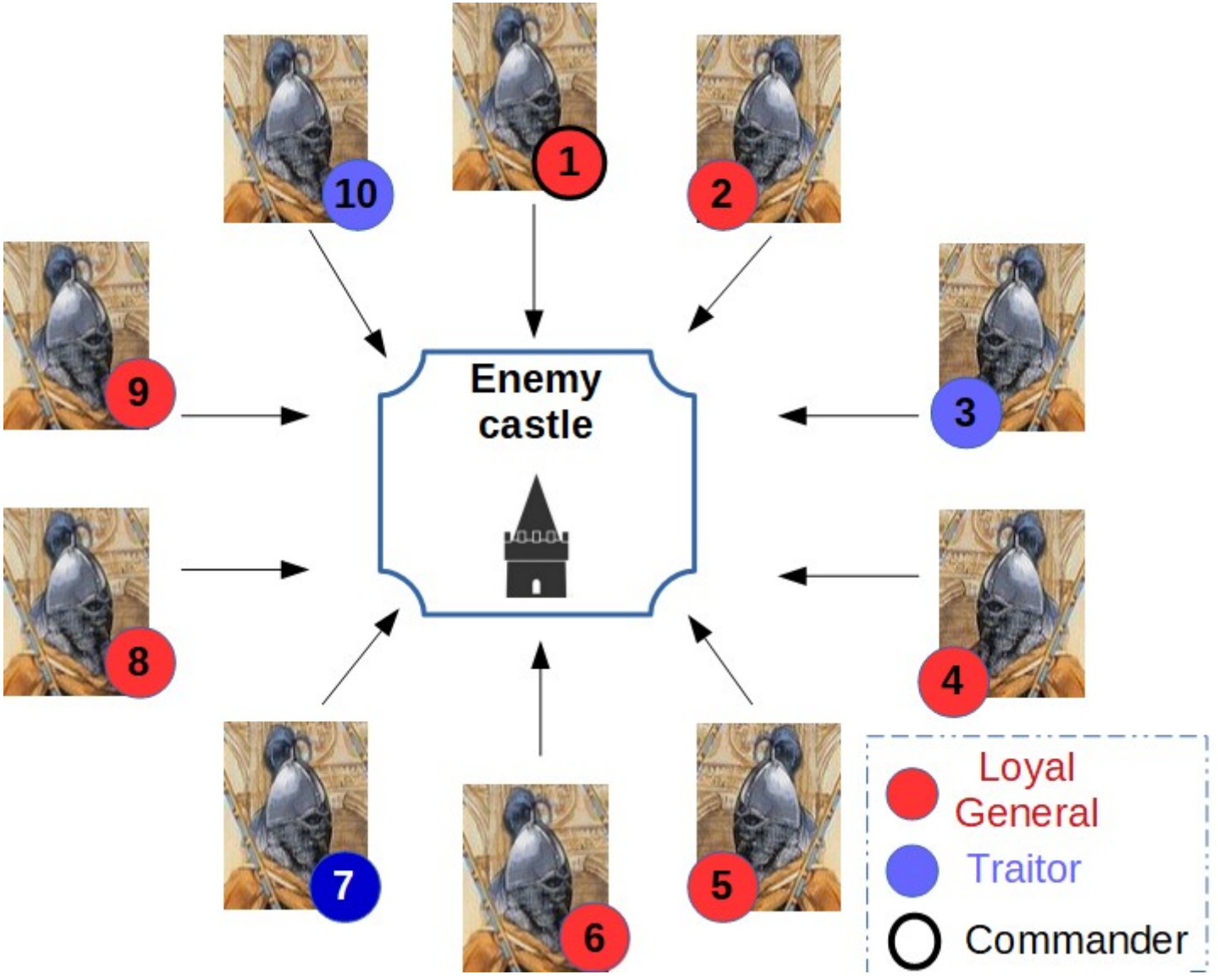
The Byzantine Generals Problem and Solutions

李旭东 Li-Xudong

leexudong@nankai.edu.cn


10
1
2
9
Enemy
castle
3
8
4
7
6
5
Loyal
General
Traitor
Commander

Attack
进攻
or
Retreat
撤退

Lamport, et al. "The Byzantine Generals Problem." Acm Transactions on Programming Languages & Systems 4.3(1982):382-401.

The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement. It is shown that, using only oral messages, this problem is solvable if and only if more than two-thirds of the generals are loyal; so a single traitor can confound two loyal generals. With unforgeable written messages, the problem is solvable for any number of generals and possible traitors. Applications of the solutions to reliable computer systems are then discussed.

Categories and Subject Descriptors: C.2.4. [**Computer-Communication Networks**]: Distributed Systems—*network operating systems*; D.4.4 [**Operating Systems**]: Communications Management—*network communication*; D.4.5 [**Operating Systems**]: Reliability—*fault tolerance*

General Terms: Algorithms, Reliability

Additional Key Words and Phrases: Interactive consistency

Objectives

- The Byzantine Generals Problem
- Scenarios and Scopes
- Lamport’s solutions
- Discussions

Lamport:
The Byzantine Generals Problem (BGP)

- A group pf generals of the Byzantine army camped with their troops around an enemy city
  - 1. Communicating only by messenger
  - 2. All generals must agree upon a common battle plan
  - 3. But one or more of them may be traitors who will try to confuse the others

n generals , m traitors 叛徒

- **The problem**: how to find an algorithm to ensure that the loyal generals will reach a same battle plan (i.e. consistency)

Lamport:
The Byzantine Generals Problem

Byzantine Generals Problem. A commanding general must send an order to his $n - 1$ lieutenant generals such that

IC1. All loyal lieutenants obey the same order.
IC2. If the commanding general is loyal, then every loyal lieutenant obeys the order he sends.

Two requirements!

Conditions IC1 and IC2 are called the *interactive consistency* conditions. Note that if the commander is loyal, then IC1 follows from IC2. However, the commander need not be loyal.

- **IC1: consistency (interactive consistency)**
- **IC2: correctness**

Example 1 (m=1,n=3)

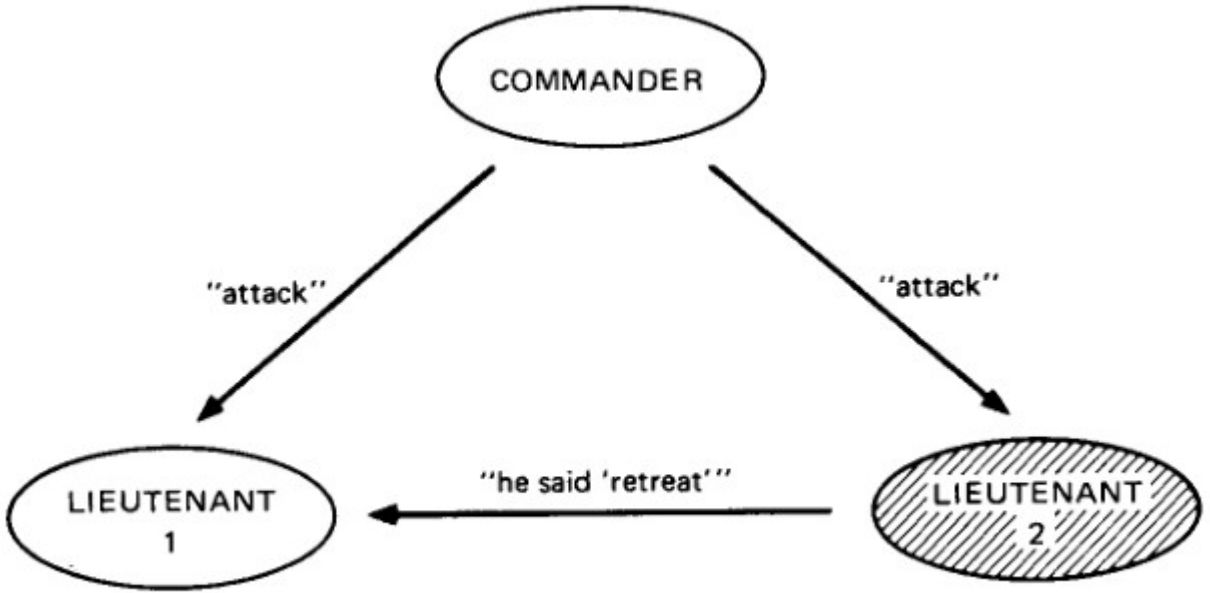


Fig. 1. Lieutenant 2 a traitor.

A: Attack
R: Retreat

C: A
L1:{A,R} => ?
L2: X (traitor)

- For L1:
  - In fact, L1 does not know who is traitor
- So,
- Case 1.1: R
  - The result cannot meet IC1
    - C and L1 make different actions
- Case 1.2:A
  - ! Temporarily correct

Example 1 (m=1,n=3)

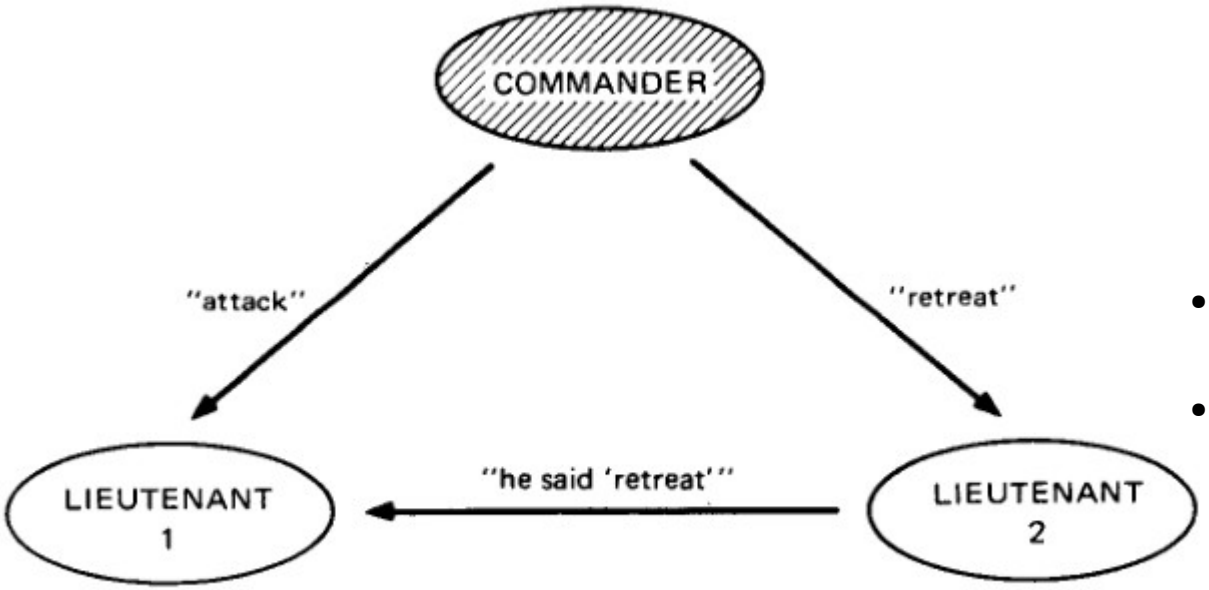


Fig. 2. The commander a traitor.

C: X (traitor)
L1: {A,,R} => ?
L2: {R,,A} => ?

- For L1 & L2:
  - They don't know who is the traitor
- Case 2.1: obey the commander
  - L1:A,
  - L2:R
  - The result cannot meet IC1
- Case 2.2: don't obey the commander
  - L1:R,
  - L2:A
  - The result cannot meet IC1

The Byzantine Node

- These traitors are called byzantine nodes.
- These loyal generals are called non-byzantine nodes.

Lamport’s Solutions

Basic conclusion

- No solution with fewer than 3m+1 generals can cope with m traitors

Lamport's solution: Oral Messages Algorithm

- Oral Messages Algorithm 口头消息算法
  - OM(m): m traitors
  - **majority**(v_1, v_2, … , v_{n-1}) : every general makes a decision by it
- Conditions
  - **A1: Every message that is sent is delivered correctly**
  - **A2: The receiver of a message knows who sent it**
  - **A3: The absence of a message can be detected**

Lamport: Oral Messages Algorithm

- If there are m traitors
  - When the total of generals n >=3*m +1 , The OM(m) algorithm can solves the Byzantine Generals Problem
  - i.e. the OM(m) algorithm can ensure the above two requirements
    - IC1, IC2

Definition of Majority()

- majority(v1,v2,…,vn)
  - If a majority of the values vi equal v, then majority(v1,v2,…,vn) equals v
  - The function can be defined by customer ...

Oral Messages Algorithm

The following algorithm requires only the aforementioned property of *majority*.

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 $j \neq i$, 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*$(v_1, \ldots, v_{n-1})$.

AL:
Modified Oral Messages Algorithm

- New definition: MOM(m,n)
  - m: traitors
  - N: total leaders include commander and generals
- For the i-th General: Gi
  - V_i: Get the order (value) from its Commander
  - V_i:Send the order (value) as Commander to others
  - $\overline{V}_i$: Gi as Commander, Get the final order (value, i.e. state of sub-system) of all subordinate loyal generals
  - V_i: the final order (value, i.e. final decision) of Gi
    - V_i = majority(V_{j0} ,V_{j1} , …, $\overline{\mathbf{V}}_{\mathbf{i}}$, $V_{j\,j+1}$, …, $V_{j\,n-2}$)

AL: Modified Oral Messages Algorithm

- MOM(m,n): n>=3m+1
  - If m=0:
    - (1) The commander (i.e. G_0) sends his value (V_0) to every subordinate general
    - (2) Each subordinate general uses the above value(V_0) he receives from the commander, or uses the value RETREAT if he receives no value
    - (3) In fact, let $\overline{V}_0$ be the state of all subordinate generals (who are all loyal), $\overline{V}_0=V_0$
    - (4) In fact, let V_0 be the final order of the commander (G_0) , so $V_0=\overline{V}_0=V_0$
  - Cont.,

AL: Modified Oral Messages Algorithm

- MOM(m,n): n>=3m+1
  - If m>0:
    - (1) The commander sends his value (V_0) to every subordinate general
    - (2a) For each i-th subordinate general (Gi), let V_i be the value he receives from the commander, or uses the value RETREAT if he receives no value ;
      - 关键点1：（作为子将军，在递归中），别的将军commander会向Gi发送命令V_i
    - (2b), For each Gi, Gi acts as the commander in Algorithm **MOM(m-1,n-1)** to send the value V_i to each of the n-2 other subordinate generals
      - 关键点2：作为新的commander，在此做递归了
    - (2c),For each Gi, let $\overline{V}_i$ be the state of all subordinate loyal generals in **MOM(m-1,n-1)** when Gi acts as the commander
    - ...

AL:
Modified Oral Messages Algorithm

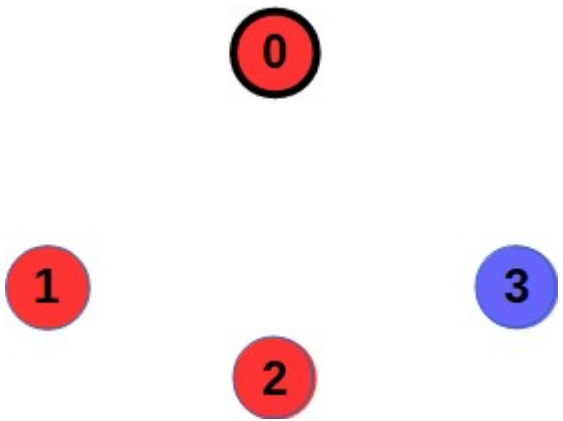
- MOM(m,n): n>=3m+1
  - If m>0:
    - (3a) For each i-th subordinate general (Gi), and for each j<>i, let V_j be value Gi received from Gj in step(2b) (Using Algorithm MOM(m-1,n-1)), or uses the value RETREAT if he receives no value ;
    - (3b), For each Gi, Gi uses the value V_i = majority(V_{j0} ,V_{j1} , …, $\overline{V}_i$, $V_{j\,j+1}$, …, $V_{j\,n-2}$) as his final decision
    - (3c),In fact, let V_0 be the state of all loyal generals, V_0 = majority (V_1,V_2,…,V_{n-1})

Oral Messages Algorithm

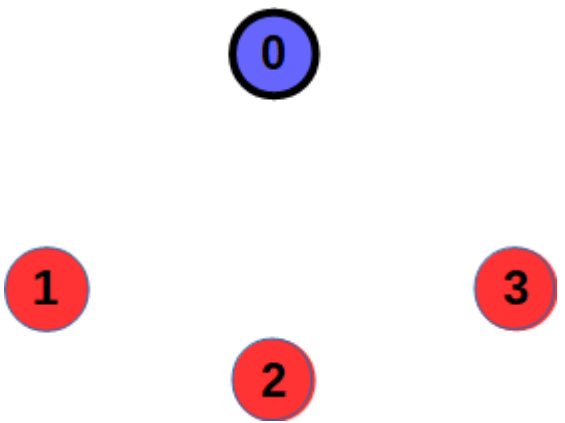
- Examples
  - (m,n) : n >= 3*m + 1
  - (1,3)
  - (1,4)
  - (2,7)
  - ...

Example2: MOM(1,4)

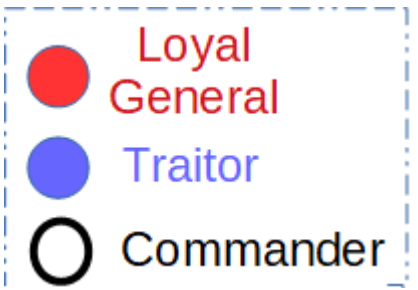
Case 1 :



Case 2 :



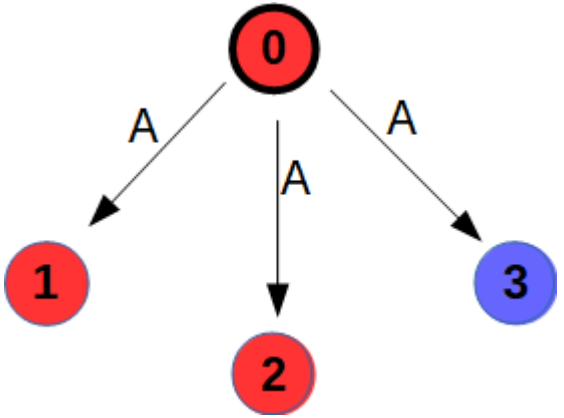
(m=1,n=4)



Example2: MOM(1,4) – Case 1

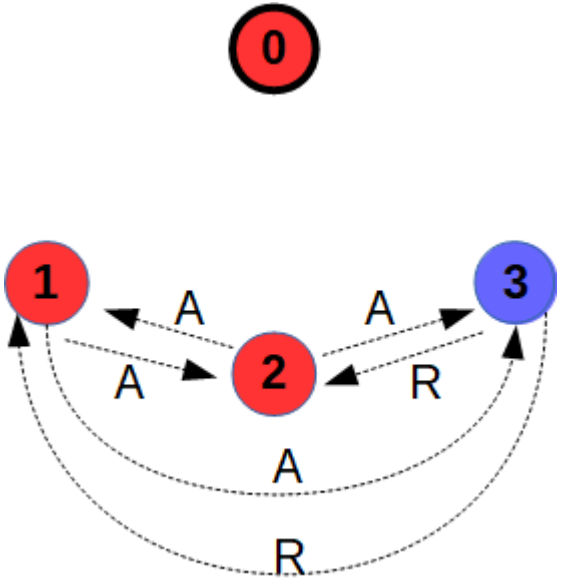
Example2: MOM(1,4) – case 1

MOM(1,4): Step 1

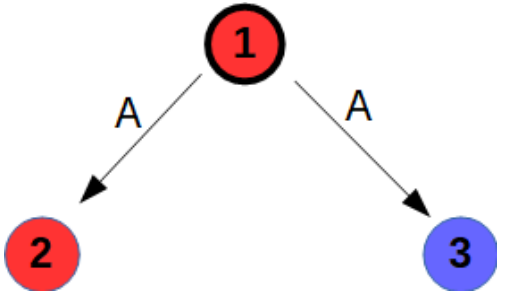


MOM(1,4): Step 2

- *For G1,G2,G3*
  - *Do MOM(0,3)*

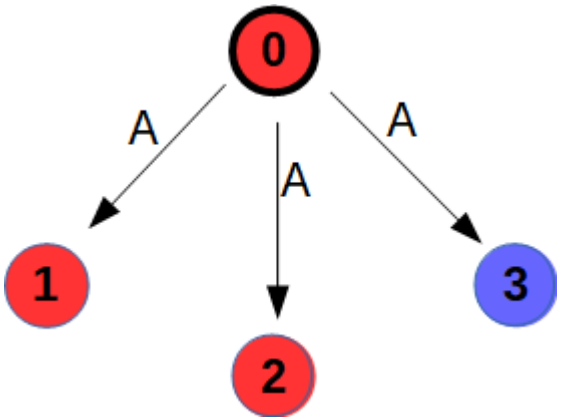


- MOM(1,4):Step 2
  - For G1
  - Do MOM(0,3)
  - $\overline{V}_1$=A
- MOM(1,4):Step 3
  - For G1
  - V_1=majority($\overline{V}_1$,V_2,V_3)
    - =majority(A,A,R)
    - =A
  - For G2 : same to G1



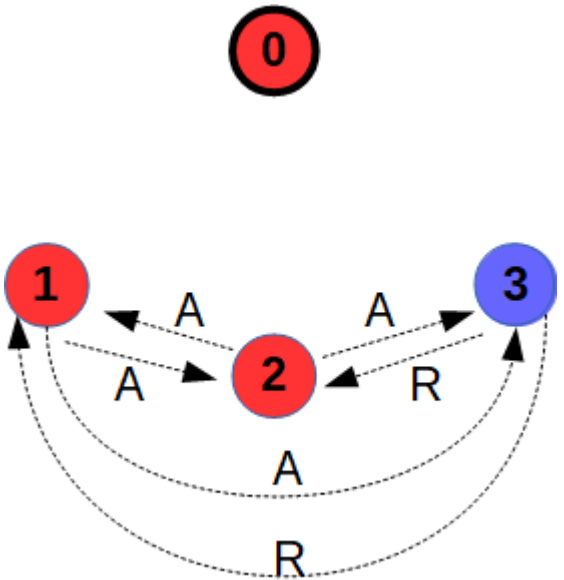
Example2: MOM(1,4) – case 1

MOM(1,4): Step 1



MOM(1,4): Step 2

- *For G1,G2,G3*
  - *Do MOM(0,3)*



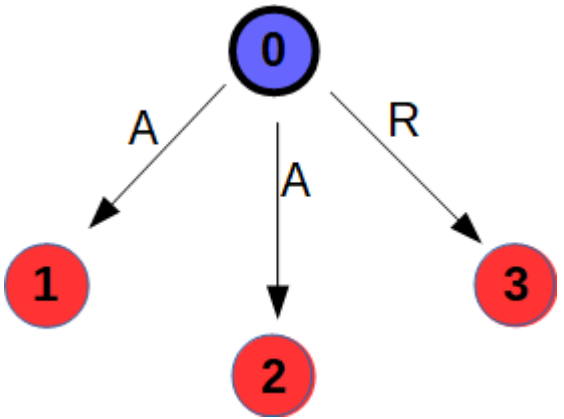
- MOM(1,4):Step 3
  - For G3:
    - He is traitor
    - So, V_3= X
    - X means any action (A or R)
  - At last, For G_0 (Commander)
    - $\overline{V}_0$=majority(V_1,V_2,V_3)
      - =majority(A,A,X)
      - =A
    - So, V_0=$\overline{V}_0$=V
  - IC1 ok
  - IC2 ok

Example2: MOM(1,4) – Case 2.1

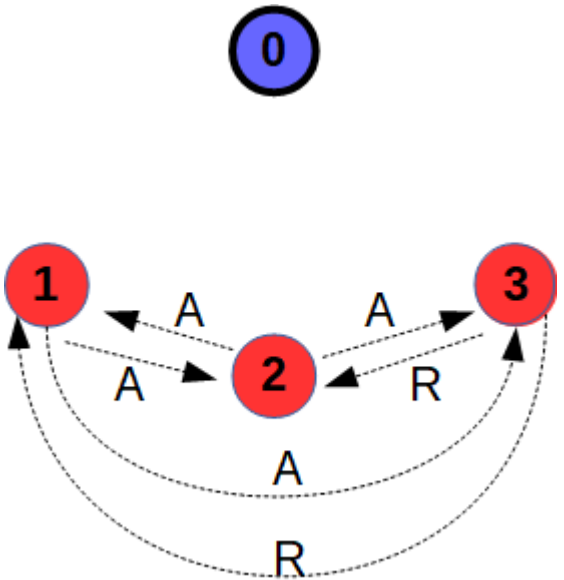
Example2: MOM(1,4) – case 2.1

Case 2.1 (A,A,R)

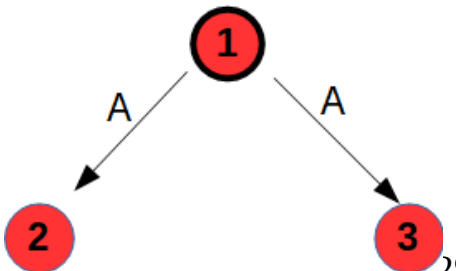
MOM(1,4): Step 1



MOM(1,4): Step 2



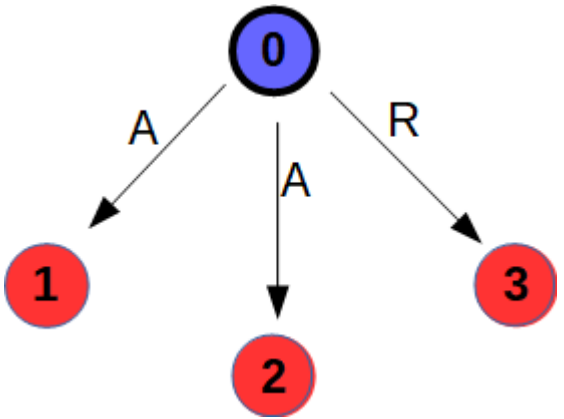
- MOM(1,4):Step 2
  - For G1
  - Do MOM(0,3)
  - $\overline{V}_1$=A
- MOM(1,4):Step 3
  - For G1
  - V_1=majority($\overline{V}_1$,V_2,V_3)
  - =majority(A,A,R)
  - =A
  - For G2 : same to G1



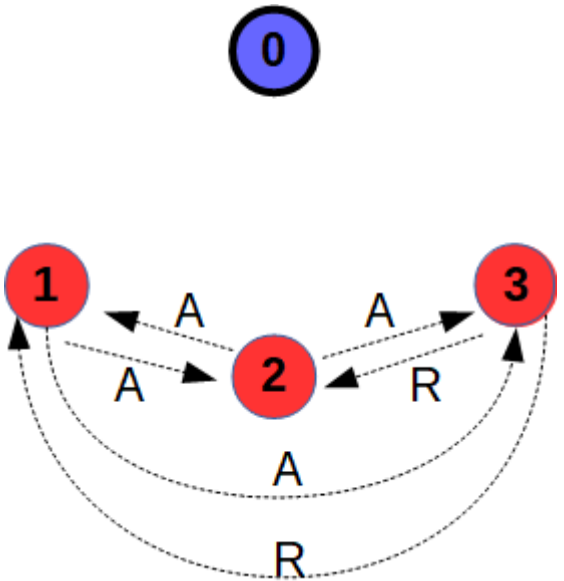
Example2: MOM(1,4) – case 2.1

Case 2.1 (A,A,R)

MOM(1): Step 1



MOM(1): Step 2



- MOM(1,4):Step 2
  - For G3
  - Do MOM(0,3)
  - $\overline{V}_3$=R
- MOM(1,4):Step 3
  - For G3
  - V_3=majority(V_1,V_2,$\overline{V}_3$)
  - =majority(A,A,R)
  - =A
- So, V_1=V_2=V_3=A

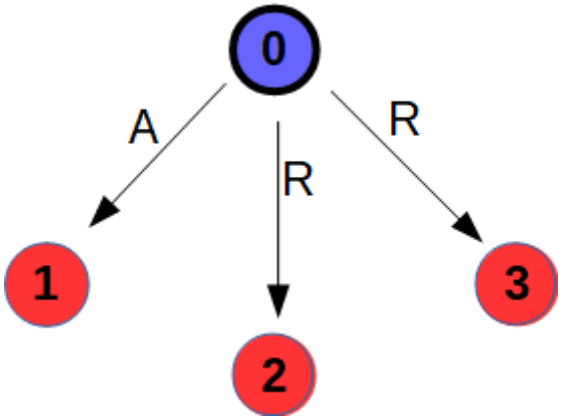
- IC1 ok
- ~~IC2 (G0 is not loyal)~~

Example2: MOM(1,4) – Case 2.2

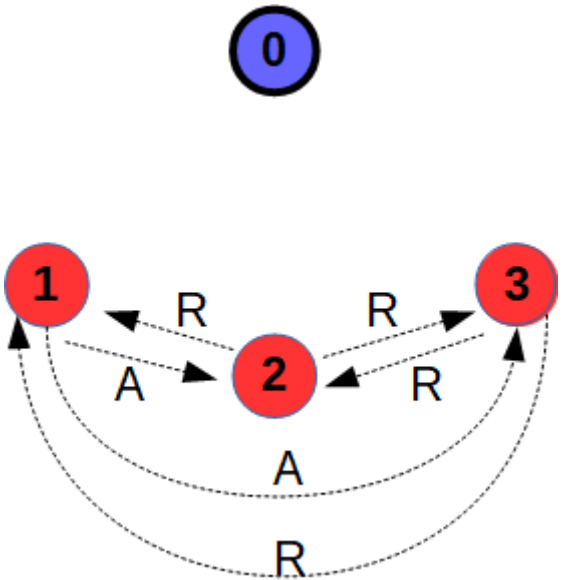
Example2: MOM(1,4) – case 2.2

Case 2.2 (A,R,R)

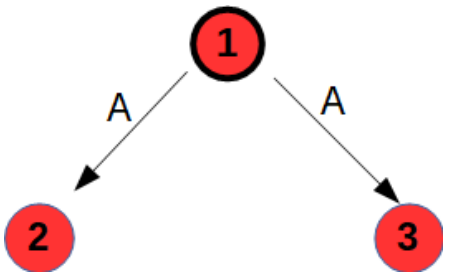
MOM(1,4): Step 1



MOM(1,4): Step 2



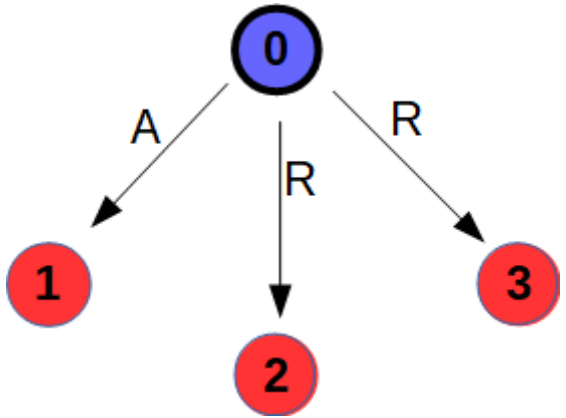
- MOM(1,4):Step 2
- For G1
- Do MOM(0,3)
- $\overline{V}_1$=A
- MOM(1,4):Step 3
- For G1
- V_1=majority($\overline{V}_1$,V_2,V_3)
- =majority(A,R,R)
- =R



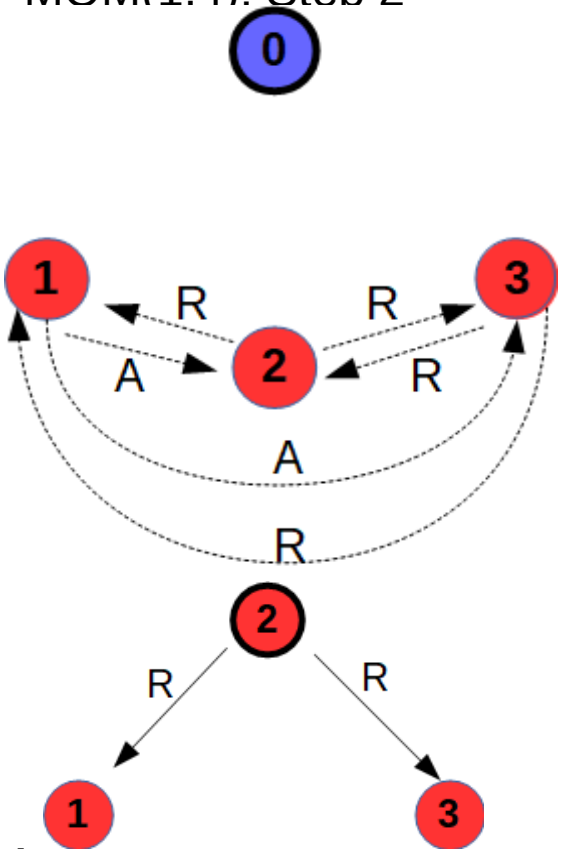
Example2: MOM(1,4) – case 2.2

Case 2.2 (A,R,R)

MOM(1,4): Step 1



MOM(1,4): Step 2



- MOM(1,4):Step 2
  - For G2
  - Do MOM(0,3)
  - $\overline{V}_2$=R
- MOM(1,4):Step 3
  - For G2
  - V_2=majority(V_1,$\overline{V}_2$,V_3)
  - =majority(A,R,R)
  - =R
  - For G3: same to G2
  - So, V_1=V_2=V_3=R

- IC1 ok
- ~~IC2 (G0 is not loyal)~~

Example2: MOM(1,4) – Case 2.3

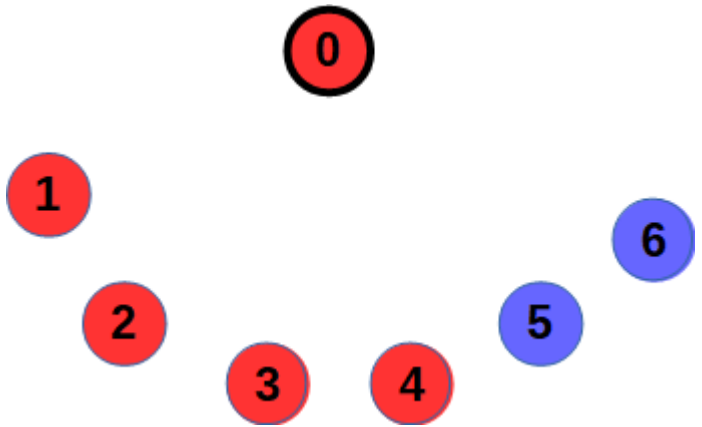
Example2: MOM(1,4) – case 2

- Case 2.3 (R,R,R)
- Case 2.4 (A,A,A)
  - IC1 ok
  - ~~IC2 (G0 is not loyal)~~

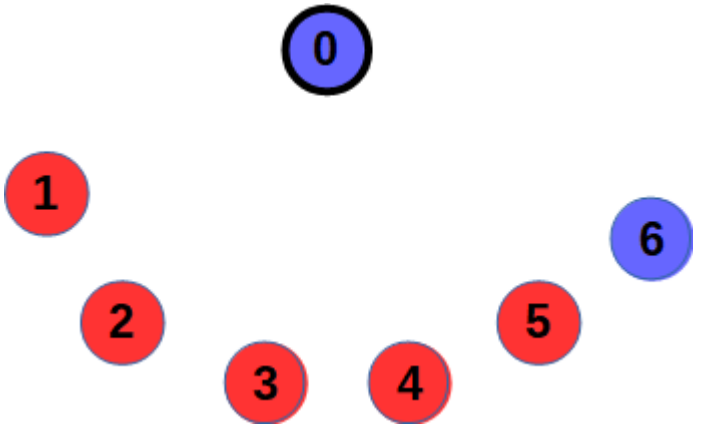
Example3: MOM(2,7)

Example3: MOM(m) (m=2,n=7)- case 3

Case 3.1



Case 3.2

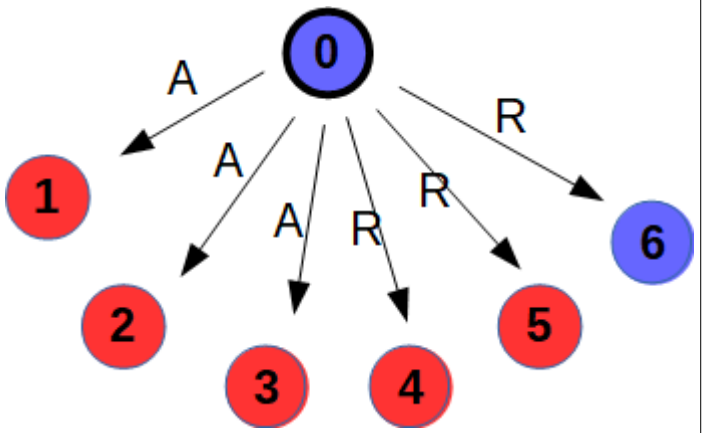


Example3: MOM(2,7) – Case 3.2

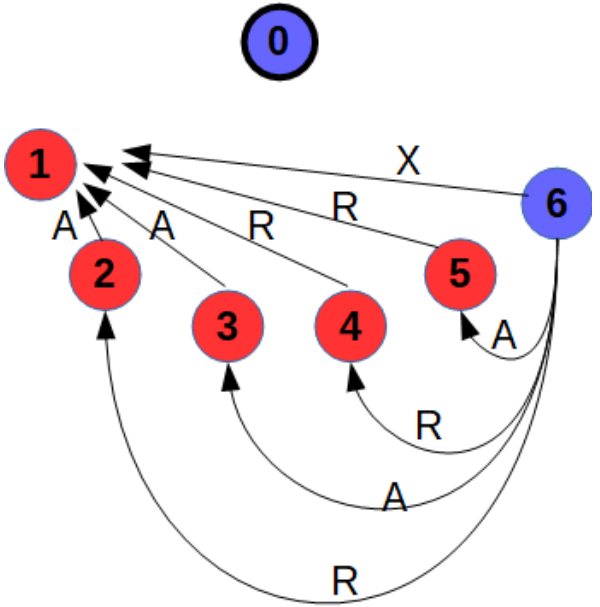
Example3: MOM(2,7)- case 3.2

Case 3.2: Commander is traitor, (A,A,A,R,R,R)the most complex

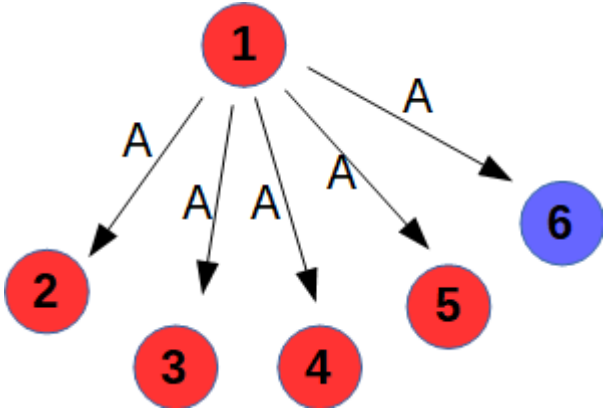
MOM(2,7):Step 1



- MOM(2,7):Step 2
  - *For G1,G2, …,G6*
    - *MOM(1,6)*



- MOM(2,7): Step 2
  - For G1: V_1=A
  - Do MOM(1,6): Based on previous case MOM(1,4)
  - $\overline{V}_1$=A
- MOM(2,7):Step 3
  - For G1
  - V1=majority($\overline{V}_1$,V_2,V_3,…,V_6)
    - =majority(A,A,A,R,R,X)
  - For G2,G3: same to G1

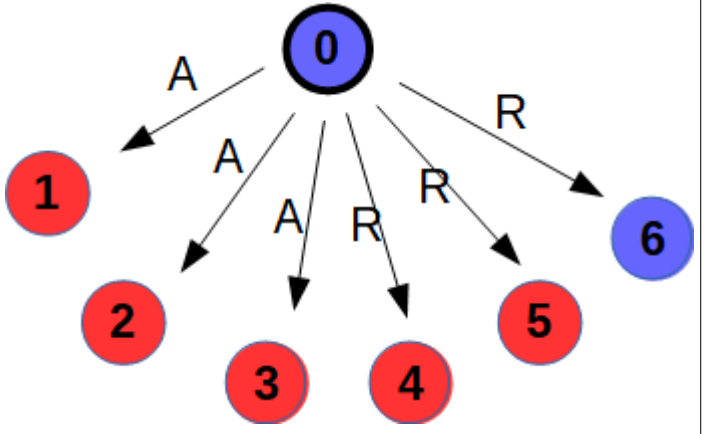


leexudong@nankai.edu.cn

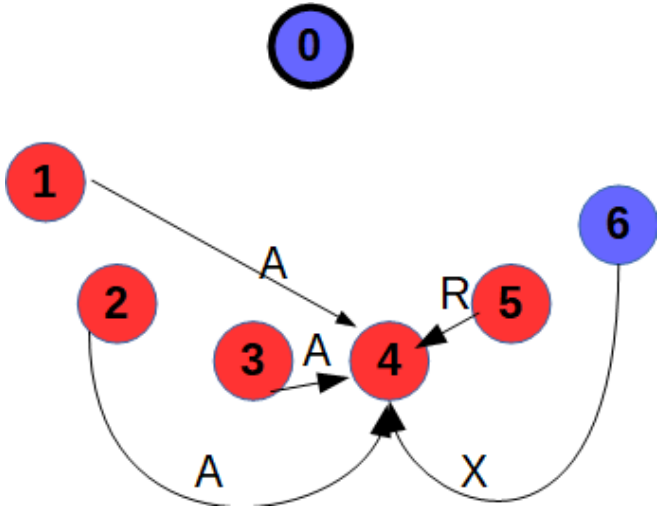
Example3: MOM(2,7)- case 3.2

Case 3.2: Commander is traitor, (A,A,A,R,R,R)the most complex

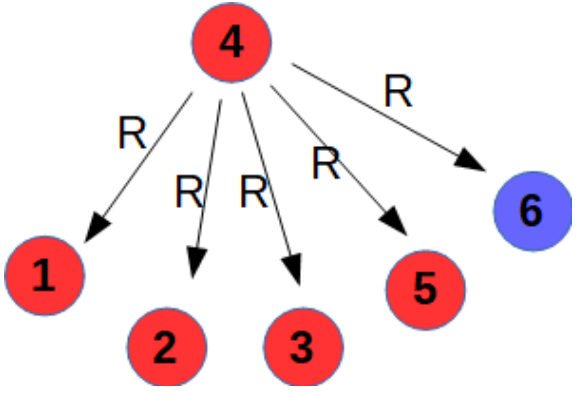
MOM(2,7):Step 1



- MOM(2,7):Step 2
  - *For G1,G2, …,G6*
    - *MOM(1,6)*



- MOM(2,7): Step 2
  - For G4: V_4=R
  - Do MOM(1,6): Based on previous case MOM(1,4)
  - $\overline{V}_4$=R
- MOM(2,7):Step 3
  - For G4
  - V4=majority(V_1,V_2,V_3,$\overline{V}_4$,V_5,V_6)
    - =majority(A,A,A,R,R,X)
    - =V1=V2=V3
  - For G5: same to G4

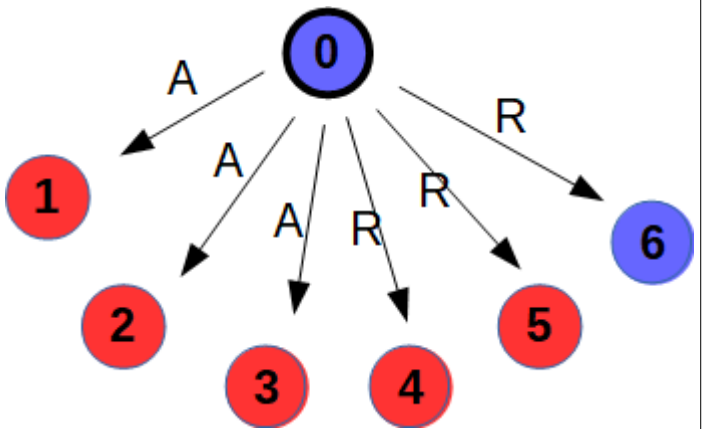


leexudong@nankai.edu.cn

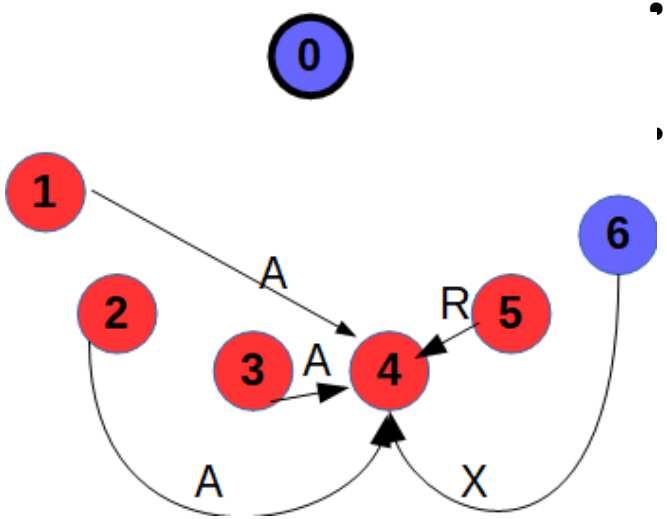
Example3: MOM(2,7)- case 3.2

Case 3.2: Commander is traitor, (A,A,A,R,R,R)the most complex

MOM(2,7):Step 1



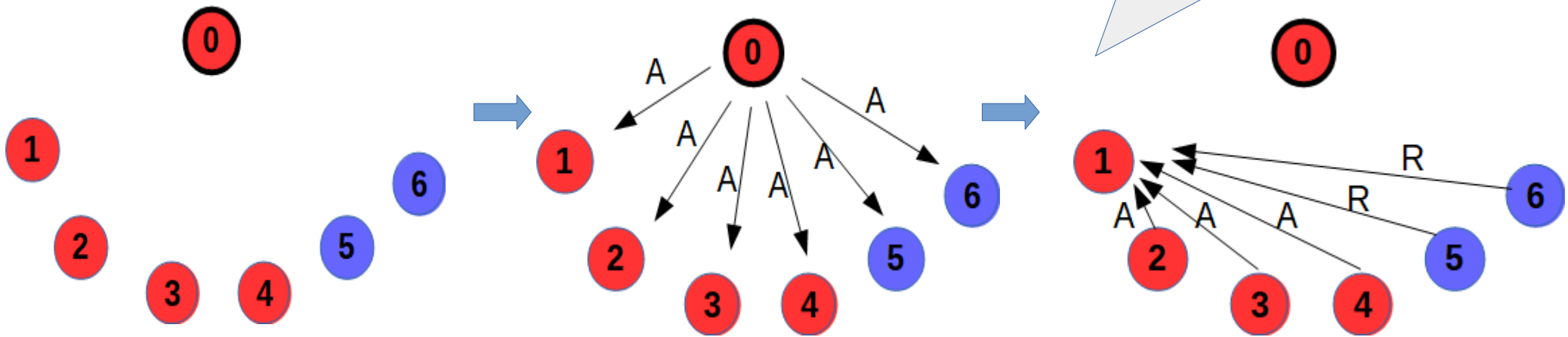
- MOM(2,7):Step 2
  - *For G1,G2, …,G6*
    - *MOM(1,6)*



- MOM(2,7):Step 3
  - For G1,G2,G3,G4,G5
  - V1=V2=V3=V4=V5
    - =majority(A,A,A,R,R,X)
  - For G6: traitor
  - At last
  - $\overline{V}_0$=majority(V1,V2,V3,V4,V5, V6)
    - IC1 ok
    - ~~IC2 (G0 is not loyal)~~

Example3: MOM(2,7)- case 3

- Let's go back case 3.1



- IC1 is OK
- IC2 is OK (All loyal generals make the same and correct decision which is same to commander)

Is it Correct?

OM(m) (m=3,n=10)

OM(m) (m=4,n=13)

OM(m) (m=5,n=13)

...

How to Prove the OM(m) is correct ?

LEMMA 1. *For any m and k, Algorithm OM(m) satisfies IC2 if there are more than 2k + m generals and at most k traitors.*

PROOF. The proof is by induction on m. IC2 only specifies what must happen if the commander is loyal. Using A1, it is easy to see that the trivial algorithm OM(0) works if the commander is loyal, so the lemma is true for $m = 0$. We now assume it is true for $m - 1$, $m > 0$, and prove it for m.

In step (1), the loyal commander sends a value v to all $n - 1$ lieutenants. In step (2), each loyal lieutenant applies OM($m - 1$) with $n - 1$ generals. Since by hypothesis $n > 2k + m$, we have $n - 1 > 2k + (m - 1)$, so we can apply the induction hypothesis to conclude that every loyal lieutenant gets $v_j = v$ for each loyal Lieutenant j. Since there are at most k traitors, and $n - 1 > 2k + (m - 1)$ $\geq 2k$, a majority of the $n - 1$ lieutenants are loyal. Hence, each loyal lieutenant has $v_i = v$ for a majority of the $n - 1$ values i, so he obtains *majority*$(v_1, \ldots, v_{n-1}) = v$ in step (3), proving IC2. □

THEOREM 1. *For any m, Algorithm OM(m) satisfies conditions IC1 and IC2 if there are more than 3m generals and at most m traitors.*

PROOF. The proof is by induction on m. If there are no traitors, then it is easy to see that OM(0) satisfies IC1 and IC2. We therefore assume that the theorem is true for OM($m-1$) and prove it for OM(m), $m > 0$.

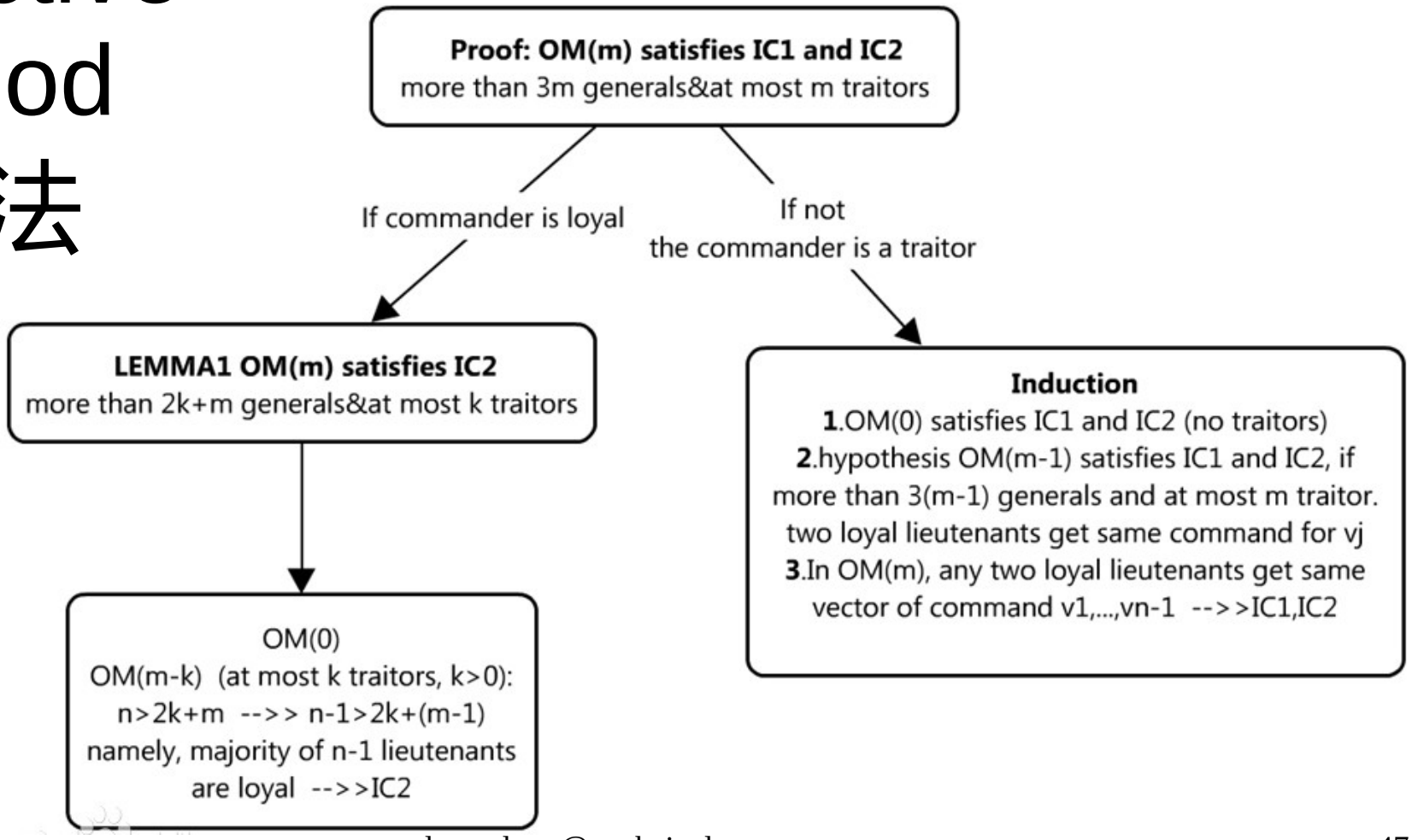
We first consider the case in which the commander is loyal. By taking k equal to m in Lemma 1, we see that OM(m) satisfies IC2. IC1 follows from IC2 if the commander is loyal, so we need only verify IC1 in the case that the commander is a traitor.

There are at most m traitors, and the commander is one of them, so at most $m-1$ of the lieutenants are traitors. Since there are more than $3m$ generals, there are more than $3m-1$ lieutenants, and $3m-1 > 3(m-1)$. We may therefore apply the induction hypothesis to conclude that OM($m-1$) satisfies conditions IC1 and IC2. Hence, for each j, any two loyal lieutenants get the same value for v_j in step (3). (This follows from IC2 if one of the two lieutenants is Lieutenant j, and from IC1 otherwise.) Hence, any two loyal lieutenants get the same vector of values $v_1, \ldots, v_{n-1}$, and therefore obtain the same value *majority*$(v_1, \ldots, v_{n-1})$ in step (3), proving IC1. □

Inductive method 归纳法

- MOM(m,n): n>=3m+1
  - If m=0: ok;
  - If m=1,n=4: ok
  - So, suppose m=k is ok
  - For m=k+1
    - 3(k+1)+1=3k+3+1=(3k+1)+3
    - It means:
      - Same to: has 1 traitor, has 4 generals
      - MOM(1,4) is ok

Inductive method 归纳法



Advantages of OM(m)

- Tolerate Arbitrary Faults
  - Reliability on distributed systems
  - Failed components
  - Conflicting information
- Get Consistence Result

Disadvantages of OM(m)

- Computation Complex
- Synchronous Mechanism

Alternative Solutions ?

Lamport:
Signed Messages Algorithm 签名消息

- Add The Fourth Condition:
  - **A4 (a) A loyal general's signature cannot be forged, and any alteration of the contents of his signed messages can be detected;**
  - **A4 (b) Anyone can verify the authenticity of a general's signature**

Lamport: Signed Messages Algorithm

- Signed Messages Algorithm
  - SM(m)
  - Choice(v_1, v_2, … , v_{n-1})

Our algorithm assumes a function *choice* which is applied to a set of orders to obtain a single one. The only requirements we make for this function are

1. If the set V consists of the single element v, then $choice(V) = v$.
2. $choice(\emptyset)$ = RETREAT, where $\emptyset$ is the empty set.

Note that one possible definition is to let $choice(V)$ be the median element of V—assuming that there is an ordering of the elements.

Lamport: Signed Messages Algorithm

Initially $V_i = \emptyset$.

(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 If Gi is loyal, Gi should authenticate the message firstly

(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 $v:0:j_1:\cdots:j_k$ and v is not in the set V_i, then Gi should authenticate the message firstly

(i) he adds v to V_i;
(ii) if $k < m$, then he sends the message $v:0:j_1:\cdots:j_k:i$ 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 $choice(V_i)$.

Note that in step (2), Lieutenant i ignores any message containing an order v that is already in the set V_i.

Lamport: Signed Messages Algorithm

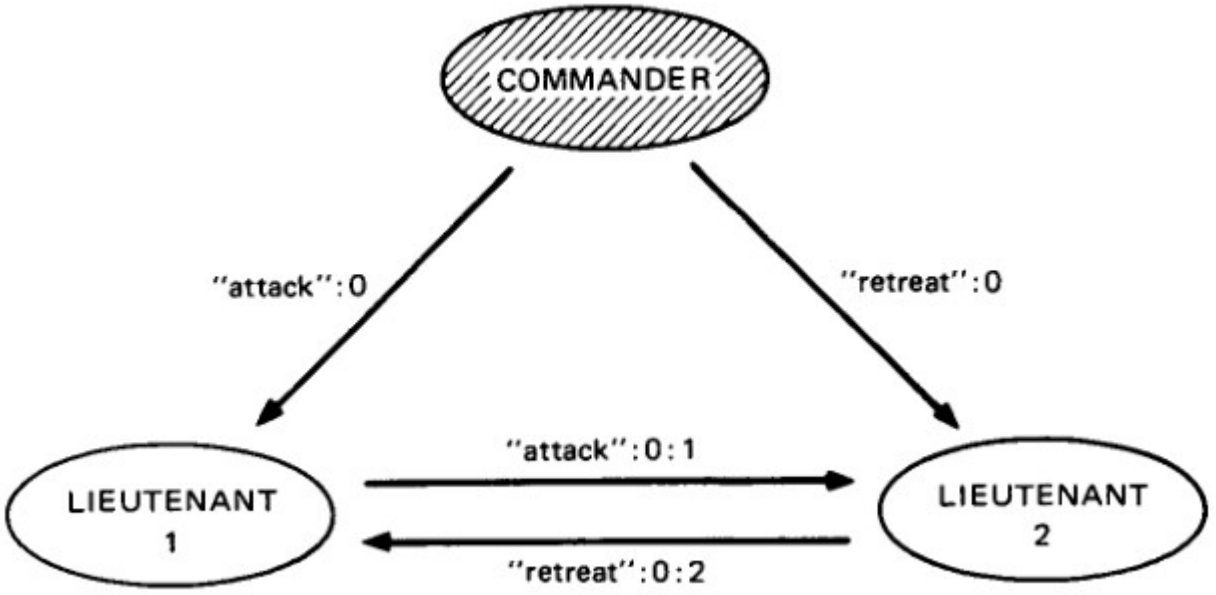


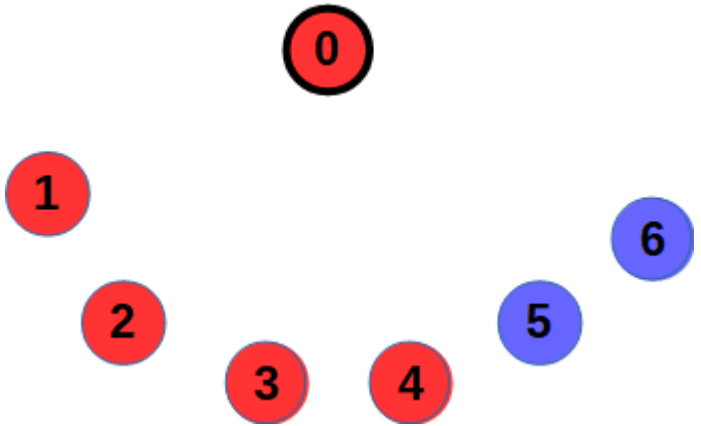
Fig. 5. Algorithm SM(1); the commander a traitor.

- L1 :
  - Step1: V1={A:0}
  - Step2: V1={A:0,R:0:2}
    - V1={A,R}
  - Step3: choice(V1)
- L2:
  - Step1: V2={R:0}
  - Step2: V2={R:0,A:0:1}
    - V2={A,R}
  - Step3: choice(V2)

Lamport: Signed Messages Algorithm

m=2, n=7

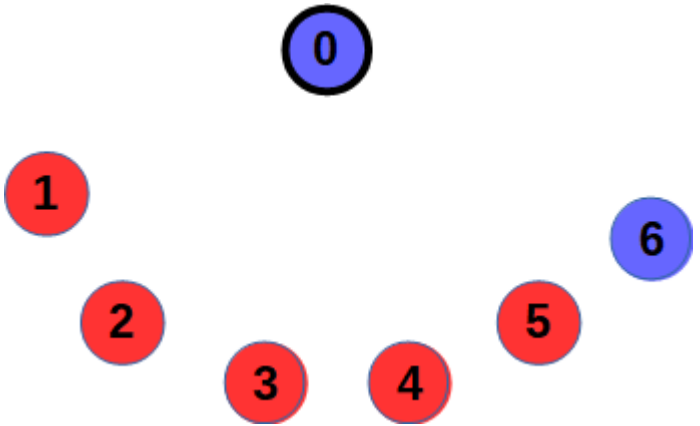
Case 3.1



Lamport: Signed Messages Algorithm

m=2, n=7

Case 3.2



Alternative Solutions ?

Summary

References

- Lamport, et al. "The Byzantine Generals Problem." Acm Transactions on Programming Languages & Systems 4.3(1982):382-401.
- Castro, Miguel, and B. Liskov. "Practical Byzantine fault tolerance." Symposium on Operating Systems Design & Implementation 1999.
- Aublin, Pierre Louis , S. B. Mokhtar , and V. Quema . "RBFT: Redundant Byzantine Fault Tolerance." 2013 IEEE 33rd International Conference on Distributed Computing Systems IEEE Computer Society, 2013.

Q&A ?

Write All Steps with MOM(3,10)

Thanks!