Byzantine Generals: Specification

- A system consists of a single general and N commanders. (The commanders and the general are different).
- There are M faulty officers among the general and her commanders. The general may or may not be faulty.

1

Problem Specification

- The general gives a command. The command is either "attack" or "retreat." For our purposes a command is a Boolean value.
- A non-faulty general sends the same command to all commanders.
- A faulty general may send different commands to different commanders, e.g., send "attack" to two commanders and "retreat" to the others.

2

Problem Specification

- A general's command is either *attack* or *retreat*.
- Propose a protocol by which:
  1) **Consensus**: All non-faulty commanders attack or all non-faulty commanders retreat.
  2) **Correctness**: All non-faulty commanders obey a non-faulty general.

3

Change in model of distributed system

- A message is in flight for at most some bounded time T.
- This allows agents to operate in a semi-synchronous fashion using a number of "rounds."
- If an agent x does not receive a message from another agent y by the end of a round r (where r = 0, 1, 2, ….) then agent y did not send a message to agent x at the beginning of round r.

4

Why do we need this *significant* change to our model of distributed systems?

- Because consensus may never be reached if there is even one faulty process.
- (Fischer, Lynch, Patterson FLP theorem)
- More about this later.

5

Faulty Commanders: Forging Messages

- Consider two cases:
  1. A faulty commander can forge messages and signatures
  2. A faulty commander cannot forge messages and signatures
- In the forgeable case: a faulty commander may get an "attack" message from the general, and send a forged copy of the message saying "retreat" to some officers, and send true copies of the message to others.

6

Simple Case

- Four officers (3 commanders and 1 general)
- At most one faulty officer.
- Consider case where messages can be forged by faulty officers.

7

Simple Case: Assumptions

- Synchronous messages, or "rounds"
- Every message sent by a non-faulty officer u to a non-faulty officer v on round r is received by v in round r without error.

8

Solution to Simple Case: First Round

- Multiple rounds of messages.
- First round: general sends commands to all commanders
  - Non-faulty general sends same command to all commanders.
  - Faulty general may send different commands to different commanders.

9

Solution to Simple Case: Second Round

- Second round: Each commander sends copies of the message it received from the general to all the other commanders. Copies may be forged.
- Treat no value received from an officer as a default value – say "retreat."

Solution to Simple Case: Third Round

- Each commander takes the majority value of the messages that it received from the others, i.e., from the general on the first round and from the other two commanders on the second round.
- Each commander then commits to the majority value that it received.

Will This Strategy Work?

- Each non-faulty commander's decision is based on the majority of values it receives.
- The general's decision is her initial decision.

Try to Prove Correctness of Strategy

Consider two cases:

1) General is non-faulty
   - So at most one of the commanders is faulty
2) General is faulty
   - So all commanders are non-faulty

Case: General is Non-Faulty

- All commanders get same message, say *attack.*
- The non-faulty commanders (at least 2 of them) exchange the message, say *attack.*
- Each non-faulty commander has the same message from the general and from at least one other non-faulty commander.
- So, it doesn't matter what the faulty commander does.

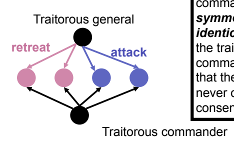
Case: General is Faulty

- Each commander has a message: either *attack* or *retreat*.
- All commanders are non-faulty. They take the majority of the messages they receive.
- Since all commanders are non-faulty they compute exactly the same function (determining majority) on exactly the same data (the set of commands they received from the general), and hence obtain exactly the same output.

Question: For the forgeable case

- Will the same algorithm work with a total of 3t officers (including the general) and t faults? For example, 6 officers – general and 5 commanders – with 2 faulty and 4 non-faulty officers?

The 3t officers case

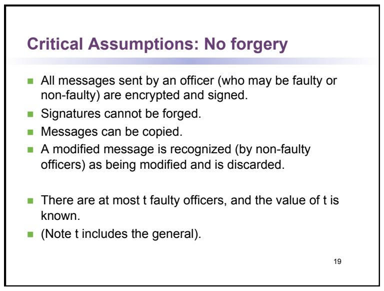


Splitting the non-faulty commanders into ***two symmetric groups of identical size***, allows the traitorous commander to ensure that the two groups never come to a consensus.

Arbitrarily Many (*t*) Faulty Officers: Algorithm Using Signatures

All processes know t

No forgery

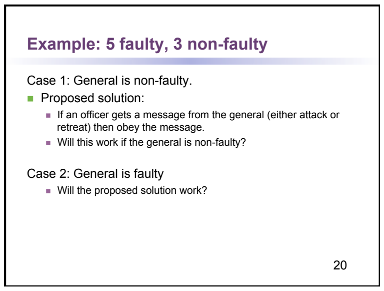


Critical Assumptions: No forgery

- All messages sent by an officer (who may be faulty or non-faulty) are encrypted and signed.
- Signatures cannot be forged.
- Messages can be copied.
- A modified message is recognized (by non-faulty officers) as being modified and is discarded.

- There are at most t faulty officers, and the value of t is known.
- (Note t includes the general).

19

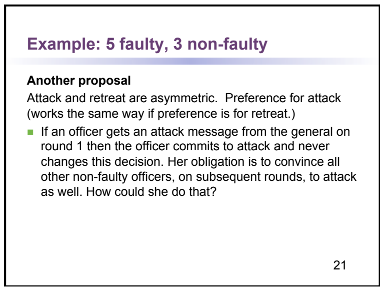


Example: 5 faulty, 3 non-faulty

Case 1: General is non-faulty.
- Proposed solution:
  - If an officer gets a message from the general (either attack or retreat) then obey the message.
  - Will this work if the general is non-faulty?

Case 2: General is faulty
  - Will the proposed solution work?

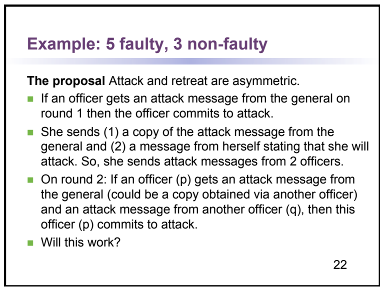
20



Example: 5 faulty, 3 non-faulty

Another proposal
Attack and retreat are asymmetric. Preference for attack (works the same way if preference is for retreat.)
- If an officer gets an attack message from the general on round 1 then the officer commits to attack and never changes this decision. Her obligation is to convince all other non-faulty officers, on subsequent rounds, to attack as well. How could she do that?

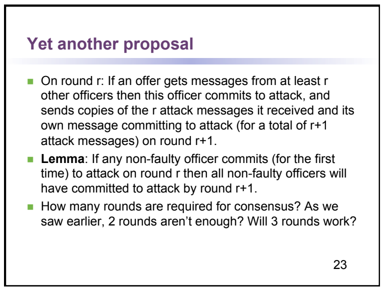
21



Example: 5 faulty, 3 non-faulty

The proposal Attack and retreat are asymmetric.
- If an officer gets an attack message from the general on round 1 then the officer commits to attack.
- She sends (1) a copy of the attack message from the general and (2) a message from herself stating that she will attack. So, she sends attack messages from 2 officers.
- On round 2: If an officer (p) gets an attack message from the general (could be a copy obtained via another officer) and an attack message from another officer (q), then this officer (p) commits to attack.
- Will this work?

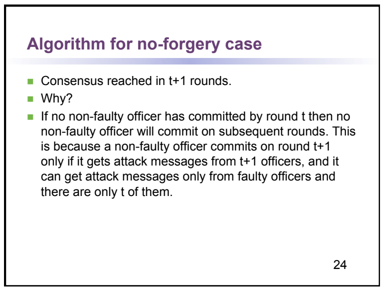
22



Yet another proposal

- On round r: If an offer gets messages from at least r other officers then this officer commits to attack, and sends copies of the r attack messages it received and its own message committing to attack (for a total of r+1 attack messages) on round r+1.
- **Lemma**: If any non-faulty officer commits (for the first time) to attack on round r then all non-faulty officers will have committed to attack by round r+1.
- How many rounds are required for consensus? As we saw earlier, 2 rounds aren't enough? Will 3 rounds work?

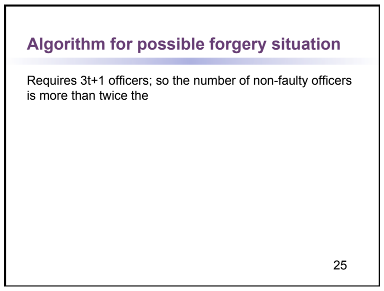
23



Algorithm for no-forgery case

- Consensus reached in t+1 rounds.
- Why?
- If no non-faulty officer has committed by round t then no non-faulty officer will commit on subsequent rounds. This is because a non-faulty officer commits on round t+1 only if it gets attack messages from t+1 officers, and it can get attack messages only from faulty officers and there are only t of them.

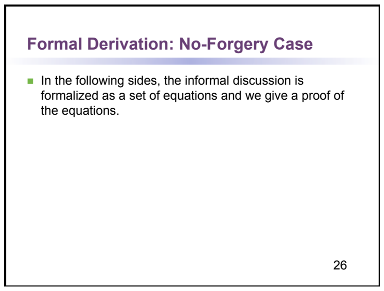
24



Algorithm for possible forgery situation

Requires 3t+1 officers; so the number of non-faulty officers is more than twice the

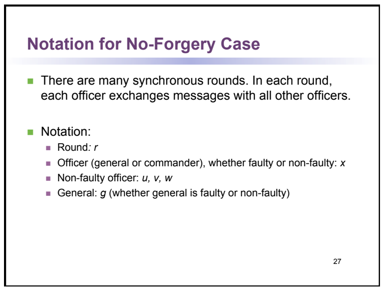
25



Formal Derivation: No-Forgery Case

- In the following sides, the informal discussion is formalized as a set of equations and we give a proof of the equations.

26



Notation for No-Forgery Case

- There are many synchronous rounds. In each round, each officer exchanges messages with all other officers.

- Notation:
  - Round: *r*
  - Officer (general or commander), whether faulty or non-faulty: *x*
  - Non-faulty officer: *u, v, w*
  - General: *g* (whether general is faulty or non-faulty)

27

Variables of Officer u on Round r

- *commit*[*u*,*r*]: **Boolean**. *commit*[*u*,*r*] means that officer *u* is committed to attack from round *r* onwards. Once committed to attack, a non-faulty officer does not change its decision.
- *confirm*[*u*,*x*,*r*]: **Boolean**. *confirm*[*u*,*x*,*r*] means that officer *u* has received a *commit[x, k]* message for *k* at most *r*.
- *confirmationCount*[*u*,*r*]: **integer**. Count of number of confirmations that *u* has at round *r, i.e.,* the number of officers *x* for which *u* has received *confirm*[x,*k*] messages, for *k* at most *r*

28

Asymmetry between attack and retreat

- We treat no message in a round as a *retreat* message.
- Our algorithm determines that either:
1. All non-faulty officers set a local value, which is initially false, to true (commit to attack) by round K
2. All non-faulty officers never modify a local value, which is initially false.
- The algorithm is asymmetric with respect to the two outcomes (attack, retreat), but we could choose either of them to represent value **true**.

29

Rule for Commitment for Commanders

commit[*u, r*] =
commit[*u, r* – 1]
OR
((*confirmationCount*[*u, r*] >= *r*) AND *confirm*[*u*,*g*,*r*])

30

Algorithm: Non-Faulty Officer *u* on Round *r*

Commitment message:

If committing for the first time in round *r*,
(*i.e.*, *commit*[*u*,*r*-1] = **false** and *commit*[*u*,*r*] = **true),**
then send *commitMessage*[*u*] to all officers.

31

Algorithm: Non-Faulty Officer *u* on Round *r*

Relaying:

If u receives a *commitMessage*[*x*] it relays the message to all officers.

i.e.,
If confirming *x* for the first time in round *r*,
(*i.e., confirm*[*u*,*x*,*r* – 1] = **false** and *confirm*[*u*,*x*,*r*] = **true**)
then send (a copy of) *commitMessage*[*x*] to all officers.

32

Initial Conditions: Round *r* (*r* = 0)

For non-faulty officer *u* other than general *g*:
- *commit*[*u*,0] = **?**
- *confirm*[*u*,*x*,0] = **?**
- *confirmationCount*[*u*, 0] = ?

33

Initial Conditions: Round *r* (*r* = 0)

For non-faulty officer *u* other than general *g*:
- *commit*[*u*,0] = **false**
  - Non-faulty officer u has not committed to attack on round 0
- *confirm*[*u*,*x*,0] = **false,** for all x
  - Non-faulty officer *u* has not received *commit[x, k]* message for k at most 0.
- *confirmationCount*[*u*, 0] = 0
  - The number of officers x for which officer u has received *commit[x, k]* messages, for k at most 0, is 0

34

Initial Conditions for Non-Faulty General

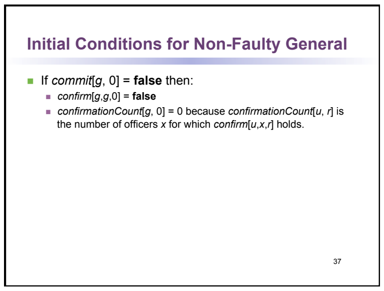
- For general *g*: *commit*[*g*, 0] is an arbitrary Boolean value. The value is **true** signifies generals' command is "attack."
- For all *v* different from *g*: *confirm*[*g*,*v*,0] = **false**
  - For all officers *v* other than the general itself, the general has not received a confirmation from general v on round 0.
- *confirm*[*g*,*g*,0] = *commit*[*g*, 0]

35

Initial Conditions for Non-Faulty General

- If *commit*[*g*, 0] = **true** then:
  - *confirm*[*g*,*g*,0] = **true**
  - *confirmationCount*[*g*, 0] = 1 because *confirmationCount*[*u*, *r*] is the number of officers *x* for which *confirm*[*u*,*x*,*r*] holds.
  - General *g* sends a *commitMessage*[*g*] to all officers indicating that the general is committed.

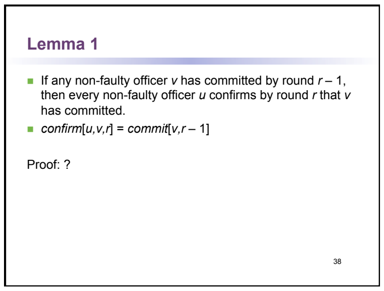
36



Initial Conditions for Non-Faulty General

- If *commit*[*g*, 0] = **false** then:
  - *confirm*[*g*,*g*,0] = **false**
  - *confirmationCount*[*g*, 0] = 0 because *confirmationCount*[*u*, *r*] is the number of officers *x* for which *confirm*[*u*,*x*,*r*] holds.

37

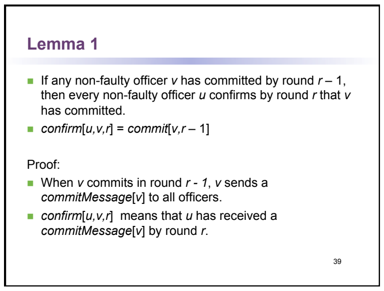


Lemma 1

- If any non-faulty officer *v* has committed by round *r* – 1, then every non-faulty officer *u* confirms by round *r* that *v* has committed.
- *confirm*[*u*,*v*,*r*] = *commit*[*v*,*r* – 1]

Proof: ?

38



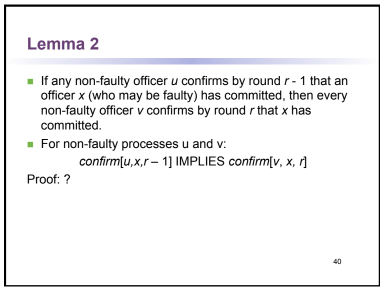
Lemma 1

- If any non-faulty officer *v* has committed by round *r* – 1, then every non-faulty officer *u* confirms by round *r* that *v* has committed.
- *confirm*[*u*,*v*,*r*] = *commit*[*v*,*r* – 1]

Proof:

- When *v* commits in round *r - 1*, *v* sends a *commitMessage*[*v*] to all officers.
- *confirm*[*u*,*v*,*r*] means that *u* has received a *commitMessage*[*v*] by round *r*.

39



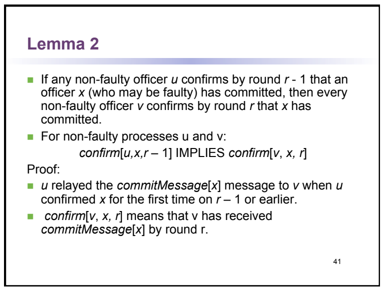
Lemma 2

- If any non-faulty officer *u* confirms by round *r* - 1 that an officer *x* (who may be faulty) has committed, then every non-faulty officer *v* confirms by round *r* that *x* has committed.
- For non-faulty processes u and v:

  confirm[*u*,*x*,*r* – 1] IMPLIES *confirm*[*v*, *x*, *r*]

Proof: ?

40



Lemma 2

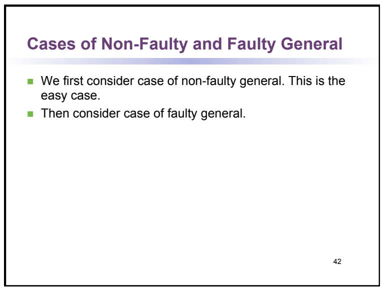
- If any non-faulty officer *u* confirms by round *r* - 1 that an officer *x* (who may be faulty) has committed, then every non-faulty officer *v* confirms by round *r* that *x* has committed.
- For non-faulty processes u and v:

  confirm[*u*,*x*,*r* – 1] IMPLIES *confirm*[*v*, *x*, *r*]

Proof:

- *u* relayed the *commitMessage*[*x*] message to *v* when *u* confirmed *x* for the first time on *r* – 1 or earlier.
- *confirm*[*v*, *x*, *r*] means that v has received *commitMessage*[*x*] by round r.

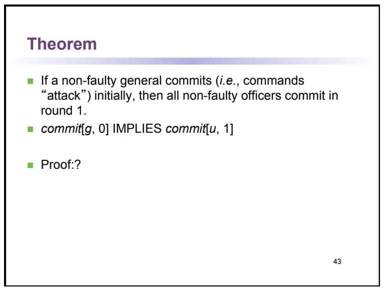
41



Cases of Non-Faulty and Faulty General

- We first consider case of non-faulty general. This is the easy case.
- Then consider case of faulty general.

42

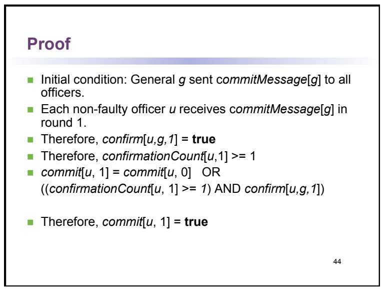


Theorem

- If a non-faulty general commits (*i.e.*, commands "attack") initially, then all non-faulty officers commit in round 1.
- *commit*[*g*, 0] IMPLIES *commit*[*u*, 1]

- Proof:?

43

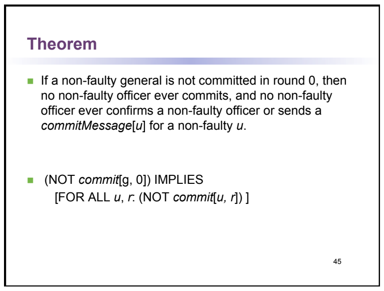


Proof

- Initial condition: General *g* sent *commitMessage*[*g*] to all officers.
- Each non-faulty officer *u* receives *commitMessage*[*g*] in round 1.
- Therefore, *confirm*[*u*,*g*,*1*] = **true**
- Therefore, *confirmationCount*[*u*,1] >= 1
- *commit*[*u*, 1] = *commit*[*u*, 0] OR ((*confirmationCount*[*u*, 1] >= *1*) AND *confirm*[*u*,*g*,*1*])

- Therefore, *commit*[*u*, 1] = **true**

44



Theorem

- If a non-faulty general is not committed in round 0, then no non-faulty officer ever commits, and no non-faulty officer ever confirms a non-faulty officer or sends a *commitMessage*[*u*] for a non-faulty *u*.

- (NOT *commit*[g, 0]) IMPLIES

  [FOR ALL *u*, *r*: (NOT *commit*[*u*, *r*])]

45

Proof

- *commit[g,0] = commit[g,r]*, all *r*
- Since *commit[g,0]* = FALSE

  commit[g,r] = FALSE, all r

- So, in round *r* + 1:
  - Officer *u* does not receive a *commit*[*g,r*] message.
  - So, *confirm*[*u,g,r* + 1] remains **false**.
- For all r, *confirm[u,g,r]* = FALSE, for all r

46

Proof

- Induction on *r*.
- Assume hypothesis holds for round *r*.

- *commit*[*u*, *r* + 1] = *commit*[*u*, *r*]

  OR

  ((*confirmationCount*[*u*, *r* + 1] >= *r* + 1) AND

  confirm[*u,g,r* + 1])

- So, *commit*[*u*, *r* + 1] remains **false**.

47

Non-faulty general is the easy case.

- If general says attack all non-faulty officers commit in round 1.
- If general says retreat then all non-faulty officers never commit in any round.
- So, if the general is non-faulty the decision is made at the end of round 1.

- What could happen if the general is faulty?

48

Theorem

- If a non-faulty officer commits in round *r*, then all non-faulty officers commit by round *r* + 1.

- Note: This theorem holds whether the general is faulty or non-faulty.

49

Proof

- Suppose officer *u* commits for the first time in round *r*.
- *commit*[*u*, *r*] = *commit*[*u*, *r* – 1] OR

  ((*confirmationCount*[*u*, *r*] >= *r*) AND *confirm*[*u,g,r*])
- Since u is committing for the *first* time in round *r*:

  ((*confirmationCount*[*u*, *r*] >= *r*) AND *confirm*[*u,g,r*])

50

Proof

- Officer *u* sends officer *v* the *r* commit messages it has received by round *r* (including the general's), and also send its own commit message in round *r*.
- (*commit*[*v*, *r* + 1] = *commit*[*v*, *r*]) OR

  ((*confirmationCount*[*v*, *r* + 1] >= *r* + 1) AND

  confirm[*v,g,r* + 1])
- *confirmationCount*[*v*, *r* + 1] >= *r* + 1, because *v* received the *r* commit messages that *r* used to commit and *r*'s own message.
- *confirm*[*v,g,r* + 1] = **true** because *v* received the *commitMessage*[*g*] from *u*.

51

Theorem

- If no non-faulty officers commit in the first *t* rounds, then no non-faulty officer will commit in later rounds.

52

Proof

- *commit[u, t+1]* =

  commit[*u*, *t*]) OR

  ((*confirmationCount*[*u*, *t* + 1] >= *t* + 1) AND

  confirm[*u,g,t* + 1])
- By assumption, *commit*[*u*, *t*] = **false**
- Since *commit*[*v*, *t*] = **false** for all non-faulty officers *v*, it follows that (*confirmationCount*[*u*, *t+1*] =< *t*)
- Hence, *commit*[*u*, *t* + 1] = **false**

53

Termination Round

- Stop at end of round t + 1.

54