

High Level Architecture

Bryant Mayfield

HLA

- HLA handles distributed simulations.
- Works equally well on either large-scale networks or single computers.
- Uses Object Oriented rules and data structures to simplify programming.

HLA Components

- RTI – Backbone used to handle low level protocols and manage data.
- Federation – Used to group and manage Federates under an RTI
- Federates – Hold objects and their attribute values

Runtime Infrastructure Services

- Federation Management
- Declaration Management
- Object Management
- Ownership Management
- Time Management

Federation Management

- Handles creation and deletion federation execution.
- Specifies membership through a .fed file
- Provides services that modify the entire execution.
- Common Functions: Federation Save

Declaration Management

- Manages transport of object attribute and interaction information.
- Federates declare what interactions and attributes they can broadcast or intercept.
- Not responsible for any security.

Object Management

- Allocates and instantiates objects and gives them IDs.
- These IDs are then used for message transactions.
- Two qualities: reliable, best-effort.
- Common Functions: send/receive Interaction, update/reflect Object Attributes.

Ownership Management

- Determines what attributes an object owns.
- Keeps track of who has deletion rights.
- Garbage collection.

Time Management

- Responsible for synchronization and rate.
- Timestamp: Organize according to timestamp
Best Effort or Receive.
- Receive: Process Messages as they are received
- Common Functions: Time set, Time Advance
Grant

Basic HLA Code

```
RTI= RuntimeInfrastructure.getRTIambassador(Host, Port);  
RTI.createFederateExecution(FederationName, "FedFile");  
RTI.joinFederateExecution(FederateName, FederationName, FederateObject);
```

```
InteractionObject= RTI.getInteractionClassHandle(InteractionName);  
ParameterObject= RTI.getParameterClassHandle(ParameterName);  
AttributeObject= RTI.getObjectAttributeHandle(AttributeName);
```

```
RTI.subscribeInteractionClass(InteractionObject);  
RTI.publishInteractionClass(InteractionObject);
```

```
Parameters= RTI.createParameters(); /* */  
Parameters.add(ParameterObject, String("Hello World").getBytes());  
RTI.sendInteraction(InteractionObject, Parameters);
```

Basic HLA Code

```
RTIclass
{
    receiveInteraction(InteractionClass, InteractionObject);
    {
        Parameters= InteractionObject.getParameterHandle();
        message= Parameters.extractValue(); /* */
        out.print(message);
    }

    /*Attribute Handling*/

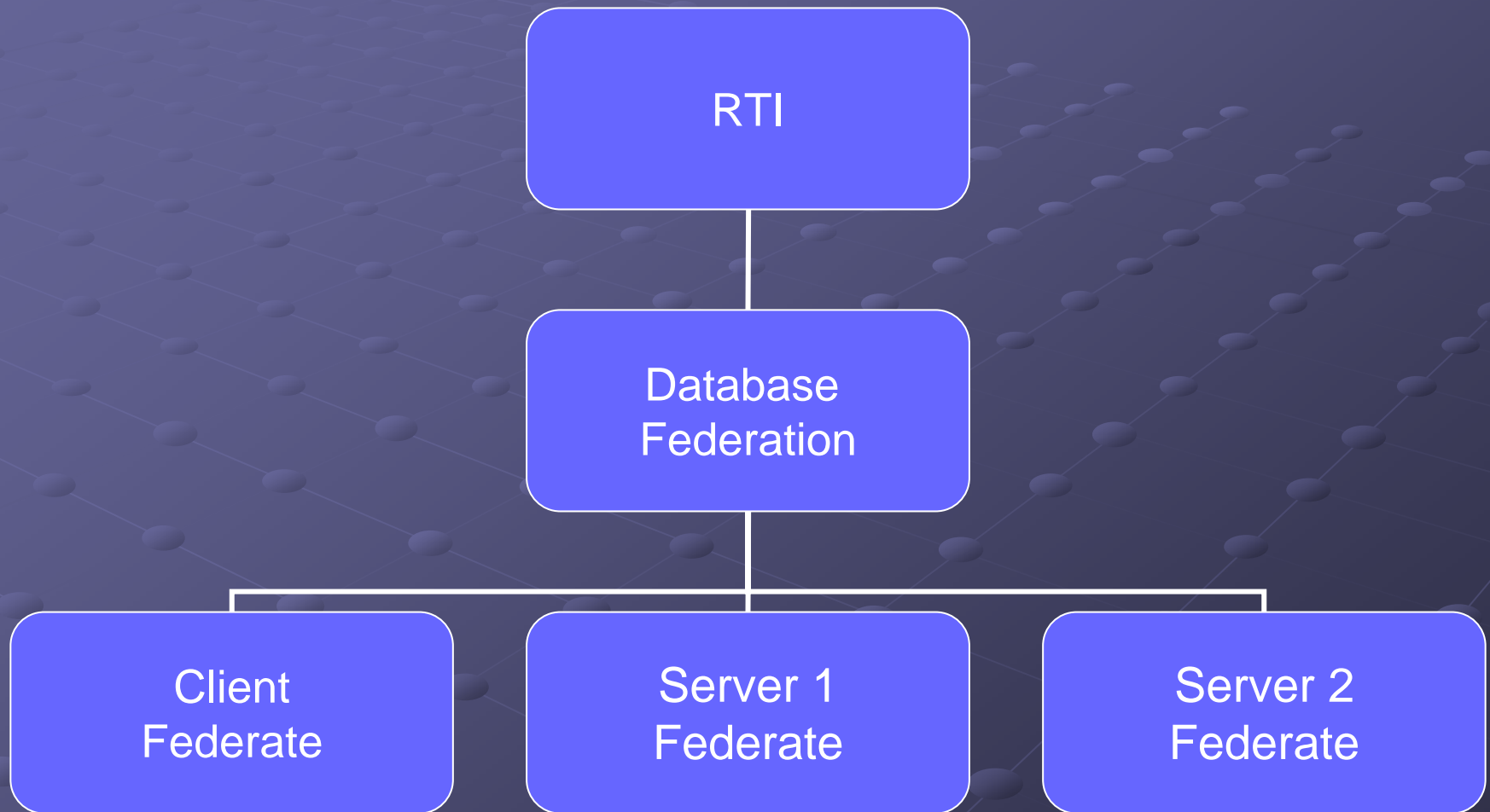
    /*Time Management*/

}
}
```

Basic .fed File

```
(FED
  (Federation FedFileName )
  (FEDversion v1_3 )
  (spaces )
  (objects
    (class objectRoot
      (class MyObjectClass
        (attribute AttributeName best-effort receive) )
      )
    )
  (interactions
    (class InteractionRoot reliable timestamp
      (class InteractionName reliable timestamp
        (parameter ParameterName ) )
      )
    )
  )
)
```

Example Database Federation



RTI

- RTIs implement the services of the HLA.
- Many services are optional and ignored.
- Additional services can be added but this damages portability and interoperability.

Advantages

- **Reusability:** Very little has to be changed between programs.
- **Portability:** Only the RTI is concerned with OS specific nuances.
- **Interoperability:** Networks with the same RTI share protocols.

Disadvantages

- Federations and Federates can only hold values and send requests to the RTI.
- Unable to micromanage network processes.
- Must use encryption for security.

Project: HLA Bridge

- HLA doesn't provide for communication between two Federations.
- This is patchable with a Bridge Federate.
- A Bridge is a member of two Federations, usually on different RTIs.
- It translates between the protocols using two Federate Ambassador objects called surrogates.

HLA Bridge

- Can be used by contractors on a shared simulation to mask proprietary federates.
- Can join two different RTI networks.
- Smart bridges can provide additional security: Firewall, logging, etc.
- Goal is to add as few Services to the RTI as possible. Keep the bridge “dumb.”

Bridge Problems

- **Deadlock:** Bridge Federates wait on themselves when Consensus is needed.
- **Affects:** Federation Save, Time Management, Barrier Synchronization
- **Possible Solutions:** Add services to RTI, Force bridge to temporarily resign.

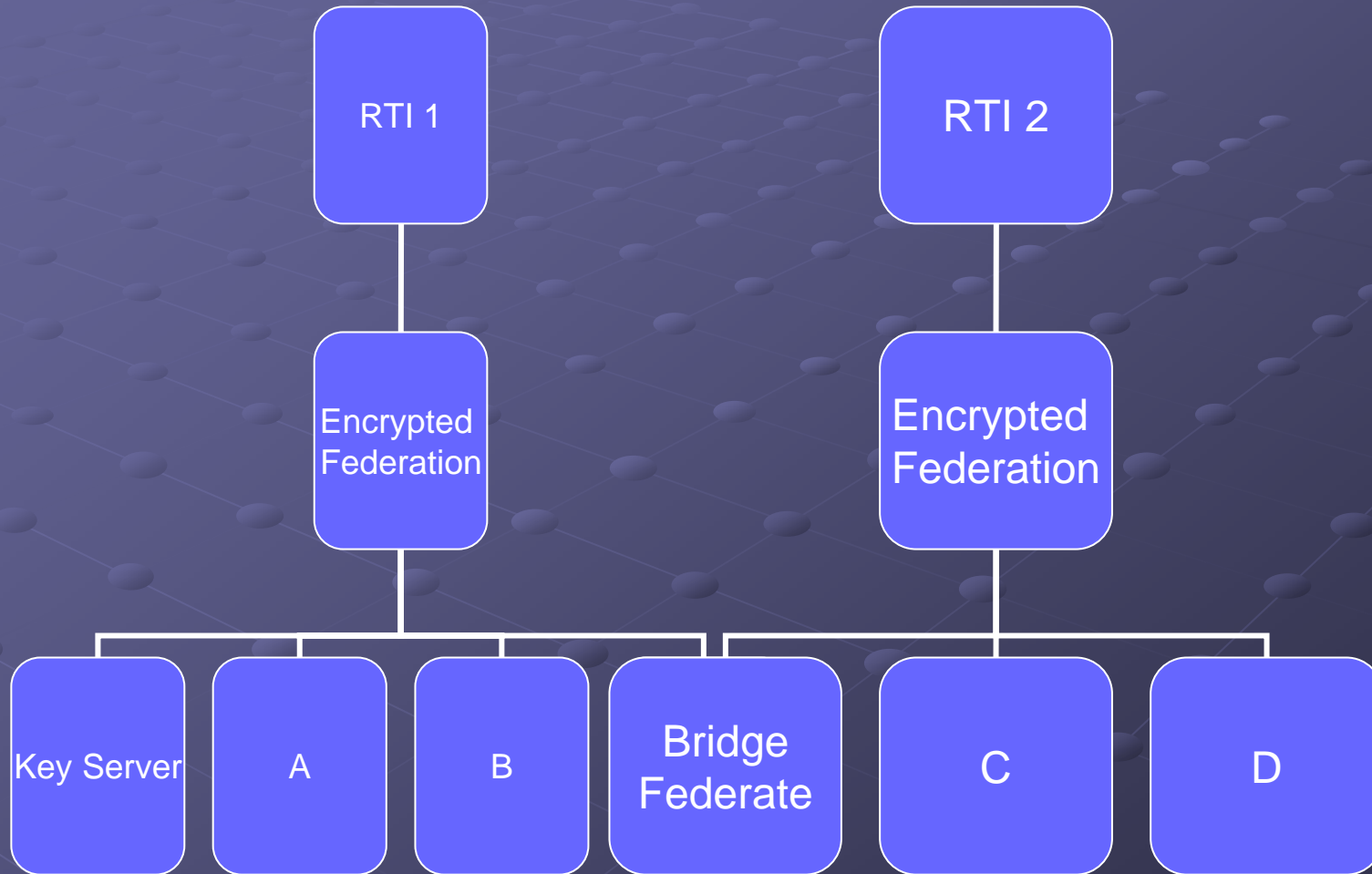
Bridge Problems

- Passing Attributes: values are given to the bridge to cross the other side.
- Cloning: When a bridge resigns both RTIs take possession of the passed attributes.
- Possible Solution: record passed attributes. Then delete them from the wrong RTI before resignation.

Bridge Problems

- Missed Errors: RTIs do not listen for certain critical errors from Federates.
- Possible Solution: Smart bridge logs failed commands, queues related requests until failed command is successfully repeated.
- Most complicated of the problems.

Bridged Key Server



Sources

James O. Calvin, “An Introduction to the High Level Architecture Runtime Infrastructure”, MITRE Corporation, McLean, VA

Dingel, 2001, A Feasibility Study of the HLA Bridge, IEEE, Carnegie Mellon University, Pittsburgh, PA

Dingel, 2002, Bridging the HLA: Problems and Solutions, IEEE, Fort Worth, TX

www.pitch.se – pRTI and chat.java

www.mitre.org – HLA text and pRTI trial

www.cc.gatech.edu/computing/pads/fdk/ - FDK RTI

<http://www.cs.ucf.edu/courses/cda4932.spr02/>

