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

Lecture 11: Presentation

Lecturer: Multiple *Scribe: Lily Li*

11.1 A Randomized Concurrent Algorithm for Disjoint Set Union

1. Speaker: Eric Di Tomasso
2. Summary: an extension of Tarjan's talk on Disjoint Set Union. Data structure consists of operations $Unite$, $SameSet$, and $Find$. Processes are given priority (this is used in later analysis). The analysis gives a probability for the number of ancestors of each process.
3. What did I like: it is quite ambitious to go through the material that took Tarjan an hour to cover.
4. Easy to understand: definition of operations and algorithms.
5. Hard to understand: analysis (but this is due to the nature of the material covered).
6. Improvements: give some intuition of how the proof works. Some audience members might find it difficult to follow along with the equations, but might be able to ground their understanding if they had some intuition as to what was going on.

11.2 An Optimal Multi-Writer Snapshot Algorithm

1. Speaker: Renjie Liao
2. Summary: Three settings for the snapshot objects: single updater single scanner, multi-updater single scanner, multi-updater and multiple updater. This implementation is linearizable: linearization points are $X = false$ for scanner and $A[i] = v$ for updater. Multiple new instructions were introduced: Load-Linked, Stored Conditional, Validates.
3. What did I like: the color coded time line provides a good visualization of the low-level operations that are occurring. Further it was a really good idea to keep the algorithm up on a separate slide. It was useful to refer to it when I was confused.
4. Easy to Understand: the algorithm for a single-updater single scanner was still quite straight forward (there were only two functions *update* and *scan* reminiscent to the snapshot objects shown in-class).
5. Hard to understand: what was the purpose of the reorder of the for-loop on slide 10? It would be preferable if you explain the motivation for this.
6. Improvements: the code became more complex as we move through the settings and the font got smaller. Please increase the font size (or maybe simplify the code using pseudo-code conventions). Motivate the newly introduced function. What purpose do these functions serve? (This might have been one of the counter-examples, but stating it explicitly would have helped with comprehension).

11.3 The Computability of Relaxed Data Structures: Queues and Stacks as Examples

1. Speaker: Zejun (Thomas) Yu
2. Summary: focus on the queue data structure. Computability is measure by consensus number and data structure is relaxed (in terms of the queue, need output oldest element). Generalize the definition of a queue to $Q[a, b, c]$ which represent possible inserts, possible removals and possible head reads. There is quite a bit of nuance when increasing/decreasing the values of a, b, and c.
3. What did I like: examples! Seeing how the queues differ for different values of a, b, and c was quite useful for understanding.
4. Easy to Understand: the examples of $queue[a, b, c]$ for fixed a, b, and c.
5. Hard to understand: the last proof. I got lost trying to remember what you were proving.
6. Improvements: test laptop before hand to avoid technical difficulties. In this presentation in particular, it would be nice to get a "road-map" of the topics covered (the second half of the presentation was dominated by the consensus number proof and it overwhelms the results shown previously). Further, keep what you are proving on the board.

11.4 Simple and Optimal Randomizes Fault-Tolerant Rumor Spreading

1. Speaker: Ruijie Deng
2. Summary: broadcast problem in complete graph with f processes which can initially crash. Whispering and gossip-based protocols. In the fault-free case it is easy to see why we only need $O(\log n)$ rounds (only one message is passed in each round between processes). The GP algorithm proceeds by a divide and conquer-like procedure . There is also a randomized version of the GP algorithm.
3. What did I like: the results of the paper are interesting.
4. Easy to Understand: the problem and model definition.
5. Hard to understand: the steps of the GP algorithm.
6. Improvements: please speak up (the tea kettle was quite loud)! Maybe add some picture (like what you did with the tree notation) to illustrate what is going on during the complexity analysis. It is a lot easier understand circles with arrows than lists of numbers.

11.5 Deterministic Leader Election Takes $\Theta(D + \log n)$ Bit Rounds

1. Speaker: Wenyuan
2. Summary: Synchronous message passing model, but the benefit here is that the bit complexity is $O(1)$. Apparently this is because there is only seven different ways you can change your identifier. By the way all process modify their identifier by appending its length. During the algorithm, at each step its talks to its neighbors and modifiers its identifier slightly to be some sort of prefix.

3. What did I like: the idea of the paper were quite interesting.

4. Easy to Understand: main definitions and the updates once I remembered that the identifiers were modified.

5. Hard to understand: initially, when you started talking about the updates, it was really unclear as to how this contributed to solving the leader election at all.

6. Improvements: please give some intuition for the notation (page 5 $\tilde{6}$). It would have been helpful to know why you needed to pad the identifier to the length before you started talking about how to update the Z_u's. Further it might help to put the useful notation and remarks on the board when you talked about the updates just so we had something to refer to.

11.6 Distributed Detection of Cycles

1. Speaker: Zehui (Joyce) Zhou

2. Summary: Using the CONJEST model. Detection cycles using property testing (one sided error). There are two phases: in the first phase, everyone decides on an edge. In the next phase you want to check if the edge belong to some C_k. You want to broadcast all edge disjoint paths, being careful to pick particular paths since the number of paths you are allowed to send along is constant.

3. What did I like: examples! Also, the way you presented the algorithm gave me a good intuition of was going on.

4. Easy to understand: the way the algorithm choose paths to forward. (Thanks, the pictures were really helpful!)

5. Hard to understand: the complexity proof for the algorithm. There are several symbols and I was somewhat lost as to what they all represented. Further, I thought you were using bit-complexity, so it was unclear how the claim related to that.

6. Improvements: add slide numbers and (this might be a personal preference) don't use the power-point templates (you can squeeze in more information if you don't use their standard headers).

11.7 A Simple Deterministic Distributed MST Algorithm with Near-Optimal Time and Message Complexity

1. Speaker: Mengye Ren

2. Summary: The CONJEST model. There are three phases: build a forest, build a BFS, and use the BFS to assist with merging. There were also several algorithms presented: Pipeline MST, GHS (based on Boruvka's Algorithm), and their combination. The eventual algorithm has high message complexity because of the Pipeline MST. If instead we use another technique to merge the the components then you get better message complexity. Eventually we get $O(\sqrt{n}\log n)$ time complexity and $O(m + n\log n\log^* n)$ message complexity.

3. What did I like: "road-map" at the beginning of the presentation and the chart that you filled out during the presentation. This is good for keeping track of the partial results you covered throughout the presentation.

4. Easy to understand: generally, I thought the entire presentation is quite well done and easy to understand.
5. Hard to understand: nothing of note.
6. Improvements: again, the presentation is quite well done. Just a small thing: please add the reference page to the presentation (this is particularly useful here since several papers were mentioned and it was not entirely clear which was the one you read)!