Consensus

The consensus problem (crudely speaking, reaching agreement) is the mother of all (well, most) problems in distributed computing. Examples are

- Leader election
- Mutual exclusion
- Decision to commit or abort in transactions
- Clock synchronization

The problem becomes tricky if some of them are faulty. We are interested in this non-trivial version.
An impossibility result in distributed consensus

FLP Theorem (due to Fischer, Lynch, Patterson):
In asynchronous distributed systems, it is impossible to design a consensus protocol that will tolerate the crash failure of even a single process

Keep in mind a story problem . . .

Seven members of a busy household decided to hire a cook, since they do not have time to prepare their own food. Each member of the household separately interviewed every applicant for the cook’s position. Each member formed his or her independent opinion "yes" (means “hire”) or "no" (means “don't hire”). These members will now have to communicate with one another to reach a uniform final decision about whether the applicant will be hired. The process will be repeated with the next applicant, until someone is hired.
**Bivalent and Univalent States**

A decision state is *bivalent*, if starting from that state, there exist at least two distinct executions leading to two distinct decision values 0 or 1. Otherwise the state is univalent.

**Example**

In a 5-set tennis match, the state after 6-4, 3-6, 7-6 is bivalent, but the state after 6-4, 6-2, 7-6 is univalent.

**Lemma** No execution can lead from a 0-valent state to a 1-valent state or vice versa.
Lemma Every consensus protocol must have a bivalent initial state.

Proof by contradiction.

Assume that all states are univalent.

<table>
<thead>
<tr>
<th>State</th>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0-valent</td>
</tr>
<tr>
<td>State 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>State 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>State 0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>State 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1-valent</td>
</tr>
</tbody>
</table>

Let two consecutive rows differ in the k\textsuperscript{th} position only, such that the first row is 0-valent and the next row is 1-valent. If process k fails, then both states must lead to the same decision, which is impossible.
**Lemma.** In a consensus protocol, starting from any initial bivalent state $I$, there must exist a reachable bivalent state $T$, such that every action taken by some process $p$ in state $T$ leads to either a 0-valent or a 1-valent state.

Can action 0 and action 1 be taken by *two distinct* processes $p$ and $q$? We argue that it is not possible.
**Case 1.** *Action 0 = read, action 1 = write*

Action 0 does not influence action 1. So action 1 is also feasible in $\mathsf{T}_0$.

$\mathsf{e}_0$ = a computation in which $p$ crashed after action 0

$\mathsf{e}_1$ = a computation that excludes $p$. 

---

(a) 

(b)
Note that e0 and e1 are indistinguishable to the outside world, but the final outcomes are different! This is impossible, so p=q.

**Case 2.** Action 0 = read, action 1 = write (on the same variable).

Let p write first. Then this value will be overwritten by the value written by q.

e0 = a sequence in which p writes and then fails
e1 = a sequence in which q writes and the rest of the sequence does not include p

The decisions are different, but these are indistinguishable to the outside world! So p = q.

Call p a *decider process*.

The impossibility result follows from this (but not trivial) - we will argue in the class.