



# Network-Aware Distributed Algorithms

Nitin Vaidya

Electrical and Computer Engineering

University of Illinois at Urbana-Champaign



# Acknowledgements

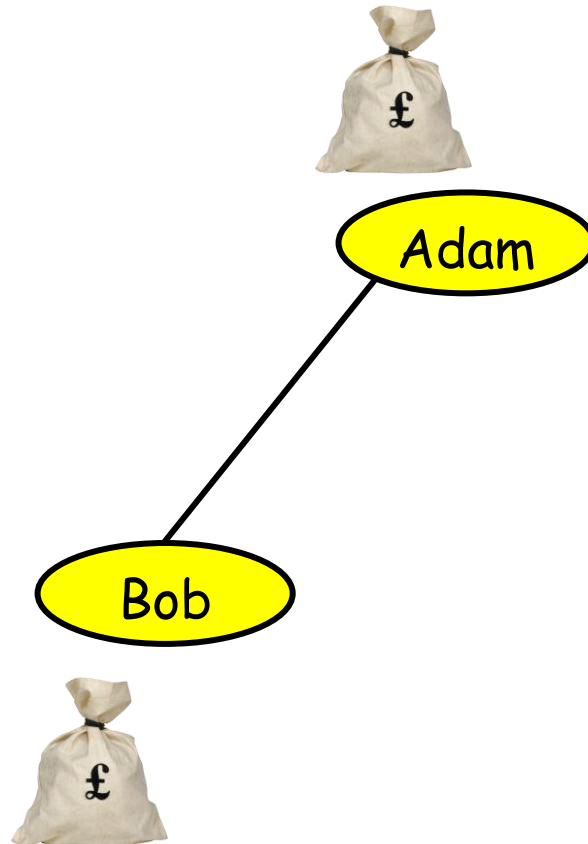
- Guanfeng Liang
- Lewis Tseng
- Prof. Alejandro Dominguez-Garcia
- Prof. Christoforos Hadjicostis

Work supported in part by



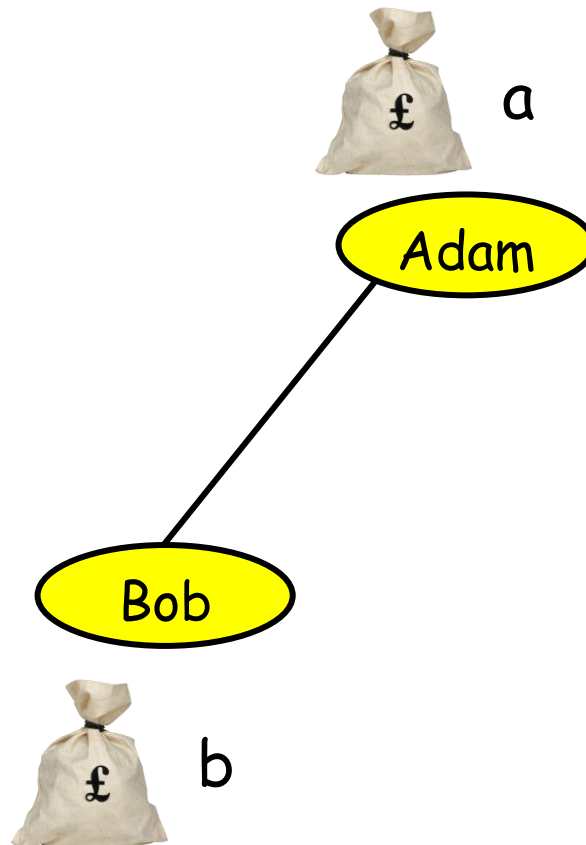
# Puzzler

# Puzzler



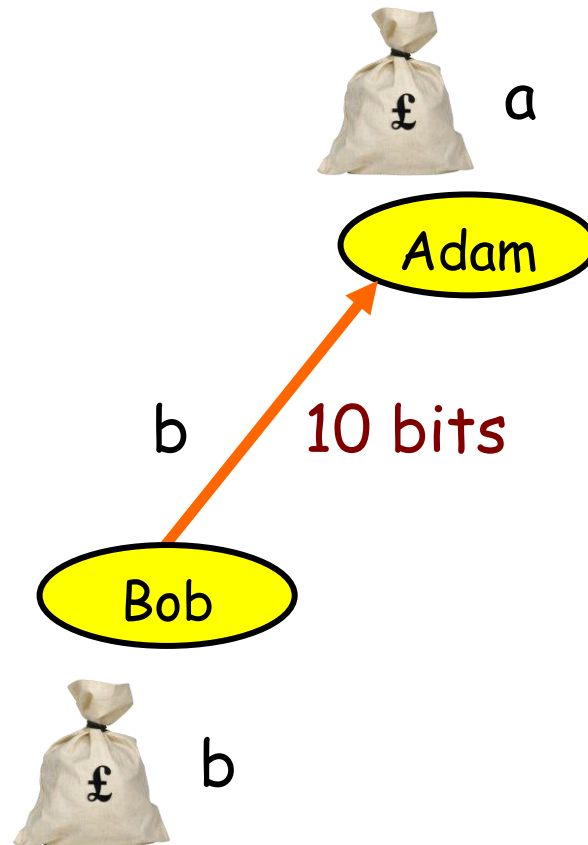
# Puzzler

10-bit inputs



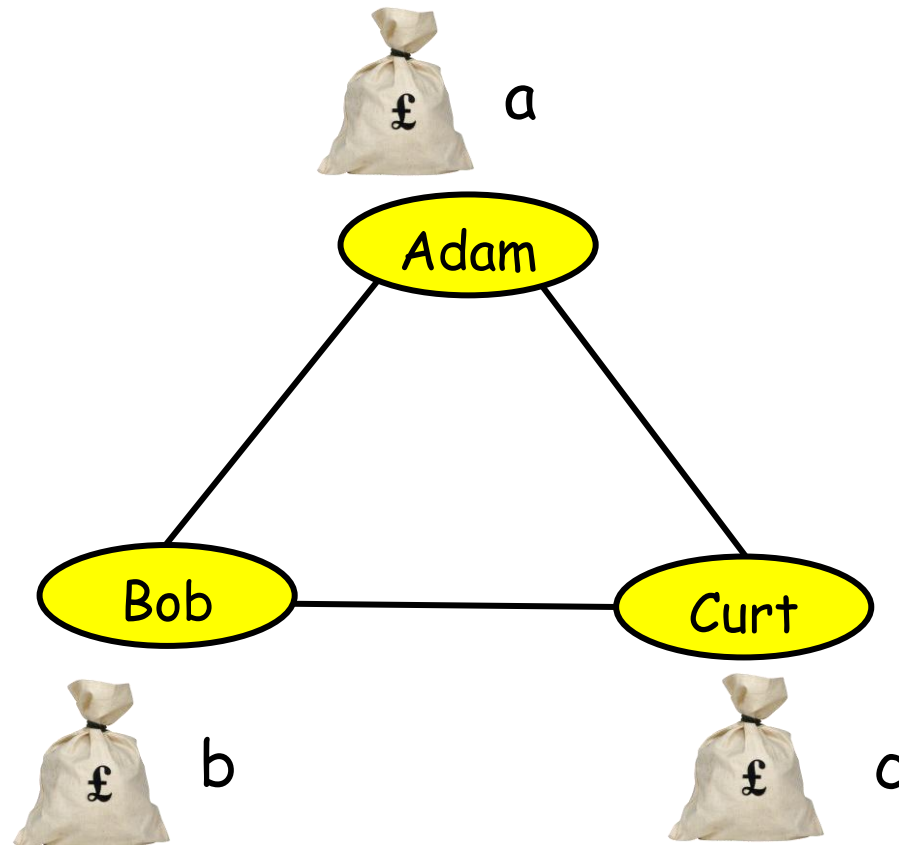
# Puzzler

10-bit inputs



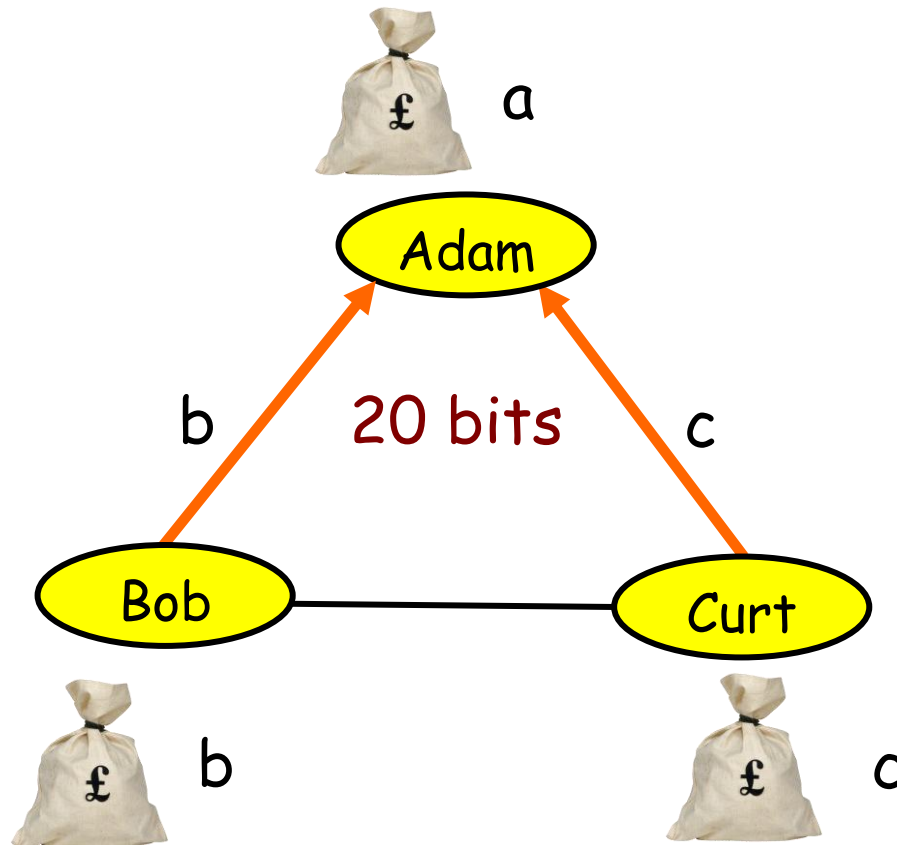
# Puzzler

10-bit inputs



# Puzzler

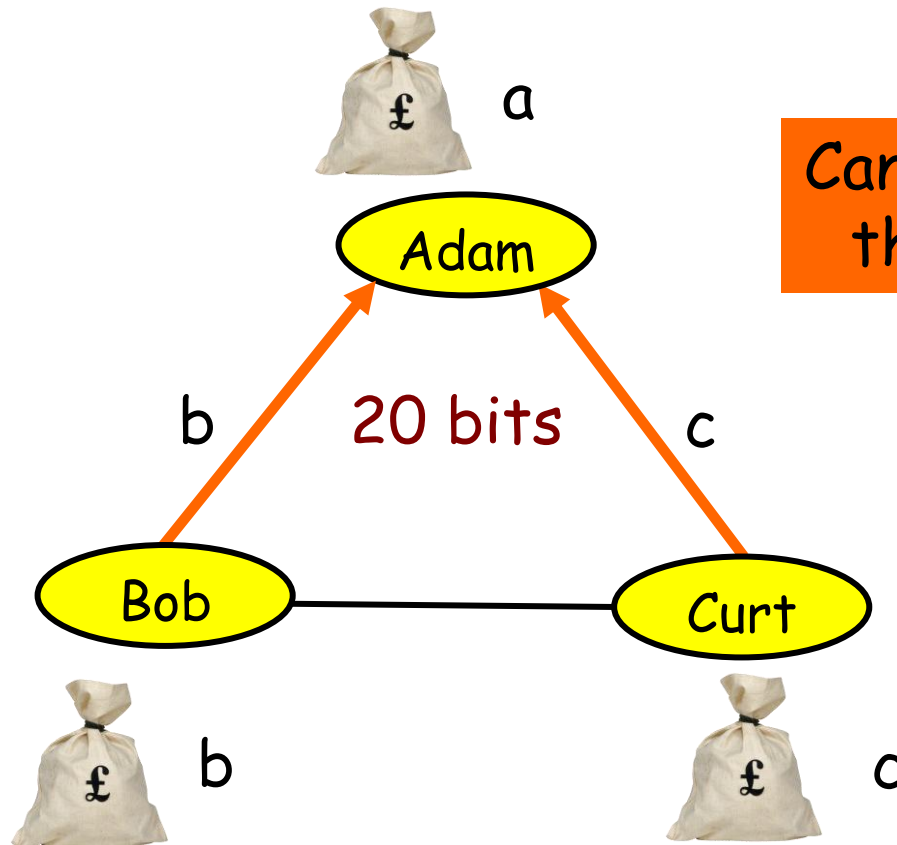
10-bit inputs





# Puzzler

10-bit inputs

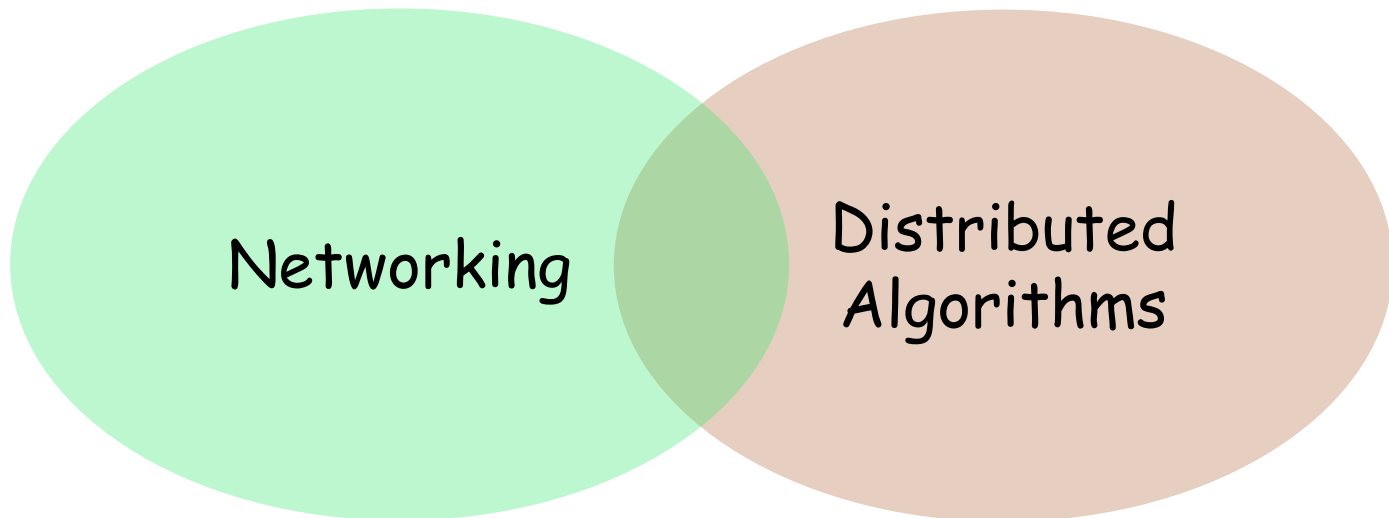


Can we do better than 20 bits?

Now back to the

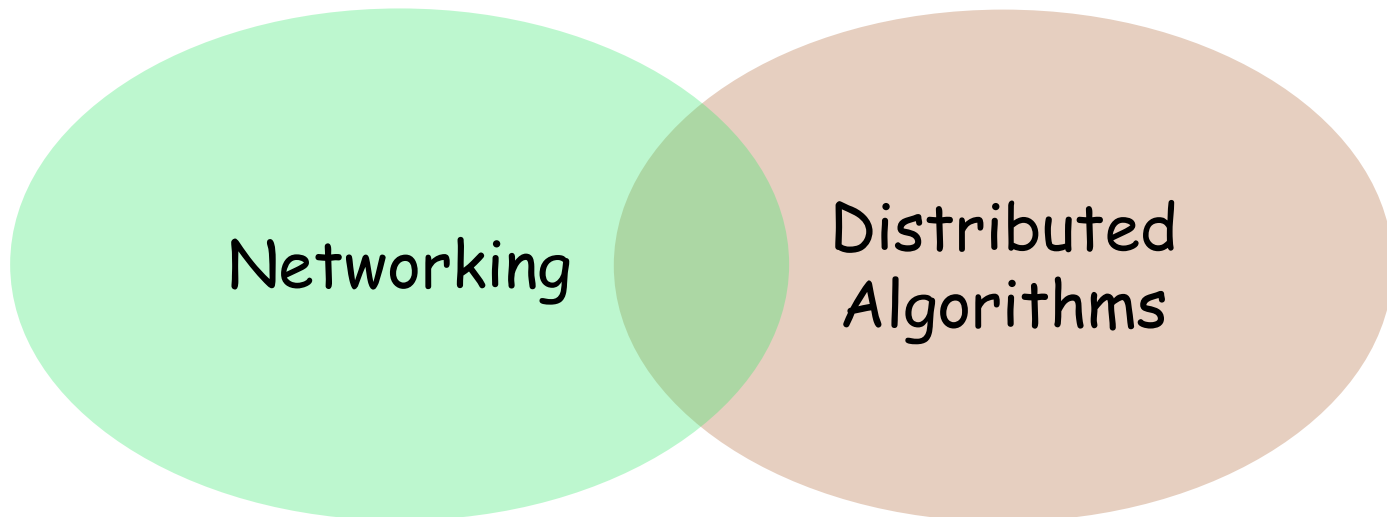
originally scheduled program ...

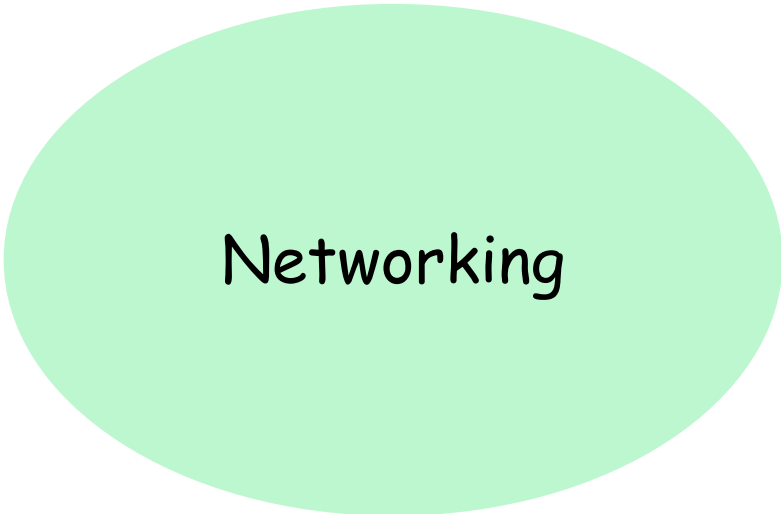
# Network-Aware Distributed Algorithms



# Distributed Algorithms & Networking

- Problems with overlapping scope
- But cultures differ

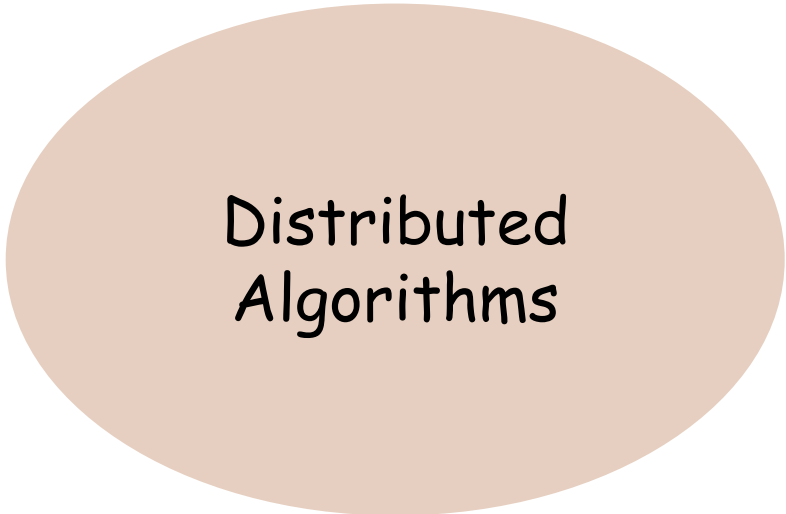




Networking

"Accurate" network  
models

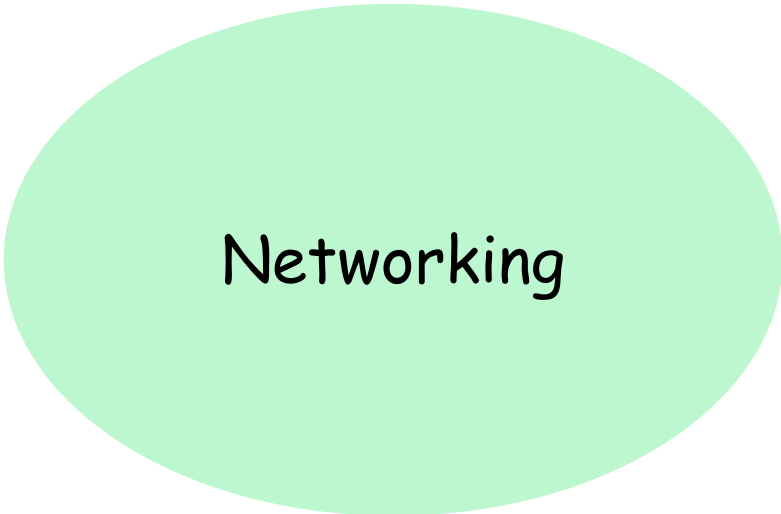
Constants matter



Distributed  
Algorithms

Simple network models

Emphasis on  
order complexity



## Networking

"Accurate" network models

Constants matter

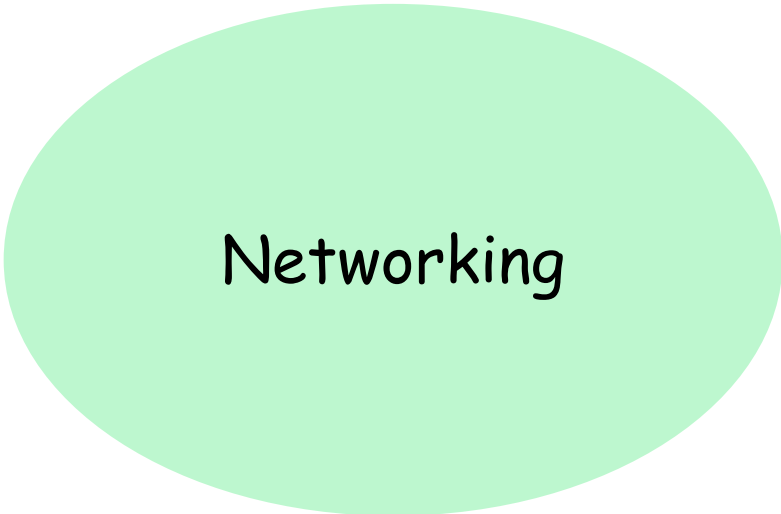
Information transfer  
(typically "raw" info)



## Distributed Algorithms

Simple network models

Emphasis on  
order complexity

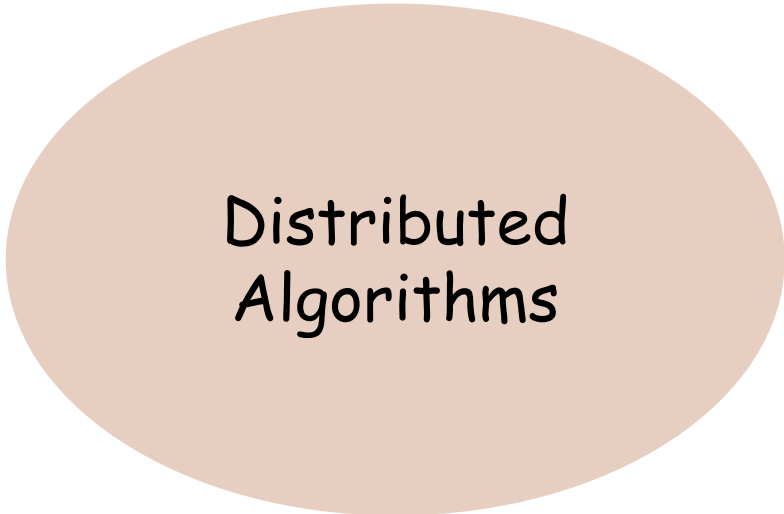


## Networking

"Accurate" network models

Constants matter

Information transfer  
(typically "raw" info)



## Distributed Algorithms

Simple network models

Emphasis on  
order complexity

Computation  
affects communication

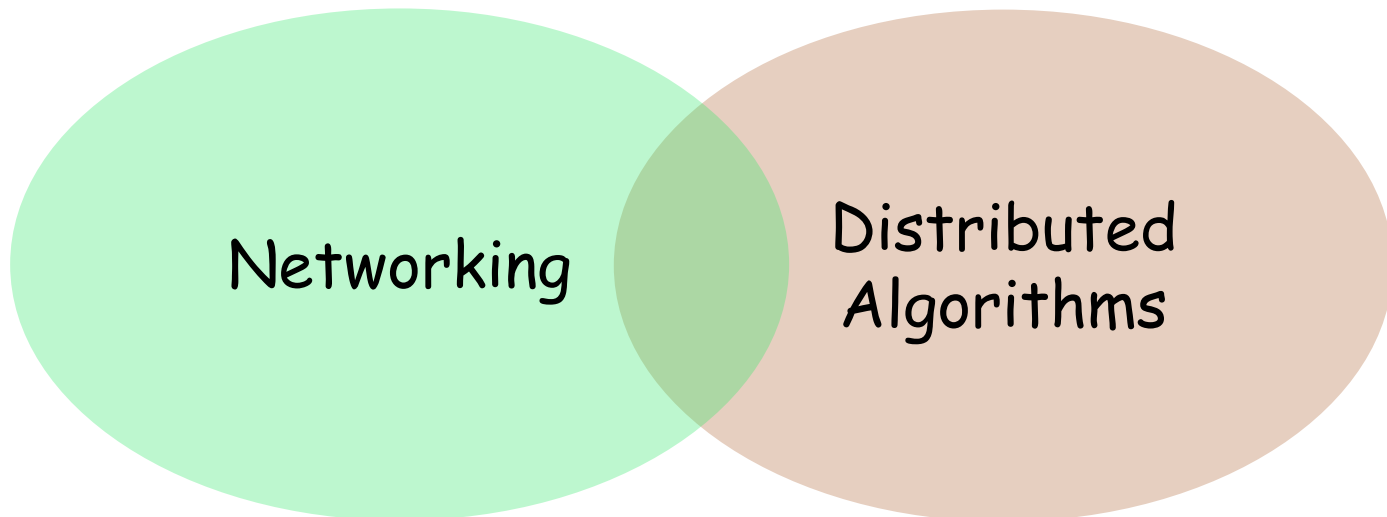
# Popular Network Models

- Point-to-point graphs
- Broadcast channel
- Unit disk graph (wireless broadcast)
- SINR threshold model (wireless interference)



# Unsurprising Insight

- “Accurate” network models  
can lead to more interesting problems



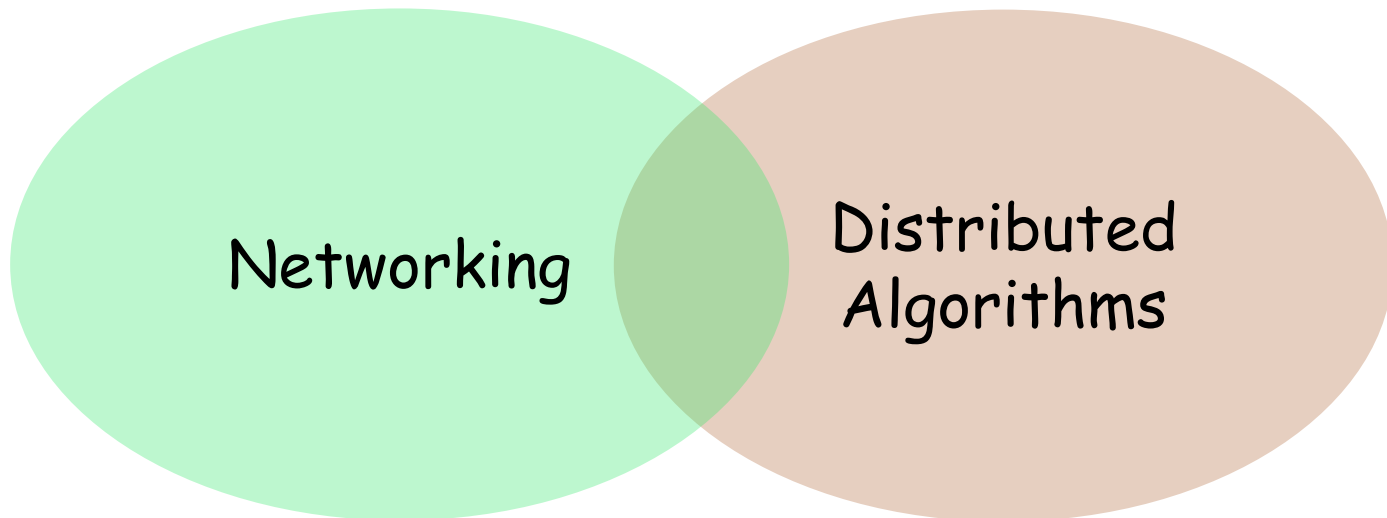
All models are wrong;  
some models are useful.

AND  
SOME ARE  
JUST CUTE

-- George Box

# This talk

- Example ... consensus



# Consensus

- Multiple parties / agents / nodes
  - Initial input at one or more nodes
- All nodes **agree** in the end
- Some notion of **validity** for agreed value

# Consensus ... Dictionary Definition

- Majority of opinion
- General agreement

# Consensus



# Consensus



Validity: Decide on



# Consensus



Validity: Decide on ??

Majority rule



# Consensus



**Validity:** Decide on



**Majority rule**

# Consensus



Validity: Decide on ??

Average consensus

# Consensus

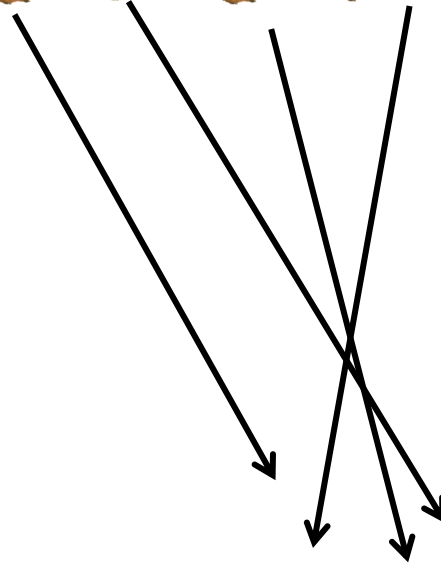
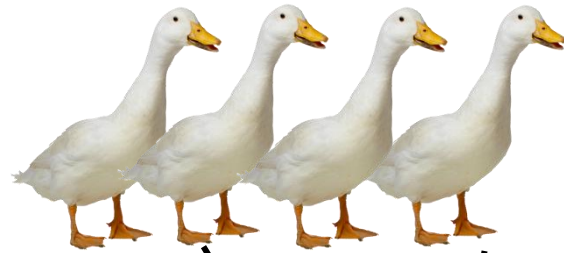


**Validity:** Decide on

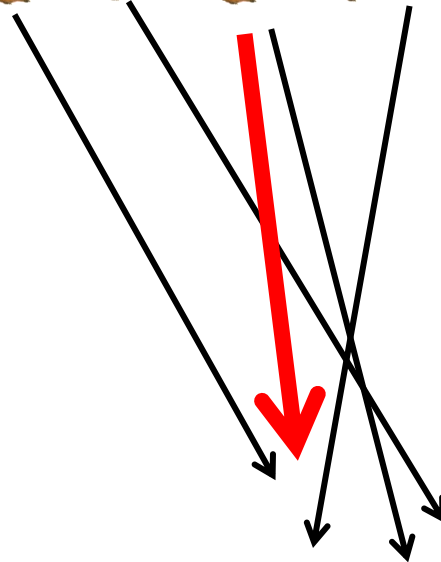
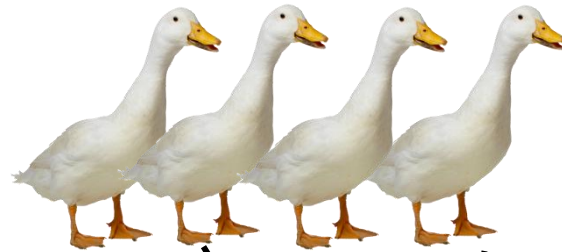


Average consensus

# Flock of Birds (or Robots)



# Flock of Birds (or Robots)



Average consensus



## Many Faces of Consensus

- All nodes have non-null input / only a subset do
- No failures / failures allowed (node/link)
- Synchronous/asynchronous
- Deterministically correct / probabilistically correct
- Exact agreement / approximate agreement
- Global communication / local communication

# Consensus in Practice

- Fault-tolerant file systems
- Fault-tolerant servers
- Distributed control
- Social networks

# This talk

- Byzantine broadcast
- Average consensus



# This talk

- Byzantine broadcast
- Average consensus



# Byzantine Broadcast

# Client-Server Model

Client



**command A**



Server



**state 0**

# Client-Server Model

Client



**command A**

**response (A)**

Server



**state A**

# Client-Server Model

Client



**command A**

**%\*\$&#**

Server



# Replicated Servers

Client



**command A**

**A**

**A**

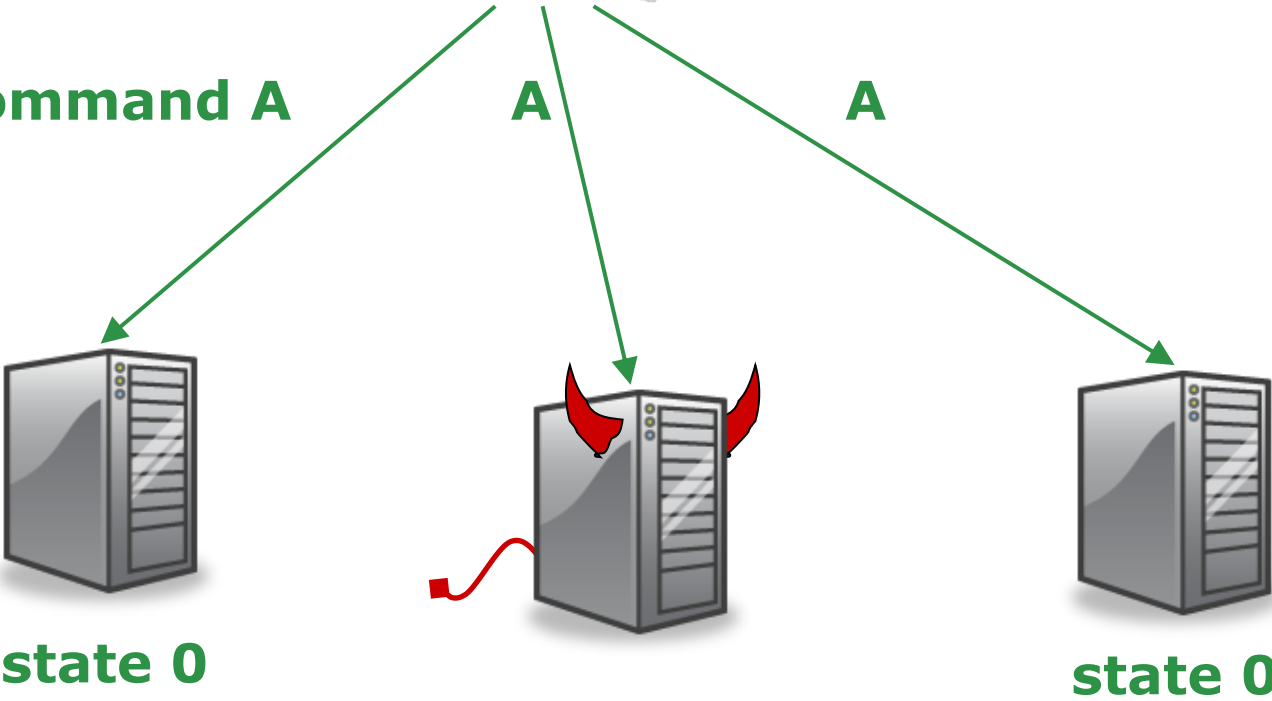
Servers



**state 0**



**state 0**



# Replicated Servers

Client



response (A)

response (A)

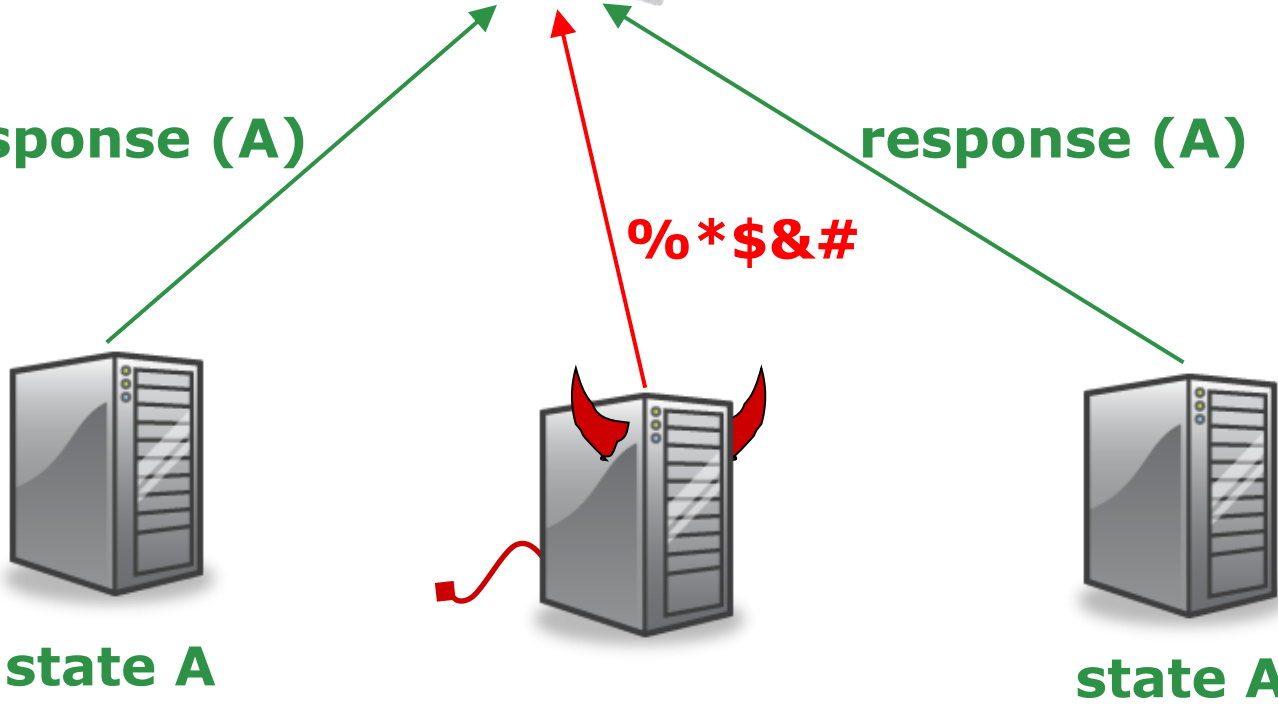
%\*\$&#

Servers



state A

state A



# Replicated Servers

Client



**command A**

**command C**

**command B**



**state A**

**state B**

**state C**

Servers



# Replicated Servers

Client



**command A**

**command B**

**command C**

Servers



Replication fails

# Replicated Servers

Client



**command A**

**command B**

**command C**

Servers



Must agree on same request

# Byzantine Broadcast

- Source node  $S$  broadcasts to others
- $n - 1$  other nodes

# Byzantine Broadcast

Source  $S$  an input (command)

- Fault-free nodes agree on identical value
- $S$  fault-free  $\rightarrow$  agree on its input
- Up to  $f$  Byzantine node failures

# Byzantine Fault Model

- **Nodes** may fail
  - Arbitrarily bad behavior
    - Packet tampering
    - Packet dropping
- ... anything goes

## Many Faces of Consensus

- All nodes have non-null input / only a subset do
- No failures / failures allowed (node/link)
- Synchronous/asynchronous
- Deterministically correct / probabilistically correct
- Exact agreement / approximate agreement
- Global communication / local communication

# Byzantine Broadcast

Example algorithm

[Lamport, Shostak, Pease 1982]

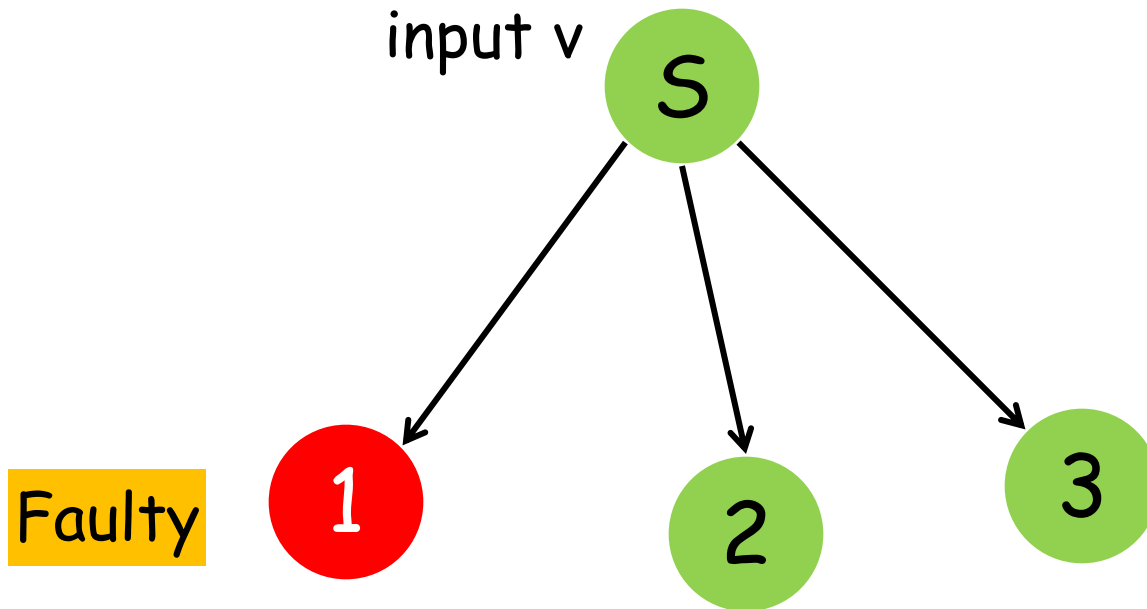
- 4 nodes
- At most 1 faulty node

$$n = 4$$

$$f = 1$$

# Byzantine Broadcast

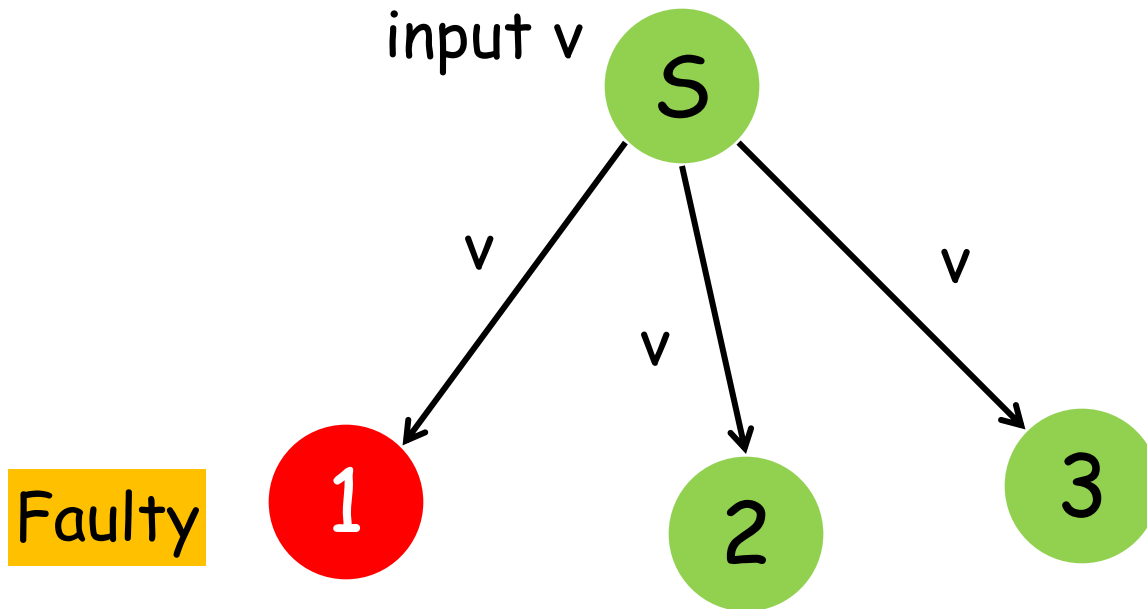
$n = 4$



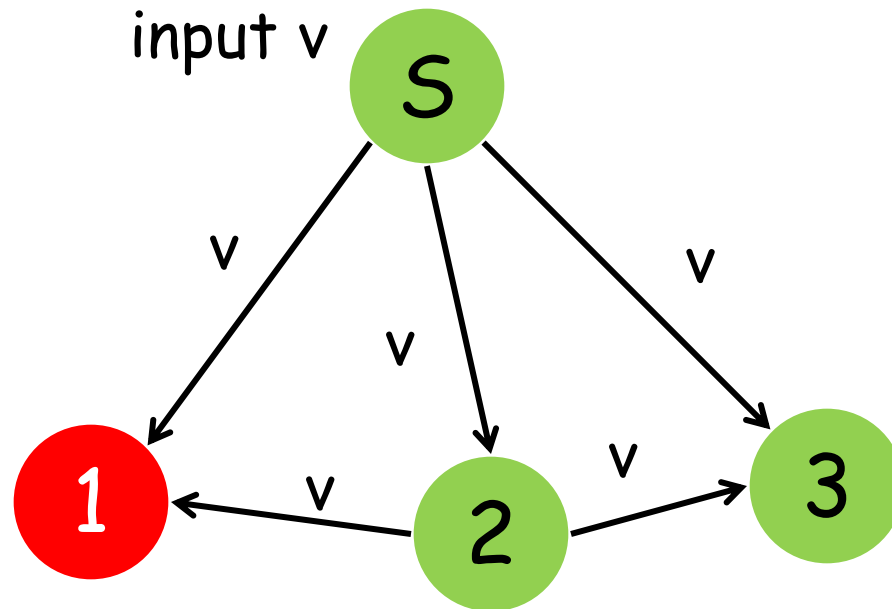


# Byzantine Broadcast

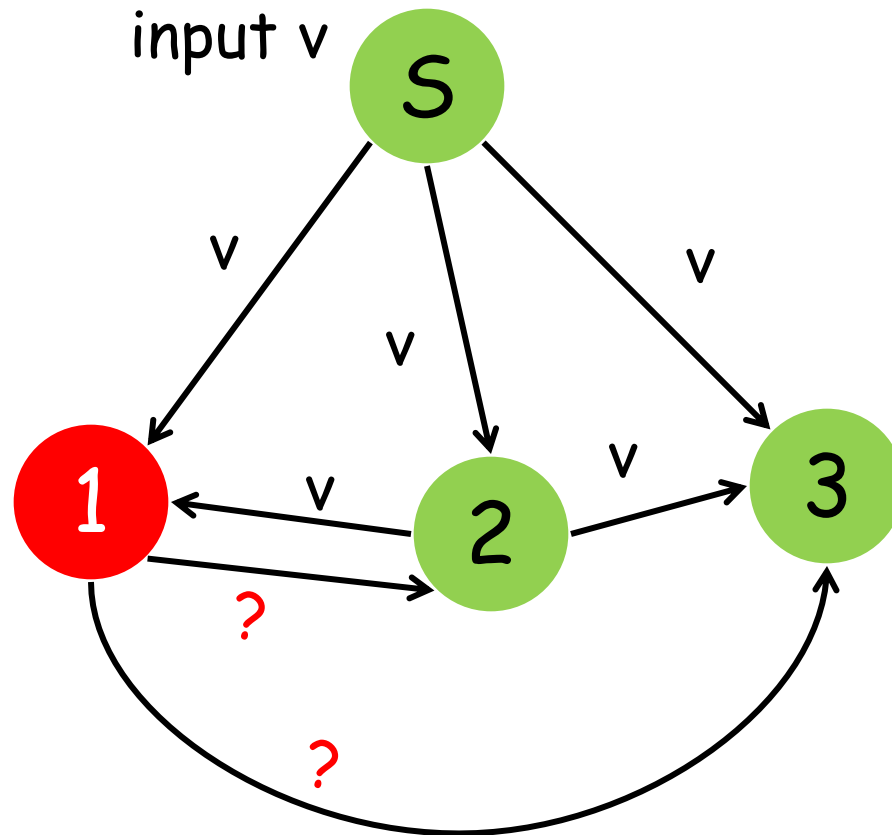
$n = 4$



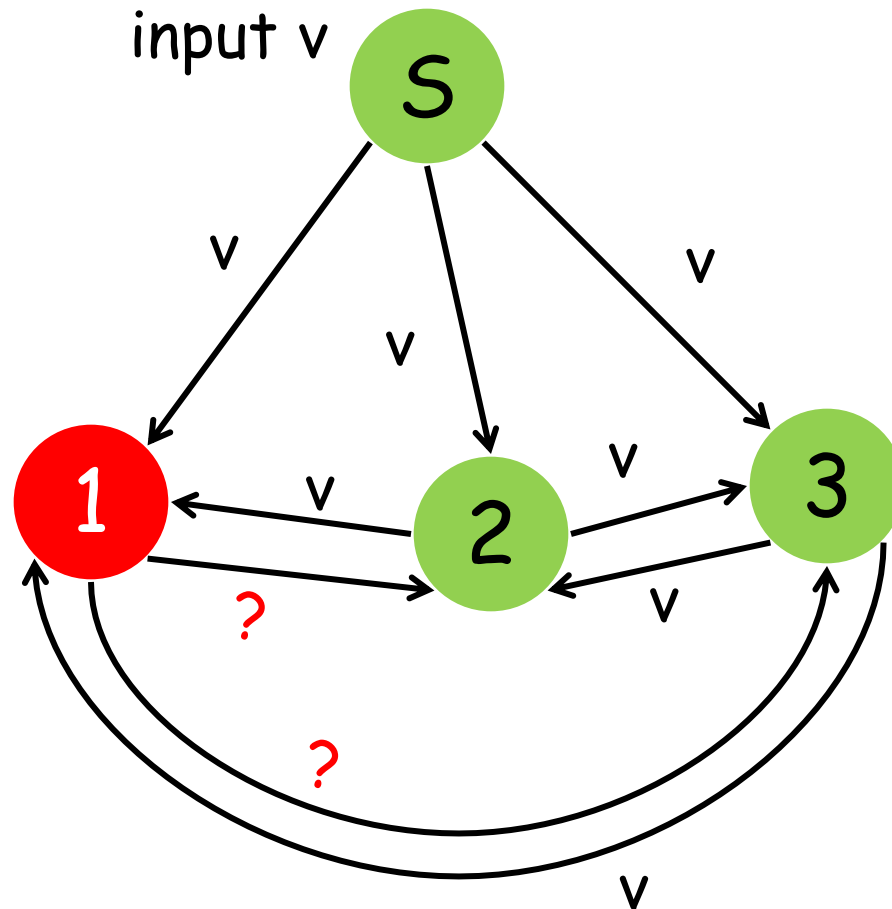
# Broadcast



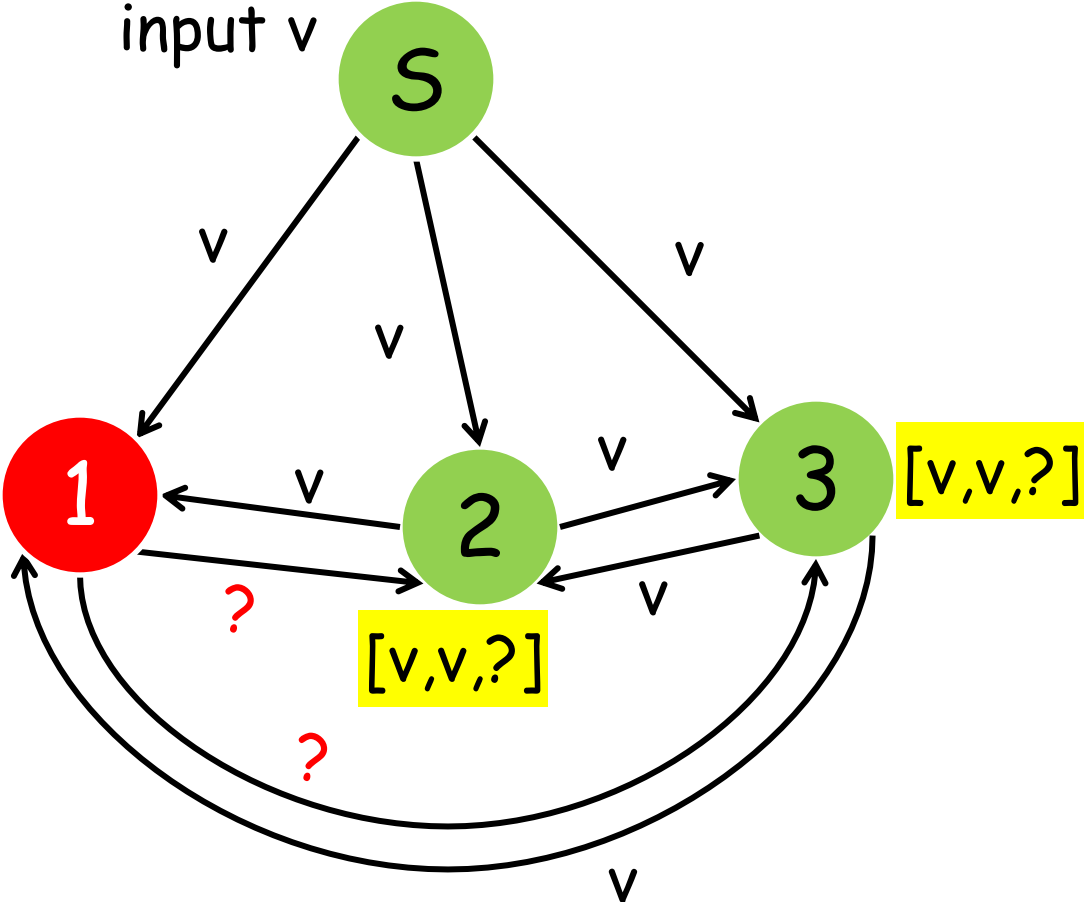
# Broadcast



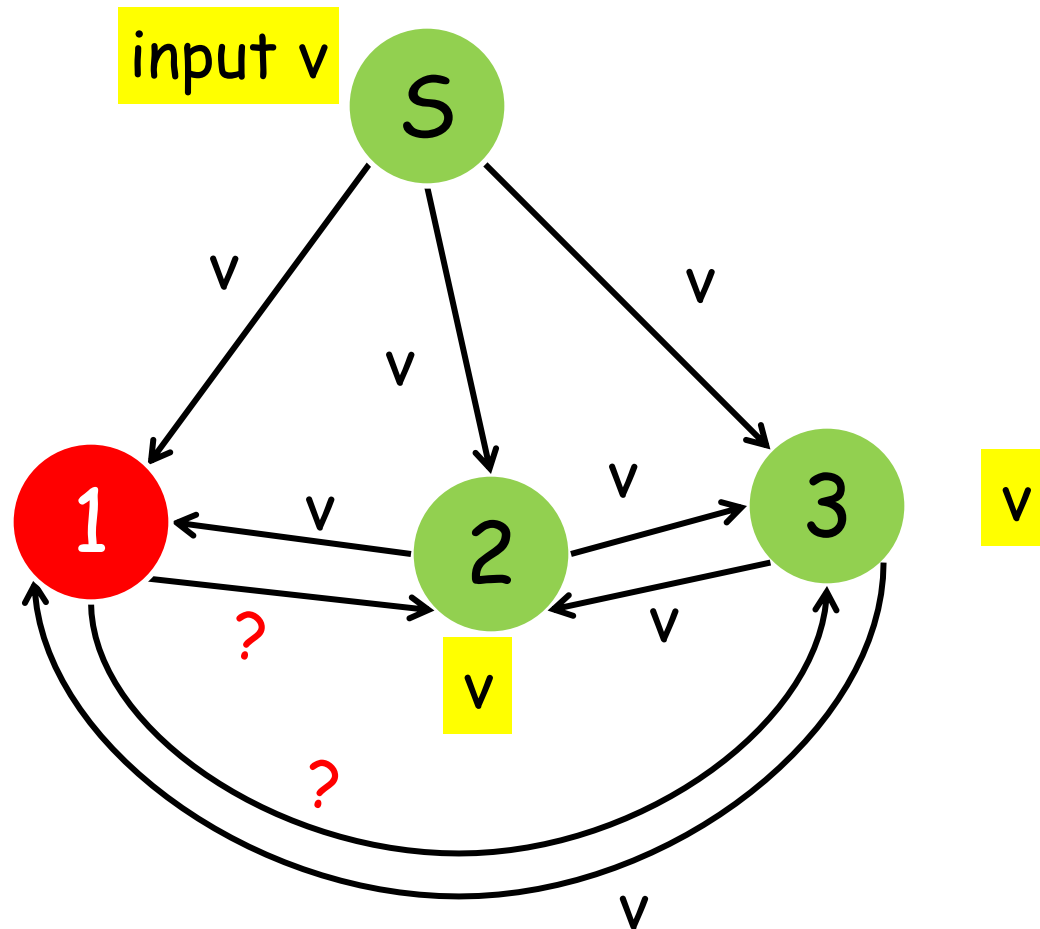
# Broadcast



# Broadcast

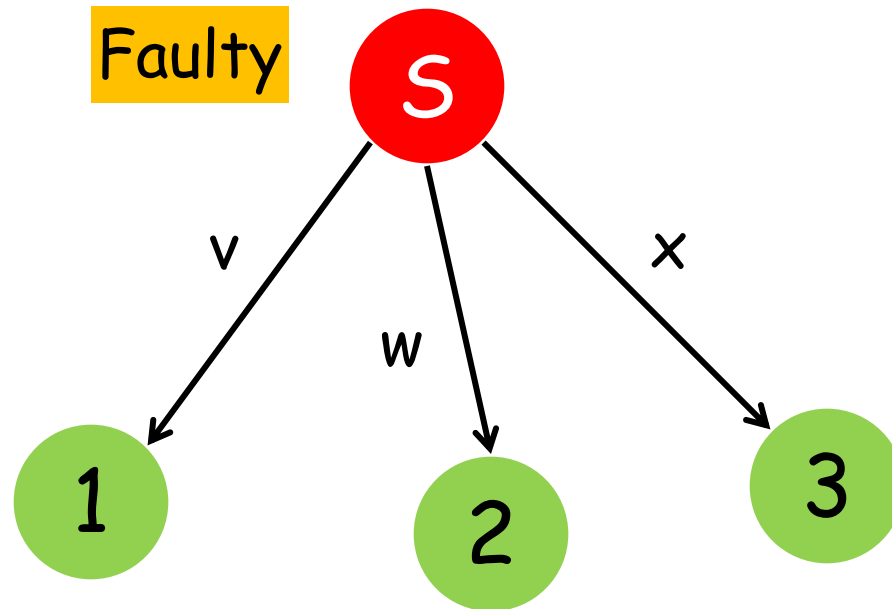


# Broadcast



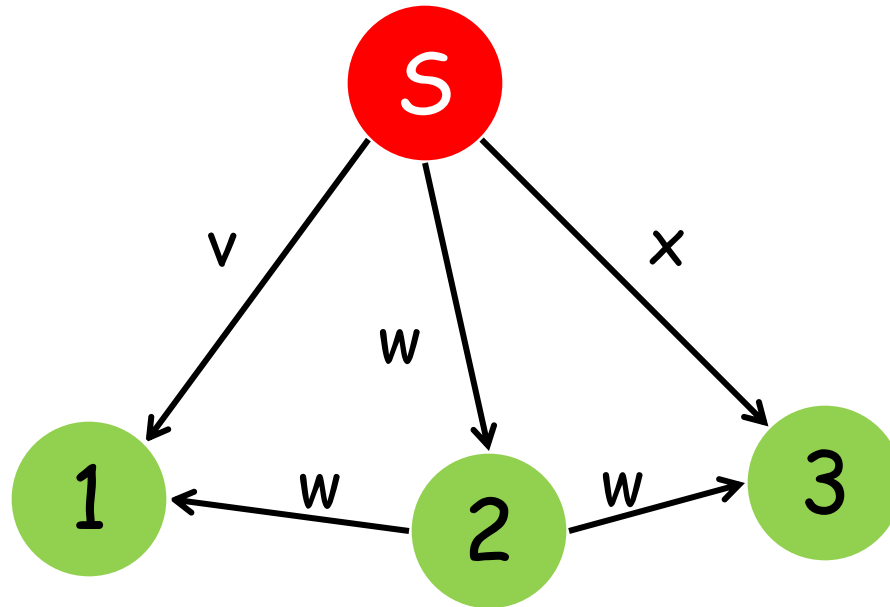
Majority vote  
→ Correct

# Broadcast



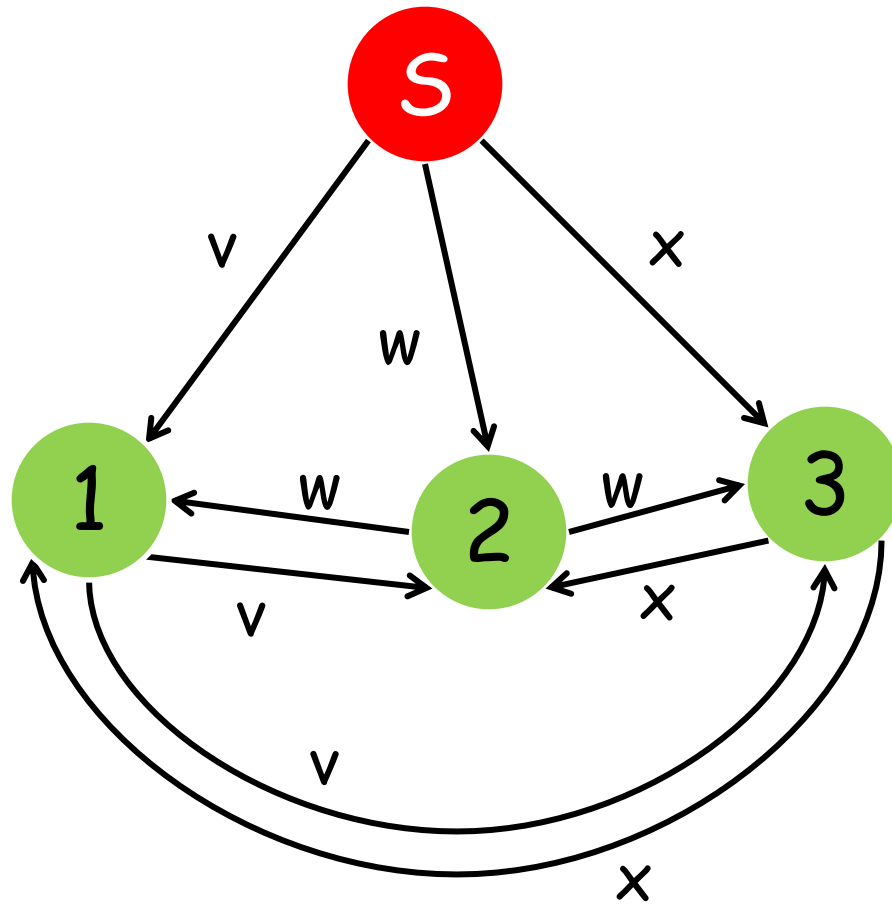
Bad source may attempt to diverge state at good nodes

# Broadcast

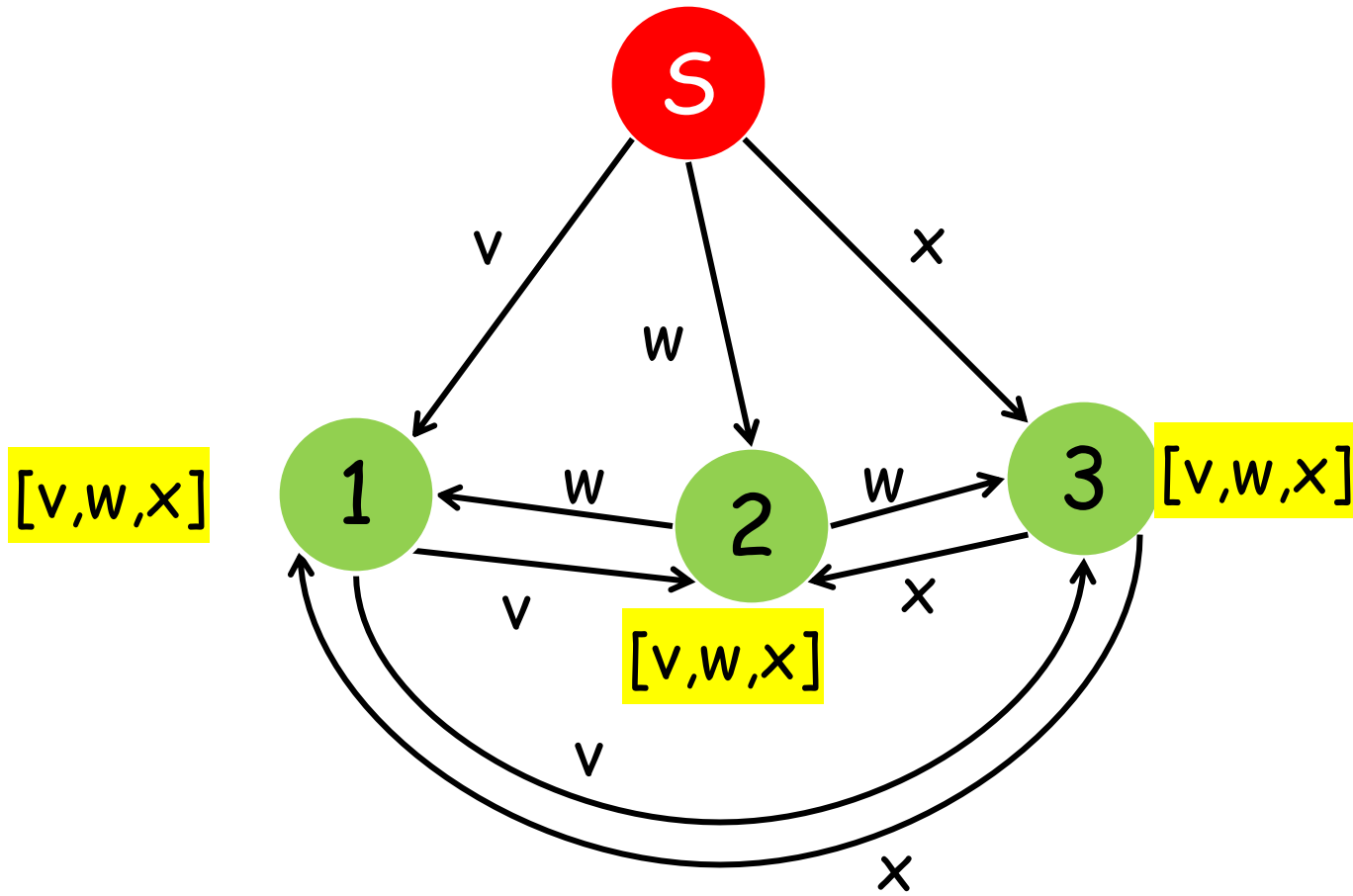




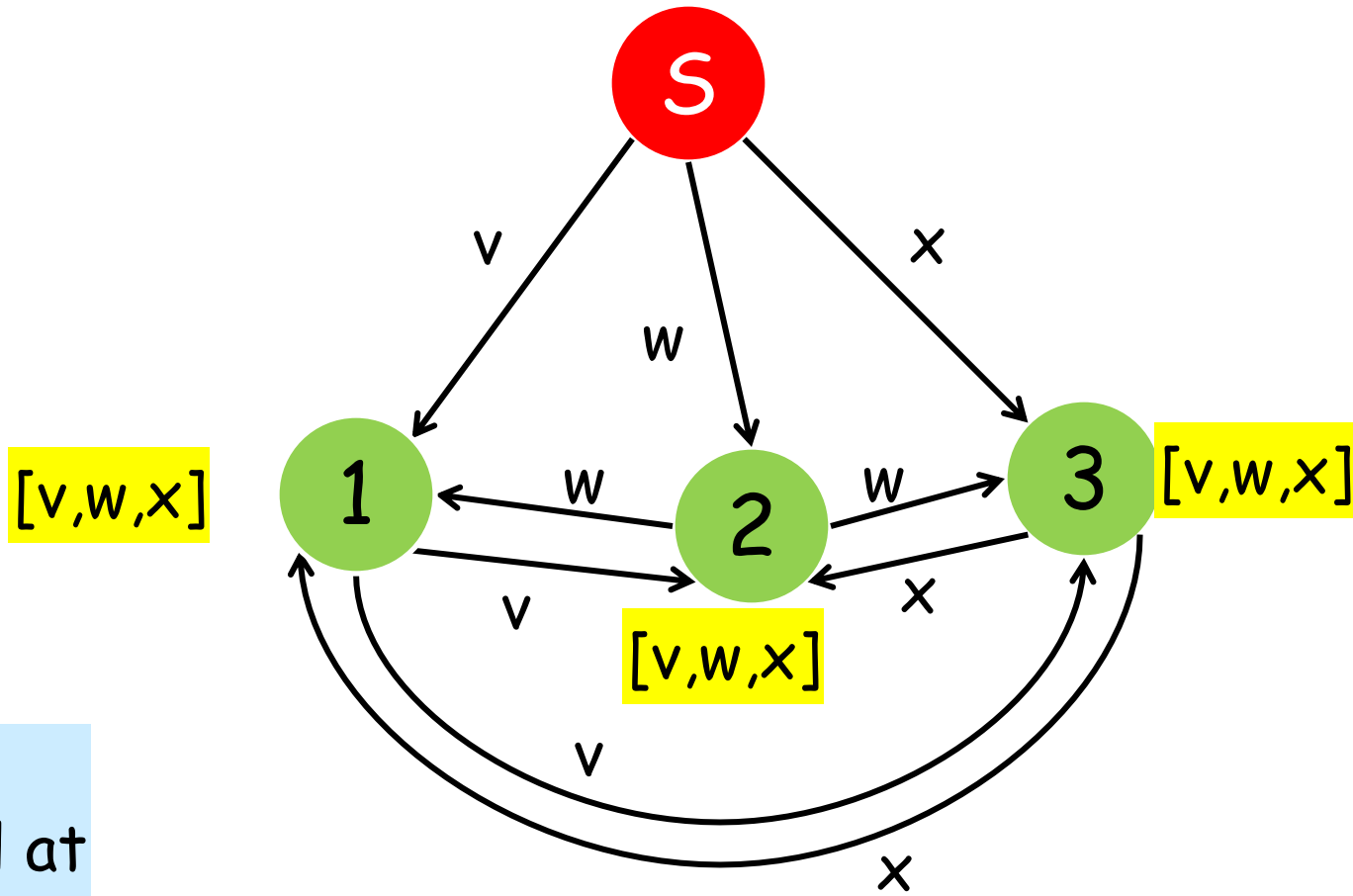
# Broadcast



# Broadcast



# Broadcast

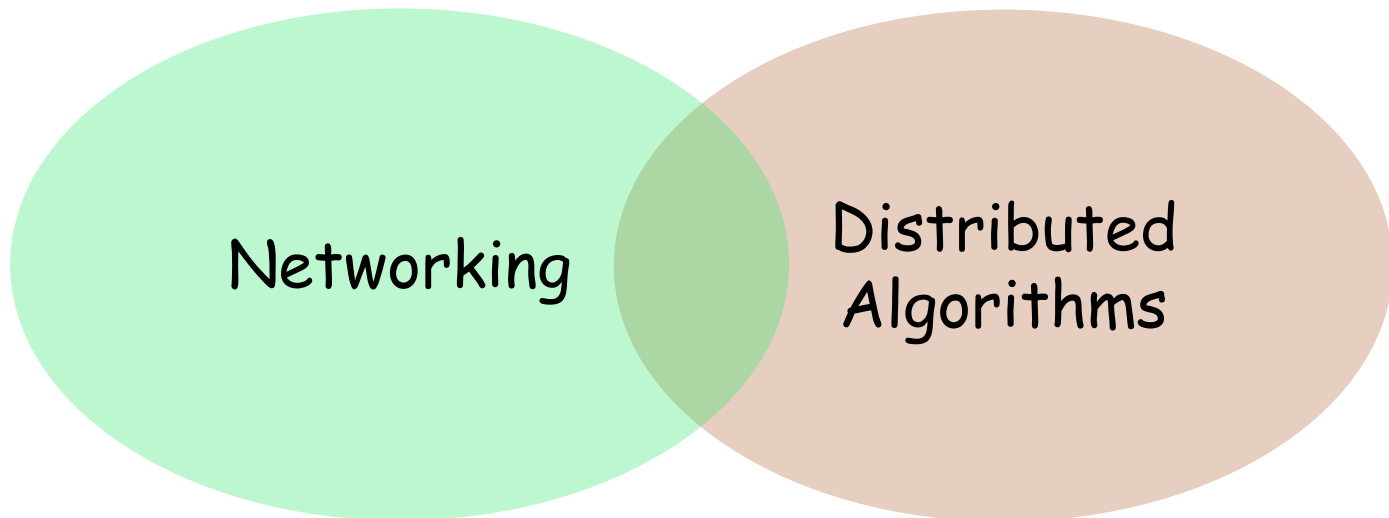


Vote  
identical at  
good nodes

# Known Bounds

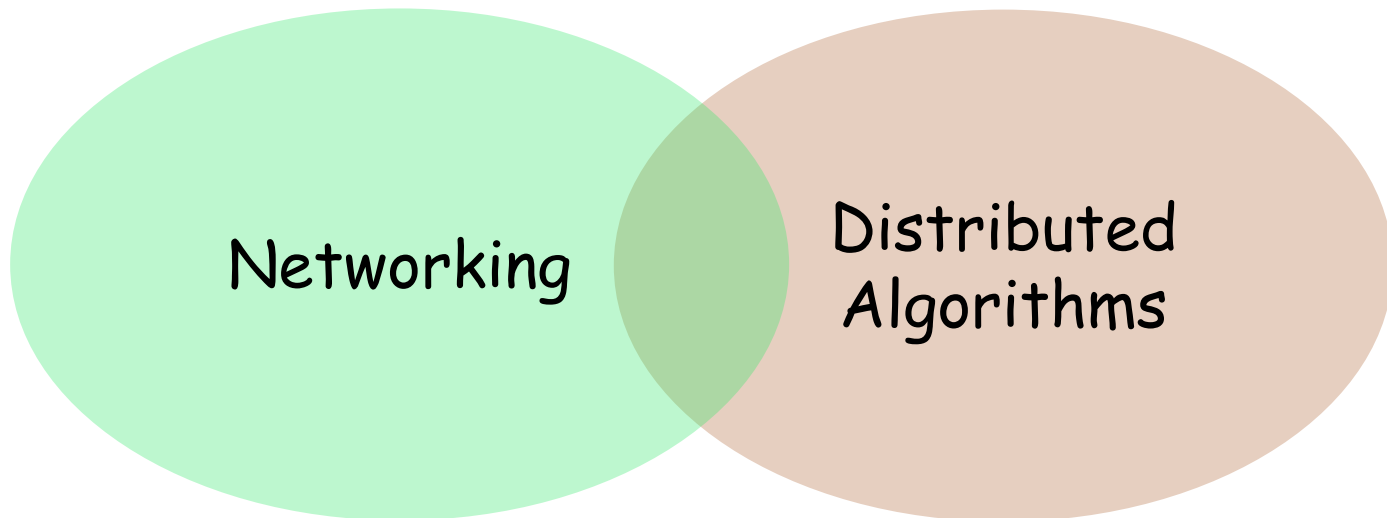
- $n \geq 3f + 1$  nodes to tolerate  $f$  failures
- Connectivity  $\geq 2f + 1$
- $\Omega(n^2)$  messages in worst case
- $f+1$  rounds of communication

# Impact of Network



# Impact of Network

- How to quantify the impact ?



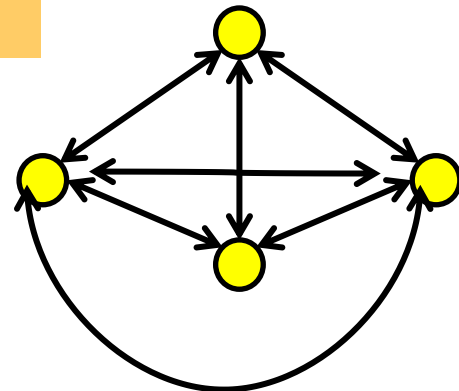
# Metric 1: Communication Cost per Bit

$$\frac{\text{Total communication cost (in bits)}}{\text{Number of bits of Byzantine broadcast}}$$

# Metric 1: Communication Cost per Bit

$$\frac{\text{Total communication cost (in bits)}}{\text{Number of bits of Byzantine broadcast}}$$

Ignores network characteristics





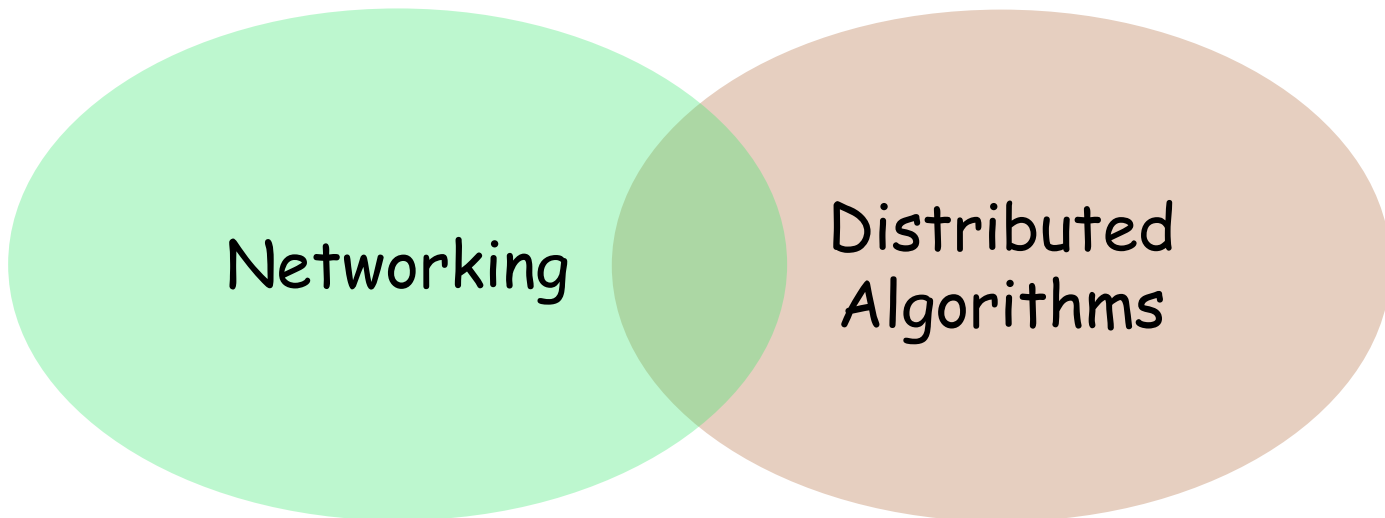
## Metric 2: Throughput

- Borrow notion of **throughput** from networking
- $b(t)$  = number of bits agreed upon in  $[0,t]$

$$\textit{Throughput} = \lim_{t \rightarrow \infty} \frac{b(t)}{t}$$

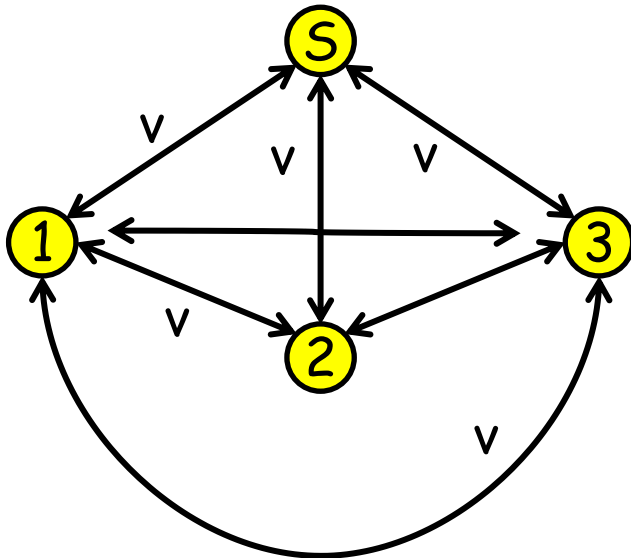
# Impact of Network

- How does the network affect Byzantine broadcast/consensus ?



## Consider earlier algorithm ...

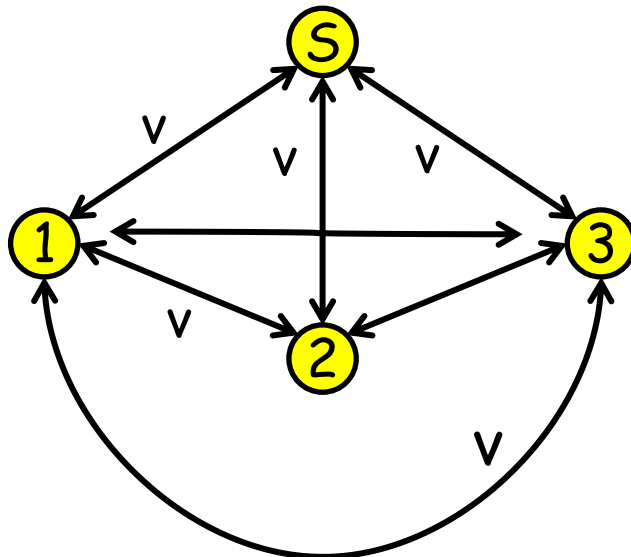
- All data sent on each link once  
→ broadcast throughput 10



Each directed link rate = 10

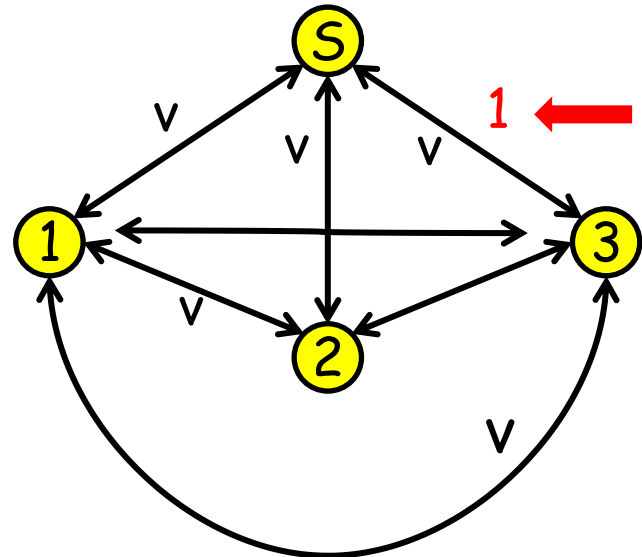
# Example

■ broadcast throughput 10



Each directed link rate = 10

■ broadcast throughput 1

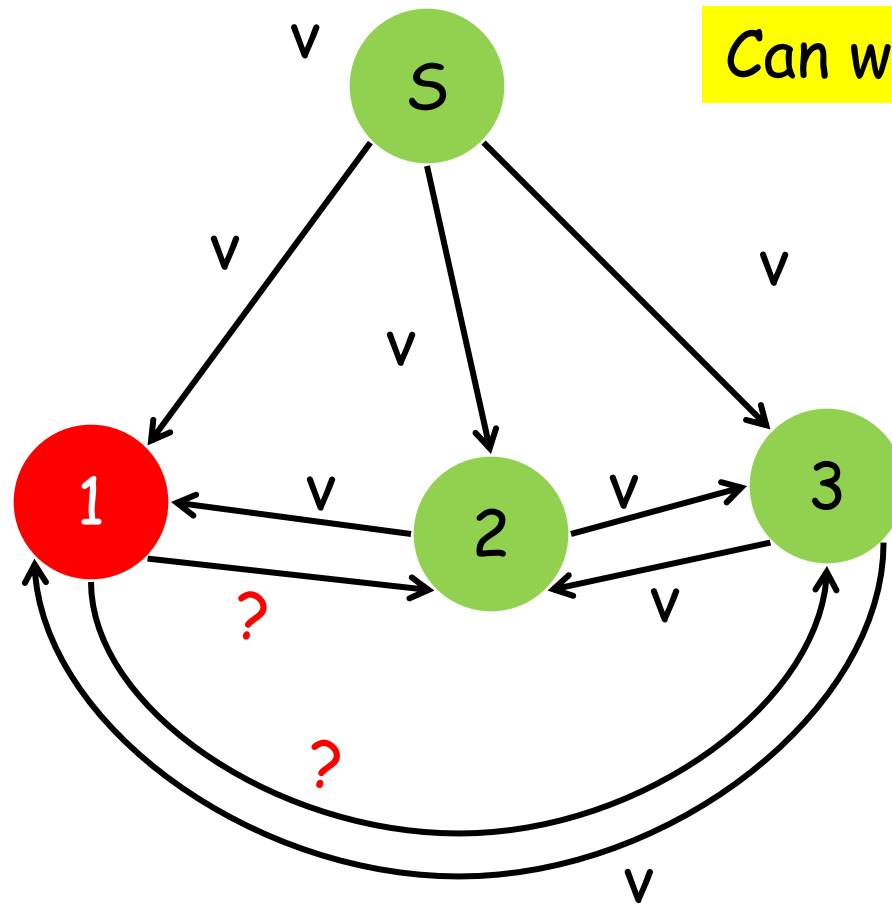


All other links = 10

# Point-to-Point Networks

- How to best exploit available link capacity ?
  - Symmetric case
  - Asymmetric case

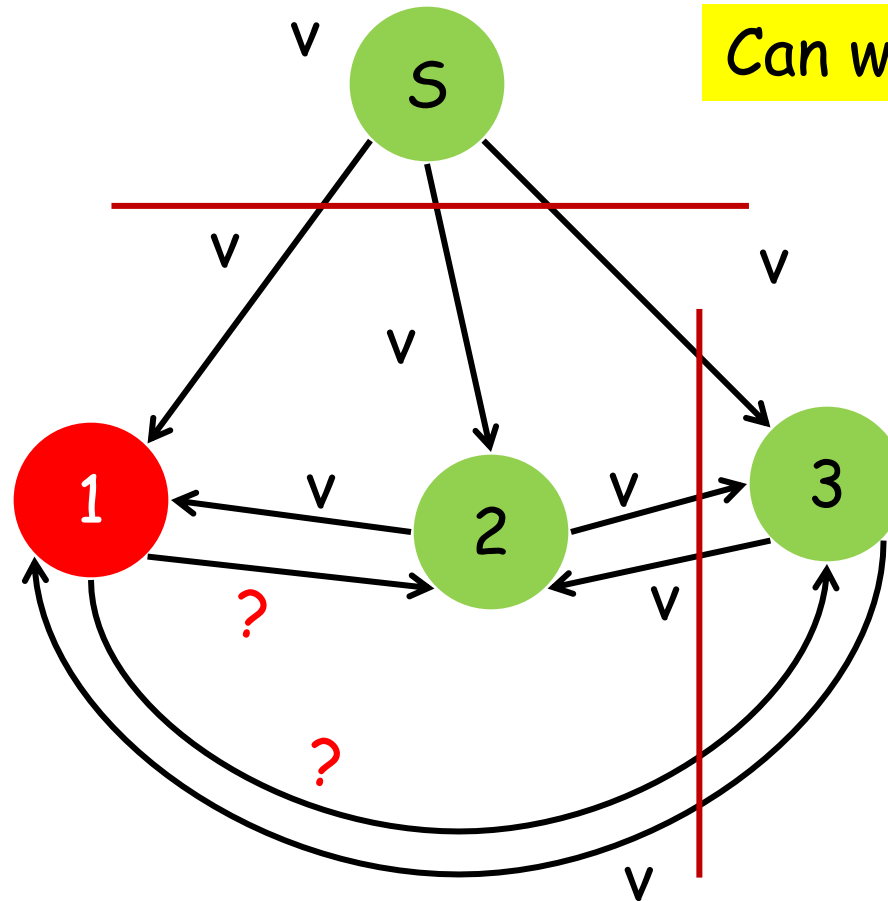
# Symmetric Case



Can we do better ?

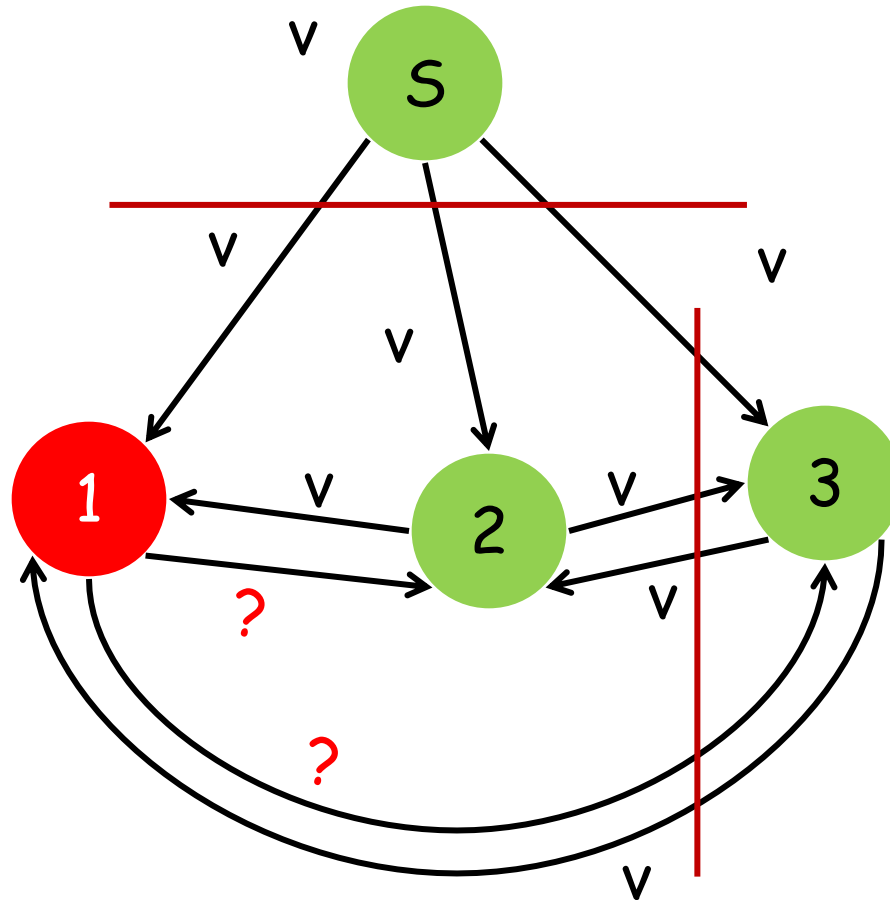
# Symmetric Case

- "Replication" code



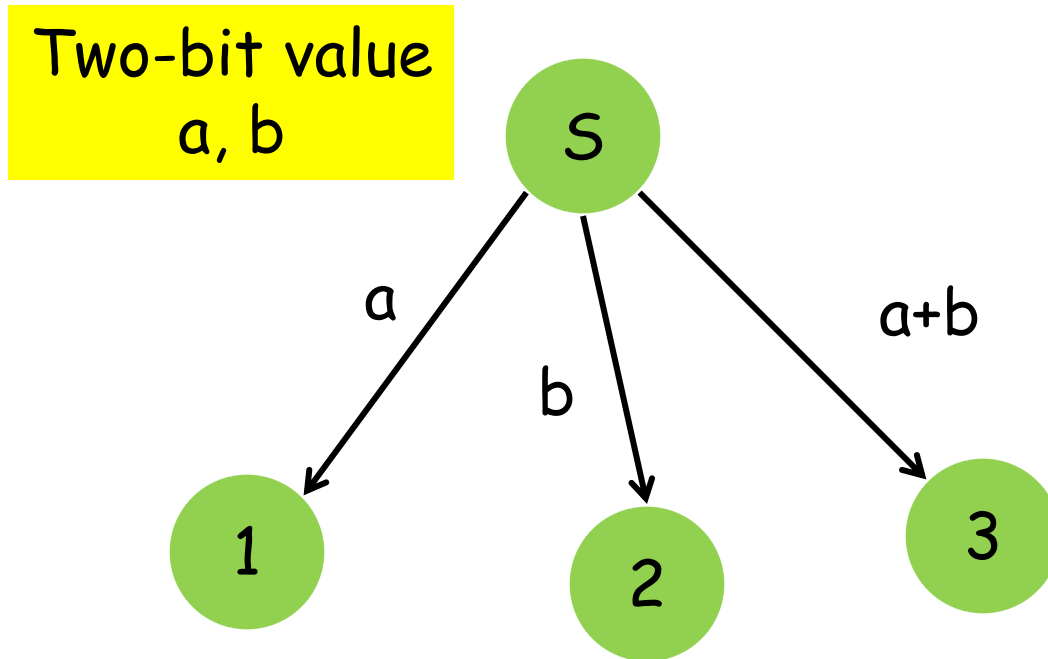
# Can we do better ?

- More efficient code ... standard tool in Communication

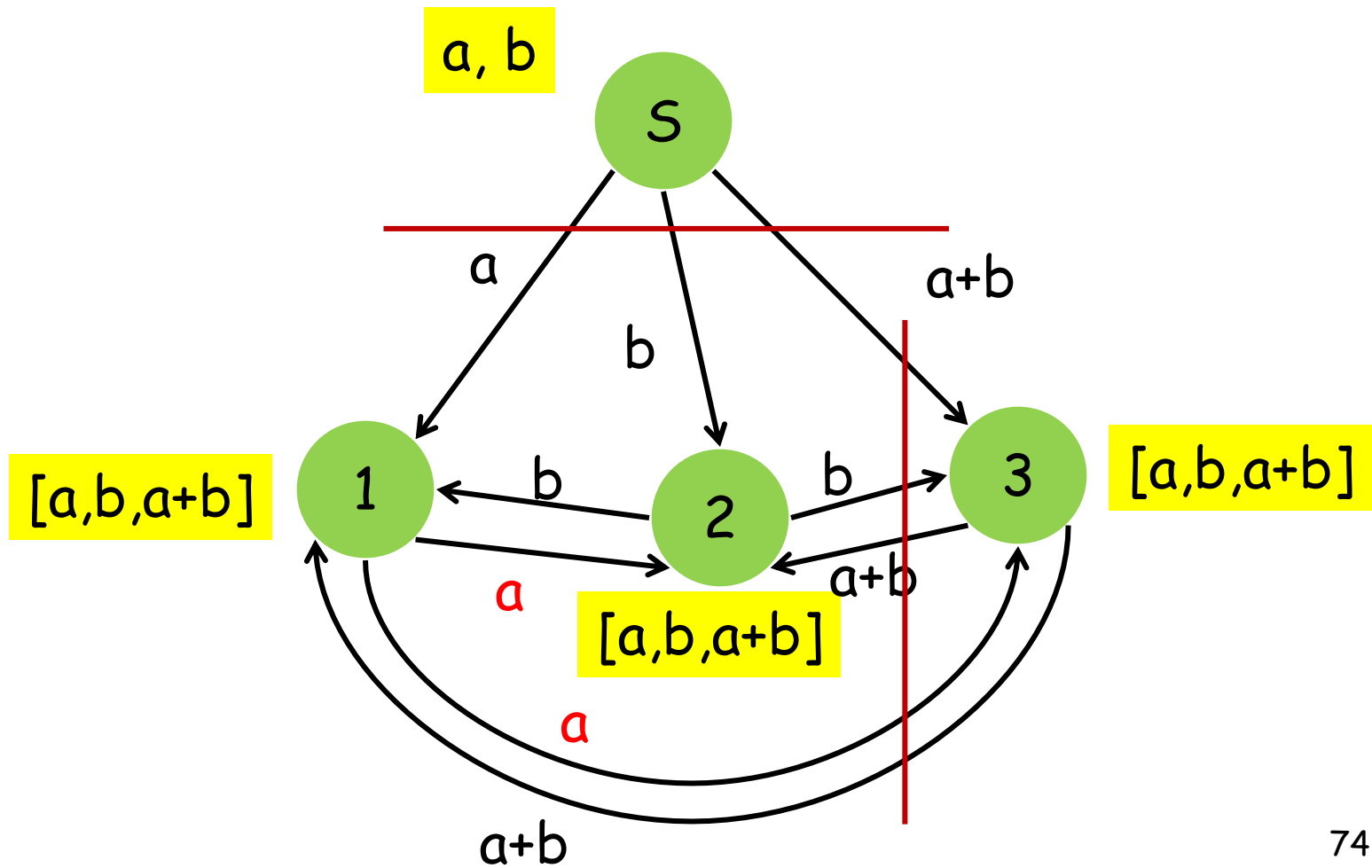




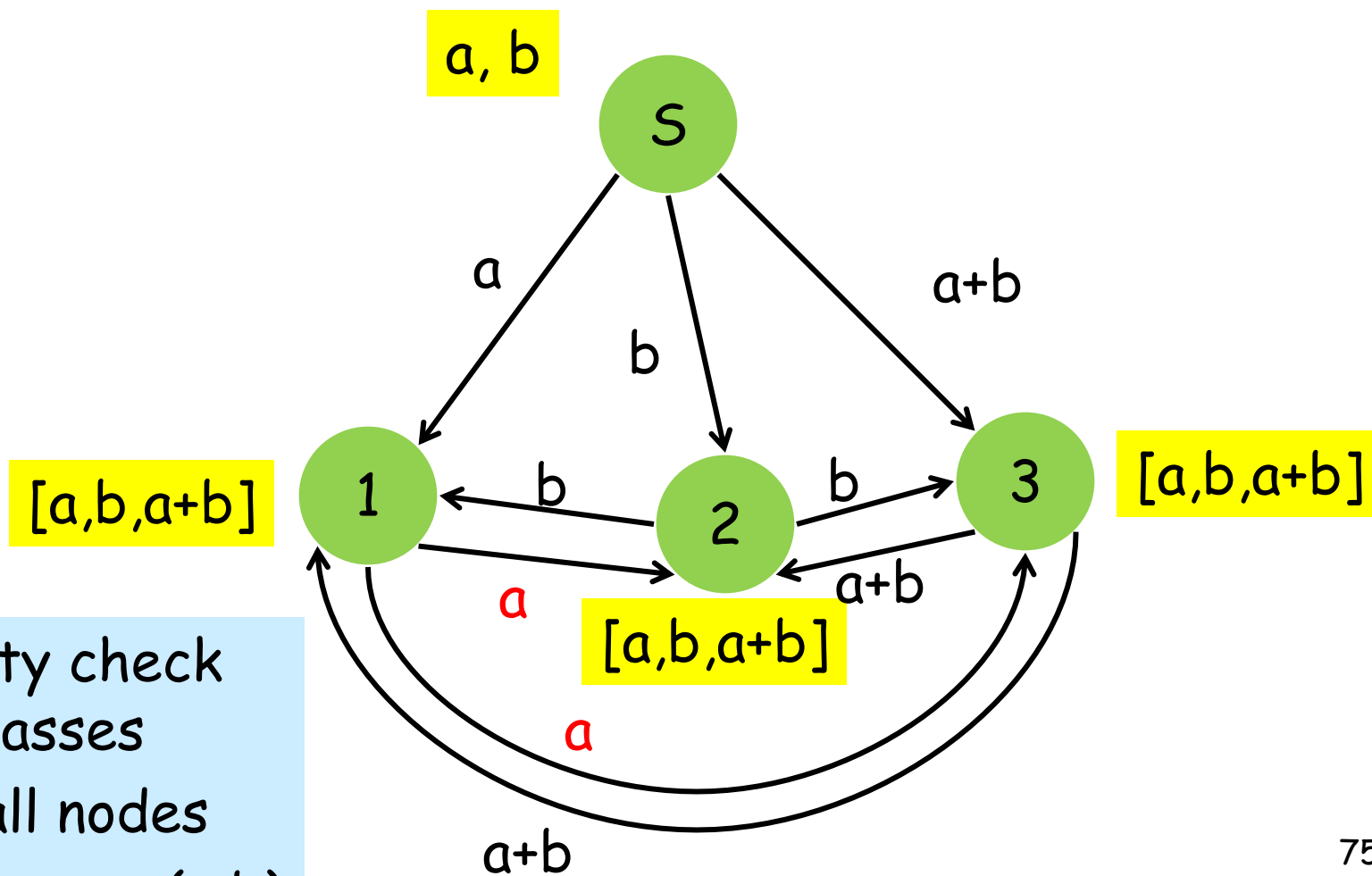
# Make Common Case Fast



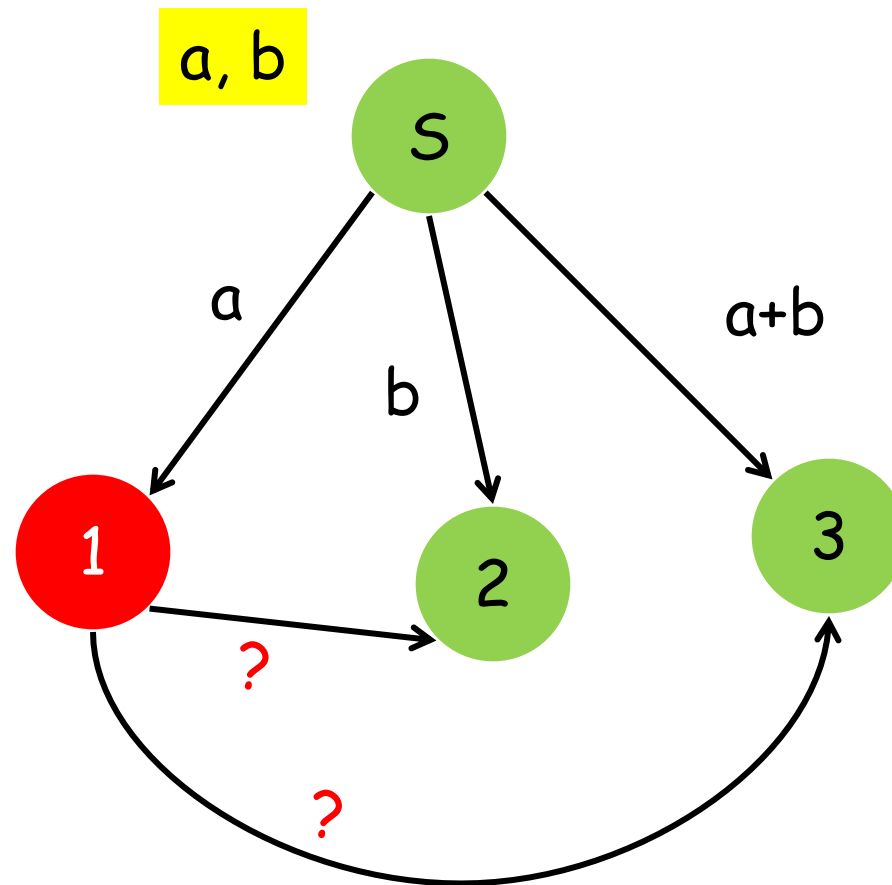
# Make Common Case Fast



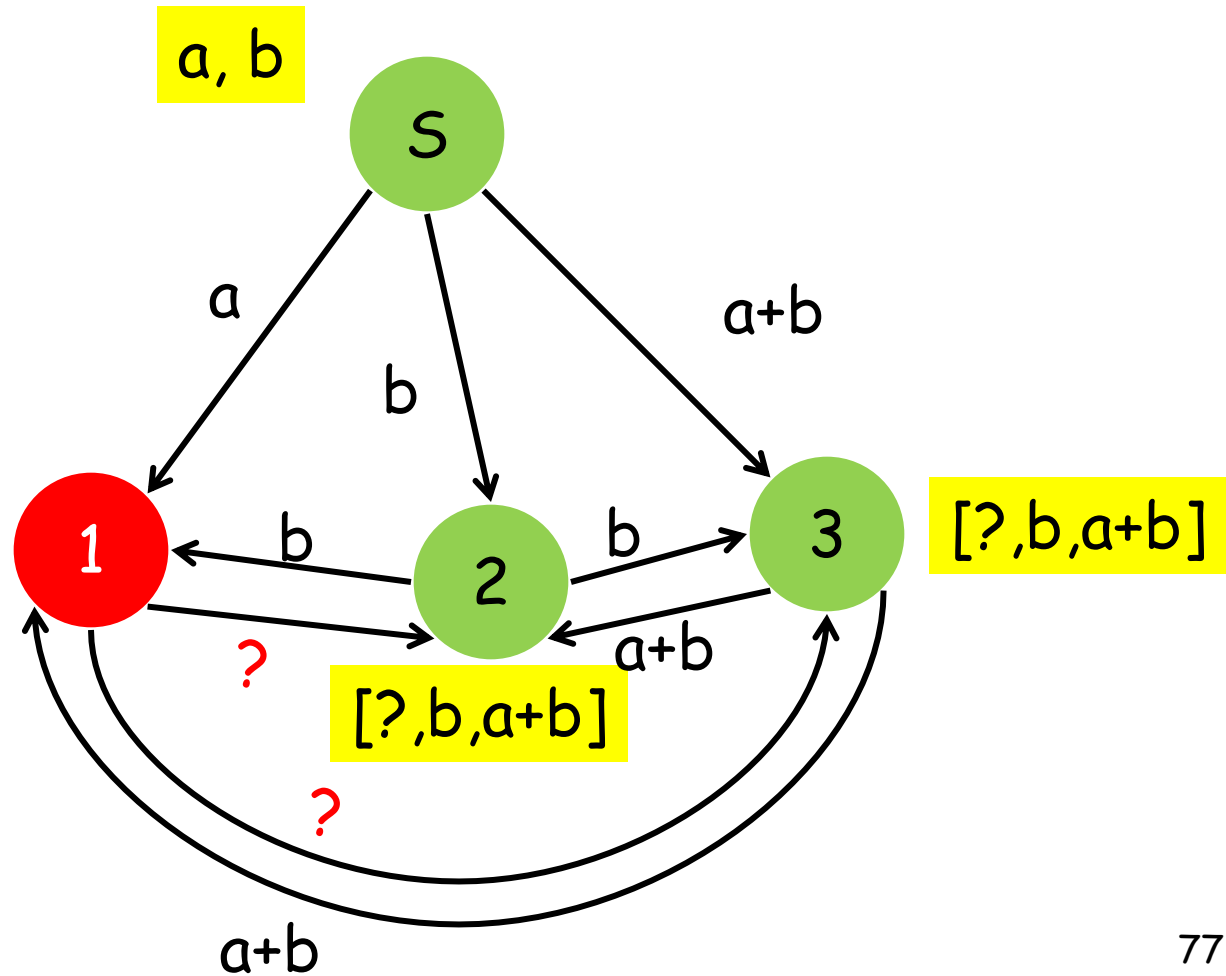
# Make Common Case Fast



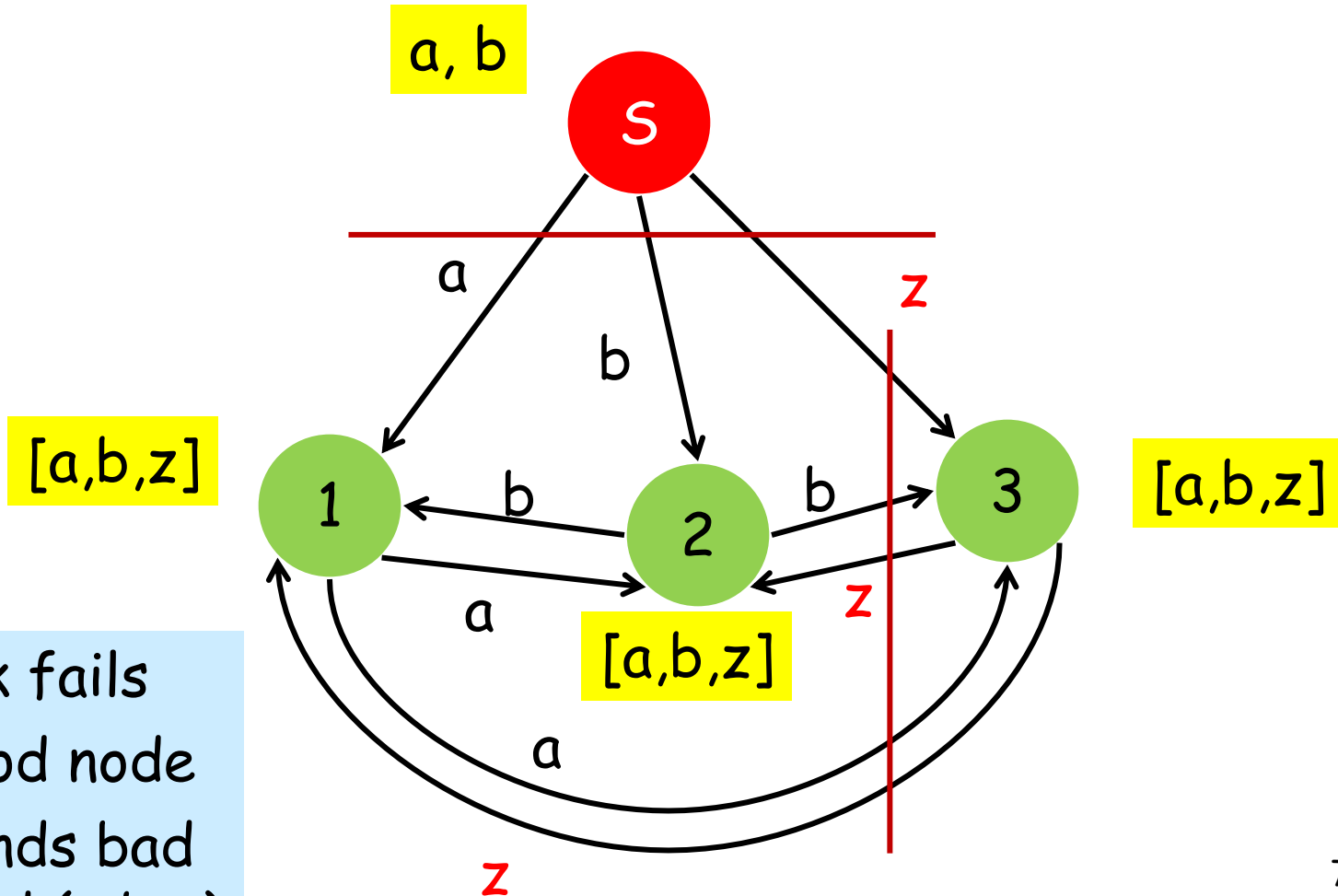
# Make Common Case Fast



# Make Common Case Fast



# Make Common Case Fast



# After Failure Detection

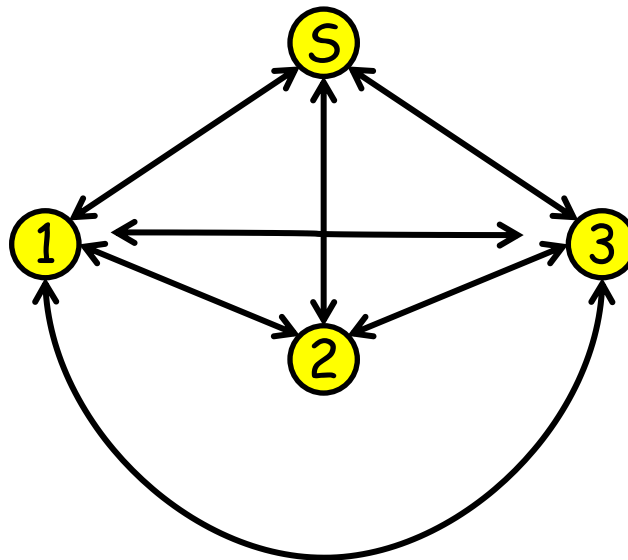
- More work required after failure detection
- But not too many times

# Symmetric Case

- Per link capacity  $R$

→ Byzantine broadcast rate  $(n-1-f)R$

Optimal



$n = 4$   
 $f = 1$   
→  $2R$



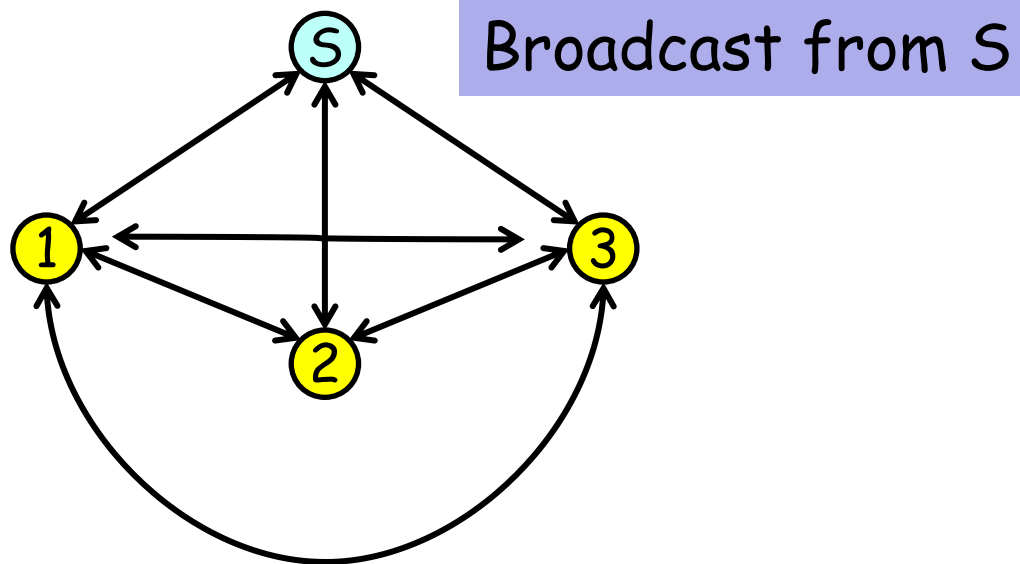
# Arbitrary Networks

Optimal Byzantine Broadcast algorithm **unknown**

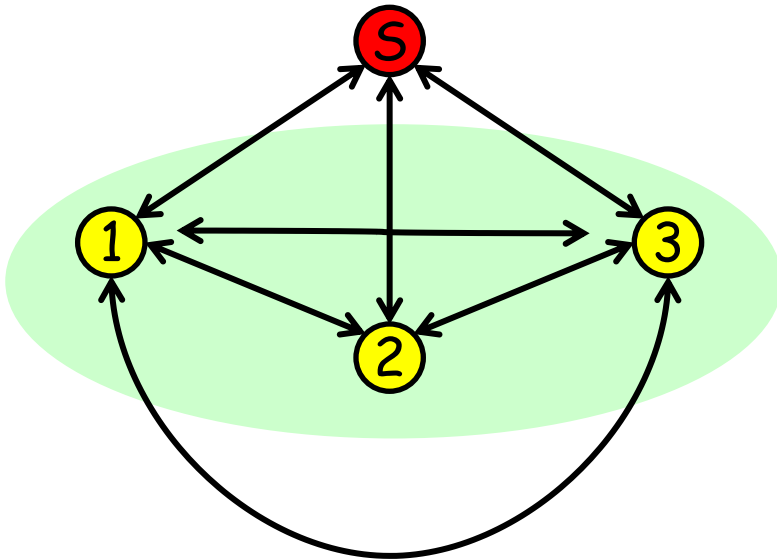
→ Throughput within constant factor

# Algorithm Sketch

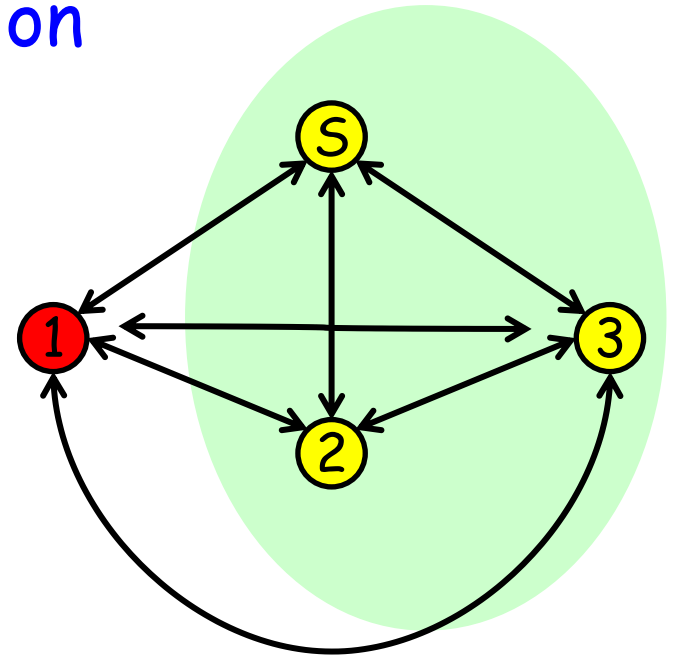
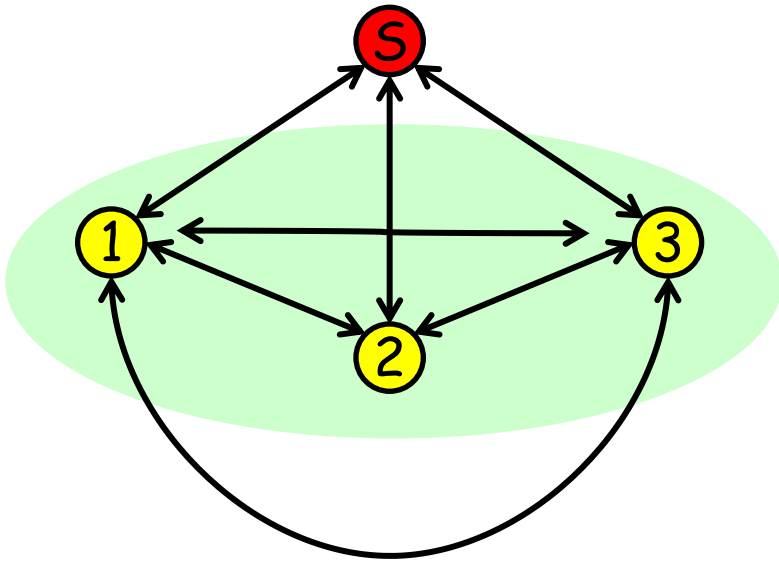
- Broadcast data without fault tolerance



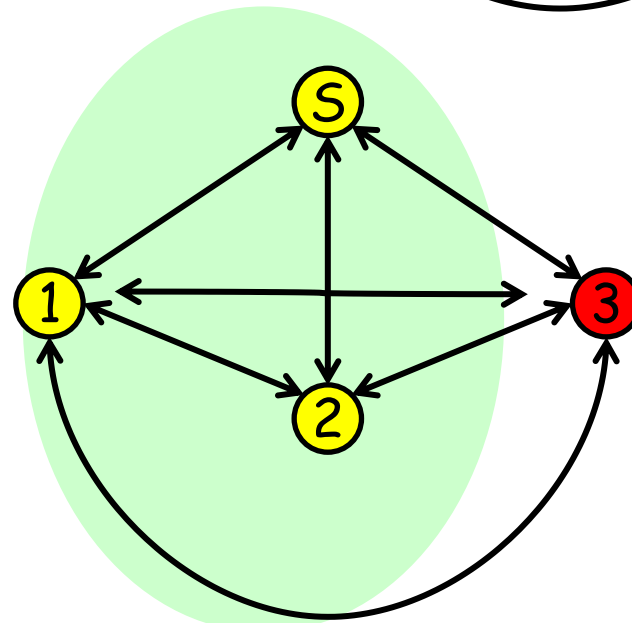
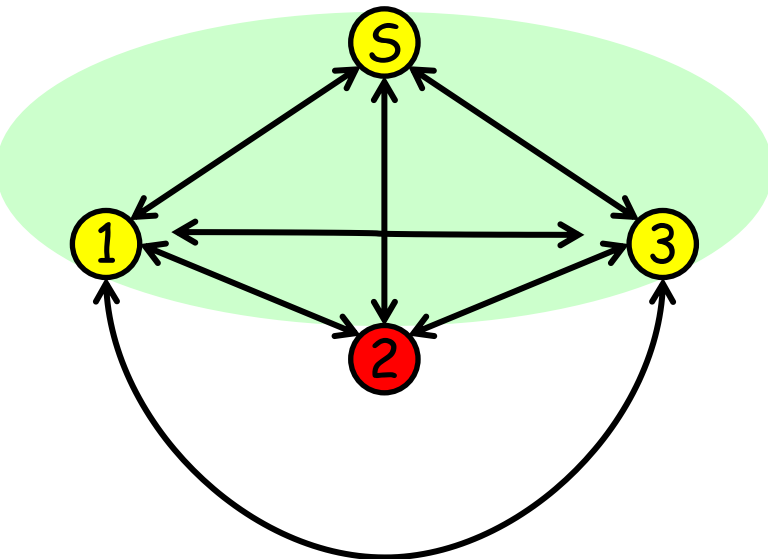
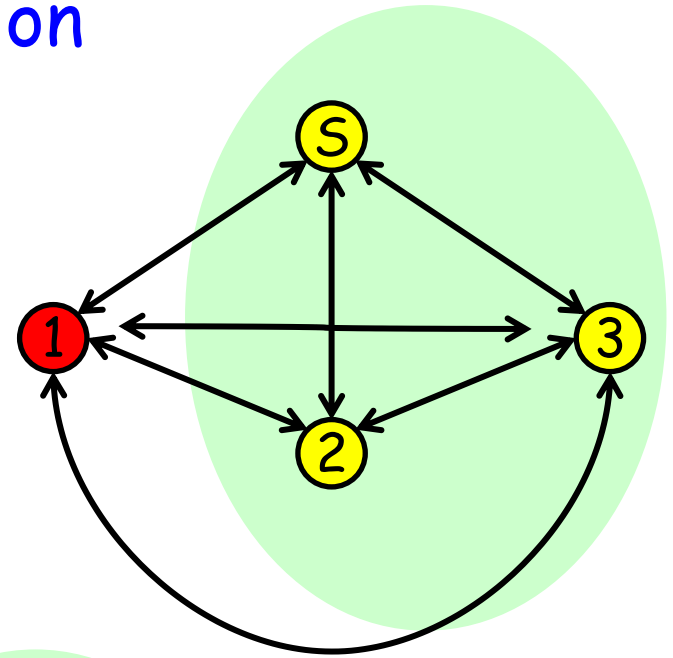
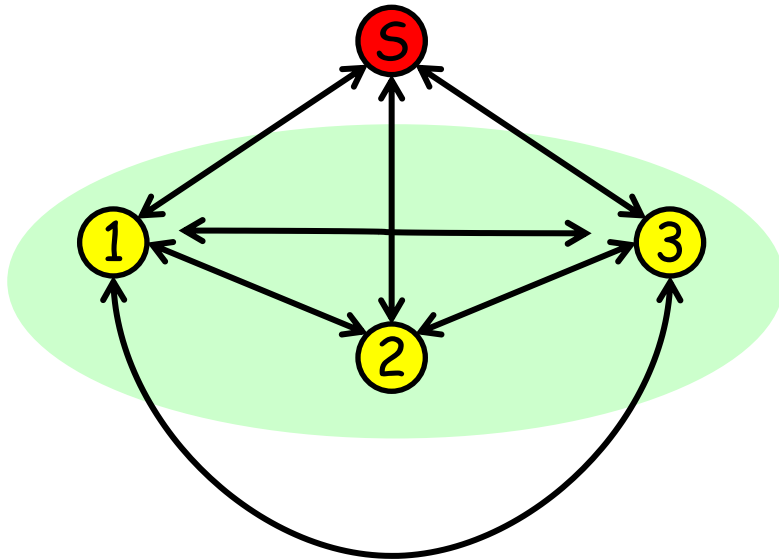
# Failure Detection



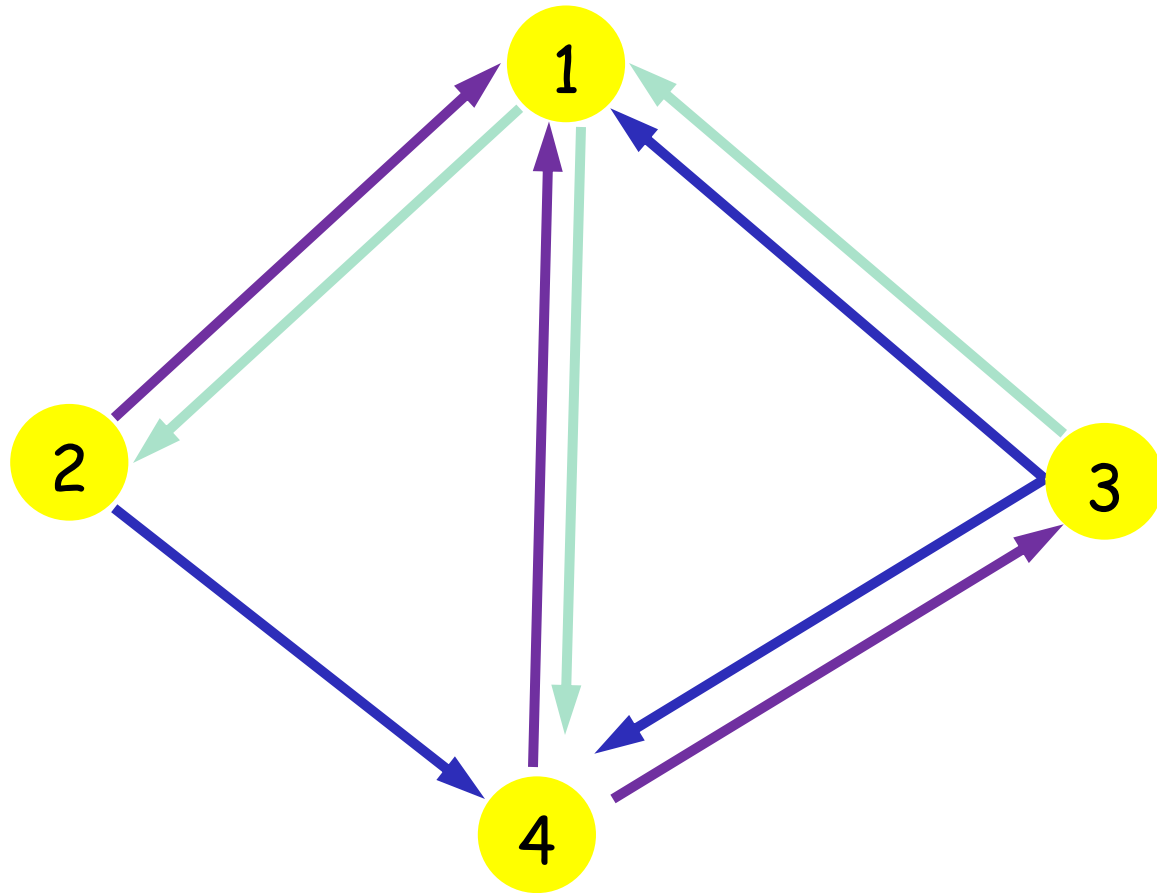
# Failure Detection



# Failure Detection

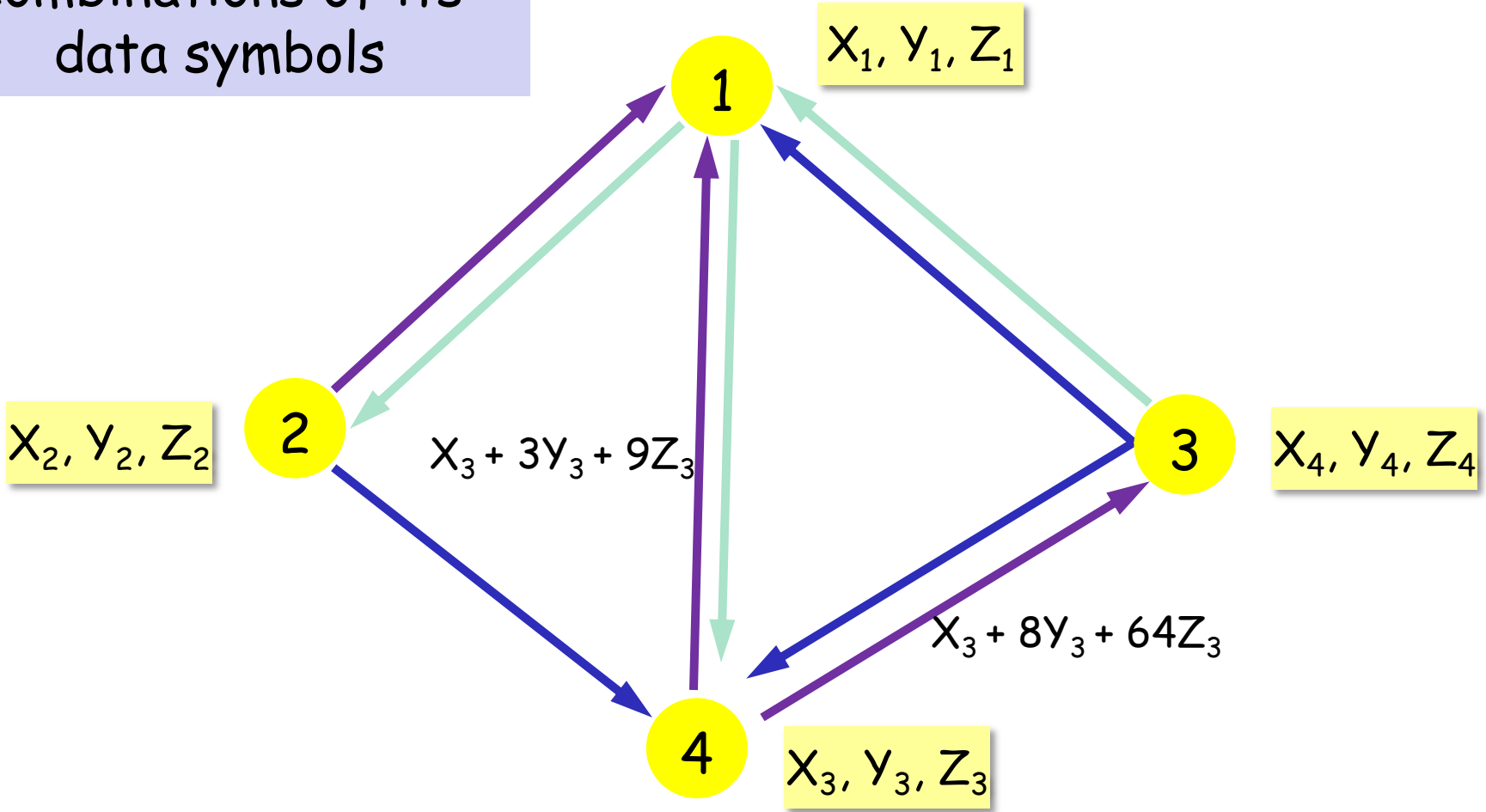


# Local Coding



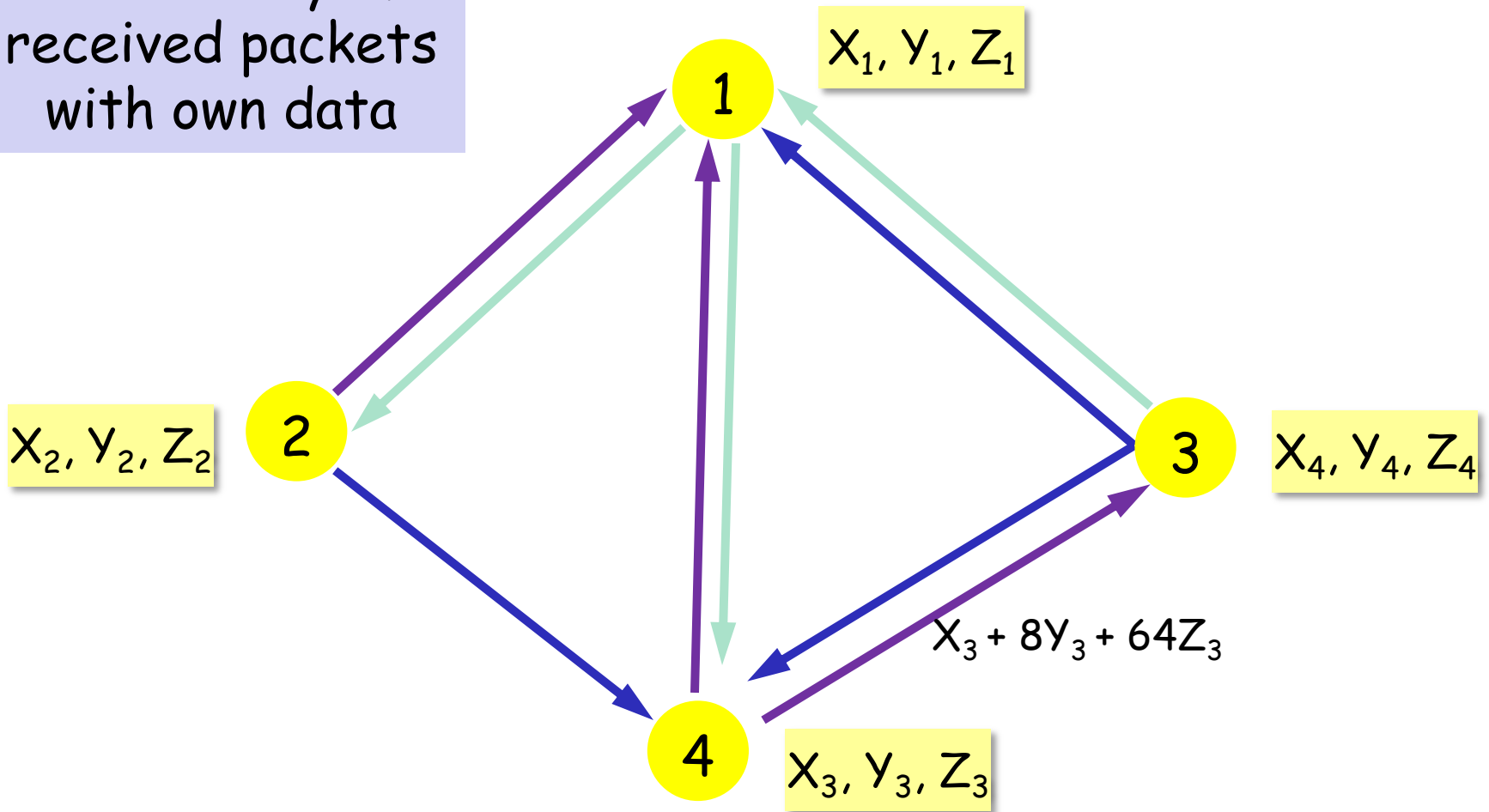
Each directed link can carry 1 symbol

Each node sends linear combinations of its data symbols



Each node checks consistency of received packets with own data

$$X_1 + 3Y_1 + 9Z_1 = X_3 + 3Y_3 + 9Z_3 \quad ?$$

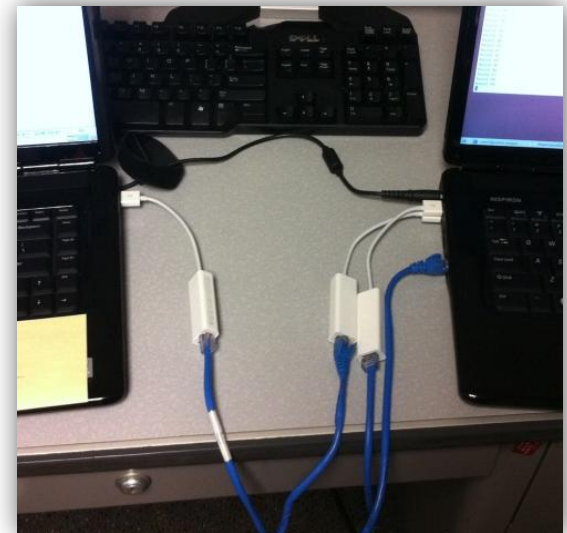


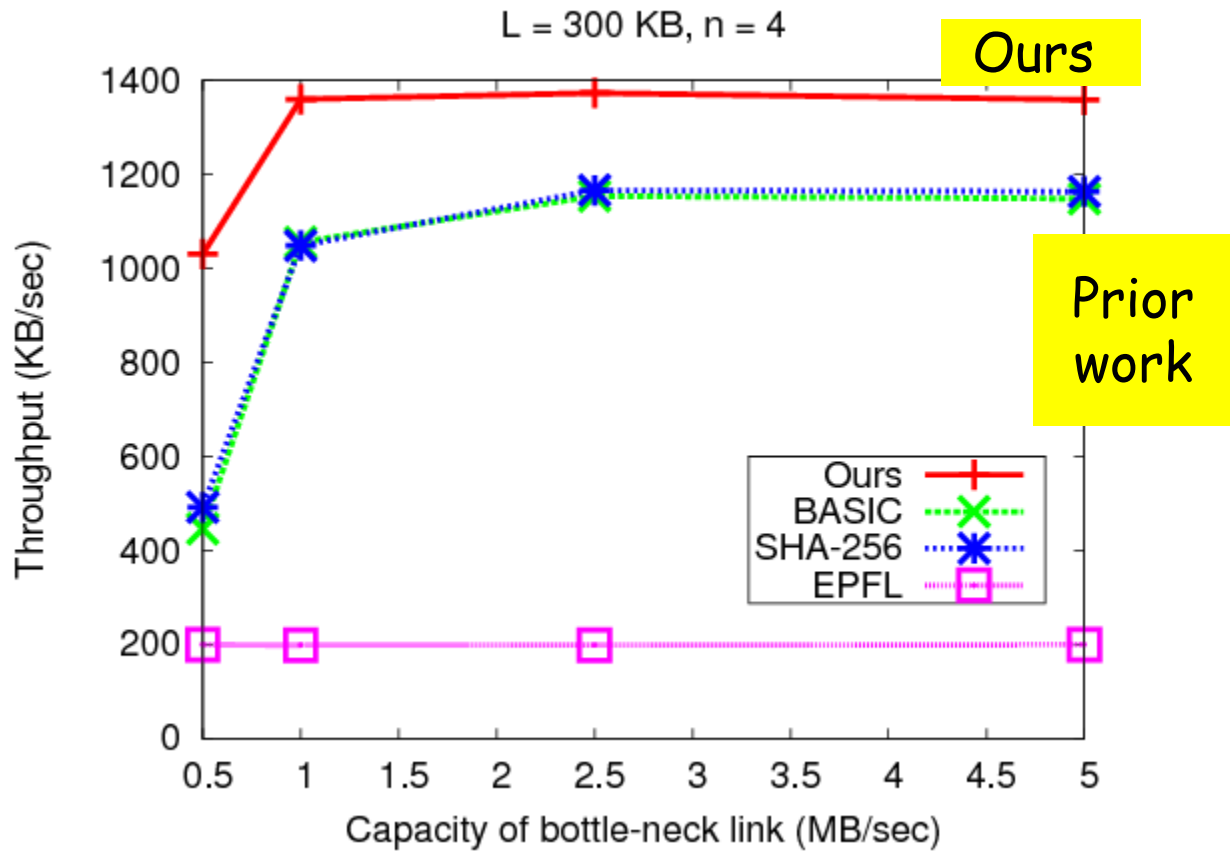
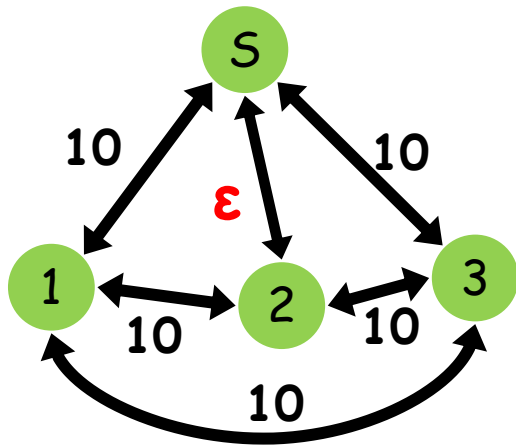


# Failure Detection

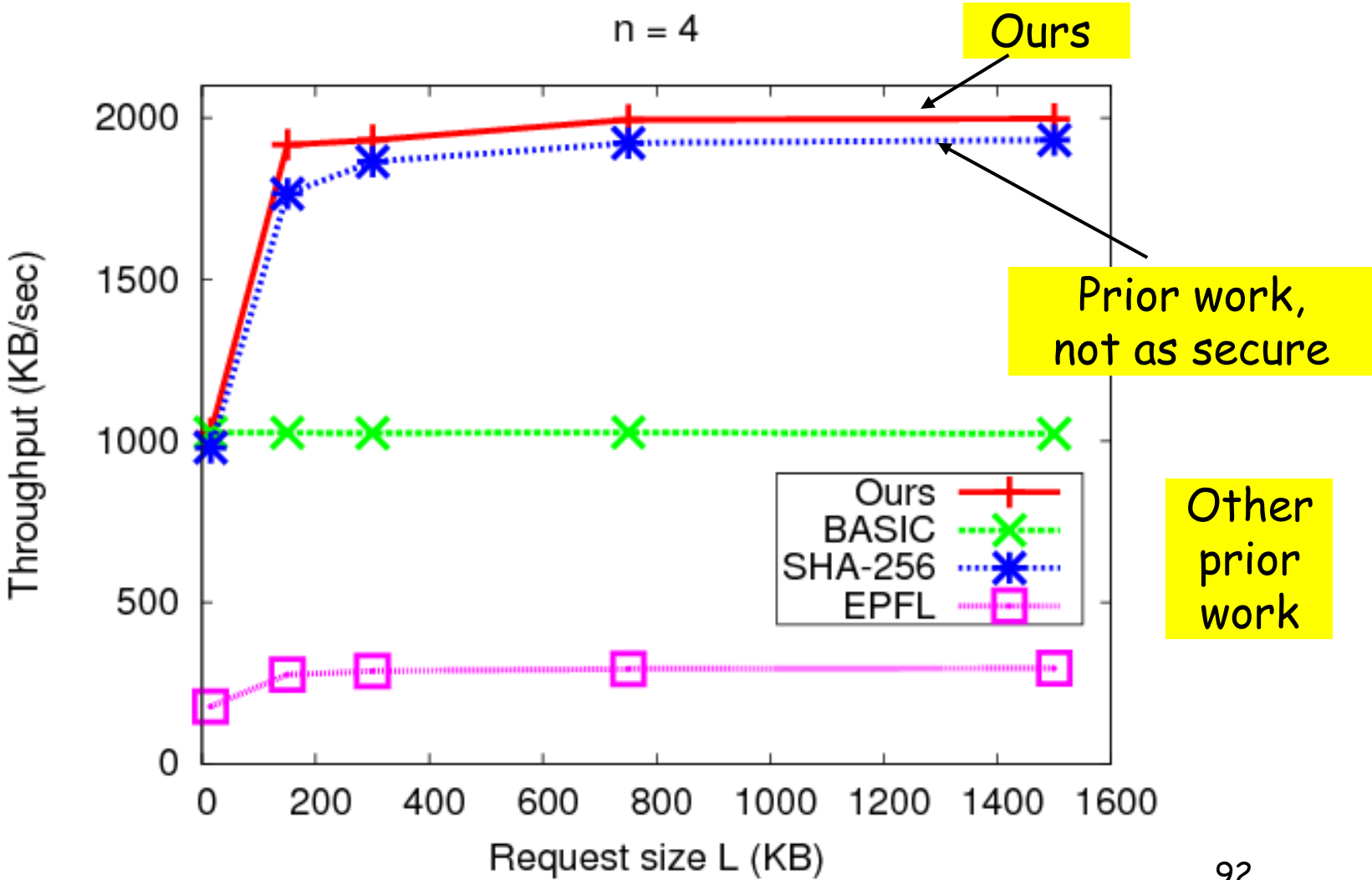
- Equality function
- Faulty nodes should not be able to make unequal values appear equal
- Utilize link capacities

# Experimental Evaluation





# Ethernet: Failure-Free Case



# Wrap-Up

# This Talk

- Byzantine broadcast

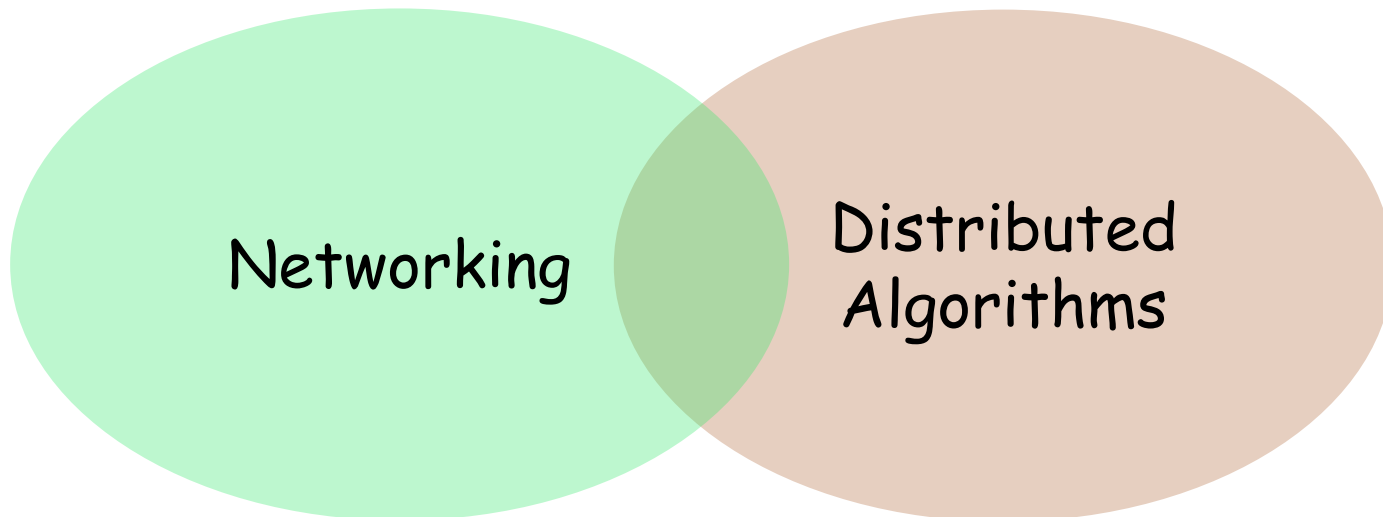
- To illustrate

impact of network

on algorithm design & performance

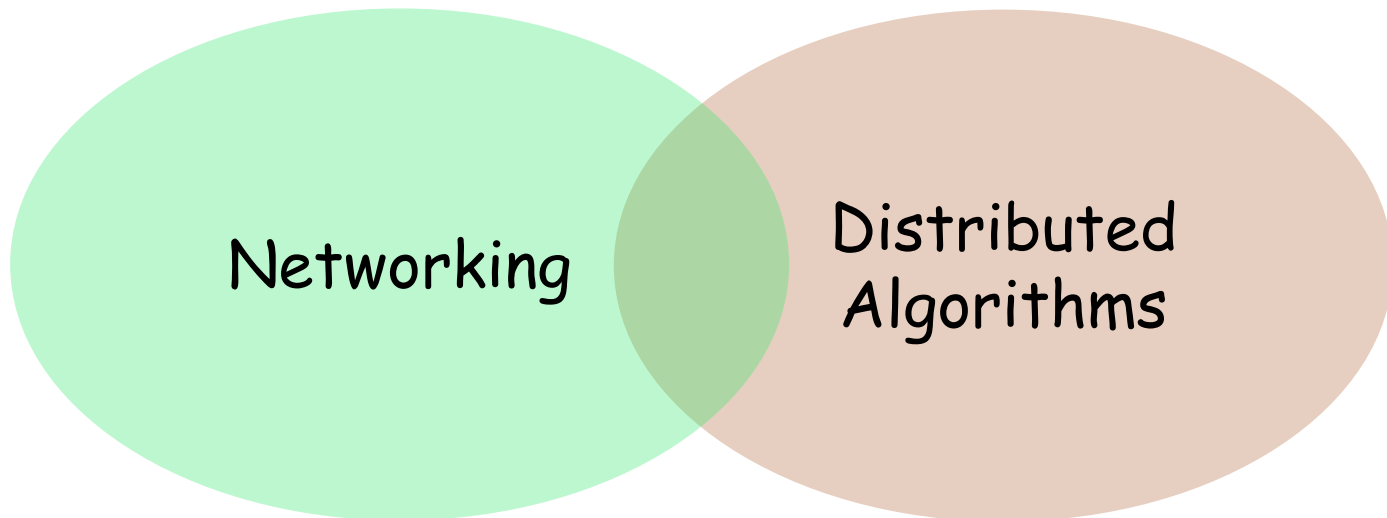
# Rich Problem Space

- More realism in network model can change solutions quite significantly



# Rich Problem Space

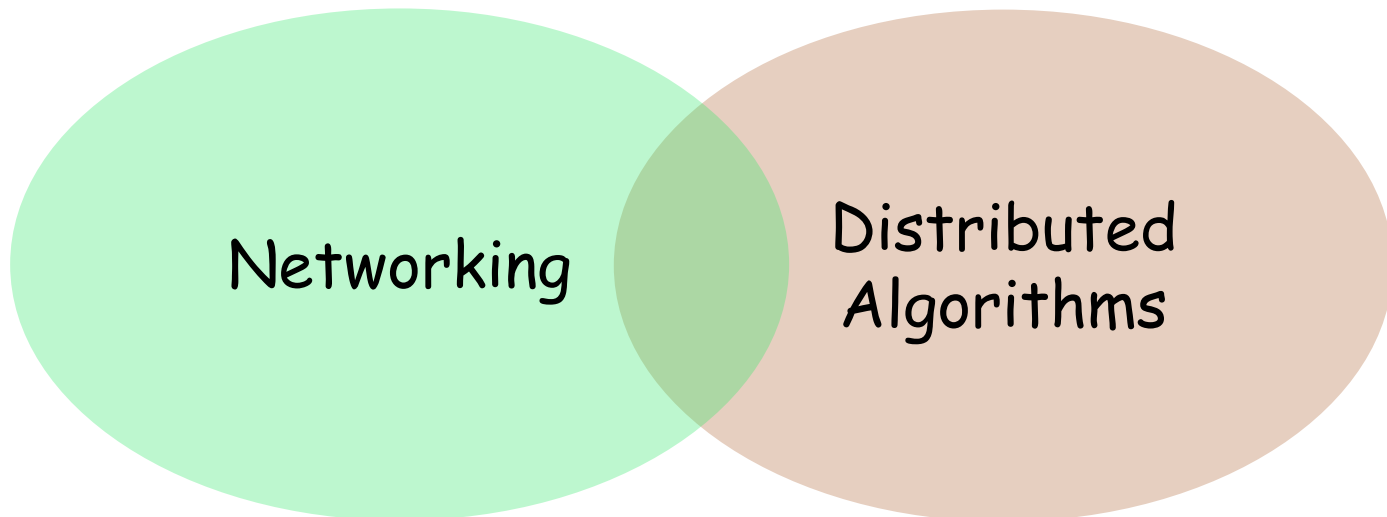
- Networks ... wired, wireless
- Computations ... many of interest
- Metrics ... how to capture impact of networks?





# Rich Problem Space

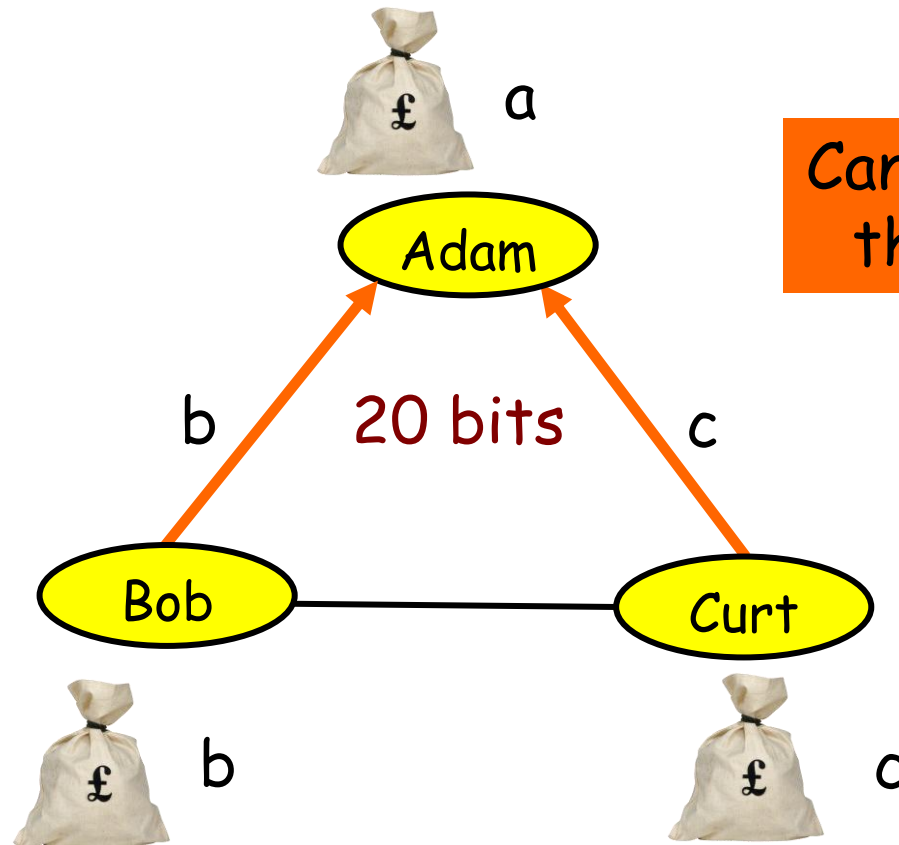
Need new ways to  
formulate & solve  
old problems





Thanks!

# Puzzler



Can we do better than 20 bits ?



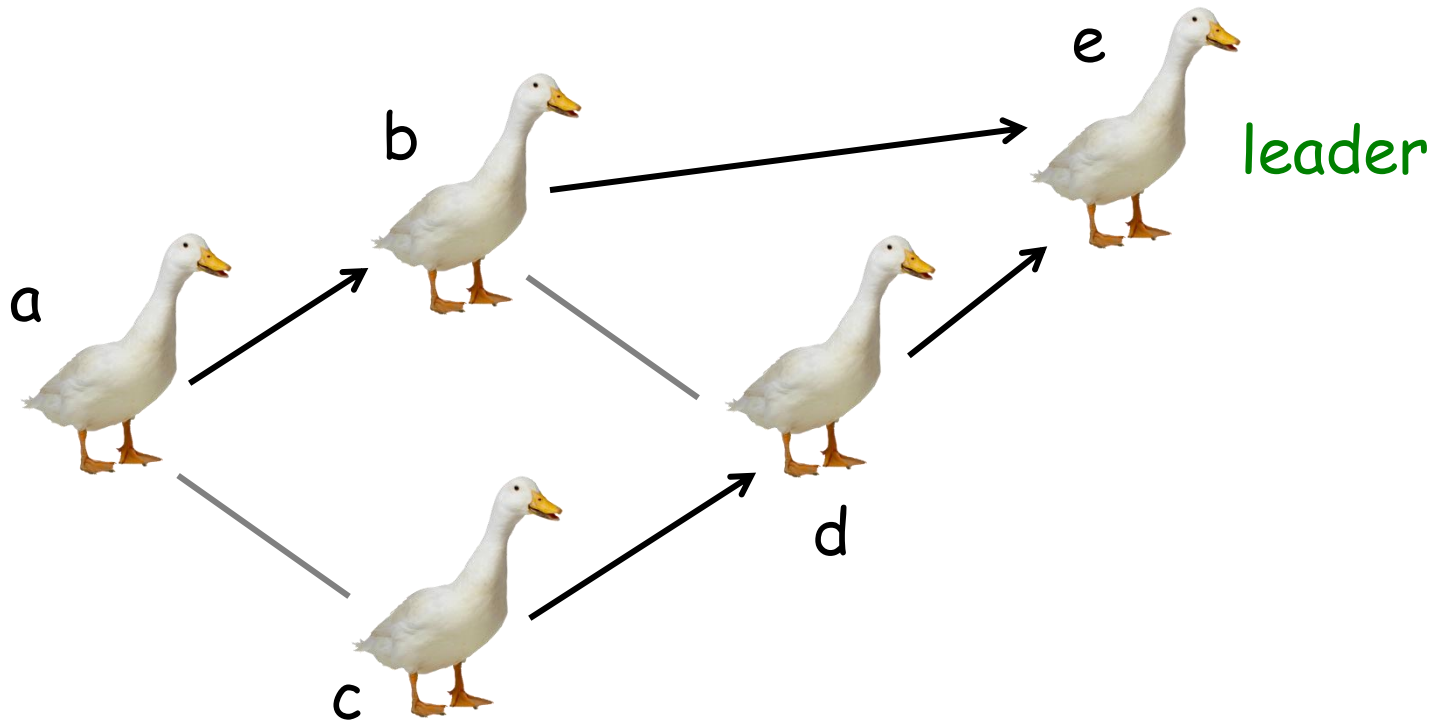
Thanks!



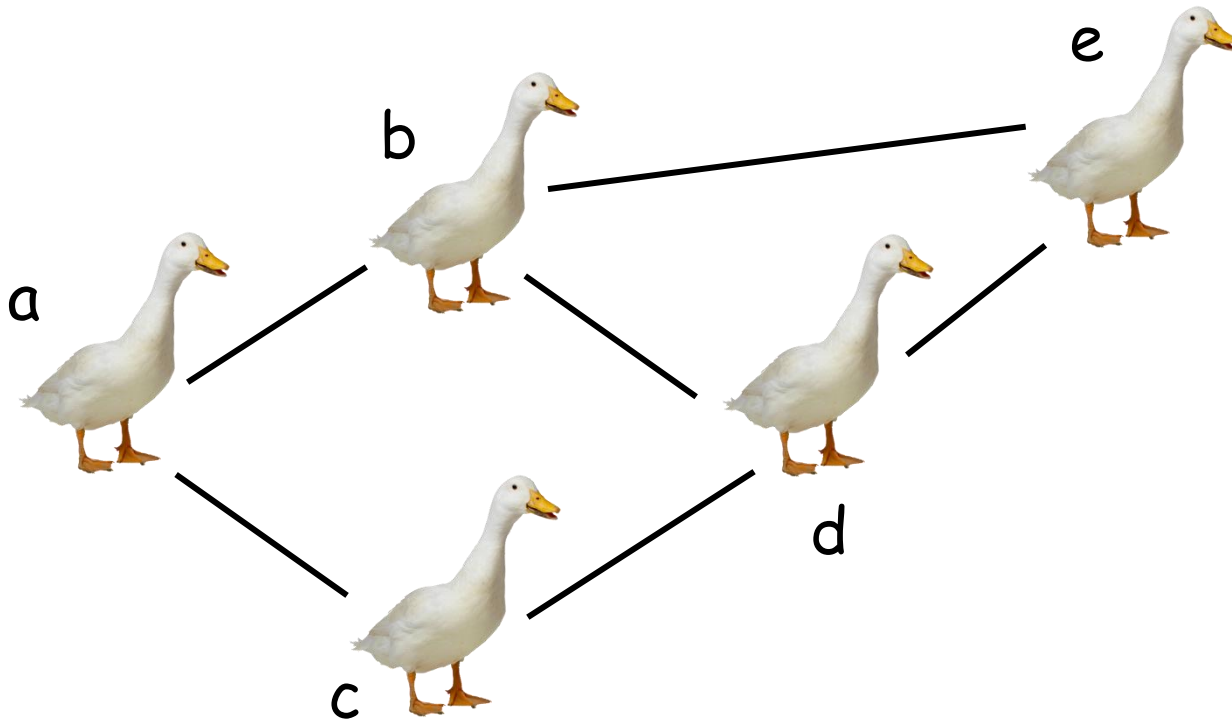
Thanks!

# Average Consensus

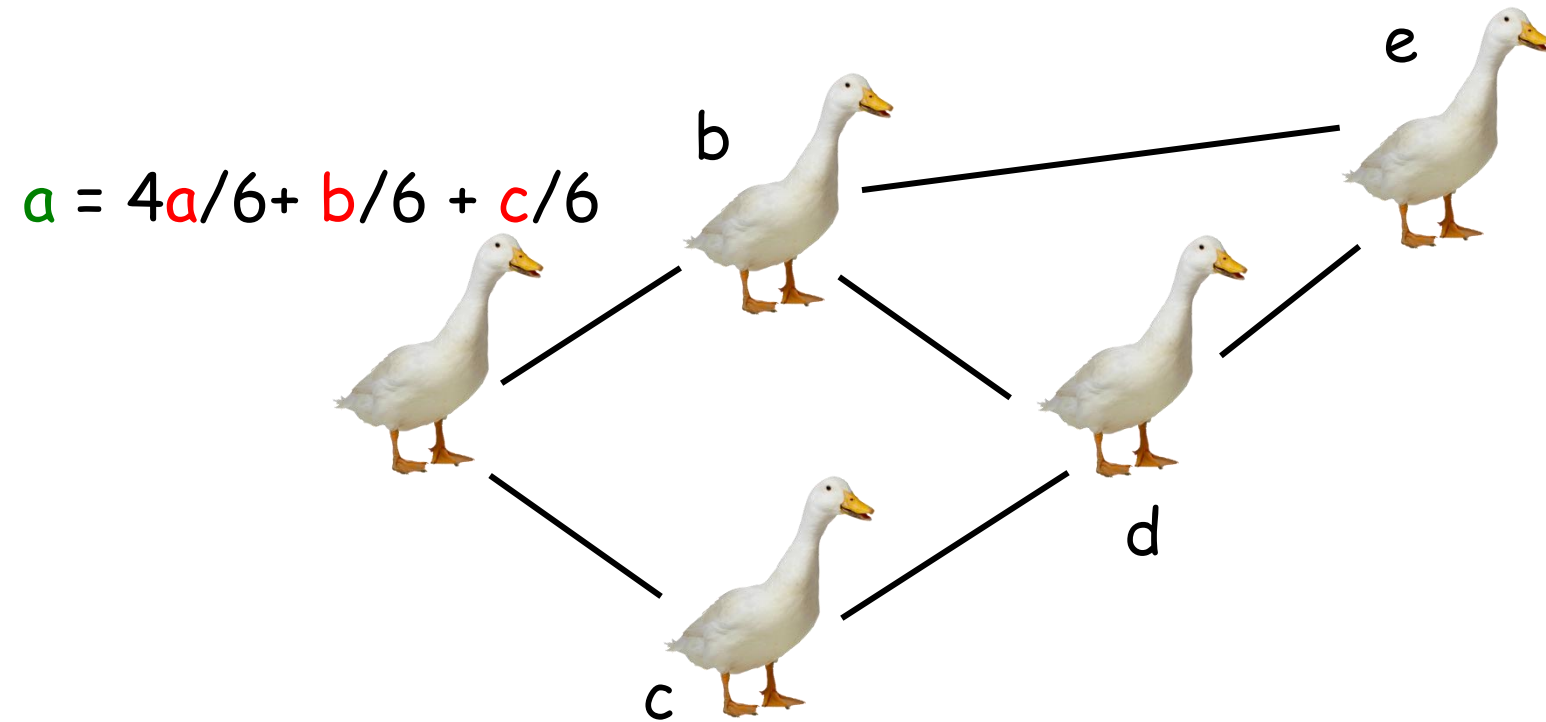
- Centralized solution



# Iterative Average Consensus

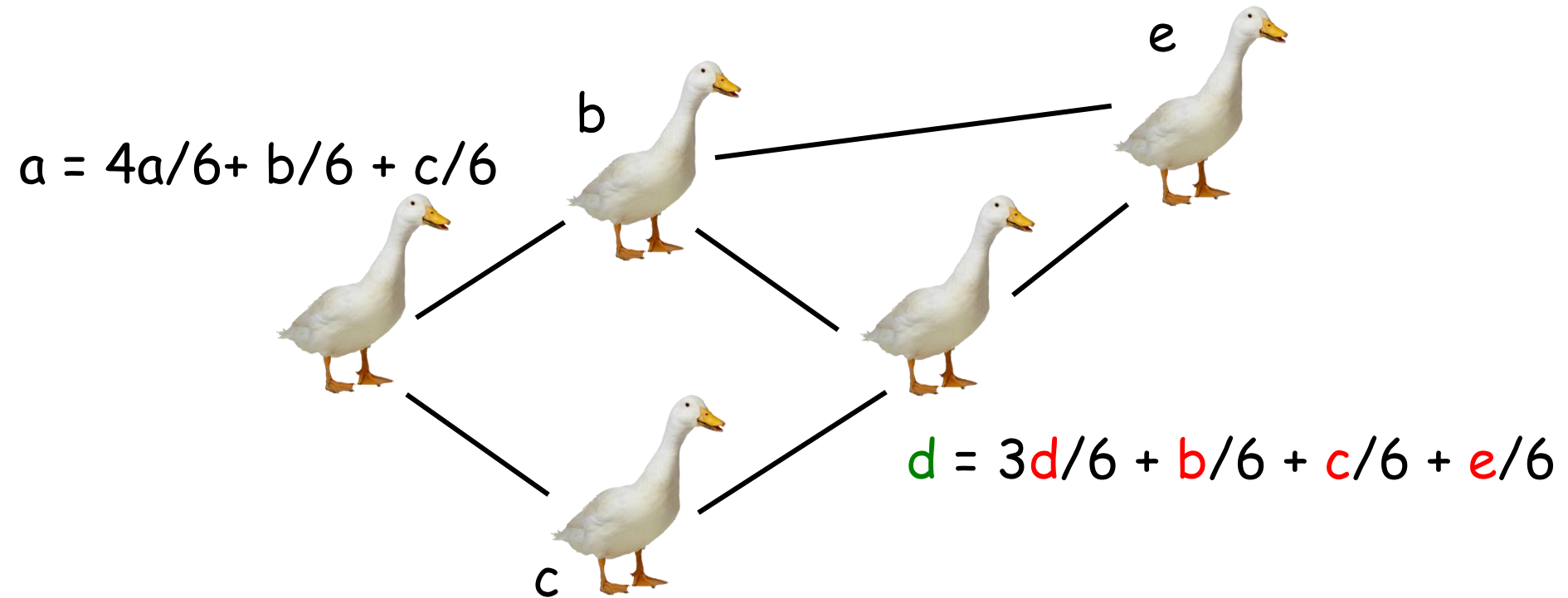


# Iterative Average Consensus





# Iterative Average Consensus

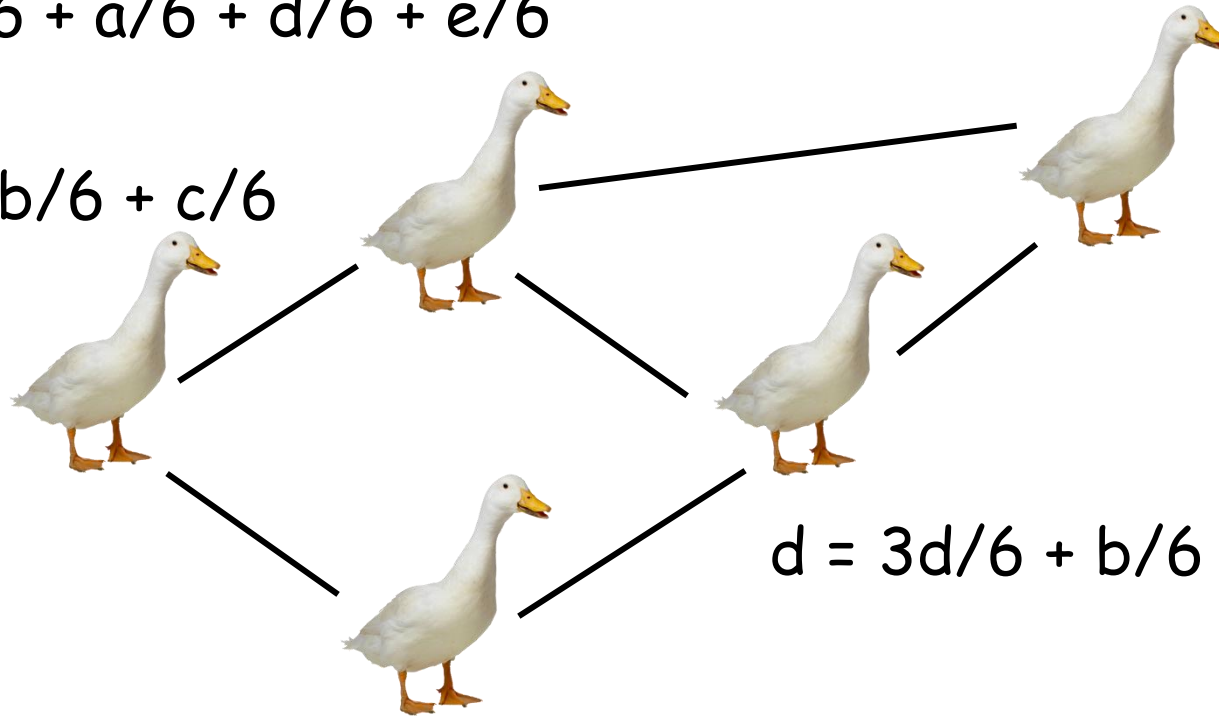


# Iterative Average Consensus

$$b = 3b/6 + a/6 + d/6 + e/6$$

$$e = 3e/6 + b/6 + c/6 + d/6$$

$$a = 4a/6 + b/6 + c/6$$



$$d = 3d/6 + b/6 + c/6 + e/6$$

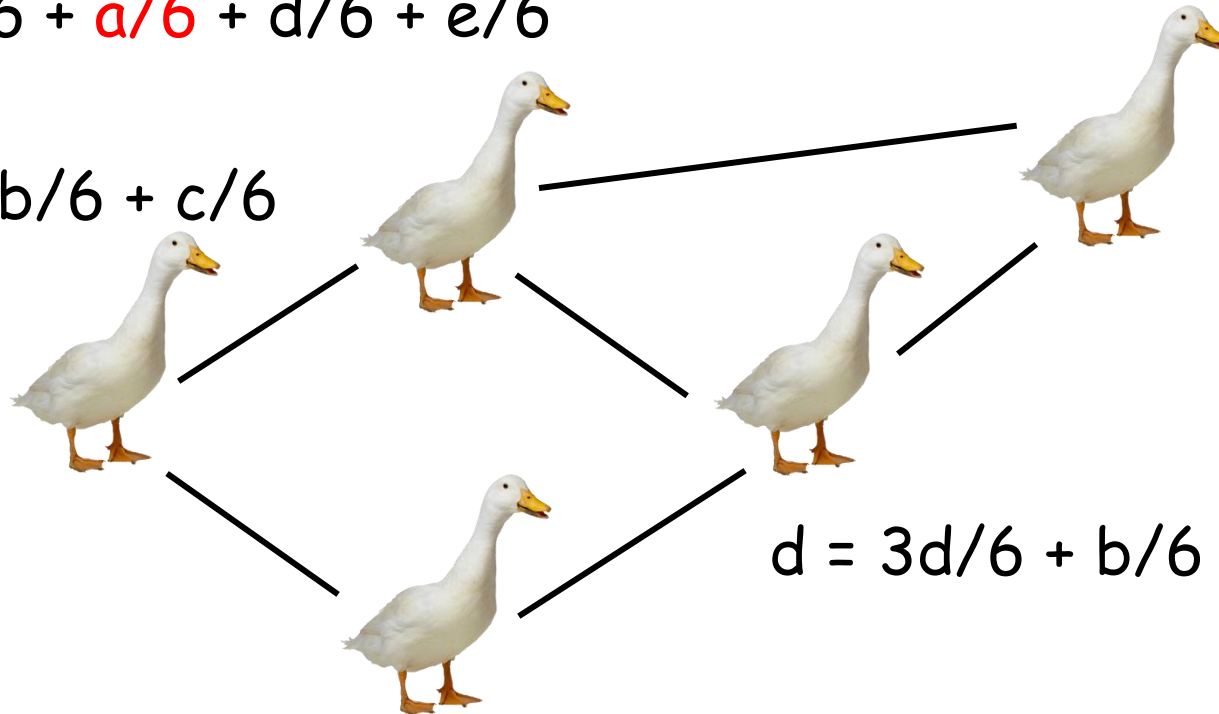
$$c = 4c/6 + a/6 + d/6$$

# Iterative Average Consensus

$$b = 3b/6 + a/6 + d/6 + e/6$$

$$e = 3e/6 + b/6 + c/6 + d/6$$

$$a = 4a/6 + b/6 + c/6$$



$$d = 3d/6 + b/6 + c/6 + e/6$$

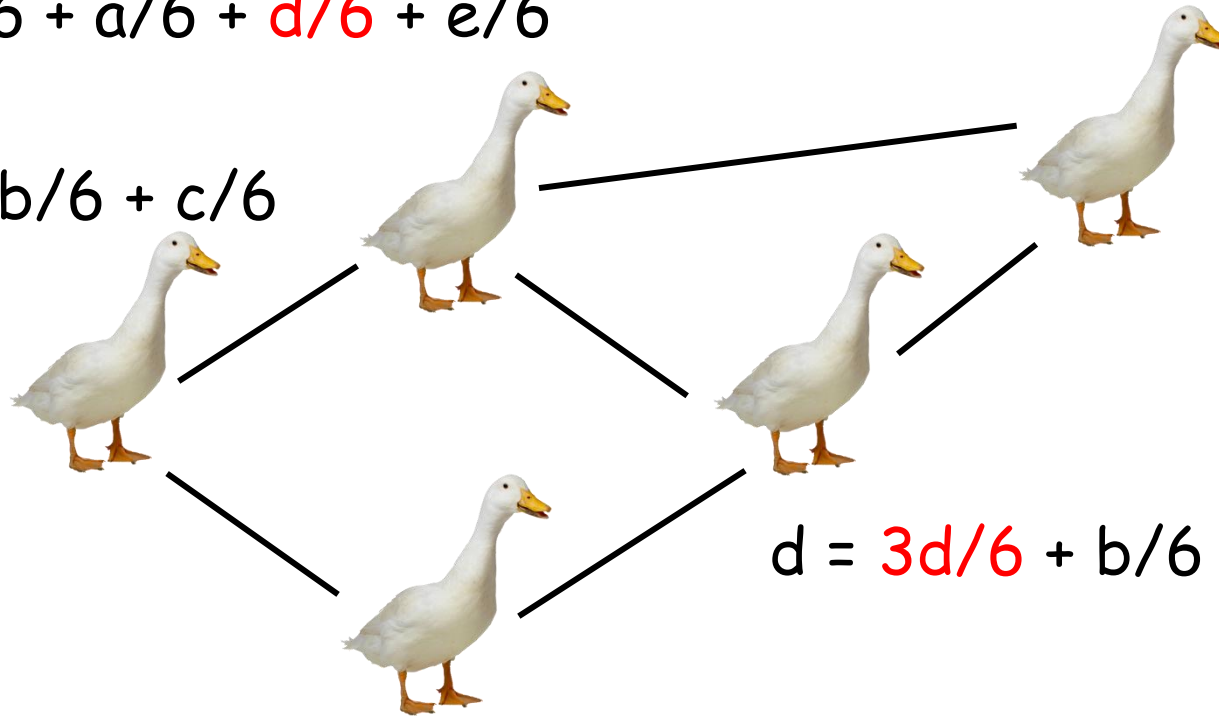
$$c = 4c/6 + a/6 + d/6$$

# Iterative Average Consensus

$$b = 3b/6 + a/6 + d/6 + e/6$$

$$e = 3e/6 + b/6 + c/6 + d/6$$

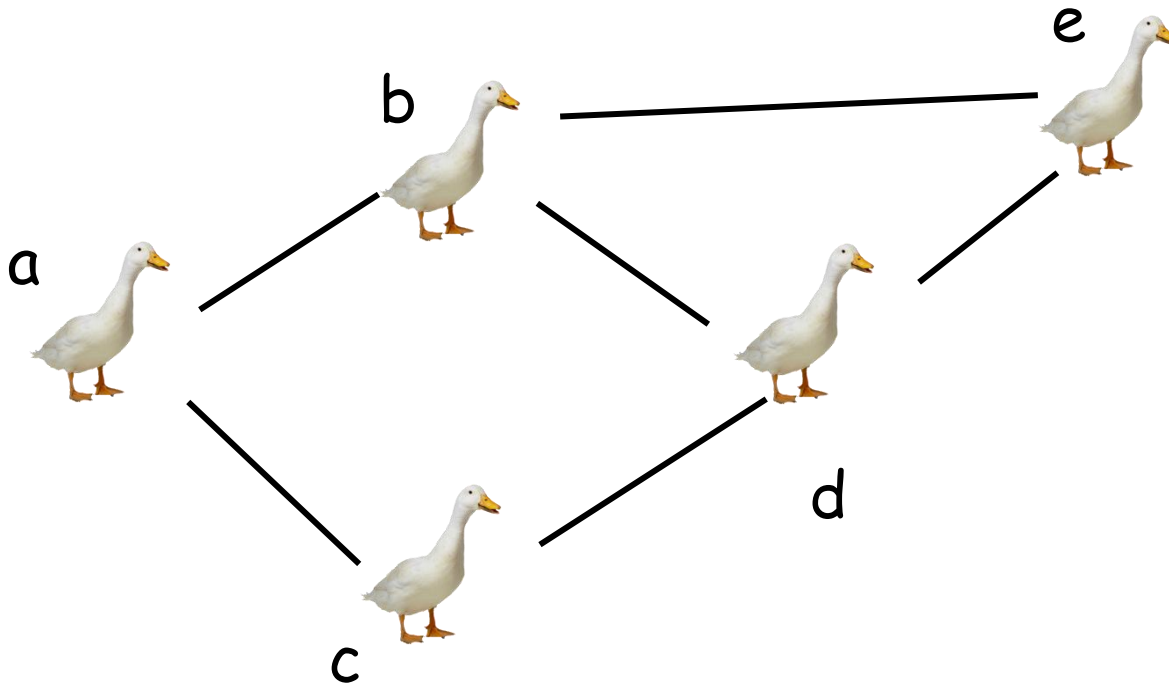
$$a = 4a/6 + b/6 + c/6$$



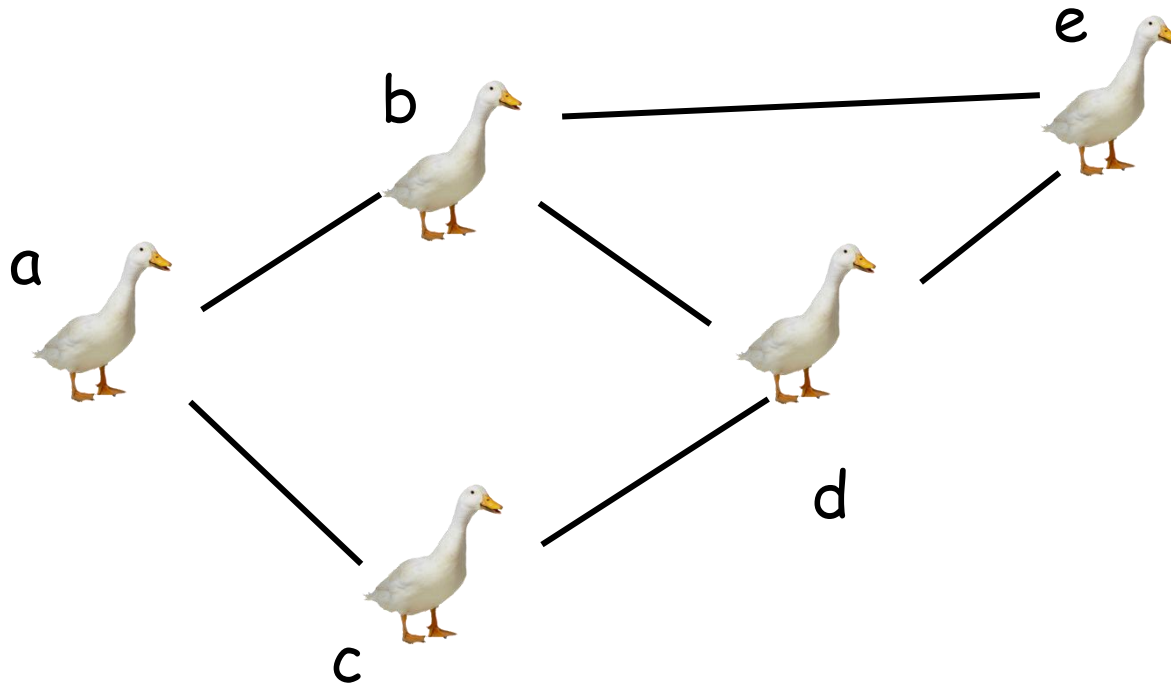
$$d = 3d/6 + b/6 + c/6 + e/6$$

$$c = 4c/6 + a/6 + d/6$$

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 4/6 & 1/6 & 1/6 & 0 & 0 \\ 1/6 & 3/6 & 0 & 1/6 & 1/6 \\ 1/6 & 0 & 4/6 & 1/6 & 0 \\ 0 & 1/6 & 1/6 & 3/6 & 1/6 \\ 0 & 1/6 & 0 & 1/6 & 4/6 \end{bmatrix} \begin{bmatrix} a \\ b \\ e \\ d \\ e \end{bmatrix}$$



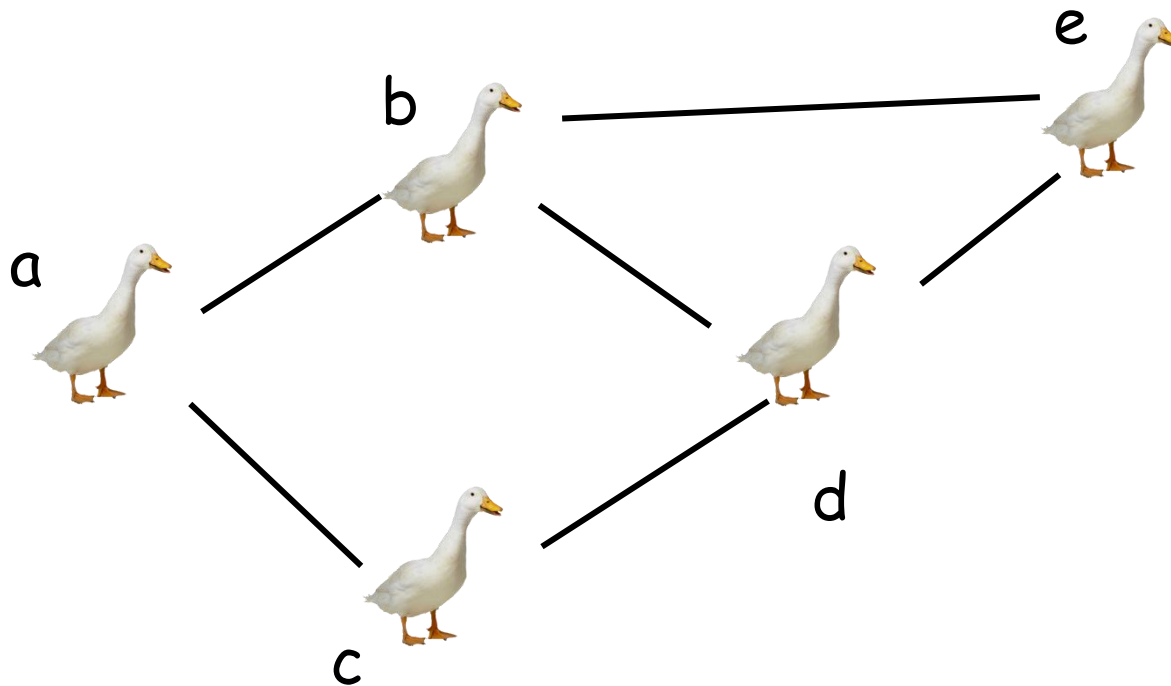
$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 4/6 & 1/6 & 1/6 & 0 & 0 \\ 1/6 & 3/6 & 0 & 1/6 & 1/6 \\ 1/6 & 0 & 4/6 & 1/6 & 0 \\ 0 & 1/6 & 1/6 & 3/6 & 1/6 \\ 0 & 1/6 & 0 & 1/6 & 4/6 \end{bmatrix} \begin{bmatrix} a \\ b \\ e \\ d \\ e \end{bmatrix}$$



$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 4/6 & 1/6 & 1/6 & 0 & 0 \\ 1/6 & 3/6 & 0 & 1/6 & 1/6 \\ 1/6 & 0 & 4/6 & 1/6 & 0 \\ 0 & 1/6 & 1/6 & 3/6 & 1/6 \\ 0 & 1/6 & 0 & 1/6 & 4/6 \end{bmatrix} \begin{bmatrix} a \\ b \\ e \\ d \\ e \end{bmatrix}$$

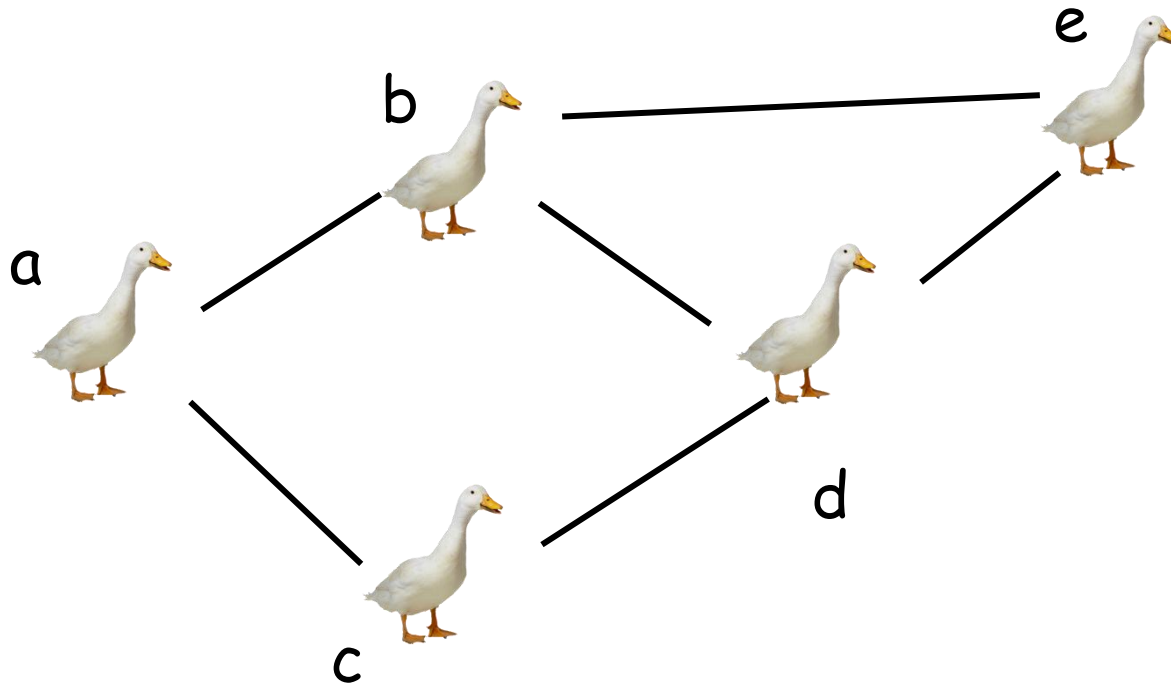


Node B sends fractions of its mass to neighbors



$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 4/6 & 1/6 & 1/6 & 0 & 0 \\ 1/6 & 3/6 & 0 & 1/6 & 1/6 \\ 1/6 & 0 & 4/6 & 1/6 & 0 \\ 0 & 1/6 & 1/6 & 3/6 & 1/6 \\ 0 & 1/6 & 0 & 1/6 & 4/6 \end{bmatrix} \begin{bmatrix} a \\ b \\ e \\ d \\ e \end{bmatrix}$$

Node B accumulates mass sent by neighbors

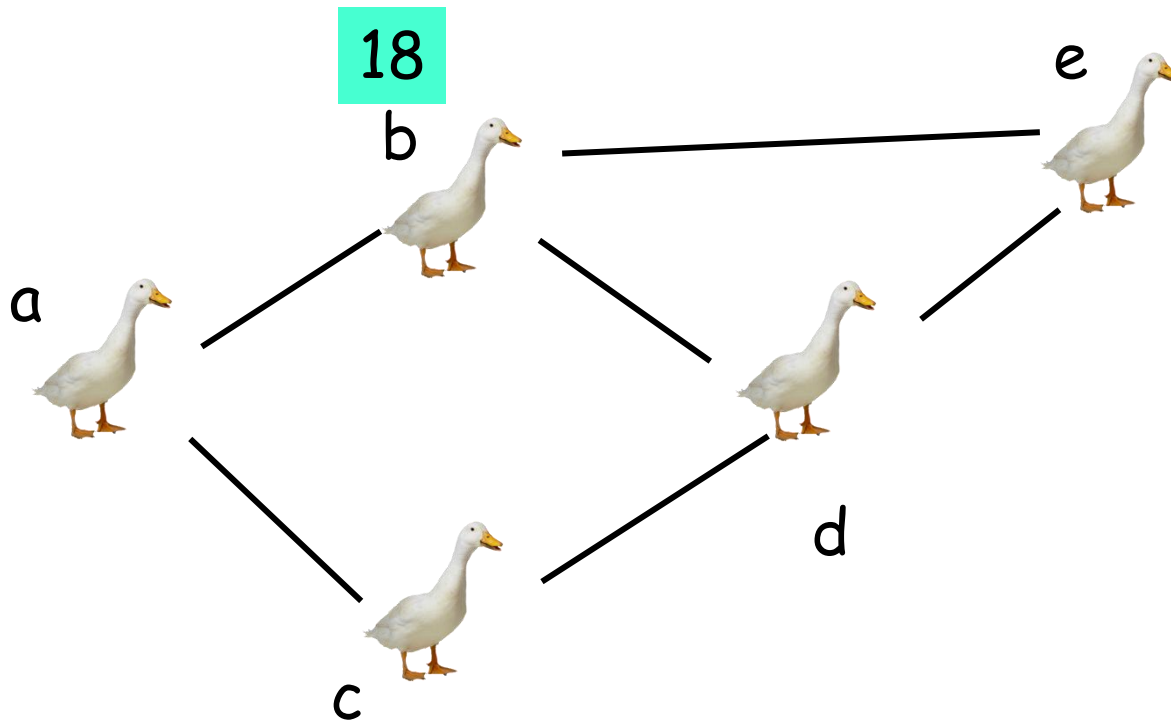




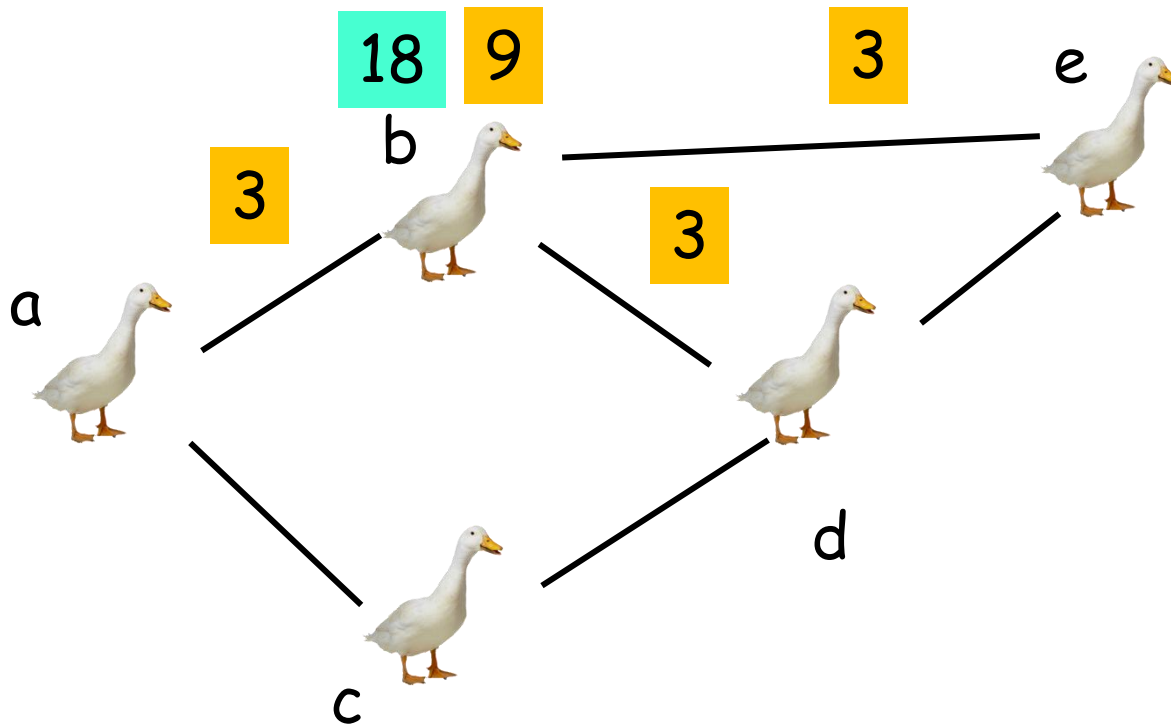
## Well-Known Result

- State of the nodes converges to average
- Results assuming loss-less links

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 4/6 & 1/6 & 1/6 & 0 & 0 \\ 1/6 & 3/6 & 0 & 1/6 & 1/6 \\ 1/6 & 0 & 4/6 & 1/6 & 0 \\ 0 & 1/6 & 1/6 & 3/6 & 1/6 \\ 0 & 1/6 & 0 & 1/6 & 4/6 \end{bmatrix} \begin{bmatrix} a \\ b \\ e \\ d \\ e \end{bmatrix}$$



$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 4/6 & 1/6 & 1/6 & 0 & 0 \\ 1/6 & 3/6 & 0 & 1/6 & 1/6 \\ 1/6 & 0 & 4/6 & 1/6 & 0 \\ 0 & 1/6 & 1/6 & 3/6 & 1/6 \\ 0 & 1/6 & 0 & 1/6 & 4/6 \end{bmatrix} \begin{bmatrix} a \\ b \\ e \\ d \\ e \end{bmatrix}$$



# Wireless Network Model

- Time varying topology  
... mobility of nodes, links breaking, etc.
- Algorithm converges to average  
if available links are reliable  
and the topology is connected over time

## More Accurate Model ?

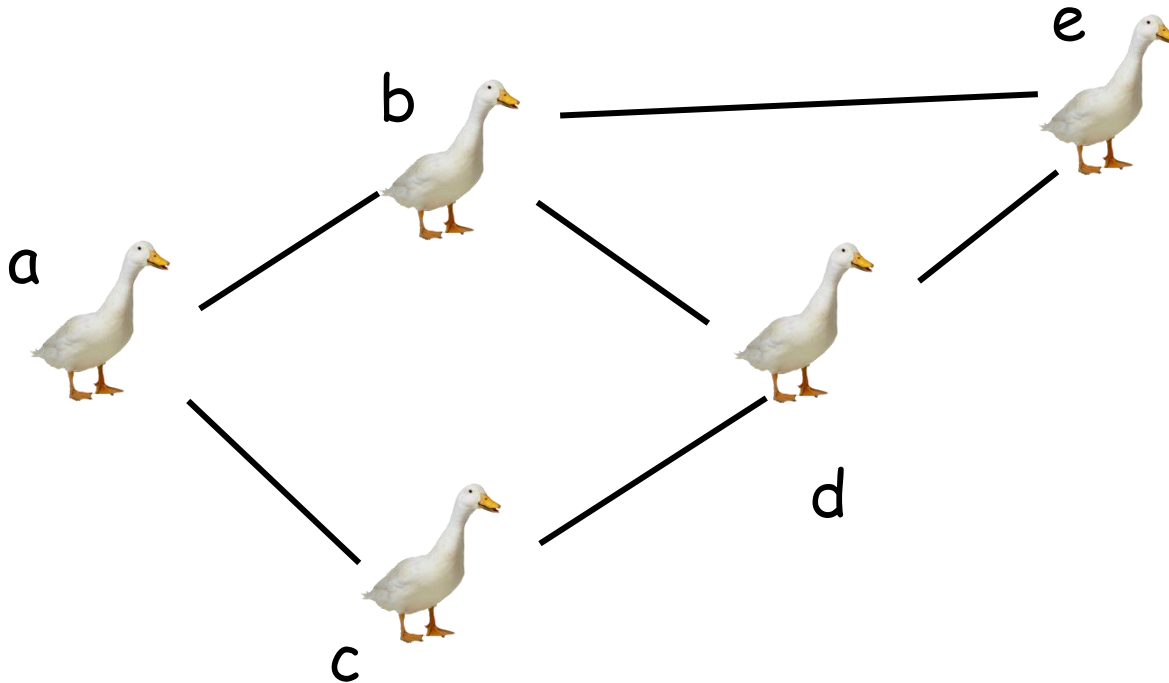
- Unreliable transmissions
- "Mass transfer" needs to be reliable for the algorithm to work
  
- B should know that A has received mass
- A should know that B knows that A has received mass
- ...
- Common knowledge required

# Unreliable Links

- How to design iterative algorithms in presence of unreliable links
- Changes the problem & solution approach significantly
- Possible to converge to average

# Lossy Links

- Node B may not be able to reliably transfer mass to a neighbor





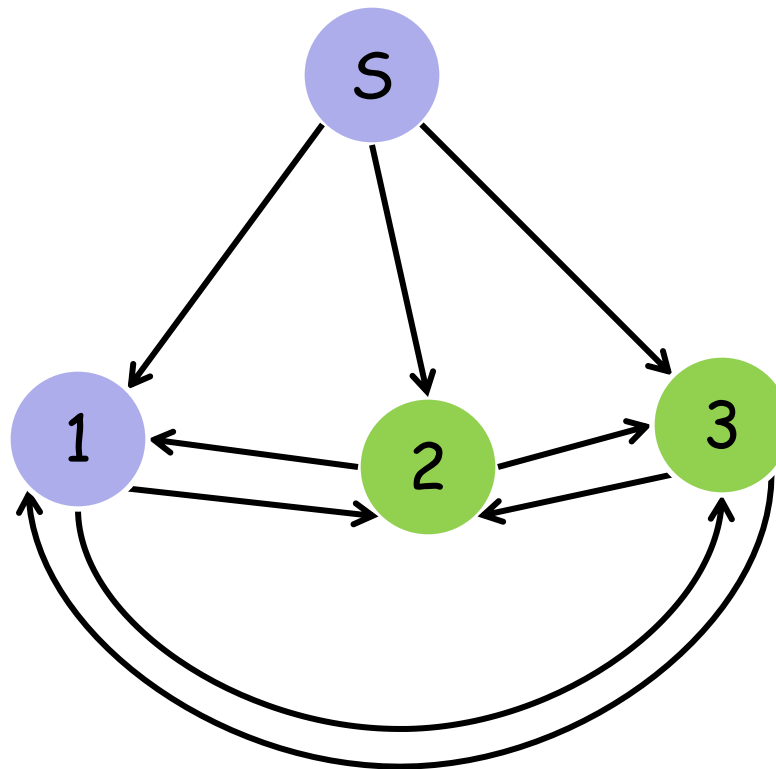
Thanks!



# Asymmetric Networks

- Upper bound 1 on throughput

min-cut( $S, X$  |  $f$  peers removed)

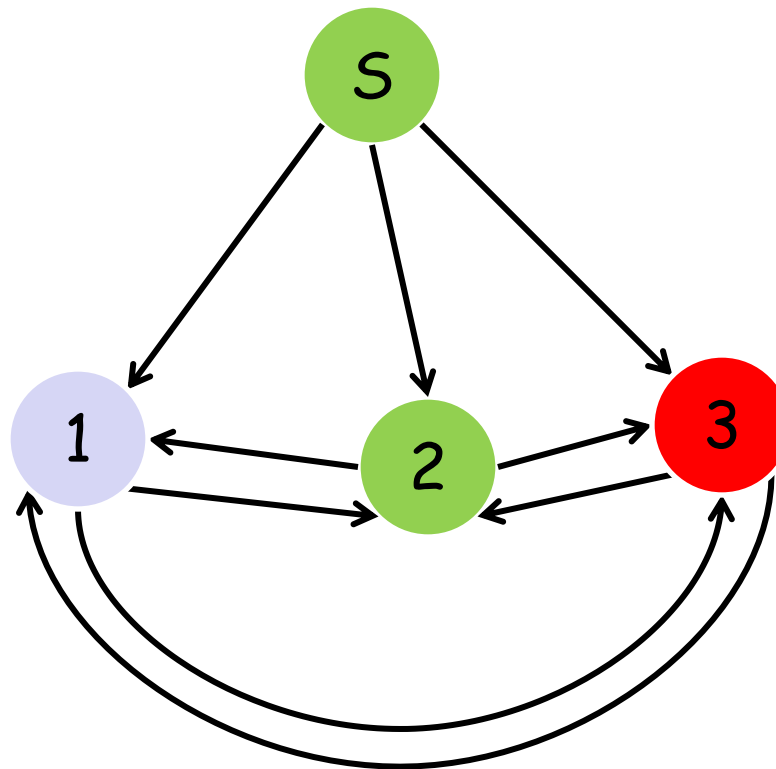


$f = 1$   
 $X = 1$

# Asymmetric Networks

- Upper bound 1 on throughput

min-cut( $S, X$  |  $f$  peers removed)

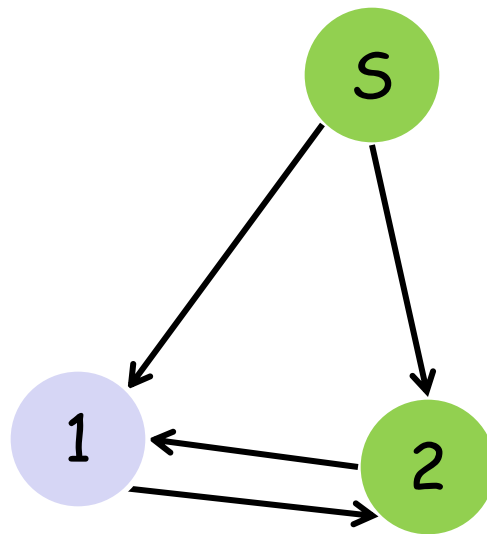


$f = 1$   
 $X = 1$

# Asymmetric Networks

- Upper bound 1 on throughput

min-cut( $S, X$  |  $f$  peers removed)

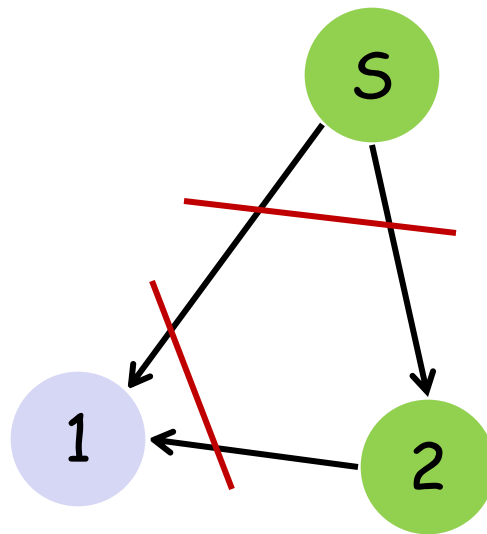


$$f = 1$$
$$X = 1$$

# Asymmetric Networks

- Upper bound 1 on throughput

min-cut( $S, X$  |  $f$  peers removed)

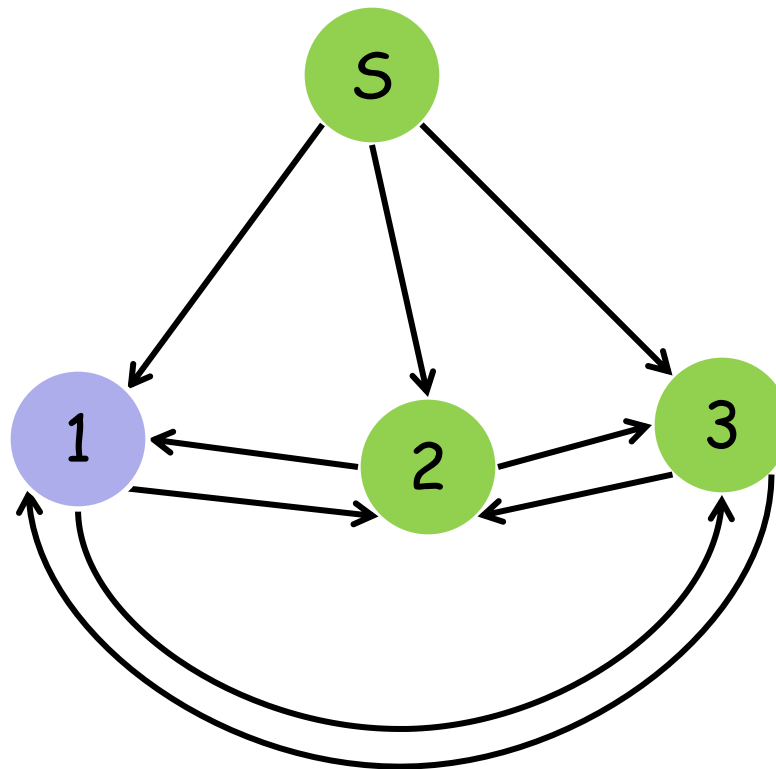


$$f = 1$$
$$X = 1$$

# Asymmetric Networks

- Upper bound 2 on throughput

incoming( $X$  |  $f$  nodes removed)

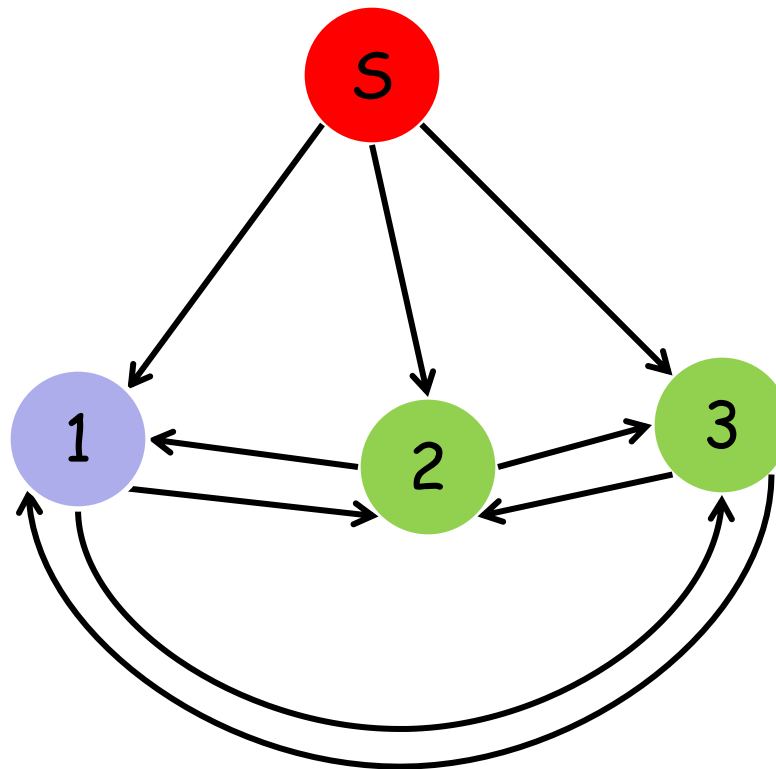


$$f = 1$$
$$X = 1$$

# Asymmetric Networks

- Upper bound 2 on throughput

incoming( $X$  |  $f$  nodes removed)

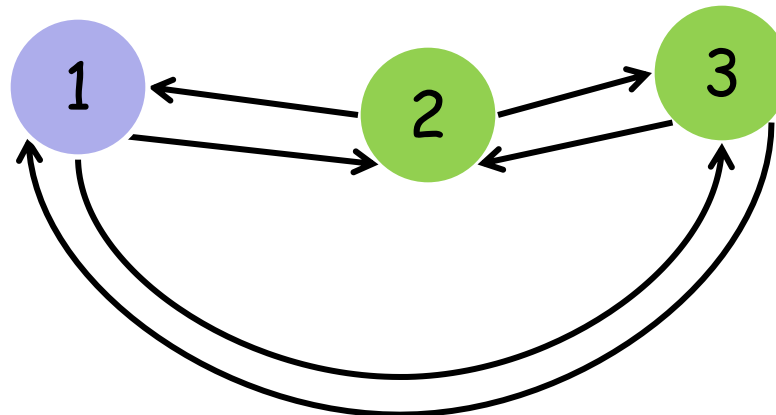


$$f = 1$$
$$X = 1$$

# Asymmetric Networks

- Upper bound 2 on throughput

incoming( $X$  |  $f$  nodes removed)

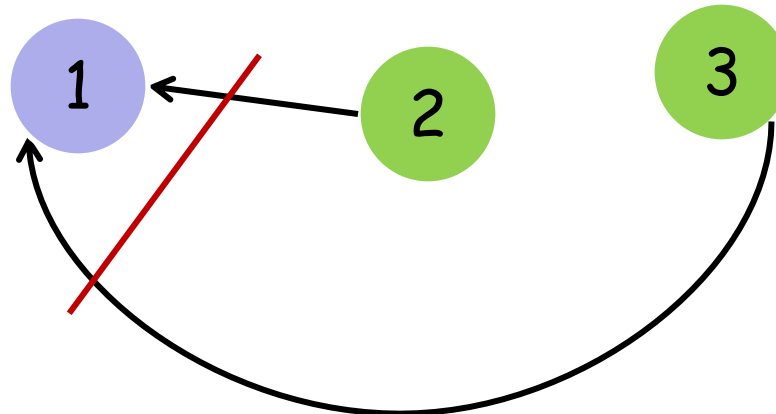


$$f = 1$$
$$X = 1$$

# Asymmetric Networks

- Upper bound 2 on throughput

incoming( $X$  |  $f$  nodes removed)



$$f = 1$$
$$X = 1$$



# 4-Node Networks

- Our approach using  
capacity-dependent coding  
optimal

# Arbitrary Networks

## Reduction

Consensus with Byzantine fault tolerance

→ Consensus with Byzantine fault detection

→ Multi-party equality (with local communication)

# Local Coding

- No forwarding of packets
- Code and check locally
- Desirable property when using in Byzantine broadcast ... faulty nodes cannot tamper packets, if they don't forward anything

# Claims

- Bad nodes cannot tamper someone else's packets
- If no good node finds inconsistency,  
their values are identical
- This equality checking helps  
achieve Byzantine broadcast within  
constant fraction of optimal

# After Failure Identified

