

# Agreement Protocols

*CS60002: Distributed Systems*

Pallab Dasgupta  
Professor,  
Dept. of Computer Sc. & Engg.,  
Indian Institute of Technology Kharagpur



# Classification of Faults

- **Based on components that failed**
  - **Program / process**
  - **Processor / machine**
  - **Link**
  - **Storage**
  - **Clock**
  
- **Based on behavior of faulty component**
  - **Crash – just halts**
  - **Failstop – crash with additional conditions**
  - **Omission – fails to perform some steps**
  - **Byzantine – behaves arbitrarily**
  - **Timing – violates timing constraints**

# Classification of Tolerance

- **Types of tolerance:**
  - **Masking** – system always behaves as per specifications even in presence of faults
  - **Non-masking** – system may violate specifications in presence of faults. Should at least behave in a well-defined manner
- **Fault tolerant system should specify:**
  - **Class of faults tolerated**
  - **What tolerance is given from each class**

# Core problems

- **Agreement (multiple processes agree on some value)**
- **Clock synchronization**
- **Stable storage (data accessible after crash)**
- **Reliable communication (point-to-point, broadcast, multicast)**
- **Atomic actions**

# Overview of Consensus Results

- Let  $f$  be the maximum number of faulty processors.
- Tight bounds for message passing:

	Crash failures	Byzantine failures
<b>Number of rounds</b>	$f + 1$	$f + 1$
<b>Total number of processors</b>	$f + 1$	$3f + 1$
<b>Message size</b>	polynomial	polynomial

# Overview of Consensus Results

- *Impossible* in asynchronous case.
  - Even if we only want to tolerate a single crash failure.
  - True both for message passing and shared read-write memory.

# Consensus Algorithm for Crash Failures

## Code for each processor:

$v :=$  my input

at each round 1 through  $f+1$ :

if I have not yet sent  $v$  then send  $v$  to all

wait to receive messages for this round

$v :=$  minimum among all received values and

current value of  $v$

if this is round  $f+1$  then decide on  $v$

# Correctness of Crash Consensus Algo

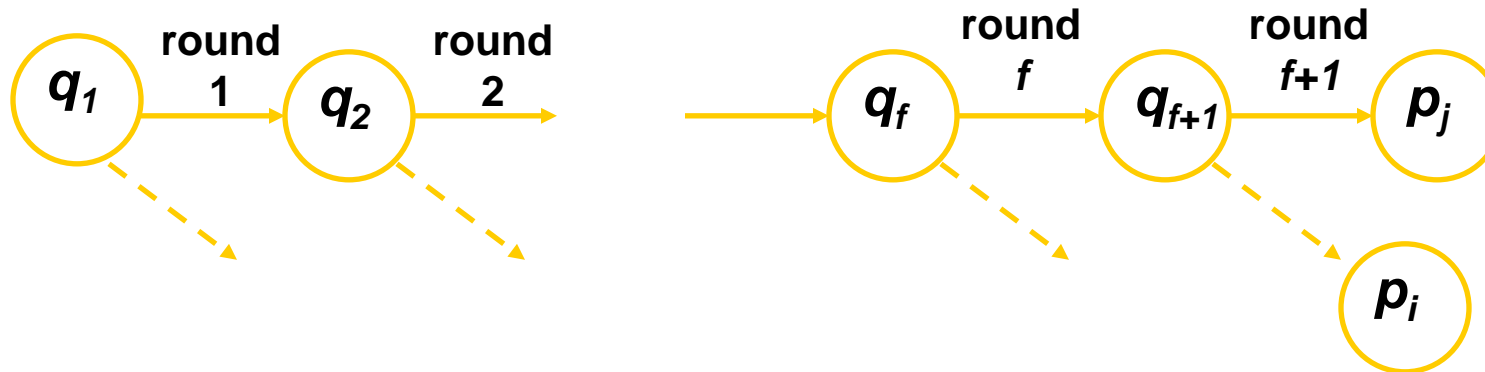
- **Termination:** By the code, finish in round  $f + 1$ .
- **Validity:** Holds since processors do not introduce spurious messages
  - if all inputs are the same, then that is the only value ever in circulation.



# Correctness of Crash Consensus Algo

## Agreement:

- Suppose in contradiction  $p_j$  decides on a smaller value,  $x$ , than does  $p_i$ .
- Then  $x$  was hidden from  $p_i$  by a chain of faulty processors:



- There are  $f + 1$  faulty processors in this chain, a contradiction.

# Performance of Crash Consensus Algo

- Number of processors  $n > f$
- $f + 1$  rounds
- $n^2 \cdot |V|$  messages, each of size  $\log|V|$  bits, where  $V$  is the input set.

# Lower Bound on Rounds

## Assumptions:

- $n > f + 1$
- every processor is supposed to send a message to every other processor in every round
- Input set is  $\{0,1\}$

# Byzantine Agreement Problems

## Model :

- Total of  $n$  processes, at most  $m$  of which can be faulty
- Reliable communication medium
- Fully connected
- Receiver always knows the identity of the sender of a message
- Byzantine faults
- Synchronous system
  - In each round, a process receives messages, performs computation, and sends messages.

# Byzantine Agreement

- Also known as Byzantine Generals problem
  - One process  $x$  broadcasts a value  $v$ 
    - Agreement Condition: All non-faulty processes must agree on a common value.
    - Validity Condition: The agreed upon value must be  $v$  if  $x$  is non-faulty.

# Variants

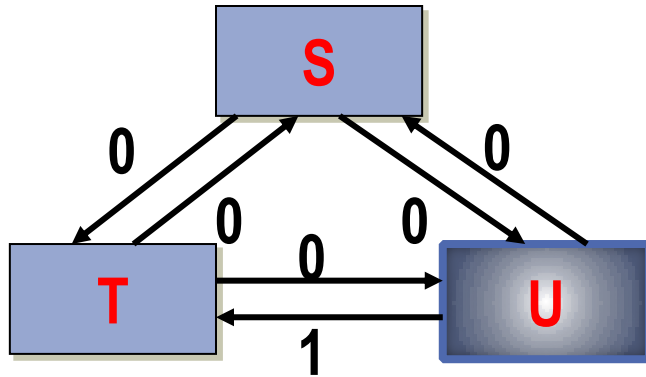
- Consensus
  - Each process broadcasts its initial value
    - Satisfy agreement condition
    - If initial value of all non-faulty processes is  $v$ , then the agreed upon value must be  $v$
- Interactive Consistency
  - Each process  $k$  broadcasts its own value  $v_k$ 
    - All non-faulty processes agree on a common vector  $(v_1, v_2, \dots, v_n)$
    - If the  $k^{\text{th}}$  process is non-faulty, then the  $k^{\text{th}}$  value in the vector agreed upon by non-faulty processes must be  $v_k$
- *Solution to Byzantine agreement problem implies solution to other two*

# Byzantine Agreement Problem

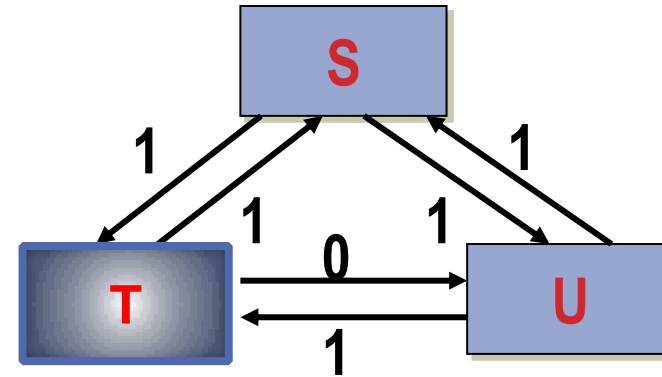
- No solution possible if:
  - asynchronous system, or
  - $n < (3m + 1)$
- Lower Bound:
  - Needs at least  $(m+1)$  rounds of message exchanges
- “*Oral*” messages – messages can be forged / changed in any manner, but the receiver always knows the sender

# Proof

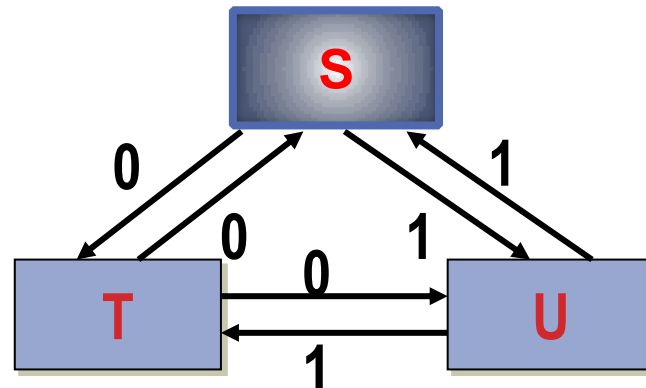
**Theorem:** *There is no  $t$ -Byzantine-robust broadcast protocol for  $t \geq N/3$*



**Scenario-0: T must decide 0**



**Scenario-1: U must decide 1**



**Scenario-2:**

- similar to Scenario-0 for T
- similar to Scenario-1 for U
- T decides 0 and U decides 1





# Lamport-Shostak-Pease Algorithm contd..

For  $t > 0$ , Broadcast(  $N, t$  ):

## Pulse

- 1 The general sends  $\langle \text{value}, x_g \rangle$  to all processes, the lieutenants do not send.  
Receive messages of pulse 1.  
Lieutenant  $p$  acts as follows:  
if a message  $\langle \text{value}, x \rangle$  was received from  $g$  in pulse-1 then  $x_p = x$  else  $x_p = \text{undef}$  ;  
Announce  $x_p$  to the other lieutenants by acting as a general in  
 $\text{Broadcast}_p( N - 1, t - 1 )$  in the next pulse

## Pulse

- $t+1$  Receive messages of pulse  $t+1$ .  
The general decides on  $x_g$ .  
For lieutenant  $p$ :  
A decision occurs in  
 $\text{Broadcast}_q( N - 1, t - 1 )$  for each lieutenant  $q$   
 $W_p[q] = \text{decision in } \text{Broadcast}_q( N - 1, t - 1 )$   
 $y_p = \max (W_p)$

# Features

- Termination: If *Broadcast*(  $N, t$  ) is started in pulse 1, every process decides in pulse  $t + 1$
- Dependence: If the general is correct, if there are  $f$  faulty processes, and if  $N > 2f + t$ , then all correct processes decide on the input of the general
- Agreement: All correct processes decide on the same value

*The Broadcast(  $N, t$  ) protocol is a  $t$ -Byzantine-robust broadcast protocol for  $t < N/3$*

**Time complexity:  $O( t + 1 )$     Message complexity:  $O( N^t )$**