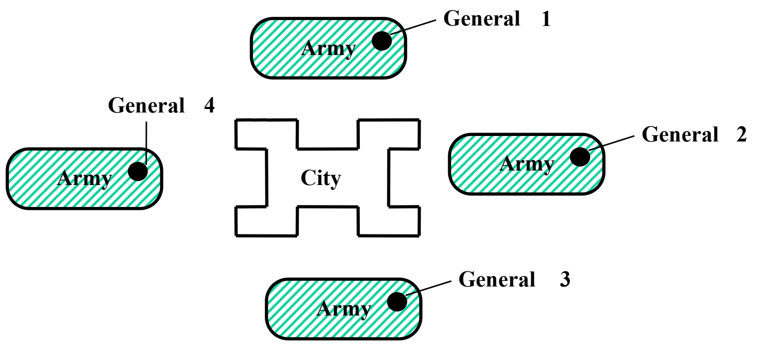
The Byzantine Generals Problem
(Consensus in the presence of uncertainties)



All loyal generals must agree on the same plan of action (attack or retreat) despite the presence of traitors. Generals can communicate only by message passing. Traitors may do anything they wish.

Problem definition:

A commanding general must send an order to his n-1 lieutenant generals such that:

{ Given a network of n processes which can communicate with one another only by means of messages over bi-directional channels - ensure that a process sends an item of data to n-1 others such that}

IC1: All loyal lieutenants obey the same order.
{reliably operating processes receive the same item}

IC2: If the commanding general is loyal, then every loyal lieutenant obeys the order he sends.
{if the sending process is operating reliably then the item received is identical to the item sent}

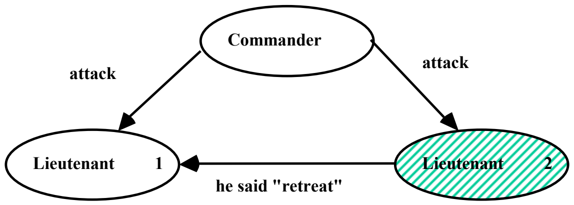
Interactive Consistency

IC1 & IC2 are known as the **interactive consistency** conditions. Note that if the commander is loyal IC1 follows from IC2. However the commander may be a traitor.

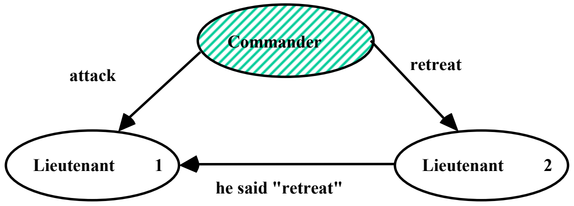
{ The implication for computing systems is that a solution to the Byzantine generals problem allows reliable communication in the presence of commission errors ias well as omission errors. Handling only omission is the more usual case (fail-stop model} as in the 2-phase commit protocol}

Impossibility Results

Consider the following two cases with 3 generals:



Case A - Lieutenant 2 is a Traitor



Case B - Commander is a Traitor

In case a) to satisfy IC2 Lieutenant 1 should attack

In case b) if Lieutenant 1 attacks he violates IC1.

Lieutenant 1 cannot distinguish, from the information available to him, between case a) & case b).

No solution exists for three generals that works in the presence of a single traitor.

General Impossibility Result:
No solution with fewer than **3m+1** generals can cope with **m** traitors.

A Solution with Unsigned Messages:

[Lamport L, Shostak R, Pease M, The Byzantine Generals Problem, ACM TOPLAS 4(3) (July 1982) pp382-401.]

Message Passing Assumptions:

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.

Assumptions A1 and A2 prevent a traitor from interfering with the communication between two other generals, since by A1 he cannot interfere with the messages they do send, and by A2 he cannot confuse their intercourse by introducing spurious messages. A3 foils a traitor who tries to prevent a decision by simply not sending messages.

{ For computers systems A1 & A2 imply that the algorithm works for processors directly connected by point to point links and that a link failure counts as one of the m failures since it is indistinguishable from a processor failure. A3 requires that senders and receivers have clocks synchronised to some maximum error and that the maximum message generation and transmission time is known. }

The Algorithm: For **n** generals and **m** traitors **(n>3m)**

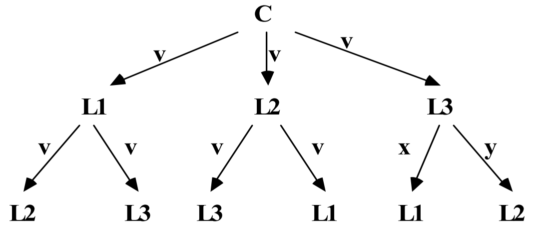
Require default value **v_{def}** if traitorous commander does not send a message (e.g. RETREAT)
Define function **majority(v_1,...,v_{n-1}) = v** if a majority of the values **v_i = v**.

Algorithm UM(n,0) # no traitors case
(1) The Commander sends **v** to every lieutenant.
(2) Each lieutentent uses the value received from the commander or **v_{def}** if he receives no value.

Algorithm UM(n,m) # **m** traitors case
(1) The Commander sends **v** to every lieutenant.
(2) Foreach **Lieutenant$_i$** ,
let **v_i** = value received from commander
or **v_{def}** if no value received.
send **v_i** to n-2 other lieutenants using **UM(n-1,m-1)**
(3) Foreach **i** & each **j _ i**,
let **v_j** = value Lieutenant$_i$ received from Lieutenant$_j$
in step (2) or **v_{def}** if no value received.

Lieutenant$_i$ uses the value **majority(v_1,...,v_{n-1})**

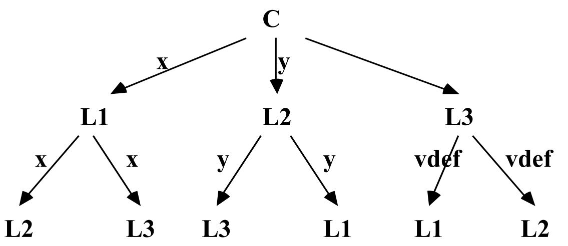
Example: **n=4 m=1 Um(4,1)** First case: L3 is a traitor.



At the end of stage 1:
L1: v_1 = v
L2: v_2 = v
L3: v_3 = v
At the end of stage 2:
L1: v_1 = v, v_2 = v, v_3 = x
L2: v_1 = v, v_2 = v, v_3 = y
L3: v_1 = v, v_2 = v, v_3 = v

At the end of Stage 2 each of the lieutenants has received a set of values and arrives at the same decision (IC1) ; and the value sent by C is the majority value (IC2).

Example: n=4 m=1 UM(4,1) Second case: C is a traitor.



At the end of stage 1:

L1: $v_1 = x$
L2: $v_2 = y$
L3: $v_3 = v_{def}$

At the end of stage 2:

L1: $v_1 = x$, $v_2 = y$, $v_3 = v_{def}$
L2: $v_1 = x$, $v_2 = y$, $v_3 = v_{def}$
L3: $v_1 = x$, $v_2 = y$, $v_3 = v_{def}$

The three loyal lieutenants receive the same value majority(x,y,v_{def}) and the constraints IC1 & IC2 are respected.

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

PROOF: (by induction on **m**)

From **A1** it is obvious that **UM(0)** works if the commander is loyal. i.e. **UM(0)** satisfies **IC2**.

Now assume **UM(m-1)** satisfies **IC2** for **m>0** and prove it for **m**.

In step(1), the loyal commander sends a value **v** to **n-1** lieutenants. In step(2) each loyal lieutenant applies **UM(m-1)**.

- By hypothesis we have **n>2k+m** or **n-1>2k+(m-1)**.
- By the induction hypothesis, every loyal lieutenant gets $v_j = v$ from each loyal lieutenant **j**.
- Since there are at most **k** traitors and **n-1 > 2k + (m-1) >= 2k** i.e. **k< (n-1)/2** 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 so he obtains **majority(v_i,..v_{n-1})=v** in step(3) satisfying **IC2**.

THEOREM: *For any **m**, **UM(m)** satisfies **IC1 & IC2** if there are more than **3m** generals and at most **m** traitors.*

PROOF: (by induction on **m**)

If there are no traitors it is easy to see using **A1** that **UM(0)** satisfies **IC1 & IC2**.

Now assume **UM(m-1)** satisfies **IC1& IC2** for **m>0** and prove it for **m**.

case A) - assume the commander is loyal.

- by taking **k = m** in the LEMMA, **UM(m)** satisfies **IC2**.
  Since **IC1** follows from **IC2** if the commander is loyal, we now only consider :-

case B) - the commander is a traitor

- there are at most **m** traitors and the commander is a traitor, therefore at most **m-1** of the lieutenants are traitors. Since there are more than **3m** generals there must be more than **3m-1** lieutenants and **3m - 1 > 3(m - 1)**.
- Hence we can apply the induction hypothesis to conclude that **UM(m-1)** satifies **IC1 & IC2**.
- Hence, for each **j**, any two loyal lieutenants get the same value for v_j in step(3). (follows from **IC2** if one of the two lieutenants is **j**, from **IC1** otherwise.)
- Hence, any two lieutenants get the same vector of values and therefore the same **majority(v_i,..v_{n-1})** in step(3), proving **IC1**.

Complexity of UM(n,m)

Applying **UM(n,m)** first causes the issuing of **n-1** messages. Each message invokes **UM(n-1,m-1)** which causes **n-2** messages to be issued etc.

Stage 1: **(n-1)** messages
Stage 2: **(n-1)(n-2)** messages
• • • • • • • •
Stage m+1: **(n-1)(n-2). . .(n-(m+1))** messages

Total messages is **$O(n^{m+1})$.**

*Note: The **m+1** stages of message exchange is a fundamental characteristic of algorithms which arrive at a consensus in the presence of **m** possible faulty processes.*

Identifying messages

Messages can be unambiguously identified (stage, recursive call) by postfixing the message with the process that sent it.

e.g. **v:C,L1** (see tree diagrams).

At stage **k** the message length will be value **+ k** identifiers

A Solution with Signed Messages:

Restrict traitor's ability to lie by allowing generals to send unforgeable signed messages.

Additional message passing assumption:

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

Notation: **v:j:i** - value **v** signed by **j** and then value **v:j** signed by **i**. General_0 is the commander.

Need function **choice(V)** which selects a value **v** from a set of values **V** such that:

choice({v}) = v;
choice({}) = $\mathbf{v_{def}}$**;**

Note that choice is used to obtain the consensus value, it does not have to be majority or median value.

The Algorithm: For **n** generals and **m** traitors where **n** may be any number (although the problem is vacuous for **n<m+2**).

In the following, each lieutenant **i** maintains a set $\mathbf{V_i}$ of properly signed orders he has so far received. (With a loyal commander the set does not contain more than a single element).

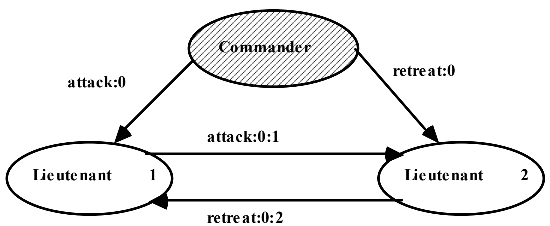
Algorithm SM(m) Initally $\mathbf{V_i}$ **= {}**
(1) Commander sends his signed value to every lieutenant.

(2) Foreach **i**:
- **(A)** If lieutenant **i** receives a message **v:0** and he has not yet received an order,then:
  - (i) sets $\mathbf{V_i}$ to **{v}**
  - (ii) sends **v:0:i** to every other lieutenant.
- **(B)** If lieutenant **i** receives a message of the form $\mathbf{v:0:j_1: \ldots :j_k}$ and **v** is not in the set $\mathbf{V_i}$ then
  - (i) $\mathbf{V_i := V_i + \{v\}}$
  - (ii) if **k<m** then send the message $\mathbf{v:0:j_1: \ldots :j_k:i}$ to every lieutenant other than $\mathbf{j_1, \ldots, j_k}$.

(3) Foreach **i**:
When no more messages lieutenant **i** obeys the order **choice(**$\mathbf{V_i}$**)**.

Example: SM(1)

Commander is a Traitor



V1 = V2 = choice({attack,retreat})

Note: with signed messages, the lieutenants can detect the commander is a traitor since his signature appears on two different orders and by A4 only he could have signed them.

Complexity: No of messages: $\mathbf{O(n^{m+1})}$ No of stages: **m+1**

[Note: An $O(n^2)$ messages algorithm using signed messages is developed in: *Dolev, D., & Strong, H., R., "Authenticated Algorithms for Byzantine Agreement", SIAM Journal of Computer, 12(4) (1983), pp 656-666.* The reduction is achieved by a process only retransmitting values which it has not previously sent. Still requires m+1 stages.]