University of Waterloo
Faculty of Engineering
Department of Electrical and Computer Engineering

ECE 454
Assignment 2

Prepared by
Chan, Carl
UW Student ID Number: 20383063
UW User ID: c73chan@uwaterloo.ca
and
Li, Debin
UW Student ID Number: 20389489
UW User ID: d73li@uwaterloo.ca

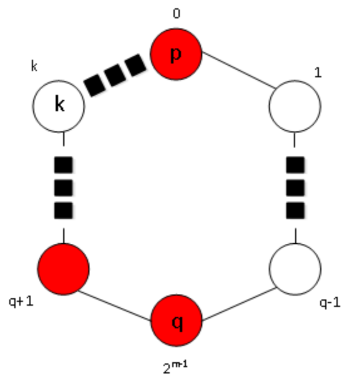
2 June 2014

1. The claim “$q - p \geq (k - p)/2$” will be disproven by counterexample. This claim is false for any case where there are equal or more peers between k and q as there are between q and p. For instance, the figure below has m-bit chord architecture, with peers (in red) between k and q, but not q and p. Assume that there is at least peer from “k” to “p” so p does not just route back to itself. For the purposes of our example, assume that q has identifier 2^{m-1} so that every entry in the finger table of p will point to it. Now, since in this diagram there are at least 2 peers between k and p, but none between q and p, the original claim is obviously false.

$$If: k - p \geq 1 \rightarrow \frac{k-p}{2} \geq 0.5$$

$$q - p = 0$$

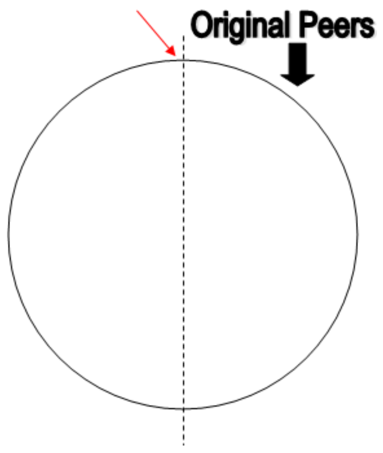
$$\therefore q - p \geq \frac{k-p}{2} \text{ is not always true}$$



i	$F_p[i] = succ(p + 2^{i-1})$
1	q
2	q
…	…
m	q

2.

a. The worst case is updating n-1 finger tables. Imagine m-bit chord architecture where the original number of peers (n-1) does not exceed 2^{m-1}. That is, no more than half the identifiers are peers. Imagine also that all the original peers are clustered together in one half of the chord ring.

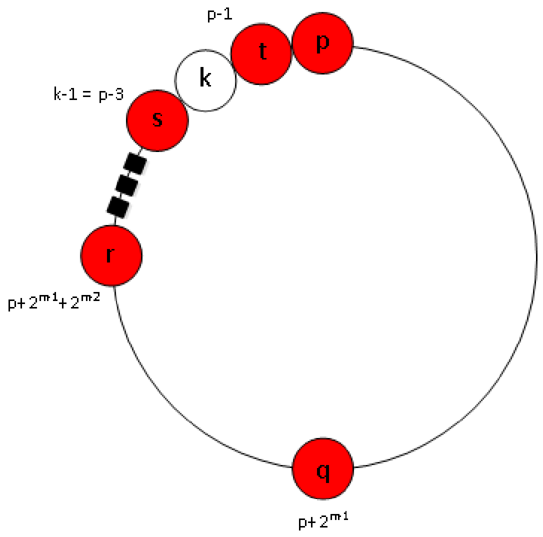


The first peer (call it “a”) on the right side will have its F[m] entry point to itself, as there are no peers with identifiers greater than or equal to a+2^{m-1} on the left side. The last peer (call it “b”) on the right side will also have its F[m] entry point to “a” due to the same reasoning.

If a new peer “c” is inserted at the red arrow, just before the border between the half with peers and the half without, at least the F[m] entries in the finger tables of “a” and “b” will have to be updated to point to “c” as $a + 2^{m-1} < b + 2^{m-1} \leq c < a$. As “a” and “b” are the first and last original peers on the right side, any peers between them will also have to update at least their F[m] entries. Thus, in this case all n-1 original peers must update their finger tables.

b. The worst case is for one peer to have to update its lookup table. This is as inserting a new peer in this case is similar to inserting into a linked list, where only the predecessor of the inserted node must update its pointer. None of the other peers need to update as their own “next” peers have not changed.

3. Imagine a chord ring with m-bit identifiers. Have “k” be located right before “t”, which in turn is right before “p”, so the maximum number of hops from p must be taken to find “k”. Say there is a peer “q” with an identifier 2^{m-1} more than “p”. This means it will be the F[m] entry in the finger table of “p”, which is used to find “k”. When looking for succ(p+2^{m-1}) for F[m], no keys will be skipped as “q” is right there.



Imagine another peer “r” which has identifier 2^{m-2} more than “q”, such that the finger table of “q” will have F[m-1]=r to forward to find “k”. Again, no keys are skipped when finding succ(q+2^{m-2}). Have each subsequent forwarding peer be spaced out in this pattern of halving distances. This means that the distance between the current forwarding node and “s”, the peer right before “k”, is halved each hop, without skipping any keys.

This means as m is equal or greater than 3, the number of hops to “s” is $\log_2(2^m-3)=m$. The last hop from “s” to “k” makes the total number of hops exactly m+1, which is linear. This proves that the worst case is at least $\Theta(m)$.

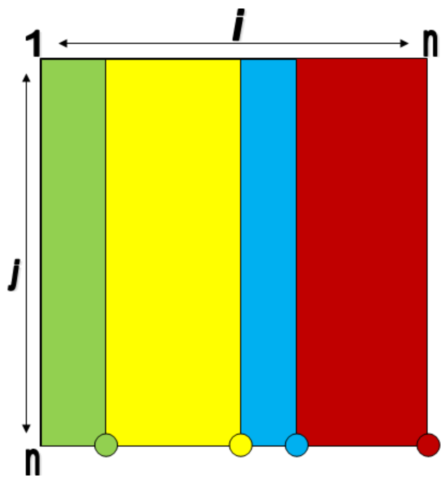
4. To map the coordination $ID\ \langle i,j\rangle$ of the CAN, the following invertible function can be used:

$$f: [1,n] \times [1,n] \rightarrow [0, n^2-1]$$
$$f: (i-1)n + (j-1)$$

For Inverse:

$$j = (f \bmod n) + 1$$
$$i = \frac{(f - (f \bmod n))}{n} + 1$$

Assuming that the mapped Chord identifiers satisfy the property that requests for an ID are routed to the successor of that ID, the corresponding topology of the 2-D CAN would be similar to the one below. The circles on the corners are the peers, which are responsible for the rectangular regions of the same colour.



5. Assume that there are “n” nodes in total (the one from whose perspective we are currently looking and the n others).

Assume that there are no exchanges of partial views. A node cannot become neighbours with more nodes outside the c in its partial view by exchanging information with its neighbours.

Assume that the neighbour relationship is not bi-directional. This is if R picked P to be its neighbour, P is considered a neighbour of R from R’s perspective, but R is not one from P’s perspective. Thus, for both R and P to be mutual neighbours to each other, they would both have to pick the other as one of its c neighbours.

As the question stated that P and Q are both neighbours of R, we assume that R initiated a unidirectional neighbour relation with P and Q, but P and Q don’t necessarily have R in their list of c neighbours, such that both P and Q have c slots available for picking their neighbours.

In the question, "one another" is a reciprocal pronoun, which means there needs to be a bi-directional relationship between P and Q. Both of the following condition needs to be satisfied:

- P needs to have a unidirectional neighbour relation with Q
- Q needs to have a unidirectional neighbour relation with P

In either of these two cases, the total number of possibilities for choosing c neighbours out of n-1 possibilities (the number of peers other than the current one) is:

$$n_t = \binom{n-1}{c}$$

If we fix one of the c neighbours to be a specific node, we still have to choose c-1 random neighbours out of the remaining n-2 possibilities:

$$n_{1fixed} = 1 \cdot \binom{n-2}{c-1} = n_{P\to Q} = n_{Q\to P}$$

As the probabilities of P choosing Q and Q choosing P are independent, we multiply the two together to find the probability of a bi-directional neighbour relationship:

$$\begin{aligned}
&Prob_{P\leftrightarrow Q} = Prob_{P\to Q} \cdot Prob_{Q\to P} \\
&= \frac{n_{P\to Q}}{n_t} \cdot \frac{n_{Q\to P}}{n_t} \\
&= \frac{\binom{n-2}{c-1}}{\binom{n-1}{c}} \cdot \frac{\binom{n-2}{c-1}}{\binom{n-1}{c}} = \left(\frac{\binom{n-2}{c-1}}{\binom{n-1}{c}}\right)^2 \\
&= \left(\frac{\frac{(n-2)!}{(c-1)!\,(n-c-1)!}}{\frac{(n-1)!}{c!\,(n-c-1)!}}\right)^2 = \left(\frac{c}{n-1}\right)^2
\end{aligned}$$

$$\therefore Prob_{P\leftrightarrow Q} = \left(\frac{c}{n-1}\right)^2$$

6. Assume that only one peer notices the original super-peer's loss and initiates the election process at the beginning.

   For the bully election system, assume ELECTION, OK, and COORDINATOR are the costs of an election, ok, and coordinator packet, respectively. The worst case occurs if the peer with the lowest process ID initiates an election. At the first step, that peer sends out n-1 election packets, which causes n-1 ok packets to be sent back. All the other n-1 peers will then send out (n-process_id) election packets of their own to all the peers with higher process IDs. All these peers will respond with an ok packet. This happens for all the peers. Afterwards, the coordinator will send out n-1 coordinator packets to the other peers. Thus, the overall cost is:

Bully: $(\sum_{k=1}^{n}(n-k)(ELECTION+OK))+(n-1)COORDINATOR$

$$= \frac{1}{2}(n-1)n(ELECTION+OK)+(n-1)COORDINATOR$$
$$= (n-1)\left(\frac{n(ELECTION+OK)}{2}+COORDINATOR\right)$$

For the ring election system, assume processID, COORDINATOR, and ELECTION are the costs of a single process identifier, a coordinator packet, and an election packet with one process identifier, respectively. The election packet will have to make n hops while carrying at least one process ID. Each hop adds one extra process ID to the election packet, until there are n-1 extra ones in the packet. Finally, the coordinator packet will need to do n hops.

Ring: $\sum_{k=1}^{n}(k-1)(processID)+n\times(ELECTION+COORDINATOR)$

$$= \frac{n(n-1)}{2}processID+n(ELECTION+COORDINATOR)$$
$$= n\left(\frac{n-1}{2}processID+ELECTION+COORDINATOR\right)$$

7. If a node has sent out an election request, let it kill all election requests attempting to pass through it if the node's ID is not in the election list and all the IDs in the list are smaller than the node ID. This will kill off all but one election request. Thus, of the peers who sent election requests, only the one with the highest process ID will have its election request live long enough to return.