Assignment2—Report

Code Part:

State class：

```
public synchronized NodeInfo closestPrecedingFinger(int id) {
	/*
	 * TODO: Get closest preceding finger for id, to continue search at
that
	 * node. Hint: See DHTBase.inInterval()
	 */
	for (int i = IRouting.NFINGERS - 1; i >= 0; i--) {
		//if (finger[i] ∈ (n, id)) return finger[i]
		if(DHTBase.inInterval(finger[i].id,info.id, id, false)){
			return finger[i];
			}
		}
	return info;
}
```

Use a for loop to lookup in the finger table if the finger[i].id is between the current node.id and id that need to find then return the finger[i] or return the current node.

The code in DHT.java ,webClient.java,NodeResource.java and NodeService.java

In the DHT.java:

```
	private NodeInfo getSucc(NodeInfo info) throws Failed {
		NodeInfo localInfo = this.getNodeInfo();
		if (localInfo.addr.equals(info.addr)) {
			return getSucc();
		} else {
			// TODO: Do the Web service call
			return client.getSucc(info);
		}

	}
```

If not in the local node then call the web service by using the client object.

In WebClient.java:

```
	/*
	 * @author Tianpei Luo
	 * Get the succuessor pointer at a node.
	 */
	public NodeInfo getSucc(NodeInfo node) throws DHTBase.Failed {
		URI succPath = UriBuilder.fromUri(node.addr).path("succ").build();
		info("client getSucc(" + succPath + ")");
		Response response = getRequest(succPath);
		if (response == null || response.getStatus() >= 300) {
			throw new DHTBase.Failed("GET /succ");
		} else {
			NodeInfo succ = response.readEntity(nodeInfoType).getValue();
			return succ;
		}
	}
```

Initialize the path ,calling the getRequest() by passing the path and get the value from the response with using the nodeInfoType.

Response in NodeResourse.java:

```
	@GET
	@Path("succ")
	@Produces("application/xml")
	public Response getSucc(){
		return new NodeService(headers,uriInfo).getSucc();
	}
```

Using the Jax-rs to return the NodeService.getSucc():

```
	public Response getSucc() {
			advanceTime();
			info("getSucc()");
			return response(dht.getSucc());
	}
```

By response the current node getSucc() method to return the successor of query node.

In the DHT.java:

```
protected NodeInfo getPred(NodeInfo info) throws Failed {
	NodeInfo localInfo = this.getNodeInfo();
	if (localInfo.addr.equals(info.addr)) {
		return getPred();
	} else {
		/*
		 * TODO: Do the Web service call
		 */
		return client.getPred(info);
	}
}
```

If not in the local node then call the web service by using the client object.

In WebClient.java:

```
/*
 * Get the predecessor pointer at a node.
 */
public NodeInfo getPred(NodeInfo node) throws DHTBase.Failed {
	URI predPath = UriBuilder.fromUri(node.addr).path("pred").build();
	info("client getPred(" + predPath + ")");
	Response response = getRequest(predPath);
	if (response == null || response.getStatus() >= 300) {
		throw new DHTBase.Failed("GET /pred");
	} else {
		NodeInfo pred = response.readEntity(nodeInfoType).getValue();
		return pred;
	}
}
```

Initialize the path ,calling the getRequest() by passing the path and get the value from the response with using the nodeInfoType.

Response in NodeResourse.java:

```
@GET
@Path("pred")
@Produces("application/xml")
public Response getPred() {
	return new NodeService(headers, uriInfo).getPred();
}
```

Using the Jax-rs to return the NodeService.getSucc():

```
	public Response getPred() {
	advanceTime();
	info("getPred()");
	return response(dht.getPred());
}
```

By response the current node getPred() method to return the Predecessor of query node.

In the DHT.java:

```
	protected NodeInfo closestPrecedingFinger(NodeInfo info, int id) throws Failed
{
		NodeInfo localInfo = this.getNodeInfo();
		if (localInfo.equals(info)) {
			return closestPrecedingFinger(id);
		} else {
			if (IRouting.USE_FINGER_TABLE) {
				return client.closestPrecedingFinger(info, id);
			} else {

				return getSucc(info);
			}
		}
	}
```

If not in the local node then call the web service closestPrecedingFinger(info, id);

In WebClient.java:

```
	public NodeInfo closestPrecedingFinger(NodeInfo node,int id) throws DHT-
Base.Failed{
		UriBuilder ub = UriBuilder.fromUri(node.addr).path("findClosestPreced-
ingFinger");
		URI getPath = ub.queryParam("id", id).build();
		info("client findtheclosestfinget("+getPath+")");
		Response response = getRequest(getPath);
		if (response == null || response.getStatus() >= 300) {
			throw new DHTBase.Failed("GET /findClosestPrecedingFin-
ger?id=ID");
		} else {
			NodeInfo cFinger = response.readEntity(nodeInfoType).getValue();
			return cFinger;
		}
	}
```

Initialize the path ,calling the getRequest() by passing the path and get the value from the response with using the nodeInfoType.

Response in NodeResourse.java:

```
@GET
@Path("findClosestPrecedingFinger")
@Produces("application/xml")
public Response findClosestPrecedingFinger(@QueryParam("id") String index){
	int id = Integer.parseInt(index);
	return new NodeService(headers,uriInfo).findClosestPrecedingFinger(id);
}
```

Using the Jax-rs to pass the queryparam id to the method of the nodeservice and call NodeService.findClosestPrecedingFinger(id);

```
public Response findClosestPrecedingFinger(int id) {
		advanceTime();
		info("findClosestPrecedingFinger()");
		return response(dht.closestPrecedingFinger(id));
	}
```

By response the current node closestPrecedingFinger(id) method to return the closet predecessor finger table of id;

The get method is DHT.java which call client.get(n,k) if the n is not the current node:

```
protected String[] get(NodeInfo n, String k) throws Failed {
		if (n.addr.equals(info.addr)) {
			try {
				return this.get(k);
			} catch (Invalid e) {
				severe("Get: invalid internal inputs: " + e);
				throw new IllegalArgumentException(e);
			}
		} else {
			/*
			 * Retrieve the bindings at the specified node.
			 *
			 * TODO: Do the Web service call.
			 */
			return client.get(n,k);

		}
	}
```

In the DHT.java ,findSuccessor() will find the successor of node k and return the successor node

```
	/*
	 * Find the successor of k, given the URI for a node. Used as client part of
	 * the join protocol.
	 */
	protected NodeInfo findSuccessor(URI addr, int id) throws Failed {
		return client.findSuccessor(addr, id);
	}
```

The method in WebClient.java:

```
	 * Find the successor of k, given the URI for a node. Used as client part of
	 * the join protocol.
	 */
	public NodeInfo findSuccessor(URI addr, int id) throws DHTBase.Failed {
		// TODO Auto-generated method stub
		UriBuilder ub = UriBuilder.fromUri(addr).path("find");
		URI findPath = ub.queryParam("id", id).build();
		info("clinet findSuccessor("+findPath+")");
		Response response=getRequest(findPath);
		if(response == null || response.getStatus()>=300){
			throw new DHTBase.Failed("GET /find?id=ID");
		}else{
			return response.readEntity(nodeInfoType).getValue();
		}
	}
```

The method below will call the getRequest() and get the response of a node py using the NodeResource path:

```
	@GET
	@Path("find")
	@Produces("application/xml")
	public Response findSuccessor(@QueryParam("id") String index) {
		int id = Integer.parseInt(index);
		return new NodeService(headers, uriInfo).findSuccessor(id);
	}
```

The findSuccessor() method in the nodeService has already exist in NodeService.java:

```
public Response findSuccessor(int id) {
		try {
			advanceTime();
			info("findSuccessor()");
			return response(dht.findSuccessor(id));
		} catch (Failed e) {
			throw new
WebApplicationException(Response.Status.SERVICE_UNAVAILABLE);
		}
	}
```

The last two method is add and delete :

The add method is DHT.java:

```
public void add(NodeInfo n, String k, String v) throws Failed {
		if (n.addr.equals(info.addr)) {
			try {
				add(k, v);
			} catch (Invalid e) {
				severe("Add: invalid internal inputs: " + e);
				throw new IllegalArgumentException(e);
			}
		} else {
			/*
			 * TODO: Do the Web service call.
			 */
			client.add(n,k,v);

		}
	}
```

The code in WebClient.java:

```
	public void add(NodeInfo node, String key, String value) throws
DHTBase.Failed{
		UriBuilder ub = UriBuilder.fromUri(node.addr).path("add");
		URI addPath = ub.queryParam("key", key).queryParam("value",
value).build();
		TableRep tablerep = new TableRep(null,null,1);
		tablerep.entry[0] = new TableRow(key,new String[]{value});
		info("client add("+addPath+")");
		Response response = putRequest(addPath,Entity.xml(tablerep));
		if(response==null || response.getStatus()>=300){
			throw new DHTBase.Failed("PUT /add?key=KEY&val=VAL");
		}
	}
```

Just use the put request to put the tablerep into the target node.

```
	@PUT
	@Path("add")
	@Produces("application/xml")
	public Response add(TableRep tablerep) throws Invalid{
		return new
NodeService(headers,uriInfo).add(tablerep.entry[0].key,tablerep.entry[0].vals);
	}
```

Call the nodeService by passing the key and value of tablerep into the method

```
	public Response add(String key, String[] vals) throws Invalid {
			advanceTime();
			info("add()");
			dht.add(key, vals[0]);
			return response();
	}
```

The last one is the delete method:

```
	public void delete(NodeInfo n, String k, String v) throws Failed {
		if (n.addr.equals(info.addr)) {
			try {
				delete(k, v);
			} catch (Invalid e) {
				severe("Delete: invalid internal inputs: " + e);
				throw new IllegalArgumentException(e);
			}
		} else {
			/*
			 * TODO: Do the Web service call.
			 */
			client.delete(n,k,v);

		}
	}
```

If the node is not the current node then call the web service again.

```
	public void delete(NodeInfo node, String key, String value) throws
DHTBase.Failed{
		UriBuilder ub = UriBuilder.fromUri(node.addr).path("delete");
		URI delPath = ub.queryParam("key", key).queryParam("value",
value).build();

		info("client delete("+delPath+")");
		Response response = deleteRequest(delPath);
		if(response==null || response.getStatus()>=300){
			throw new DHTBase.Failed("DELETE /delete?key=KEY&val=VAL");
		}
	}
```

Same with the code above, using a deleteRequest to delete the delPath by passing the key and value.

The delete request:

```
	private Response deleteRequest(URI uri) {
		try {
			Response cr = client.target(uri)
					.request(MediaType.APPLICATION_XML_TYPE)
					.header(Time.TIME_STAMP, Time.advanceTime())
					.delete();
			processResponseTimestamp(cr);
			return cr;
		} catch (Exception e) {
			error("Exception during delete request: " + e);
			return null;
		}
```

The delete path in NodeResource:

```
	@DELETE
	@Path("delete")
	@Produces("application/xml")
	public Response delete(@QueryParam("key") String key,@QueryParam("value")
String value) throws Invalid{
		return new NodeService(headers,uriInfo).delete(key,value);
	}
```

The nodeService of delete:

```
	public Response delete(String key, String value) throws Invalid{
			advanceTime();
			info("delete()");
			dht.delete(key, value);
			return response();
	}
```

Test part:

Join operation, for node 44 and node23 to join the node 55:

```
Entering command-line interface...
dht<44> join http://ec2-54-174-135-12.compute-1.amazonaws.com:8080/dht
```

```
dht<23> join http://ec2-54-174-135-12.compute-1.amazonaws.com:8080/dht
```

User routes operation to see the successor and predecessor of current node and finger table:

```
dht<55> routes
Predecessor: [id=44,addr=http://ec2-54-165-246-138.compute-1.amazonaws.com:8080/dht]
Successor  : [id=23,addr=http://ec2-54-209-186-10.compute-1.amazonaws.com:8080/dht]
Fingers:
Formula  Key  Succ
-------  ---  ----
 55+2^0   56  [id=23,uri=http://ec2-54-209-186-10.compute-1.amazonaws.com:8080/dht]
 55+2^1   57  [id=23,uri=http://ec2-54-209-186-10.compute-1.amazonaws.com:8080/dht]
 55+2^2   59  [id=23,uri=http://ec2-54-209-186-10.compute-1.amazonaws.com:8080/dht]
 55+2^3   63  [id=23,uri=http://ec2-54-209-186-10.compute-1.amazonaws.com:8080/dht]
 55+2^4    7  [id=23,uri=http://ec2-54-209-186-10.compute-1.amazonaws.com:8080/dht]
 55+2^5   23  [id=23,uri=http://ec2-54-209-186-10.compute-1.amazonaws.com:8080/dht]
```

We could found that the predecessor has changed to node 44 and successor has changed to node 23 and all the finger table has point to node 23 which is correct.

The routes in node 44 and node 23:

```
Formula  Key  Succ
-------  ---  ----
 23+2^0   24  [id=44,uri=http://ec2-54-165-246-138.compute-1.amazonaws.com:8080/dht]
 23+2^1   25  [id=44,uri=http://ec2-54-165-246-138.compute-1.amazonaws.com:8080/dht]
 23+2^2   27  [id=44,uri=http://ec2-54-165-246-138.compute-1.amazonaws.com:8080/dht]
 23+2^3   31  [id=44,uri=http://ec2-54-165-246-138.compute-1.amazonaws.com:8080/dht]
 23+2^4   39  [id=44,uri=http://ec2-54-165-246-138.compute-1.amazonaws.com:8080/dht]
 23+2^5   55  [id=55,uri=http://ec2-54-174-135-12.compute-1.amazonaws.com:8080/dht]
```

```
Predecessor: [id=23,addr=http://ec2-54-209-186-10.compute-1.amazonaws.com:8080/dht]
Successor  : [id=55,addr=http://ec2-54-174-135-12.compute-1.amazonaws.com:8080/dht]
Fingers:
Formula  Key  Succ
-------  ---  ----
 44+2^0   45  [id=55,uri=http://ec2-54-174-135-12.compute-1.amazonaws.com:8080/dht]
 44+2^1   46  [id=55,uri=http://ec2-54-174-135-12.compute-1.amazonaws.com:8080/dht]
 44+2^2   48  [id=55,uri=http://ec2-54-174-135-12.compute-1.amazonaws.com:8080/dht]
 44+2^3   52  [id=55,uri=http://ec2-54-174-135-12.compute-1.amazonaws.com:8080/dht]
 44+2^4   60  [id=23,uri=http://ec2-54-209-186-10.compute-1.amazonaws.com:8080/dht]
 44+2^5   12  [id=23,uri=http://ec2-54-209-186-10.compute-1.amazonaws.com:8080/dht]
```

Which means the node has insert into the DHT network.

Add test (local node):

1. Added foo3 fourth in the node 55 which should be add in the local node

```
dht<55> add foo3 fourth
```

2.User bindings to see if the node has the entry of the foo3.

```
dht<55> bindings
KEYSTRING  ID  VALUES
     foo3  45  {fourth}
```

Add test (remote node):

1. Added key:foo and value:first in the node 55 as remote node

```
dht<55> add foo first
```

2. Found that the foo and first has been send to the node 23, nothing store in node 55.

```
dht<55> bindings
No entries.
```

```
dht<23> bindings
KEYSTRING  ID  VALUES
      foo   6  {first}
```

Get test (local)

1. Testing the get method in node44 and get the response with the value of {first}.

```
dht<44> get foo2
{second,third}
```

Get test (remote)

1.Testing the get method in node44 for the foo which .

```
dht<44> get foo
{first}
```

Delete test:

Bindings before delete:

```
dht<23> bindings
KEYSTRING  ID  VALUES
     bar2  63  {234}
      bar  19  {123}
     bar4   1  {456}
      foo   6  {first}
```

Delete test in node 23 with delete (foo ,first)

```
dht<23> delete foo first
```

Result:

```
dht<23> bindings
KEYSTRING  ID  VALUES
     bar2  63  {234}
      bar  19  {123}
     bar4   1  {456}
      foo   6  {}
```

The node that store key foo. We could found that the value has been deleted.

Some other operations is in the video.