Rough Timeline for Today

• Talk about logistics and lab 0
• Raft: You should all know this better than I do.
• Quiz and midterm course evaluation.
Lab 0 and 2
Some Observations

• Nearly everyone used locks or bounded channels of length 1.
• Not the Go way to concurrency.
• Some office hour questions on Erlang, etc. -- where you apply those paradigms
• IMO harder to think about than processing everything on one thread.
• See Lab 2 starter code for an alternative.
Some Observations

• Concurrent programming: adding methods affects existing methods.

• Might necessitate adding concurrency control where none was needed.

• Might simplify concurrency control where previously needed.

• This is not an intro class: starter code exists to help you get started.

• You are responsible for ensuring no bugs in code you hand in.
Some Observations

• Half of the class did not run `gofmt` on code they handed in.

• No points docked for formatting.

• But... why? Tools exist to be used, can't help you if you don't use them.

• Similarly several people did not run `go vet` and `golint`.

• Would have picked up some build errors for those who had those.
Some Observations

• Lots of questions about how to test for concurrency.

• Few actual tests 😞

```go
code snippet here
```
Lab 2

• Went out Monday: not sure how many people have had a chance to look.

• Please try -- if possible -- to use the structure in the starter code.

• Please ask if something in the lab write up is unclear.
  • Not sure anyone has read it.
  • I could not understand it yesterday.

• Start early: there are subtleties here, need time to get it right.
Survey
What is the Problem?

In what order should these commands be run?

1. Client: set("s1", ...)
   get("s0") -> ...
2. Client: set("s0", ...)
   get("s1") -> ...
3. Client: set("s2", ...)
   get("s1") -> ...
4. Client: set("s1", ...)
   get("s0") -> ...
5. KV-Store: set("s1", ...)
6. KV-Store: set("s0", ...)
7. KV-Store: set("s2", ...)
8. KV-Store: set("s1", ...)

Sequential order of commands:

1. Client: set("s1", ...)
2. Client: get("s0") -> ...
3. Client: set("s0", ...)
4. Client: get("s1") -> ...
5. Client: set("s2", ...)
6. Client: get("s1") -> ...
7. Client: set("s1", ...)
8. Client: get("s0") -> ...
9. KV-Store: set("s1", ...)
10. KV-Store: set("s0", ...)
11. KV-Store: set("s2", ...)
12. KV-Store: set("s1", ...)

However, the specific order depends on the implementation details of the KV-Store and the use case.
A Possible Solution

![Diagram of KV-Store interactions]

- KV-Store: `set("s1", 42)`
- Client: `set("s1", 25)`
- KV-Store: `set("s1", 1729)`
- Client: `set("s1", 42)`
- Client: `set("s1", 25)`
- KV-Store: `set("s1", 1729)`
- Client: `set("s1", 1729)`
A Possible Solution

![Diagram showing KV-Store and Client interactions with set operations on key-value pairs.]

- KV-Store
  - set("s1", 25)
  - set("s1", 42)
  - set("s1", 1729)

- Client
  - set("s1", 25)
  - set("s1", 42)
  - set("s1", 1729)
A Possible Solution

set("s1", 42)

Client

set("s1", 25)

KV-Store

set("s1", 42)

Client

set("s1", 42)

set("s1", 25)

set("s1", 1729)

KV-Store

set("s1", 1729)

Client

set("s1", 1729)
Raft Observation

- Paxos originally explained as a single decree leaderless protocol.
- Leader added for efficiency, e.g., in MultiPaxos from last lecture.
- Reduces conflicts. Reduces number of messages in the common case.
- But leader election is independent of actual protocol.
- Raft combines these two processes in one protocol.
  - Leader election responsible for ensuring a good leader.
  - Consensus is just leader deciding on ordering and replicating decision.
Raft Log

Entries with the same term from the same leader.

Terms should not go back

- Primitive: Maintain a consistent log across participants
### Raft Log

<table>
<thead>
<tr>
<th>Index</th>
<th>Term</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>set(x, 5)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>set(x, 6)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>get(x)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Term</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>set(x, 5)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>set(x, 6)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>set(z, 2)</td>
</tr>
</tbody>
</table>

Two entries with the same index and term are identical.
Raft Log

<table>
<thead>
<tr>
<th>Index</th>
<th>Term</th>
<th>Command</th>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>set(x, 5)</td>
<td>0</td>
<td>set(x, 6)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>set(x, 6)</td>
<td>2</td>
<td>get(x)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>get(x)</td>
<td>3</td>
<td>set(y, 6)</td>
</tr>
</tbody>
</table>

If two logs agree on an entry, they agree on the prefix.
Raft Timeline

Leader Election
Reign
Leader Election

Term
Raft within a Term

\[\text{set}(x, 5)\]

\[\text{set}(x, 6)\]
Raft within a Term

How should the followers respond?
Raft within a Term

What can the leader infer from this interaction?
Raft within a Term

\[
\begin{array}{c|c}
0 & 1 \\
0 & 0 \\
\text{set}(x, 5) & \text{set}(x, 6) \\
\end{array}
\]

\[
\begin{array}{c|c}
0 & 1 \\
0 & 0 \\
\text{set}(x, 5) & \text{set}(x, 6) \\
\end{array}
\]

\[
\text{AppendEntries}(\text{term}=-1, \text{prevIdx}=-1, \text{prevTerm}=-1, [(0,0, \text{set}(x, 5)), (1,0, \text{set}(x,6))] )
\]
Raft within a Term

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>set((x, 5))</td>
<td>set((x, 6))</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>set((x, 5))</td>
<td>set((x, 6))</td>
</tr>
</tbody>
</table>
Raft within a Term

0 1
0 0
set(x, 5) set(x, 6)

0 1
0 0
set(x, 5) set(x, 6)

0 1
0 0
set(x, 5) set(x, 6)
Log entries can be deleted.
When is it safe to execute commands?
Need to ensure some log entries are never deleted.
Safety Through Leader Election

- Want to make sure some set of entries always remain in the log.

- Can only execute actions for such committed entries.

- Need to ensure this without communicating with former leader(s).

- Leaders might die or disappear.

- How to do this?
Safety Through Leader Election

- Make use of quorum intersection.
- How?
Leader Election

- set(x, 5)
- set(x, 6)

RequestVote(term=1, lastLogIndex = 1, lastLogTerm = 0)

How should the followers respond?
Leader Election

How should the followers respond?

RequestVote(term=2, lastLogIndex = 2, lastLogTerm = 0)

```
Leader Election

How should the followers respond?

RequestVote(term=2, lastLogIndex = 2, lastLogTerm = 1)
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Figure 8

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Is this stable?
Figure 8

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Is this legal?
Figure 8
Configuration Change
Why?

• Want to be able to change the set of servers.
  • Take down servers for maintenance.
  • Add new servers to replace failed ones.
  • Other reasons.
How?

- Use a **special** log message which contains the set of servers.
- Use Raft to replicate this to everyone.
How Special?

• All peers use configuration as soon as logged.
• Why safe?
  • We know how to revert this change.
## Protocol

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

*Note: The table represents the protocol with each row indicating a set of actions. The numbers represent different states or steps in the protocol.*
What happens now?
### Protocol

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>C-all</td>
</tr>
</tbody>
</table>

**Note:** The diagram illustrates a protocol with steps involving setting values to specific positions. The shaded cells indicate the progression of the protocol.
### Protocol

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>C-all</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>C-all</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>C-all</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>C-all</td>
</tr>
</tbody>
</table>

0 indicates that the protocol is valid, while 1 indicates it is invalid.
Protocol

0 | 1 | 2 | 3 | 4 | 5
---|---|---|---|---|---
0 | 0 | 0 | 0 | 0 | 0
set(x, 5) | set(x, 6) | ... | ... | ... | C-all

0 | 1 | 2 | 3 | 4 | 5
---|---|---|---|---|---
0 | 0 | 0 | 0 | 0 | 0
set(x, 5) | set(x, 6) | ... | ... | ... | C-all

0 | 1 | 2 | 3 | 4 | 5
---|---|---|---|---|---
0 | 0 | 0 | 0 | 0 | 0
set(x, 5) | set(x, 6) | ... | ... | ... | C-all

0 | 1 | 2 | 3 | 4 | 5
---|---|---|---|---|---
0 | 0 | 0 | 0 | 0 | 0
set(x, 5) | set(x, 6) | ... | ... | ... | C-all
## Protocol

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>set(x, 5)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>C-all</td>
<td>0</td>
</tr>
<tr>
<td>set(x, 6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>C-new</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>set(x, 5)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>C-all</td>
<td>0</td>
</tr>
<tr>
<td>set(x, 6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>C-new</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>set(x, 5)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>C-all</td>
<td>0</td>
</tr>
<tr>
<td>set(x, 6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>C-new</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>set(x, 5)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>C-all</td>
<td>0</td>
</tr>
<tr>
<td>set(x, 6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>C-new</td>
<td>0</td>
</tr>
</tbody>
</table>
Quiz