Distributed Computing Robustness

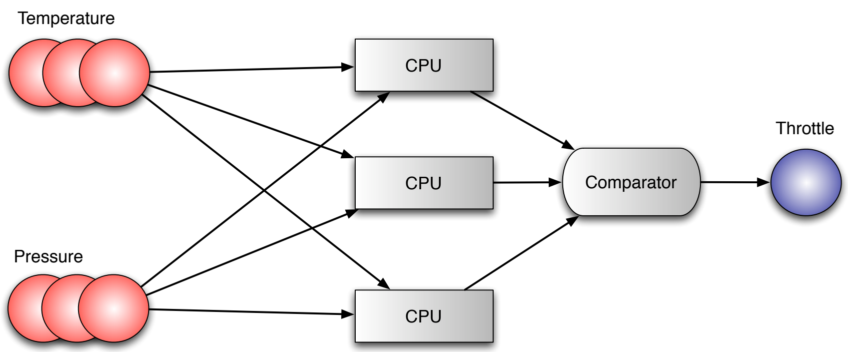
Sébastien Tixeuil
sebastien.tixeuil@lip6.fr

Motivation

Approach

- *Faults* and *attacks* occur in the network
- The network's user must *not* notice something wrong happened
- A *small* number of faulty components
- **Masking** approach to fault/attack tolerance

Principle



Problems

- Replicated input sensors may not give the same data
- Faulty input sensor or processor may not fail gracefully
- The system might not be tolerant to software bugs

Byzantine Generals

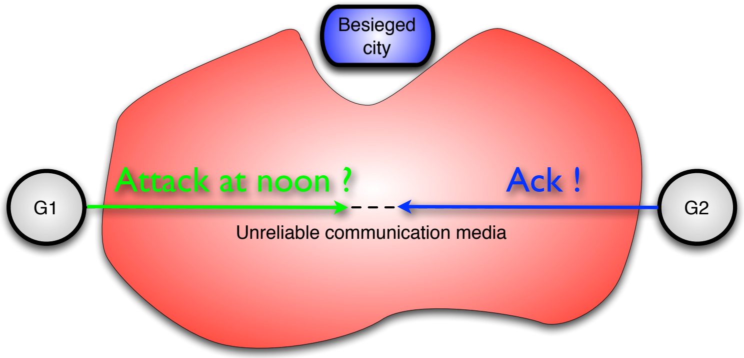
Settings

- Byzantine generals are camping outside an enemy city
- Generals can communicate by sending messengers
- Generals must decide upon common plan of action
- Some of the Generals can be traitors

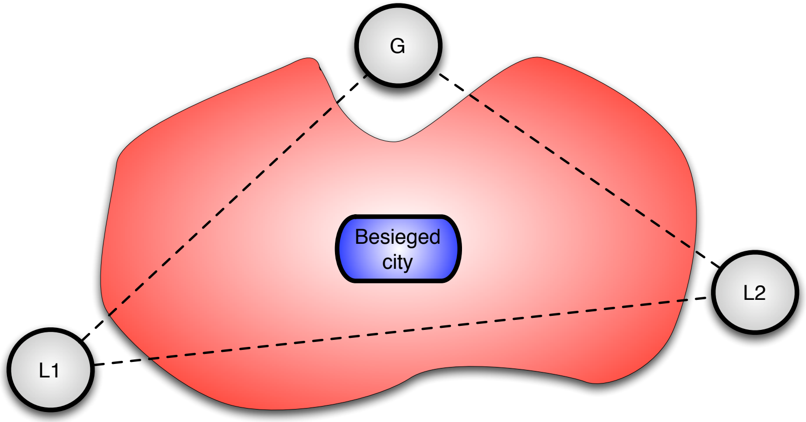
Goal

- All loyal generals decide upon the same plan of action
- A small number of traitors cannot cause the loyal generals to adopt a bad plan

Two Generals Paradox



The Byzantine Generals Problem



The (simple) Byzantine Generals Problem

- Generals lead *n* divisions of the Byzantine army
- The divisions communicate via reliable messengers
- The generals must **agree** on a plan (“attack” or “retreat”) even if some of them are **killed** by enemy spies

Oral Model

- **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

Solution?

plan: **array of** {A,R}; finalPlan: {A,R}

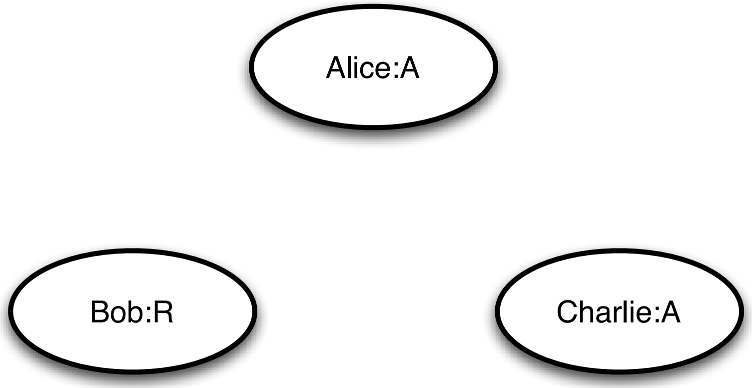
1: plan[myID] := *ChooseAorR*()

2: for all other G *send*(G, myID, plan[myID])

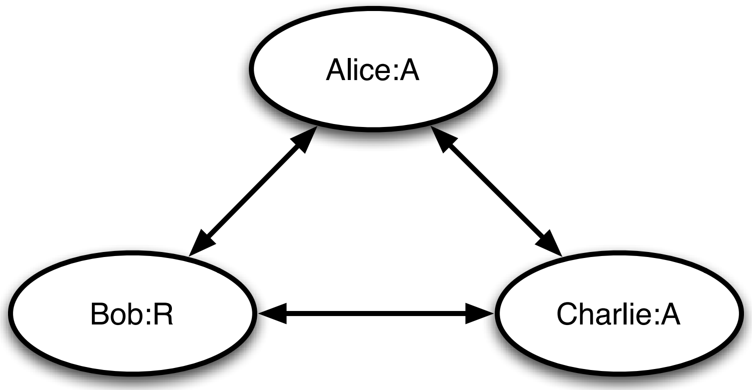
3: for all other G *receive*(G, plan[G])

4: finalPlan := *majority*(plan)

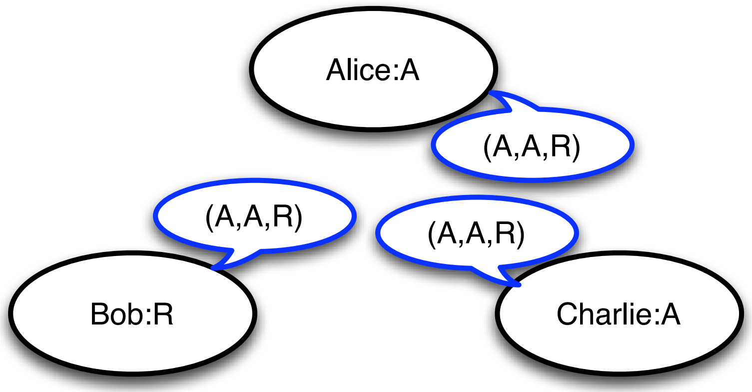
Reliable Networks



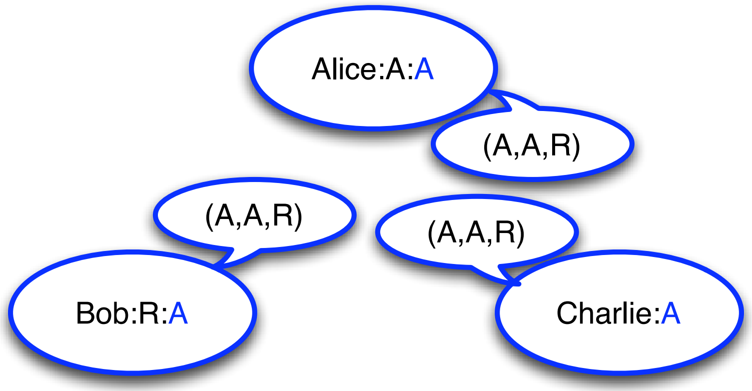
Reliable Networks



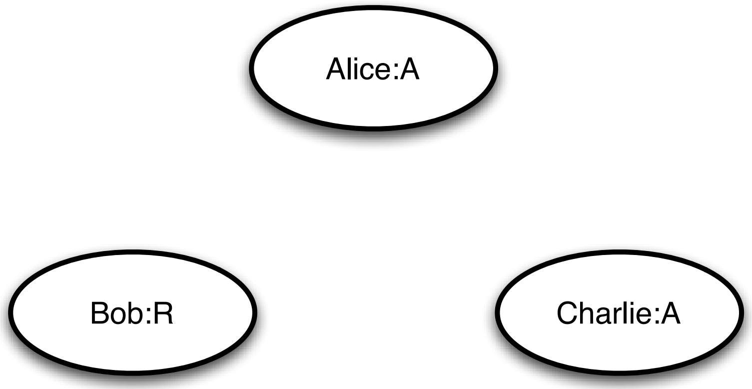
Reliable Networks



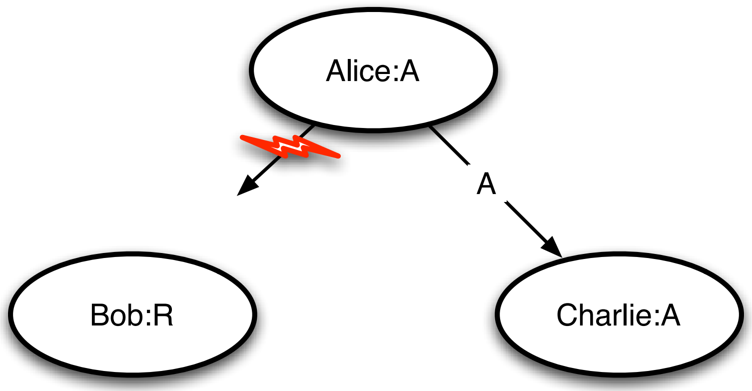
Reliable Networks



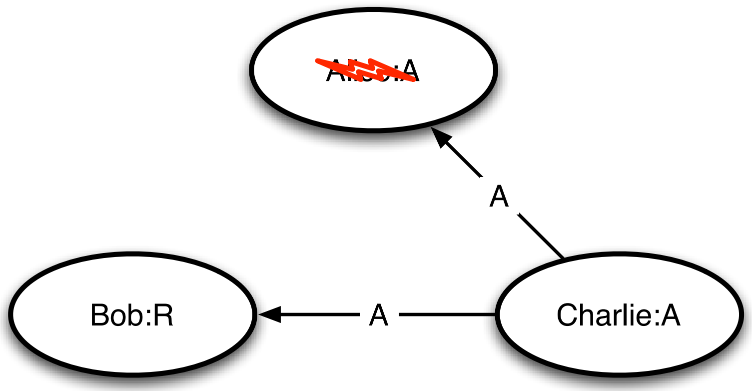
Crashing Networks



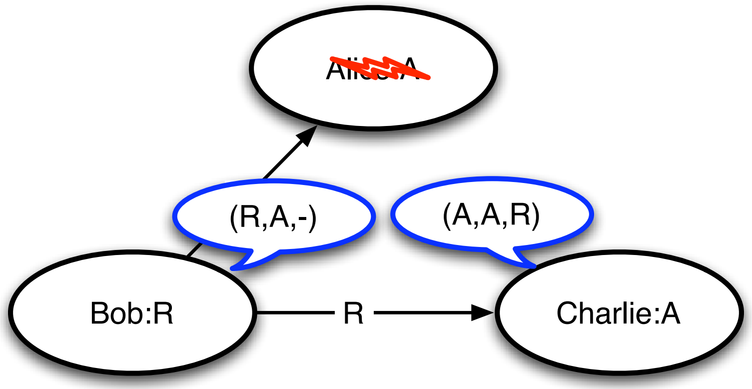
Crashing Networks



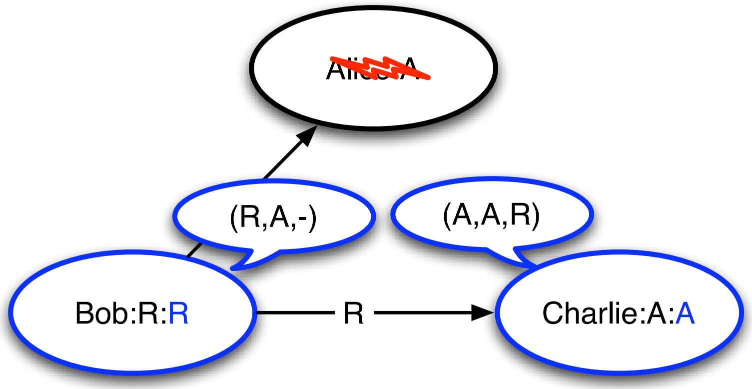
Crashing Networks



Crashing Networks



Crashing Networks



The Byzantine Generals Problem

- A general and *n-1* lieutenants lead n divisions of the Byzantine army
- The divisions communicate via messengers that can be captured or delayed
- The generals must **agree** on a plan ("attack" or "retreat") even if some of them are **traitors** that want to prevent agreement

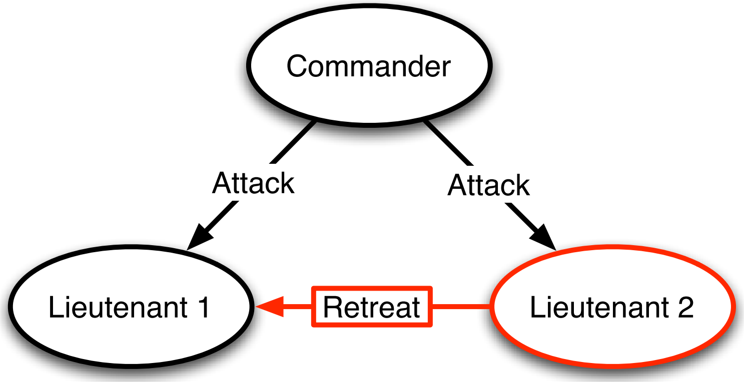
The Byzantine Generals Problem

- A commanding general must sent an order to his n-1 lieutenants 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

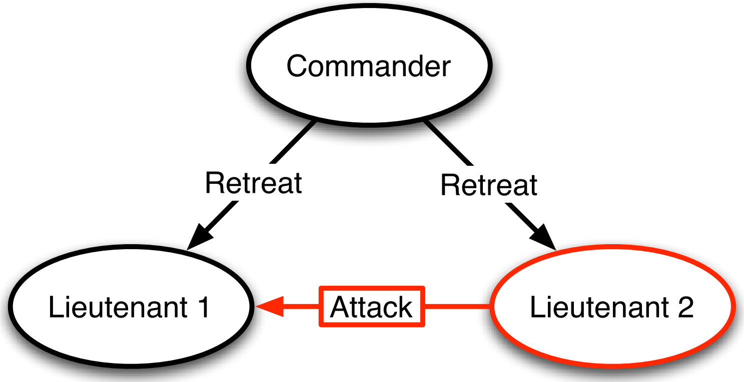
Oral Model

- **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

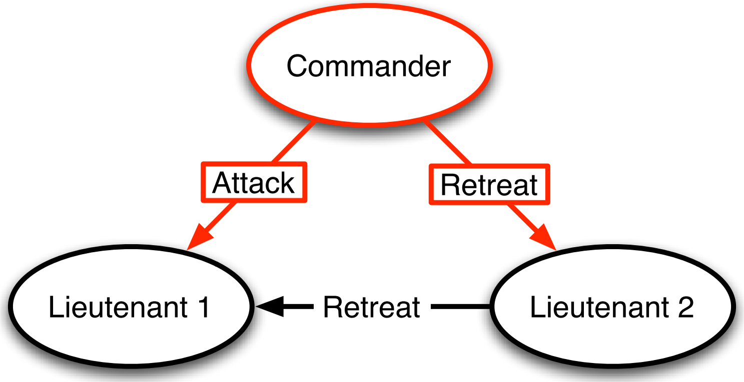
3k+1 nodes are necessary (oral model)



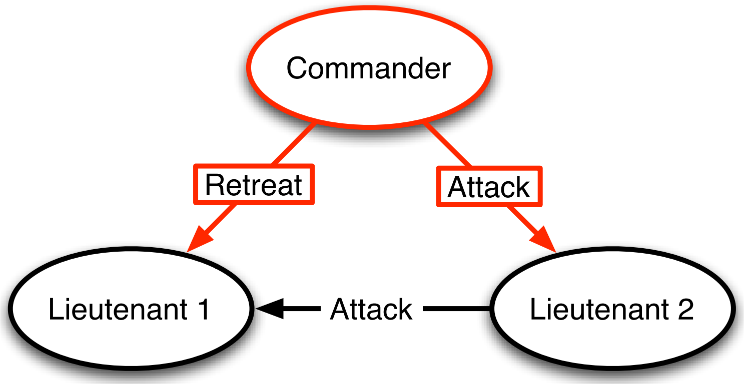
3k+1 nodes are necessary (oral model)



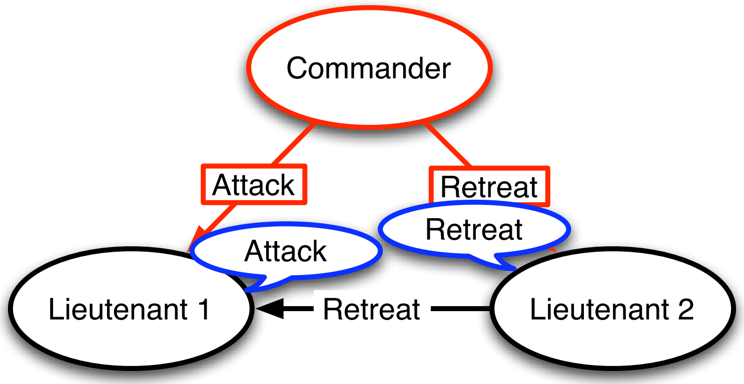
3k+1 nodes are necessary (oral model)



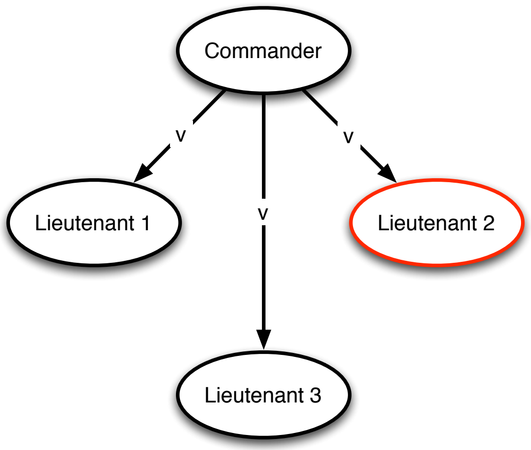
3k+1 nodes are necessary (oral model)



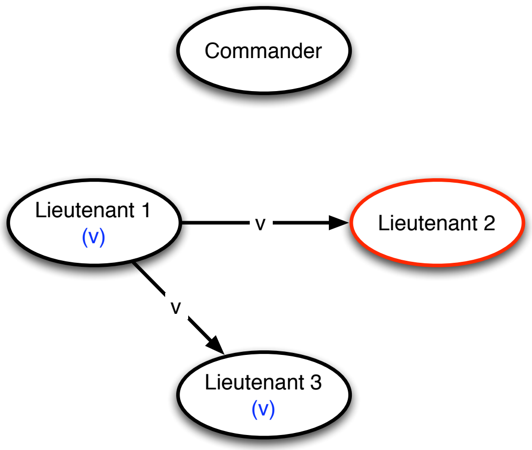
3k+1 nodes are necessary (oral model)



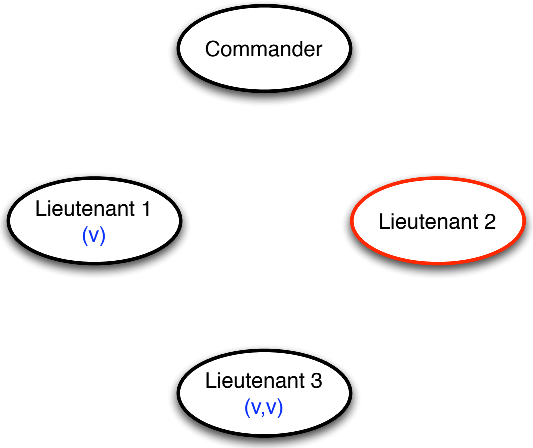
3k+1 nodes are sufficient (oral model)



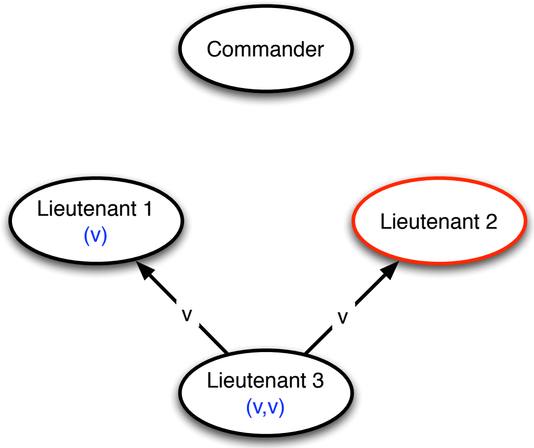
3k+1 nodes are sufficient (oral model)



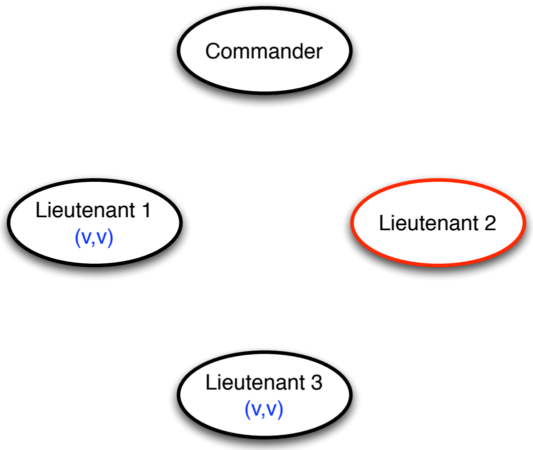
3k+1 nodes are sufficient (oral model)



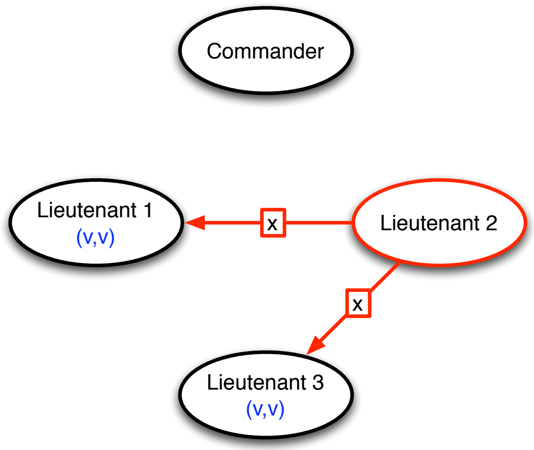
3k+1 nodes are sufficient (oral model)



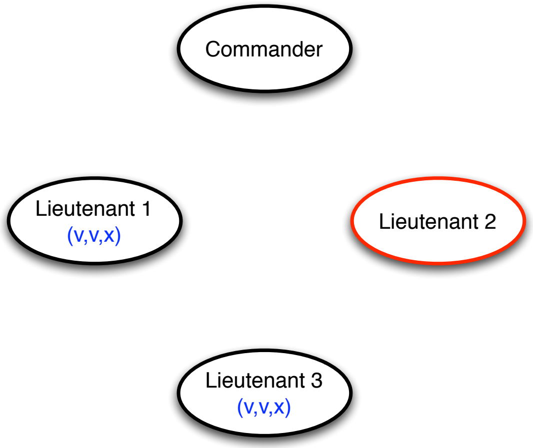
3k+1 nodes are sufficient (oral model)



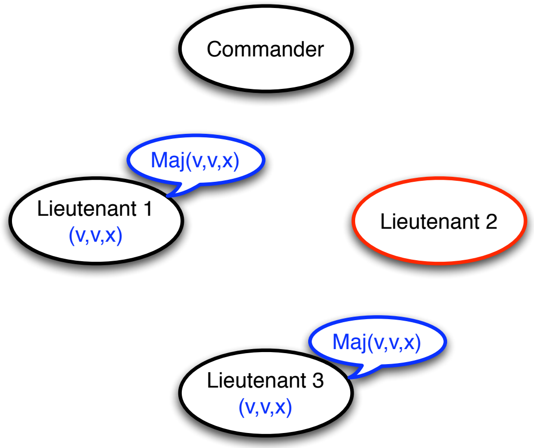
3k+1 nodes are sufficient (oral model)



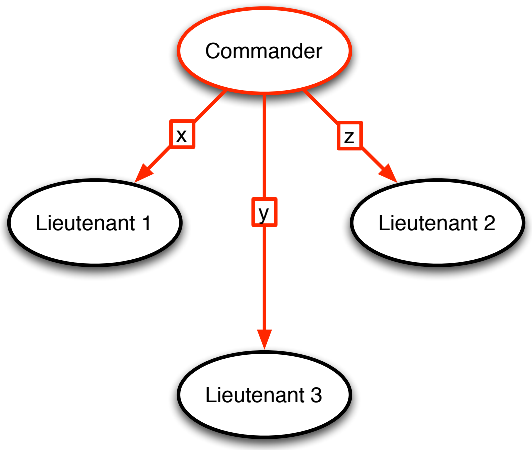
3k+1 nodes are sufficient (oral model)



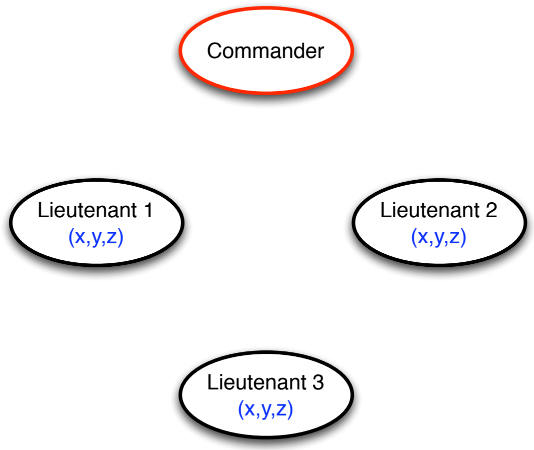
3k+1 nodes are sufficient (oral model)



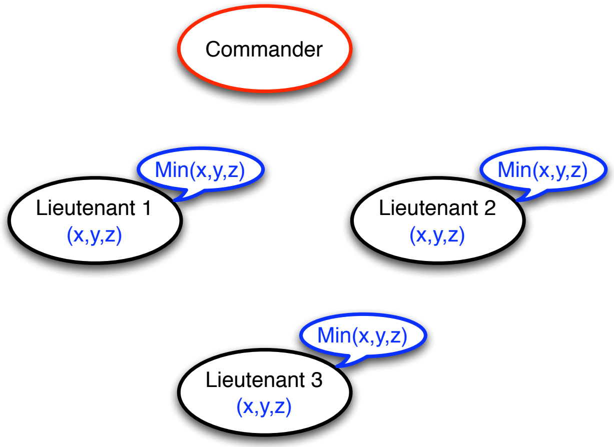
3k+1 nodes are sufficient (oral model)



3k+1 nodes are sufficient (oral model)



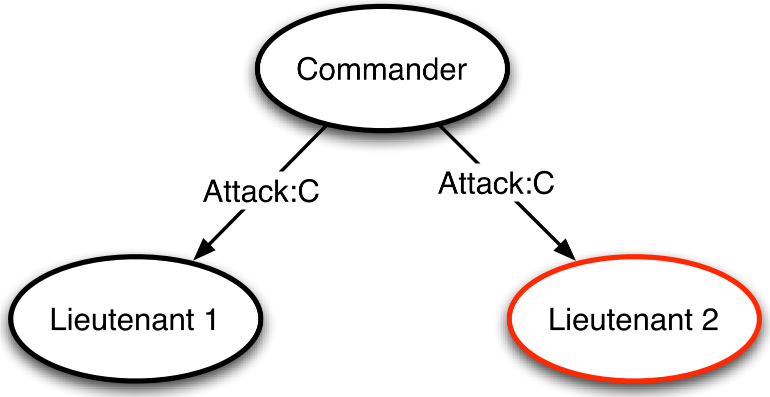
3k+1 nodes are sufficient (oral model)



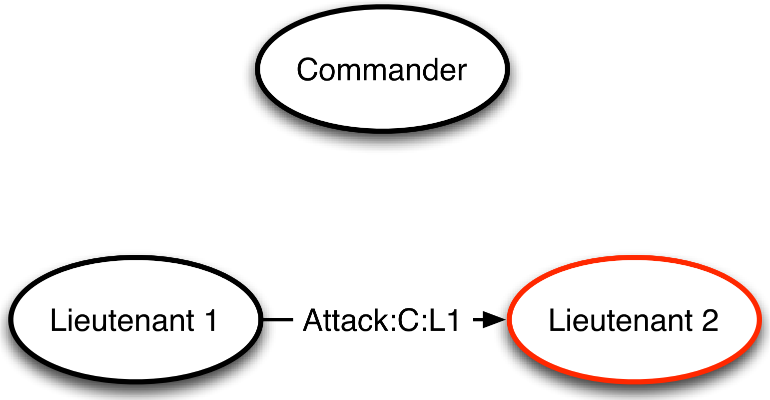
Written Model

- **A1-A3**: Same as before
- **A4**:
  - A loyal general's signature cannot be forged, and any alteration of the contents of his signed messages can be detected
  - Anyone can verify the authenticity of a general's signature

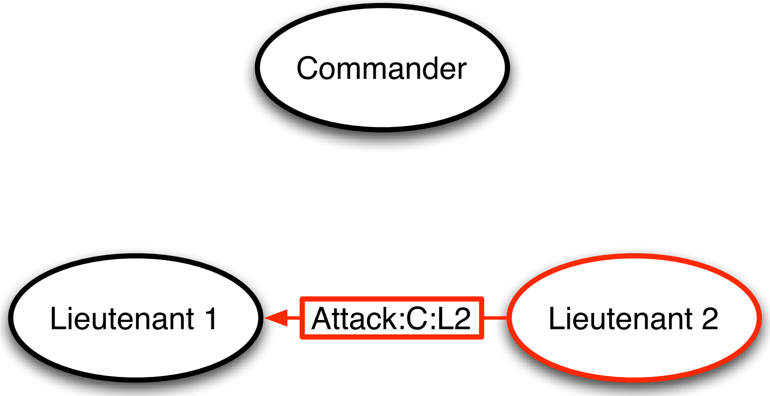
k+2 nodes are sufficient (written model)



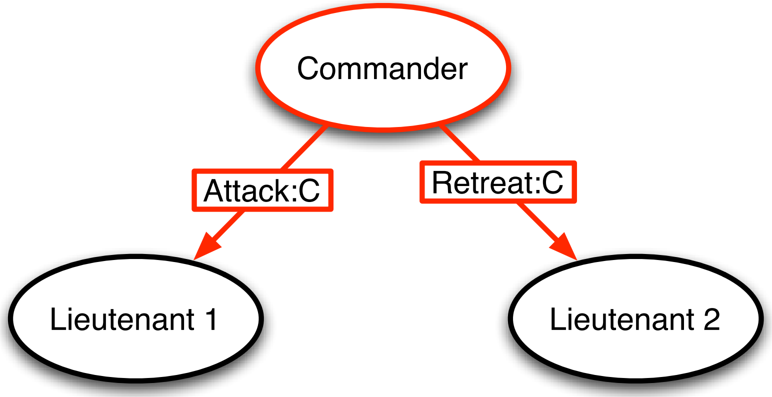
k+2 nodes are sufficient (written model)



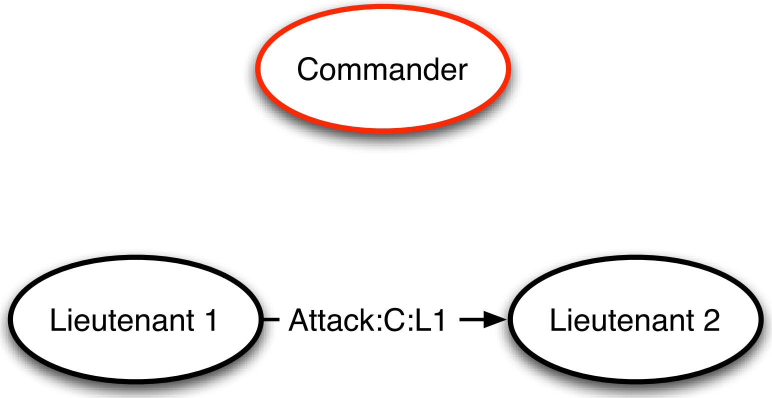
k+2 nodes are sufficient (written model)



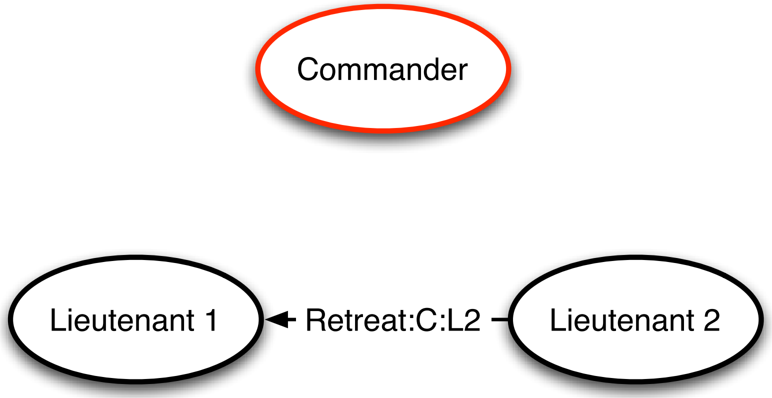
k+2 nodes are sufficient (written model)



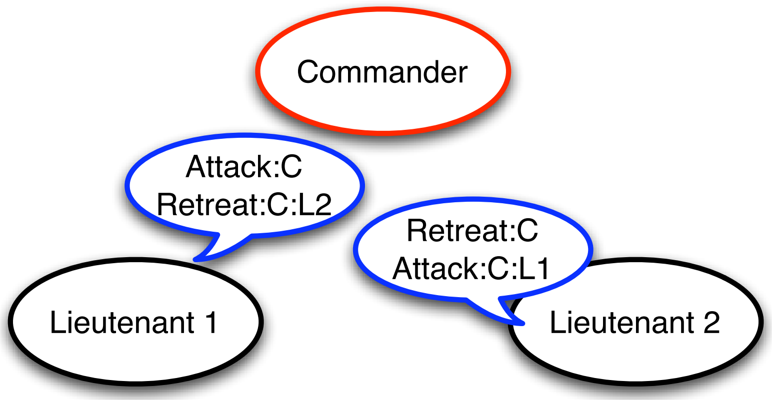
k+2 nodes are sufficient (written model)



k+2 nodes are sufficient (written model)

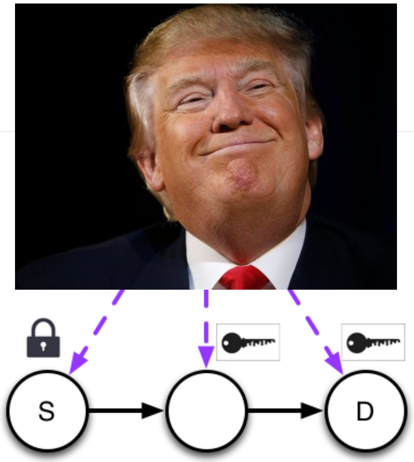


k+2 nodes are sufficient (written model)

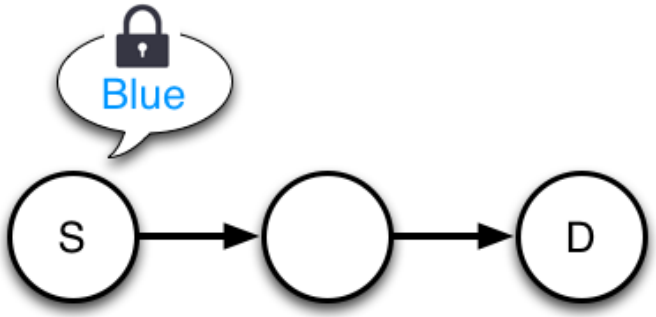


Why not Cryptography?

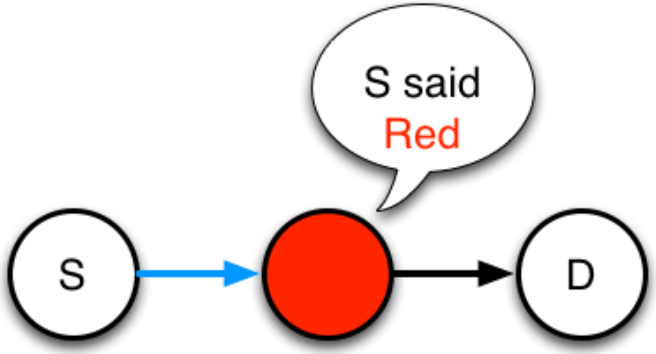
PKI



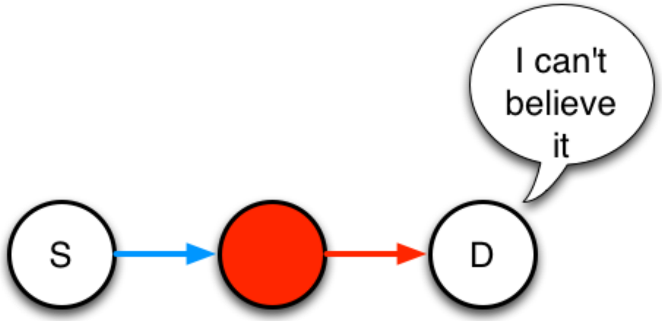
Example



Example



Example



Trusted third party

Trusted keys

Trusted Software

Arbitrary Networks

Topology Discovery

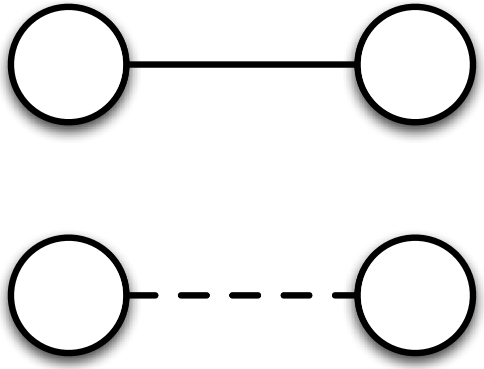
Topology Discovery

- **Given**
  - asynchronous network
  - up to *k* Byzantine nodes
  - each node knows its immediate neighbors identifiers
- **Goal**
  - each node must discover the complete network topology

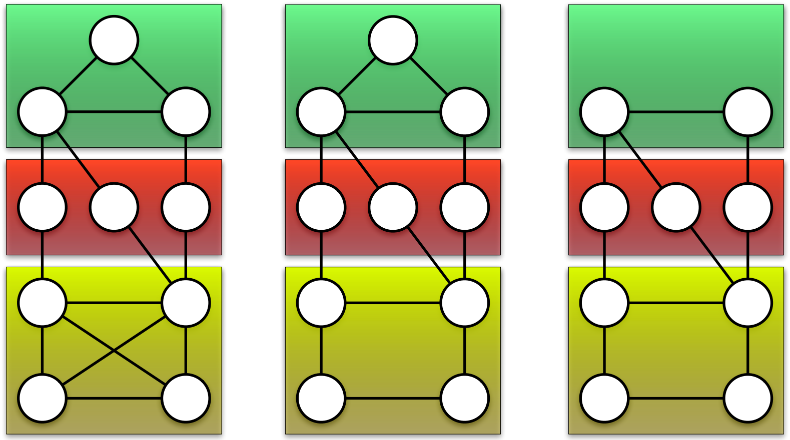
Weak Topology Discovery

- **Termination**
  - either all non-faulty processes determine the system topology or at least one detects fault
- **Safety**
  - for each non-faulty process, the determined topology is subset of actual
- **Validity**
  - fault detected only if it indeed exists

Weak Topology Discovery



Weak Topology Discovery



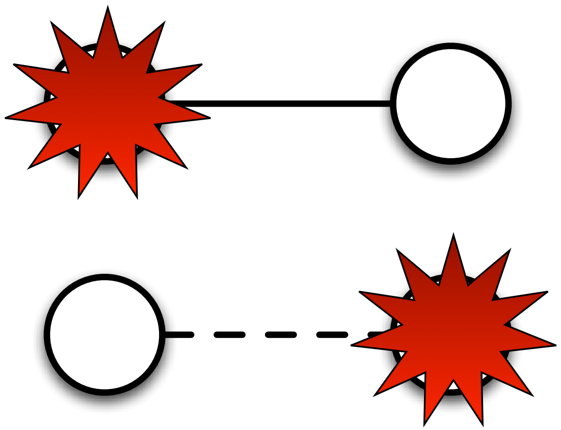
Weak Topology Discovery

- **Bounds**
  - cannot determine presence of edge if two adjacent nodes are faulty
  - cannot be (completely) solved if network is less than *k+1* connected

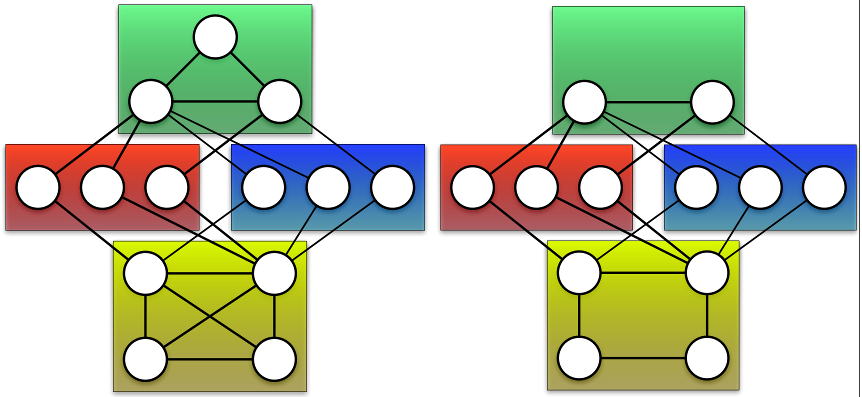
Strong Topology Discovery

- **Termination**
  - all non-faulty processes determine the system topology
- **Safety**
  - for each non-faulty process the determined topology is subset of actual

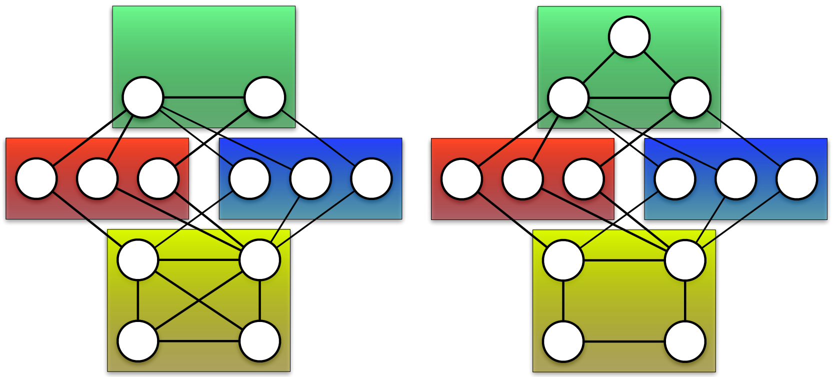
Strong Topology Discovery



Strong Topology Discovery



Strong Topology Discovery



Strong Topology Discovery

- **Bounds**
  - cannot determine presence of edge if one neighbor is faulty
  - cannot be solved if network is less than *2k+1* connected

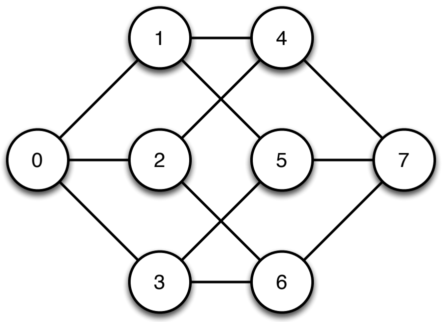
Solutions Preliminaries

- **Main idea**
  - *Menger's theorem*: if a graph is *k* connected then for any two vertices there exists two internally node-disjoint paths connecting them
  - a single (non-source) node cannot compromise info if it travels over two node-disjoint paths

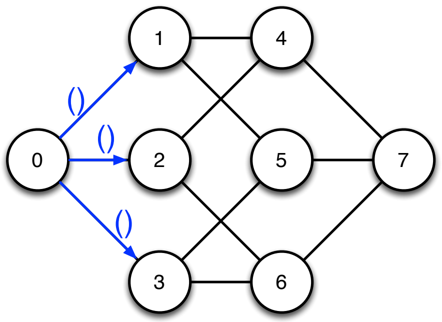
Dolev's Algorithm

- Store traveled path in message, forward message that contains simple path to all outgoing links
- Accept message if received through k+1 node-disjoint paths

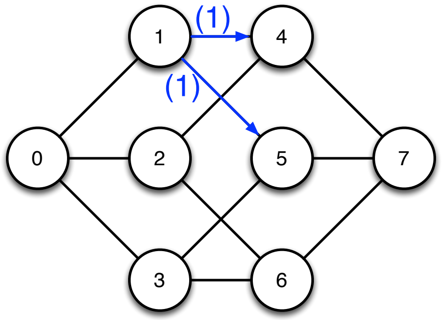
Dolev's Algorithm



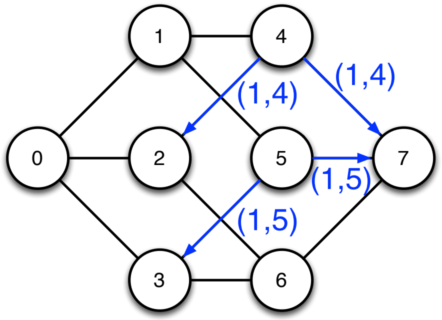
Dolev's Algorithm



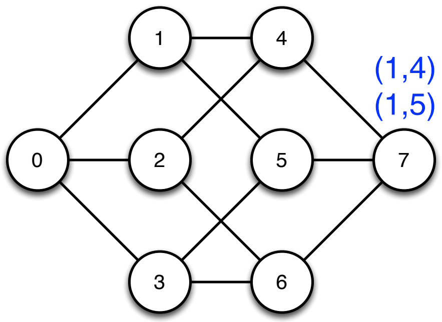
Dolev's Algorithm



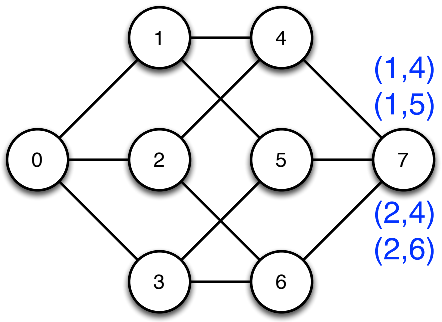
Dolev's Algorithm



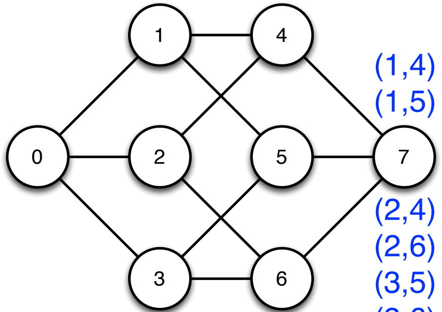
Dolev's Algorithm



Dolev's Algorithm

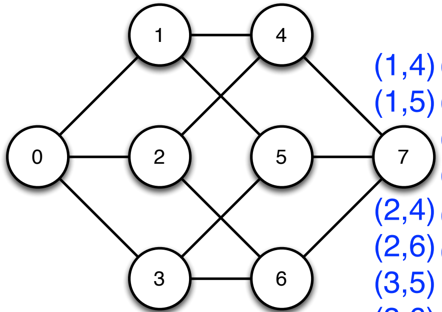


Dolev's Algorithm



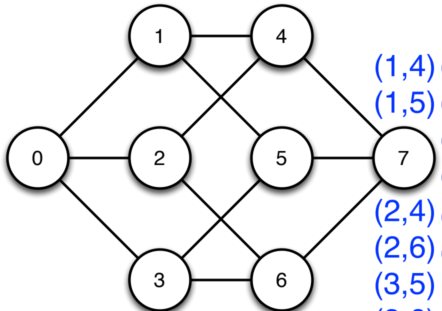
(1,4)
(1,5)
(2,4)
(2,6)
(3,5)
(3,6)

Dolev's Algorithm



(1,4) (1,4,2,6)
(1,5) (1,5,3,6)
(2,4,1,5)
(2,6,3,5)
(2,4) (3,5,1,4)
(2,6) (3,6,2,4)
(3,5)
(3,6)

Dolev's Algorithm



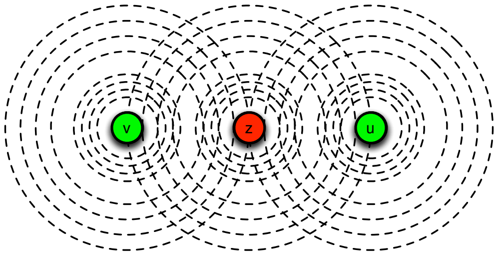
(1,4) (1,4,2,6) (1,4,2,6,3,5)
(1,5) (1,5,3,6) (1,5,3,6,2,4)
(2,4,1,5) (2,4,1,5,3,6)
(2,6,3,5) (2,6,3,5,1,4)
(2,4) (3,5,1,4) (3,6,2,4,1,5)
(2,6) (3,6,2,4) (3,5,1,4,2,6)
(3,5)
(3,6)

Wireless Networks

Secure Positioning

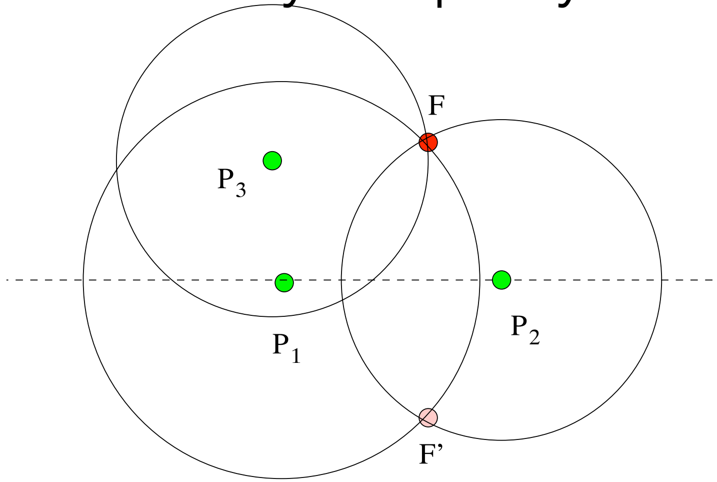
Traps and Pitfalls

- No way to assess sender

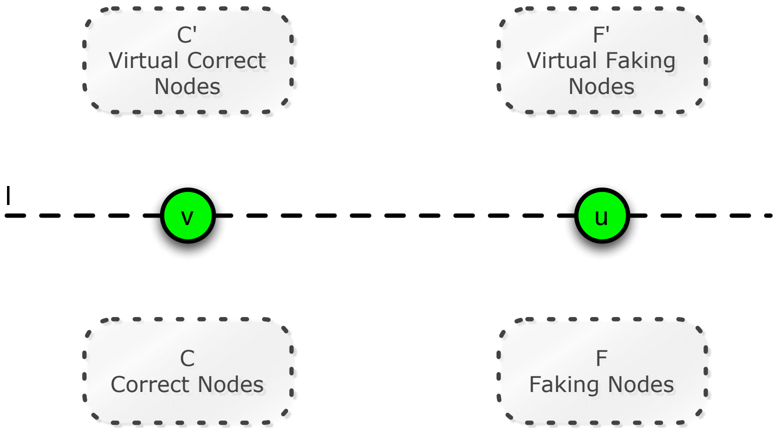


- Byzantine must lie consistently

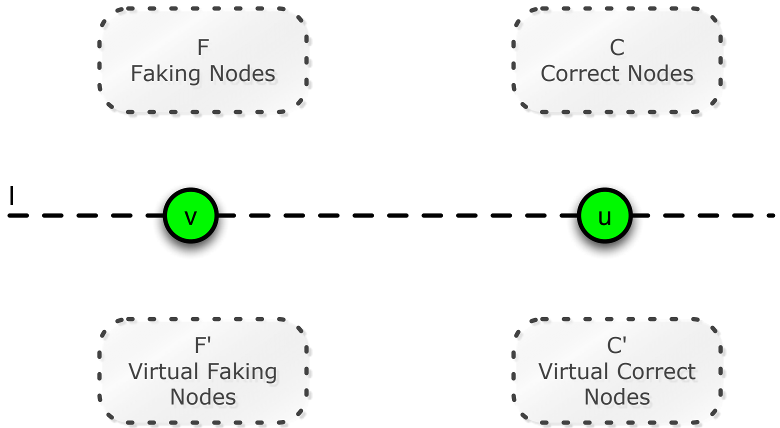
A Key Property



Lower bound



Lower bound



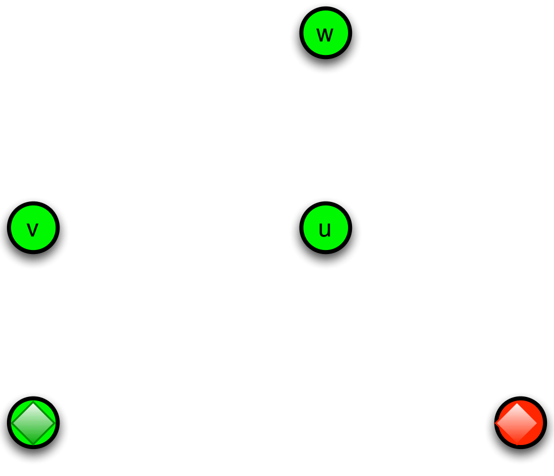
Assumptions

- No three nodes are colinear
- No more than *f* faking nodes, with *n-f-2 > f*
- Distance is impossible to fake
- Faking nodes send at most one message per round

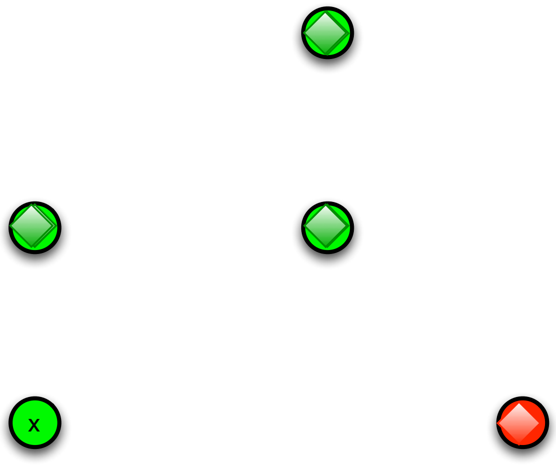
A Naive Protocol

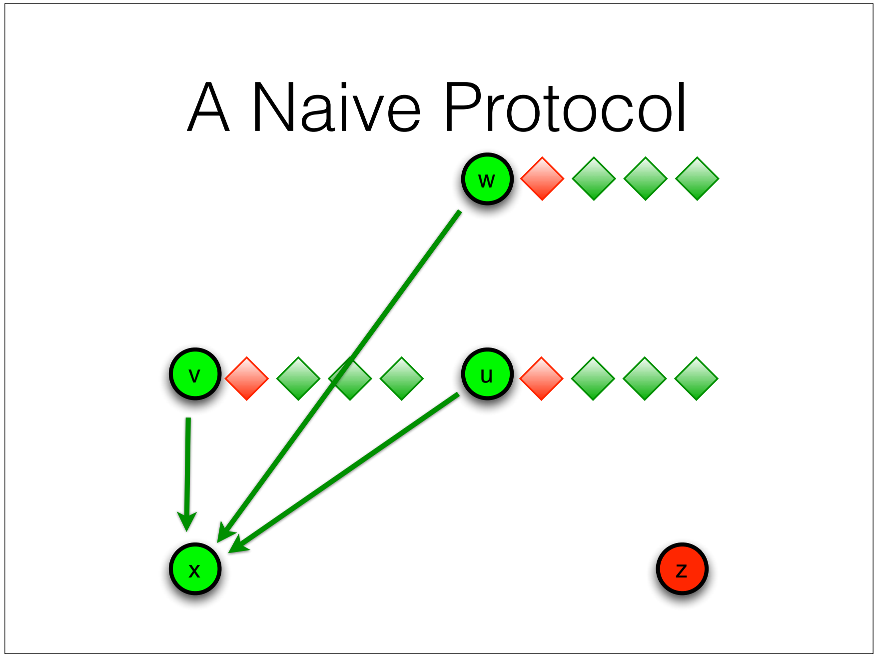
- For every annoucement by a node v
  - *Report* **OK(v)** if perceived distance matches annouced distance, else *report* **KO(v)**
- *Count* **OK(v)**s and **KO(v)**s for every report
  - If #**KO(v)** > #**OK(v)** - 2, v is faulty

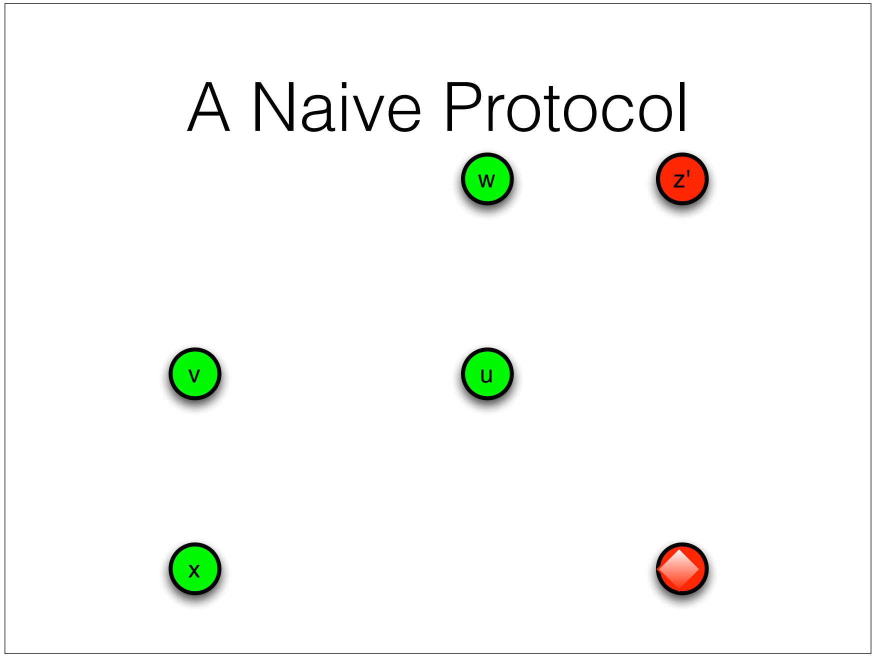
A Naive Protocol

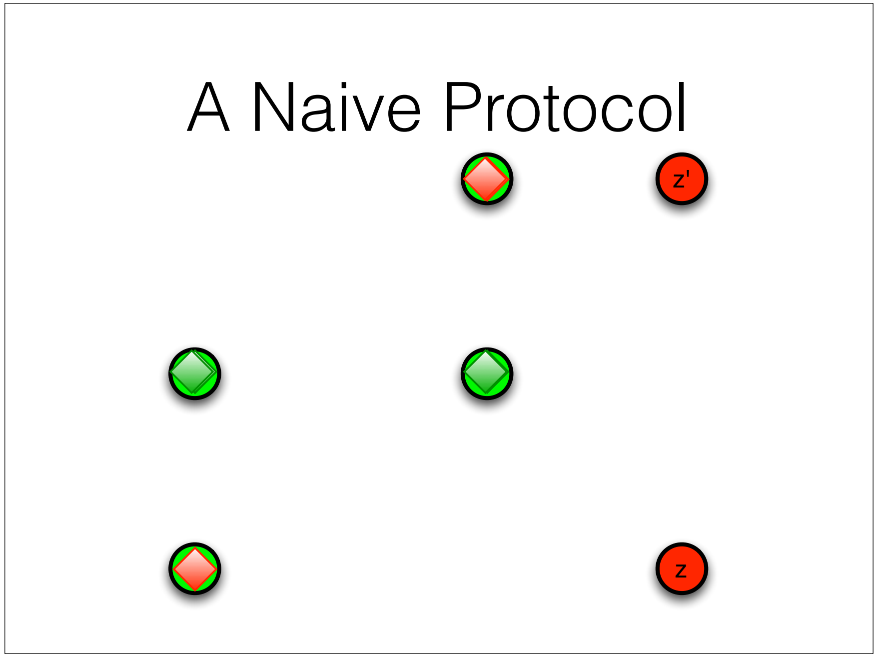


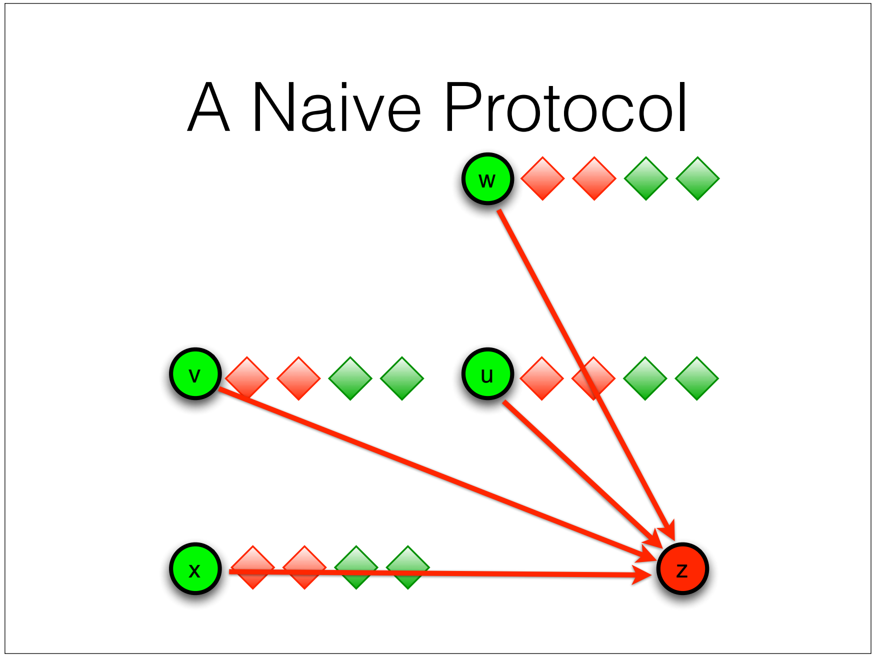
A Naive Protocol




A Naive Protocol
w
v
u
x
z


A Naive Protocol
w
z'
v
u
x

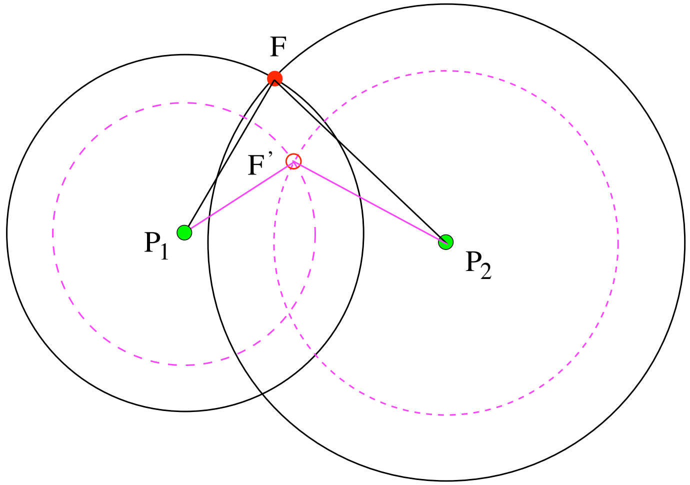

A Naive Protocol
z'
z


A Naive Protocol
w
v
u
x
z

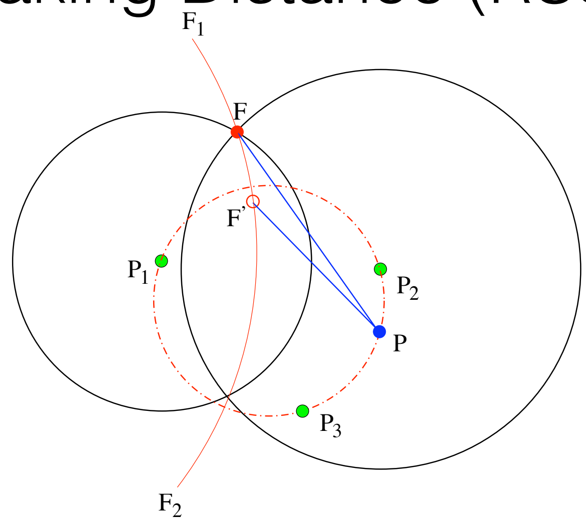
Faking the Distance

- **RSS** $S_r = S_s \left(\frac{\lambda}{4\pi d} \right)^2$
  - Change emitting signal strength
  - Must be consistent for *all* nodes
- **ToF** & **DAT**
  - Change processing speed or timestamps
  - Must be consistent for *all* nodes

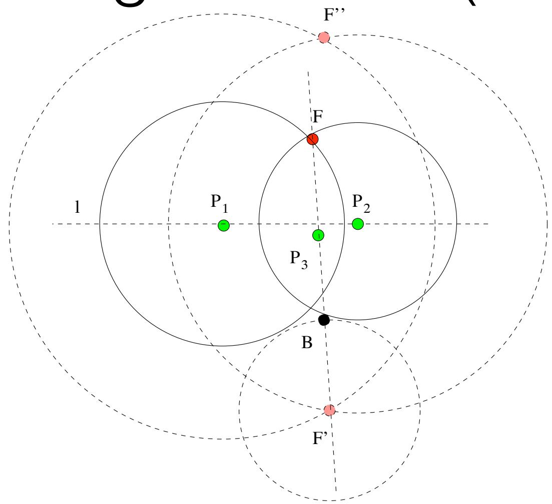
Faking Distance (RSS)



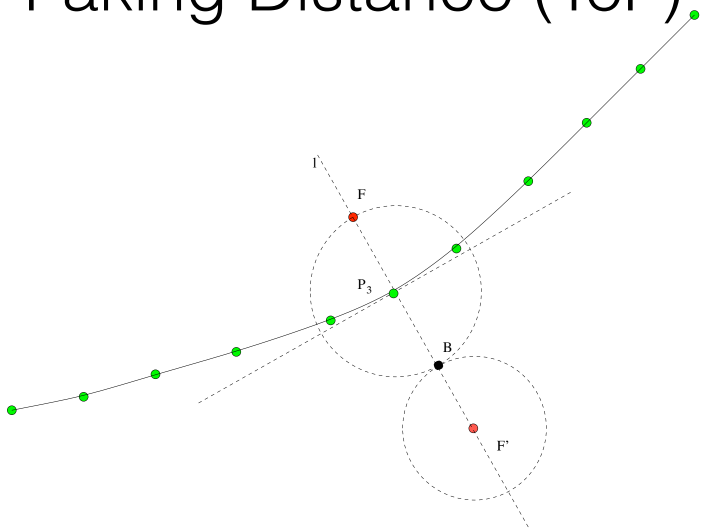
Faking Distance (RSS)



Faking Distance (ToF)



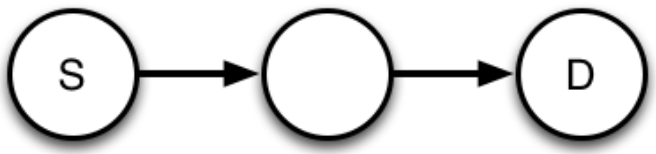
Faking Distance (ToF)



Dynamic Networks

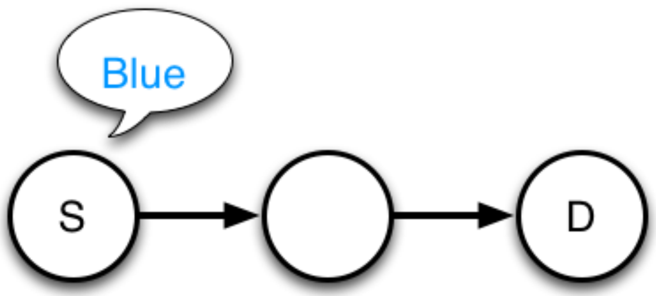
Reliable Broadcast

Context

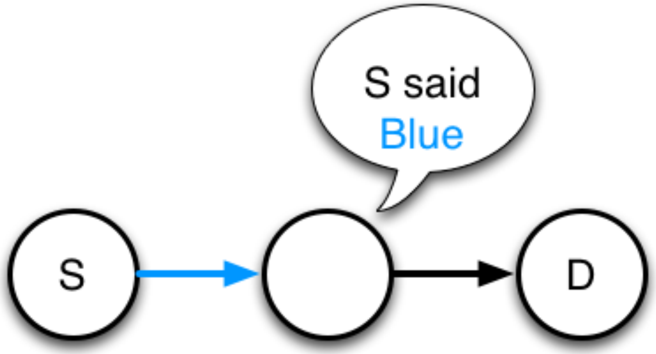


Information broadcast in multi hop networks

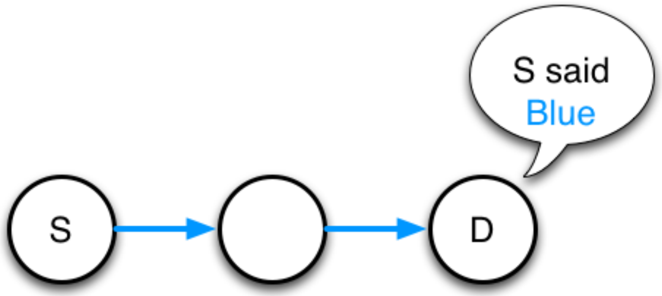
Example



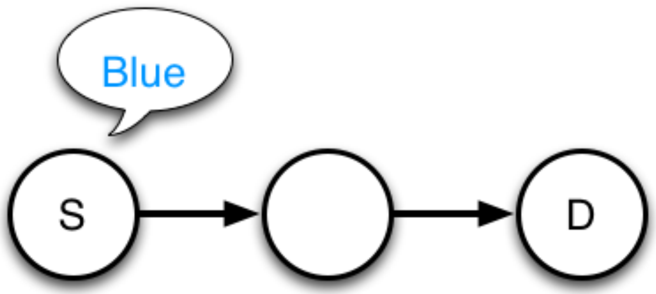
Example



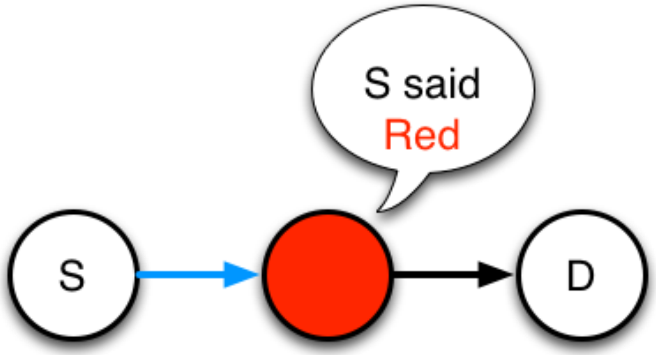
Example



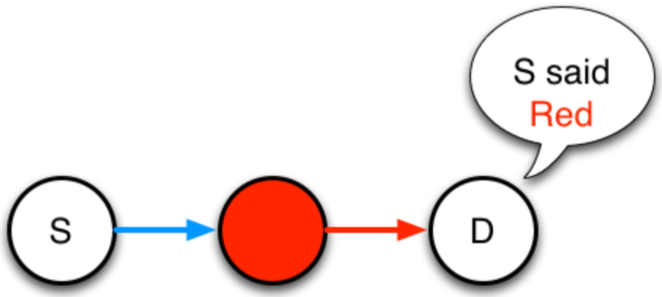
Example



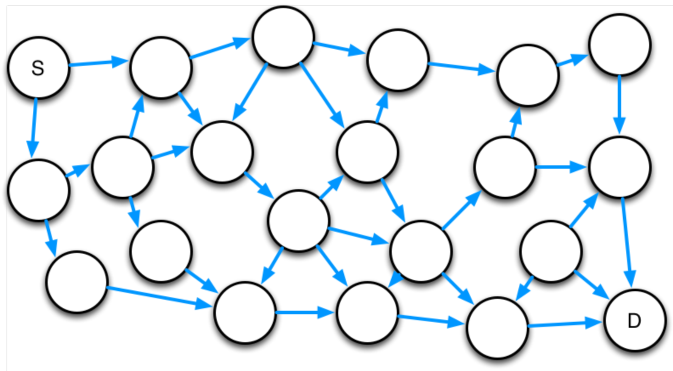
Example



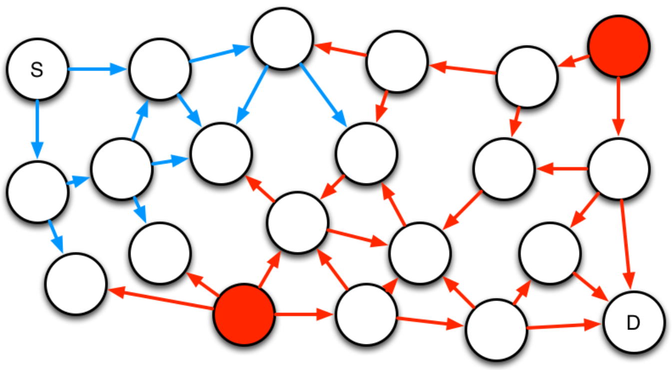
Example



Information Broadcast



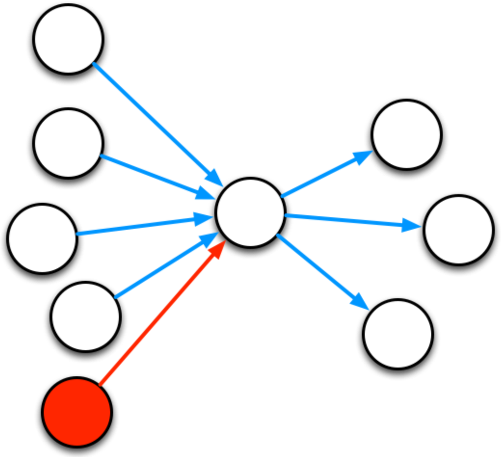
Information Broadcast



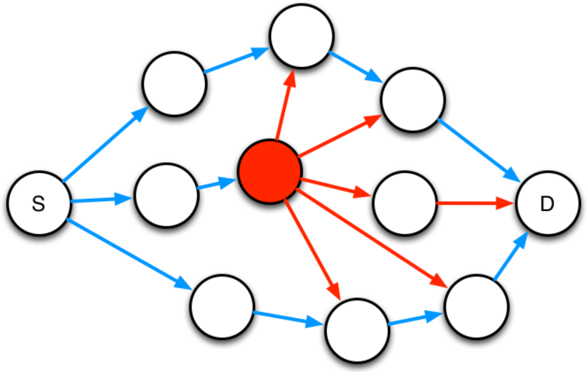
Objective

- **Broadcast** algorithms resilient to **Byzantine** Failures
  - No **false** message ever accepted
  - **Correct** messages always received

Local Vote



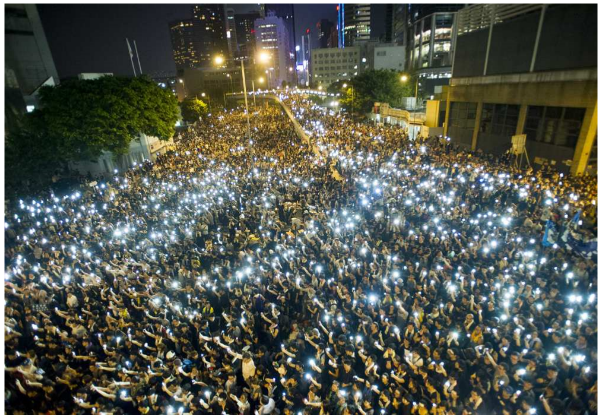
Vote on Multiple Paths



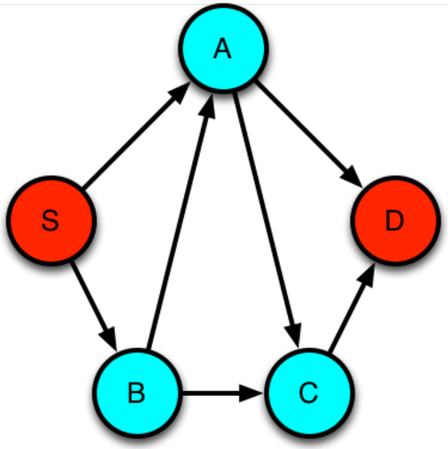
Condition for reliable communication in static networks

- k = number of Byzantine nodes
- **Condition**: 2k+1 node-disjoint paths between the source and the destination

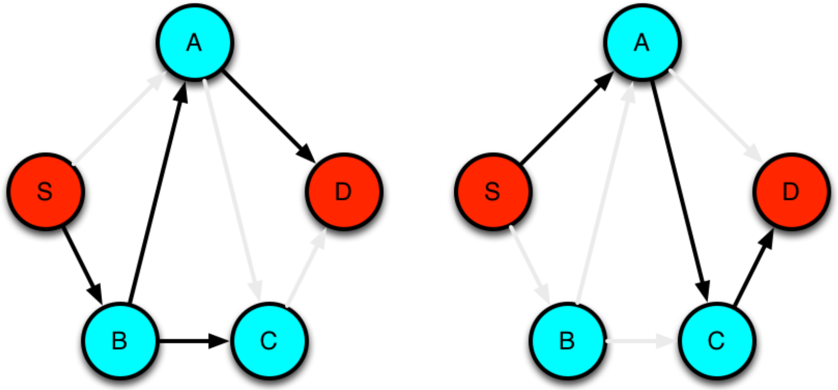
Enter Dynamic Networks



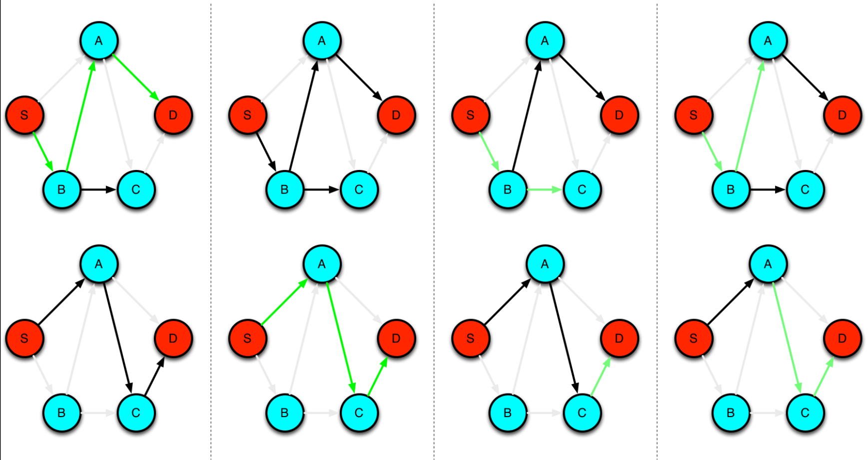
Menger's Theorem



Menger's Theorem



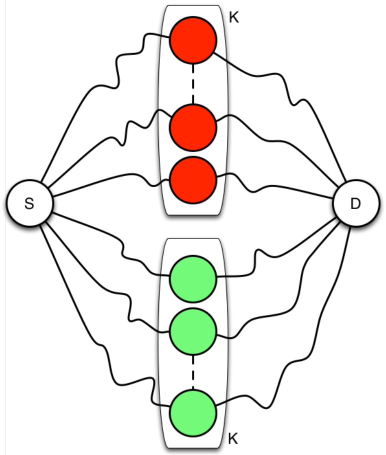
Menger's Theorem



Condition in Dynamic Networks

- k=number of Byzantine nodes
- **Condition**: 2k+1 nodes must be removed to cut all dynamic paths

Necessary Condition



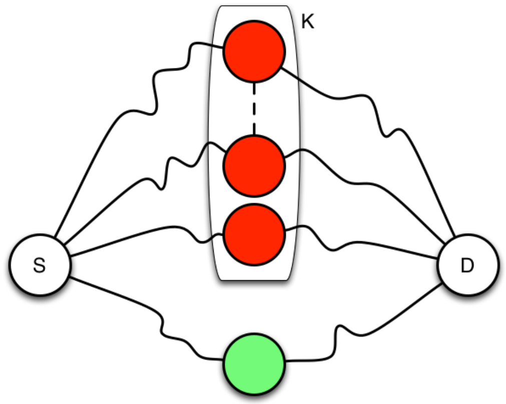
Sufficient Condition

- Send the message through *all* journeys
- Register the journeys
- When a set of journeys that cannot be cut by *2k* nodes is collected, accept the message

Condition in Dynamic Networks with Cryptography

- k = number of Byzantine nodes
- **Condition**: k+1 nodes must be removed to cut all journeys

Necessary Condition with Cryptography



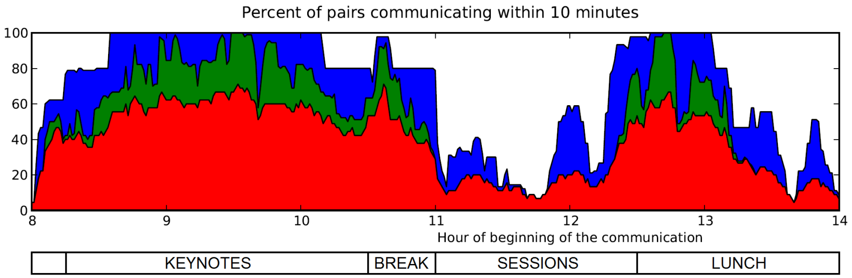
Sufficient Condition with Cryptography

- Send the message through all journeys
- When a cryptographically acceptable message arrives, accept it

Case Studies

- Participants in a conference
- Agents in the subway

Participants Interacting in a Conference

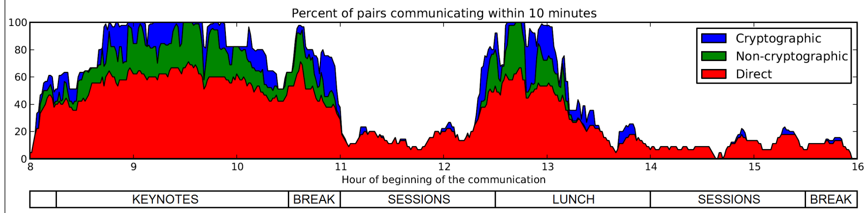


Unreliable multihop communication

Reliable multi hop communication

Direct communication

Participants Interacting in a Conference

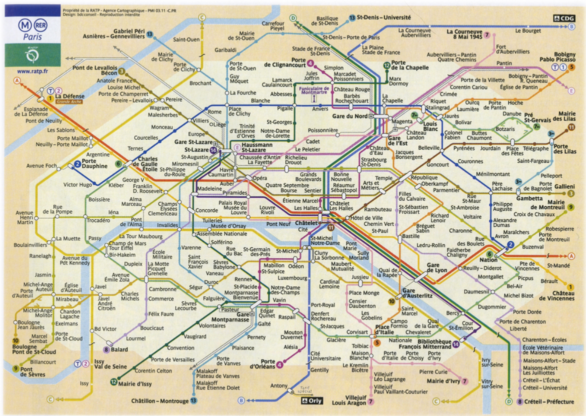


Cryptographic multihop communication

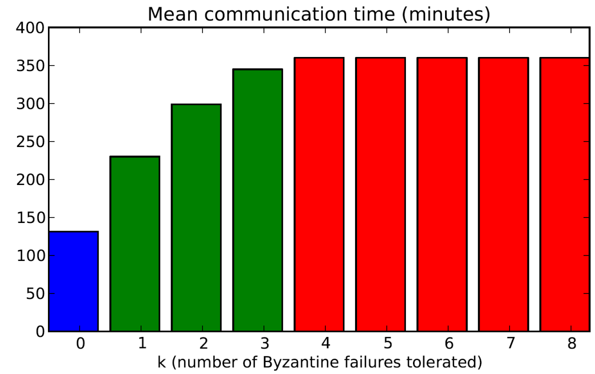
Non-Cryptographic multi hop communication

Direct communication

Paris Subway Users

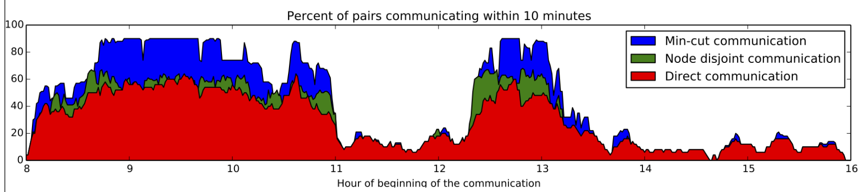


Paris Subway Users



Unreliable multihop communication
Reliable multi hop communication
Direct communication

IF vs. IFF



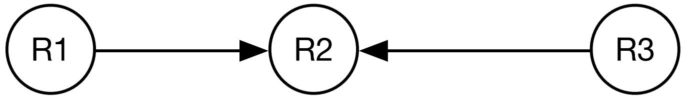
Min-cut multihop communication
Node-disjoint multihop communication
Direct communication

Byzantine Robots

Gathering and Convergence

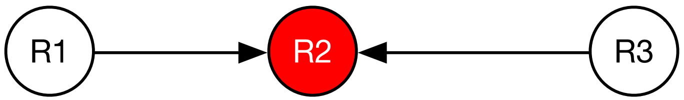
Gathering

Possibility of FSYNC Byzantine Gathering with n>3f



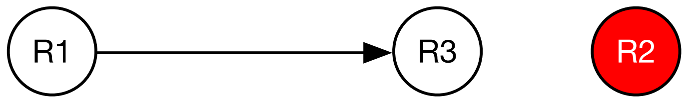
Noa Agmon, David Peleg: Fault-Tolerant Gathering Algorithms for Autonomous Mobile Robots. SIAM J. Comput. 36(1): 56-82 (2006)

Possibility of FSYNC Byzantine Gathering with n>3f

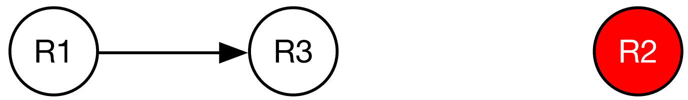


Possibility of FSYNC Byzantine Gathering with n>3f

Possibility of FSYNC
Byzantine Gathering with n>3f

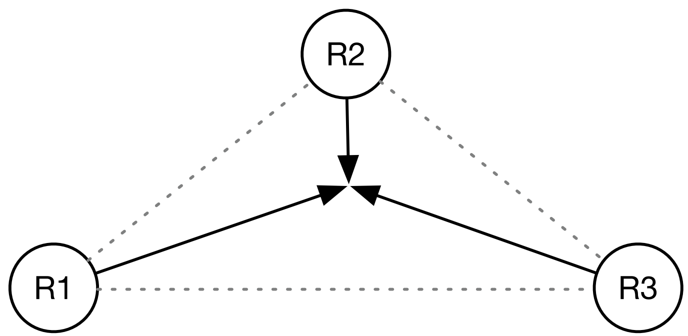


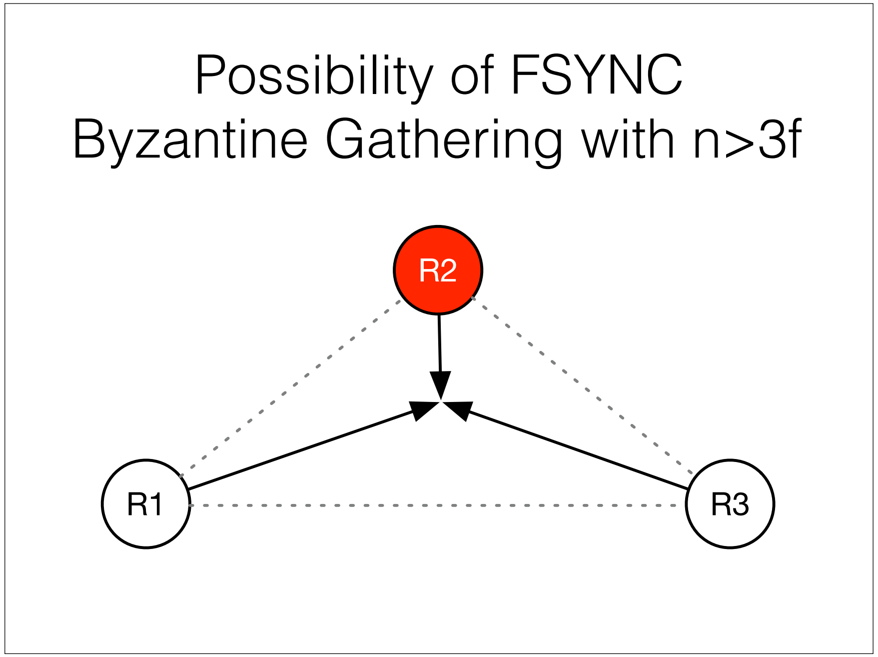
Possibility of FSYNC
Byzantine Gathering with n>3f

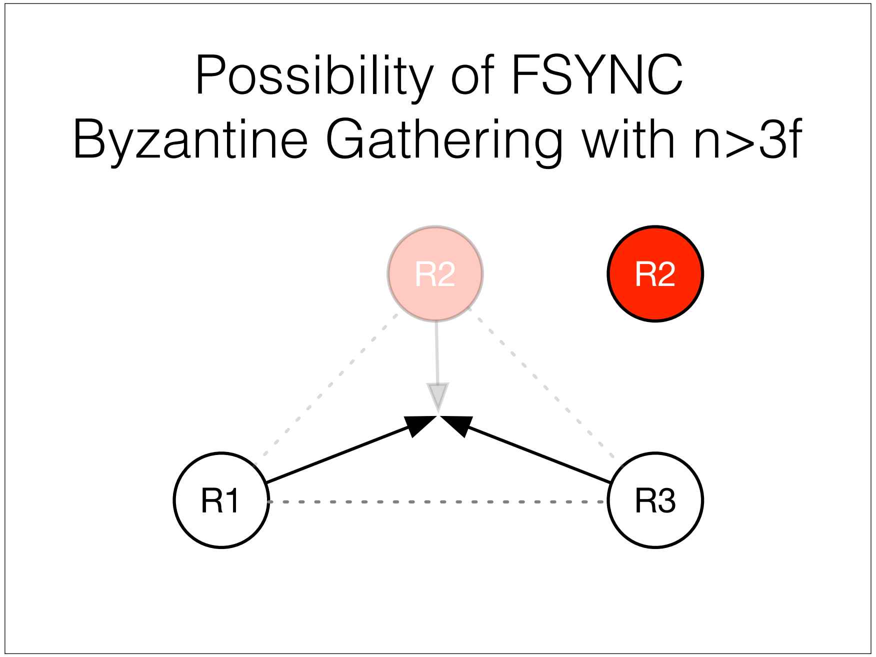


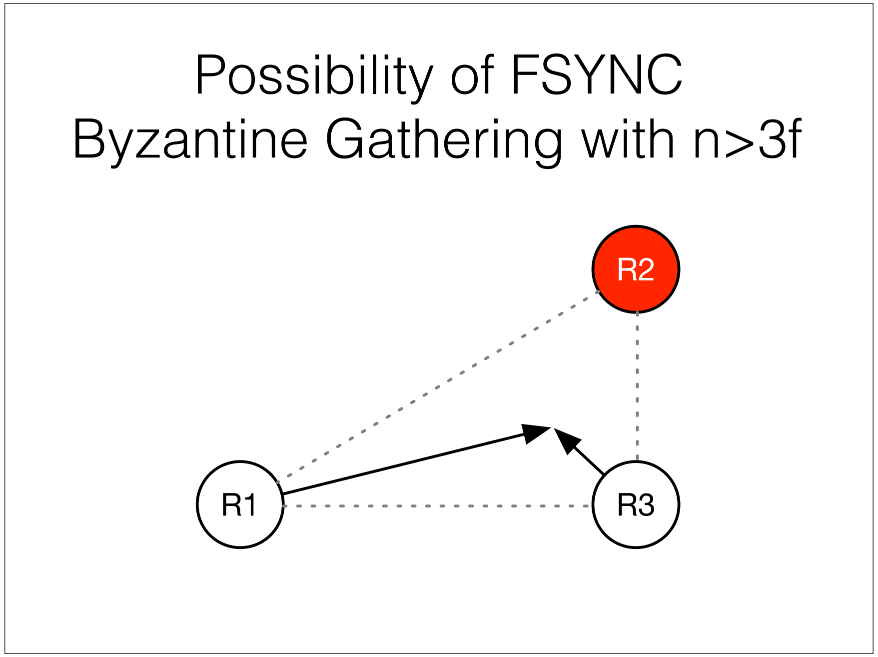
Possibility of FSYNC
Byzantine Gathering with n>3f

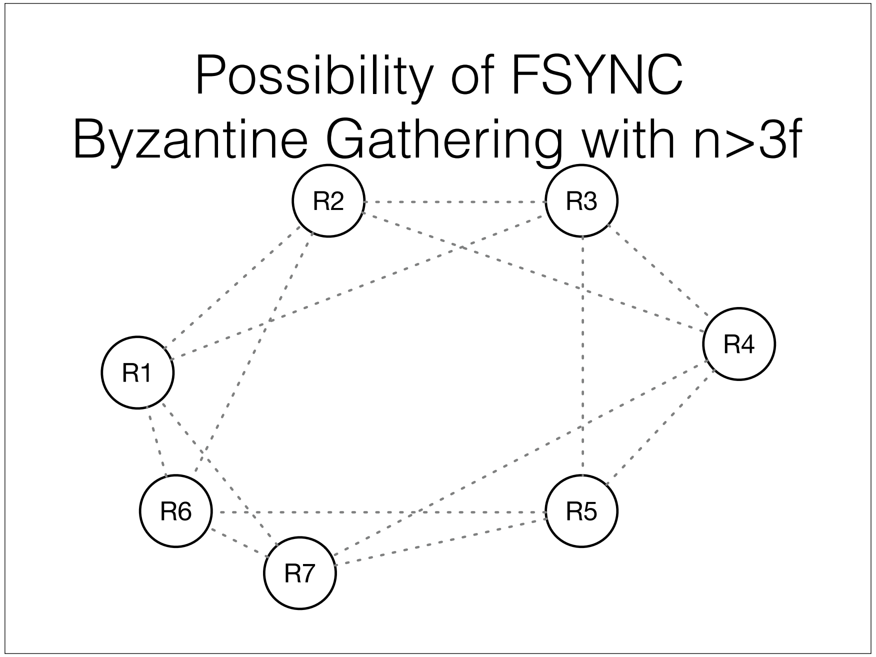
Possibility of FSYNC
Byzantine Gathering with n>3f



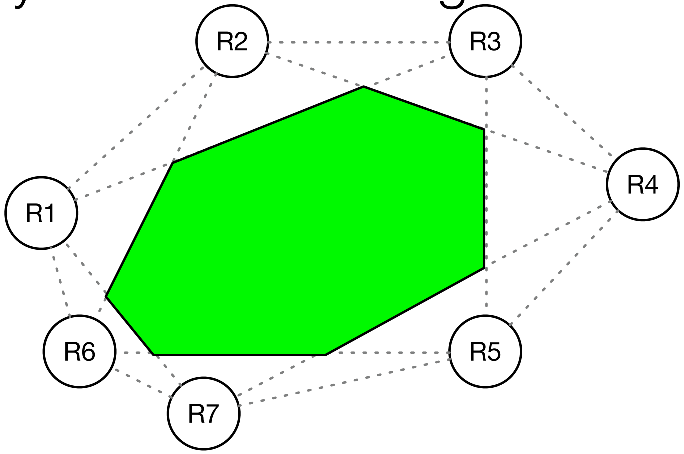

Possibility of FSYNC
Byzantine Gathering with n>3f
R2
R1
R3


Possibility of FSYNC
Byzantine Gathering with n>3f
R2
R2
R1
R3

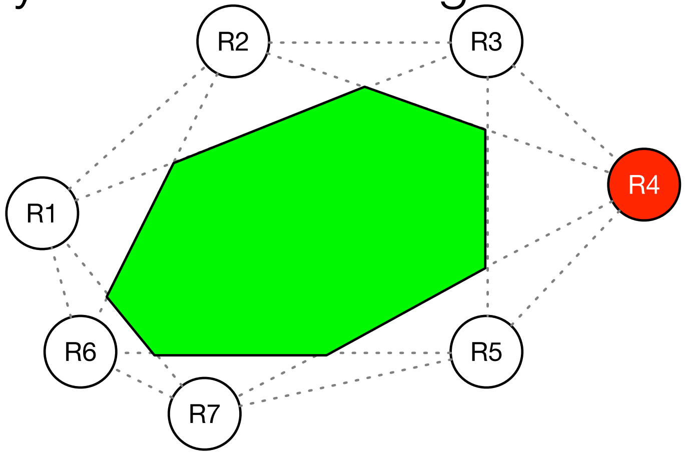

Possibility of FSYNC
Byzantine Gathering with n>3f
R2
R1
R3


Possibility of FSYNC
Byzantine Gathering with n>3f
R2
R3
R4
R1
R6
R5
R7

Possibility of FSYNC Byzantine Gathering with n>3f



Possibility of FSYNC Byzantine Gathering with n>3f



Byzantine Tolerant Gathering and Convergence

	2D Gathering
FSYNC	Yes *n>3f*
SSYNC	No n=3, f=1
ASYNC	

Noa Agmon, David Peleg: Fault-Tolerant Gathering Algorithms for Autonomous Mobile Robots. SIAM J. Comput. 36(1): 56-82 (2006)

Byzantine Tolerant Gathering and Convergence

	2D Gathering
FSYNC	Yes *n>3f*
SSYNC	No, n>f, f>0 bounded scheduler & randomness
ASYNC	

Xavier Défago, Maria Gradinariu Potop-Butucaru, Julien Clément, Stéphane Messika, Philipp Raipin Parvédy: Fault and Byzantine Tolerant Self-stabilizing Mobile Robots Gathering - Feasibility Study -. CoRR abs/1602.05546 (2016)

Byzantine Tolerant Gathering and Convergence

	2D Gathering
FSYNC	Yes *n>3f*
SSYNC	No, n>f, f>0, deterministic bounded scheduler & memory & non uniform & common axes
ASYNC	

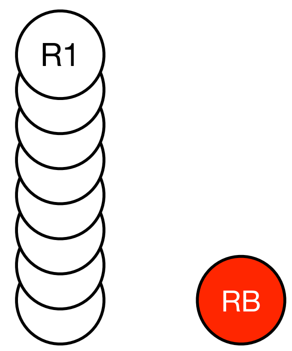
Taisuke Izumi, Zohir Bouzid, Sébastien Tixeuil, Koichi Wada: The BG-simulation for Byzantine Mobile Robots. CoRR abs/1106.0113 (2011)

Convergence

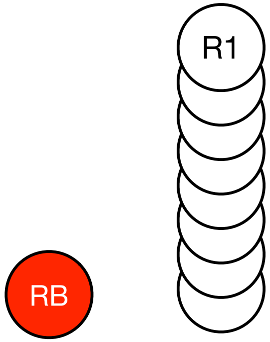
1D Convergence with Byzantine Robots

- **Shrinking**: the distance between correct robots eventually decreases
- **Cautious**: positions of correct robot always remain within the range of correct robots
- Shrinking is *necessary*
- Shrinking+Cautious is *sufficient*

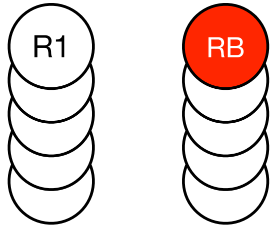
Weak Multiplicity Detection is Necessary



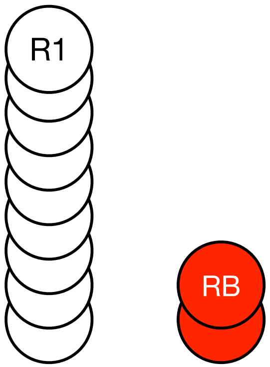
Weak Multiplicity Detection is Necessary



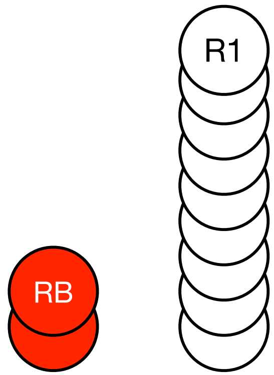
Weak Multiplicity Detection is Necessary



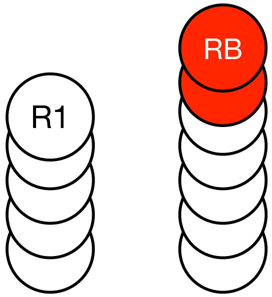
Strong Multiplicity Detection is Necessary



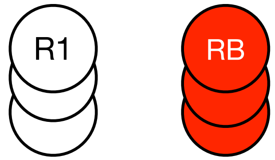
Strong Multiplicity Detection is Necessary



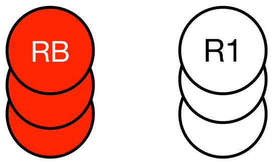
Strong Multiplicity Detection is Necessary



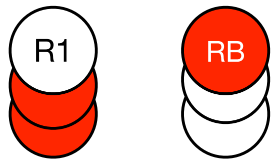
n>2f is Necessary in FSYNC



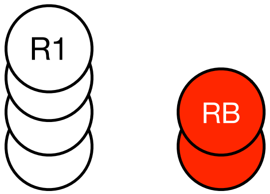
n>2f is Necessary in FSYNC



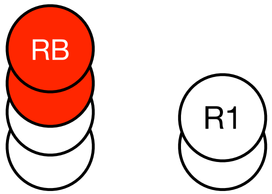
n>2f is Necessary in FSYNC



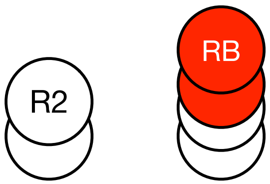
n>3f is Necessary in SSYNC



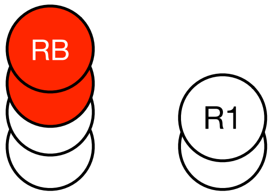
n>3f is Necessary in SSYNC



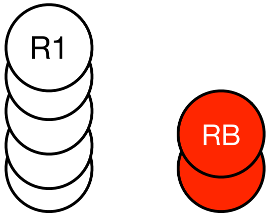
n>3f is Necessary in SSYNC



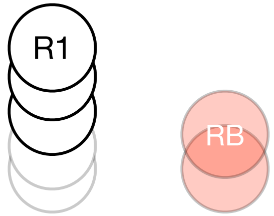
n>3f is Necessary in SSYNC



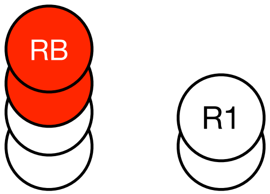
1D Convergence with f Byzantine Robots



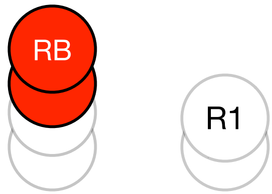
1D Convergence with f Byzantine Robots



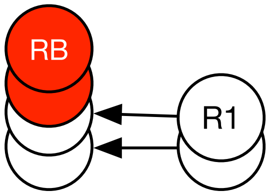
1D Convergence with f Byzantine Robots



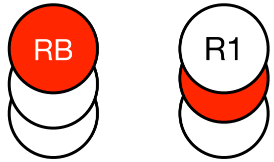
1D Convergence with f Byzantine Robots



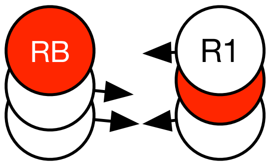
1D Convergence with f Byzantine Robots



1D Convergence with f Byzantine Robots



1D Convergence with f Byzantine Robots



Byzantine Tolerant Gathering and Convergence

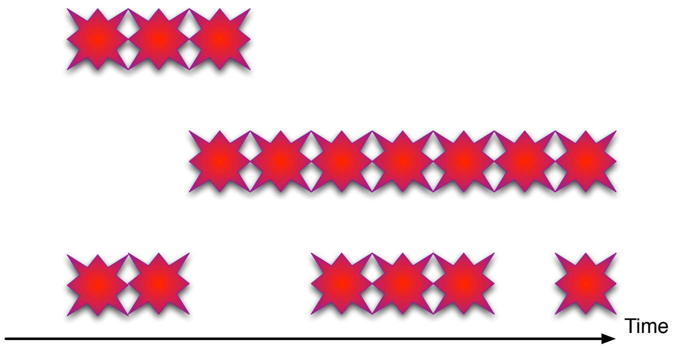
	2D Gathering	1D Convergence
FSYNC	Yes *n>3f*	Yes *n>2f*
SSYNC	No*	Yes *n>3f*
ASYNC		Yes *n>5f*

Open Questions (Byzantine Robots)

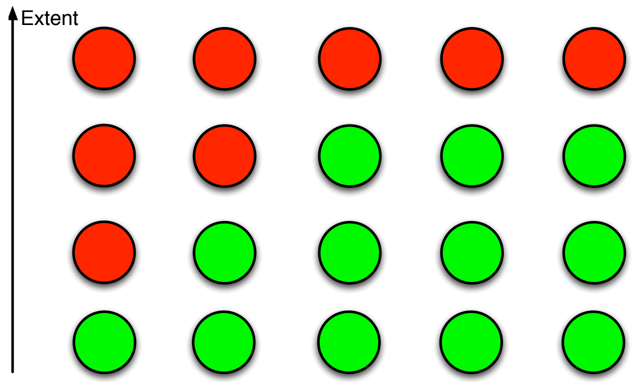
- Lower bound for 2D FSYNC Gathering (w.r.t. *f*)?
- Sufficient condition for 2D SSYNC Gathering?
- Sufficient condition for 2D Convergence?

Faults, Attacks, and Fault-tolerance

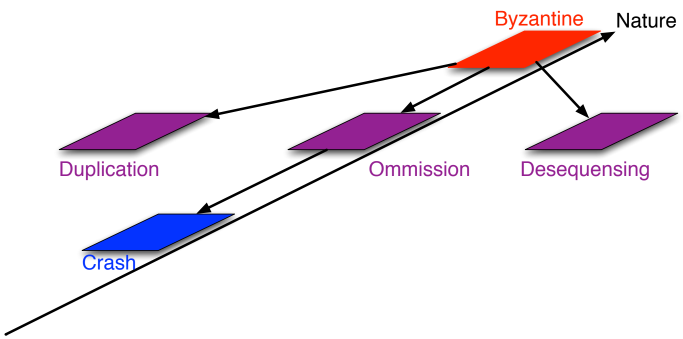
Faults & Attacks



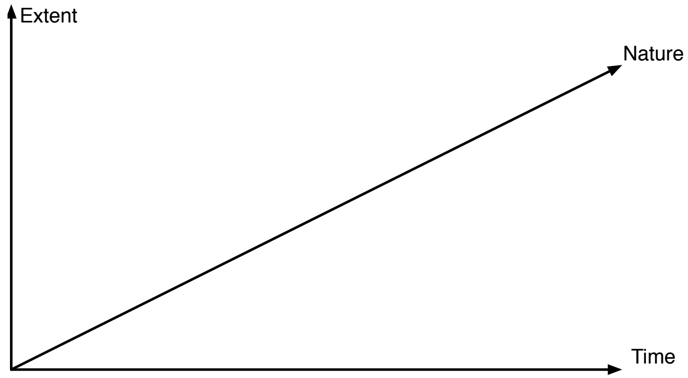
Faults & Attacks



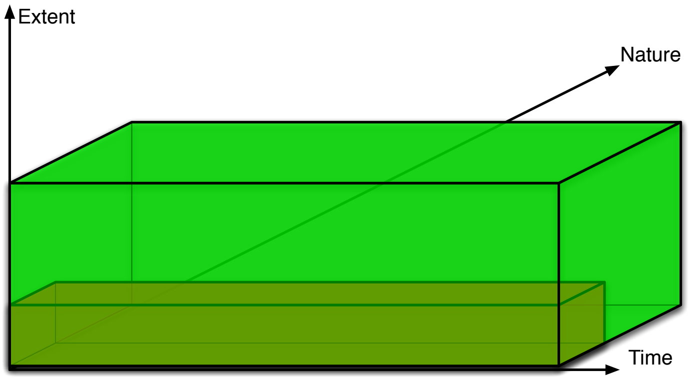
Faults & Attacks



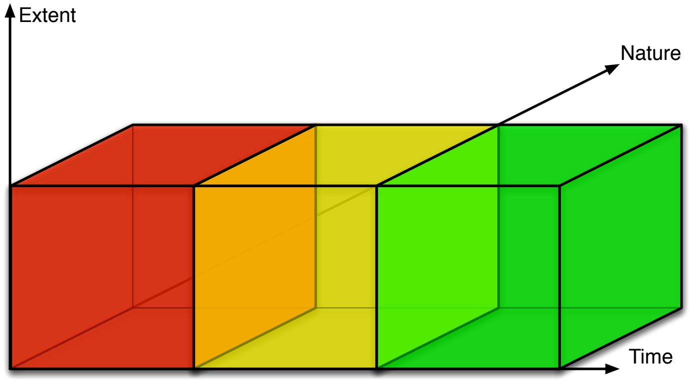
F&A Tolerance



Robust Protocols



Self-stabilization



Multi-Tolerance ?