Distributed Algorithms
Lecture 1

Lecture by Dr. Gadi Taubenfeld
Typeset by Steven Karas

2017-03-29
Last edited 18:17:01 2017-03-29

1 Administrative Minutiae

Attendence is not enforced, but is absolutely recommended.

1.1 Exam - 70% of grade

1.2 Homeworks - 30% of grade

3 theoretical homeworks. Done individually, not in pairs. Strictly enforced late policies. Homeworks are not necessary to take exam.

2 Course Material

2.1 Shared memory model

This section of the course will be covered through the textbook.[1]

2.2 Message passing

This section of the course will be covered through papers and notes.

[1] Synchronization Algorithms and Concurrent Programming, Gadi Taubenfeld, Prentice Hall

3 Synchronization

Two basic types of synchronization: contention and coordination. In contention, actors are contending over limited shared resources (mutual exclusion). Coordination is building consensus between actors.

Synchronziation is easy between humans, but the communication between computers is restricted: reading/writing notes; sending/receiving messages.

3.1 The Too-much-milk problem

A couple living together always want to have milk in their kitchen. They want to avoid having too much milk, because it will go bad before they drink it. An example is given on slide 5 of how too much milk can be bought if they don't communicate.

Correctness We need to satisfy these constraints to prove correctness:

- Mutual exclusion - only one actor performs the action.
- Progress - at least one actor will eventually perform the action

If you can summarize the requirements for a problem, and you're missing one or the other, then you probably defined your problem wrong.

Primitives

- Set a flag
- Read a single flag
- Remove a flag

A bad solution

```
Milk():
  leave note with my name
  if no note with other name:
    if no milk:
      buy milk
  remove note with my name
```

If both people execute this at the same time, the same steps, they will never buy milk, even if they need it, violating the principle of progress.

Another bad solution

```
Milk():
  if no note:
    if no milk:
      leave a note
      buy milk
      remote the note
```

This trivially results in a violation of mutual exclusion.

Yet another bad solution

```
leave note with A
wait while a note with B
if no milk:
  buy milk

remove note with A
```

```
leave note with B
if no note with A:
  if no milk:
    buy milk

remove note with B
```

This requires a timing assumption, and can cause a fairness issue where A is blocked indefinitely.

A correct solution Using 4 notes: A1, A2, B2, B1. The code is almost symmetric:

```
write A1
write A2 if B2 else remove A2
wait while B1 and
  ((A2 and B2) or
  (no A2 and no B2))

buy milk if no milk

remove A1
```

```
write B1
write B2 if no A2 else remove B2
wait while A1 and
  ((A2 and no B2) or
  (no A2 and B2))

buy milk if no milk

remove B1
```

If we add another actor, we can extend this algorithm further.

3.2 An atomic register

x is an atomic register, initially 0.

```
for i = 1 to 5:
  x = x + 1
```

After running two actors with this code, x can take any value between 2 and 10.

3.3 The Two-Generals Problem

First presented by Jim Gray[2]

The blue army has surrounded the red army on both sides. If the blue armies attack at the same time, they win. If they attack at different times, the red army wins.

There is no **fixed length** protocol that solves this problem. Proof in the slides.

3.4 Contention

Summary of topics that may be covered in the course (not necessarily all)

- Mutual Exclusion

[2] *Notes on Data Base Operating Systems*, 1978 pages 465-466

- Non/Blocking synchronization - concurrent data structures
- Coarse-grained locking
- Fine-grained locking - deadlocking

3.5 Coordination

3.5.1 Consensus

Covered in chapter 9 of the book.

Impossibility of consensus There is no algorithm that solves the consensus problem using atomic read/write registers in the presence of a single faulty actor. We will cover a full proof of this later in the course.

3.5.2 Barrier Synchronization

Covered in chapter 5 of the book. Set a barrier that all the actors must reach before continuing onwards.

Example: Prefix sum Slides.

3.5.3 Leader election

Not covered in the textbook. $O(n \log n)$ messages is possible.

3.5.4 Distributed graph algorithms

Minimum weighted spanning tree $O(n \log n + E)$ solution is possible. MIS, BFS, etc...

Snapshot How to learn about the global state of the network from the local state of each actor. Used for deadlock detection, termination detection, etc.

3.5.5 Renaming

Slides.

3.5.6 Clock synchronization

Trying to have multiple actors show more or less the same time on their clocks.

4 Model of computation

4.1 Relative power of communication primitives

More powerful primitives give more powerful tools.

- safe registers
- atomic registers (read/write)
- test & set
- swap

- compare and swap
- load-link/store-conditional
- read-modify-write
- semaphore
- monitor

5 Next time

Mutual exclusion