Tutorial : Byzantine agreement

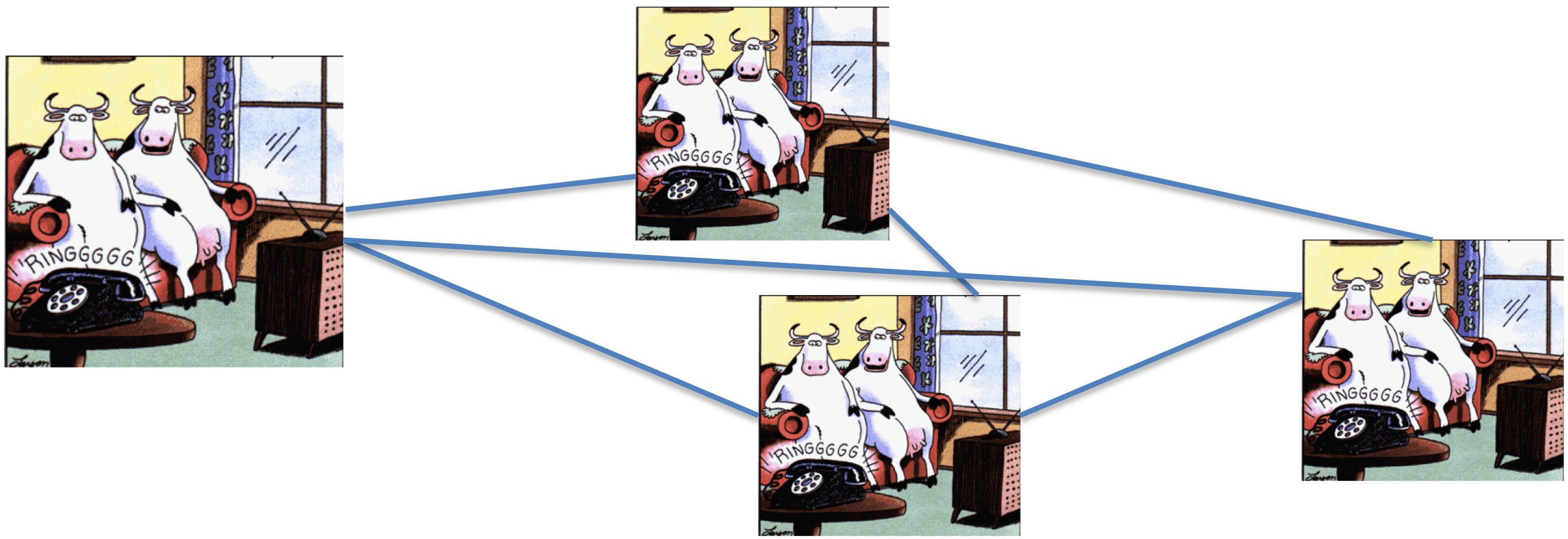
Valerie King
University of Victoria
Victoria, Canada

Byzantine Generals Problem

"We imagine that several divisions of the Byzantine army are camped outside an enemy city, each division commanded by its own general. The generals can communicate with one another only by messenger. After observing the enemy, they must decide upon a common plan of action. However, some of the generals may be traitors, trying to prevent the loyal generals from reaching agreement..."

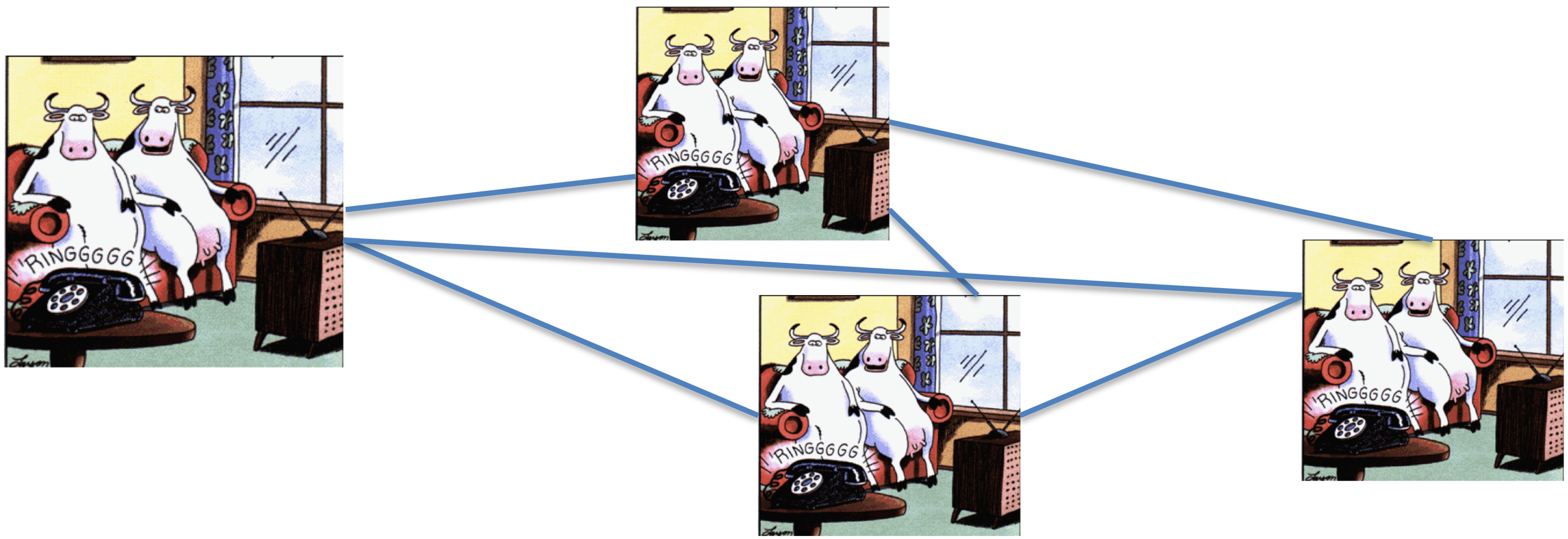
--Lamport, Shostak and Pease, 1978

Byzantine Agreement



To model worst case faults in networks where processors communicate via point-to-point links

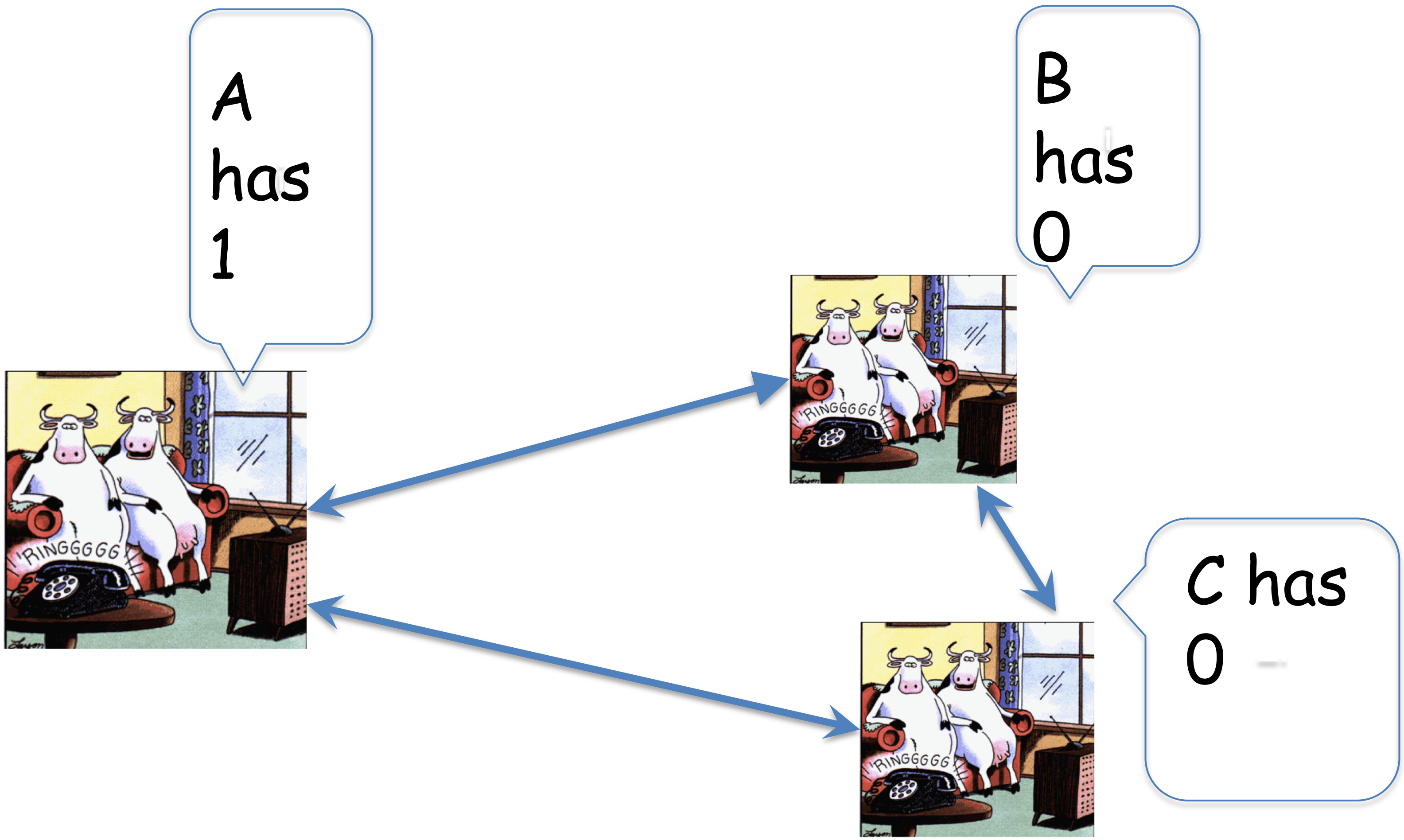
Byzantine Agreement



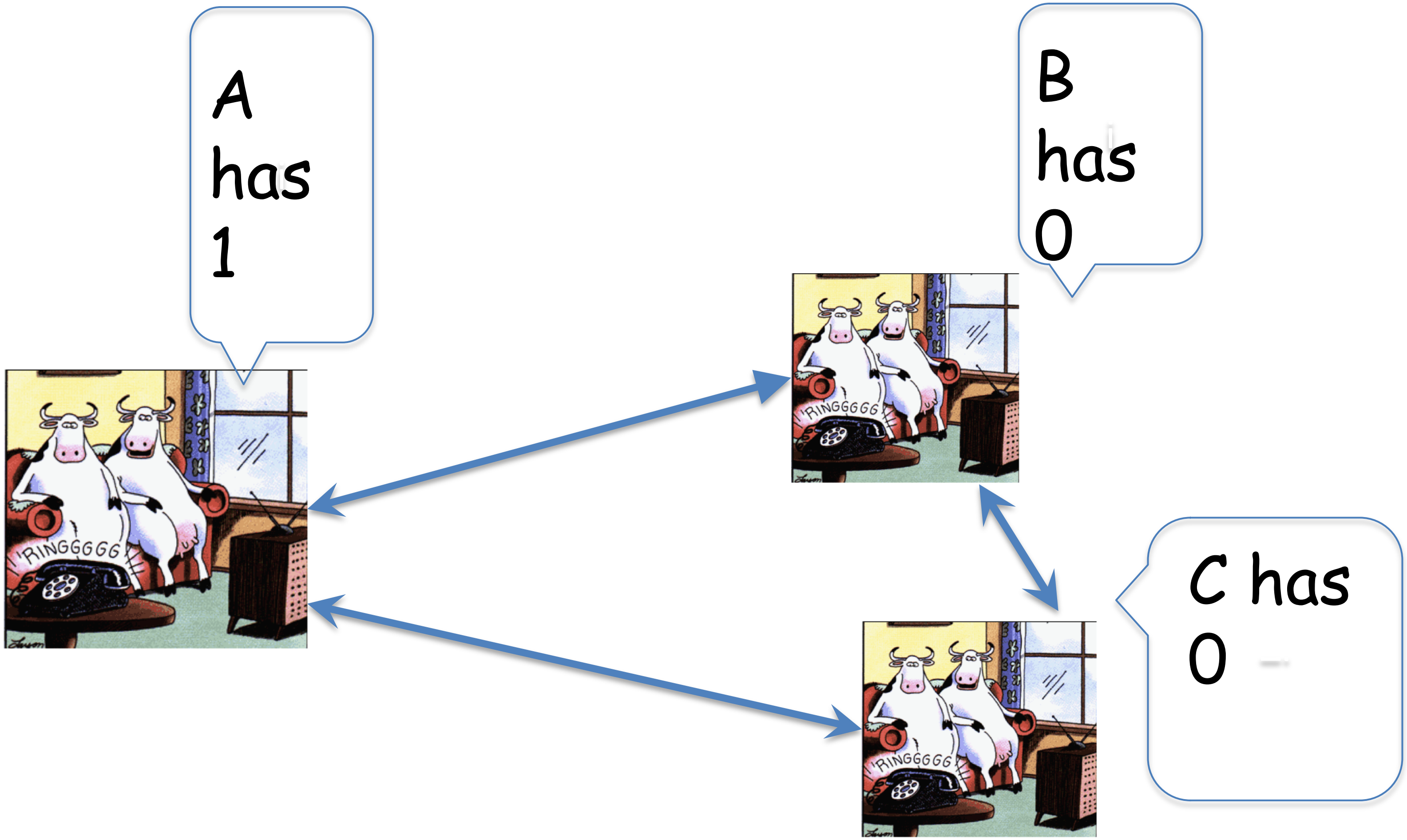
To model worst case faults in networks where processors communicate via point-to-point links

All pairs are connected

Source of message known to recipient



Start with initial bits; exchanges messages, then output same bit. If all start with the same bit, must output that bit.



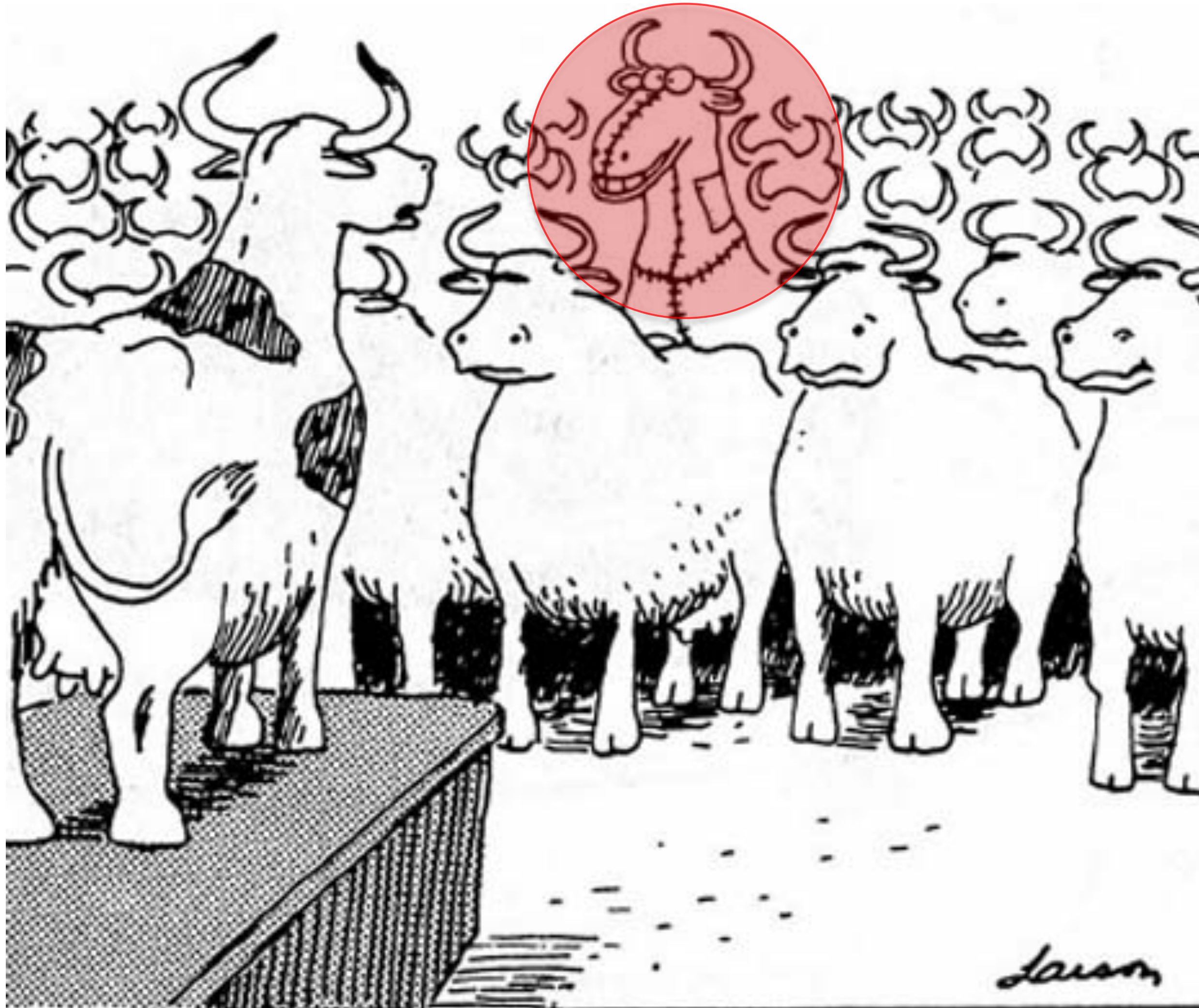
Agreement Protocol: Send each other input bit and vote

Byzantine Adversary (BA)

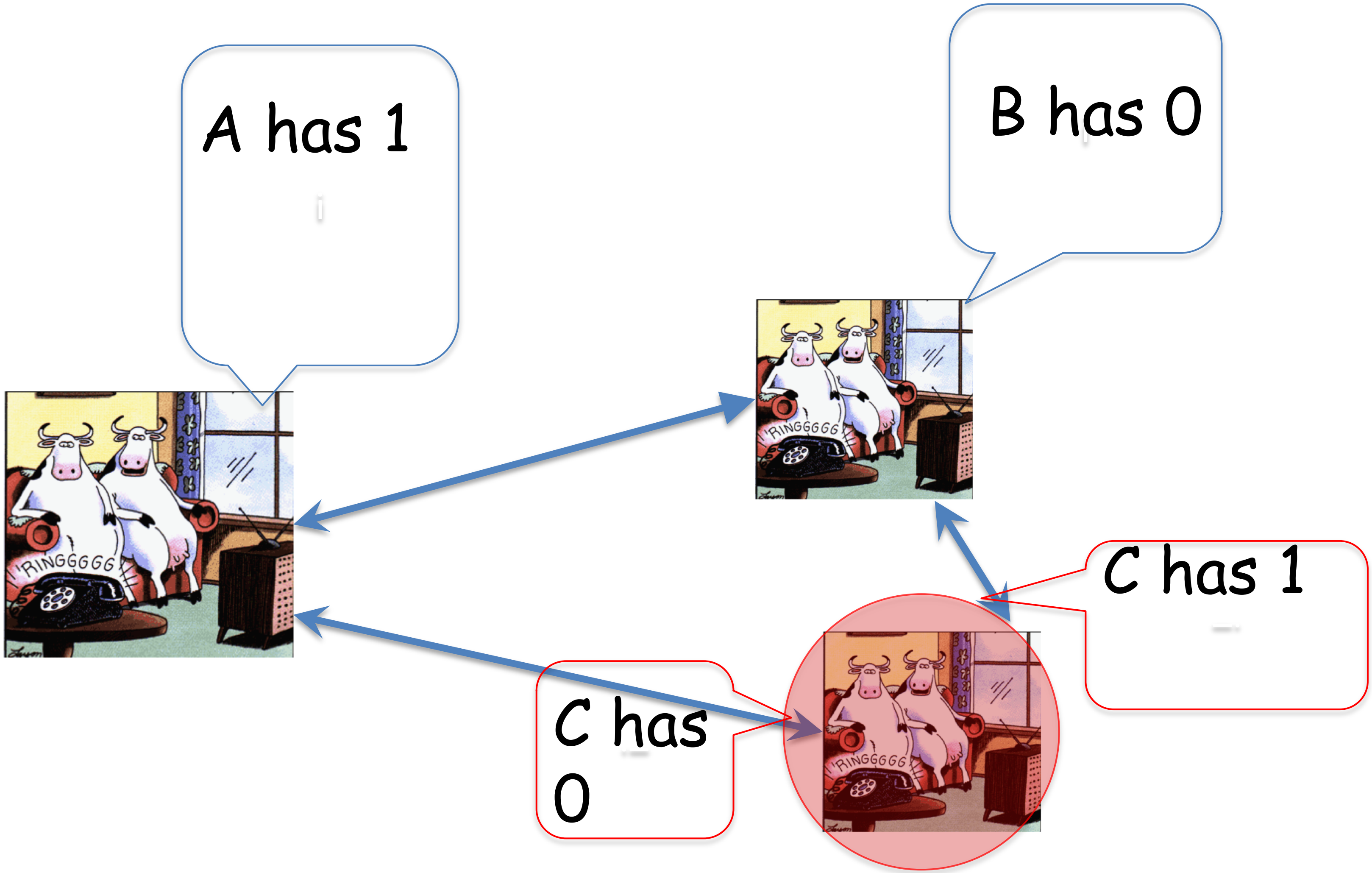
n nodes

t bad nodes behave arbitrarily

Worst case input



"The revolution has been postponed . . . We've discovered a leak."



Agreement Protocol: Vote and output majority.
Requires t< n/3
Without some signature scheme, A can’t prove what C sent to B (no “authentication”)

Synchronous model

- Proceeds in rounds: Time=number of rounds
- Round: A) All nodes send messages

  B) All nodes receive all messages sent

There is a deterministic algorithm that takes $t+1$ rounds and this is the best possible, even in the authenticated setting.

Works by detecting bad nodes.

The asynchronous model

Adversary schedules message delivery, no global clock

→ At any step, a node must act before hearing from all n-t nodes and t of these nodes which send may be bad

How do you measure time?

- Initial step when all or some nodes may send messages, then event-driven:
- each node waits for an event before acting
- Time = length of longest chain of events where each event depends on the previous one occurring or equivalently
- Time= # of maximum time units where the max time to send a message from one node to another takes 1 time unit

Famous impossibility result

Crash fault: A node dies.

In the worst case,

ONE crash fault makes (deterministic) agreement impossible with asynchrony.

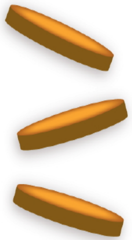
(1982: Fischer, Lynch and Patterson)

2007 Nancy Lynch wins the Knuth Prize for lifetime achievement, with this result called "fundamental in all of computer science".

Randomness, time and messages

- Can be used to save time and communication
- In the asynchronous model, it's necessary

What kind of randomness?



- A random bit "global coin" known to all

OR

- "private coinflips": Each node has access to its own random bits which are generated as needed

Randomness and the power of the adversary

"adversary" ==worst case faults

Using randomness: coinflips are made during the algorithm
Adversary does NOT know their outcomes until they are flipped

- Can the adversary wait to see the coinflips before choosing whom to corrupt?
- Then it is an "adaptive adversary"
- Else it is "static"

Randomness and the power of the adversary

- Can the adversary wait to see the coinflips before choosing whom to corrupt?
- Then it is an "adaptive adversary"
- Else it is "static"

With the static version, the algorithm can elect a leader which decides.

Outline for tutorial

Part I

- Rabin’s global coin alg
- Ben-Or’s with private coins
  - Reliable broadcast, multicast

Part II

- Averaging samplers

Global Coin Alg, t <n/8 (synch version of Rabin)

Repeat

- Each node sends its bit to all
  - maj <- majority bit received,
  - tally <-number of maj bits received
- If global coin = heads, threshold <- L=5n/8

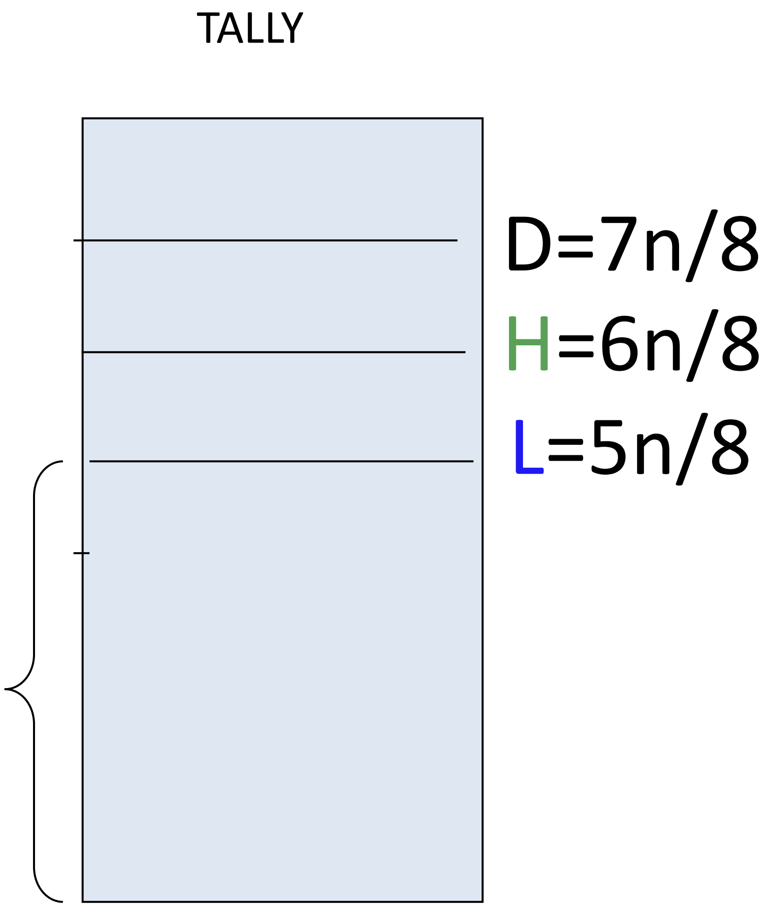
  Else threshold <- H= 6n/8

  tally >= threshold then vote <-- maj

  Else vote <- 0
- If tally >= D= 7n/8 then Decide maj

Why this works: 2 thresholds

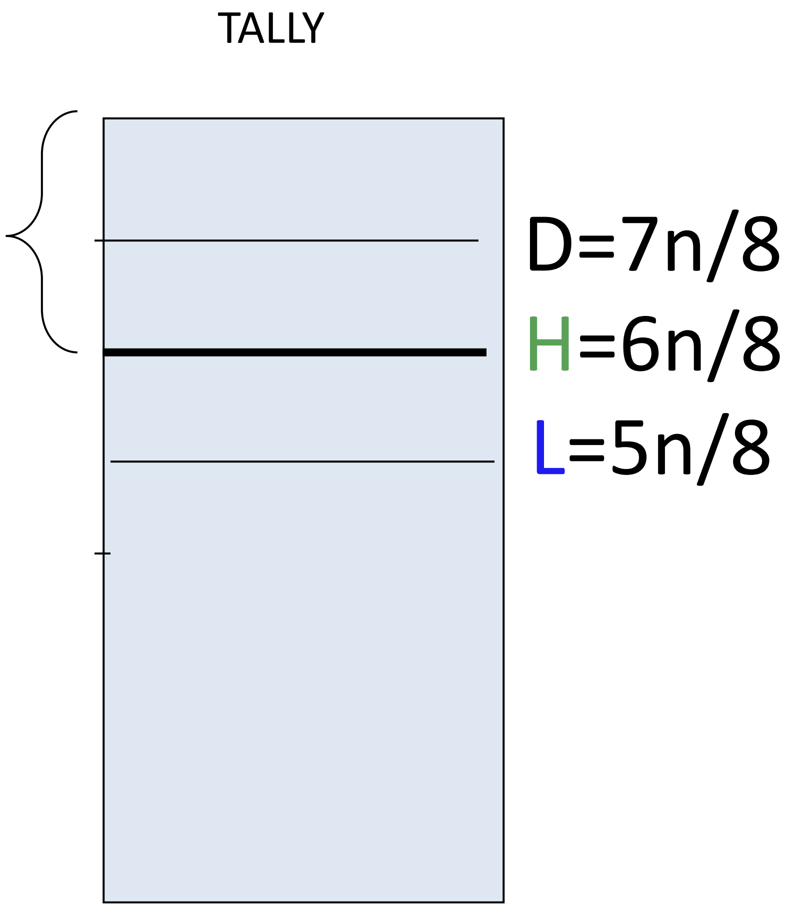
Adversary can only affect number received by t



If maj is not unique,
ALL < L so all set to 0
and decide next round

Why this works: 2 thresholds

Adversary can only affect number received by t



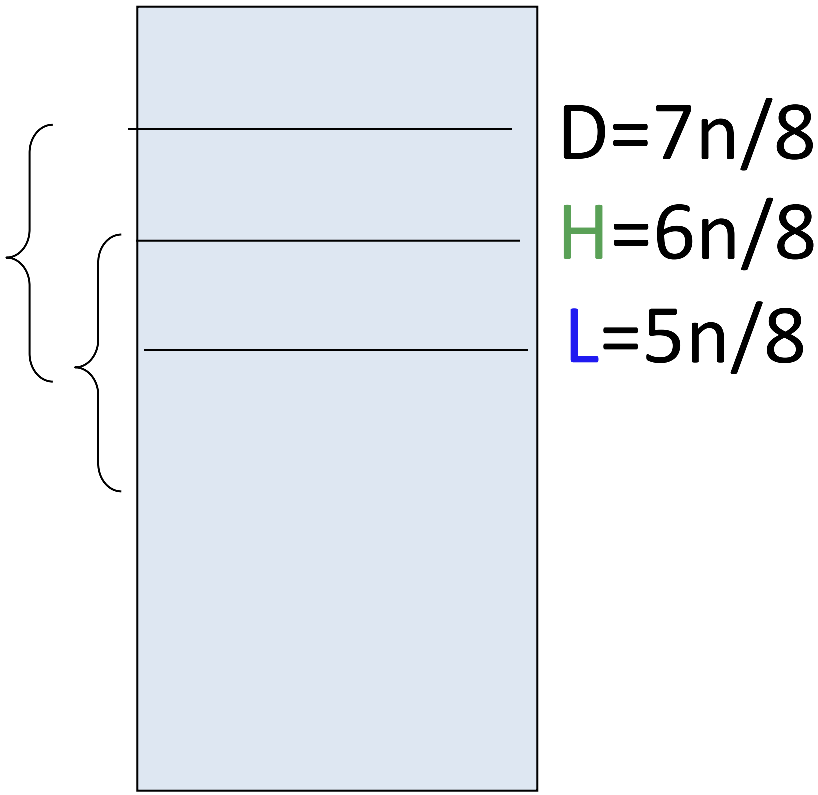
ALL> H, all set to maj

ALL decide current round or next

Why this works: 2 thresholds

Adversary can only affect number received by t

CASE: TALLY tiers



Otherwise, all nodes in two consecutive tiers.

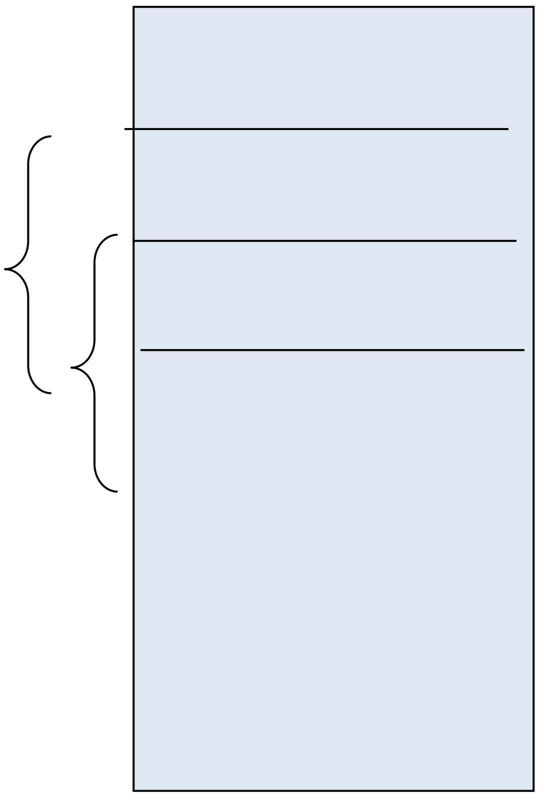
D>All > L: All keep maj if threshold is L

H > All: All set to 0 if threshold is H

Why this works: 2 thresholds

Adversary can only affect number received by t

CASE: TALLY tiers



D=7n/8

H=6n/8

L=5n/8

What if the threshold is NOT the right one?

No decision, repeat

Asynchronous with private coins

Ben-Or Byzantine Agreement $t<n/5$

$r=1$

While not decided each p repeats:

do <u>Broadcast</u> of vote b_p

$v \leftarrow$ majority value

tally $\leftarrow$ size of majority

CASE: tally

A) $> (n+t)/2$ then Decides on v

B) $> t$ then $b_p \leftarrow v$

C) else $b_p \leftarrow$ private coinflip

Increment r

Broadcast (p)

- Sends ($b_{p,}$ r) to all
- Waits until votes for round r received from *n-t*
  - *Can only wait this long or alg may stall*

- If > *(n+t)/2* of same vote v received, then sends (echo,v,r) to all
  - *Ensures >half good nodes had same value so only 1 such v*
  - Else sends (echo, nil, r) to all

- Waits until n-t (echo,*, r) messages received

Analyzing Ben-Or Byzantine Agreement $t<n/5$

While not decided each p repeats:

do <u>Broadcast</u> of vote b_p

$v \leftarrow$ majority value

tally $\leftarrow$ size of majority

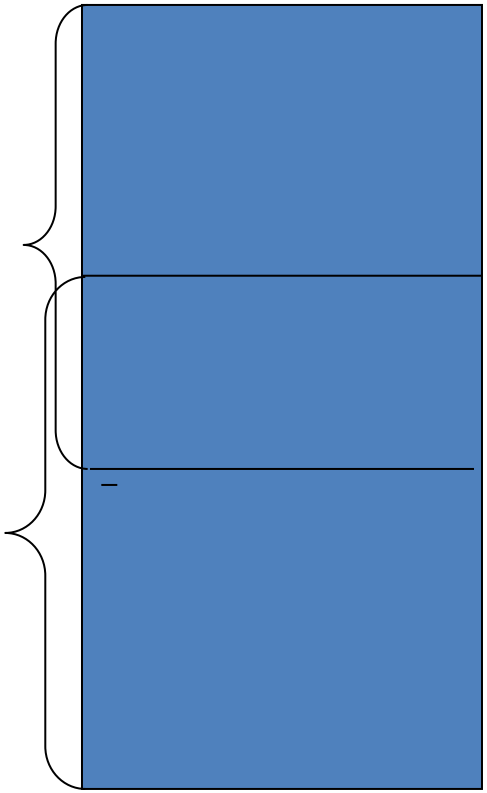
CASE: tally

A) $> (n+t)/2$ then Decides on v

B) $> t$ then $b_p \leftarrow v$

C) else $b_p \leftarrow$ private coinflip

Two thresholds

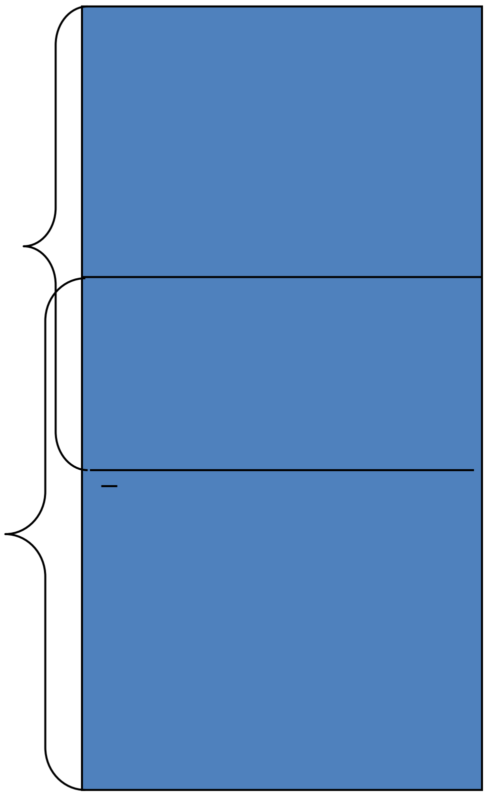


A Deciding point (all above maintaining pt)

B Maintaining point (only 1 value possible)

If tally of all nodes above A, they decide, and because of property of echoes, they decide on same value

Two thresholds



A Deciding point (all above maintaining pt)

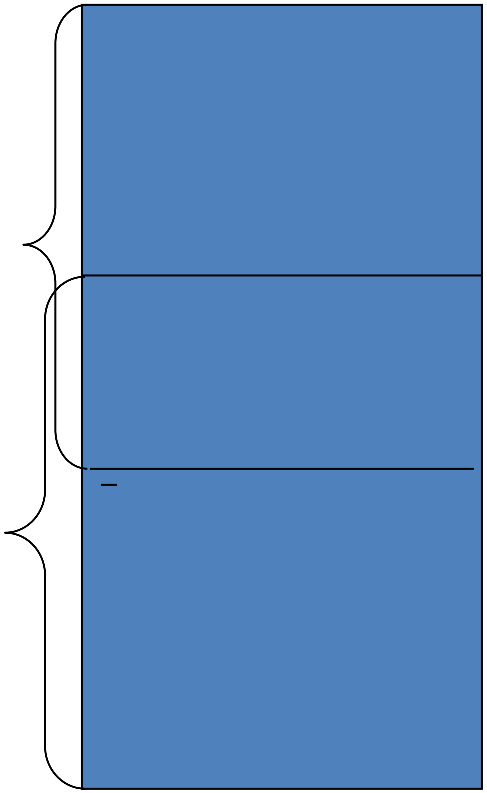
B Maintaining point (only 1 value possible)

If one node decides--> tally> 2t+1

--> tally > t+1 for all nodes

--> All hold same vote, all decide next round

Two thresholds



A Deciding point (all above maintaining pt)

B Maintaining point (only 1 value possible)

If there is no tally above A, then some nodes may be in CASE C

Decision occurs if coin flips <u>all</u> agree and they agree with bits held by nodes in CASE B

Observe 1:

Ben-Or’s iterations can be repeated until private coins agree with each other and with the maintained bit.

Ends when 4n/5 good nodes hold the same value

Bracha improves this to 2n/3, and $t<n/3$ by having nodes by a verification routine that enures nodes act consistently (or are detected).

Observe 2:

For $t \le \sqrt{n}/4$ then w/const prob it works the first time:

Let X be the #heads-#tails when n coins are tossed, normal distribution with

$\sigma^2 = \sum(\mathrm{E}[X^2_i] - E[X_i]^2) = n(1/2) - n(1/4) = n/4$

$\sigma = \sqrt{n}/2 = 2t$

$\Pr(X > 2t) =$

If

#heads - #tails > 2t or. #tails - #heads > 2t

→ Adv can't affect majority value →1/2 prob. of fair coin

Reliable Broadcast (Bracha)

A node p broadcasts a message m to all other nodes. If if $t<n/3$

- If all nodes start with the same bit, all decide the same bit within 3 steps
- If any good node decides on a bit, all nodes will decide the same bit.

Bracha's Reliable Broadcast

{p a node, m message}

1. p sends (*init*, m) to all nodes
2. Upon receiving (*init*, m) from n-t other nodes,
3. Send (*echo*, m) to all nodes
4. Upon receiving (n+t)/2 (*echo*, m) or t+1 (*ready*,m)
5. Send (*ready*, m) to all nodes
6. Upon receiving n-t (*ready*, m), decide m

CASE: Suppose good nodes start with a 1

{p a node, m message}

1. p sends (*init*, m) to all nodes
2. Upon receiving *(init*, m) from n-t other nodes,
3. Send (*echo*, m) to all nodes
4. Upon receiving (n+t)/2 (*echo*, m) or t+1 (*ready*,m)
5. Send *(ready*, m) to all nodes
6. Upon receiving n-t (*ready*, m), decide m

All n-t good nodes receive and send

{p a node, m message}

1. p sends (*init*, m) to all nodes
2. Upon receiving (*init*, m)
3. Send (*echo*, m) to all nodes
4. Upon receiving (n+t)/2 (*echo*, m) or t+1 (*ready*,m)
5. Send (*ready*, m) to all nodes
6. Upon receiving n-t (*ready*, m), decide m

All n-t good nodes receive and send

All good nodes receive echoes and send

{p a node, m message}

1. p sends (*init*, m) to all nodes
2. Upon receiving (*init*, m)
3. Send (*echo*, m) to all nodes
4. Upon receiving (n+t)/2 (*echo*, m) or t+1 (*ready*,m)
5. Send (*ready*, m) to all nodes
6. Upon receiving n-t (*ready*, m), decide m

All n-t good nodes receive and send

All good nodes receive echoes and send

All good nodes receive ready and decide

{p a node, m message}

1. p sends (*init*, m) to all nodes
2. Upon receiving (*init*, m)
3. Send (*echo*, m) to all nodes
4. Upon receiving (n+t)/2 (*echo*, m) or t+1 (*ready*,m)
5. Send (*ready*, m) to all nodes
6. Upon receiving 2t+1 (*ready*, m), decide m

CASE: Suppose one good node decides m

t+1 good nodes send ready

{p a node, m message}

1. p sends (*init*, m) to all nodes
2. Upon receiving (*init*, m)
3. Send (*echo*, m) to all nodes
4. Upon receiving (n+t)/2 (*echo*, m) or t+1 (*ready*,m)
5. Send (*ready*, m) to all nodes
6. Upon receiving 2t +1 (*ready*, m), decide m

’

all good nodes will send ready, all decide

t+1 good nodes sent ready

1. {p a node, m message}
2. p sends (*init*, m) to all nodes
3. Upon receiving (*init*, m)
4. Send (*echo*, m) to all nodes
5. Upon receiving n-t (*echo*, m) or t+1 (*ready*,m)
6. Send (*ready*, m) to all nodes
7. Upon receiving 2t +1 (*ready*, m), decide m

Ready messages sent by a good node only if majority of good nodes agree on echo message, can't have two different values

{p a node, m message}

1. p sends (*init*, m) to all nodes
2. Upon receiving (*init*, m)
3. Send (*echo*, m) to all nodes
4. Upon receiving (n+t)/2 (*echo*, m) or t+1 (*ready*,m)
5. Send (*ready*, m) to all nodes
6. Upon receiving 2t +1 (*ready*, m), decide m

Properties of:
Reliable Broadcast

if $t<n/3$

- If all nodes start with the same bit, all decide the same bit within 3 steps
- If any good node decides on a bit, all nodes will decide the same bit.

Multicast (Ran, Ben-Or)

Each node p inputs a bit. All nodes decide on the same subset of at least n-t bits

Remaining bits are ambiguous (nil or correct)

P1	**P2**	**P3**	…		**Pn**
1	1 0	0		-	nil

n-t

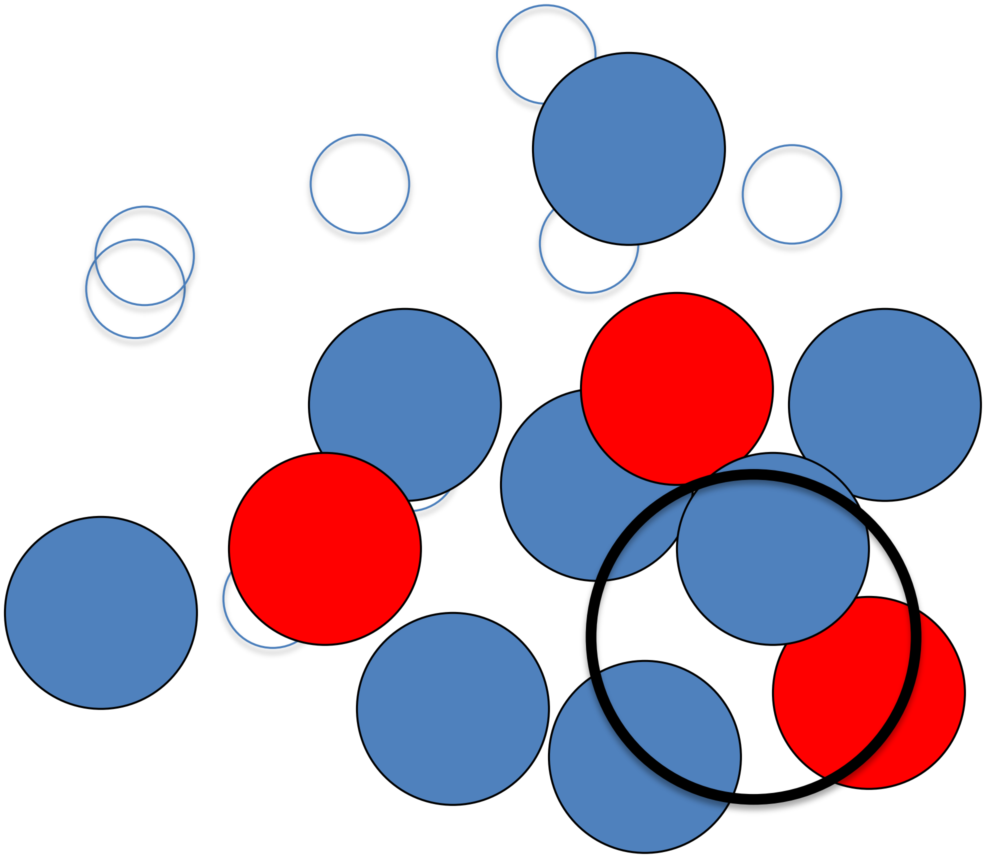
Implementing multicast

- Each node uses Reliable Broadcast in parallel to send P1

  their bit and waits until it decides at least n- t
bits
- Spread: uses Reliable Broadcast to broadcast the subset of bits decided
- Fill in missing bits which appear in t+1 decided subsets

Part II

Randomness for choosing representative committees



U=Set of all nodes

S is θ-representative of U, |U| if

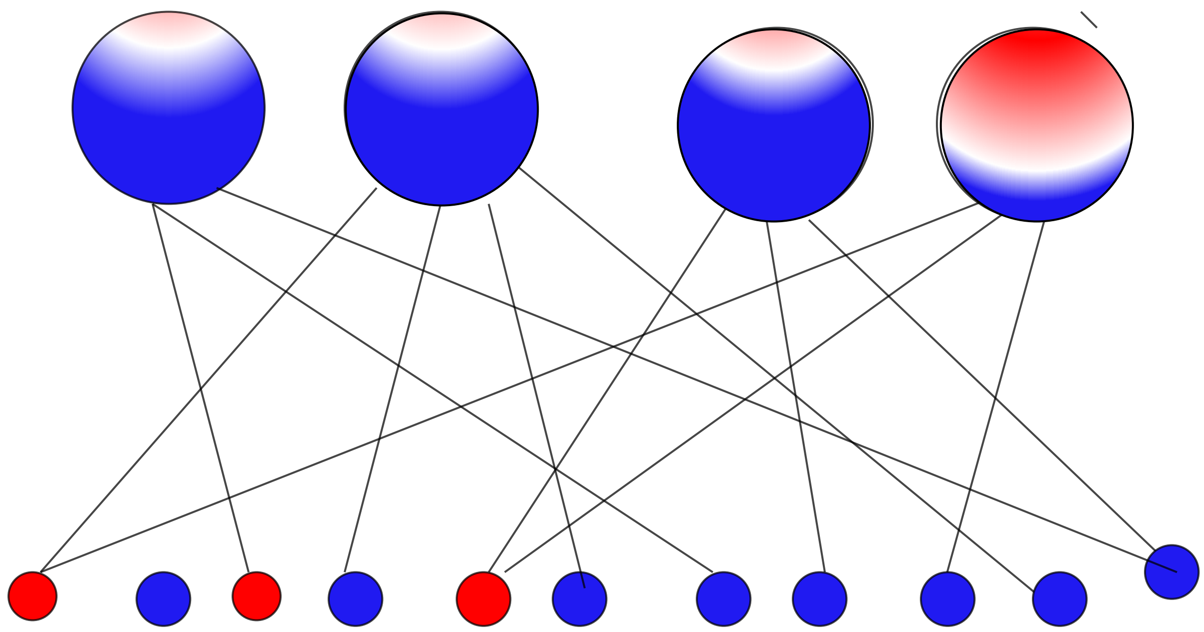
$|BAD \cap S|/|S| <$

$|BAD \cap U|/|U| + \theta$

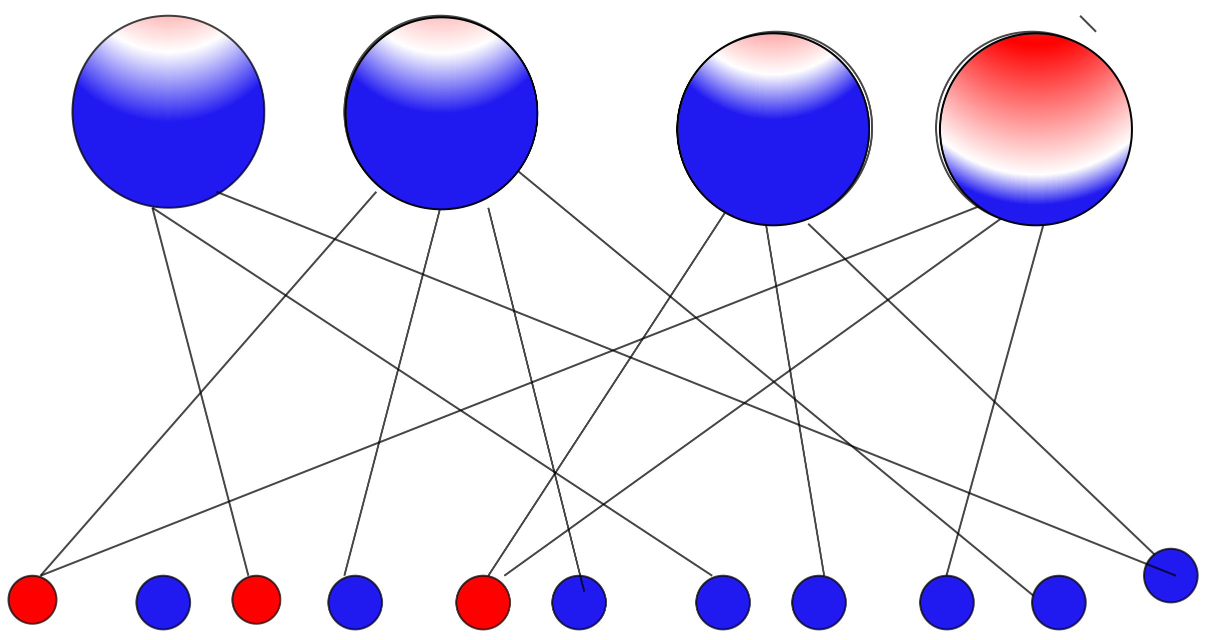
A set of mostly representative committees can be built deterministically:

averaging sampler, extractor, disperser, Bracha committee

|U|=n, 1-1/log n fraction of committees are representative, for <u>ANY</u> subset of BAD nodes



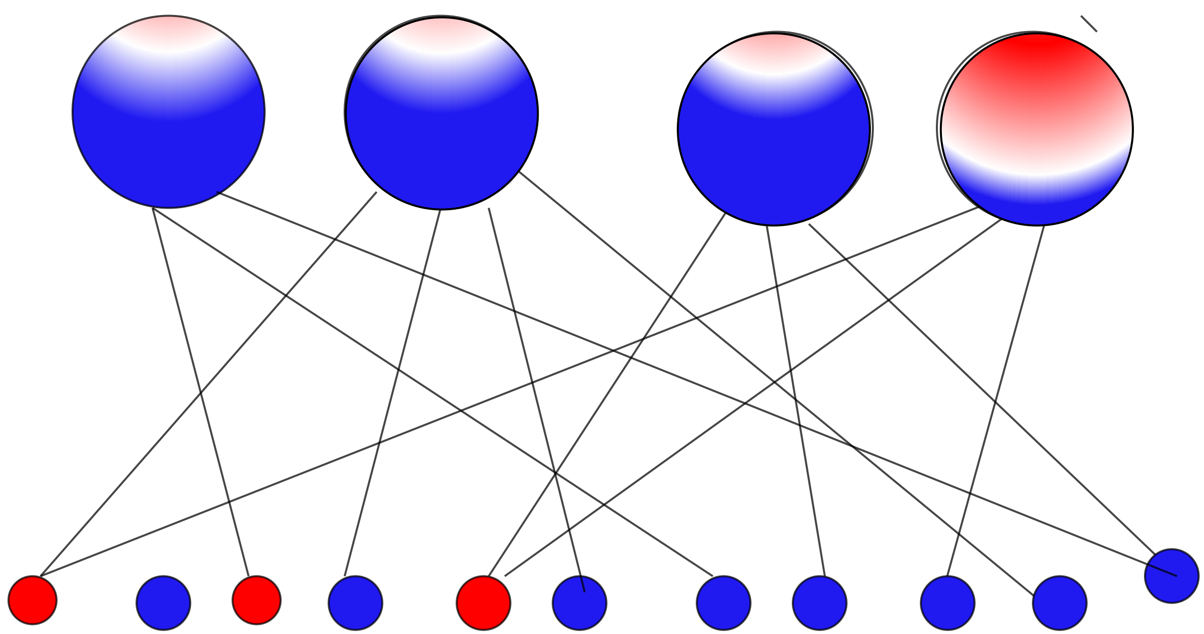
G is a (θ,δ) sampler if no more than δ fraction of committees are θ-representative, for <u>ANY</u> subset of BAD nodes (Zuckerman)



G is a $(\theta\delta,)$ sampler if no more than δ fraction of committees are θ-representative, for ANY subset of BAD nodes

Proof: Let d be the size of the committee, r be the number of committees

If d=O(log $(1/\delta)/\theta^2$ and r>n/δ, there is a sampler w.h.p.



Probabilistic method

.

To show there exists a graph with a set of properties e_1, e_2, …,e_k

- Show that the probability that any of these properties fail to occur is < 1
  by taking a union bound

- $Pr(e_1)+Pr(e_2) +\ldots+Pr(e_k) < 1$

Proving existence of sampler

Fix a set of bad nodes B, fix a set of $r\delta$> n non-representative committees C'

X be the number of edges from $r\delta$ committees to bad nodes.

X =sum of $r\delta$ d independent coin flips X_i =1 w/ prob =|B|/n, else 0

E[X]= $r\delta$ d(|B|/n)

By a Chernoff-Hoeffding bound, for any a>0, n independent coinflips in (0,1)

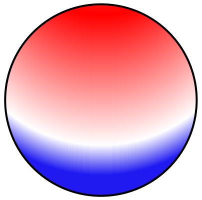
Pr(X >E[X])+ a) $\leqq$ exp (-2a^2 /n)

Here, n= |C'|*d= $r\delta$ d, a= θ ($r\delta$ d)

= exp(-2θ^2 $r\delta$ d)

Thus, Pr(C' are all unrepresentative) $\leqq$ exp (- $r\delta$ d θ^2/2)

Proving existence of a sampler



Now we don't want this to property to hold

For <u>any</u> Bad set B

For <u>any</u> subset of committees of size > delta

So taking the union bound over all possible such sets,

There are $< 2^n$ bad sets

And $\binom{r}{r\delta} < \left(\frac{e}{\delta}\right)^{r\delta}$ possible subsets of committees

Taking the union bound

$< (2)^n \left(\frac{e}{\delta}\right)^{r\delta}$. * $\exp((-\theta^2 r\delta d/2)$

Let $d=(4 \ln 2) (\log (1/\delta)/ \theta^2$ recalling $r\delta > n$

$=\exp(n \ln 2 + r\delta \ln\left(\frac{1}{\delta}\right) - r\delta (4 \ln 2) \ln\left(\frac{1}{\delta}\right) /2) <0$

Therefore there is some G which has no C'

References

- Introduction of the problem and the impossibility result (Pease, Shostak, Lamport 1980)
- Deterministic synchronous BA with poly(n) messages and optimal time(Garay and Moses)
- Rabin's global coin flip alg (from Motwani and Raghavan "Randomized Algorithms" text)
- Samplers and randomness extraction, defined in Zuckerman, 1997

Thank you!

Questions?