Byzantine Agreement Redux (TvS 7.2.3)

- Complicated stuff, explanation series #2….
- Recall if
  - Independent failures (no collusion/conspiracies)
  - No two-faced behavior
  Then can
  - Vote on 2k+1 replies/values
  - Tolerate k bad values
- Harder if either condition not true
- Generic goal:
  - Have all non-faulty processes reach consensus
  - Do it in a finite number of steps
Baseline Problem

• Easier base case: “two-army problem”
  – Red army in valley
  – 2 Blue armies camped on hillsides (Blue1 and Blue2)
  – Blue1 and Blue2 must agree on one bit (attack/retreat)
  – Complication: unreliable message delivery: courier can be captured
    • l.e., omission failure of link

• Bottom line
  – After any step, Blue1 and Blue2 cannot be sure the other got the last message
  – Ergo, if that step was needed in the protocol, one cannot finalize the agreement
  – Intuitive inductive argument shows agreement can never be reached.
Harder Problem

• Assumptions
  – Communication is perfect
  – Processes (generals) are not perfect
  – “Byzantine Generals problem”

• Definition
  – Red army is still encamped in valley
  – \{Blue(1), \ldots, Blue(N)\} armies camped in hills
  – Communication is point-point and perfect
    • Maybe telephone, or radio with no jamming
  – M of N generals are traitors (faulty) and try to prevent the (N-M) loyal (correct) generals from correct agreement
  – Problem: algorithm where correct generals reach agreement
Harder Problem (cont)

• Generalize the problem
  – Each general knows his/her troop strength

• Goal: exchange troop strengths so when done
  – Each general has vector of N troop strengths
  – General(i) loyal $\rightarrow$ all loyal generals have correct #Blue(i)
  – General(i) traitor: undefined for loyal and traitor generals

• Recursive algorithm from Lamport et al 1982
Example where $N=4$ and $M=1$, takes 4 steps:

1. Every general sends a (reliable) message to every other general with its troop strength (a)
2. Each general sends the others vector received in #1 (b)
3. Each general sends others vectors received in #2 (c)
4. Generals vote; correct ones decide (1,2,UNKNOWN,4)
### Harder Problem (cont.)

<table>
<thead>
<tr>
<th></th>
<th>Step Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Got(1, 2, x, 4)</td>
</tr>
<tr>
<td>2</td>
<td>Got(1, 2, y, 4)</td>
</tr>
<tr>
<td>3</td>
<td>Got(1, 2, 3, 4)</td>
</tr>
<tr>
<td>4</td>
<td>Got(1, 2, z, 4)</td>
</tr>
<tr>
<td></td>
<td><strong>1 Got</strong> (1, 2, y, 4)</td>
</tr>
<tr>
<td></td>
<td><strong>2 Got</strong> (a, b, c, d)</td>
</tr>
<tr>
<td></td>
<td><strong>3 Got</strong> (1, 2, z, 4)</td>
</tr>
<tr>
<td></td>
<td><strong>4 Got</strong> (i, j, k, l)</td>
</tr>
</tbody>
</table>

(b) \( (1, 1, 1, a, e, 1, b, f, 2, 2, 2, c, g, y, z, z, k, 4 4 4 d h 4 4 l) \)

(c) \( ( ( (1, 1, 1, a, e, 1, b, f, 2, 2, 2, c, g, y, z, z, k, 4 4 4 d h 4 4 l) ) ) \)

Steps again

3. Each general sends others vectors received in #2 (c)
4. Generals vote; correct ones decide (1,2,UNKNOWN,4)

- Algorithm generalizes to more than \( N=4, M=1 \) recursively
  - Will cover when we go through the paper(s), Lamport et al 1982…
Harder Problem (cont)

- Consider case N=3, M=1 & same algorithm:

- Lamport et al 1982 proof: with M faulty processes, need (2M+1) correct ones to reach agreement
Harder Problem (cont.)

- Intuition: need to achieve a majority vote among loyal generals
- Need to ensure that
  - Vote with M traitors,
  - And any loyalists misled (temporarily confused) by traitors
  - Still adds up to the majority vote of the loyalists
- Can only ensure this when >2/3 of votes same
- I.e., if >2/3 of generals agree on same decision, must be the same majority vote by the loyal generals
- M=3, N=(3M+1)=10 ....
  - 3 traitors, detected as such
  - 3 loyal but misled generals
  - Leaves 4 to outvote the 3 misled loyal generals