CSC2221: Introduction to Distributed Computing **Fall 2017**

Lecture 12: Presentation

Lecturer: Multiple *Scribe: Lily Li*

This is a general comment for people who used power-point: please export your presentation as a pdf. Not everyone has a Windows PC and variants of the power-point software can really mangle your presentations.

12.1 An $O(\sqrt{n})$ Space Bound for Obstruction-Free Leader Election

1. Speaker: Jing Yao (Jason) Li
2. Summary: the algorithm uses $O(n)$ multi-reader multi-writer registers. Step complexity is $O(\log n)$ and space complexity is $O(n)$ (we used $O(n \log n)$ registers in class so how was the $O(n)$ space obtained?) Algorithm proceeds in phases using a snapshot object. Two lose conditions: either you are in a smaller phase number than someone else or you see another process has written to two cells. The algorithm is obstruction-free with $O(n^{\frac{3}{2}})$ step complexity for solo execution.
3. What did I like: the flow of the presentation was well thought out and the timing (that is: duration of the presentation and the amount of time spent talking about each sub-topic) was well balanced.
4. Easy to understand: the definitions, algorithm description, and the proof of the first two lemmas were well explain with pictures.
5. Hard to understand: the proof of the third lemma. You made some claims that you did not prove because of time constraints (fair enough), but you did not keep them on the board during the explanation so it was difficult to recall what they were.
6. Improvements: please speak up! Further, it would be helpful to have the code (and the terminating conditions) up while you are proving the lemmas.

12.2 On the Number of Objects with Distinct Power and the Linearizability of Set Agreement Objects

1. Speaker: Kevan Hollbach
2. Summary: the consensus hierarchy conjecture asks if all objects of the same hierarchy are equivalent. There are non-deterministic and deterministic (recent) counter examples when we are in consensus level 2. Consider set agreement (more general version of consensus). Require validity, k-agreement, and wait-free termination (k-agreement means that the n processes return k values). Introduced $(n, k) - SA$, this object is non-linearizable for $k > 1$. A variant, $(n, k) - LAS$, is linearizable.
3. What did I like: the topic of the presentation sparked interest in this audience member since it was related to several professors at the university. Also the explanation of set agreement was well done. The intuition for different objects helped with the understanding.
4. Easy to understand: setup was very good, it was easy to see how this result extends ideas that we learned in class regarding the consensus numbers.

5. Hard to understand: the later parts of the proof of linearization and the second result of the paper. I was not clear what the second result was claiming.

6. Improvements: since. When proving linearizability, it would help to use different colors to indicate proposals by different processes. Also, I might have miss-hear, but you mention snapshot objects at some point, but it was not clear how it was used. The road-map of the presentation you had at the beginning was good, but could have been more detailed (to reflect the amount of time you are going to spend on each topic). Finally, speak with confidence! You have a great topic that you understood well, but frequent interjections of 'ah' and 'um' were a bit distracting (at least to this audience member).

12.3 An Almost-Surely Terminating Polynomial Protocol for Asynchronous Byzantine Agreement with Optimal Resilience

1. Speaker: Gregory Rosenthal

2. Summary: polynomial protocol for asynchronous byzantine agreement. We are in an asynchronous complete message passing network with n processes ($\leq t$ Byzantine). What we are going to develop is a common coin protocol. There is a constant number of expected rounds (this why the algorithm is almost surely terminating). Secret sharing is done via polynomial interpolation. A dealer decides upon a polynomial and constructs shares of it to pass along to the other processes. To deal with a possibly faulty dealer we need to use a moderator (which also can be faulty).

3. What did I like: the speaker delivered the presentation in an audible and confident manner.

4. Easy to understand: the problem, model, and the general intuition of the common coin and its application to Byzantine agreement.

5. Hard to understand: implementation of the moderated-weak SVSS. It was quite an involved algorithm and it was difficult to keep track of everything.

6. Improvements: the presentation slides were quite text-heavy. It would be helpful if there was anyway to visualize what was going on (the interaction of the dealer, moderator, and processes).

12.4 Byzantine Vector Consensus in Complete Graphs

1. Speaker: Chaoqi Wang

2. Summary: Fully connected network with n process (of which f are Byzantine) the input is a d-dimensional real-valued vector. We are doing consensus so termination, agreement, and validity (where the agreed-upon value is in the convex hull of the inputs of the non-faulty processes). Observe that it is not possible to do consensus on each of the dimension (there was a pretty good example with four processes where one was Byzantine). In the synchronous model, the results are similar to what we achieved for consensus on a real valued input.

3. What did I like: your diagrams make the claims/ counter-examples much easier to understand.

4. Easy to understand: it was easier for me to understand the necessary condition. The pictures definitely helped here.

5. Hard to understand: the motivation for introducing the multi-set. From what I understand we need them to prove the sufficiency result, but how are the multi-sets used in the algorithm?

6. Improvements: try to speak to the audience. If you are simply reading from the slides, this reduces the amount of engagement the audience might feel (if necessary you could have a script and occasionally refer to the script).

12.5 Two-Bit Messages are Sufficient to Implement Atomic Read/Write Registers in Crash-prone Systems

1. Speaker: Ruize (Richard) Luo
2. Summary: the two bits are for the control, you still need to send the message! There is a further relaxation of the input buffers being FIFO. This is going to be a modification of the ABD algorithm for the first part. We need to deal with the sequence numbers so we do not have to send them along with the message. To do this we will keep a local sequence number variable for each process and update this as more messages are received (first assume the buffers are FIFO).
3. What did I like: the long running example of the $WRITE$ operation was very well done. It communicates the intuition of what the algorithm is doing, possible problems when we transition to the case where the input buffers are not FIFO and the modifications to the algorithms.
4. Easy to understand: the algorithm and its modifications. The purpose and implementation of the check bit was also very well explained (this was accomplished by the clean and clear presentation slides).
5. Hard to understand: noting of note.
6. Improvements: there is not much I can say here. Your presentation was well done as were your slides (those animated slides must have taken quite a bit of effort). There was some technical difficulties, but there was not too much you could have done to help with that.

12.6 Distributed Snapshots: Determining Global States of Distributed Systems

1. Speaker: Sachit Ramjee
2. Summary: Uses unidirectional FIFO channels. The difficulty occurs because we are using the asynchronous model. We will use marker messages to record states from the perspective of the process taking the snapshot. The snap-shot operation begins when a process decides to take a snap-shot (sends out a marker message N'). A processes receiving the marker will check to see if it has recorded its states and following the marker-sending rules if it has not. Pre-recording and post-recording events apparently are defined in-terms of the process recording its states. You can swap these and derive a situation where all prerecording events occur before all post-recording events.
3. What did I like: examples! These really help with the explanation of the algorithm.
4. Easy to understand: the snapshot algorithm and its associated examples.
5. Hard to understand: what is the purpose of the permutation to move the pre-recording events before the post-recording events? (it is to show that the algorithm has the desired behavior?)
6. Improvements: don't end on a blank slide. Add a summary slide or something to look at so we can look-back and see everything that you have talked about.

12.7 Viewstamped Replication: A New Primary Copy Method to Support Highly-Available Distributed Systems

1. Speaker: Yang Fang
2. Summary: We want a piece of data stored in a server to be wildly-available so we will replicate the data. We want the replicas to satisfy: initial state are all the same, determinism and coordination. This seems somewhat applied (that is applied to actual servers and clients). The presented approach is based on a primary copy technique.
3. What did I like: this is a definitely a different type of paper than the other presentations, so the novelty is nice.
4. Easy to understand: the setup was clear.
5. Hard to understand: but what are you trying to do now that you have things set-up?
6. Improvements: try to talk to the audience more. There were instances where you glanced at the audience but for the most part you were talking to the slide.

12.8 Distributed Computing Building Blocks for Rational Agents

1. Speaker: Mohammad Kidwai
2. Summary: the model is deterministic synchronous. The distributed problem that we are going to consider are: consensus, wake up, and leader election. Rational agents will take an action which maximize utility at each step. We want to identify solution properties: existence, efficiency and resilience. This paper build two basic building blocks for game theoretic distributed algorithms: wake-up and knowledge sharing.
3. What did I like: I appreciate having the model, problems, and solution properties up while you talked about the introduction.
4. Easy to understand: the transition between game theory and distributed computing. The motivation for the problem is well defined.
5. Hard to understand: the actual descriptions of the two building blocks.
6. Improvements: speak up and speak with confidence!