FAULT-TOLERANT AGREEMENT

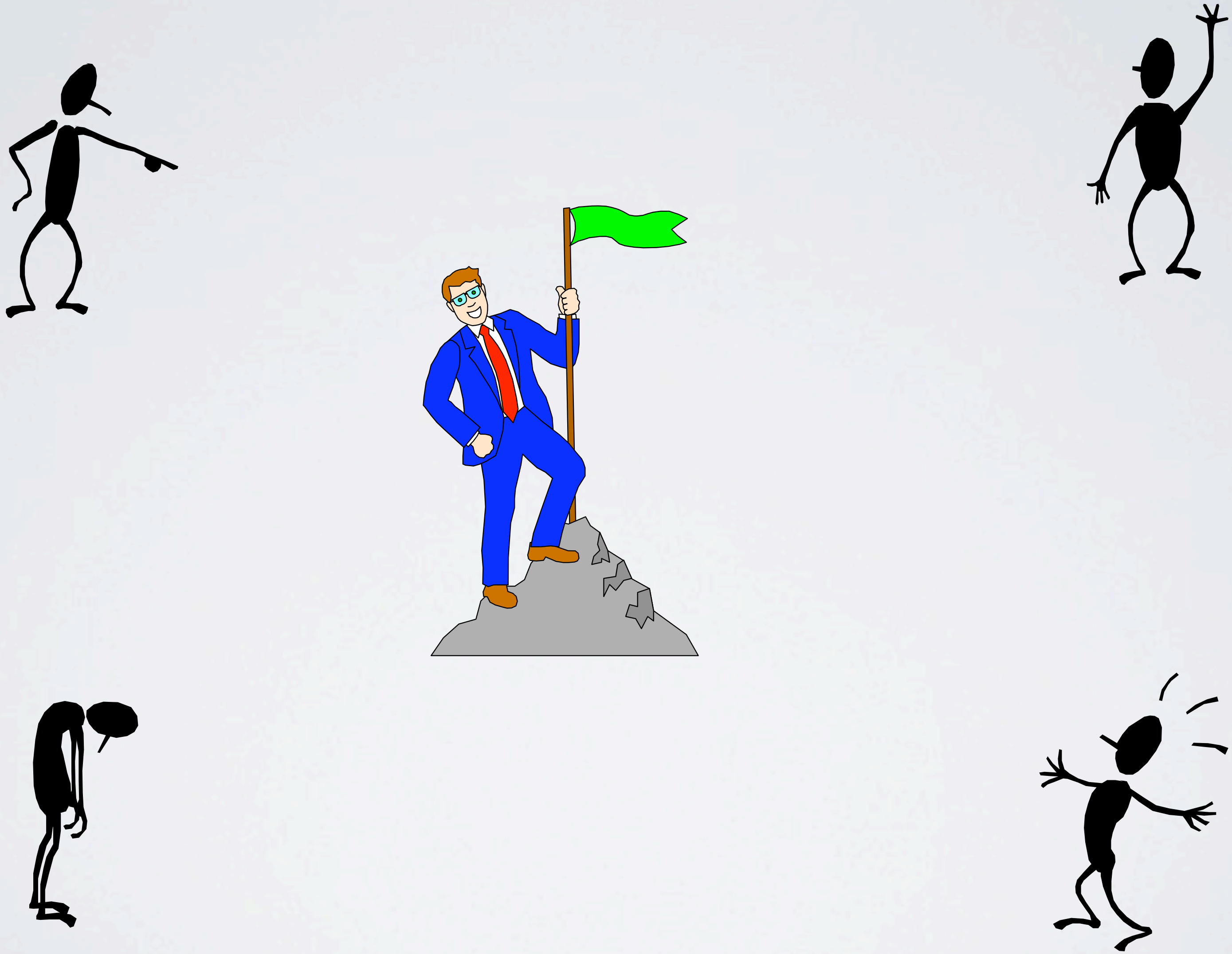
- Having discussed the issues of addressing malicious act in the context of dependability, we will now look at a classic solution to agreeing in the presents of faults:

  Byzantine Agreement

- This paper was not written with our interpretation of survivability, but will a great starting point to discuss the strength and weakness of agreement based solutions to survivability.

- The following set of slides is from the fault-tolerance course.

BYZANTINE GENERAL PROBLEM



BYZANTINE GENERAL PROBLEM

- Objective
  - A) All loyal generals must decide on the same plan of action
  - B) A "small" number of traitors cannot cause the loyal generals to adopt a "bad" plan.
- Types of agreement
  - exact agreement
  - approximate agreement
- Applications, e.g.
  - agreement in the presence of faults
  - event, clock synchronization

BYZANTINE GENERAL PROBLEM

- Key to disagreement
  - 1) Initial disagreement among loyal generals
  - 2) Ability of traitor to send conflicting messages
    - asymmetry
- Reduction of general problem to simplex problem with 1 General and n-1 Lieutenants
  - General gives order
  - Loyal Lieutenants must take single action

BGP: SIMPLEX

- Want

  IC1: All loyal Lieutenants obey the same order

  IC2: If the commanding General is loyal, the every loyal Lieutenant obeys the order he sends

  - IC1 & IC2 are called *Interactive Consistency Conditions.*
  - If the General is loyal, then IC1 follows from IC2.
  - However, the General need not be loyal.
- Any solution to the simplex problem will also work for multiple-source problems.
  - the i^{th} General sends his value *v(i)* by using a solution to the BGP to send the order "use *v(i)* as my value", with the other Generals acting as the lieutenants.

BGP: ORAL MESSAGE SOLUTION

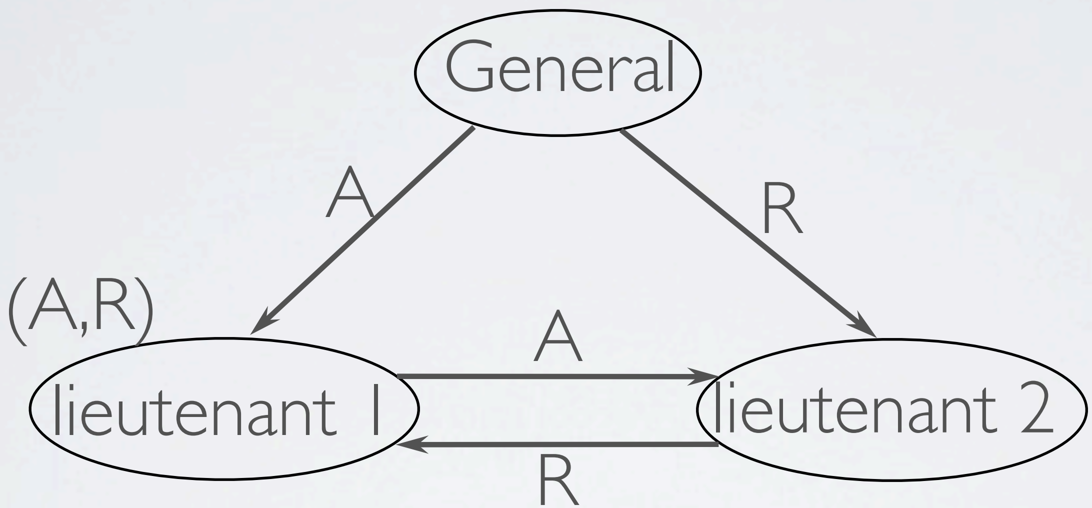
- Oral Message
  - message whose contents are under the control of the sender (possibly relays)
- Practical implication, sensor example
  - General = sensor
  - Lieutenants = processor redundantly reading sensor
  - Initial disagreement
    - time skew in reading, bad link to sensor
    - analog - digital conversion error, any threshold function
  - Asymmetry
    - communication problem, noise, V-level, bit timing

BGP: ORAL MESSAGE SOLUTION

- The Byzantine Generals Problem seems deceptively simple, however
- no solution will work unless more than two-third of the generals are loyal.
- Thus, there exists no 3-General solutions to the single traitor problem using <u>oral messages</u>
- Assume the messages sent are
  - A = Attack
  - R = Retreat

BGP: ORAL MESSAGE SOLUTION

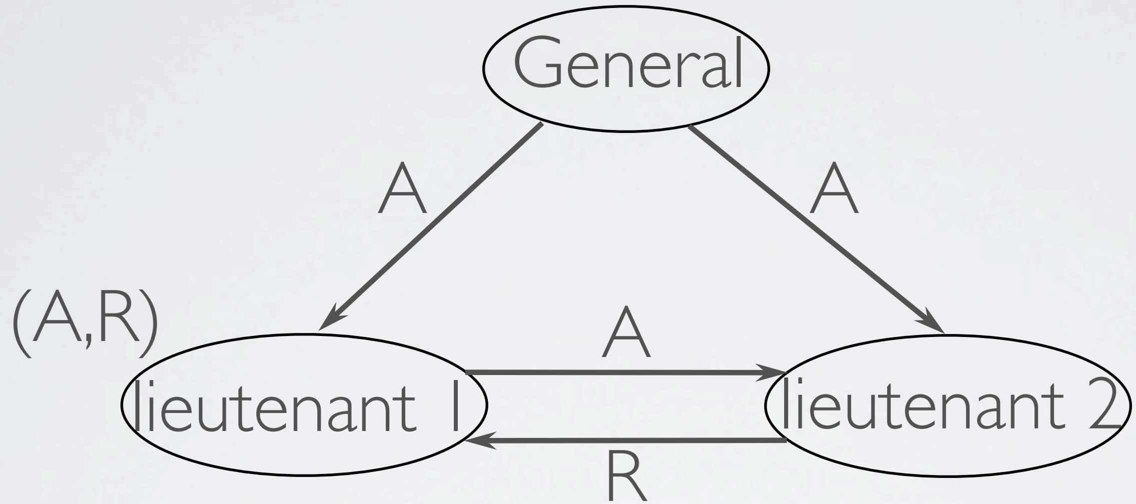
- Case 1: Commander is traitor:



  - commander is lying
  - who does lieutenant 1 believe
  - could pick default

BGP: ORAL MESSAGE SOLUTION

- Case 2: Lieutenant 2 is traitor:



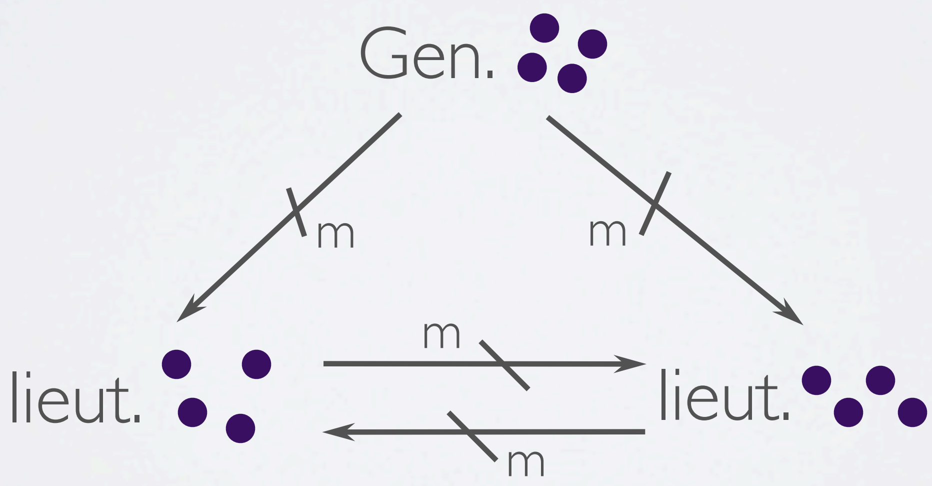
  - lieutenant 2 is lying
  - who does lieutenant 1 believe
  - could pick default, but what if it is R
    - then General has A and Lieutenant 1 has R !!!

BGP: ORAL MESSAGE SOLUTION

- Given case 1 and case 2, lieutenant 1 cannot differentiate between both scenarios, i.e. the set of values lieutenant 1 has is (A,R).
- In general: Given m traitors, there exists no solution with less than 3m+1 generals for the <u>oral message</u> scenario.
- Assumptions about Oral Messages
  - every message that is sent is delivered correctly
  - the receiver of a message knows who send it
  - the absence of a message can be detected
  - how realistic are these assumptions?

BGP: ORAL MESSAGE SOLUTION

- General case:
  - regroup generals
    - n Albanian generals
    - n/3 act as unit => 3 general Byzantine General Problem



BGP: ORAL MESSAGE SOLUTION

Algorithm OM(0)

1) The commander sends his value to every lieutenant

2) Each lieutenant uses the value he receives from the commander, or uses the value RETREAT if he receives no value

Algorithm OM(m), m>0

1) The commander sends his value to every lieutenant.

2) For each i, let v_i be the value lieutenant i receives from the commander, or else be RETREAT if he receives no value. Lieutenant i acts as the commander in Algorithm OM(m-1) to send the value v_i to each of the n-2 other lieutenants.

3) For each i, and each j != i , let v_j be the value lieutenant i received from lieutenant j in step 2) (using algorithm OM(m-1), or else RETREAT if he received no such value. Lieutenant i uses the value

$$majority(v_1, \ldots, v_{n-1})$$

BGP: ORAL MESSAGE SOLUTION

OM(m) -- same thing, different wording

IF m = 0 THEN

a) commander sends his value to all other (n-1) lieutenants.

b) lieutenant uses value received or default (i.e. RETREAT if no value was received).

ELSE

a) each commander node sends value to all other (n-1) lieutenants

b) let v_i = value received by lieut. i (from commander OR default if there was no message)

Lieut. i invokes OM(m-1) as commander, sending v_i to other (n-2) lieutenants.

c) let v_{ji} = value received from lieutenant j by lieutenant i.

Each lieutenant i gets v_i = maj(what everyone said j said in previous round, except j himself)

trust myself more than what others say I said

EXAMPLE N=4 => ONE TRAITOR

- procedure OM(1)

IF {not valid since m=1}

ELSE

1) commander transmits to L1,L2,L3

2) values are received by L1,L2,L3

so lieuts call OM(0)

procedure OM(0)

IF {m=0}

1) each lieut sends value

to

other 2 lieuts

ELSE {not valid}

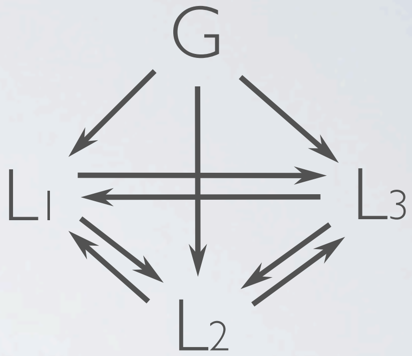
each lieut has

received 3 values

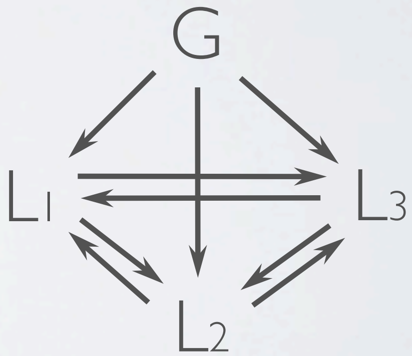
(use majority)

BGP EXAMPLE

- case 1: L3 is traitor
  v0 = 1
  each loyal L has vector
  110 or 111 => maj(1 1 0/1) = 1



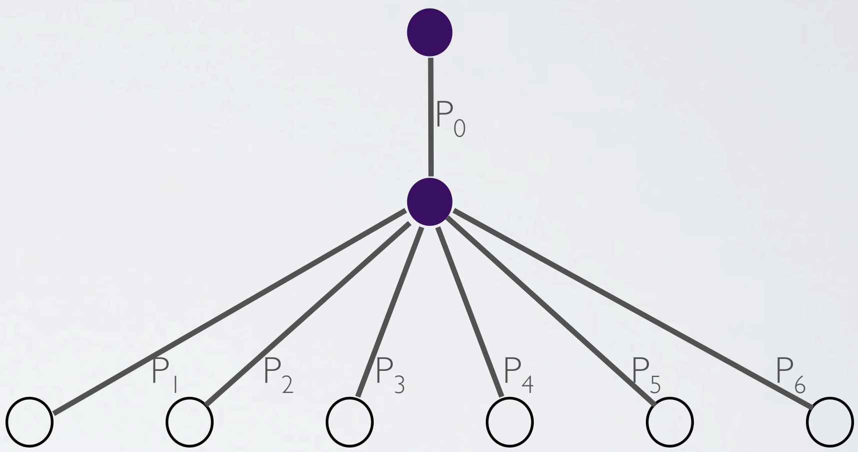
- case 2: G is traitor
  v0 => L1=1 L2=1 L3=0
  L1 has 110
  L2 has 110 maj() = 1
  L3 has 011



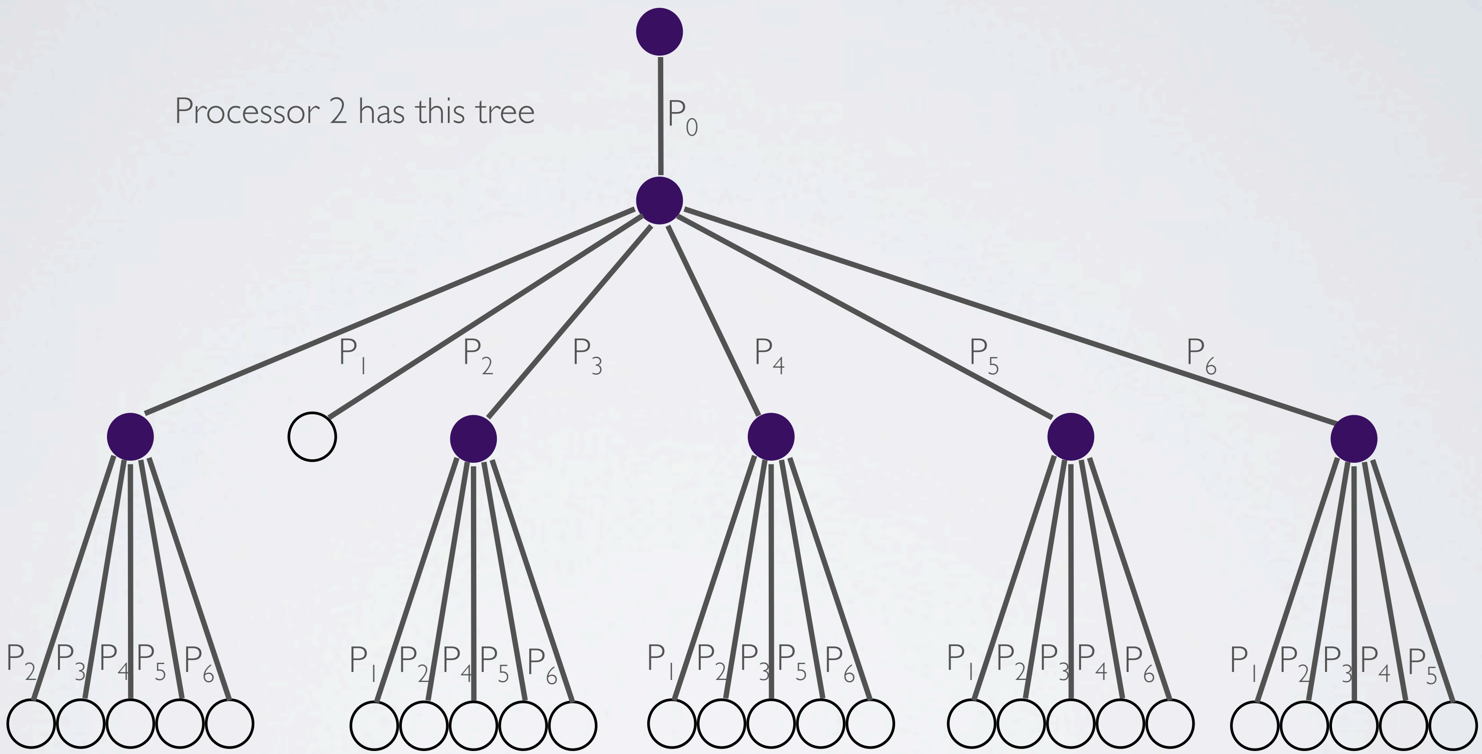
BGP WITH N =7

General sends message

After first rebroadcast



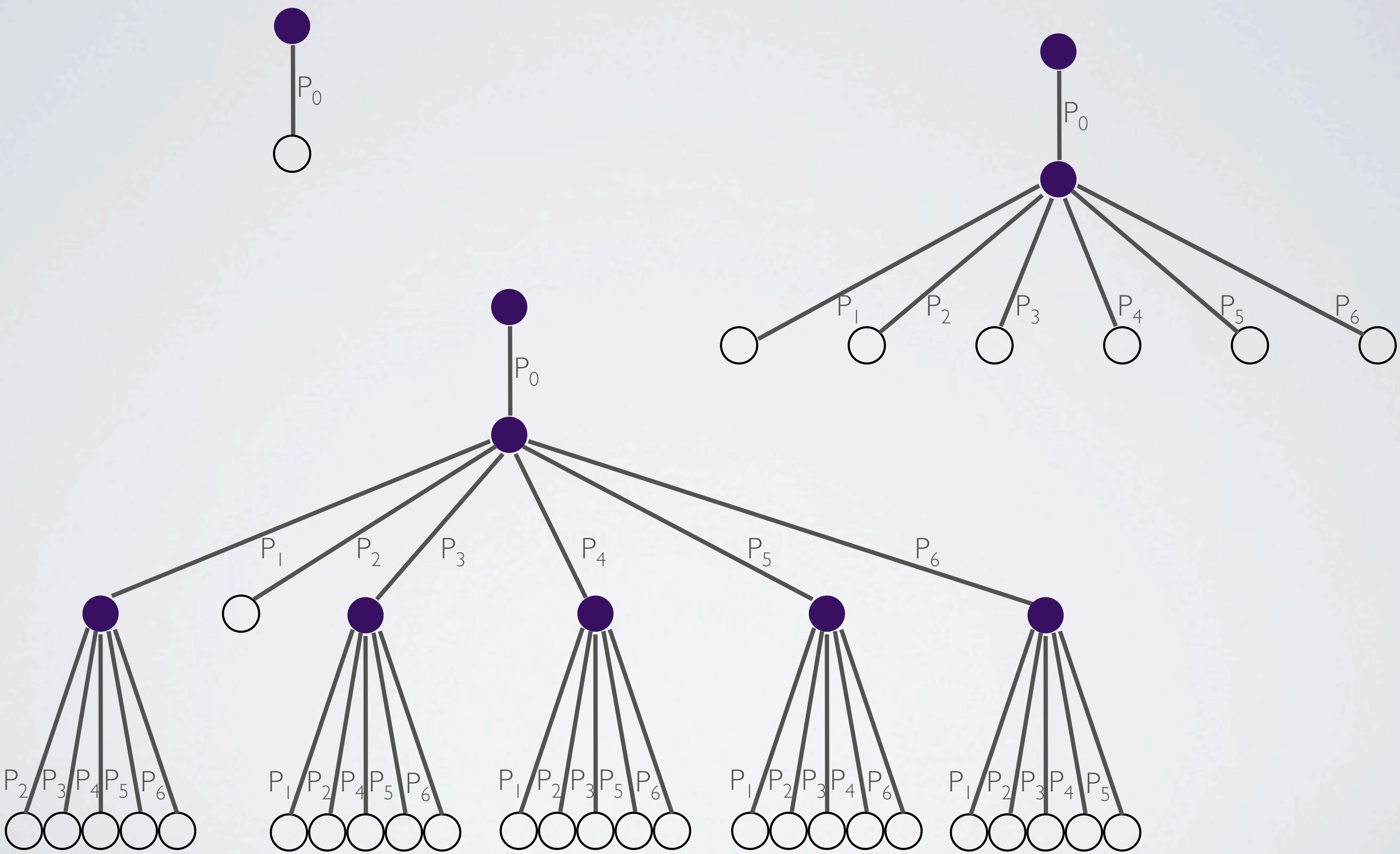
BGP WITH N =7


Processor 2 has this tree
P_0
P_1 P_2 P_3 P_4 P_5 P_6
P_2 P_3 P_4 P_5 P_6
P_1 P_2 P_4 P_5 P_6
P_1 P_2 P_3 P_5 P_6
P_1 P_2 P_3 P_4 P_6
P_1 P_2 P_3 P_4 P_5

BGP WITH N =3M+1

extra blank

BGP WITH N =7


P0
P0
P1 P2 P3 P4 P5 P6
P0
P1 P2 P3 P4 P5 P6
P2 P3 P4 P5 P6
P1 P2 P4 P5 P6
P1 P2 P3 P5 P6
P1 P2 P3 P4 P6
P1 P2 P3 P4 P5

SIGNED MESSAGES

- Traitors ability to lie makes Byzantine General Problem so difficult.
- If we restrict this ability, then the problem becomes easier
- Use authentication, i.e. allow generals to send unforgeable signed messages.

SIGNED MESSAGES

- Assumptions about *Signed Messages*

  A1: every message that is sent is delivered correctly

  A2: the receiver of a message knows who send it

  A3: the absence of a message can be detected

  A4: a loyal general's signature cannot be forged, and any alteration of the contents of his signed messages can be detected. Anyone can verify the authenticity of a general's signature

  Note: no assumptions are made about a traitor general, i.e. a traitor can forge the signature of another traitor.

SIGNED MESSAGES

- Signed message algorithm assumes a *choice* function
  - if a set V has one single element v, then $choice(V) = v$
  - choice(Φ) = R, where Φ is the empty set
    - RETREAT is default
  - choice(A,R) = R
    - RETREAT is default
  - set V is <u>not</u> a multiset (recall definition of a multiset)
  - thus set V can have at most 2 elements, e.g. $V = \{A,R\}$.

SIGNED MESSAGES

- Signing notation
  - let *v:i* be the value *v* signed by general *i*
  - let *v:i:j* be the message *v:i* counter-signed by general *j*
- each general *i* maintains his own set V_i containing all orders he received
- Note: do not confuse the set V_i of orders the general received with the set of all messages he received. Many different messages may have the same order.

BGP: SIGNED MESSAGE SOLUTION

SM(m) -- from Lam82

Initially $V_i = \Phi$

1) The commander signs and sends his value to every lieutenant

2) For each i

A) If lieutenant i receives a message of the form v:0 from the commander and he has not yet received any order, then

i) he lets V_i equal {v}

ii) he sends the message $v{:}0{:}i$ to every other lieutenant

B) If lieutenant i receives a message of the form $v{:}0{:}j_1{:}\ldots{:}j_k$ and v is not in the set V_i, then

i) he adds v to V_i

ii) if $k<m$, then he sends the message $v{:}0{:}j_1{:}\ldots{:}j_k{:}i$ to every lieutenant other than $j_1,\ldots,j_k$

3) for each i: When lieutenant i will receive no more messages, he obeys the order $choice(V_i)$.

ALGORITHM SM(M)

- the SM(m) algorithm for signed messages works for

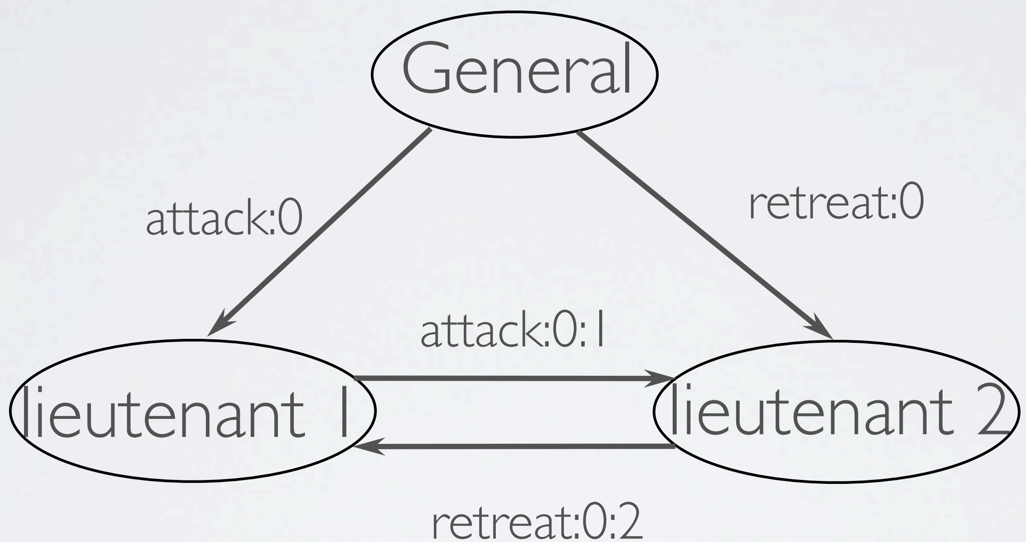
$$N \geq m + 2$$

  i.e. want non faulty commander and at least one non faulty lieutenant

- How does one know when one does not receive any more messages?
  - by ***missing message assumption*** A3, we can tell when all messages have been received
  - this can be implemented by using synchronized rounds
- Now traitor can be detected!
  - e.g. 2 correctly signed values => general is traitor

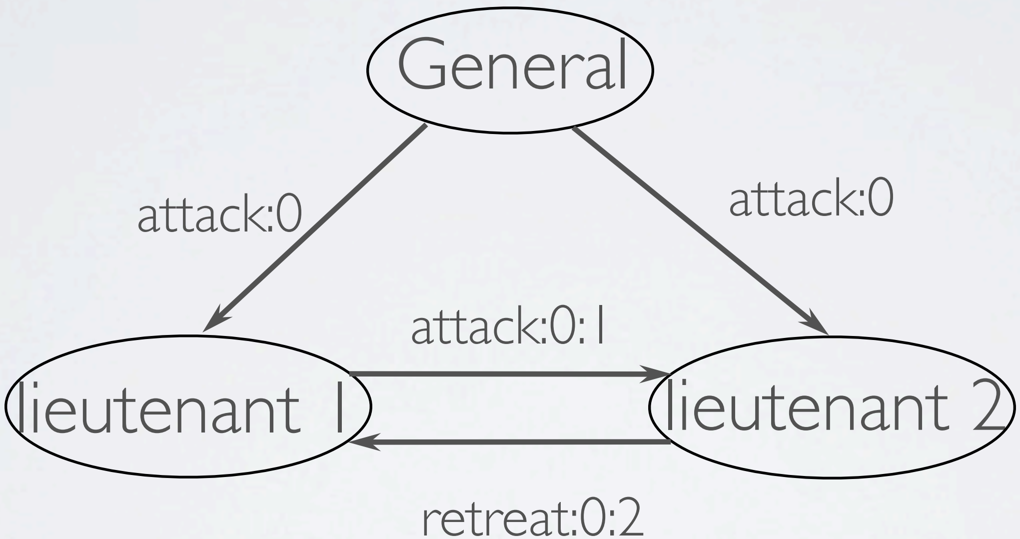
ALGORITHM SM(M)

- example, general is traitor



ALGORITHM SM(M)

- example, lieutenant 2 is traitor



ALGORITHM SM(M)

- example:
  - SM(0)
    - general sends *v:0* to all lieutenants
    - processor *i* receives *v:0* $V_i=\{v\}$
  - SM(1)
    - each lieut. countersigns and rebroadcasts *v:0*
    - processor *i* receives (*v:0:1*, *v:0:2*,..., *v:0:(N-1)*)

ALGORITHM SM(M)

- case 1: commander loyal, lieutenant j = traitor
  - all values <u>except</u> $v{:}0{:}j$ are v

    $\Rightarrow v \in V_i \quad \forall$ loyal lieut. i

  - processor j cannot tamper

    $\Rightarrow V_i = \{\mathrm{v}\} \quad \forall$ loyal lieut. i

- case 2: commander = traitor, => all lieut. loyal
  - all lieutenants correctly forward what they received
    - agreement: yes
    - validity: N/A

ALGORITHM SM(M)

- case 1: commander loyal, 2 lieutenants are traitors
  - want each loyal lieut to get $V=\{v\}$
  - round 0 => all loyal lieuts get v from commander
  - other rounds:
    - traitor cannot tamper
    - => all messages are v or Φ
- case 2: commander traitor + 1 lieut. traitor
  - round 0: all loyal lieuts receive $v{:}0$
  - round 1:
    - traitors send <u>one</u> value or Φ
  - round 2:
    - another exchange (in case traitor caused split in last round)
    - traitor still can <u>not</u> introduce new value

      => agreement: yes

      validity: N/A

ALGORITHM SM(M)

- Cost of signed message
  - encoding one bit in a code-word so faulty processor cannot "stumble" on it.
  - e.g.
    - unreliability of the system $F_S = 10^{-10}/h$
    - unreliability of single processor $F_P = 10^{-4}/h$
    - want: Probability of randomly generated valid code word

$$P = \frac{10^{-10}}{10^{-4}} = 10^{-6} \approx 2^{-20}$$

    - given 2^i valid codewords, want (20+*i*) bits/signature
    - e.g. Attack/Retrieve
      => 2^1
      => 21 bit signature

AGREEMENT

- Important notes:
  - there is no way to guarantee that different processors will get the same value from a possibly faulty input device, except having the processors communicate among themselves to solve the Byz.Gen. Problem.
  - faulty input device may provide meaningless input values
    - all that Byz.Gen. solution can do is guarantee that all processors use the same input value.
    - if input is important, then use redundant input devices
    - redundant inputs cannot achieve reliability. It is still necessary to insure that all non-faulty processors use the redundant data to produce the same output.

AGREEMENT

- Implementing BGP is no problem
- The problem is implementing a message passing system that yields respective assumptions, i.e.:

  A1: every message that is sent is delivered correctly

  A2: the receiver of a message knows who send it

  A3: the absence of a message can be detected

  A4: a loyal general's signature cannot be forged, and any alteration of the contents of his signed messages can be detected. Anyone can verify the authenticity of a general's signature