Assignment 1 addendum

ECE 454 / 750: Distributed Computing

Instructor: Dr. Wojciech Golab wgolab@uwaterloo.ca

Assignment TA: Hua Fan h27fan@uwaterloo.ca

Firewall ☹

- Some of the ecelinux servers are firewalled and are not allowing remote connections except on a handful of ports.
- As a workaround test your code by running all components on the **same machine**.
- Example:

```
java -cp "ece454750s15a1.jar:../lib/*"
ece454750s15a1.BEServer
-host ecelinux1 -pport 11234 -mport 21234 -ncores 2
-seeds ecelinux1:31234,ecelinux1:41234,ecelinux1:51234
```

Helpful hints

- Do ask technical questions in the discussion forum.
- Do implement and test one feature at a time.
- Do use Thrift protocols and transports consistently.

- Do not store unnecessary files (e.g., from sample projects) in your project directory.
- Do not hardcode host names or port numbers.
- Do not overthink the design.

Some basic facts

- FE and BE nodes must both implement the A1Password and A1Management service.
- The two services in the same process, or running on the same host, must use distinct ports.
- Each FE and BE node is a separate Java process and is given the same list of seeds.
- An FE seed is an FE node whose own host and management service port appear in the list of seeds.
- FE seeds never leave the cluster permanently, but they may be restarted occasionally.

Example file organization

BE node:

- ece454750s15a1/BEServer.java (contains mainline)
- ece454750s15a1/BEPasswordHandler.java
- ece454750s15a1/BEManagementHandler.java

FE node:

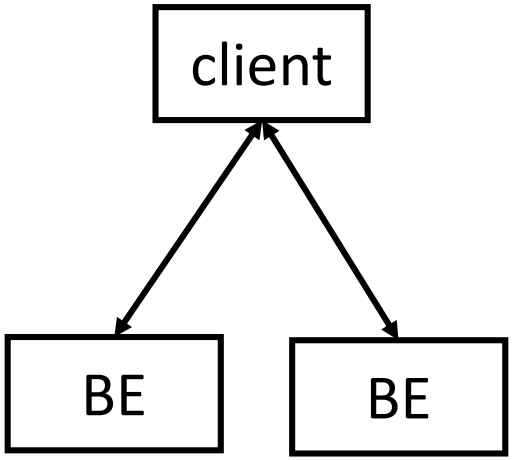
- ece454750s15a1/FEServer.java (contains mainline)
- ece454750s15a1/FEPasswordHandler.java
- ece454750s15a1/FEManagementHandler.java

Client:

- ece454750s15a1/TestClient.java (contains mainline)

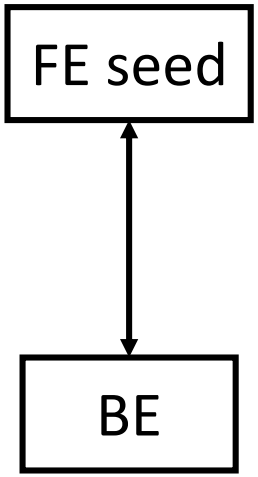
Where do I begin?

- Implement the A1Password service in the BE node.
- The hashPassword and checkPassword procedures require about one line of code each to implement in the service handler (see http://www.mindrot.org/projects/jBCrypt/).
- Have the client connect directly to the BE nodes and call hashPassword.



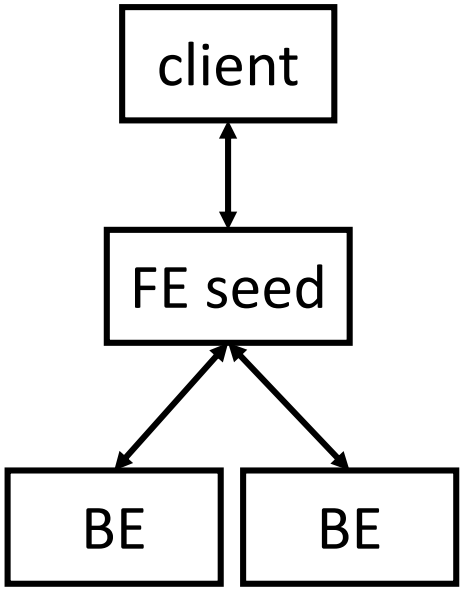
Step 2: BE node joins cluster

- Implement the A1Management service in the FE node. Add an interface to this service to allow BE nodes to join the cluster.
- On startup a BE node contacts one of the FE seeds and reports the following details from its command line arguments: host, pport, mport, ncores.
- Have the FE seed store this data using internal data structures. Use thread-safe structures from the java.util.concurrent package.



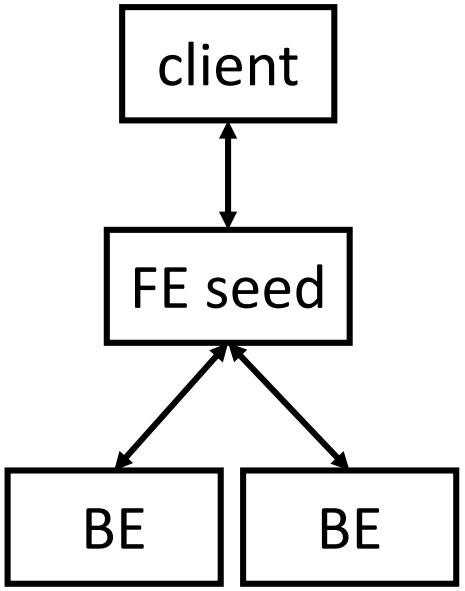
Step 3: FE forwards requests

- Implement the A1Password service in the FE node.
- When the FE node receives a request from the client, it connects to one of the BE nodes that have registered with this FE node.
- The FE creates and tears down a connection to a BE node for each request it receives from the client.



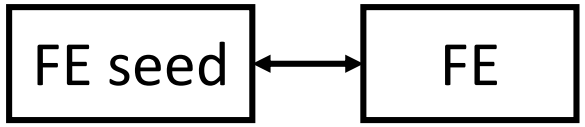
Step 4: load balancing

- When forwarding requests, initially have the FE node choose one of the BE nodes uniformly at random.
- Later on modify the implementation so that the choice of BE node takes into account the number of cores allocated to the BE node.



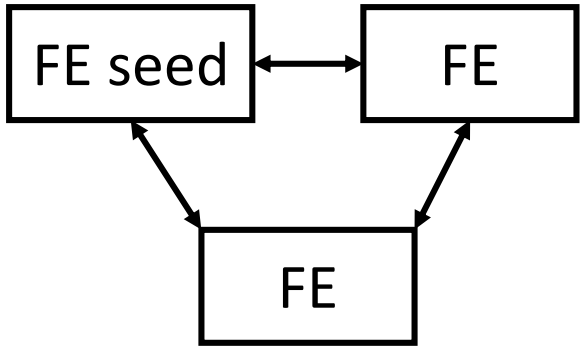
Step 5: FE node joins cluster

- On startup an FE node contacts one of the FE seeds, similarly to a BE node in step 2, and reports the same details (host, etc.).
- The FE seed stores this data using an internal data structure. Information about FE nodes and BE nodes is stored separately.



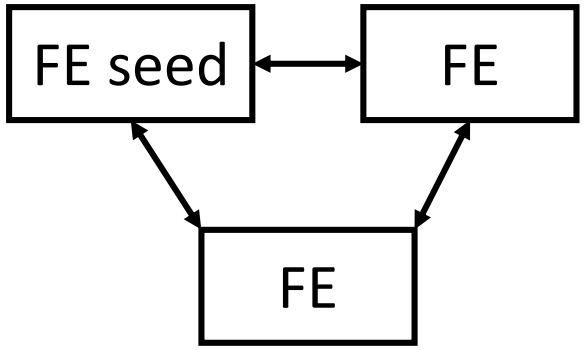
Step 6: FE node gossip protocol

- The FE nodes periodically exchange their lists of known FE and BE nodes by calling a procedure in the A1Management service.
- When a BE node joins the cluster, its information should be propagated to all the FE nodes in less than 1 second.
- Run this protocol in a separate thread at 100ms intervals.



Step 7: FE node gossip protocol

- Similarly, when the crash failure of an FE or BE node is detected (e.g., connection refused exception) the FE nodes inform each other.
- After a BE node fails the FE nodes should stop sending requests to it within a few seconds.



Step 8: final touches

- Smart connection handling: FE node tries to reuse connections to BE nodes efficiently rather than creating them and tearing them down for each request received from the client.
- ServiceUnavailable exception: when all else fails (i.e., all the BE nodes are down for about 60 seconds) the FE node should throw a ServiceUnavaialble exception at the client.