Project 3

Yao Liu
Nodes join and leave the DHT

- In Project 2, DHT nodes are started once and remain accessible. Fingertable is computed given the information about all nodes in the DHT and sent to each node.

- In Project 3, nodes will join the DHT one by one. Once a new node joins, it will:
  - update the fingertables of existing nodes that are affected by the join.
  - assume the responsibility for files that were held by other node.

- Nodes may also gracefully leave the DHT
Node 25 joins the Chord DHT

The figure below shows how an RPC system works in practice. Explain what is in the runtime library.

Typically parameterizable calls to the underlying transport-level interface, but notably also library routines for converting data structures to host-independent representations.

In RPC, a client needs to bind to a server. What does this mean and how can it be realized?

Binding in this case means that the client makes all the necessary preparations to allow it to call procedures maintained by the server. It is accomplished by first having the server register, one way or the other, to which network-level and transport-level address it is accepting incoming requests. This can be done through a separate, well-known directory server. A client asks for this contact information, after which it can, for example, set up a TCP connection to the appropriate server.

Explain how name resolution works in Chord by resolving $k = 30$ starting from node 21 in the following example. Do the same for $k = 19$ from 21.

In Chord, the finger table entry $FT_p[i]$ of peer $p$ is equal to $\text{succ}(p + 2^i)$. Explain how Chord's finger tables can be extended to incorporate proximity routing.

There is no reason why $p$ can't just keep a whole number of references to nodes in the range $[p + 2^i, p + 2^i + 1]$. In that case, when it is required to look up a key $k$, it can decide to route that request to the peer with the smallest id $k$ that it knows, but which is also closest to itself.
Create a new fingertable for Node 25

• Node 25 can contact an arbitrary node that already exists in the DHT, and ask this node to compute fingertable entries for it using the \texttt{findSucc()} call.
The figure below shows how an RPC system works in practice. Explain what is in the runtime library.

C compiler

Uuidgen

IDL compiler

C compiler

Linker

C compiler

C compiler

Linker

Server stub

object file

Server

object file

Client stub

object file

Client

object file

Client stub

Client code

Header

Server stub

Interface definition file

Server code

Typically parameterizable calls to the underlying transport-level interface, but notably also library routines for converting data structures to host-independent representations.

In RPC, a client needs to bind to a server. What does this mean and how can it be realized?

Binding in this case means that the client makes all the necessary preparations to allow it to call procedures maintained by the server. It is accomplished by first having the server register, one way or the other, to which network-level and transport-level address it is accepting incoming requests. This can be done through a separate, well-known directory server. A client asks for this contact information, after which it can, for example, set up a TCP connection to the appropriate server.

Explain how name resolution works in Chord by resolving \( k = 30 \) starting from node 21 in the following example. Do the same for \( k = 19 \) from 21.

Fingertable updates after Node 25 joins

In Chord, the finger table entry \( FT_p[i] \) of peer \( p \) is equal to \( \text{succ}(p + 2^i 1) \). Explain how Chord's finger tables can be extended to incorporate proximity routing.

There is no reason why \( p \) can't just keep a whole number of references to nodes in the range \( [p + 2^i 1, p + 2^i] \). In that case, when it is required to look up a key \( k \), it can decide to route that request to the peer with the smallest id \( k \) that it knows, but which is also closest to itself.
Fingertable updates

- For each node \( p \) that \( p + 2^i \) belongs to the interval \((\text{pred}(\text{node}_{\text{new}}), \text{node}_{\text{new}}]\), the new node will update node \( p \)'s \( i \)\(^{\text{th}} \) entry in the fingertable. (counting from 0)
  - use the \textit{updateFinger()} call

- A new node affects \( O(\log(N)) \) other fingertable entries in the system, on average

- Number of messages per node join= \( O(\log(N) \times \log(N)) \)
Pull files from Node 28 to Node 25

Node 25 also need to assume responsibility for files with key \{22, 23, 24, 25\}.

1. Node 25 uses `setNodePred()` to set the new predecessor for Node 28
2. Send `pullUnownedFiles()` call to Node 28
3. Node 28 retrieves a set of RFiles whose keys are among \{22, 23, 24, 25\}, and returns them.
An existing node leaves the DHT

• Similar to how we deal with node joining

• For each node $p$ that $p+2^i$ belongs to the interval $(\text{pred}(\text{node}_{\text{leave}}), \text{node}_{\text{leave}}]$, we need to update node $p$’s $i^{\text{th}}$ entry (counting from 0) in the fingertable from $\text{node}_{\text{leave}}$ to $\text{succ}(\text{node}_{\text{leave}})$.

• Push files it was responsible to its successor. Use the $\text{pushUnownedFiles}()$ call.
For debugging

• Use the `cmp_fingertables` code supplied.
• Compare your fingertable with the output from `cmp_fingertables`.