COP 701
Assignment 1 - Distributed Ledger

Date: 28-Jul-2017

Instructions

The aim is to design a bitcoin system. Unlike BitCoin, we will not use cryptographic primitives, or other hashing mechanisms. You can write your code in **C/C++/Python/Java**.

- Let us assume we have a network of **N** nodes, where each node knows the identity(account number) of the rest of the nodes. ($\mathbf{N} \geq 75$)
- A node can have two states at any point in time: <in the network, online> , or <not in the network, offline>. This means that at any point of time, **M** nodes can actually be considered to be alive, where $\mathbf{M} \leq \mathbf{N}$.
- You can assume that with a random probability, a node suddenly chooses to become offline. Furthermore, it becomes offline without informing anyone (fail-stop failure mode).
- Let us define a transaction, as an agreement between three nodes: Sender, Receiver, and Witness. All three need to say commit for the transaction, for it to be actually committed. Assume that with some random probability, a node decides to say "abort". Each transaction is uniquely identified by its scalar Lamport clock: node id, local transaction number (monotonically increasing sequence).
- **Task 1**: Maintain a **Distributed Hash Table** to store public keys of nodes.(35 points)
- **Task 2**: Simulate the 2-phase commit protocol among the three nodes that want to do a transaction.(10 points)
- **Task 3**: If a transaction is successfully committed, the sender will broadcast it to all the online nodes. A transaction must contain transaction id, account numbers of all the 3 parties involved, amount to be transferred, list of input transactions (check BitCoin video for details) and a digital signature.(10 points)
- **Task 4**: On receiving the broadcast, the receiver will verify the digital signature and verify the input transactions. Once verified the node adds it to its transaction list.(5 points)
- **Task 5**: It is possible that multiple transactions are broadcast concurrently. All the nodes should perceive the same order of broadcasts (Atomic globally ordered broadcast). You can use virtual synchrony to ensure this.

At the end of simulation, all the online nodes should have the same contents of the transaction list. You can verify this by printing a hash of the contents of the transaction list. All the hashes should be the same.(20 points)

- **Task 6**: Besides this, ensure that each transaction can be used only once as an input transaction. This is needed to prevent "double spend" attacks.(20 points)

Submission Details

- Deadline : 1 September 2017
- Please ZIP all source files and then submit.
- ZIP Name : $< roll1 >_< roll2 >_< roll3 >$_CSPA1.zip.
- Please check the moodle link for submissions.

Academic Integrity Code

- Academic honesty is required in all your work. Discussion of assignments on Piazza is fine(TAs will monitor Piazza) but looking at someone else's work and then doing your is not. You must do all written and programming assignments on your own, except where group work is explicitly authorised. If you use parts of a solution or code from other sources (such as Internet etc.), you should explicitly mention it in your submission. Letting your work become available or visible to others is also cheating. The first instance of cheating will straight-away invite an 'F' grade in the course and a referral to the disciplinary committee.
- In the past years disciplinary action has been taken on students found copying.
- Moss (for a Measure Of Software Similarity) will be run and your code will be matched to check similarity with this year and previous years submissions.

References

- Explanation of Bit Coins : https://www.youtube.com/watch?v=Lx9zgZCMqXE
- Notes on Virtual Synchrony : http://privateweb.iitd.ac.in/%7Esrsarangi/docs/dist/ft.pdf
- Notes on Distributed Hash Tables : http://www.cse.iitd.ac.in/ srsarangi/csl860/docs/pastry-lec.pdf