Online Resit Exam Distributed Algorithms

Vrije Universiteit Amsterdam, 1 July 2020, 18:30-22:00

By participating in this exam, I declare to understand that taking an online exam during this corona crisis is an emergency measure to prevent study delays as much as possible. I know that fraud control will be tightened and realize that a special appeal is being made to trust my integrity. With this statement, I promise to:

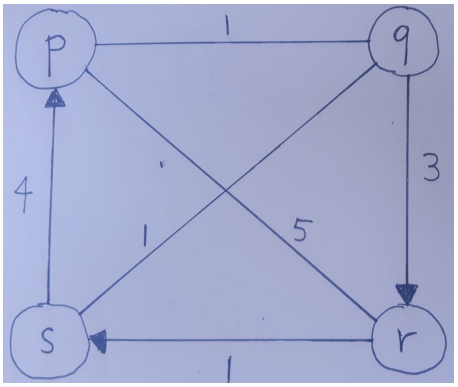
- make this exam completely on my own,
- not share my solutions with other students, and
- make myself available for any oral explanation of my answers.

(The 7 exercises in this exam sum up to 90 points; each student gets 10 points bonus.)

1. Describe in detail a computation of Tarry's algorithm on a complete, undirected network of four processes in which a spanning tree is constructed that it not a depth-first search tree. (12 pts)

   Solution: Consider a complete network with processes p_0, p_1, p_2, p_3, where p_0 is the initiator. The token travels from p_0 via p_1 to p_2, after which p_1 has parent p_0 and p_2 has parent p_1. Now, to build a spanning tree that is not a depth-first search tree, p_2 sends the token through the frond edge to p_0 (instead of to p_3), and p_0 sends the token on to p_3, which makes p_0 its parent. Next, the token travels from p_3 via p_1 back to p_3, and then via p_2 back to p_3. (Alternatively, the token could first travel from p_3 via p_2 back to p_3, and then via p_1 back to p_3.) Finally, p_3 sends the token to its parent p_0, which sends the token to p_2, which sends the token to its parent p_1, which sends the token to its parent p_0, after which the computation terminates.

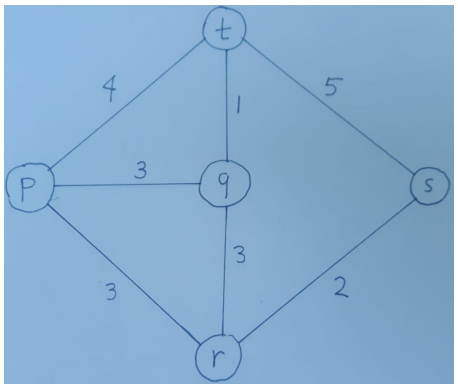
2. Consider the network depicted in example 8.2, whereby the initial sink tree is adapted by changing the parent of r from p to s.



Explain in detail why no computation of the Merlin-Segall algorithm on this adapted network computes the correct distance values in one round. (12 pts)

Solution: In round one, q only sends distance messages when it has received a message from its parent r. In turn, r sends messages only when it has receiced a message from its parent s. And in turn, s only sends messages when it has received a message from its parent p. So inevitably, s will receive a message from its parent p before it gets a message from q, meaning that inevitably s sends a suboptimal distance value to r in round one.

3. Give one possible computation of the Gallager-Humblet-Spira algorithm on the undirected network below to determine a minimum spanning tree.



Note that three channels have the same weight. To avoid deadlock, we define an ordering on these channels: $pq < pr < qr$.

During the computation, the handling of **test** messages from r and t and of a **connect** message from r should be delayed at p. (20 pts)

Solution: Since channel weights are not unique, we use channel IDs for fragment names instead.

q and t change channel qt from basic to branch and send $\langle \textbf{connect}, 0\rangle$ to each other. Moreover, p changes channel pq from basic to branch (because $pq < pr$) and sends $\langle \textbf{connect}, 0\rangle$ to q. Moreover, r and s change channel rs from basic to branch and send $\langle \textbf{connect}, 0\rangle$ to each other.

q and t receive each other's **connect** messages and merge into one fragment with name qt and level 1 by sending $\langle \textbf{initiate}, qt, 1, \mathit{find}\rangle$ to each other. Next, q receives p's **connect** message and sends $\langle \textbf{initiate}, qt, 1, \mathit{find}\rangle$ to p. Moreover, r and s receive each other's **connect** messages and merge into one fragment with name rs and level 1 by sending $\langle \textbf{initiate}, rs, 1, \mathit{find}\rangle$ to each other.

q and t receive each other's **initiate** messages and send $\langle \textbf{test}, qt, 1\rangle$ to r and p, respectively. Moreover, r and s receive each other's **initiate** messages and send $\langle \textbf{test}, rs, 1\rangle$ to p and t, respectively.

r and t receive the **test** messages from q and s, respectively, and reply with **accept**. Moreover, p receives the **test** messages from t and r and postpones replying to these messages, because the **test** messages are at level 1 while p is at level 0.

p receives q's **initiate** message and joins the fragment with name qt and level 1. Now p can reply to the **test** messages from t and r with **reject** and **accept**, respectively. When t receives p's **reject**, it sends $\langle \textbf{test}, qt, 1\rangle$ to s, which replies with **accept**. Moreover, p sends $\langle \textbf{test}, qt, 1\rangle$ to r, which replies with **accept**.

r and s receive the **accept** from p and t, respectively, and send $\langle \textbf{report}, 3, pr\rangle$ and $\langle \textbf{report}, 5\rangle$ to each other. As a result, r sends $\langle \textbf{connect}, 1\rangle$ to p, which postpones replying to this message because both the **connect** message and p are at level 1.

p sends $\langle \textbf{report}, 3, pr\rangle$ to q, which next sends $\langle \textbf{report}, 3, pr\rangle$ to t, while t sends $\langle \textbf{report}, 5\rangle$ to q. As a result, s sends $\langle \textbf{changeroot}\rangle$ to p, which then sends $\langle \textbf{connect}, 1\rangle$ to r.

Since p and r have sent $\langle \textbf{connect}, 1\rangle$ to each other, now they send $\langle \textbf{initiate}, pr, 2, \mathit{find}\rangle$ to each other. From p this message travels via q to t, and from r it travels on to s.

Finally, t and s send **test** messages to each other, leading to them to reject st, and likewise q and r send **test** messages to each other, leading to them to reject qr. So $\langle \textbf{report}, \infty\rangle$ messages flow to the core edge pr of the fragment and the computation terminates.

4. Explain why the rotating coordinator crash consensus algorithm may not terminate if it employs an incomplete, strongly accurate failure detector. (12 pts)

Solution: Let a process crash before the round in which it is the coordinator, but let this process never be suspected to have crashed by the other processes. Then the round in which the crashed process is the coordinator will never complete.

5. In the voting phase of the two-phase commit protocol, why must participants in a distributed transaction copy the tentative changes they made during the transaction to stable storage right before and not right after sending **yes** to the coordinator? (10 pts)

   Solution: Else a participant could crash after voting **yes** but before copying its tentative changes to stable storage, so that these tentative changes are lost in the crash. If all participants in the distributed transaction happen to vote **ye**, so that the coordinator decides to commit the transaction, after restarting the crashed participant, can't recover its tentative changes.

6. Give an example to show how in the Chord ring, a search for a file by a peer p may overshoot its target due to an improper *succ* value at another peer q, resulting from a recently joined peer s. Also explain how the peer r could act when it gets the request that overshoot its target. (12 pts)

   Solution: Let peer p at ID 0 in the ring search for a file with hash value 5, which is located at peer r at ID 6 in the ring. Let peer q be at ID $succ_p = 4$ in the ring, so that $finger_p[1] = finger_p[2] = finger_p[3] = q$ and let $succ_q = 6$. Peer p starts the search for the file by contacting peer q, who relays the request to peer r. In the meantime, peer s joins the ring at ID 5, finds that r is its successor, and obtains the file with hash value 5 from r. Now the request from q arrives at r, who no longer holds the requested file.

   One possibility is that r next contacts s, since r knows that the file was transferred to s. Another possibility is that r asks p to restart its query, because the stabilization procedure will restore the proper *succ* values, so that the search will be successful in a next attempt.

7. Consider the Winternitz signature scheme with $k = 11$ and $\ell = 3$. Let 01101010011 be the hash of Alice's message to Bob. Explain how Alice signs her message, taking into account the checksum, and how Bob verifies this signature. (12 pts)

   Solution: $k = 11$ and $\ell = 3$, so $n = 4$.

One 0 is padded at the left of the hash 01101010011 of Alice's message. The 4 binary substrings of length 3 that constitute the resulting string, i.e., 001, 101, 010 and 011, are binary representations of the numbers 1, 5, 2 and 3, respectively.

Alice computes as checksum $(7-1)+(7-5)+(7-2)+(7-3) = 17$, which has as binary representation 10001. One 0 is padded at the left to make the length of this string divisible by 3. The 2 binary substrings of length 3 that constitute the resulting string, i.e., 010 and 001, are binary representations of the numbers 2 and 1, respectively.

Alice generates a private key of 6 random numbers $X_1 \parallel X_2 \parallel X_3 \parallel X_4 \parallel X_5 \parallel X_6$ and publishes the corresponding public key $h(h^7(X_1) \parallel h^7(X_2) \parallel h^7(X_3) \parallel h^7(X_4) \parallel h^7(X_5) \parallel h^7(X_6))$. She signs her message with $h(X_1) \parallel h^5(X_2) \parallel h^2(X_3) \parallel h^3(X_4) \parallel h^2(X_5) \parallel h(X_6)$.

To verify the signature, Bob computes the hash of the message and the checksum, thus determining the sequence of numbers $1, 5, 2, 3, 2, 1$. He applies h^6 to $h(X_1)$, h^2 to $h^5(X_2)$, h^5 to $h^2(X_3)$, h^4 to $h^3(X_4)$, h^5 to $h^2(X_5)$, and h^6 to $h(X_6)$. Finally, he applies h to the concatenation of the 6 resulting strings and checks that the outcome coincides with Alice's public key.