Network Algorithms and Complexity

Agreement in Unreliable Distributed Systems

Aris Pagourtzis, Dimitris Sakavalas

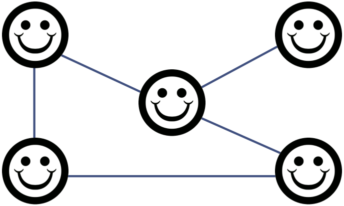
CoReLab, NTUA

Introduction

Distributed Computing
In an unreliable environment

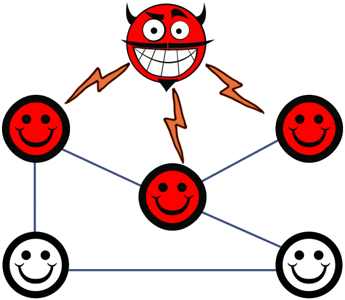
- Several interacting entities (players/agents) that cooperate to achieve a common goal in the absence of a central authority.

Distributed Computing
In an unreliable environment



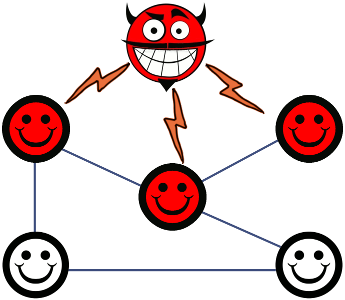
- Several interacting entities (players/agents) that cooperate to achieve a common goal in the absence of a central authority.
- Players arranged in a communication network G.

Distributed Computing
In an unreliable environment



- Several interacting entities (players/agents) that cooperate to achieve a common goal in the absence of a central authority.
- Players arranged in a communication network G.
- Central adversary corrupts/controls players and makes them misbehave (e.g. false messages, crash).

Distributed Computing
In an unreliable environment



- Several interacting entities (players/agents) that cooperate to achieve a common goal in the absence of a central authority.
- Players arranged in a communication network G.
- Central adversary corrupts/controls players and makes them misbehave (e.g. false messages, crash).
- Goal: Achieve common goal despite the presence of corruptions.

Agreement under corruptions

Two major variations of the problem [Lamport, Shostak, Pease, 1982]

Broadcast (Byzantine Generals)

The goal is to have some designated player, called the dealer, consistently send a message to all other players.

Agreement under corruptions

Two major variations of the problem [Lamport, Shostak, Pease, 1982]

Broadcast (Byzantine Generals)

The goal is to have some designated player, called the dealer, consistently send a message to all other players.

"Consistently": All non-corrupted players agree on the same value.

Agreement under corruptions

Two major variations of the problem [Lamport, Shostak, Pease, 1982]

Broadcast (Byzantine Generals)

The goal is to have some designated player, called the dealer, consistently send a message to all other players.

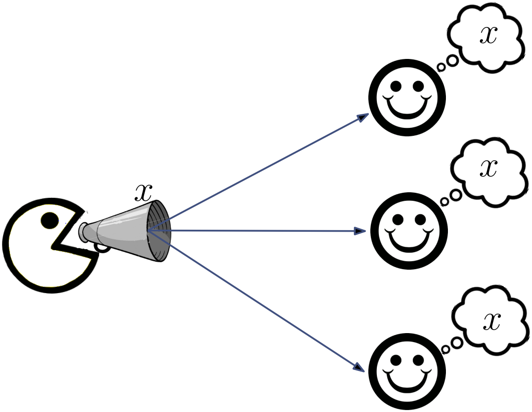
"Consistently": All non-corrupted players agree on the same value.

Consensus (Byzantine Agreement)

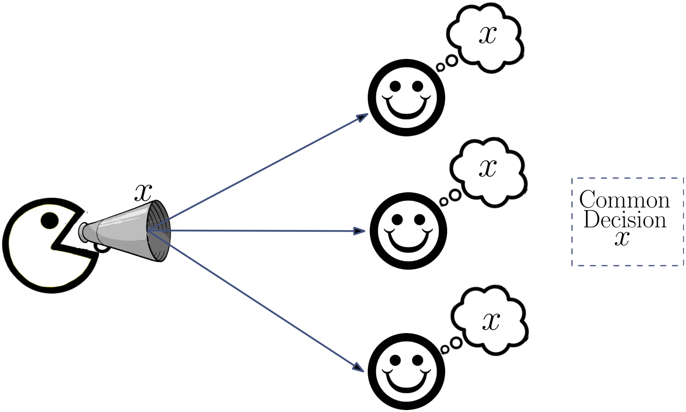
Goal: Make all players agree on the same output value given that every player starts with an input value.

If all correct players hold the same input value then the output value is required to be the same as this input value.

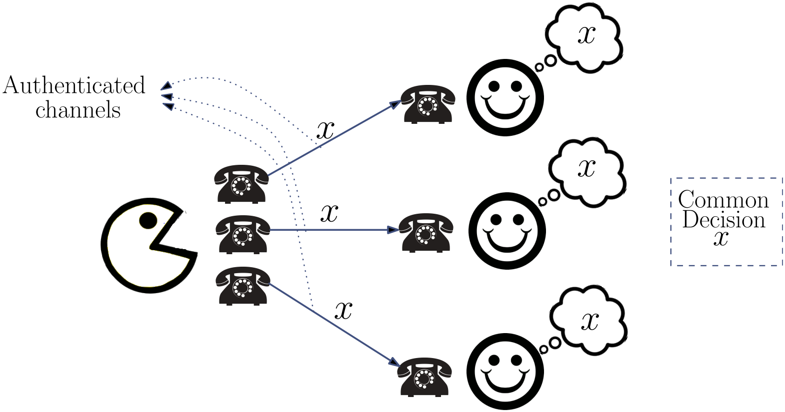
Ideal Broadcast



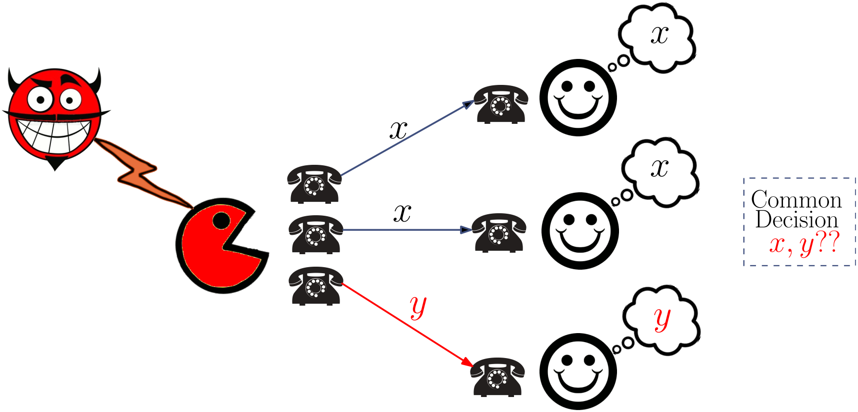
Ideal Broadcast



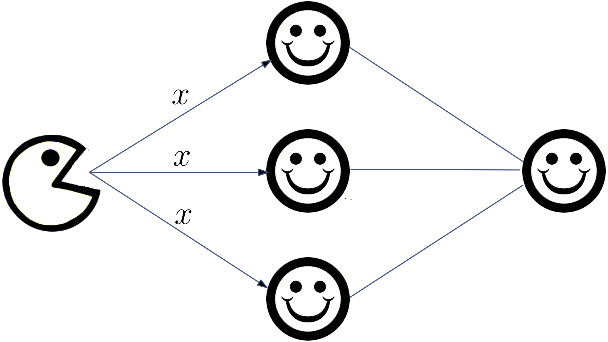
Real Broadcast



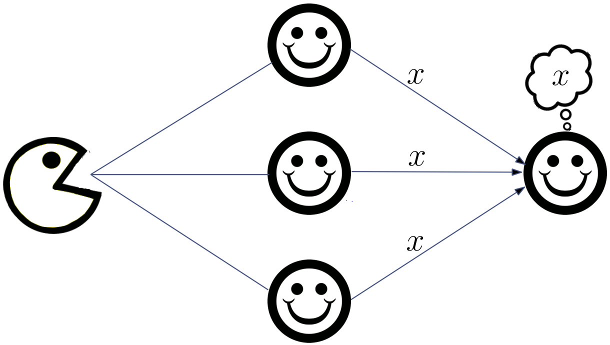
Real Broadcast with Corrupted Dealer



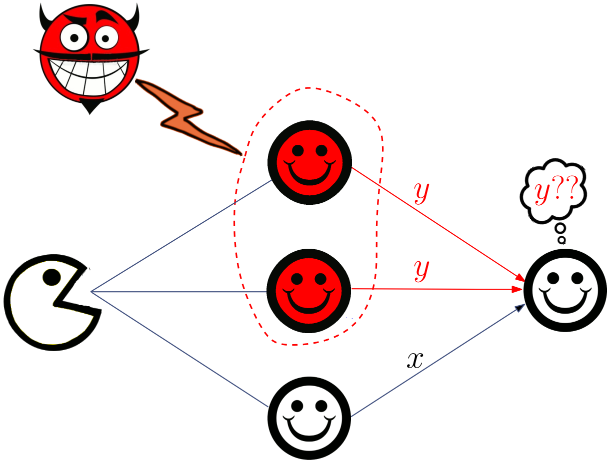
Broadcast in Incomplete Networks



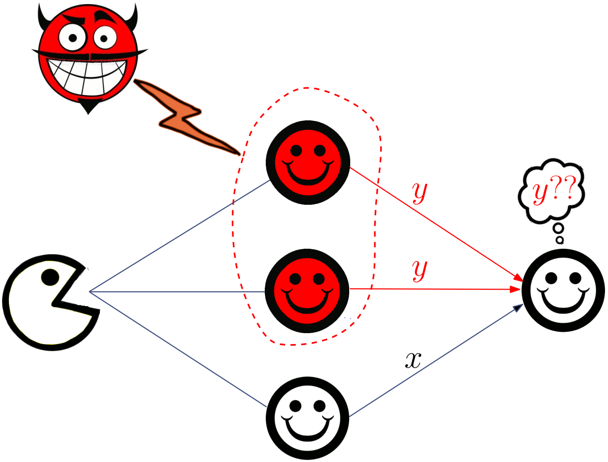
Broadcast in Incomplete Networks



Broadcast in Incomplete Networks



Broadcast in Incomplete Networks



Even Broadcast with an honest dealer is non trivial in this case.

Problem Definition

Player Set: $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$, **Initial input space:** $\mathcal{X}$,
Corrupted players set:$\mathcal{T} \subseteq \mathcal{V}$, **Honest Players Set:** $\mathcal{H} = \mathcal{V} \setminus \mathcal{T}$
Each $v \in \mathcal{V}$ finally outputs (decides on) a value **decision(v)**.

Problem Definition

Player Set: $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$, **Initial input space:** $\mathcal{X}$,
Corrupted players set:$\mathcal{T} \subseteq \mathcal{V}$, **Honest Players Set:** $\mathcal{H} = \mathcal{V} \setminus \mathcal{T}$
Each $v \in \mathcal{V}$ finally outputs (decides on) a value **decision(v)**.

Broadcast (Byzantine Generals)

Dealer $D \in \mathcal{V}$ with **input value** $\mathbf{x_D} \in \mathcal{X}$. Π is a Broadcast protocol for $\mathcal{V}$ if it satisfies:

1. **(Consistency)**
   All honest players decide on the same value $decision(v)$.
2. **(Validity)**
   If D is honest then all honest players decide on the dealer's value x_D.

Problem Definition

Player Set: $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$, **Initial input space:** $\mathcal{X}$,
Corrupted players set:$\mathcal{T} \subseteq \mathcal{V}$, **Honest Players Set:** $\mathcal{H} = \mathcal{V} \setminus \mathcal{T}$
Each $v \in \mathcal{V}$ finally outputs (decides on) a value **decision(v)**.

Broadcast (Byzantine Generals)

Dealer $D \in \mathcal{V}$ with **input value** $\mathbf{x_D} \in \mathcal{X}$. Π is a Broadcast protocol for $\mathcal{V}$ if it satisfies:

1. **(Consistency)**
   All honest players decide on the same value *decision*(v).
2. **(Validity)**
   If D is honest then all honest players decide on the dealer's value x_D.

Consensus (Byzantine Agreement)

Every player $v \in \mathcal{V}$ has an **input value** $\mathbf{x_v} \in \mathcal{X}$. Π is a Consensus protocol for $\mathcal{V}$ if it satisfies:

1. **(Consistency)**
   All honest players decide on the same value *decision*(v).
2. **(Validity)**
   If all honest players have the same input value x then all honest players decide x.

Adversary Restriction

Extreme Corruption cases, e.g., Consensus with one honest player..

Adversary Restriction

Extreme Corruption cases, e.g., Consensus with one honest player..

Possible corruption sets should be restricted, e.g. by cardinality.

Adversary Restriction

Extreme Corruption cases, e.g., Consensus with one honest player..

Possible corruption sets should be restricted, e.g. by cardinality.

- t-**Threshold Adversary:** Can corrupt any player set $\mathcal{T} \subseteq \mathcal{V}, |\mathcal{T}| \leq t$.

Adversary Restriction

Extreme Corruption cases, e.g., Consensus with one honest player..

Possible corruption sets should be restricted, e.g. by cardinality.

- t-**Threshold Adversary:** Can corrupt any player set $\mathcal{T} \subseteq \mathcal{V}, |\mathcal{T}| \leq t$.
- t-**Resilient protocol:** Achieves goal for any corruption set $\mathcal{T}, |\mathcal{T}| \leq t$

Adversary Restriction

Extreme Corruption cases, e.g., Consensus with one honest player..
Possible corruption sets should be restricted, e.g. by cardinality.

- t-**Threshold Adversary:** Can corrupt any player set $\mathcal{T} \subseteq \mathcal{V}, |\mathcal{T}| \leq t$.
- t-**Resilient protocol:** Achieves goal for any corruption set $\mathcal{T}, |\mathcal{T}| \leq t$

Theorem.

No t-resilient consensus protocol exists, for $n \geq 2$ and $t \geq n/2$.

Adversary Restriction

Extreme Corruption cases, e.g., Consensus with one honest player..

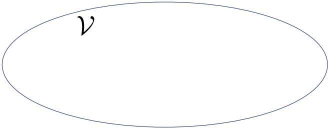
Possible corruption sets should be restricted, e.g. by cardinality.

- t-**Threshold Adversary:** Can corrupt any player set $\mathcal{T} \subseteq \mathcal{V}, |\mathcal{T}| \leq t$.
- t-**Resilient protocol:** Achieves goal for any corruption set $\mathcal{T}, |\mathcal{T}| \leq t$

Theorem.

No t-resilient consensus protocol exists, for $n \geq 2$ and $t \geq n/2$.

Proof.



Adversary Restriction

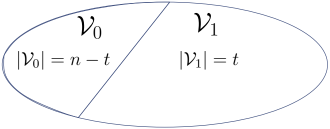
Extreme Corruption cases, e.g., Consensus with one honest player..
Possible corruption sets should be restricted, e.g. by cardinality.

- t-**Threshold Adversary:** Can corrupt any player set $\mathcal{T} \subseteq \mathcal{V}, |\mathcal{T}| \leq t$.
- t-**Resilient protocol:** Achieves goal for any corruption set $\mathcal{T}, |\mathcal{T}| \leq t$

Theorem.

No t-resilient consensus protocol exists, for $n \geq 2$ and $t \geq n/2$.

Proof.



$x_v = i, \text{ if } v \in \mathcal{V}_i$

Adversary Restriction

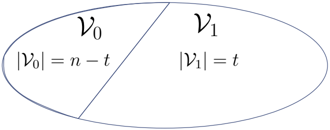
Extreme Corruption cases, e.g., Consensus with one honest player..
Possible corruption sets should be restricted, e.g. by cardinality.

- t-**Threshold Adversary:** Can corrupt any player set $\mathcal{T} \subseteq \mathcal{V}, |\mathcal{T}| \leq t$.
- t-**Resilient protocol:** Achieves goal for any corruption set $\mathcal{T}, |\mathcal{T}| \leq t$

Theorem.

No t-resilient consensus protocol exists, for $n \geq 2$ and $t \geq n/2$.

Proof.



$x_v = i, \text{ if } v \in \mathcal{V}_i$

$\mathcal{V}_0, \mathcal{V}_1, \emptyset$ **are corruptible**

Adversary Restriction

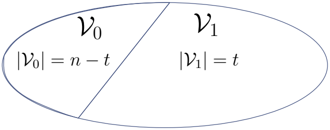
Extreme Corruption cases, e.g., Consensus with one honest player..
Possible corruption sets should be restricted, e.g. by cardinality.

- t-**Threshold Adversary:** Can corrupt any player set $\mathcal{T} \subseteq \mathcal{V}, |\mathcal{T}| \leq t$.
- t-**Resilient protocol:** Achieves goal for any corruption set $\mathcal{T}, |\mathcal{T}| \leq t$

Theorem.

No t-resilient consensus protocol exists, for $n \geq 2$ and $t \geq n/2$.

Proof.



$x_v = i, \text{ if } v \in \mathcal{V}_i$

$\mathcal{V}_0, \mathcal{V}_1, \emptyset$ **are corruptible**

Output

1. If all honest players output x and $\mathcal{T} = \mathcal{V}_x$ then validity is violated.
2. If honest players compute different outputs and $\mathcal{T} = \emptyset$ then consistency is violated. □

Broadcast and Consensus equivalence

Theorem.

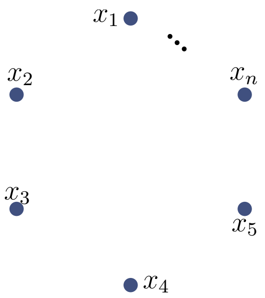
If $t < n/2$ then (efficient) Broadcast is achievable iff (efficient) Consensus is achievable.

Broadcast and Consensus equivalence

Theorem.

If $t < n/2$ then (efficient) Broadcast is achievable iff (efficient) Consensus is achievable.

Proof.



" $\Rightarrow$ "

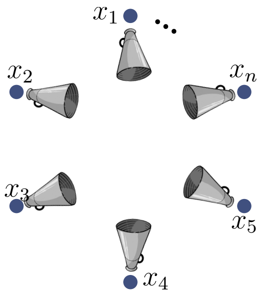
1. Each player v_i holds input value x_i.

Broadcast and Consensus equivalence

Theorem.

If $t < n/2$ then (efficient) Broadcast is achievable iff (efficient) Consensus is achievable.

Proof.



" $\Rightarrow$ "

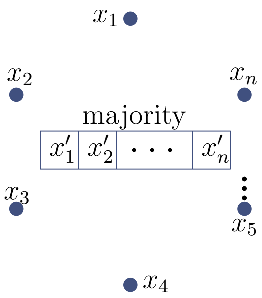
1. Each player v_i holds input value x_i.
2. All players broadcast their input value.

Broadcast and Consensus equivalence

Theorem.

If $t < n/2$ then (efficient) Broadcast is achievable iff (efficient) Consensus is achievable.

Proof.



" $\Rightarrow$ "

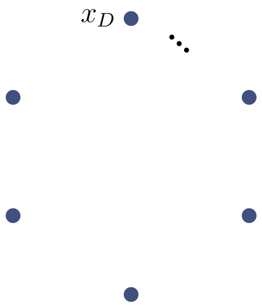
1. Each player v_i holds input value x_i.
2. All players broadcast their input value.
3. Each player obtains the same vector of agreed values and decides on majority.

Broadcast and Consensus equivalence

Theorem.

If $t < n/2$ then (efficient) Broadcast is achievable iff (efficient) Consensus is achievable.

Proof.



" $\Rightarrow$ "

1. Each player v_i holds input value x_i.
2. All players broadcast their input value.
3. Each player obtains the same vector of agreed values and decides on majority.

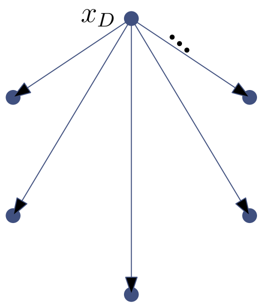
" $\Leftarrow$ "

Broadcast and Consensus equivalence

Theorem.

If $t < n/2$ then (efficient) Broadcast is achievable iff (efficient) Consensus is achievable.

Proof.



" $\Rightarrow$ "

1. Each player v_i holds input value x_i.
2. All players broadcast their input value.
3. Each player obtains the same vector of agreed values and decides on majority.

" $\Leftarrow$ "

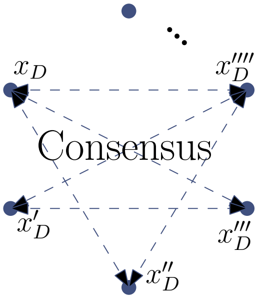
1. Dealer sends input value x_D to all players.

Broadcast and Consensus equivalence

Theorem.

If $t < n/2$ then (efficient) Broadcast is achievable iff (efficient) Consensus is achievable.

Proof.



" $\Rightarrow$ "

1. Each player v_i holds input value x_i.
2. All players broadcast their input value.
3. Each player obtains the same vector of agreed values and decides on majority.

" $\Leftarrow$ "

1. Dealer sends input value x_D to all players.
2. Players run Consensus on the values received by the dealer. □

Adversary Model

Corruption Type

- **Passive:** Obtains all internal data of corrupted players.

Adversary Model

Corruption Type

- **Passive:** Obtains all internal data of corrupted players.
- **Active (Byzantine):** Full control of corrupted players.

Adversary Model

Corruption Type

- **Passive:** Obtains all internal data of corrupted players.
- **Active (Byzantine):** Full control of corrupted players.
- **Fail-Stop (Fault):** Makes corrupted players crash at any time.

Adversary Model

Corruption Type

- **Passive:** Obtains all internal data of corrupted players.
- **Active (Byzantine):** Full control of corrupted players.
- **Fail-Stop (Fault):** Makes corrupted players crash at any time.
- **Static/Adaptive/Mobile**

Adversary Model

Corruption Type

- **Passive:** Obtains all internal data of corrupted players.
- **Active (Byzantine):** Full control of corrupted players.
- **Fail-Stop (Fault):** Makes corrupted players crash at any time.
- **Static/Adaptive/Mobile**

Adversary's Computing Power

- **Unlimited**

Adversary Model

Corruption Type

- **Passive:** Obtains all internal data of corrupted players.
- **Active (Byzantine):** Full control of corrupted players.
- **Fail-Stop (Fault):** Makes corrupted players crash at any time.
- **Static/Adaptive/Mobile**

Adversary's Computing Power

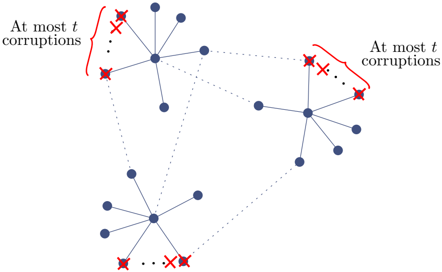
- **Unlimited**
- **Computationally Bounded**
  (to probabilistic polynomial time computations in a security parameter κ).

Admissible Corruption Sets

- *t*-Threshold Adverary [LSP82]: Can corrupt any player subset of cardinality at most t.

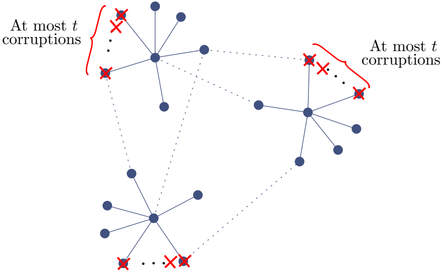
Admissible Corruption Sets

- *t*-Threshold Adverary [LSP82]: Can corrupt any player subset of cardinality at most t.
- *t*-Locally Bounded [Koo04]: Can corrupt at most t players in each neighborhood.



Admissible Corruption Sets

- *t*-Threshold Adverary [LSP82]: Can corrupt any player subset of cardinality at most t.
- *t*-Locally Bounded [Koo04]: Can corrupt at most t players in each neighborhood.



- General Adversary Model [HM97]: Monotone family (structure) $\mathcal{Z} \in 2^V$ of admissible corruption player-sets. Subsumes all other models.

Communication Model

Communication Channels

- **Authenticated**: Resistant to tampering but not to overhearing.

Communication Model

Communication Channels

- **Authenticated**: Resistant to tampering but not to overhearing.
- **Secret/Confidential:** Resistant to overhearing , but not to tampering.

Communication Model

Communication Channels

- **Authenticated**: Resistant to tampering but not to overhearing.
- **Secret/Confidential:** Resistant to overhearing , but not to tampering.
- **Secure:** Authenticated and secret channel.

Communication Model

Communication Channels

- **Authenticated**: Resistant to tampering but not to overhearing.
- **Secret/Confidential:** Resistant to overhearing , but not to tampering.
- **Secure:** Authenticated and secret channel.
- **Synchronous/Asynchronous**
  (No deterministic protocol can achieve asynchronous fault-tolerant Broadcast [FLP85]).

Communication Model

Communication Channels

- **Authenticated**: Resistant to tampering but not to overhearing.
- **Secret/Confidential:** Resistant to overhearing , but not to tampering.
- **Secure:** Authenticated and secret channel.
- **Synchronous/Asynchronous**
  (No deterministic protocol can achieve asynchronous fault-tolerant Broadcast [FLP85]).
- **Complete/Incomplete Communication Networks**

Communication Model

Communication Channels

- **Authenticated**: Resistant to tampering but not to overhearing.
- **Secret/Confidential:** Resistant to overhearing , but not to tampering.
- **Secure:** Authenticated and secret channel.
- **Synchronous/Asynchronous**
  (No deterministic protocol can achieve asynchronous fault-tolerant Broadcast [FLP85]).
- **Complete/Incomplete Communication Networks**

Asynchronous Model: Honest players cannot wait for messages from more than $n - t$ players in each round, where n is the number of players and t the number of corruptions tolerated.

Security

Security is defined with respect to a security parameter κ, allowing an error probability ϵ that is negligible in function of κ.

- **Computational/Cryptographic:** Security against a computationally bounded adversary.

Security

Security is defined with respect to a security parameter κ, allowing an error probability ϵ that is negligible in function of κ.

- **Computational/Cryptographic:** Security against a computationally bounded adversary.
- **Unconditional/Information-Theoretic:** Security against an unlimited adversary.

Security

Security is defined with respect to a security parameter κ, allowing an error probability ϵ that is negligible in function of κ.

- **Computational/Cryptographic:** Security against a computationally bounded adversary.
- **Unconditional/Information-Theoretic:** Security against an unlimited adversary.
- **Perfect Security:** Unconditional Security with zero error probability.

Security

Security is defined with respect to a security parameter κ, allowing an error probability ϵ that is negligible in function of κ.

- **Computational/Cryptographic:** Security against a computationally bounded adversary.
- **Unconditional/Information-Theoretic:** Security against an unlimited adversary.
- **Perfect Security:** Unconditional Security with zero error probability.

Consistently shared data: Typically a PKI.

Efficiency of Distributed Protocols

Synchronous round: One clock cycle of the global clock.
Asynchronous round: Time period equal to the maximum message delivery in the run.

Efficiency of Distributed Protocols

Synchronous round: One clock cycle of the global clock.
Asynchronous round: Time period equal to the maximum message delivery in the run.

Efficiency

- Round Complexity: Maximum number of rounds required by any honest player to halt in the worst case.

Efficiency of Distributed Protocols

Synchronous round: One clock cycle of the global clock.
Asynchronous round: Time period equal to the maximum message delivery in the run.

Efficiency

- **Round Complexity:** Maximum number of rounds required by any honest player to halt in the worst case.
- **Bit Complexity:** Total number of bits sent by all honest players in the worst case.

Efficiency of Distributed Protocols

Synchronous round: One clock cycle of the global clock.
Asynchronous round: Time period equal to the maximum message delivery in the run.

Efficiency

- **Round Complexity:** Maximum number of rounds required by any honest player to halt in the worst case.
- **Bit Complexity:** Total number of bits sent by all honest players in the worst case.
- **Local Computations Complexity:** Maximum over the local computational worst-case complexities of all honest players.

Efficiency of Distributed Protocols

Synchronous round: One clock cycle of the global clock.
Asynchronous round: Time period equal to the maximum message delivery in the run.

Efficiency

- **Round Complexity:** Maximum number of rounds required by any honest player to halt in the worst case.
- **Bit Complexity:** Total number of bits sent by all honest players in the worst case.
- **Local Computations Complexity:** Maximum over the local computational worst-case complexities of all honest players.

Efficiency of Distributed Protocols

Synchronous round: One clock cycle of the global clock.
Asynchronous round: Time period equal to the maximum message delivery in the run.

Efficiency

- **Round Complexity:** Maximum number of rounds required by any honest player to halt in the worst case.
- **Bit Complexity:** Total number of bits sent by all honest players in the worst case.
- **Local Computations Complexity:** Maximum over the local computational worst-case complexities of all honest players.

Fully Polynomial Protocol

Protocol of polynomial Bit, Round and Local Computations Complexity.

Broadcast Protocols

Broadcast Protocols– History

Improvement of trade-off between Resilience, BC, RC and LCC. local computation complexity.

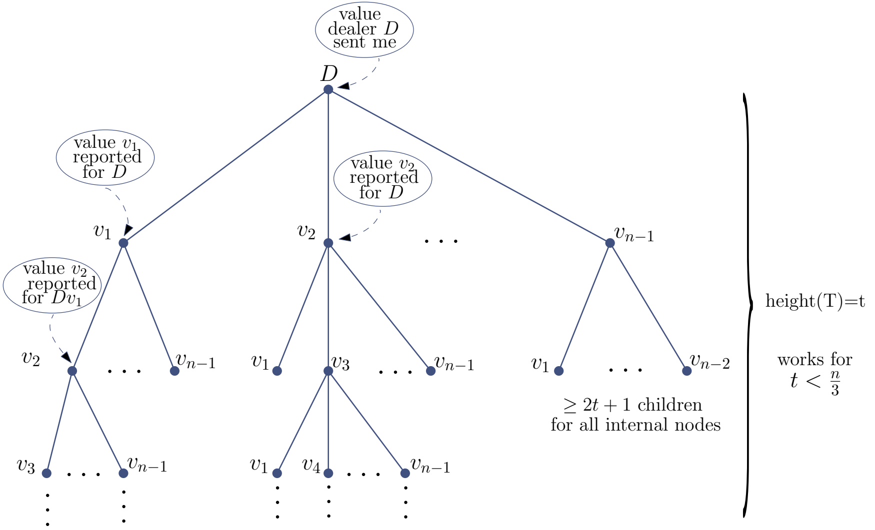
Broadcast Protocols– History

Improvement of trade-off between Resilience, BC, RC and LCC. local computation complexity.

Protocol	n	RC	BC	LCC
[PSL80]	$3t+1$	$t+1$	$exp(n)$	$exp(n)$
[DFF+82]	$3t+1$	$2t+c$	$poly(n)$	$poly(n)$
[Coa86]	$4t+1$	$t+\frac{t}{d}$	$O(n^d)$	$\exp(n)$
[BNDDS92]	$3t+1$	$t+\frac{t}{d}$	$O(n^d)$	$O(n^d)$
[MW88]	$6t+1$	$t+1$	$poly(n)$	$poly(n)$
[BG93]	$4t+1$	$t+1$	$poly(n)$	$poly(n)$
[BG91]	$(3+\epsilon)t$	$t+1$	$poly(n)\cdot O(2^{1/\epsilon})$	$poly(n)\cdot O(2^{1/\epsilon})$
[GM98]	$3t+1$	$t+1$	$poly(n)$	$poly(n)$

Exponential Information Gathering

The EIG Tree



EIG Algorithm I - Information Gathering

Information Gathering

Round 1

1. Dealer sends its initial value x_D to the $n-1$ other players and decides on x_D.
2. Each v stores value x_D in the root of $tree_v$ ($tree_v(D) := x_D$). A special default value of $\perp$ is stored if the Dealer failed to send a legitimate value in X.

EIG Algorithm I - Information Gathering

Information Gathering

Round 1

1. Dealer sends its initial value x_D to the $n-1$ other players and decides on x_D.
2. Each v stores value x_D in the root of $tree_v$ ($tree_v(D) := x_D$). A special default value of $\perp$ is stored if the Dealer failed to send a legitimate value in X.

Round h, $2 \leq h \leq t+1$

1. Each v broadcasts the leaves of its round $(h-1)$ tree.
2. Every v adds a new level to its tree, storing at node $D \ldots qr$ the value that r claims to have stored in node $D \ldots q$ in its own $tree_r$. Again, $\perp$ is used for inappropriate messages.

EIG Algorithm I - Information Gathering

Information Gathering

Round 1

1. Dealer sends its initial value x_D to the $n-1$ other players and decides on x_D.
2. Each v stores value x_D in the root of $tree_v$ ($tree_v(D) := x_D$). A special default value of $\perp$ is stored if the Dealer failed to send a legitimate value in X.

Round h, $2 \leq h \leq t+1$

1. Each v broadcasts the leaves of its round $(h-1)$ tree.
2. Every v adds a new level to its tree, storing at node $D \ldots qr$ the value that r claims to have stored in node $D \ldots q$ in its own $tree_r$. Again, $\perp$ is used for inappropriate messages.

Intuitively, v stores in node $D \ldots qr$ the value that "r says q says $\ldots$ the source said".

EIG Algorithm II - Data Conversion

After $t+1$ rounds o Information Gathering, each player v computes the commonly agreed-upon recursive function *resolve*() in order to decide.

Resolve Function

(Recursive majority of descendants of node a)
For all a sequences of $tree_v$:

$$resolve_v(a) = \begin{cases} tree(a) & \text{, if } a \text{ is a leaf;} \\ m & \text{, If } m \text{ is the majority of } resolve \text{ applied} \\ & \quad \text{to the children of } a; \\ \perp & \text{, If } a \text{ is not a leaf and no majority exists.} \end{cases}$$

EIG Algorithm II - Data Conversion

After $t + 1$ rounds o Information Gathering, each player v computes the commonly agreed-upon recursive function *resolve*() in order to decide.

Resolve Function

(Recursive majority of descendants of node a)

For all a sequences of $tree_v$:

$$resolve_v(a) = \begin{cases} tree(a) & \text{, if } a \text{ is a leaf;} \\ m & \text{, If } m \text{ is the majority of } resolve \text{ applied} \\ & \quad \text{to the children of } a; \\ \perp & \text{, If } a \text{ is not a leaf and no majority exists.} \end{cases}$$

Decision

Player v decides on the value $resolve_v(D)$.

Complexity of the EIG Algorithm

Theorem (Lamport, Shostak, Pease 1982).

The EIG Algorithm achieves Broadcast in $t + 1$ rounds provided that $n \geq 3t + 1$.

Complexity of the EIG Algorithm

Theorem (Lamport, Shostak, Pease 1982).

The EIG Algorithm achieves Broadcast in $t+1$ rounds provided that $n \geq 3t+1$.

Bit Complexity

For any $1 \leq h \leq t+1$, the h-round EIG tree has $O(n^{h-1})$ leaves, yielding messages of size $O(n^{h-1})$ in round $h+1$. Thus, BC and LCC grow exponential in t.

Complexity of the EIG Algorithm

Theorem (Lamport, Shostak, Pease 1982).

The EIG Algorithm achieves Broadcast in $t+1$ rounds provided that $n \geq 3t+1$.

Bit Complexity

For any $1 \leq h \leq t+1$, the h-round EIG tree has $O(n^{h-1})$ leaves, yielding messages of size $O(n^{h-1})$ in round $h+1$. Thus, BC and LCC grow exponential in t.

[GM98]: First $(t+1)$-round fully polynomial, optimal resilience Broadcast protocol.

Reducing the Communication Cost

- 1989: P.Berman, J.Garay, K. Perry, first communication efficient 1/3−resilient protocol. Basis of many later protocols.
- *King Consensus* Protocol. Using the equivalence of Broadcast-Consensus easily transformed in a Broadcast protocol.
- Input value space $X = \{0, 1, \perp\}$(Binary Consensus). Can be used to achieve General Consensus with an overhead of 2 extra rounds and $O(n^2 \cdot b)$ extra bits, where b : maximum length of a message [Coa87].

Weak Consistency

Weak Consistency

If an honest player v_i decides on $y_i \in \{0, 1\}$ then every other honest v_j decides on $y_i \in \{y_i, \perp\}$.

Weak Consistency

Weak Consistency

If an honest player v_i decides on $y_i \in \{0, 1\}$ then every other honest v_j decides on $y_j \in \{y_i, \perp\}$.

Protocol: $WeakConsensus(x_1 \dots x_n) \rightarrow (y_1 \dots y_n)$

1. Every $v_i \in \mathcal{V}$ sends x_i to all v_j.
   Let c_m^j be the copies of a message $m \in \{0, 1\}$ received by player v_j in this round.
2. Every v_j computes:
$$y_j = \begin{cases} m & \text{if } c_m^j \geq n - t \\ \perp & else \end{cases}$$
3. Every $v_j \in \mathcal{V}$ returns y_j

Weak Consensus Correctness

Lemma.

WeakConsensus achieves Weak Consistency and Validity for $t < n/3$.

Weak Consensus Correctness

Lemma.

WeakConsensus achieves Weak Consistency and Validity for $t < n/3$.

Proof.

Validity: Let $x_i = x$, $\forall v_i \in \mathcal{V}$.
Step 2: All $v_i \in \mathcal{H}$ collect the value x at least $n - t$ times, thus all $v_i \in \mathcal{H}$ receive the value $1 - x$ at most $t < n - t$ (since $t < n/3$) and they all decide on $y_i = x$.

Weak Consensus Correctness

Lemma.

WeakConsensus achieves Weak Consistency and Validity for $t < n/3$.

Proof.

Validity: Let $x_i = x, \ \forall v_i \in \mathcal{V}$.
Step 2: All $v_i \in \mathcal{H}$ collect the value x at least $n-t$ times, thus all $v_i \in \mathcal{H}$ receive the value $1-x$ at most $t < n-t$ (since $t < n/3$) and they all decide on $y_i = x$.
Weak Consistency: Let $v_i, v_j \in \mathcal{H}$ and $y_i = 0$. Thus $c_0^i \geq n-t$. That means that at least $n-2t$ honest players sent him this value.
Consequently

$$c_0^j \geq n-2t \Rightarrow c_1^j = n-n+2t = 2t < n-t$$

So v_j computes either $y_j = 0$ or $y_j = \perp$. □

Graded Consistency

Every $v_i \in \mathcal{V}$ computes y_i and the *grade value* $g_i \in \{0, 1\}$.

Graded Consistency

If $v_i \in \mathcal{H}$ decides on $y_i \in \{0, 1\}$ with $g_i = 1$ then every other $v_j \in \mathcal{H}$ decides on $y_j = y_i$.

Graded Consistency

Every $v_i \in \mathcal{V}$ computes y_i and the *grade value* $g_i \in \{0, 1\}$.

Graded Consistency

If $v_i \in \mathcal{H}$ decides on $y_i \in \{0, 1\}$ with $g_i = 1$ then every other $v_j \in \mathcal{H}$ decides on $y_j = y_i$.

Protocol: $\mathit{GradedConsensus}(\mathbf{x_1}, \ldots, \mathbf{x_n}) \rightarrow ((\mathbf{y_1}, \mathbf{g_1}), \ldots, (\mathbf{y_n}, \mathbf{g_n}))$

1. $(z_1, \ldots, z_n) := \mathit{WeakConsensus}(x_1, \ldots, x_n)$
2. Every $v_i \in \mathcal{V}$ sends z_i to all v_j.
3. Every v_j computes:
$$y_j = \begin{cases} 1 & \text{if } c_1^j > c_0^j \\ 0 & \mathit{else} \end{cases}, \qquad g_j = \begin{cases} 1 & \text{if } c_{y_j}^j \geq n - t \\ 0 & \mathit{else} \end{cases}$$
4. Every $v_j \in \mathcal{V}$ returns (y_j, g_j)

Graded Consensus Correctness

Lemma.

The above protocol achieves Graded Consistency and Validity remains.

Graded Consensus Correctness

Lemma.

The above protocol achieves Graded Consistency and Validity remains.

Proof.

Validity: If $\forall\, v_i, v_j \in \mathcal{H},\ x_i = x$ then $(y_i, g_i) = (x, 1)$.
Let x be the common input value. After step 1, $z_i = x, \forall v_i \in \mathcal{H}$, due to WeakConsensus. Validity remains in a similar way as in WeakConsensus.

Graded Consensus Correctness

Lemma.

The above protocol achieves Graded Consistency and Validity remains.

Proof.

Validity: If $\forall\ v_i, v_j \in \mathcal{H},\ x_i = x$ then $(y_i, g_i) = (x, 1)$.
Let x be the common input value. After step 1, $z_i = x, \forall v_i \in \mathcal{H}$, due to WeakConsensus. Validity remains in a similar way as in WeakConsensus.

Graded Consistency: Let $v_i, v_j \in \mathcal{H}$ and let v_i output $(y_i, 1)$.
That means that at least $n - 2t$ honest players sent him $z_k = y_i$.

Player v_j also receives y_i from $n - 2t$ honest players. The remaining $t + 1$ honest send him either y_i or $\perp$ due to WeakConsensus. Thus,

$$c^j_{1-y_i} \leq t < n - 2t \Rightarrow y_j = y_i$$

□

King Consistency

A player v_k is chosen to be the king.

King Consistency

If the king v_k is honest, then all honest players compute the same output $x \in \{0, 1\}$.

King Consistency

A player v_k is chosen to be the king.

King Consistency

If the king v_k is honest, then all honest players compute the same output $x \in \{0, 1\}$.

Protocol: $KingConsensus(v_k, x_1, \ldots x_n) \rightarrow (y_1, \ldots, y_n)$

1. $((z_1, g_1) \ldots, (z_n, g_n)) := GradedConsensus(x_1, \ldots, x_n)$
2. The king v_k sends z_k to all players.
3. Every v_j computes

$$y_j = \begin{cases} z_j & \text{if } g_j = 1 \\ z_k & \textit{else} \end{cases}$$

4. Every v_j returns y_j

King Consensus Correctness

Lemma.

The above protocol achieves King Consistency and Validity remains.

King Consensus Correctness

Lemma.

The above protocol achieves King Consistency and Validity remains.

Proof.

Validity: If $\forall\ v_i, v_j \in \mathcal{H},\ x_i = x$ then due to Graded Consistency of step 1 these players compute $(z_i, g_i) = (x, 1)$. Therefore Every $v_i \in \mathcal{H}$ outputs $y_i = x$.

King Consensus Correctness

Lemma.

The above protocol achieves King Consistency and Validity remains.

Proof.

Validity: If $\forall\ v_i, v_j \in \mathcal{H},\ x_i = x$ then due to Graded Consistency of step 1 these players compute $(z_i, g_i) = (x, 1)$. Therefore Every $v_i \in \mathcal{H}$ outputs $y_i = x$.

King Consistency: Let the king $v_k \in \mathcal{H}$

If $\forall\ v_i \in \mathcal{H},\ g_i = 0$ in step 1 then all honest v_i output $y_i = z_k$ in step 3.

If $\exists\ v_i \in \mathcal{H}$ with $g_i = 1$, because of Graded Consistency all honest (king included) computed the same z_i, thus they output $y_i = z_i$

□

Consensus Protocol

If we ensure that the king is honest then Consensus will be achieved.

Consensus Protocol

If we ensure that the king is honest then Consensus will be achieved. $\rightarrow$ run the KingConsensus protocol $t + 1$ times, each time with a different king:

Consensus Protocol

If we ensure that the king is honest then Consensus will be achieved. $\rightarrow$ run the KingConsensus protocol $t+1$ times, each time with a different king:

Consensus$(x_1, \ldots, x_n) \rightarrow (y_1, \ldots, y_n)$

1. For $k := 1$ *to* $t+1$
   $(x_1, \ldots, x_n) := KingConsensus(v_k, x_1, \ldots x_n)$
2. Every v_j sets $y_j := x_j$
3. Every v_j returns y_j

Consensus Protocol

If we ensure that the king is honest then Consensus will be achieved. $\rightarrow$ run the KingConsensus protocol $t+1$ times, each time with a different king:

$Consensus(x_1, \ldots, x_n) \rightarrow (y_1, \ldots, y_n)$

1. For $k := 1$ *to* $t+1$
   $(x_1, \ldots, x_n) := KingConsensus(v_k, x_1, \ldots x_n)$
2. Every v_j sets $y_j := x_j$
3. Every v_j returns y_j

Observation

If the king is honest, by King Consistency all honest players decide on the same output value v which will be their input value for the next round. Due to the fact that the KingConsensus sub-protocol maintains Validity the final decision value of each honest player will remain v.

Broadcast Protocol

Protocol: $Broadcast(x, D) \rightarrow (y_1 \ldots, y_n)$

1. Dealer D sends x to all players
2. $(y_1, \ldots, y_n) := Consensus(x_1, \ldots, x_n)$,
   with x_i the value that player v_i received from the Dealer.
3. Every $v_j \in \mathcal{V}$ returns y_j

Broadcast Protocol

Protocol: $Broadcast(x, D) \rightarrow (y_1 \ldots, y_n)$

1. Dealer D sends x to all players
2. $(y_1, \ldots, y_n) := Consensus(x_1, \ldots, x_n)$, with x_i the value that player v_i received from the Dealer.
3. Every $v_j \in \mathcal{V}$ returns y_j

Theorem ([BG89]).

The above protocol achieves Broadcast (Consensus) with resiliency $n > 3t$, $BC = O(n^2 t)$ *and* $RC = 3t + O(1)$.

Broadcast Protocol

Protocol: *Broadcast*$(x, D) \rightarrow (y_1 \ldots, y_n)$

1. Dealer D sends x to all players
2. $(y_1, \ldots, y_n) := Consensus(x_1, \ldots, x_n)$,
   with x_i the value that player v_i received from the Dealer.
3. Every $v_j \in \mathcal{V}$ returns y_j

Theorem ([BG89]).

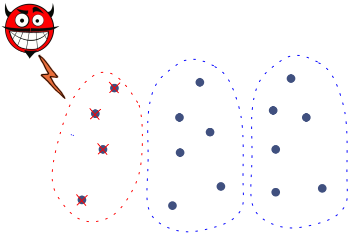
The above protocol achieves Broadcast (Consensus) with resiliency $n > 3t$, $BC = O(n^2 t)$ and $RC = 3t + O(1)$.

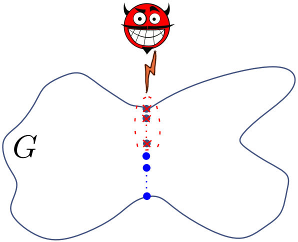
Proof. Each sub-protocol is executed $t + 1$ times and involves *one-to-all* bit communication for every player $BC = O(n^2 t)$
King Consensus: 3 rounds,one for each sub-protocol
$RC = 3t + O(1)$ □

Parameter Lower Bounds

Parameter Lower Bounds -Overview

- Resiliency: $n > 3t$ (Interactive Consistency) [PSL80]
- Bit Complexity: $BC \geq n(t+1)/4$ [DR85]
- Round Complexity: $RC \geq t+1$ [FL82, DS83]
- Connectivity of Network G: $conn(G) > 2t$ [Dol82]





Scenaria–Executions

- **State Assignment** C_i**:** An assignment of states to each player.
- **Message assignment** M_i**:** An assignment of a message to each channel.

A Scenario is defined to be an infinite sequence:

$$\sigma = C_0, M_1, C_1, M_2, C_2, \dots$$

Indistiguishable Scenaria ($\sigma \overset{v}{\sim} \sigma'$)

Two scenaria σ, σ' are indistinguishable with respect to player v, $\sigma \overset{v}{\sim} \sigma'$ if v has the same *view*(v), i.e., the same sequence of states, outgoing and incoming messages.

Scenaria–Executions

- **State Assignment** C_i**:** An assignment of states to each player.
- **Message assignment** M_i**:** An assignment of a message to each channel.

A Scenario is defined to be an infinite sequence:

$$\sigma = C_0, M_1, C_1, M_2, C_2, \ldots$$

Indistiguishable Scenaria ($\sigma \overset{v}{\sim} \sigma'$)

Two scenaria σ, σ' are indistinguishable with respect to player v, $\sigma \overset{v}{\sim} \sigma'$ if v has the same $view(v)$, i.e., the same sequence of states, outgoing and incoming messages. Scenaria σ, σ' may be scenaria of different systems.

Scenaria–Executions

- **State Assignment** C_i**:** An assignment of states to each player.
- **Message assignment** M_i**:** An assignment of a message to each channel.

A Scenario is defined to be an infinite sequence:

$$\sigma = C_0, M_1, C_1, M_2, C_2, \ldots$$

Indistiguishable Scenaria ($\sigma \overset{v}{\sim} \sigma'$)

Two scenaria σ, σ' are indistinguishable with respect to player v, $\sigma \overset{v}{\sim} \sigma'$ if v has the same $\mathit{view}(v)$, i.e., the same sequence of states, outgoing and incoming messages. Scenaria σ, σ' may be scenaria of different systems.

decision(v): deterministic function of $\mathit{view}(v)$ (Perfect Security).

Connectivity Lower Bound $(conn(G) > 2t)$

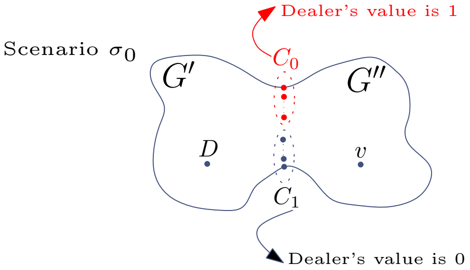
σ_0	σ_1
$x_D = 0$ $T = C_0$	$x_D = 1$ $T = C_1$

Corrupted players C_i of scenario σ_i act like in σ_{1-i}.

Connectivity Lower Bound $(conn(G) > 2t)$

σ_0	σ_1
$x_D = 0$ $T = C_0$	$x_D = 1$ $T = C_1$

Corrupted players C_i of scenario σ_i act like in σ_{1-i}.



Then,

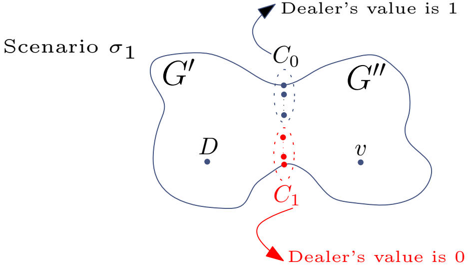
$$\forall v \in G'', \ \sigma_0 \overset{v}{\sim} \sigma_1 \Rightarrow decision_{\sigma_0}(v) = desicion_{\sigma_1}(v)$$

and thus validity is violated. □

Connectivity Lower Bound $(conn(G) > 2t)$

σ_0	σ_1
$x_D = 0$ $T = C_0$	$x_D = 1$ $T = C_1$

Corrupted players C_i of scenario σ_i act like in σ_{1-i}.



Then,

$$\forall v \in G'', \; \sigma_0 \overset{v}{\sim} \sigma_1 \Rightarrow decision_{\sigma_0}(v) = desicion_{\sigma_1}(v)$$

and thus validity is violated. □

Resiliency Lower Bound-Example

Assume that v_0, v_1, v_2 solve Broadcast in two rounds given that $t = 1$:

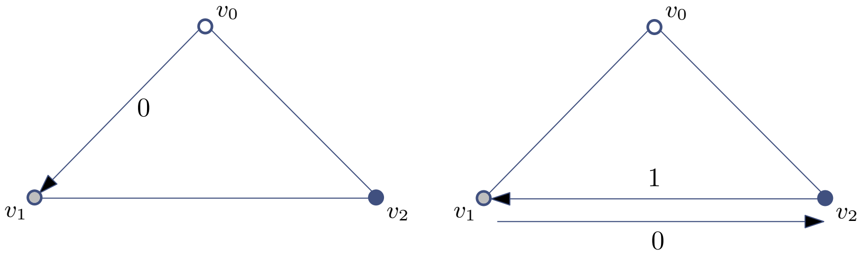
1. The dealer v_0 sends value
2. Each player reports the dealer's value

Resiliency Lower Bound-Example

Assume that v_0, v_1, v_2 solve Broadcast in two rounds given that $t = 1$:

1. The dealer v_0 sends value
2. Each player reports the dealer's value

Honest player v_1, knowing that at most one of the v_0, v_2 is corrupted, has to decide on a value that satisfies both conditions of the Broadcast problem. Consider the following *view*(v_1).



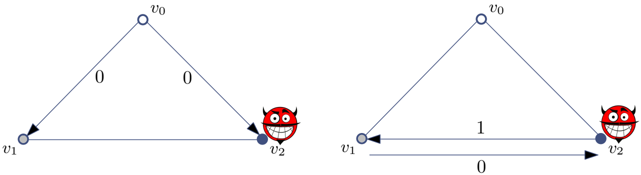
Resiliency Lower Bound-Example

Two possible scenarios σ_1(corrupted v_2) and σ_2(corrupted v_0) s.t. $\sigma_1 \overset{v_1}{\sim} \sigma_2$ (indistinguishable with respect to v_1):

Resiliency Lower Bound-Example

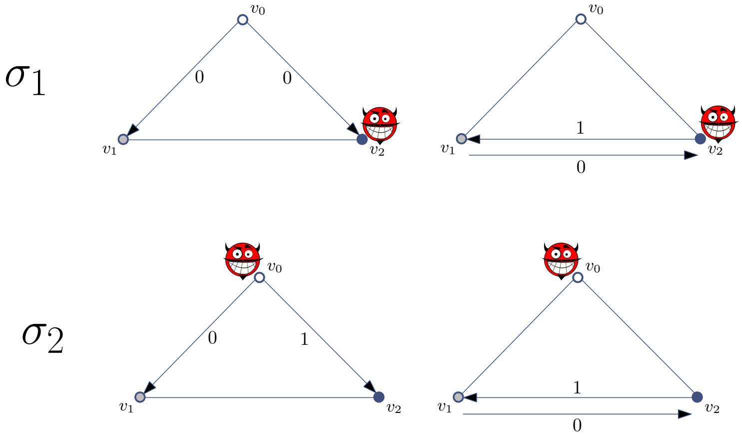
Two possible scenarios σ_1(corrupted v_2) and σ_2(corrupted v_0) s.t. $\sigma_1 \overset{v_1}{\sim} \sigma_2$ (indistinguishable with respect to v_1):

σ_1



Resiliency Lower Bound-Example

Two possible scenarios σ_1(corrupted v_2) and σ_2(corrupted v_0) s.t. $\sigma_1 \overset{v_1}{\sim} \sigma_2$ (indistinguishable with respect to v_1):



Resiliency Lower Bound-Example

Impossibility of Broadcast

If $decision(v_1) = 1$ and σ_1 holds, then validity is violated, thus

$$decision(v_1) = 0 \quad (1)$$

Resiliency Lower Bound-Example

Impossibility of Broadcast

If $decision(v_1) = 1$ and σ_1 holds, then validity is violated, thus

$$decision(v_1) = 0 \tag{1}$$

If σ_2 holds then by symmetry v_2 should decide on 1

$$decision(v_1) = 1 \tag{2}$$

$(1), (2) \Rightarrow$ Consistency is violated.

Resiliency Lower Bound-Example

Impossibility of Broadcast

If $decision(v_1) = 1$ and σ_1 holds, then validity is violated, thus

$$decision(v_1) = 0 \quad (1)$$

If σ_2 holds then by symmetry v_2 should decide on 1

$$decision(v_1) = 1 \quad (2)$$

$(1), (2) \Rightarrow$ Consistency is violated.

The algorithm uses only two rounds and particular types of messages.

Resiliency Lower Bound I

Lemma 3.1.

Three players cannot solve the Broadcast problem in the presence of one fault ($n = 3$ and $t = 1$).

Resiliency Lower Bound I

Lemma 3.1.

Three players cannot solve the Broadcast problem in the presence of one fault ($n = 3$ and $t = 1$).

Proof. Assume the existence of algorithm $\mathcal{A}$ that achieves Broadcast in system T in the presence of a corrupted player. Construct system H using two copies of T,

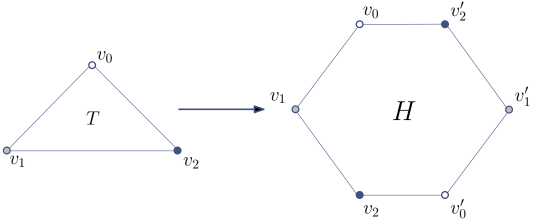


Figure : Identical copy $v'_k = v_{k+3}$ of v_k. Connect $v_{k \mod 6}$ with $v_{(k+1) \mod 6}$ and $v_{(k-1) \mod 6}$

Resiliency Lower Bound II

In H all players run $\mathcal{A}$ and have only local names for their neighbors.

Claim

For all σ_H scenario of H without adversary and $\forall k \in \{0, \ldots, 5\}$, $\exists \sigma_T$ scenario of T in which $v_{(k+2) \bmod 3}$ is corrupted s.t.

$$\sigma_H \overset{v_k}{\sim} \sigma_T \text{ and } \sigma_H \overset{v_{k+1 \bmod 6}}{\sim} \sigma_T$$

Resiliency Lower Bound II

In H all players run $\mathcal{A}$ and have only local names for their neighbors.

Claim

For all σ_H scenario of H without adversary and $\forall k \in \{0, \ldots, 5\}$, $\exists \sigma_T$ scenario of T in which $v_{(k+2) \bmod 3}$ is corrupted s.t.

$$\sigma_H \overset{v_k}{\sim} \sigma_T \text{ and } \sigma_H \overset{v_{k+1 \bmod 6}}{\sim} \sigma_T$$

For v_k and $v_{k+1 \bmod 6}$, their views are indistinguishable from their views as players $v_{k \bmod 3}$ and $v_{(k+1) \bmod 3}$ in T where the adversary corrupts $v_{(k+2) \bmod 3}$ by simply simulating all the remaining players of H.

Resiliency Lower Bound II

In H all players run $\mathcal{A}$ and have only local names for their neighbors.

Claim

For all σ_H scenario of H without adversary and $\forall k \in \{0, \ldots, 5\}$, $\exists \sigma_T$ scenario of T in which $v_{(k+2) \mod 3}$ is corrupted s.t.

$$\sigma_H \overset{v_k}{\sim} \sigma_T \text{ and } \sigma_H \overset{v_{k+1 \mod 6}}{\sim} \sigma_T$$

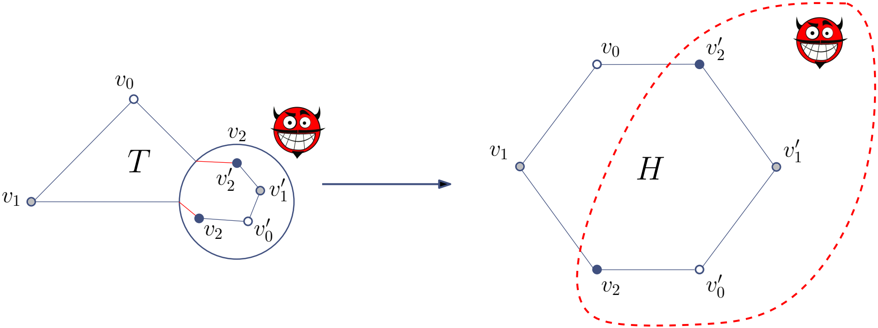
For v_k and $v_{k+1 \mod 6}$, their views are indistinguishable from their views as players $v_{k \mod 3}$ and $v_{(k+1) \mod 3}$ in T where the adversary corrupts $v_{(k+2) \mod 3}$ by simply simulating all the remaining players of H.

Thus, every such pair executes A in H without adversary and achieves Broadcast. If H exhibits contradictory behavior then A cannot exist.

Resiliency Lower Bound III

Example.

The adversary corrupts v_2 in T by simulating the subsystem of H encircled



Resiliency Lower Bound IV

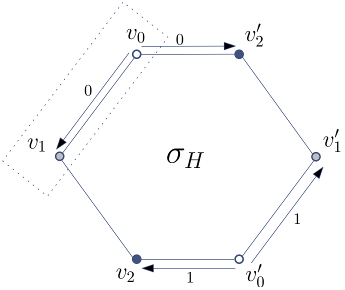
Contradictory behavior of H

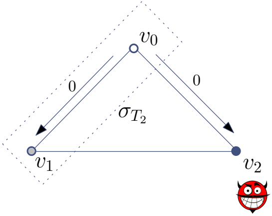
H involves two players v_0, v_0' of the type corresponding to the Dealer. Suppose they have inputs $x_0 \in \{0, 1\}$ and $x_0' = 1 - x_0$ respectively.

Resiliency Lower Bound IV

Contradictory behavior of H

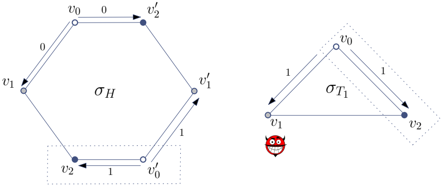
H involves two players v_0, v_0' of the type corresponding to the Dealer. Suppose they have inputs $x_0 \in \{0,1\}$ and $x_0' = 1 - x_0$ respectively.





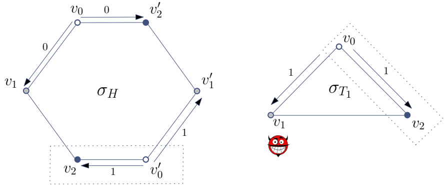
$$\sigma_H \overset{v_0}{\sim} \sigma_{T_2} \text{ and } \sigma_H \overset{v_1}{\sim} \sigma_{T_2} \Rightarrow \mathit{decision}(v_1) = 0 \quad (1)$$

Resiliency Lower Bound V

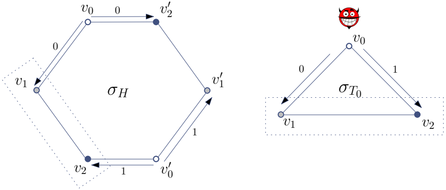


$$\sigma_H \overset{v_0'}{\sim} \sigma_{T_1} \text{ and } \sigma_H \overset{v_2}{\sim} \sigma_{T_1} \Rightarrow decision(v_2) = 1 \quad (2)$$

Resiliency Lower Bound V

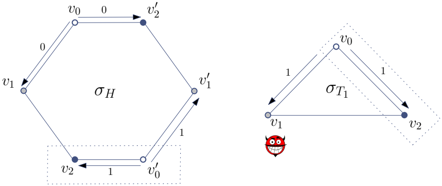


$$\sigma_H \overset{v_0'}{\sim} \sigma_{T_1} \text{ and } \sigma_H \overset{v_2}{\sim} \sigma_{T_1} \Rightarrow decision(v_2) = 1 \qquad (2)$$

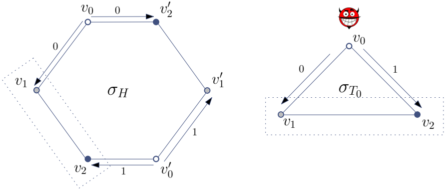


$$\sigma_H \overset{v_1}{\sim} \sigma_{T_0} \text{ and } \sigma_H \overset{v_2}{\sim} \sigma_{T_0} \Rightarrow decision(v_1) = decision(v_2) \qquad (3)$$

Resiliency Lower Bound V



$$\sigma_H \overset{v_0'}{\sim} \sigma_{T_1} \text{ and } \sigma_H \overset{v_2}{\sim} \sigma_{T_1} \Rightarrow \mathit{decision}(v_2) = 1 \qquad (2)$$



$$\sigma_H \overset{v_1}{\sim} \sigma_{T_0} \text{ and } \sigma_H \overset{v_2}{\sim} \sigma_{T_0} \Rightarrow \mathit{decision}(v_1) = \mathit{decision}(v_2) \qquad (3)$$

Relations (1), (2) and (3) yield a contradiction. □

Resiliency Lower Bound VI

Theorem 3.2.

There is no solution to the Broadcast problem for n players in the presence of t corrupted players, if $3 \leq n \leq 3t$

Resiliency Lower Bound VI

Theorem 3.2.

There is no solution to the Broadcast problem for n players in the presence of t corrupted players, if $3 \leq n \leq 3t$

Proof.

Idea: Assume Broadcast protocol A with dealer v_0 for $|\mathcal{V}| = n, |T| \geq n/3$. Transform A into B Broadcast protocol for $|\mathcal{V}| = 3, |T| = 1$.

Let partition $\mathcal{V}_0 \cup \mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ s.t. $\forall i$, $1 \leq |\mathcal{V}_i| \leq t$. We let each v_i simulate every $v \in \mathcal{V}_i$ (messages and computation steps)

Resiliency Lower Bound VI

Theorem 3.2.

There is no solution to the Broadcast problem for n players in the presence of t corrupted players, if $3 \leq n \leq 3t$

Proof.

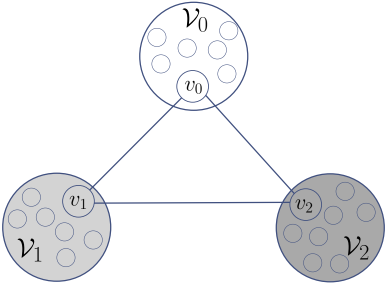
Idea: Assume Broadcast protocol A with dealer v_0 for $|\mathcal{V}| = n, |T| \geq n/3$. Transform A into B Broadcast protocol for $|\mathcal{V}| = 3, |T| = 1$.

Let partition $\mathcal{V}_0 \cup \mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ s.t. $\forall i,\ 1 \leq |\mathcal{V}_i| \leq t$. We let each v_i simulate every $v \in \mathcal{V}_i$ (messages and computation steps)

Protocol B

Player v_0: dealer in protocol B.
If in A: $v \in \mathcal{V}_i$ sends m to $u \in \mathcal{V}_j$, $i \neq j$, then
B: v_i sends m to v_j along with the identities of v, u.
If in A: $v \in \mathcal{V}_i$ decides on m, then
B: v_i decides on the value m. (If there are multiple values chooses one)

Resiliency Lower Bound VII



In A, $T_A = \mathcal{V}_j$, where $T_B = v_j$ ($|T_A| \leq t$).

Termination: From Termination of A and $v_i \in \mathcal{H}$, $\exists v \in \mathcal{V}_i$ and v decides, so does v_i in B.

Validity: From Validity in A.

Consistency: From Consistency in A.

□

Bit Complexity

Theorem 3.3 (Dolev, Reischuk 1985).

Every Broadcast protocol which handles up to t corruptions ($t < n-1$), requires at least $n(t+1)/4$ messages to be sent.

Bit Complexity

Theorem 3.3 (Dolev, Reischuk 1985).

Every Broadcast protocol which handles up to t corruptions ($t < n - 1$), requires at least $n(t + 1)/4$ messages to be sent.

Proof.

Assume scenarios:

- σ_0 with honest dealer D and $x_D = 0$
- σ_1 with honest dealer D and $x_D = 1$

Bit Complexity

Theorem 3.3 (Dolev, Reischuk 1985).

Every Broadcast protocol which handles up to t corruptions ($t < n-1$), requires at least $n(t+1)/4$ messages to be sent.

Proof.

Assume scenarios:

- σ_0 with honest dealer D and $x_D = 0$
- σ_1 with honest dealer D and $x_D = 1$

$A(v) = \{$ Players that communicate with v in at least one scenario $\}$.

Bit Complexity

Theorem 3.3 (Dolev, Reischuk 1985).

Every Broadcast protocol which handles up to t corruptions ($t < n - 1$), requires at least $n(t+1)/4$ messages to be sent.

Proof.

Assume scenarios:

- σ_0 with honest dealer D and $x_D = 0$
- σ_1 with honest dealer D and $x_D = 1$

$A(v) = \{$ Players that communicate with v in at least one scenario $\}$.

Let $\exists v \in \mathcal{V},\ s.t.\ |A(v)| \leq t.$ Consider scenario:

- σ': Scenario σ_1 with $u \in A(v)$ acting towards v as in σ_0.

Bit Complexity

Theorem 3.3 (Dolev, Reischuk 1985).

Every Broadcast protocol which handles up to t corruptions ($t < n-1$), requires at least $n(t+1)/4$ messages to be sent.

Proof.

Assume scenarios:

- σ_0 with honest dealer D and $x_D = 0$
- σ_1 with honest dealer D and $x_D = 1$

$A(v) = \{$ Players that communicate with v in at least one scenario $\}$.

Let $\exists v \in \mathcal{V},\ s.t.\ |A(v)| \leq t$. Consider scenario:

- σ': Scenario σ_1 with $u \in A(v)$ acting towards v as in σ_0.

$$\sigma' \overset{v}{\sim} \sigma_0 \Rightarrow decision_v(\sigma') = 0,\ \ and$$
$$\sigma' \overset{u}{\sim} \sigma_1 \Rightarrow decision_u(\sigma') = 1,\quad \forall u \in \{\mathcal{H} \setminus \{v\}\}$$

Bit Complexity

Theorem 3.3 (Dolev, Reischuk 1985).

Every Broadcast protocol which handles up to t corruptions ($t < n-1$), requires at least $n(t+1)/4$ messages to be sent.

Proof.

Assume scenarios:

- σ_0 with honest dealer D and $x_D = 0$
- σ_1 with honest dealer D and $x_D = 1$

$A(v) = \{$ Players that communicate with v in at least one scenario $\}$.

Let $\exists v \in \mathcal{V},\ s.t.\ |A(v)| \leq t$. Consider scenario:

- σ': Scenario σ_1 with $u \in A(v)$ acting towards v as in σ_0.

$$\sigma' \overset{v}{\sim} \sigma_0 \Rightarrow decision_v(\sigma') = 0,\ \ and$$
$$\sigma' \overset{u}{\sim} \sigma_1 \Rightarrow decision_u(\sigma') = 1,\ \ \forall u \in \{\mathcal{H} \setminus \{v\}\}$$

Hence $|A(v)| \geq t+1 \Rightarrow n(t+1)/2$ overall messages in both scenarios

$\Rightarrow$ At least $n(t+1)/4$ messages in σ_0 or σ_1. □

References I

Piotr Berman and Juan A. Garay.
Asymptotically optimal distributed consensus (extended abstract).
In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca, editors, *ICALP*, volume 372 of *Lecture Notes in Computer Science*, pages 80–94. Springer, 1989.

Piotr Berman and Juan A. Garay.
Efficient distributed consensus with n = (3 + epsilon) t processors (extended abstract).
In Sam Toueg, Paul G. Spirakis, and Lefteris M. Kirousis, editors, *WDAG*, volume 579 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 1991.

Piotr Berman and Juan A. Garay.
Cloture votes: n/4-resilient distributed consensus in t+1 rounds.
Mathematical Systems Theory, 26(1):3–19, 1993.

References II

Amotz Bar-Noy, Danny Dolev, Cynthia Dwork, and H. Raymond Strong.
Shifting gears: Changing algorithms on the fly to expedite byzantine agreement.
Inf. Comput., 97(2):205–233, 1992.

Brian A Coan.
A communication-efficient canonical form for fault-tolerant distributed protocols.
In *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, PODC '86, pages 63–72, New York, NY, USA, 1986. ACM.

B. A. Coan.
Achieving consensus in fault-tolerant distributed computer systems: protocols, lower bounds, and simulations.
PhD thesis, Cambridge, MA, USA, 1987.

References III

Danny Dolev, Michael J. Fischer, Robert J. Fowler, Nancy A. Lynch, and H. Raymond Strong.
An efficient algorithm for byzantine agreement without authentication.
Information and Control, 52(3):257–274, 1982.

Danny Dolev.
The byzantine generals strike again.
J. Algorithms, 3(1):14–30, 1982.

Danny Dolev and Rüdiger Reischuk.
Bounds on information exchange for byzantine agreement.
J. ACM, 32(1):191–204, 1985.

Danny Dolev and H. Raymond Strong.
Authenticated algorithms for byzantine agreement.
SIAM J. Comput., 12(4):656–666, 1983.

References IV

- Michael J. Fischer and Nancy A. Lynch.
  A lower bound for the time to assure interactive consistency.
  Inf. Process. Lett., 14(4):183–186, 1982.

- Michael J. Fischer, Nancy A. Lynch, and Mike Paterson.
  Impossibility of distributed consensus with one faulty process.
  J. ACM, 32(2):374–382, 1985.

- Juan A. Garay and Yoram Moses.
  Fully polynomial byzantine agreement for $n > 3t$ processors in $t + 1$ rounds.
  SIAM J. Comput., 27(1):247–290, 1998.

References V

Martin Hirt and Ueli M. Maurer.
Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract).
In James E. Burns and Hagit Attiya, editors, *PODC*, pages 25–34. ACM, 1997.

Chiu-Yuen Koo.
Broadcast in radio networks tolerating byzantine adversarial behavior.
In Soma Chaudhuri and Shay Kutten, editors, *PODC*, pages 275–282. ACM, 2004.

Leslie Lamport, Robert E. Shostak, and Marshall C. Pease.
The byzantine generals problem.
ACM Trans. Program. Lang. Syst., 4(3):382–401, 1982.

References VI

Y. Moses and O. Waarts.
Coordinated traversal: (t+1)-round byzantine agreement in polynomial time.
In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 246 –255, oct 1988.

Marshall C. Pease, Robert E. Shostak, and Leslie Lamport.
Reaching agreement in the presence of faults.
J. ACM, 27(2):228–234, 1980.