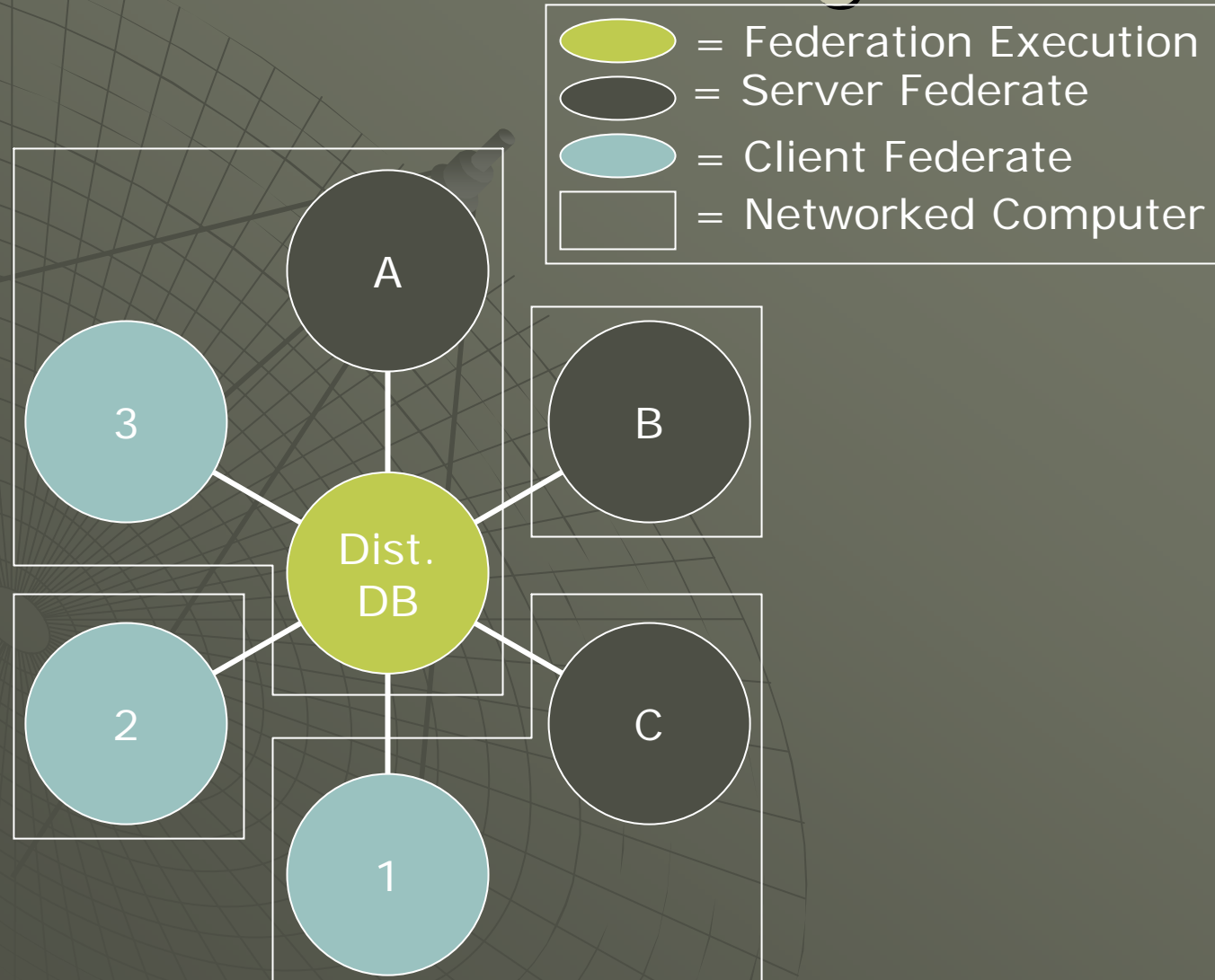


Implementing a Distributed Database Using High Level Architecture (HLA) for Use in Network Gaming.

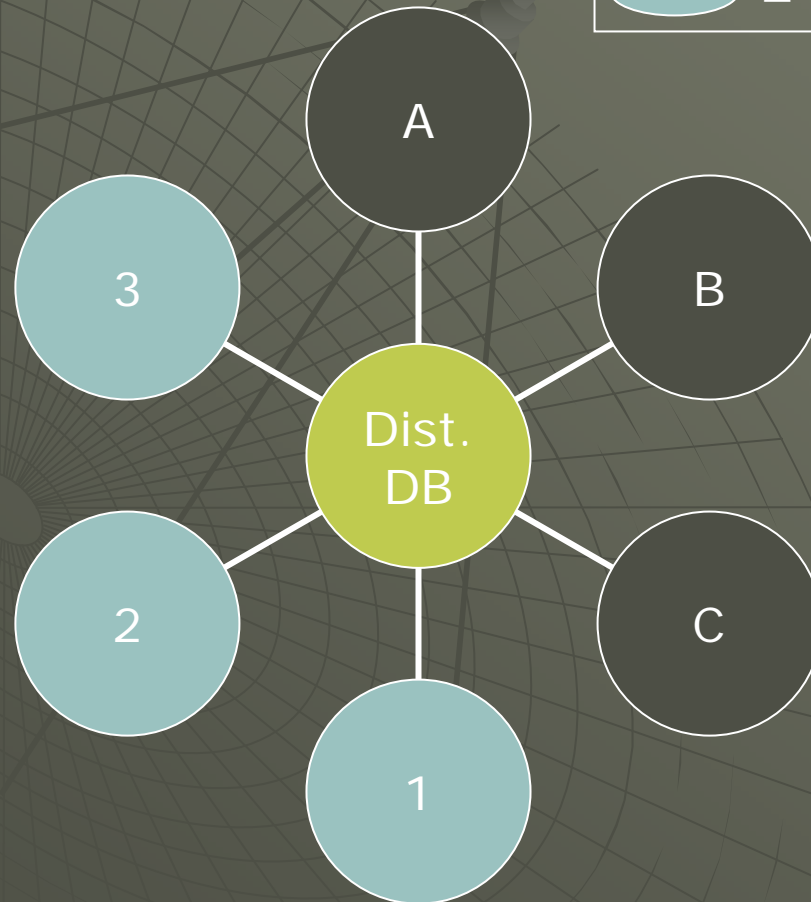
What role does HLA play?

- ◆ Tool for communication abstraction.
- ◆ Primary means for client-server and server-server communications in this database.
- ◆ Each client and server connects to the Run-Time Infrastructure (RTI) which handles all communications.
- ◆ Programs only need be concerned with what is communicated, not how.

Sample Physical Topology of Distributed Database Using HLA



Sample Logical Topology of Distributed Database Using HLA



How is HLA used?

- ◆ Federation
 - Group related Federates.
 - Handle all communication between Federates joined to Federation.
 - Like a chat room for Federates.
- ◆ Federates
 - Used to relay information between running programs.
 - Like chatters connected to chat room.
 - Can be running on different computers.

How is HLA used? (cont.)

◆ Objects

- Represent “real-world” objects in the simulated environment.
- Classes
- Instances
- Use in distributed database
 - ◆ Can represent database records.
 - ◆ Can represent database structure.

How is HLA used? (cont.)

◆ Interactions

- Represent messages between Federates.
- Sent to effect a change of some object.
- Classes
- To send Interaction, fill parameters, then transfer to RTI
- Use in distributed database
 - ◆ Can be used to send requests for records
 - ◆ Can be used to send notifications of server actions

Statically vs. Dynamically Distributed

- ◆ Statically Distributed servers use hash function to distribute records.
 - Example:
 - ◆ Server 0 holds all records where LastName starts with A-M, Server 1 holds all records where LastName starts with N-Z. (Hash: $(\text{LastName}[0] - 'A') / 13$)
 - Advantages:
 - ◆ Ease of coding.
 - ◆ Little overhead.
 - Problems:
 - ◆ Difficult to predict well balanced hash functions.
 - ◆ Changing # of servers requires restart.

Statically vs. Dynamically Distributed

- ◆ Dynamically Distributed servers use load balancing techniques to distribute records.
 - Example:
 - ◆ Any new record inserted into database is picked up by server with fewest records.
 - Advantages
 - ◆ Balances record loads better to speed up queries.
 - ◆ Takes full advantage of distribution.
 - Problems:
 - ◆ Load balancing code adds overhead.
 - ◆ Adding or removing servers can temporarily overwhelm RTI.

Client Actions

- ◆ Clients only need to connect to RTI, subscribe to response interactions, broadcast request interactions.
- ◆ Database structure is mostly irrelevant to clients.
- ◆ Clients DO need to know number of servers in system to know when all servers have responded to requests.

Client Difficulties

- ◆ Due to broadcast nature of HLA communication, what happens if Client A requests records at the same time as Client B, or if Client A requests a second set of records before the servers can respond to the previous request?
 - Requests must be sent with identifying information that is reflected by server responses.
 - Clients use reflected information to match responses with requests, and to ignore responses to other clients' requests.

Statically-Distributed Server Actions

- ◆ Listen for requests.
 - Requests for records
 - ◆ Gather copies of all records matching given criteria, then broadcast results.
 - Requests to add records
 - ◆ If record is within server's range, it is added, otherwise it is ignored.
 - Requests to delete records
 - ◆ Search records for those matching given criteria and remove them.

Dynamically-Distributed Server Actions

- ◆ Record requests and Deletion requests same as Statically-Distributed Server's.
- ◆ When adding a record, new record should go to server with fewest records.
- ◆ When new server comes online:
 - Can remain empty and gather new records.
 - Can ask other servers for a portion of their records.
- ◆ When a server goes offline:
 - Must distribute its records among remaining servers.

Work to be continued

- ◆ Finish coding of Database.
- ◆ Test speed of systems using various numbers of servers, clients, and computers, as well as different RTIs.
- ◆ Test network utilization of systems using various numbers of servers, clients, and computers, as well as different RTIs.
- ◆ Possibly test against other forms of data distribution systems.

Work to be continued

- ◆ Create games using distributed database.
 - Simple 2 player game with multiple concurrent games (e.g. online checkers).
 - Complex multi-player game with interactions between all players.
- ◆ Use results of game play to test viability of HLA for network or Internet gaming.

Expectations

- ◆ Speed increase due to added servers will vary inversely with the number of servers running per CPU.
 - Multiple servers running against 1 CPU not only add overhead, but defeat distribution.
 - A system with 1 server per CPU takes full advantage of distribution.
 - Interference due to database clients, mostly using little CPU time and network bandwidth, would be insignificant.
 - Large client requests could briefly overwhelm system, but probably not in a meaningful manner.
- ◆ Cluster system would run faster than HLA system.
 - Less network latency.
 - More expensive.

Expectations for Gaming

- ◆ HLA good for network gaming.
 - Easy to implement.
 - Inexpensive.
 - No new hardware.
 - Good network utilization.
 - RTI can be built into “Hosted” game.
 - “Spawned” games can take more responsibility for game data, increasing distribution and increasing speed.