CS 425 Distributed Systems, Fall 2010, University of Illinois at Urbana-Champaign

Homework 3: Solution

Out: Oct 11, 2010. Due Date: Oct 21, 2010.

Note: (1) Please hand in **hardcopy solutions that are typed** (you may use your favorite word processor). We will not accept handwritten solutions. Figures and equations (if any) may be drawn by hand. (2) Please **start each problem on a fresh sheet**, and **type your name at the top of each sheet**. (3) Homeworks will be **due at the beginning of class on the day of the deadline.** (4) Each problem has the same grade value as the others. (5) Unless otherwise specified, the only resources you can avail of in your HWs are the provided course materials (slides, textbooks, etc.), and communication with instructor/TA via newsgroup and e-mail.

Relevant Reading for this Homework: Lectures 9-14, Associated Readings.

1. You are tasked with modifying Gnutella (as discussed in class) to support the situation when *both* the requestor peer and the responder peer (the one with the file) are behind firewalls. Describe *all* modifications you would make to the basic Gnutella protocol. Clearly specify any new message types that your approach requires, the fields in these messages, and any changes to the Gnutella protocol (e.g., when to create these new message types, how to process them, etc.).

   Solution:

   In this case, an intermediate node that is not behind a firewall will create connections with both peers. More specific, the requestor sends a Push packet to responder. The Push packet will be routed through the same path used before for the query. The requestor sends the packet to its neighbor (named A) and A has to forward it further. A knows that the requestor is behind a firewall and replaces the ip address and port in the Push packet with its own ip and listening port. Then, A waits for the responder to establish a connection at the specified ip and port. When this happens, A sends a Pull packet to the requestor containing its own ip and listening port. When a connection with the requestor is established, A will forward all the data it receives from the responder to the requestor. The format of the Pull packet is exactly the same with the format of the Push packet.

2. A new peer to peer system, called Dork, is a modified version of the Chord P2P system. Like in Chord, Dork processes are arranged in a logical ring (with 2^m points). Each Dork process has exactly one successor process. However, a Dork process' finger tables are slightly different. A Dork process p's ith finger table entry points to the next clockwise process after $id(p) + i \times 2^k$, where k is a fixed integer. Thus, each Dork process has $2^{(m-k)} - 1$ finger table entries. Dork uses the same routing algorithm as Chord. Calculate, for a Dork system with N processes, the worst-case number of hops to route a query to any given key. (Hint: start with an example Dork system where each of the 2^m points has a process at it.)

   Solution:

   Let's assume, there is a process at each of the 2^m points. The finger table entries of a process divide the ring into equal sized zones, each having 2^k processes. A process can route to the beginning of any zone using just 1 hop. The worst case scenario occurs when the target process is the last process within that zone, which requires additional $2^k - 1$ hops using the successor pointers. Therefore, number of hops in the worst case = 2^k

3. You are given a problem called 2-Leader Election that has the following Safety and Liveness requirements:

   - Safety: For each non-faulty process p, p's $elected$ = of a set of two processes, OR = NULL.
   - Liveness: For all runs of election, the run terminates AND for each non-faulty process p, p's $elected$ is not NULL.

   Modify the Bully Algorithm described in lecture to create a solution to the 2-Leader Election problem. You may make the same assumptions as the Bully Algorithm, e.g., synchronous network. Briefly discuss why your algorithm satisfies the above Safety and Liveness, even when there are failures during the algorithm's execution.

   Solution:

   (Many variants possible)

   One possible solution: Nodes receiving zero or one reply in response to an $election$ message become the coordinators and send out the $coordinator$ message. Nodes receiving the $coordinator$ message set their 'elected' set only when they receive $coordinator$ message from 2 processes.

   One special case to handle is: when one of the coordinators detect that the other coordinator is failed. In that case it can instruct a non-faulty process with a lower id to initiate a new election.

4. In a group of 10 processes using the Ricart and Agrawala's algorithm for Mutual Exclusion, all the 10 processes simultaneously (in physical time) generate a request to enter the same Critical Section. What is the worst-case number of messages that the algorithm might transmit? (Calculate this number after everyone has exited their critical section.) You can assume that each process generates only the one request for the Critical Section.

   Solution:

   Each of the 10 processes will send a request to the other 9 processes. Total number of entry request messages: 10×9 = 90

   At first each process will reply to the other processes with higher priority than itself. Total messages in this stage: $\frac{9\times(9+1)}{2}$ = 45

   Once the highest priority process exits the critical section, it will reply to the 9 queued requests – which enables the process with the second highest priority enter the critical section. When it is done, it replies to its 8 queued requests and the cycle goes on.

   Therefore, the total number of messages in this stage: $\frac{9\times(9+1)}{2}$ = 45

   So, the worst case number of messages: $90 + 45 + 45$ = 180

5. N processes are arranged in a logical ring. Each process has a voting set comprised of itself and its immediate $(K-1)$ clockwise successor processes in the ring. For a given value of K, what is the largest that N can be in order for Maekawa's algorithm to satisfy Safety with the above voting sets?

   Solution:

   Process 1 has a voting set comprised of: $\{1, 2, \ldots, K\}$
   Process $(K+1)$ has a voting set comprised of: $\{K+1, K+2, \ldots, 2K\}$
   To have at least one process common in the two voting sets, Process $2K$ and Process 1 will have to be the same process. Therefore, the largest N is: $2K-1$

Note that, each process from Process 1 through Process K will have at least one process common in their voting set.

6. (Hint: For this question, you can use the Web as a resource). Distance vector routing protocols suffer from a problem called the "count to infinity" problem. Find out what this problem is (hint: look at the Wikipedia page for Distance Vector Routing). Then, describe it briefly in 3 sentences, and given an illustrated example for it.

   Solution:

   The distance vector routing protocol uses the Bellman-Ford algorithm for shortest path calculation, which does not prevent routing loops from happening. This causes the count-to-infinity problem. The core of the problem is that when X tells Y that it has a path somewhere, Y has no way of knowing whether it itself is on the path.

   Consider a subnet connected as A-B-C-D-E-F, and let the metric between the routers be 'number of hops'. Now suppose that A goes down. In the vector-update-process B notices that the route to A, which was distance 1, is down – B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down – so it tells B that A is only two jumps from C (C to B to A), which is false. This slowly propagates through the network until it reaches infinity.