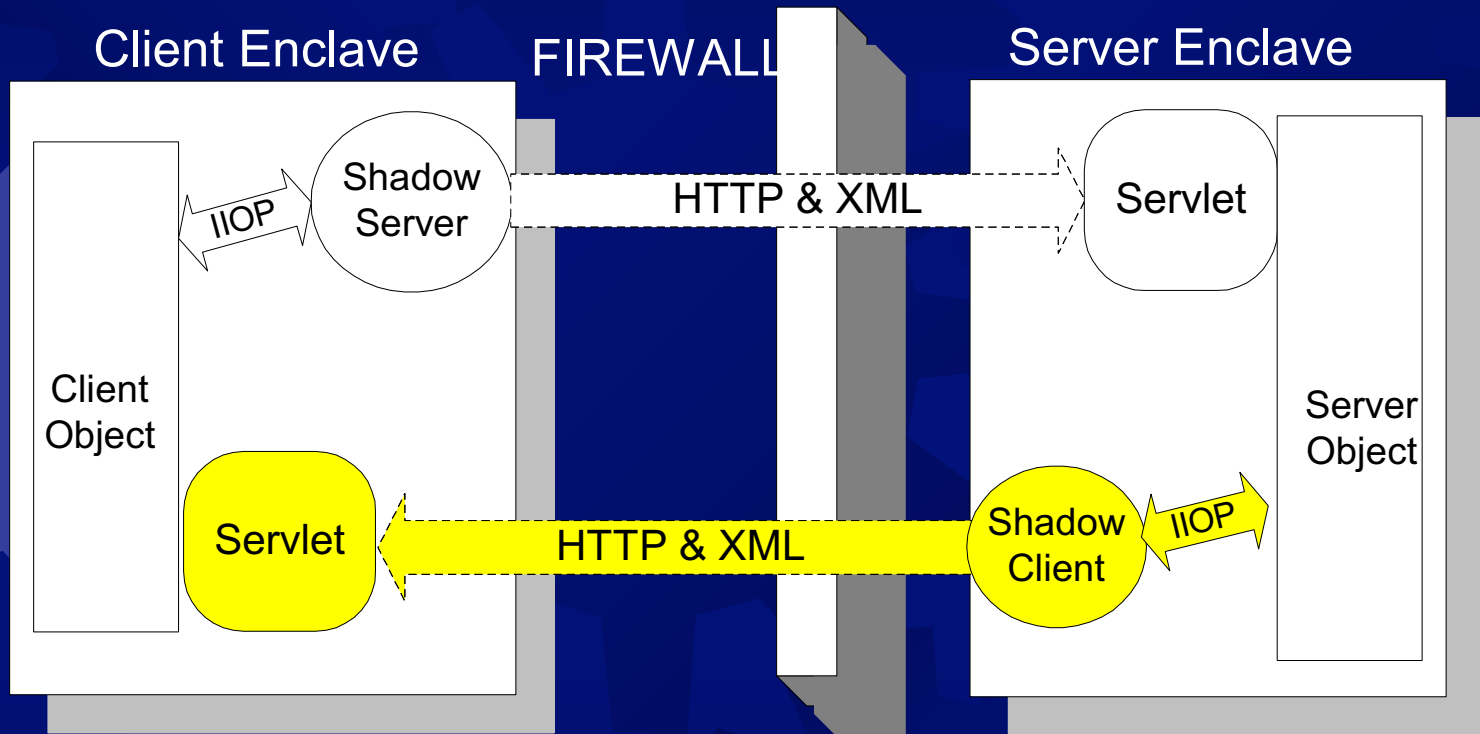
★ HTTP call:

<http://pc90003.cse.cuhk.edu.hk:8000/research/testing?%3Crequest+type%3D%22interface%22%3E+%3Caccount+type%3D%22interface%22%3E+%3Cdeposit+type%3D%22operation%22%3E+%3Cparameter+ref%3D%22in%22+order%3D%221%22%3E+%3Cfloat+name%3D%22amount%22%3E23000.45%3C%2Ffloat%3E+%3C%2Fparameter%3E+%3C%2Fdeposit%3E+%3C%2Faccount%3E+%3C%2Frequest%3E>

About Callbacks

- ☀ Client objects need to react to changes or updates that occur on the server side, callback feature is needed.
- ☀ A client object passes itself as parameter to the server object, hence the server object has the object reference of client and is able to invoke client's methods.
- ☀ A bi-directional call. Both side need to invoke a method call in another side.
- ☀ Enhance our mechanism to support callbacks

Overall Calling Procedures

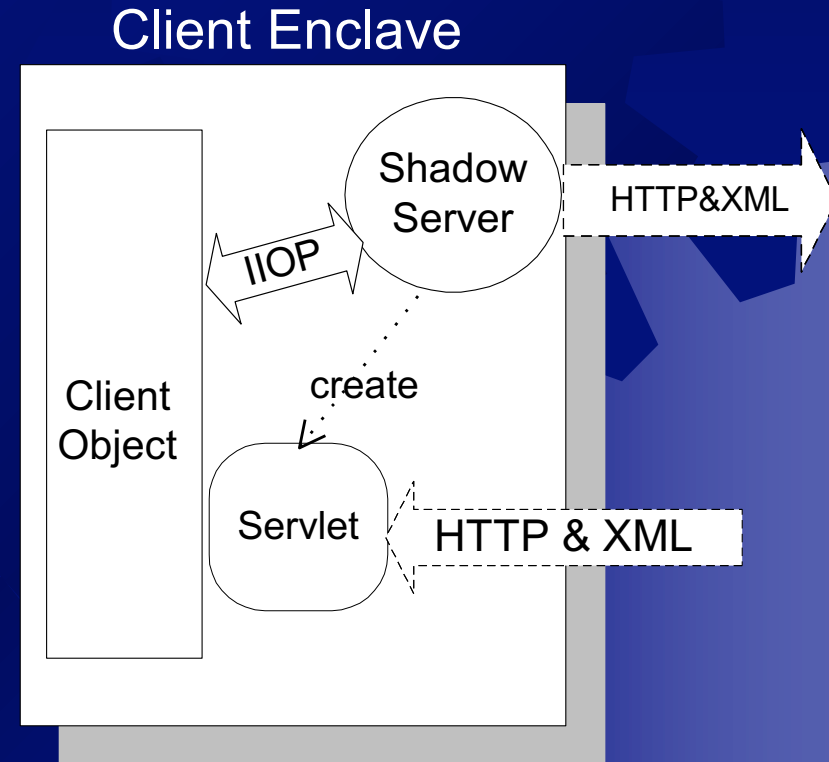


- ★ One pair of add-on components cannot support callback in a generic way, hence, we use one more pair of add-on components.

In Client Side (Callbacks)

Procedure

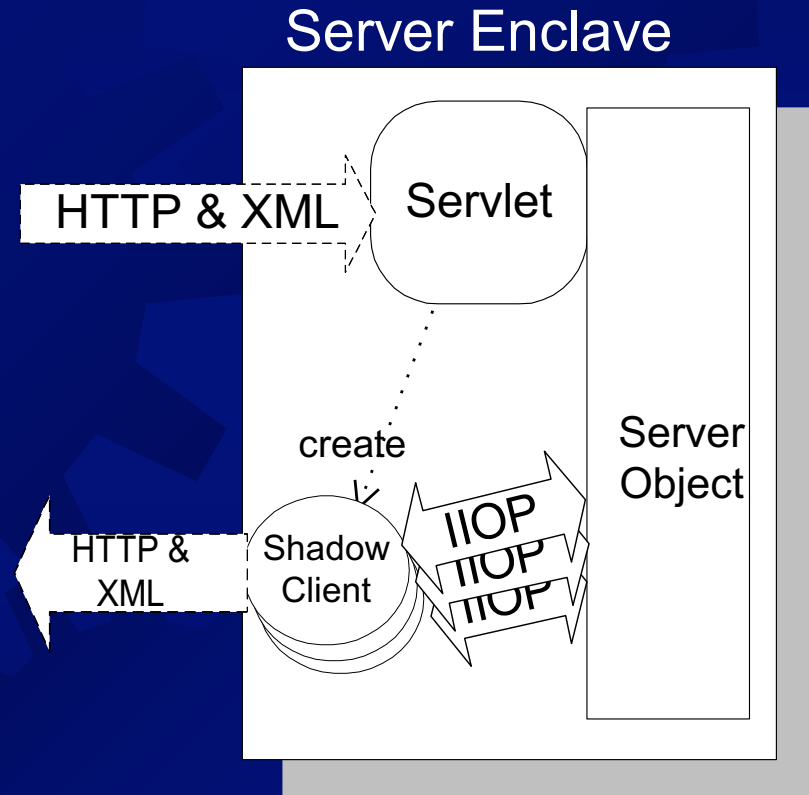
1. Client object first sends a method call to the Shadow Server.
2. Once the Shadow Server checks out there is callbacks, it will create a new Servlet component associated with the client object
3. Shadow Server sends the method request as normal to the server side, with some additional info of the new Servlet component.
4. The new Servlet component waits for callback from server side.



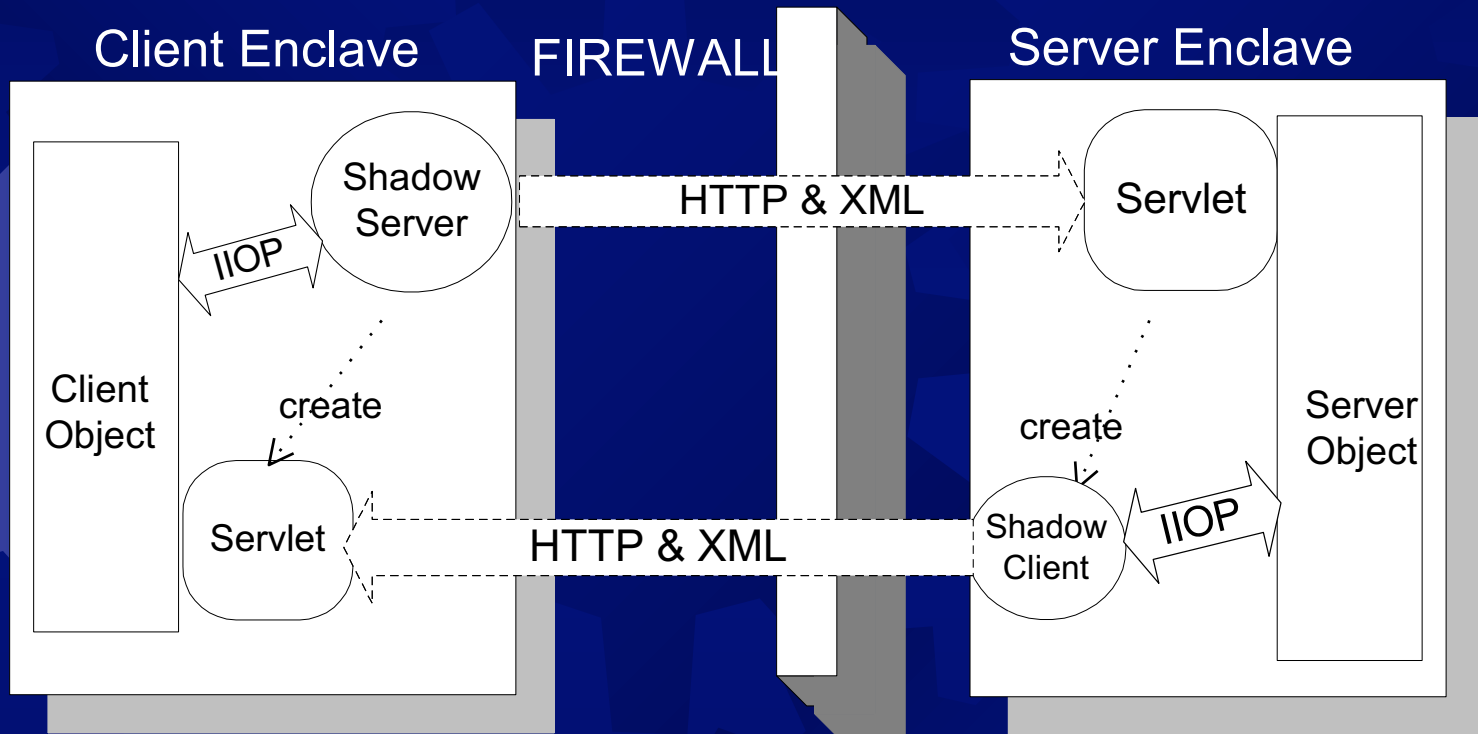
In Server Side (Callbacks)

Procedure

1. Servlet Component in the Server side receives call from outside. If there is callback, this Servlet Component will create a Shadow Client immediately.
2. Servlet Component will inform the server object the location of the Shadow Client that has required callback.
3. When there is a need to callback, server object will call the Shadow Client(s).
4. The Shadow Client(s) will invoke the Servlet Component(s) on the client side.



Overall Calling Procedures



Importance of Object Interfaces

- ★ Add-on components know the interfaces of the associated objects, hence they are perform the same as the target object.
- ★ The interfaces are very important for the creation of the add-on components.
- ★ The interfaces are also important for XML messages passing, as object structures, methods and attributes, parameters required, etc, are all included in the IDL files.
- ★ CORBA IDL defines the interfaces of objects, hence we make use of IDL to generate XML contents for those objects.



A Translator to Convert CORBA IDL to XML

A Translator for IDL to XML

- ✱ All CORBA objects have IDL to describe their interfaces.
- ✱ We implemented a translator in Perl for the conversion from IDL to XML
- ✱ The generated XML schema can help in
 - ✱ giving a standard for XML transmission messages
 - ✱ creating the add-on components.
- ✱ We need to have unique mapping from IDL to XML to avoid ambiguity in communications.
- ✱ The conversion rules will be discussed now!

IDL Basic Types Conversion

short	<short></short>
unsigned short	<ushort></ushort>
long	<long></long>
unsigned long	<ulong></ulong>
long long	<longlong></longlong>
unsigned long long	<ulonglong></ulonglong>
float	<float></float>
double	<double></double>
char	<char></char>
boolean	<boolean></boolean>

Complex Types

- ✦ Complex type means data type with complex structures, such as struct, enum, arrays, sequence, etc.
- ✦ We regard “*interface*” as complex type as well, as it presents the details of an object, which is also a kind of structure.

General Rules for Complex Type Conversion

- ★ The data type names are used as element tag names.
- ★ For complex type, they would have an attribute “*complex*” for indicating the kind of complex type.
- ★ For both basic and complex type, if they have a variable name, there will be a “*name*” attribute in the element tag.
- ★ For array and sequence, names of element tags are “*array*” & “*sequence*” respectively, without “*complex*” attribute.
- ★ Array and sequence have attributes “*size*” and “*index*” to indicate their sizes and each of their elements. Bounded in the tags are the member elements.

An Example of Complex Type

```
<sequence size="6" name="db">
  <Customer complex="struct" index="1">
    <long name="id">111</long>
    <short name="age">24</short>
    <boolean name="ismale">TRUE</boolean>
  </Customer>
  <Customer complex="struct" index="2">
    <long name="id">112</long>
    <short name="age">39</short>
    <boolean name="ismale">FALSE</boolean>
  </Customer>
  <Customer complex="struct" index="3">
    <long name="id">113</long>
    <short name="age">23</short>
    <boolean name="ismale">TRUE</boolean>
  </Customer>
</sequence>
```

```
struct Customer {
  long id;
  short age;
  boolean ismale;
}
sequence <Customer> db;
```


Description for Interfaces

- ★ We have attributes, operations, and exceptions inside an interface.
- ★ The types of attributes would be the element tag names and they have “*type*” as attribute of the tag.
Attribute float balance;
`<float type=“attribute” name=“balance”/>`
- ★ The operation names and exception names are used as their element tag names.
- ★ In operation definition, we have “*parameter*”, “*return*” and “*raises*” tags for indicating parameters passing, return values, and exceptions that may be raised.


Sample XML Message

```
<Account complex="interface">
  <notEnoughMoney type="exception">
    <string name="reason"/>
  </notEnoughMoney>
  <withdraw type="operation">
    <return> <boolean/> </return>
    <parameter ref="in" order="1">
      <float name="amount"/>
    </parameter>
    <raises> <notEnoughMoney/></raises>
  </withdraw>
</Account>
```

```
interface Account {
  exception notEnoughMoney {
    string reason;
  };
  boolean withdraw(in CashAmount amount)
    raises(notEnoughMoney);
}
```

Use of the XML Schema

- ★ The generated XML schema can help in
 - ★ creating the add-on components.
 - ★ giving a standard for XML transmission messages.
- ★ CORBA IDL is generic for mapping many different kinds of programming languages. Because of its generic property, we can use its XML schema to map to other Interface Definition Languages in other distributed environments.



Communication in Heterogeneous Distributed Environments

Across Heterogeneous Distributed Environments

- ✱ Different Distributed Environments have different communication protocols.
- ✱ CORBA systems communicate with IIOP
- ✱ DCOM systems use DCOM Protocol
- ✱ Java RMI systems talk with JRMP
- ✱ ...
- ✱ It is a problem to ask a CORBA object to communicate with a DCOM object, or ask a Java RMI object to communicate with a DCOM object, as they have NO “common language”!

Existing Solutions for Bridging across DCOM and CORBA

- ★ OMG has a specification of COM/CORBA mapping, which maps CORBA IDL to DCOM MIDL definitions.
- ★ Based on this spec., there are a number of products for bridging DCOM and CORBA. They do message conversion in binary level.
- ★ OrbixCOMet is one of the best products in the market. Middleware COMET is used between CORBA and DCOM enclaves and convert binary messages.
- ★ Yet, it is not generic. It cannot bridge between other environment, such as Java RMI.

Existing Solution for Java RMI & CORBA Communication

- ★ Similar to DCOM, OMG also provides specifications for mapping Java Interface Definition to CORBA IDL, and vice versa.
- ★ Based on the spec., Sun Microsystems provides a RMI/IIOP package for converting RMI objects to use IIOP.
- ★ Hence, converted RMI objects can communicate with CORBA objects, without bridging overheads.
- ★ But those converted RMI objects cannot communicate with original RMI objects again, as they are using different protocols.

Our Approach

- ★ We rely on the XML schema that we have described to be the “common language”
- ★ We rely on OMG mapping specifications to be the standards, and map DCOM or Java RMI to the same XML schema.
- ★ We rely on the mechanism we have introduced to provide homogeneous objects for calling. The add-on components are developed under the same environment as the callers or the callees.

Case Study for DCOM

- ★ DCOM is developed by Microsoft and mainly used in Windows environment.
- ★ Similar to CORBA, DCOM also has its interface definition language, named MIDL.
- ★ The contents in square brackets can be ignored in mapping our XML schema as they are not related to interface definition

A Sample MIDL File

```
[ uuid(7071a210-2c51-11d0-b131-004155854000), version(1.0) ]
```

```
library SimpleStocks
```

```
{
```

```
  importlib("stdole32.tlb");
```

```
[ uuid(BC4C0AB0-5A45-11d2-90C5-00A0244C655), dual ]
```

```
interface IStockMarket : IDispatch
```

```
{
```

```
  HRESULT get_price([in] BSTR p1, [out,retval] float* rtn);
```

```
}
```

```
[ uuid(BC4C0AB0-5A45-11d2-90C5-00A0244C655) ]
```

```
coclass StockMarket
```

```
{
```

```
  interface IStockMarket;
```

```
};
```

```
};
```

Mapping between DCOM and CORBA

DCOM MIDL	CORBA IDL	XML Schema
short	short	<short></short>
unsigned short	unsigned short	<ushort></ushort>
long	long	<long></long>
unsigned long	unsigned long	<ulong></ulong>
float	float	<float></float>
double	double	<double></double>
char	char	<char></char>
bool	boolean	<boolean></boolean>

- ★ If DCOM object receives tags <longlong> and <ulonglong>, they would be regarded as long and unsigned long respectively.

Special Issues in Mapping DCOM

- ✦ In MIDL, all methods has return type HRESULT, which represents error and success notifications. For actual return value, it would be the last parameter with [out, retval].
- ✦ DCOM does not have attributes in the interface definition files. For each attribute, DCOM uses two methods, `_get_<NAME>` and `_put_<NAME>` to represent.
- ✦ Application exceptions are represented as parameters of the corresponding methods.
- ✦ DCOM does not support multiple inheritance, but support multiple extension. We simply list all the resultant methods, attributes and exceptions in XML schema to handle inheritance, hence it would not be a problem.

Case Study for Java RMI

- ✦ Java RMI is developed by Sun Microsystems, and it is platform independent.
- ✦ Interface of Java RMI objects are defined in ordinary .java files with keyword “interface”.
- ✦ Each interface is defined in a separated file.

A Sample Java RMI Interface Definition File

```
public interface BankAccount extends java.rmi.Remote {  
    public void deposit (float amount)  
        throws java.rmi.RemoteException;  
    public void withdraw (float amount)  
        throws OverdrawnException, java.rmi.RemoteException;  
    public float balance()  
        throws java.rmi.RemoteException;  
}
```

Mapping between JavaRMI and CORBA

Java RMI	CORBA IDL	XML Schema
short	short	<short></short>
int	long	<long></long>
long	long long	<longlong></longlong>
float	float	<float></float>
double	double	<double></double>
char	char	<char></char>
boolean	boolean	<boolean></boolean>

- ✦ Java RMI objects do not have unsigned types. When they receive tags of unsigned types, they would be regarded as signed types.

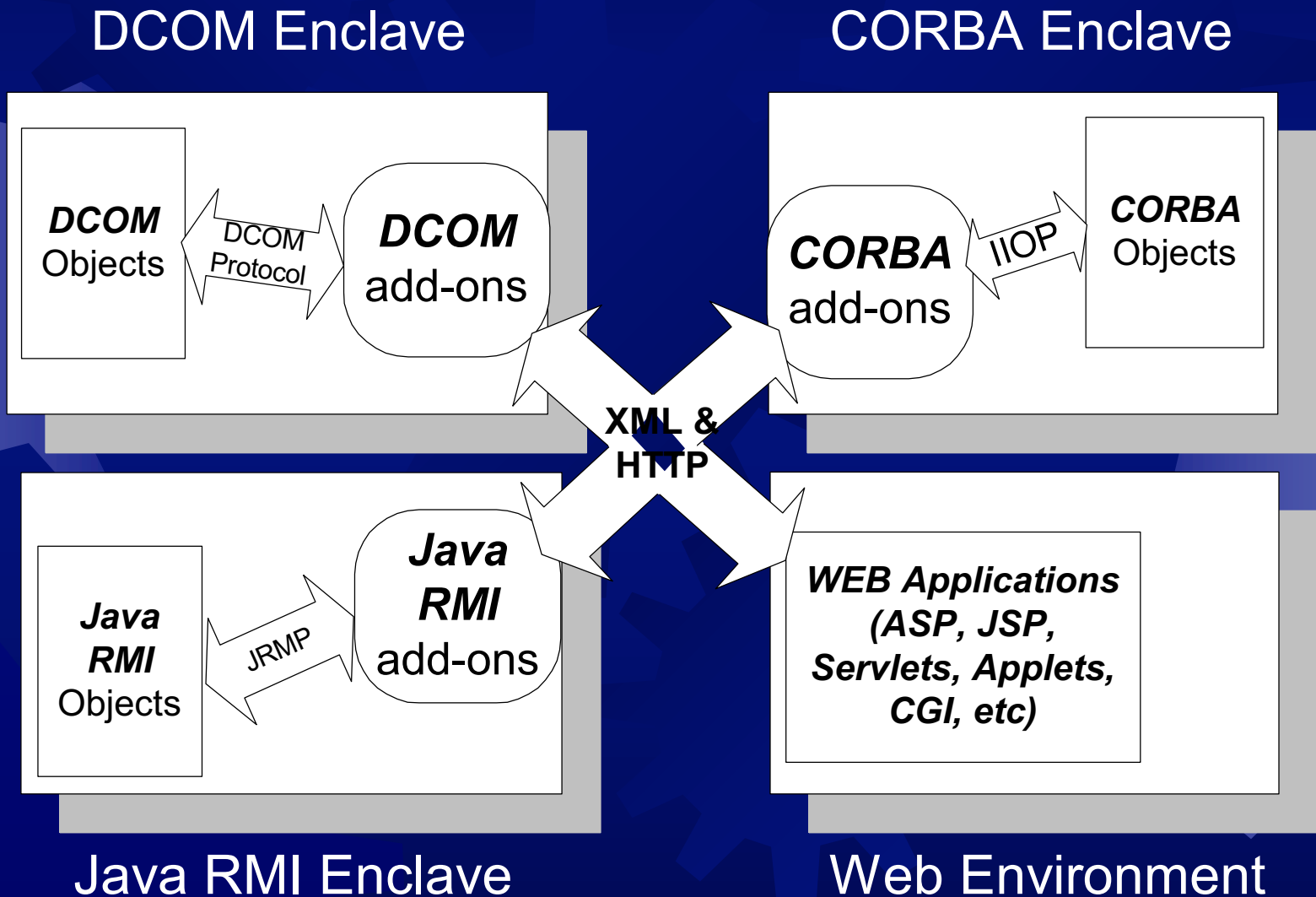
Special Issues in Mapping Java RMI


- ✦ Java RMI does not have attributes in the interface definition files. For each attribute, Java RMI uses two methods, `get<NAME>` and `set<NAME>` to represent.
- ✦ All Java RMI object methods have a system exception “RemoteException”, which this will not be included in XML schema.
- ✦ Java RMI does not have struct or enum complex type. Class definitions, without any object methods, will be used instead.

Linking to the Web

- ✦ XML and Servlets also provides a good interface to connect the objects in a system to the web.
- ✦ Based on XML schema, we can develop web applications which can communicate with objects inside DCOM, CORBA, etc. via Servlets components.

Communicating in Heterogeneous Systems

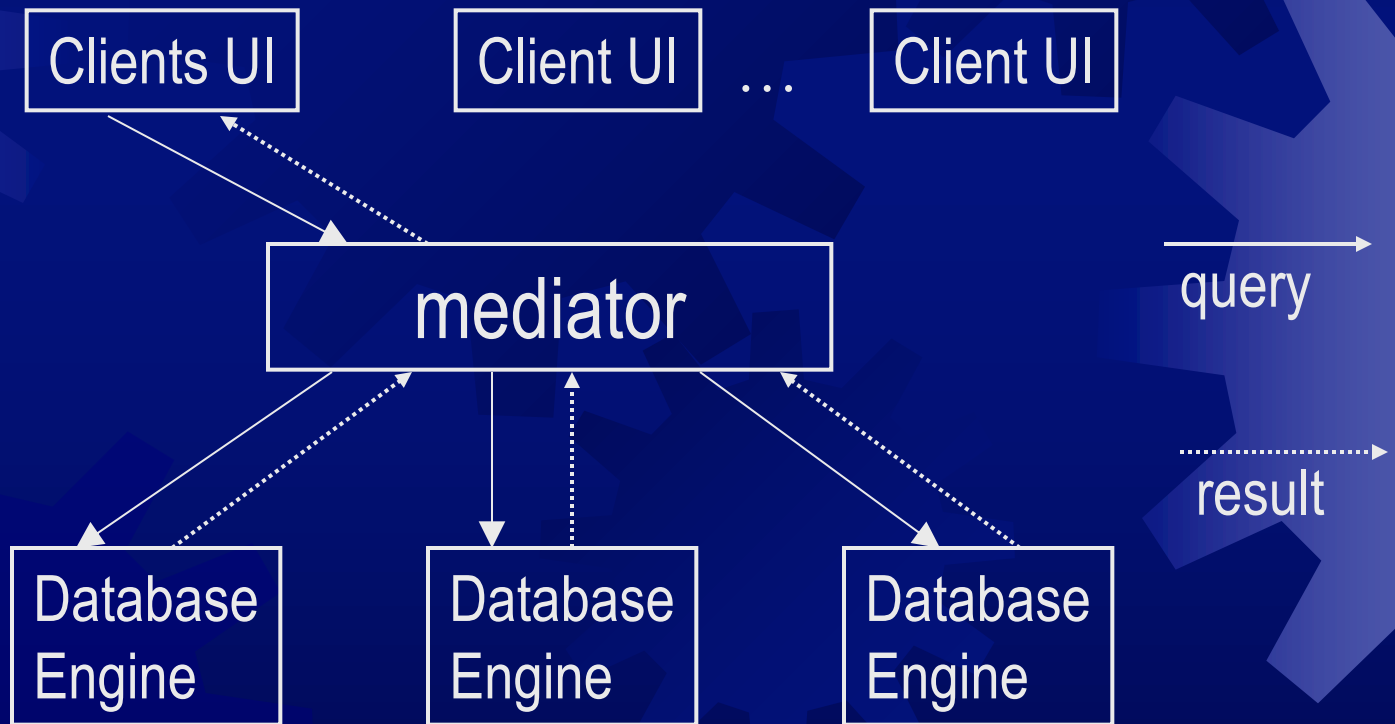




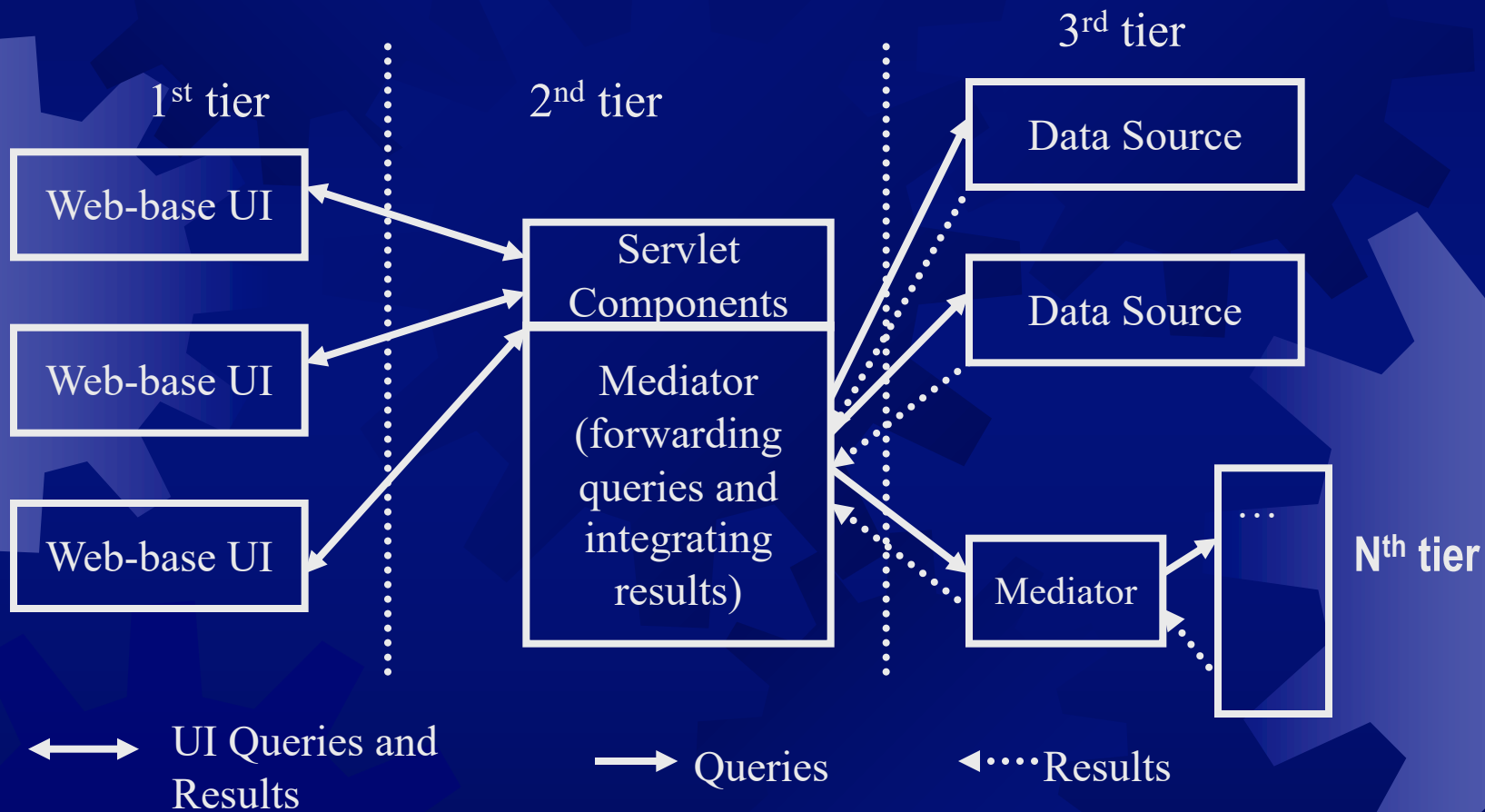
Building a Scalable Mediator-based Query System

What is mediator?

- ★ A middle layer for forwarding clients queries to appropriate sources, and integrate the data before returning to users

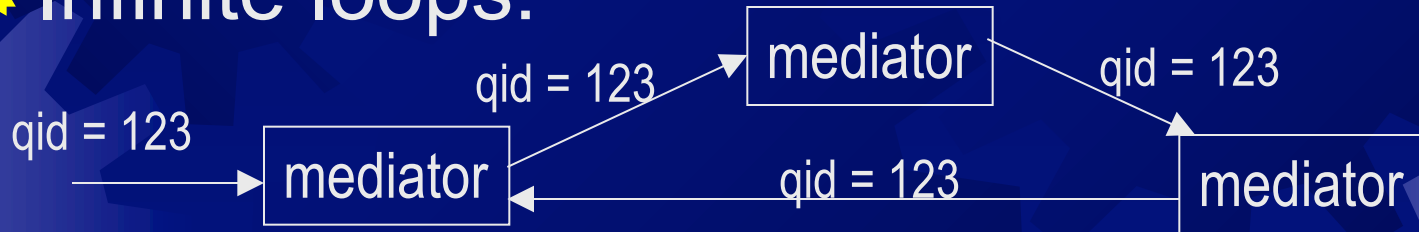


Architecture of our system



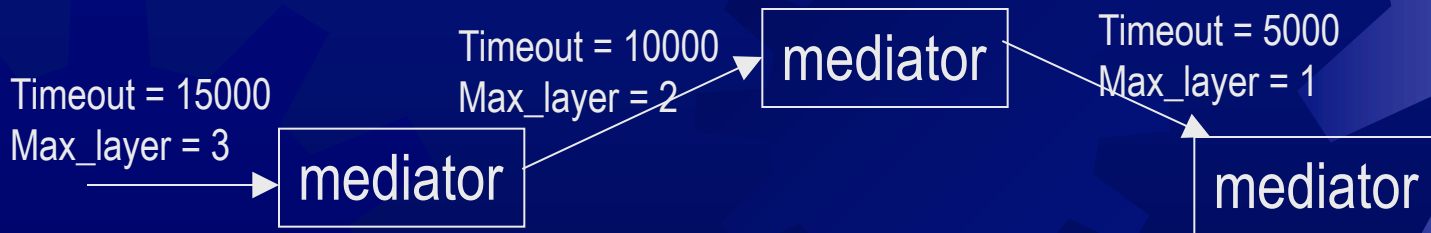
Handling some special cases

- ☀ Infinite loops:



- ☀ Broken connection

- ☀ Too many layers of traversal



IDL design of our system

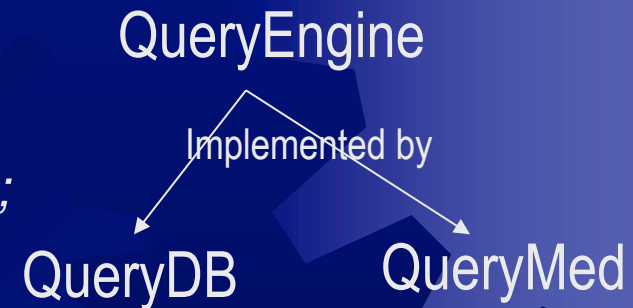
- ✦ IDL (Interface Definition Language) defines export interface of CORBA objects
- ✦ Our IDL design:
 - The parameter type for special cases handling

```
Struct SysPara  
{  
    long qid;  
    long timeout;  
    short maxlayer;  
}
```

IDL design of our system

- ✦ Mediator may make queries to Databases or Mediators. To be generic, we want Database objects and Mediator objects can have the same interface for calling.
- ✦ Both of them implement *QueryEngine* Interface

```
interface QueryEngine {  
    string query(in SysPara para,  
                in string querystmt);  
}
```



QueryDB Object

- ✦ Directly connects to the data source
- ✦ Caller calls query()
- ✦ It takes the query statement parameter and make query to data source
- ✦ Returns answer in XML string stream

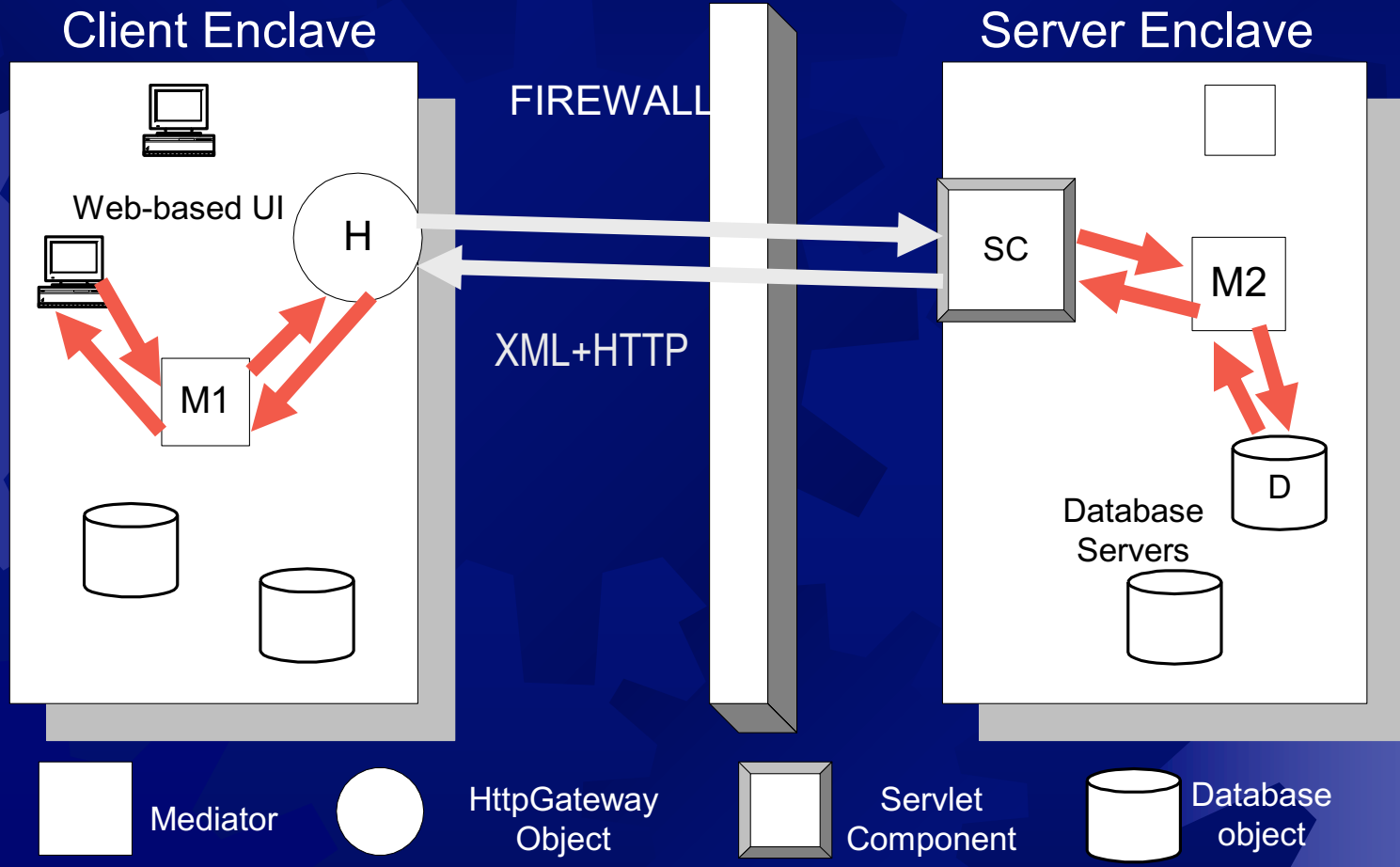
QueryMed Object

- ★ Same invoking method, `query()`
- ★ Besides *QueryEngine*, it implements another interface, *QueryMediator*

```
public interface QueryMediator {  
    public QueryEngine[] qelist();  
    public void qelist(QueryEngine[] arg);  
    public void append_result(String res);  
}
```

- ★ *qelist* holds a list of *QueryEngine* objects, i.e. *QueryMed* or *QueryDB* objects, which will be called by that mediator.
- ★ It starts a thread for each target *QueryEngine* object, and the thread will call *append_result()* to integrate results from various sources

Across Firewalls



Sample Request Message

```
<request>
  <QueryEngine type="interface">
    <query type="operation">
      <parameter ref="in" order="1">
        <SysPara complex="struct">
          <long name="qid">3984982418240339</long>
          <long name="timeout">2000</long>
          <short name="maxlayer">3</short>
        </SysPara>
      </parameter>
      <parameter ref="in" order="2">
        <string name="QueryString">
          where <news>$B</news> in "database.xml"
          <keyword>separatist</keyword> in $B
          construct <result> $B </result>
        </string>
      </parameter>
    </query>
  </QueryEngine>
</request>
```

Sample Response Message

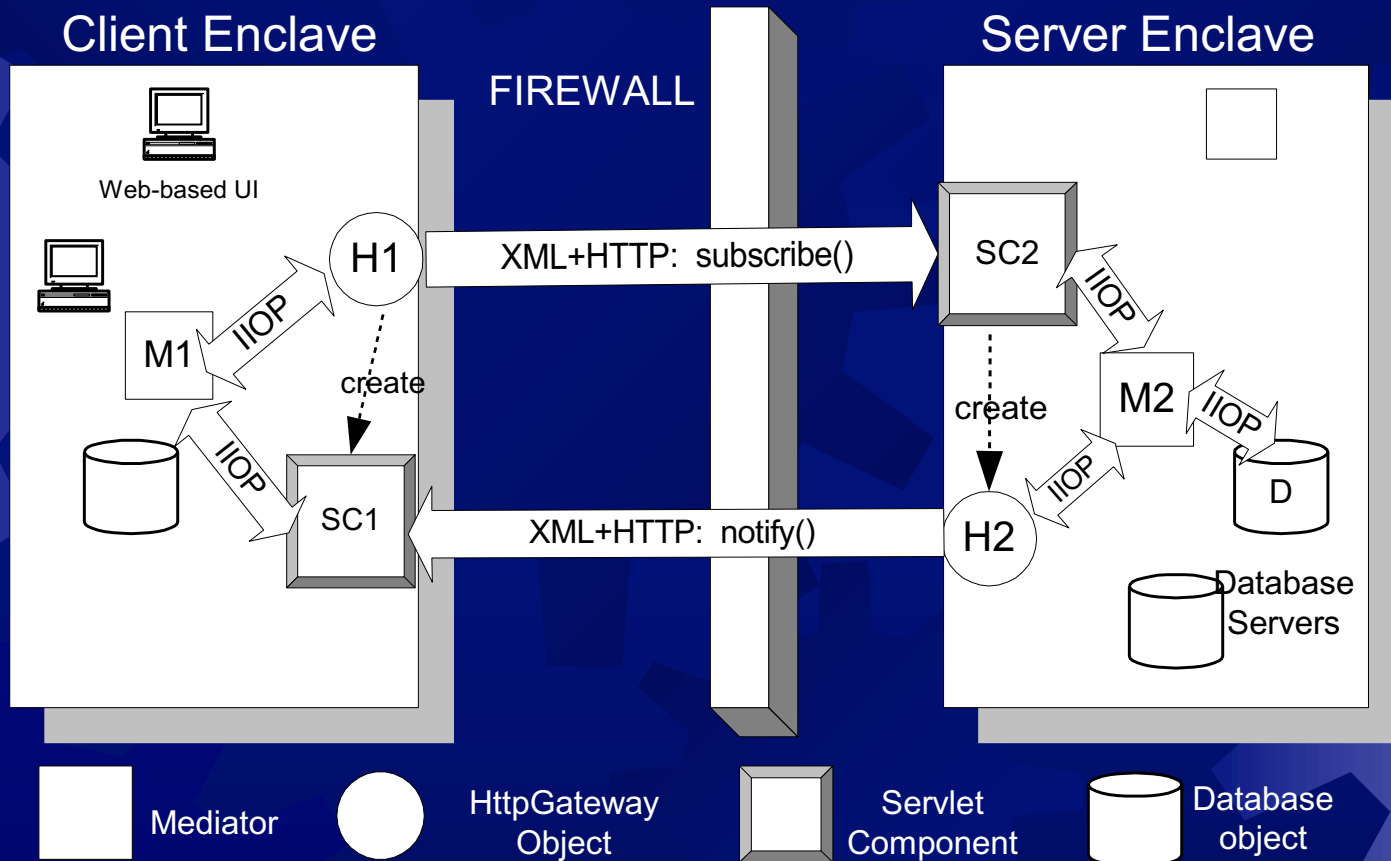
```
<response>
  <QueryEngine type="interface">
    <query type="operation">
      <return>
        <string>
          <news> <source>South China Morning Post </source> <date>
            <day>15</day><month>4</month> <year>2000</year> </date>
            <title>Press warning appropriate, says Beijing</title><content>
            Beijing yesterday defended remarks made by senior SAR-based
            official Wang Fengchao that local media should avoid reporting
            separatist views.</content> </news>
          </string>
        </return>
      </query>
    </QueryEngine>
  </response>
```

Callback Supports

- ✦ To enhance the system features, we add a subscription service in the system which requires callback.
- ✦ A Mediator can subscribe a certain topic of news and it will be notify whenever there is a update.
- ✦ The new IDL is as follow:

```
interface QueryEngine {  
    string query(in SysPara para, in string querystmt);  
    void subscribe(in QueryEngine qe, in string topic);  
    void notify(in string newContent);  
}
```

Callback Supports



Across Heterogeneous Environments

- ✦ Expanding the system across the firewalls is not enough, we want to expand our system across heterogeneous distributed environments.
- ✦ To provide generic querying to objects in across heterogeneous systems, we design the objects of DCOM and Java RMI having similar interfaces as those CORBA objects.
- ✦ For simplicity, we don't include the callback features in the DCOM and Java RMI systems.

Query System in DCOM

- ★ The MIDL design of the DCOM object, it will have the same XML transmission message schema as the CORBA system.

```
import "oidl.idl";
import "ocidl.idl";
typedef struct SysPara
{
    long qid;
    long timeout;
    short maxlayer;
}SysPara;
[ uuid(AC6EDE04-ADF2-4324-BB8C-B350295BFD5E) ]
interface ICOMQueryEngine : IDispatch
{
    HRESULT query([in] SysPara para,
                  [in] char * queryStmt
                  [out, retval] char ** rtn);
};
```

Query System in Java RMI

- ★ The Java Interface Definition Java RMI object, it will have the same XML transmission message schema as the CORBA system.

```
// SysPara.java
```

```
public class SysPara implements
    java.io.Serializable{
    public long qid;
    public long timeout;
    public short maxlayer;

    public SysPara() {
        qid=-1;
        timeout=-1;
        maxlayer=-1;
    }
}
```

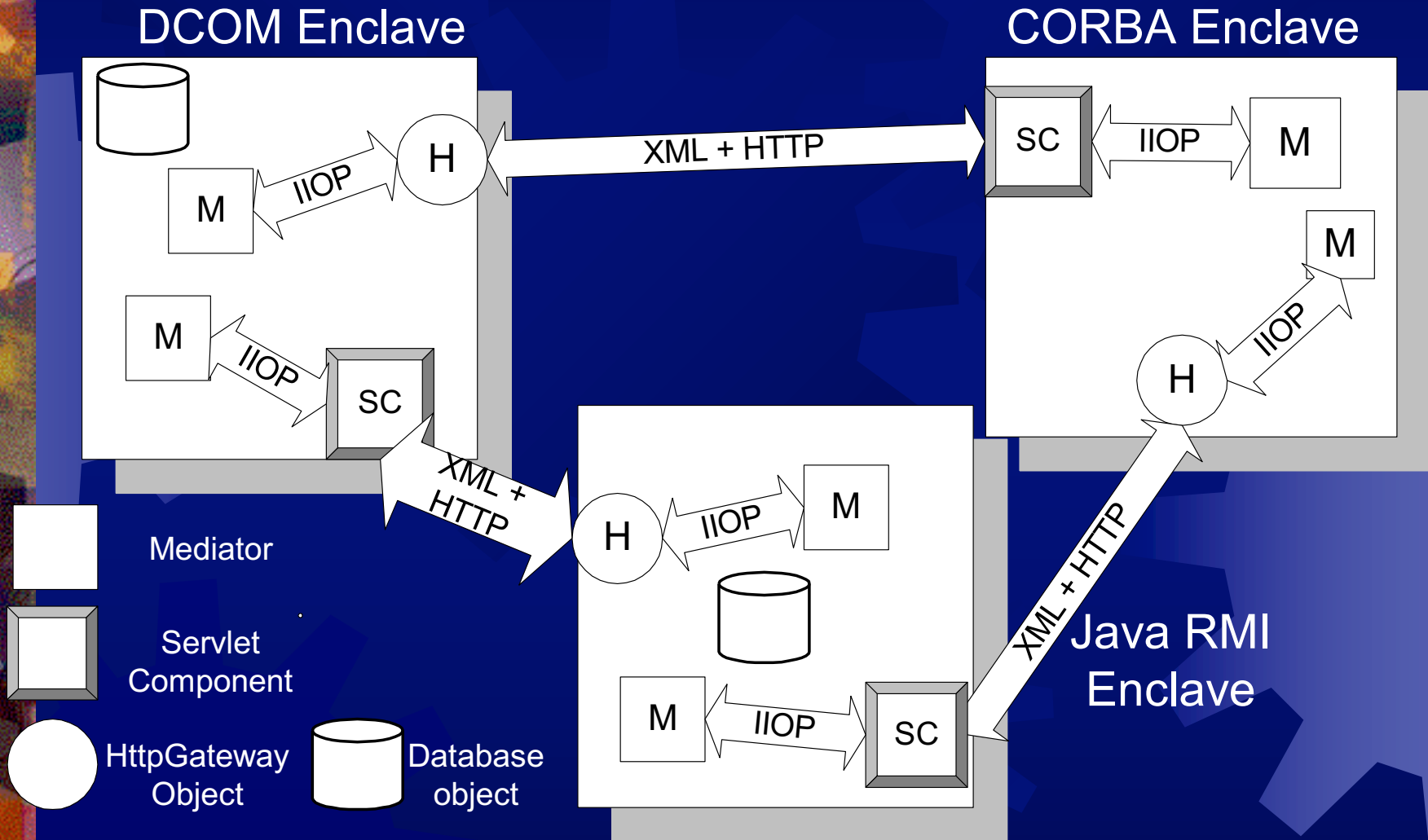
```
// QueryEngine.java
```

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface QueryEngine extends
    Remote {

    String query(SysPara para,
                String queryStmt)
        throws RemoteException;
}
```

Communication Across Heterogeneous Environments



The background features a dark blue field filled with various shades of blue gears of different sizes, some overlapping. On the left side, there is a vertical strip containing a detailed, colorful image of interlocking gears in shades of orange, yellow, and brown.

Evaluation

Performance Evaluation

- ✦ The query system shows our mechanism is practical in applying to real-life applications.
- ✦ We use the query system to evaluate the overhead of our mechanism.

Time Spent on Different Objects

Components	Time (ms)
Mediator Objects (excluding waiting time for the return of query results and connection setup time)	20 - 80
Database Objects	180 - 800
IOP Communications with CORBA enclave (connection within LAN)	10 - 100
Shadow Client or Server (excluding waiting time for the return of query results and connection setup time)	20 - 100
Servlet Components with Servlet Engine (excluding waiting time for the return of query results)	120 - 250
HTTP communications towards other enclaves (connection in WAN)	240 - 2200

Observations

- ✦ Mediator objects are light-weighted objects when comparing to Database objects.
- ✦ The performance of those add-on components are somehow similar to those light-weighted Mediator objects.
- ✦ The most time-consuming part of the whole process is the Internet connection.
- ✦ When comparing to Internet latency, the overhead of our add-on components is not significant.

Means for Enhancement

- ✱ For performance concern, we use HTTP1.1 in the system
 - ✱ Provides persistence connection.
 - ✱ Data compression is done automatically.
- ✱ For security concern, we can apply SSL over HTTP (HTTPS)
 - ✱ Provides encryption to data in transmission.
 - ✱ Provides secure channels with authentication

Advantages of our Mechanism

- ✦ It solves the incompatible firewall problems of communication protocols in distributed environment. Hence, increase the scalability.
- ✦ It solves the incompatible problem of heterogeneous distributed environments. The mechanism is generic to many different distributed environments.
- ✦ No modification is needed to the existing components in the systems when applying our mechanism. Provides transparency to existing components, and avoids potential errors.

Advantages of our Mechanism

- ✦ Systems maintain good security. External objects can only invoke internal objects which are associated with Servlet components. Traditional security methods for HTTP and Servlets can be applied.
- ✦ It prevents information loss as XML can represent data structure well.
- ✦ It provides a gateway for existing systems to link with other web applications.

Our Disadvantages

- ★ Add-on components give extra workload to the systems.
 - They are light-weighted.
 - Their overhead is negligible when comparing to Internet latency.
 - They provide great transparency to the existing objects for invoking objects in other enclaves.
- ★ XML messages is highly readable hence higher level of security is needed. Also XML lengthens the transmission messages.
 - Traditional security methods can be applied, such as HTTPS.
 - HTTP1.1 provides compression to data.

Conclusion

- ✦ We conclude our contributions in the followings:
- ✦ A generic mechanism for distributed objects to communication across firewalls has been proposed;
- ✦ An extension of the mechanism to support callback feature has been proposed;
- ✦ A schema for mapping CORBA IDL to XML format has been proposed, and a translator for that has also been implemented;
- ✦ An extension of the mechanism to support generic remote object calling in heterogeneous environment has been implemented;
- ✦ A mediator-based query system has been implemented to demonstrate our work.



<appreciation> THANK YOU! </appreciation>