Homework 3 (Failure Detection, Consensus, Peer-to-Peer, Routing, Distributed Object) - 100 Points

CS425/ECE 428 Distributed Systems, Fall 2009, Instructor: Klara Nahrstedt

Out: Thursday, October 15, **Due Date**: Thursday, October 29

Instructions: (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) Homework will be due at the **beginning of class** on the day of the deadline.

Relevant Reading for this Homework: Chapter 3

Problem 1: Failure Detection (20 Points)

Centralized heart-beating and ring heart-beating may not detect simultaneous multiple failures of processes, while all-to-all heart-beating is too expensive (in terms of messages sent per time unit). Suppose, in an **asynchronous system** that initially has ***N*** processes, you are given that at most ***N/4*** processes may crash.

(a) Design an efficient failure detector algorithm for this system. Your failure detector should satisfy completeness. Either give pseudo-code for your algorithm, or explain it using a figure.
(b) How many failure detector messages are sent by your algorithm if no failures occur?
(c) Calculate the best-case and worst-case detection times for your failure detector (hint: These are likely to occur with 1 and N/4 simultaneous failures respectively.)

Solutions:
(a) Ring-heart-beating can be modified to design a fault detector that can detect simultaneous failures, where at most N/4 simultaneous failures can occur. Processes form a ring, ordered using their id's. A process will heartbeat to its ***N/4*** clockwise successors in the ring. Completeness is satisfied since at least one of the ***N/4*** monitors of a process will be up if the process crashes, and hence its failure will be detected.

(b) $N \times (N/4)$ messages are sent each time unit, if the heart-beating period = 1 time unit.

(c) Both the best case and the worst case detection times are one time unit, assuming that the time-out is 1 time unit.

Problem 2: Distributed Graph Algorithms – Routing (20 Points)

Consider a subnet that consists of routers A, B, C, D, and E (with links having symmetrical costs, where a lower cost is more desirable).

A	
Age	
Seq	
B	5
E	3

B	
Age	
Seq	
A	5
C	8
D	2

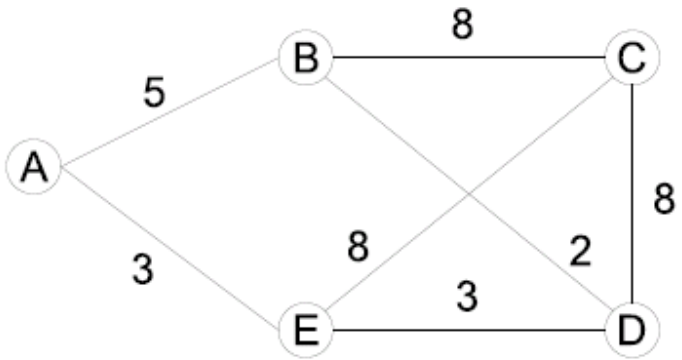
C	
Age	
Seq	
B	8
D	8
E	8

D	
Age	
Seq	
B	2
C	8
E	3

E	
Age	
Seq	
A	3
C	8
D	3

(a) Suppose the subnet uses **link state routing**. If router A receives the above link state advertisement (LSA) packets from all the other routers, **derive the network topology**. Note that an entire LSA packet contains a single age field that describes how long the information will
be valid. Each entry in the LSA packet then specifies a neighbor and the link cost to the neighbor.

Solution:



(b) Suppose the subnet uses **distance vector routing** instead, then consider the converged state, where all routes are optimal. Now, if the **link E − D goes down**, then list all the distance vector entries throughout the system that will change due to this link going down. For each changed entry, state the (old and new) next hop and the (old and
new) cost of the entry.

Solution:

Router	Destination	Old Next Hop	Old Cost	New Next Hop	New Cost
A	D	E	6	B	7
B	E	D	5	A	8
D	A	E	6	B	7
D	E	E	3	B	10
E	B	D	5	A	8
E	D	D	3	A	10

Problem 3: Consensus (20 Points)

Explain briefly why the impossibility of consensus proof (proofs of Lemmas 2 and 3 in the FLP paper) would break if the system were synchronous. Specifically, give at least one statement in the proof that may not hold in a synchronous system.

Solutions:

Lemma 3 may be violated since message transmission delays are bounded in synchronous systems, and therefore the event e = (p,m) cannot be prevented from being applied forever. Also, in case II of this lemma, the deciding run s cannot be longer than the minimum time for p to execute an instruction. Other reasonable answers are also acceptable.

Problem 4: Unstructured Peer-to-Peer Systems (20 Points)

Consider a Gnutella p2p system with 1023 total nodes that has become structured as a binary tree with height =9 (from a given root node.) What are the maximum and minimum number of nodes that can receive a Query message that is initiated with a TTL=7 from a leaf node of the tree?

Solution:

Since the tree is a full binary tree, leaf nodes are symmetric to (i.e., indistinguishable from) each other. Thus, the max and min number of messages is the same regardless of the query source. Thus, without loss of generality, assume the query starts at the leftmost leaf node – call this node **L**. Also, call its ith ancestor node as $\mathbf{L_i}$ (thus **L1** is L's parent, **L2** its parent's parent and so on).

First, the query from **L** reaches **L3** with a TTL of 5, and since L3 has a height 3 tree under it, the query will reach all descendants of **L3** (14 nodes, excluding **L**).

Second, the query from L reaches **L4** with a TTL of 4 and thus all nodes up to depth of 3 that are in the right sub-tree rooted at **L4** (total of 8 nodes, including **L4).**

Third, the query from L reaches **L5** with a TTL of 3 and thus all nodes up to depth of 2 that are in the right sub-tree rooted at **L5** (total of 4 nodes, including L5).

Fourth, the query from L reaches **L6** with a TTL of 2 and thus all nodes up to depth of 1 that are in the right sub-tree rooted at **L5** (total of 2 nodes, including L6).

Finally, the query reaches **L7** and none of its right descendants (total of 1 node).

That's a total of 14+8+4+2+1 = 29 nodes.

Problem 5: Distributed Objects (20 Points)

Consider distributed objects in the distributed systems.

(a) Specify where (component(s)) and how (protocol) does the translation between local and remote procedure/object references happen when the distributed system is using remote procedure call (RPC)?

(b) Specify where (component(s)) and how (protocol) does the translation between local and remote object references happen when the distributed system is using remote method invocation (RMI)?

Solution:

In RPC, the translation and checking if requested object (e.g., file) is local or remote happens at the client stub procedure interface.

Protocol: When a read/write procedure calls a remote object (file), the stub for the procedure, that reads/writes object, checks first if the object exists locally. If not, it calls communication module to contact server to call remote procedure to retrieve the remote object(file).

In RMI the translation between the local and remote object references happens in the "Remote Reference Module" component.

Protocol: When a local object needs to call remote object B, it contacts the proxy of object B at the client side. Proxy will access remote reference module and get information where object B resides. Once proxy for object B has the information it contacts the communication module to access server process where object B is located.