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 as 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.

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) = 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 lieutenant 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) For each Lieutenant, 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) For each i & each j _ i, let v_j = value Lieutenant, received from Lieutenant_i in step (2) or v_def if no value received. Lieutenant_i uses the value majority(v_1,...,v_n).

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 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: v1 = x
  - L2: v2 = y
  - L3: v3 = vdef

- At the end of stage 2:
  - L1: v1 = x, v2 = y, v3 = vdef
  - L2: v1 = x, v2 = y, v3 = vdef
  - L3: v1 = x, v2 = y, v3 = vdef

The three loyal lieutenants receive the same value majority(x,y,vdef) 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.

At stage 1:
- (n-1) messages

At stage 2:
- (n-1)(n-2) messages

... ...

At stage m+1:
- (n-1)(n-2) . . . .(n-(m+1)) messages

Total messages is O(nm+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.

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 A, 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) satisfies 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_j,...v_n) 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(nm+1).

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 detected.
(b) Anyone can verify the authenticity of a general’s signature.

Notation: \( v:j:1 \) - value \( v \) signed by \( j \) and then value \( v:j \) signed by \( i \). General 0 is the commander.

Need function \( \text{choice}(V) \) which selects a value \( v \) from a set of values \( V \) such that:

\[
\text{choice}(\{v\}) = v; \\
\text{choice}(\{} = v_{\text{def}}.
\]

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 \( V_i \) of properly signed orders he has so far received. (With a loyal commander the set does not contain more than a single element).

Algorithm \( \text{SM}(m) \): Initially \( V_i = {} \).

1. Commander sends his signed value to every lieutenant.

2. (A) If lieutenant \( i \) receives a message \( v:0 \) and he has not yet received an order, then:
   (i) sets \( V_i \) to \( \{v\} \)
   (ii) sends \( v:0:i \) to every other lieutenant.

   (B) If lieutenant \( i \) receives a message of the form \( v:0:j_1:...:j_k \) and \( v \) is not in the set \( V_i \) then
   (i) \( V_i := V_i + \{v\} \)
   (ii) if \( k < m \), then send the message \( v:0:j_1:...:j_k:i \) to every lieutenant other than \( j_1, ..., j_k \).

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

Example: \( \text{SM}(1) \)

Commander is a Traitor

![Diagram showing the interaction between the commander and lieutenants](image)

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: \( O(n^{m+1}) \) No of stages: \( m+1 \)

[Note: An \( O(n^2) \) messages algorithm using signed messages is developed in: Dolev, D., & Strong, H., 1983, "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.]