CS 138: Byzantine Consensus
Byzantine Generals Problem

Commanding General

Lieutenant General

Lieutenant General

Commanding General

Attack!

Attack!
Byzantine Generals Problem

Commanding General

Retreat!

Lieutenant General

Retreat!

Lieutenant General

Lieutenant General
Byzantine Generals Problem

• C1: All loyal lieutenant generals obey the same order
• C2: If the commanding general is loyal, then every loyal lieutenant general obeys the order she sends
Byzantine Agreement Problem

• All generals co-equal
  – each general $i$ has a value $v(i)$ she sends to the others

1) Every loyal general must obtain the same information $v(1), \ldots, v(n)$

2) If the $i$th general is loyal, then the value she sends must be used by every loyal general as the value of $v(i)$
Byzantine Generals Problem

Attack!

Retreat!
Byzantine Generals Problem

She said

Retreat!

Attack!

She said

Retreat!

Attack!
Byzantine Generals Problem

She said
Attack!

She said
Retreat!

Copyright © 2017 Thomas W. Doeppner. All rights reserved.
Summing Up

• Byzantine Generals Problem with 3 Generals, at most one of whom is a traitor ([3,1]BGP)
  – no solution satisfying C1 and C2
Message-Passing Assumptions

• Every message sent is delivered correctly
• The receiver of a message knows who sent it
• The absence of a message can be detected
[4, 0] Byzantine Generals Problem

Attack!

Attack!

Attack!

Attack!
[4,1] Byzantine Generals Problem

Attack!

Retreat!

She said
Attack!

She said
Retreat!

She said
Attack!

She said
Retreat!

She said
Attack!
[4,1] Byzantine Generals Problem

Attack!

She said Attack!

Retreat!

Attack!

She said Retreat!

Attack!

She said Attack!

Attack!

She said Attack!

Retreat!
Some Details

• Each general receives messages $u$, $v$, and $w$ from the others
  – if no message is received, interpret its lack as “retreat”
• Loyal general takes its order to be $\text{majority}(u, v, w)$
  – if no majority: retreat
Summing Up

- Byzantine Generals Problem with 4 Generals, one of whom is a traitor ([4,1]BGP)
  - solvable
Theorem

• If $N$ is the number of generals and $T$ is the number of traitors, then there is a solution to the Byzantine Generals Problem iff

\[ N > 3T \]
Proof

• Only if:
  – assume a solution exists for \( N \leq 3T \)
    - 3T Albanian generals can cope with \( T \) traitors
  – three Byzantine generals now take advantage of the Albanian approach to solve \([3,1]\)BGP
    - commander simulates Albanian commander plus at most \( T-1 \) lieutenant generals
    - two lieutenant generals each simulate at most \( T \) Albanian lieutenant generals
  – loyal Byzantine generals simulate loyal Albanians
  – traitorous Byzantine general does whatever it takes to mess things up
    - effectively simulates actions of up to \( T \) traitorous Albanian lieutenant generals
Proof (Continued)

• By C1: all loyal Albanian lieutenant generals obey same order
  – thus loyal Byzantine lieutenant generals obey orders obeyed by simulated Albanians

• By C2: if Albanian commander is loyal, then all loyal Albanian lieutenant generals obey her order
  – thus if Byzantine commander is loyal, her order is that of Albanian commander
Albanian Simulation
Proof (Half Done)

• This gives us a method to solve \([3,1]BGP\)
  – which can’t be done …
Proof (remainder)

• If:
  – Show that a solution exists if \( N > 3T \)
    - \( T=1 \)
      • done
    - \( T>1 \)
      • hard
    – next few slides
[7,2]BGP

- Case 1: the commander is loyal
  - six lieutenants receive order \( v \)
  - four report it to one another correctly
  - two (traitors) do not
  - correct outcome determined by majority
  - (that was easy!)
[7,2]BGP (continued)

- Case 2: the commander is a traitor (and so is someone else)
  - not so easy …
  - if the commander is a traitor, there is only one traitor among the lieutenants, so they can work out agreement assuming only one traitor
    - this is the Byzantine agreement problem, which means each lieutenant runs the algorithm
The Algorithm, part 1

• BGP(0)  // no traitors
  1) the commander sends her value to each lieutenant
  2) each lieutenant uses the value he receives from the commander
The Algorithm, part 2

• BGP(m) // m traitors

  1) the commander sends her value to each lieutenant

  2) for each $i$, let $v_i$ be the value lieutenant $i$ receives from the commander. Lieutenant $i$ acts as the commander in BGP(m-1) to send $v_i$ to each of the $n-2$ other lieutenants

  3) for each $i$ and each $k \neq i$, let $v_k$ be the value lieutenant $i$ received from lieutenant $k$ in step 2 (using BGP(m-1)). Lieutenant $i$ uses the value $\text{majority}(v_1, \ldots, v_{n-1})$
[7,1]: Commander is Loyal

C

\[ v_0 \]

L1

1v_0^0

L2

2v_0^0

L3

3v_0^0

L4

4v_0^0

L5

5v_0^0

L6

6v_0^0
[7,1]: Commander is a Traitor
[7,2]: ... and a Lieutenant is a Traitor
A Better (?) Algorithm …

BGP(m, gens, v, path, sender) {
    if (m > 0) {
        // tell others what was received
        for each g in gens−me
            sendmsg(g, BGP, m−1, gens−me, v, me·path, me)
        // wait till all resulting communication is complete
        when defined((∀g∈gens) me v g·path)
        // compute consensus value
        me v sender path = majority(v, (∀g∈gens) me v g·path)
    } else {    // m == 0
        me v sender path = v
    }
}
Complexity

• How expensive is the algorithm for BGP?
  – T+1 rounds of messages
  – $O(N^T)$ messages, for $N$ generals and $T$ traitors

• Can we do better?
  – T+1 rounds are required
  – polynomial algorithm exists, but for $N > 4T$
    - next few slides …
An Even Better Algorithm

• Agreement on one of two values
• $T$ traitors; $T+1$ phases; $N > 4T$
• In each phase, a different general is the commander
  – all generals broadcast values to one another
  – recipients determine “majority”
    - commander’s value is tie-breaker
• In at least one phase, the commander is loyal
  – consensus reached in this phase
  – doesn’t change in subsequent phases
Details

for (phase = 1; phase <= T+1; phase++) {
    // round 1: executed by each general
    broadcast value to all others
    await value \(v_j\) from each general \(G_j\)
    majority = value that occurs > N/2 times
    default value otherwise
    mult = number of times majority occurs
// round 2: executed by each general
if (this is $G_{\text{phase}}$)
   // $G_{\text{phase}}$ is (temporary) commander
   broadcast majority to all other generals
else
   receive tiebreaker from $G_{\text{phase}}$
   if (mult > N/2 + T)
      value = majority       // super majority
   else
      value = tiebreaker
}
Correctness

• Assume commander in phase \( p \) is loyal
  – its value \( x \) (from round 1) is either majority or default value
  – it broadcasts \( x \) in round 2
• Claim 1: all loyal generals (including phase \( p \) commander) agree on value
  – proof: soon
• Claim 2: if all loyal generals agree on value at beginning of phase \( i \), they agree at end of phase \( i \)
  – proof: soon
• After phase \( T+1 \), all loyal generals agree
Claim 1

• All loyal generals (including phase p commander) agree on value
  – consider all pairs of loyal lieutenants $G_i$ and $G_k$
  – they can set their values in one of three ways:
    - both set their value to the (super) majority
      • super majority must involve more than $n/2$ loyal lieutenants
      • any two such majorities must have a member in common
        – thus $G_i$ and $G_k$ have same value
      • $G_p$ must have heard from same majority
        – it also has same value
Claim 1 (continued)

- both set their value to the commander’s tie-breaking rule
  - since commander is loyal, both now agree with commander
- $G_i$ sets value to (super) majority; $G_k$ to tie-breaking rule
  - since super majority agrees with $G_i$, more than $n/2$ loyal nodes agree, thus $G_p$ agrees
  - $G_p$ value is adopted by $G_k$
    - i.e., this case is same as first case
Claim 2

- If all loyal generals agree on value at beginning of phase i, they agree at end of phase i
  - all generals receive consensus value from a majority of others in round 1
  - thus all loyal generals stay with this value in round 2
Complexity

• T+1 phases
• n·(n-1) messages in round 1
• n-1 messages in round 2
Signed Messages

Attack!

Attack!

Attack!

Attack!

Attack!
Signed Messages

Attack!

Retreat!

Attack!

Retreat!

Attack!
Failures

He said
Attack!

He said
Attack!

He said
Attack!

He said
Attack!

He said
Attack!

Attack!
Asynchronous Communication

- Processes may respond to messages at arbitrary times
  - can’t use timeouts to determine failures
- BGP has no solution
  - non-responding general might respond at any time with whatever response counters the decision made assuming it was missing
  - in practice this is surmountable
Surmounting Failure

• Recover quickly
  – state kept in non-volatile memory
• Detect failure
  – enforced timeouts
• Be unpredictable
  – randomized algorithm