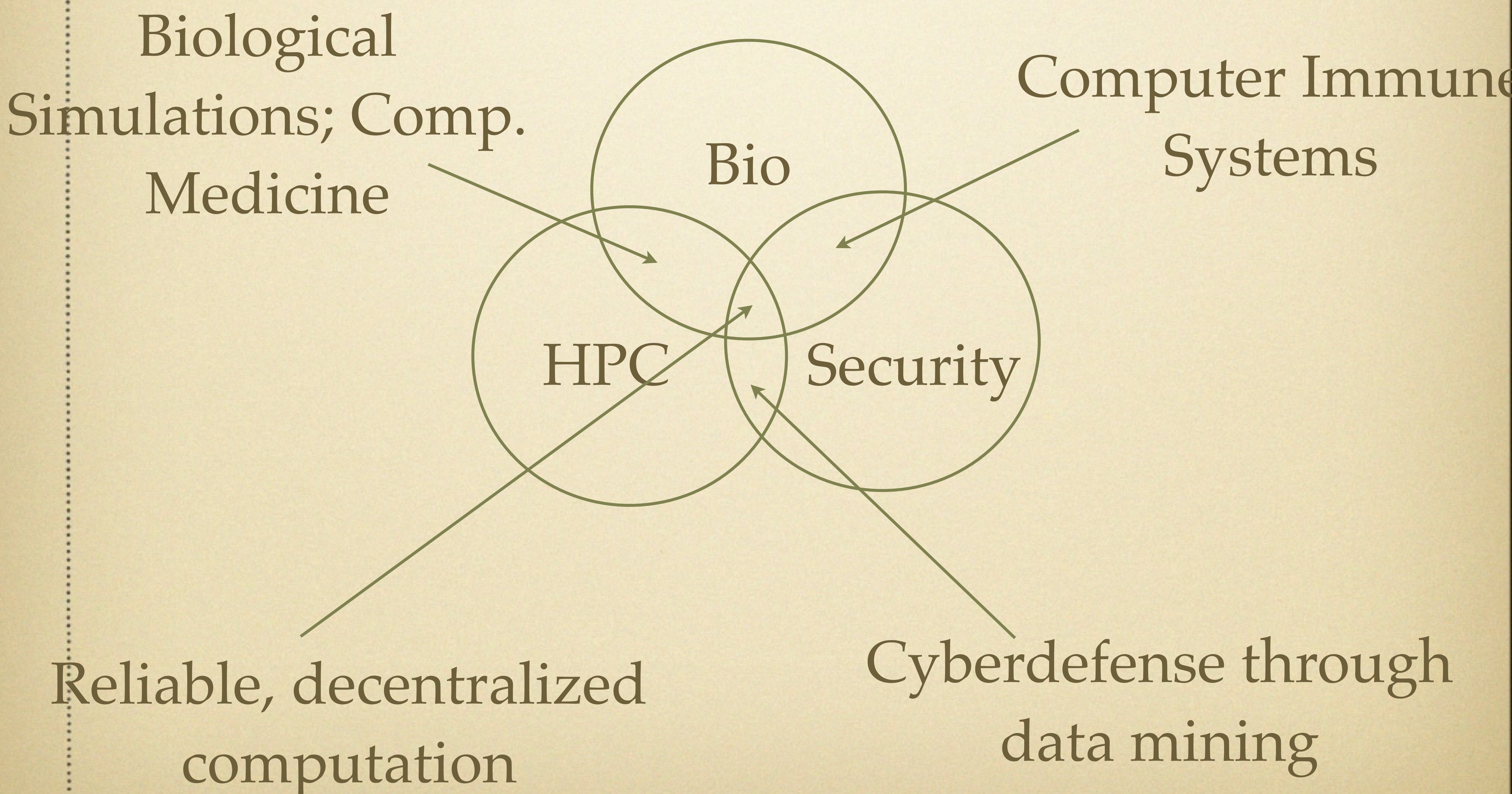
Byzantine Agreement in Polynomial Expected Time

Jared Saia

Joint with Valerie King

New Mexico

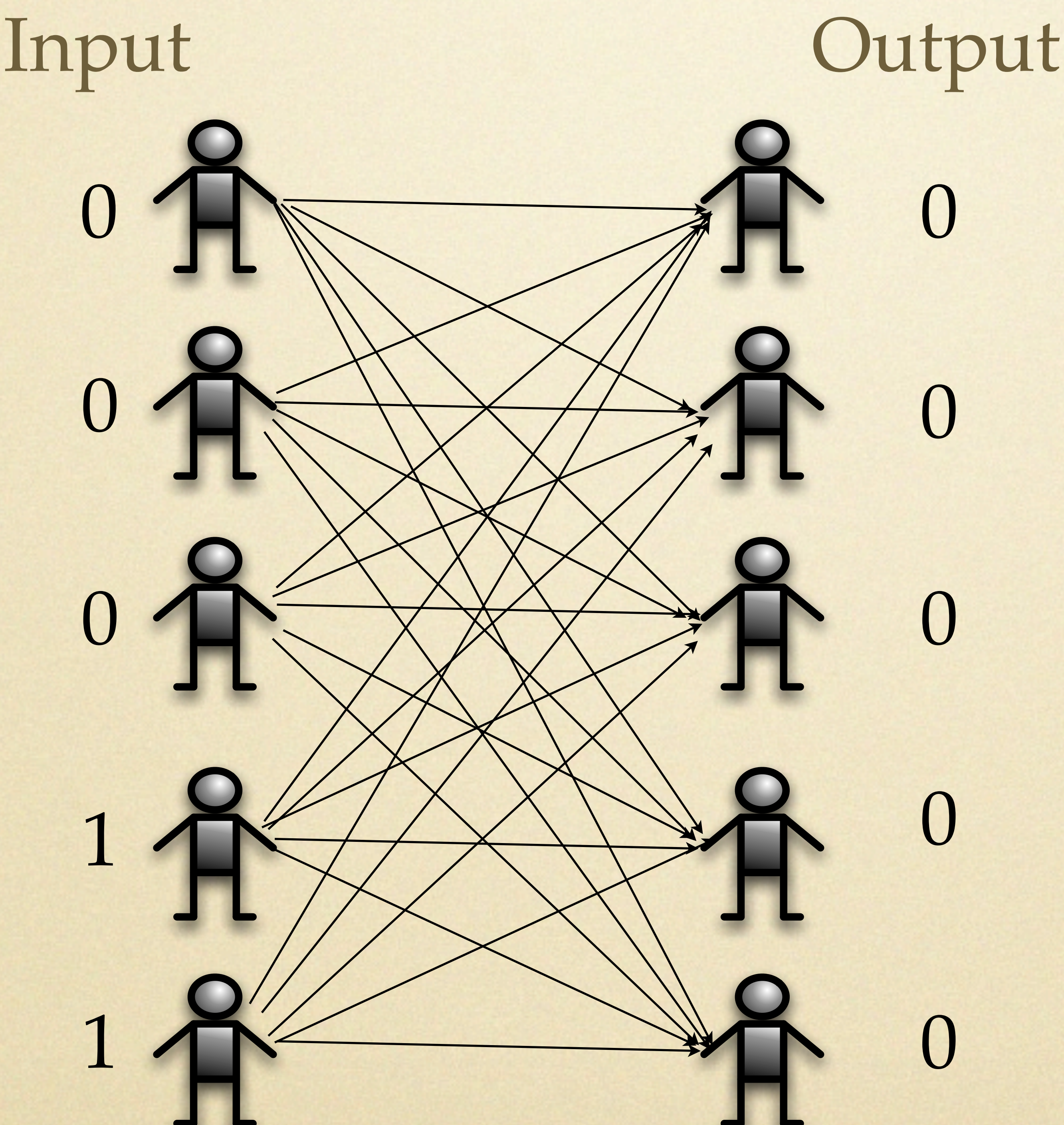
UNM CS Research



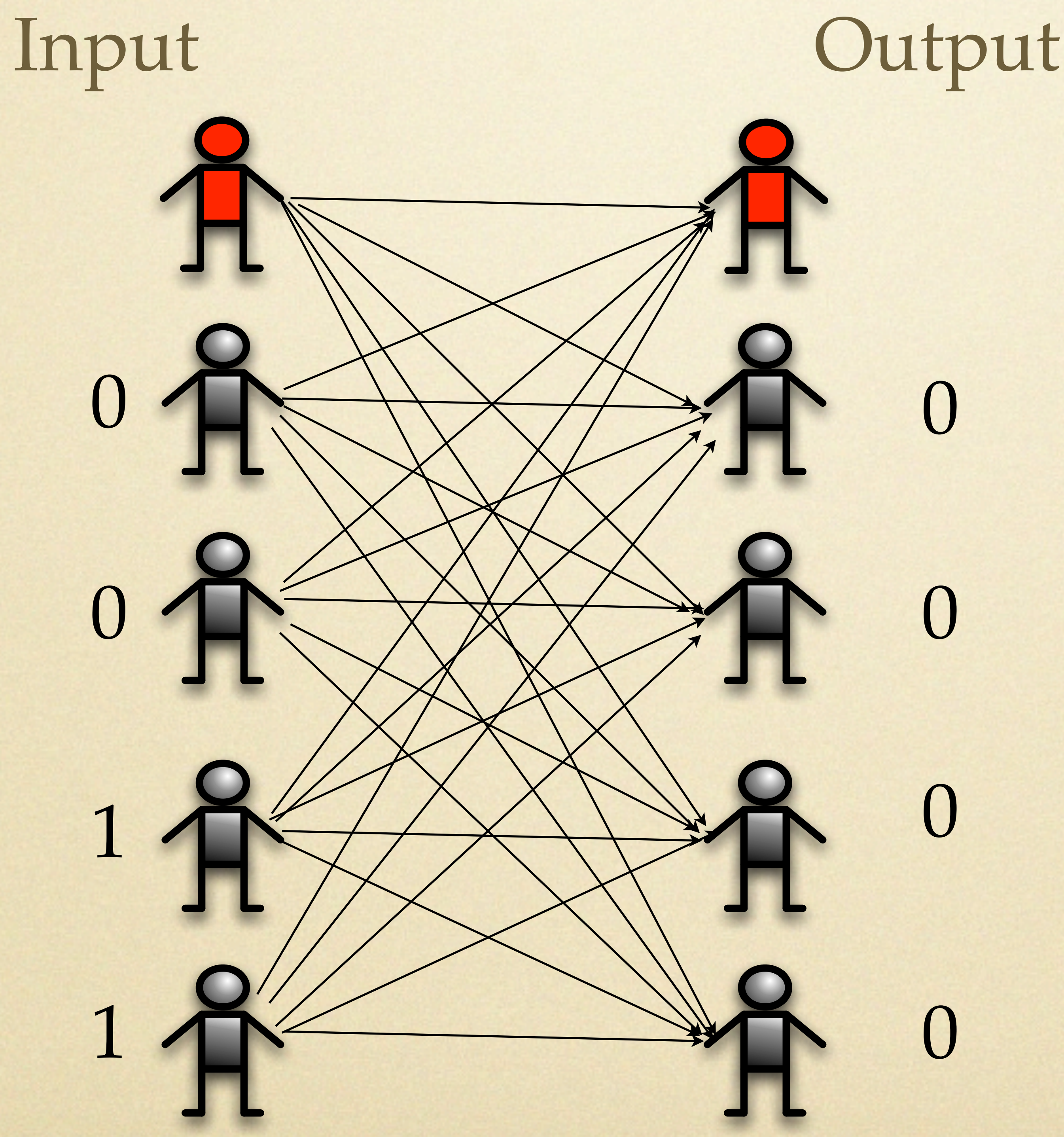
Group Decisions

- Periodically, components unite in a decision
- Idea: components vote. Problem: Who counts the votes?

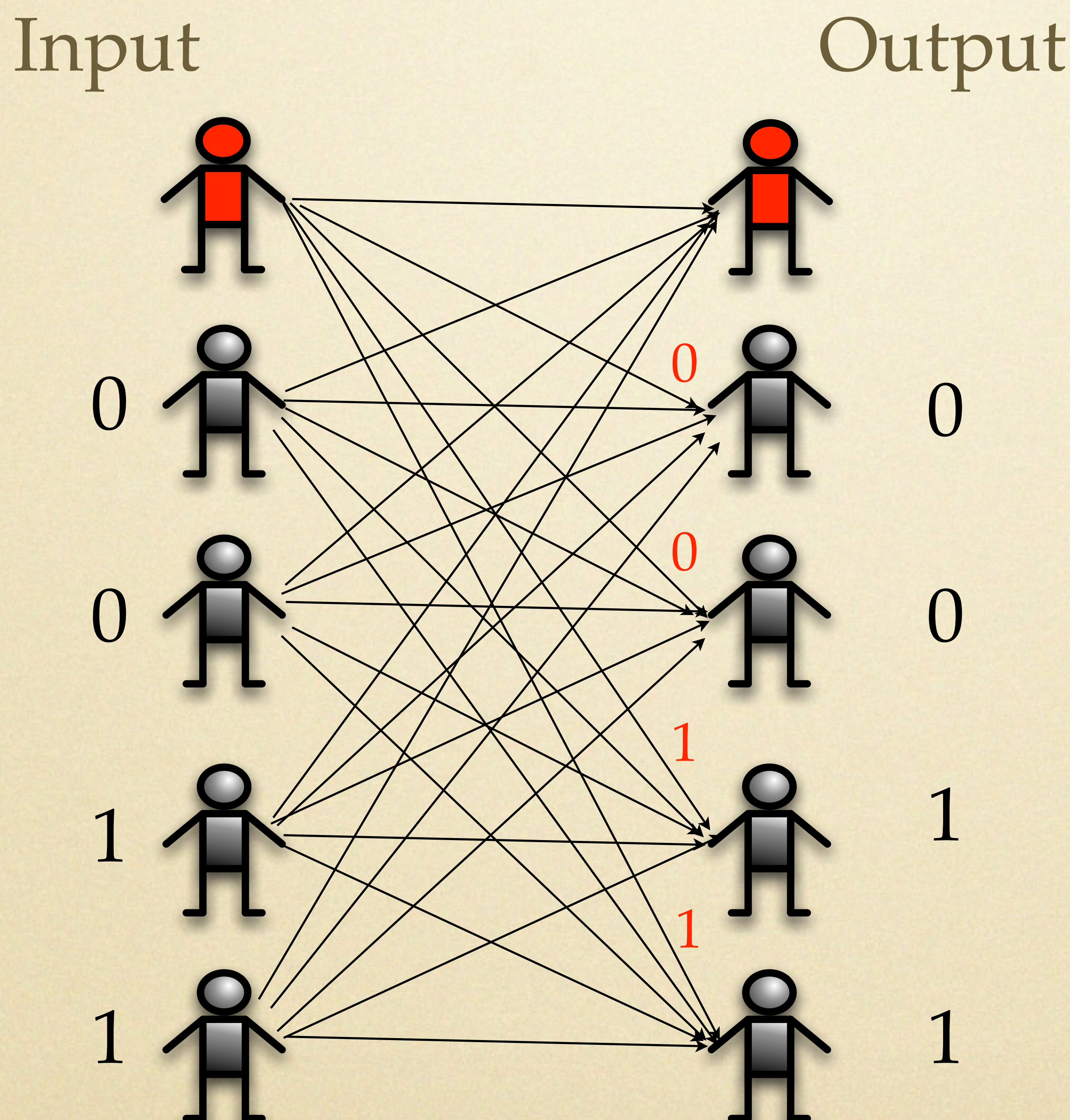
Idea: Majority Filtering



Idea: Majority Filtering



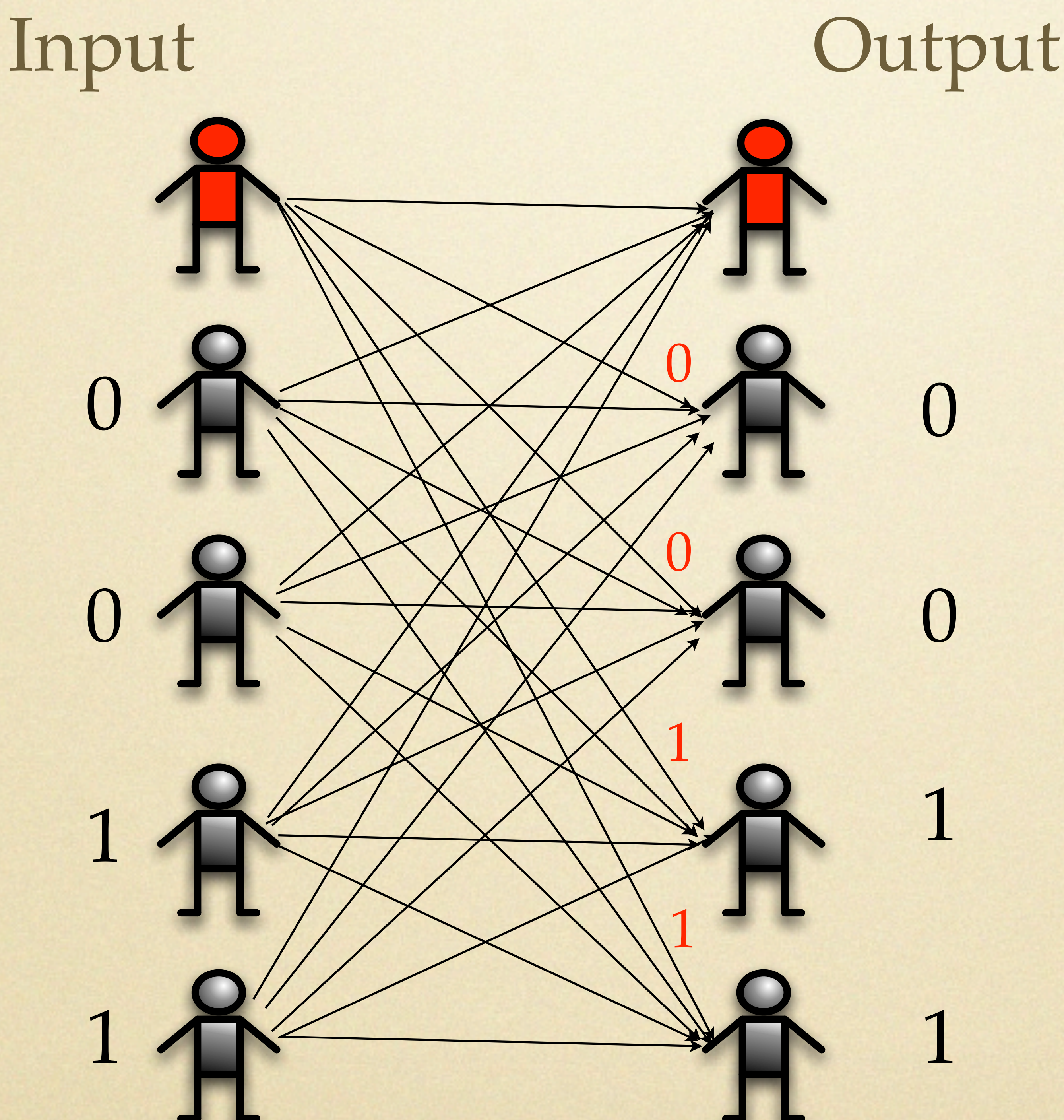
Problem



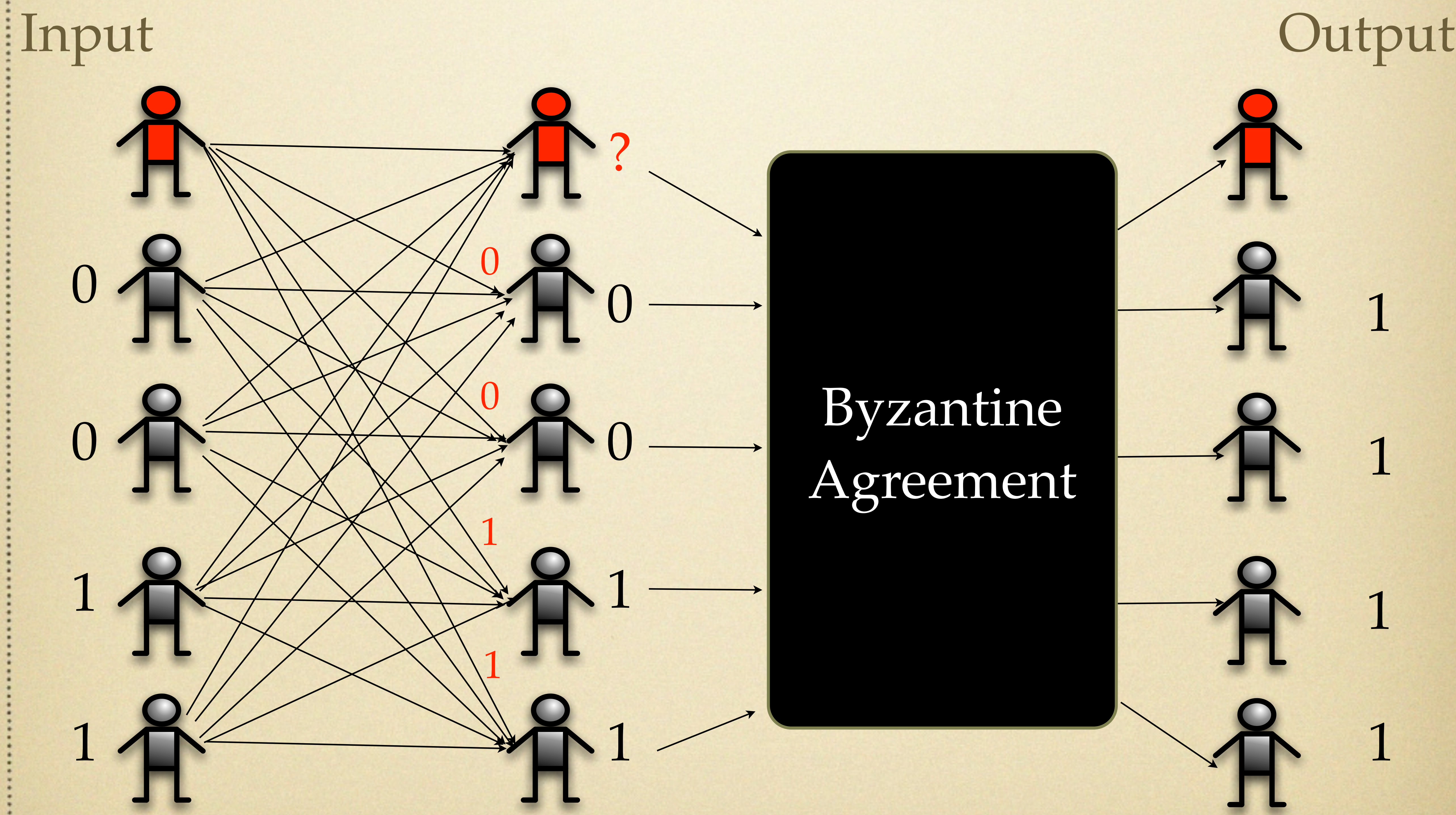
Byzantine Agreement

- Each processor starts with a bit
- Goal: 1) all good procs output the same bit; and 2) this bit equals an input bit of a good proc
- t = # bad procs controlled by an adversary

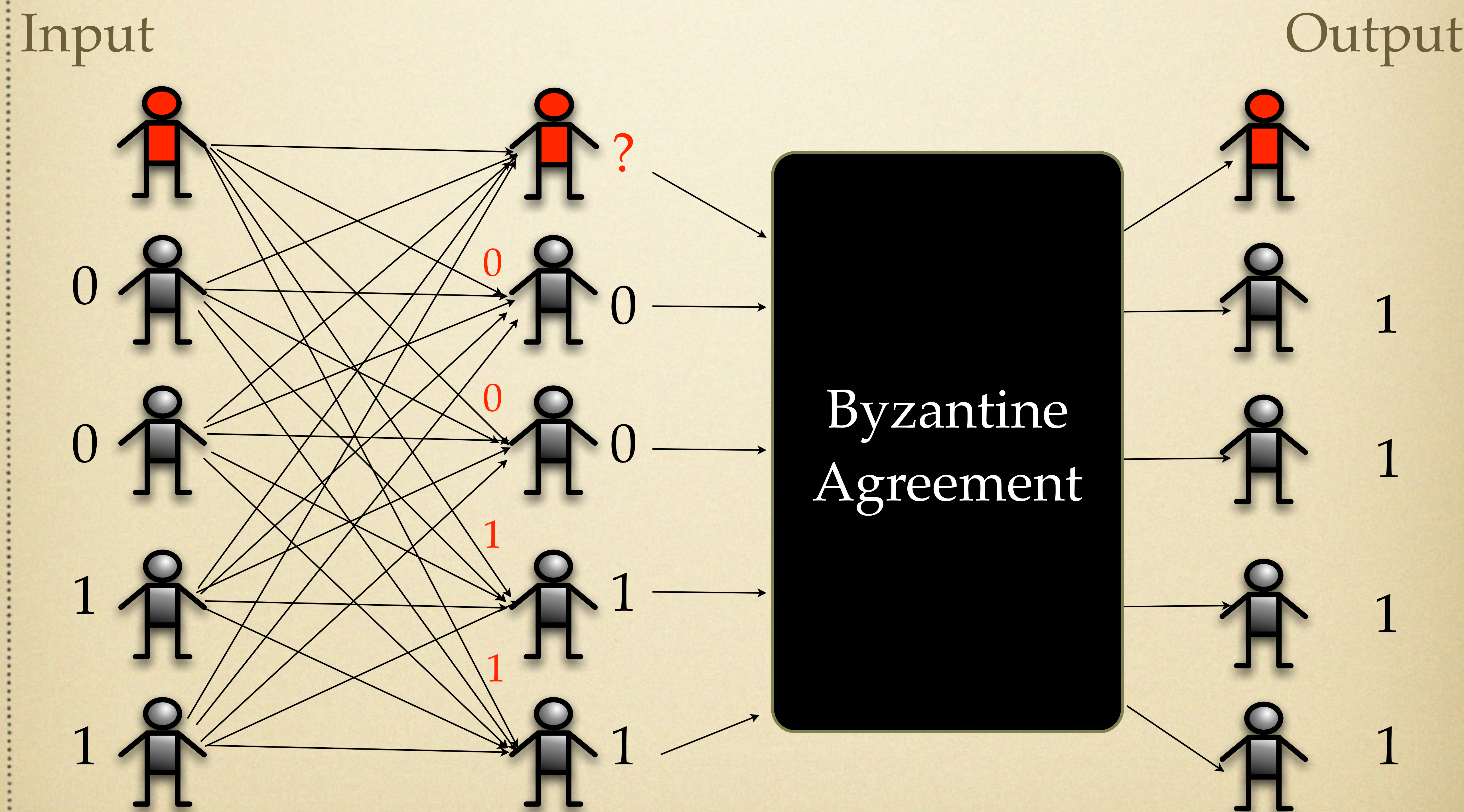
Problem



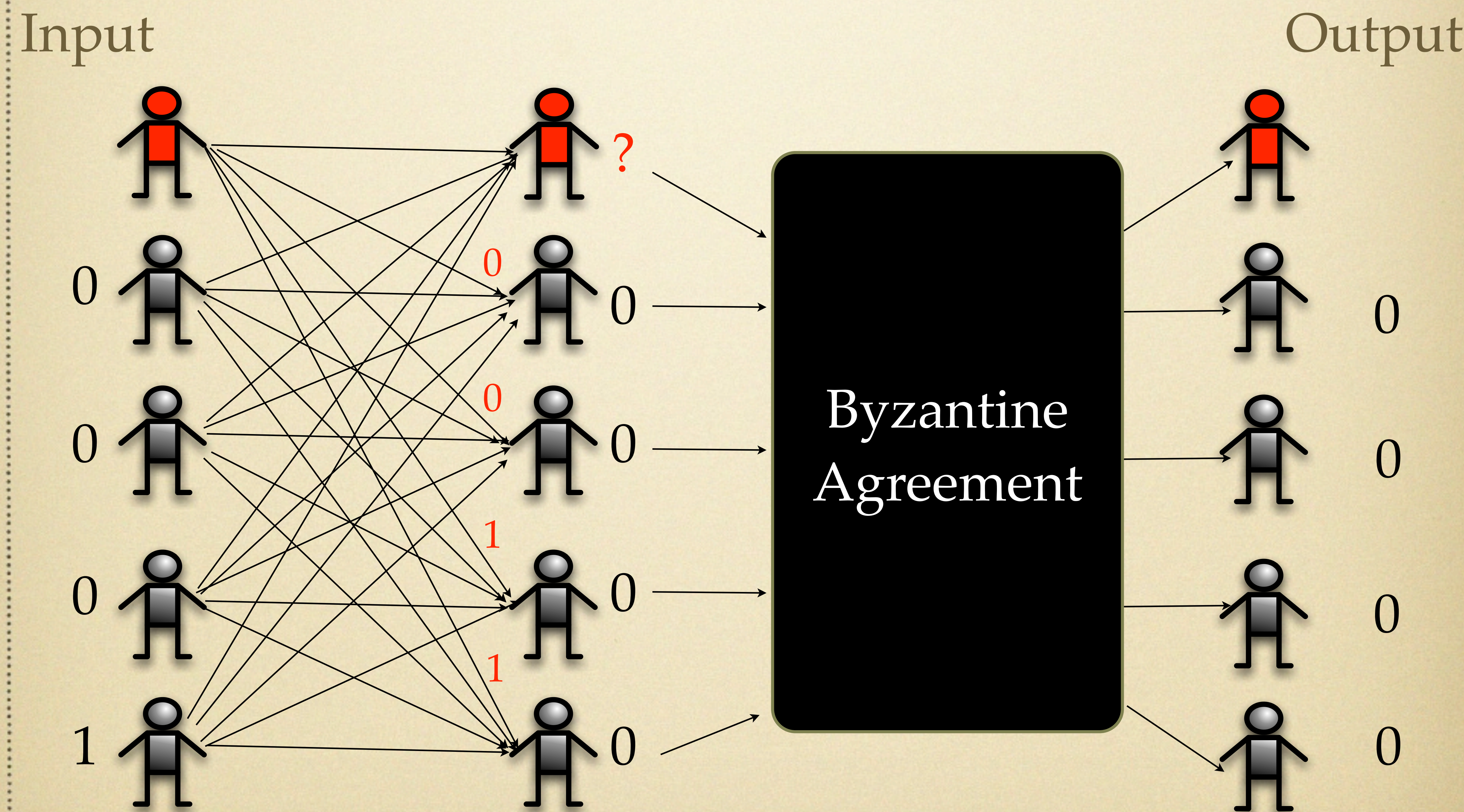
Idea



All good procs always output same bit



If majority bit held by >= 3 good procs, then all procs will output majority bit



Impossibility Result

- 1982: FLP show that 1 fault makes deterministic BA impossible in asynch model
- 2007: Nancy Lynch wins Knuth Prize for this result, called “fundamental in all of Computer Science”

Applications

- Peer-to-peer networks

  "These replicas cooperate with one another in a ***Byzantine agreement*** *protocol to choose the final commit order for updates." [KBCCEGGRWWWZ '00]*

- Rule Enforcement

  "... requiring the manager set to perform a ***Byzantine agreement protocol"*** *[NWD '03]*

- Game Theory (Mediators)

  "deep connections between implementing mediators and various agreement problems, such as ***Byzantine agreement****" [ADH '08]*

Applications

- Peer-to-peer networks

  "These replicas cooperate with one another in a ***Byzantine agreement*** *protocol to choose the final commit order for updates." [KBCCEGGRWWWZ '00]*

- Rule Enforcement

  "… requiring the manager set to perform a ***Byzantine agreement protocol"*** *[NWD '03]*

- Game Theory (Mediators)

  "deep connections between implementing mediators and various agreement problems, such as ***Byzantine agreement****" [ADH '08]*

- Also: Databases, State Machine Replication, Secure Multiparty Computation, Sensor Networks, Cloud Computing, Control systems, etc.

Model

- Public channels
- Asynchronous
- Unlimited messages for bad procs
- Adaptive adversary

Adv. takes over procs at any time, up to t total

Previous Work

- **Time** is defined to be the maximum length of any chain of messages
- In '83, Ben-Or described the first algorithm to solve BA in this model
- His algorithm requires expected exponential time
- [Ben-Or et al., '06] : ``*In the case of an asynchronous network, achieving even a polynomial rounds BA protocol is open."*

Previous Work

- **Time** is defined to be the maximum length of any chain of messages   Computation is instantaneous
- In '83, Ben-Or described the first algorithm to solve BA in this model
- His algorithm requires expected exponential time
- [Ben-Or et al., '06] : ``*In the case of an asynchronous network, achieving even a polynomial rounds BA protocol is open.*"

Our Result

- First algorithm that runs in expected polynomial time in this model
- Our algorithm runs in expected $O(n^{2.5})$ time

THEOREM 1. *Let n be the number of processors. There is a $t = \Theta(n)$ such that Byzantine Agreement can be solved in expected time $O(n^{2.5})$ and expected polynomial bits of communication, in the asynchronous message passing model with an adaptive, full-information adversary that controls up to t processors.*

BA with Global Coin, GC

Ben-Or's Algorithm

Send your vote to everyone

Let *fraction* be fraction of votes for majority bit

If *fraction* >= 2/3, set vote to majority bit; else set vote to GC

BA with Global Coin, GC

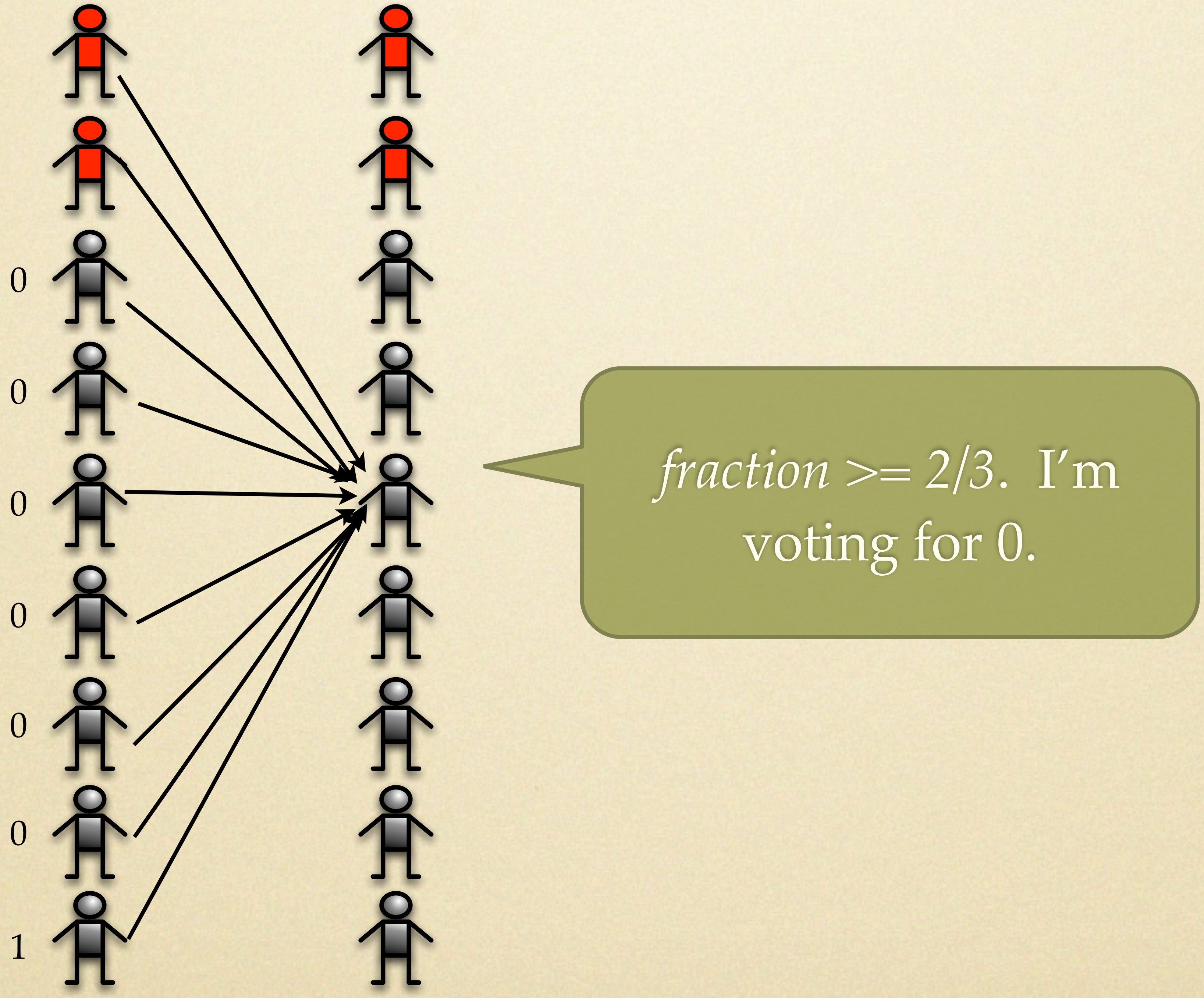
Ben-Or's Algorithm

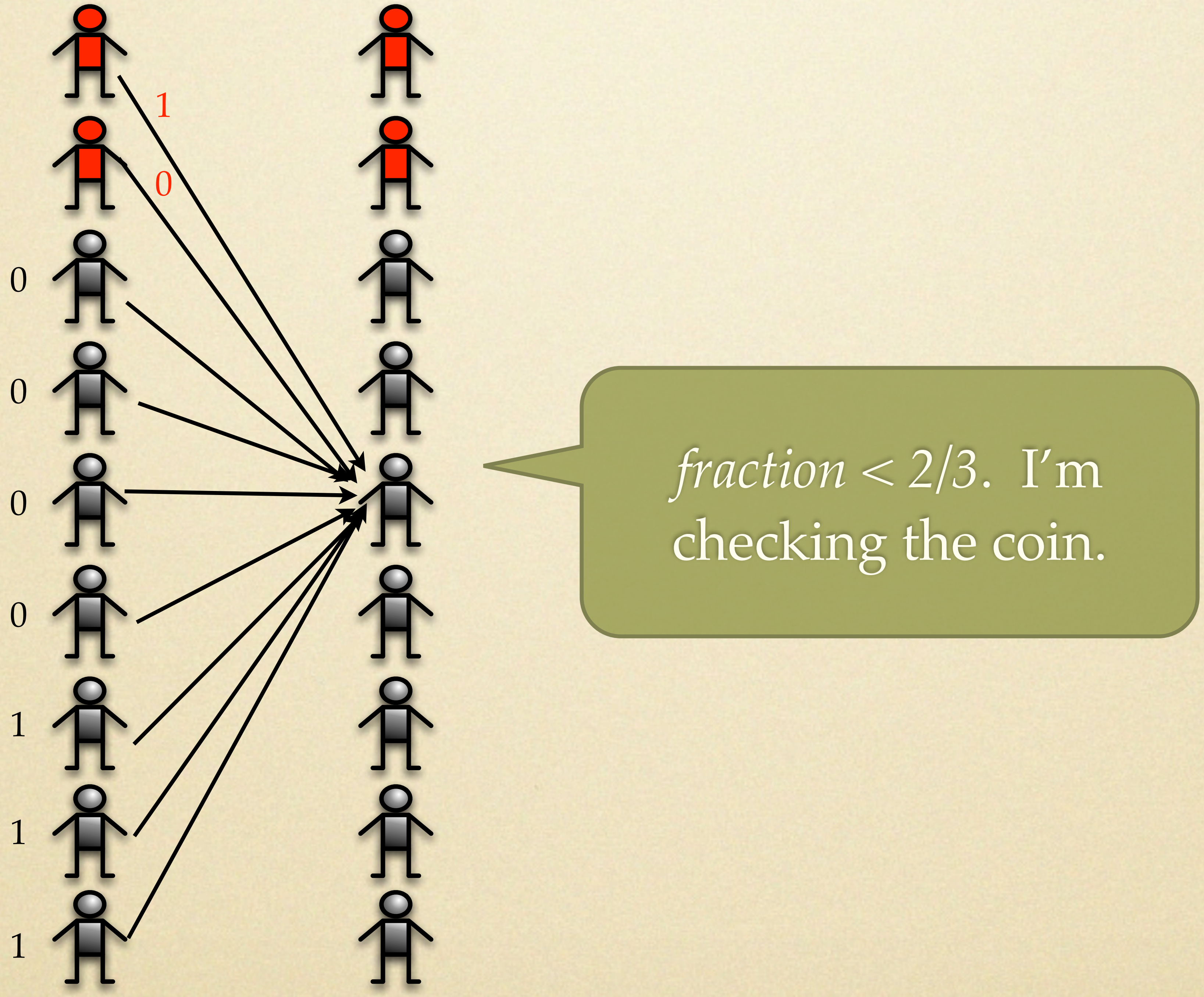
Set your vote to input bit

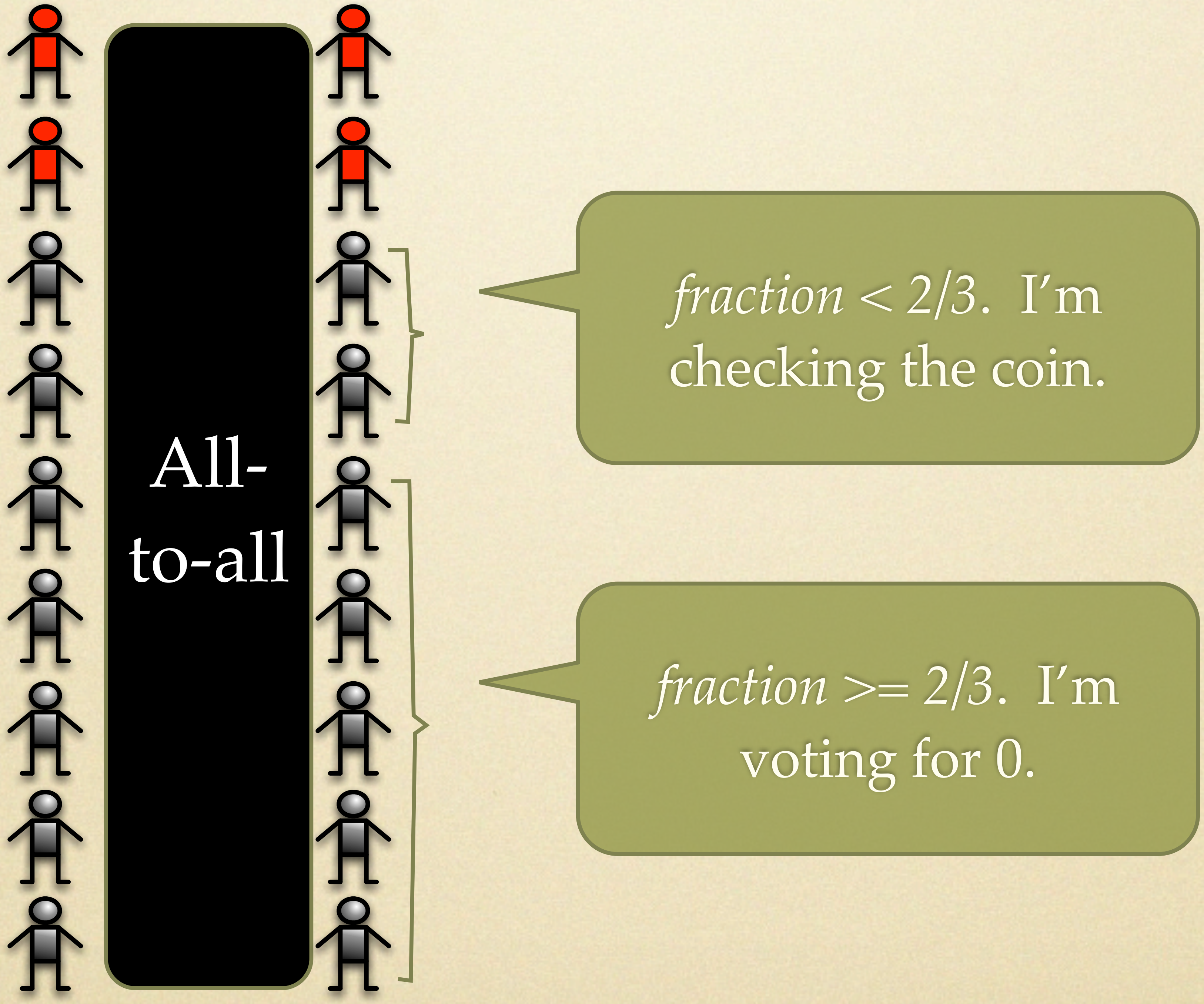
Repeat clogn times:

- Send your vote to everyone
- Let *fraction* be fraction of votes for majority bit
- If *fraction* >= 2/3, set vote to majority bit; else set vote to GC

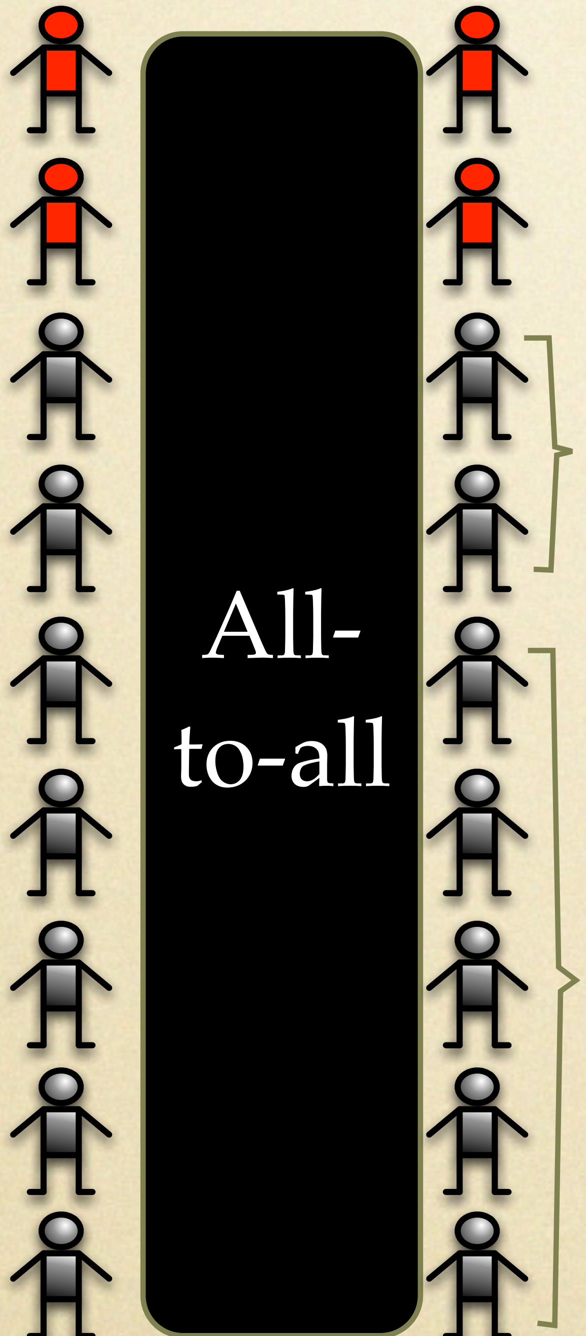
Output your vote


0
0
0
0
0
0
1
fraction >= 2/3. I'm voting for 0.


1
0
0
0
0
0
1
1
1
fraction < 2/3. I'm checking the coin.

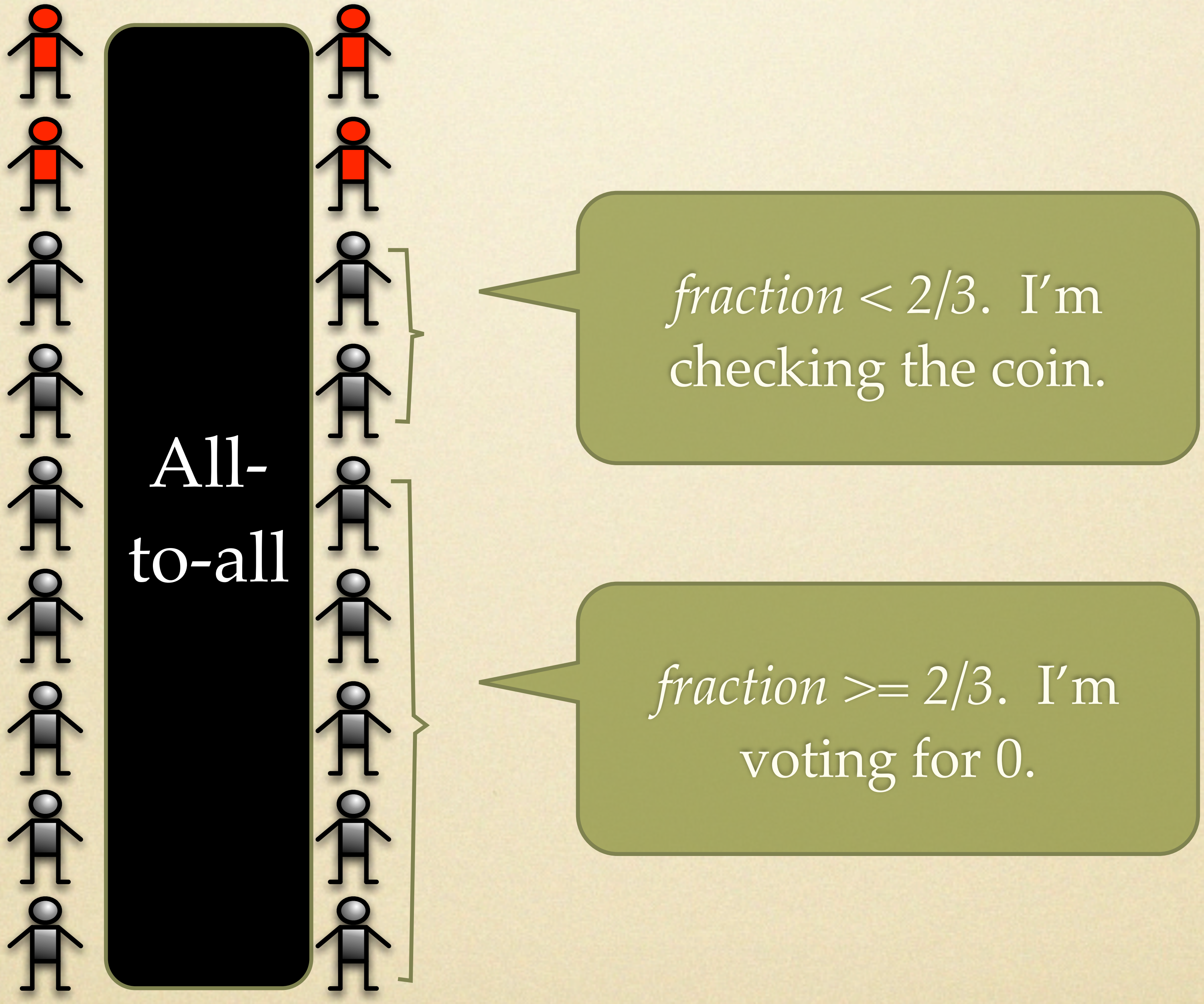

All-to-all
fraction < 2/3. I'm checking the coin.
fraction >= 2/3. I'm voting for 0.

Note: The procs with *fraction* >= 2/3 will all change vote to same value

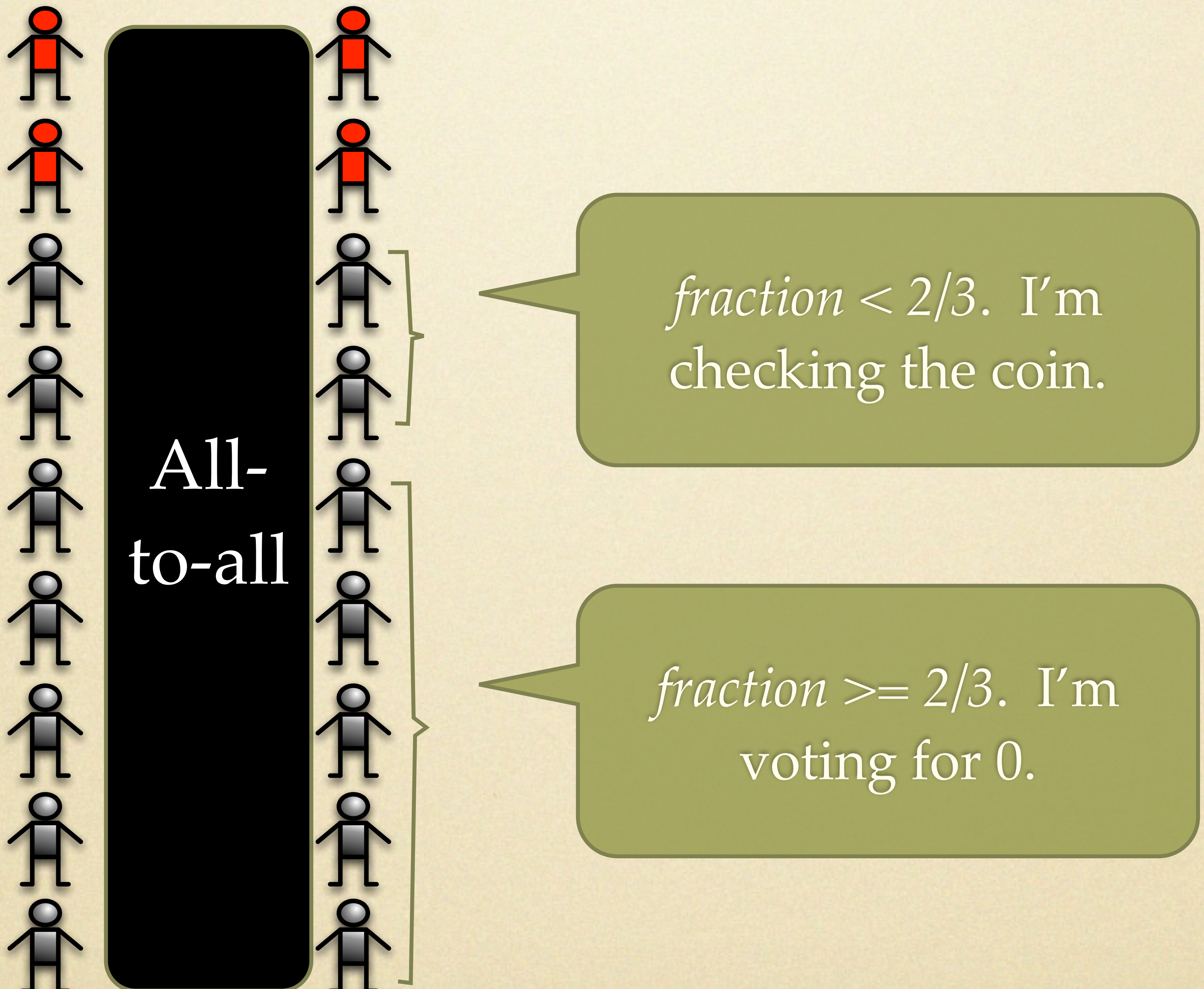


fraction < 2/3. I'm checking the coin.

fraction >= 2/3. I'm voting for 0.

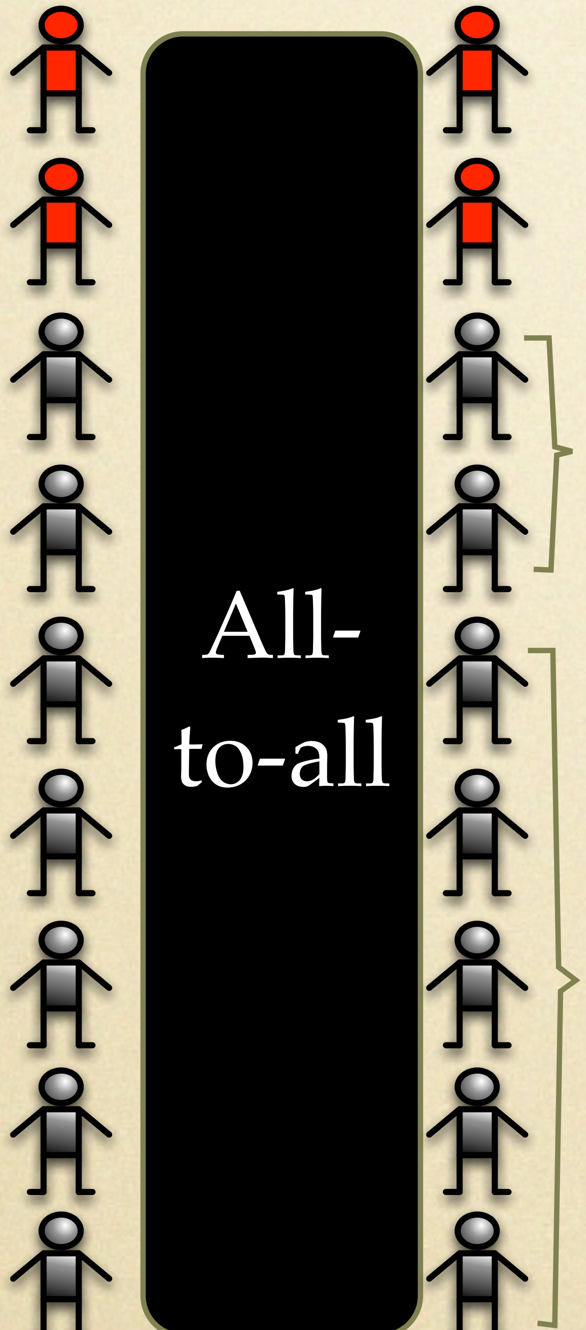

All-to-all
fraction < 2/3. I'm checking the coin.
fraction >= 2/3. I'm voting for 0.

Probability 1/2 that both groups change vo
to the same value



Probability 1/2 that both groups change vo
to the same value

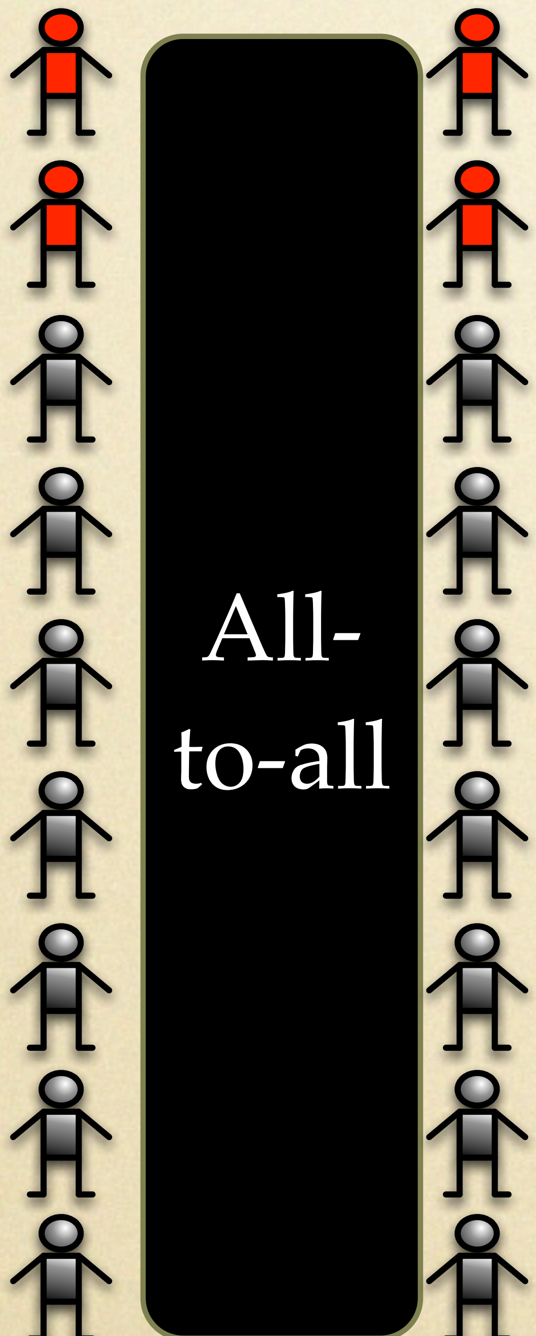
Once this happens, all votes of good procs will be equal evermore



fraction < 2/3. I'm checking the coin.

fraction >= 2/3. I'm voting for 0.

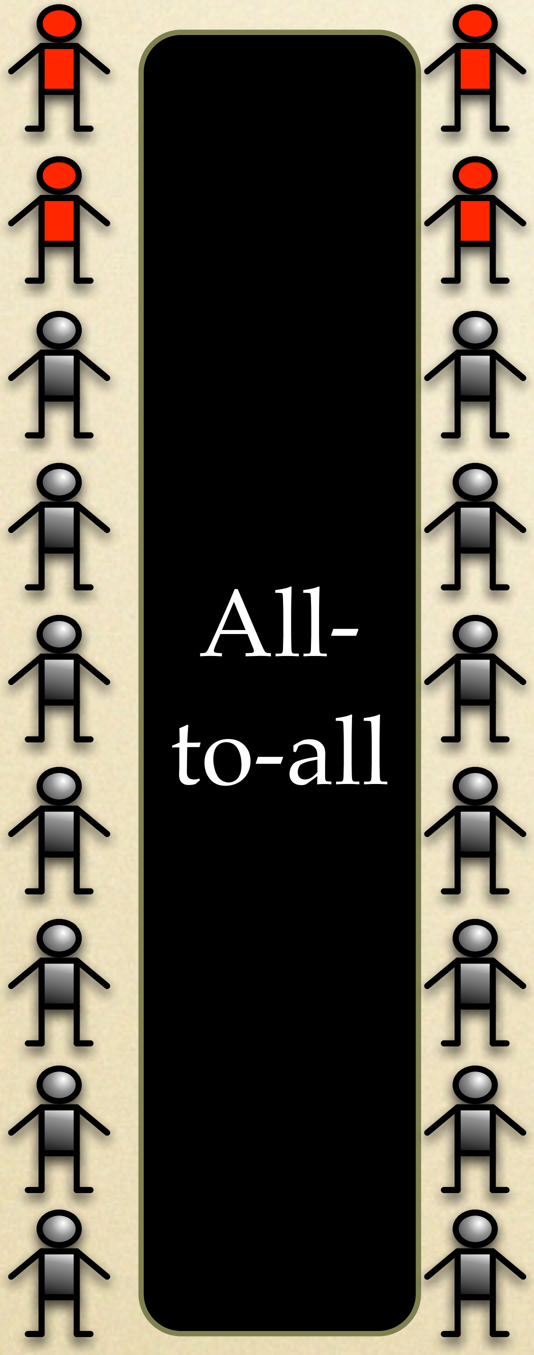
Probability 1/2 that both groups change vo
to the same value



Once this happens, all votes of good procs will be equal evermore

Prob of failure $= (1/2)^{clogn}$

Probability 1/2 that both groups change vo
to the same value



Once this happens, all votes of good procs will be equal evermore

$$\text{Prob of failure} = (1/2)^{c \log n}$$
$$= 1/n^c$$

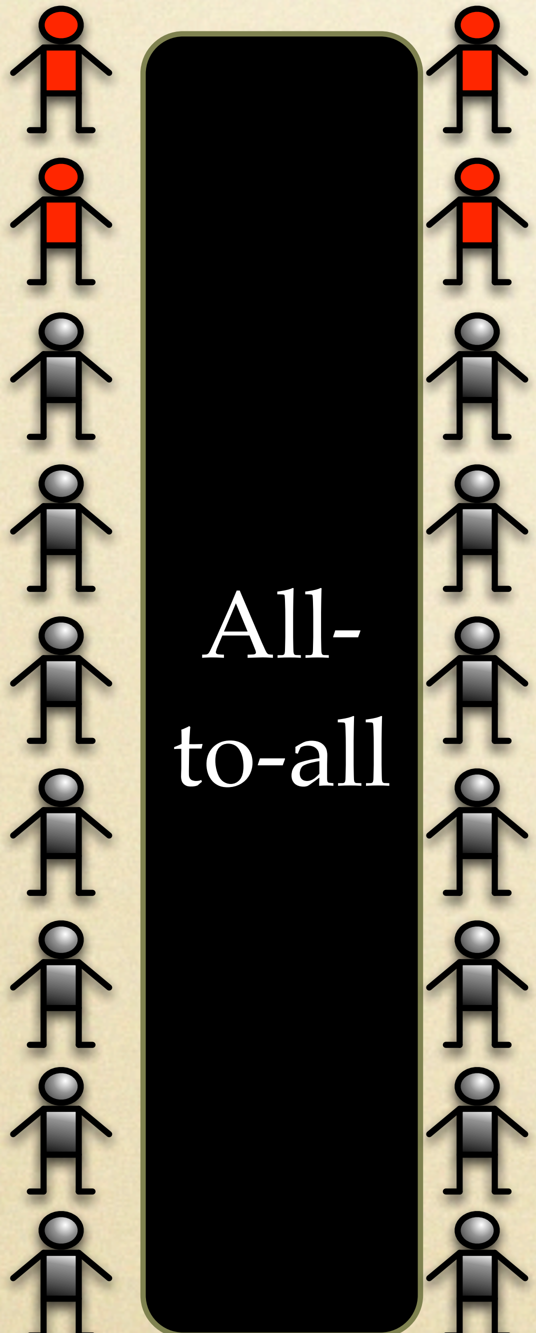
Probability 1/2 that both groups change vo
to the same value

Once this happens, all votes of good procs will be equal evermore

$$\text{Prob of failure} = (1/2)^{clogn}$$
$$= 1/n^c$$

$$\text{Prob of success} = 1 - 1/n^c$$

Probability 1/2 that both groups change vo
to the same value



Once this happens, all votes of good procs will be equal evermore

$$\text{Prob of failure} = (1/2)^{c \log n}$$
$$= 1/n^c$$

$$\text{Prob of success} = 1 - 1/n^c$$

whp

Idea for fail-stop faults (see e.g. [AC '08])

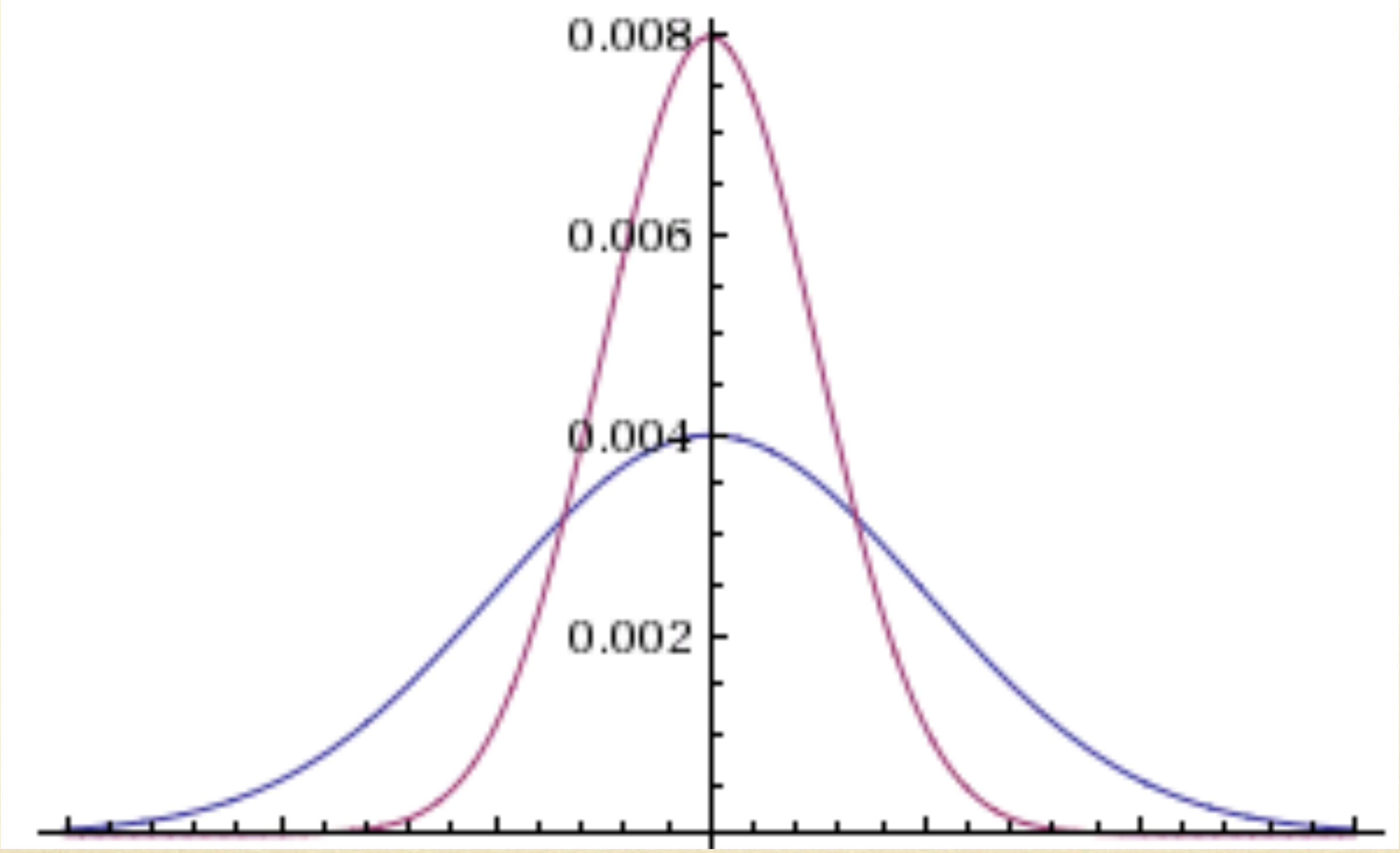
- Flip n^2 coins. Let heads be +1 and tails be -1, and dev be the sum of all coins
- With constant probability, $|dev| \geq kn$ for any constant k
- If $|dev| \geq kn$
  - Direction of dev gives a fair global coin
  - Direction of dev is robust to loosing some coins

Key Idea

- Each proc flips n coins; deviation (dev) is the sum of all n^2
- If dev $\geq$ kn, direction of dev is a fair global coin
- **Problem:** Bad procs may lie about their coinflips
- Q: Can we determine which procs are bad by studying deviation of coinflips?

Deviation Probabilities

probability

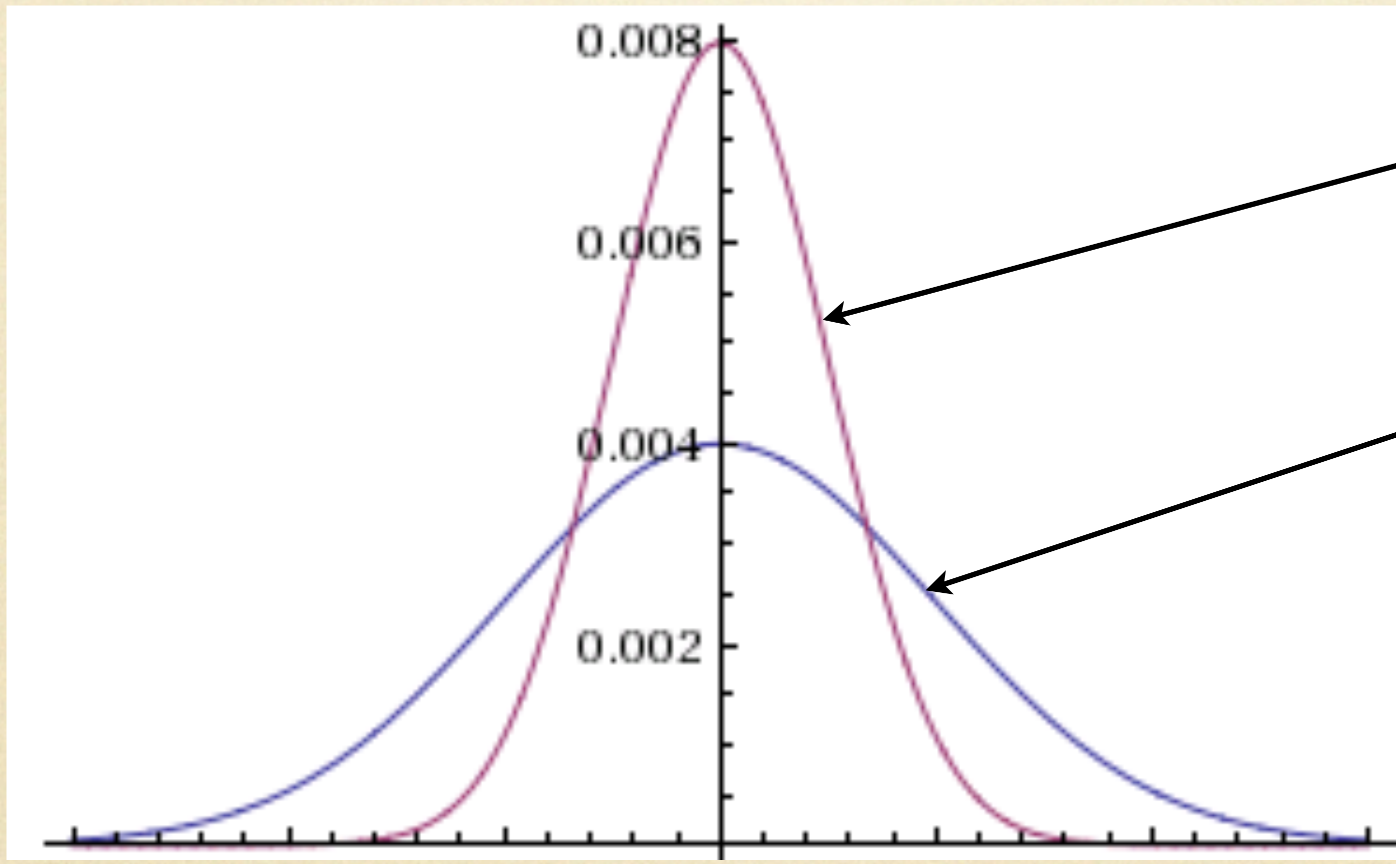


deviation

Deviation Probabilities

t procs

probability

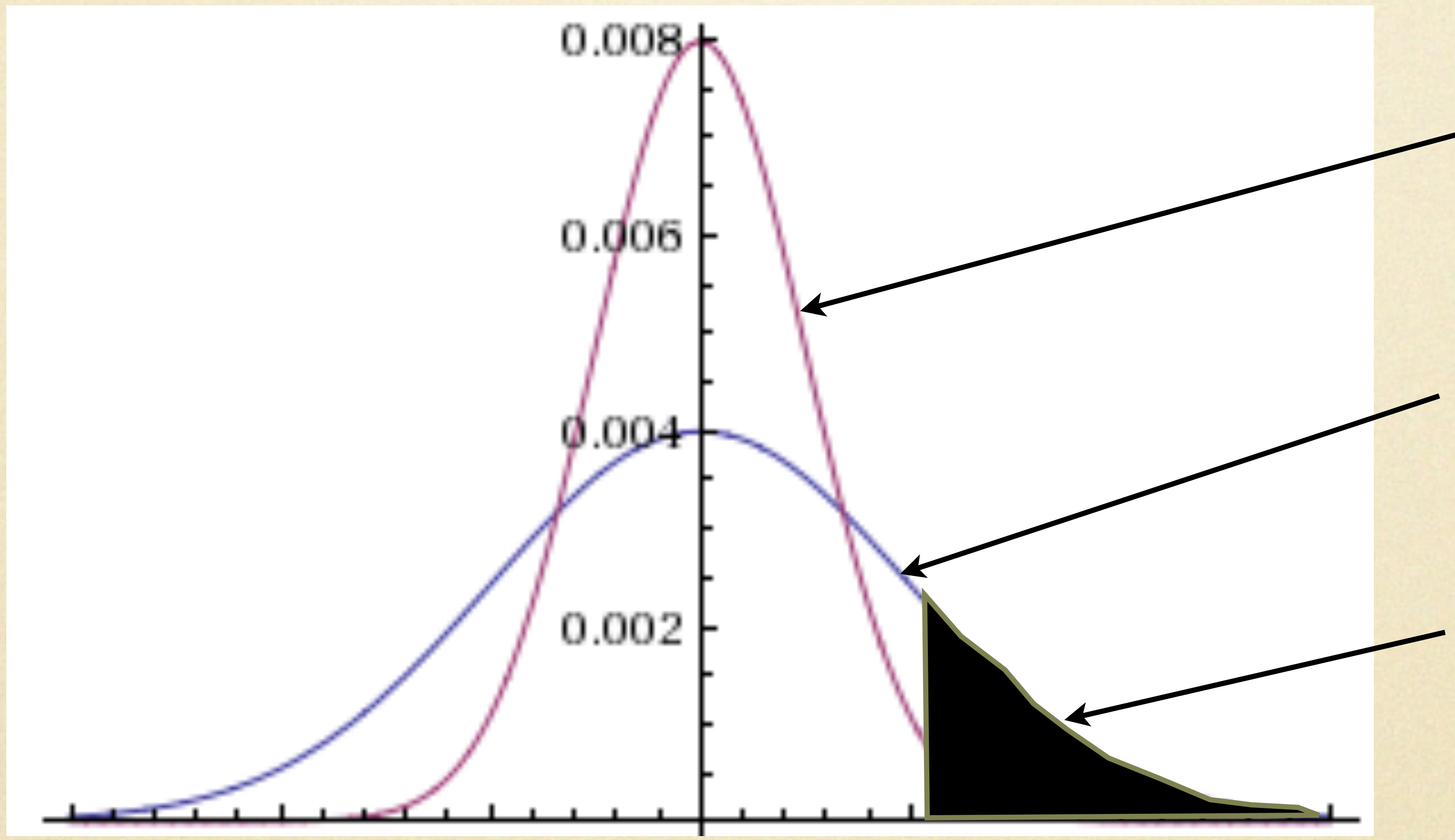


n-t procs

deviation

Deviation Probabilities

probability



t procs

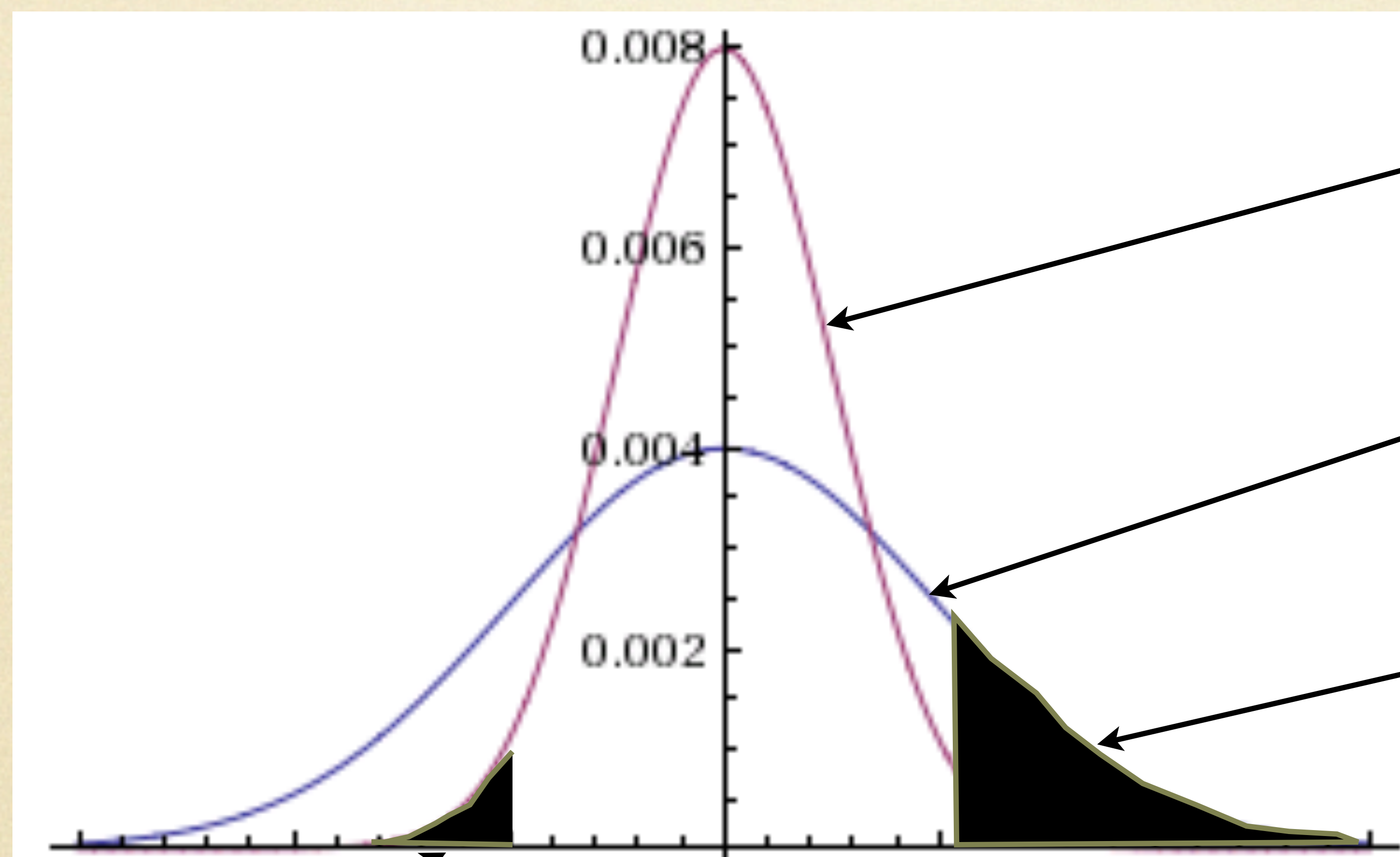
n-t procs

prob n-t procs
have dev ≥ kn

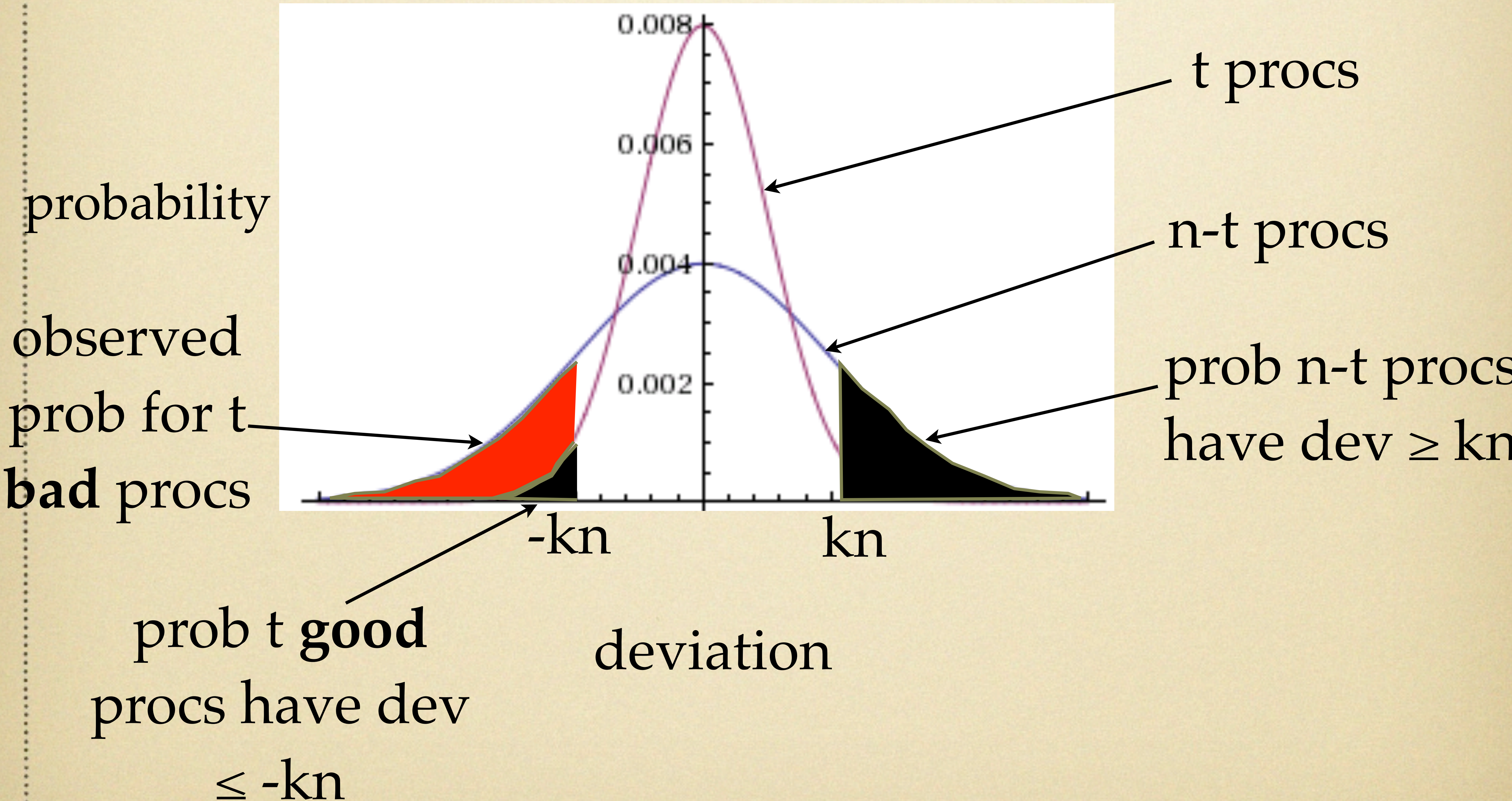
kn

deviation

Deviation Probabilities



Deviation Probabilities



Key Idea

- With constant probability, the direction of **good** dev gives a fair global coin
- Bad nodes need to generate **bad** dev. in the opposite direction of equal magnitude to foil this good event
- Problem (for bad): There are fewer bad procs than good ones; if the few bad procs generate large amounts of bad dev. repeatedly, we can find them

Reliable Broadcast (Bracha)

- All coinflip values sent using reliable broadcast
- Ensures if a message is “received” by a good proc, same message is eventually “received” by all procs
- Prevents equivocation
- Doesn’t solve BA
  - If a bad player reliably broadcasts, may be case that **no** good player “receives” the message

Coinflip Messages

- We can ensure the following:
  - Each processor broadcasts no more than n coinflips
  - Bad procs forced to be consistent about their coinflip values
  - Most (n-4t) good processors receive all but 2 coinflips from all good processors

Deviation

- We assume all coinflips are either +1 or -1
- The **deviation** of p in an iteration is the absolute value of the sum of p's coinflips
- The **direction** of p in an iteration is the sign of the sum of p's coinflips
- **idev(S,i):** absolute value of all coinflips sent by procs in S in iteration i

Perspective

- In reality, different processors may receive different sets of coinflips
- To be precise, we should have subscripts for terms like **idev** to indicate the n different perspectives
- In this talk, we omit subscripts and assume perspective of a processor that receives all coinflips

Iterations and Epochs

- In each iteration, we run Ben-Or
- There are cn iterations in an **epoch**
- In each epoch, we expect a constant fraction of iterations to be **good** i.e. dev. of good procs is $\geq B$ in right direction ($B = c'n$ for a fixed c')

Iterations and Epochs

- In each iteration, we run Ben-Or
- There are cn iterations in an **epoch**
- In each epoch, we expect a constant fraction of iterations to be **good** i.e. dev. of good procs is $\geq B$ in right direction ($B = c'n$ for a fixed c')
- In a **good** iteration, bad procs must have dev. $\geq B/2$
- (Remaining "good" deviation undone by scheduler)

Key fact

In every non-terminating epoch e, there is a set of c_2n iterations $\mathbf{I_e}$ and a set of $\leq t$ processors $\mathbf{B_e}$, such that for all i in I_e:

$$\mathbf{idev}(\mathbf{B_e},\mathbf{i}) \geq B/2$$

Bipartite Graph

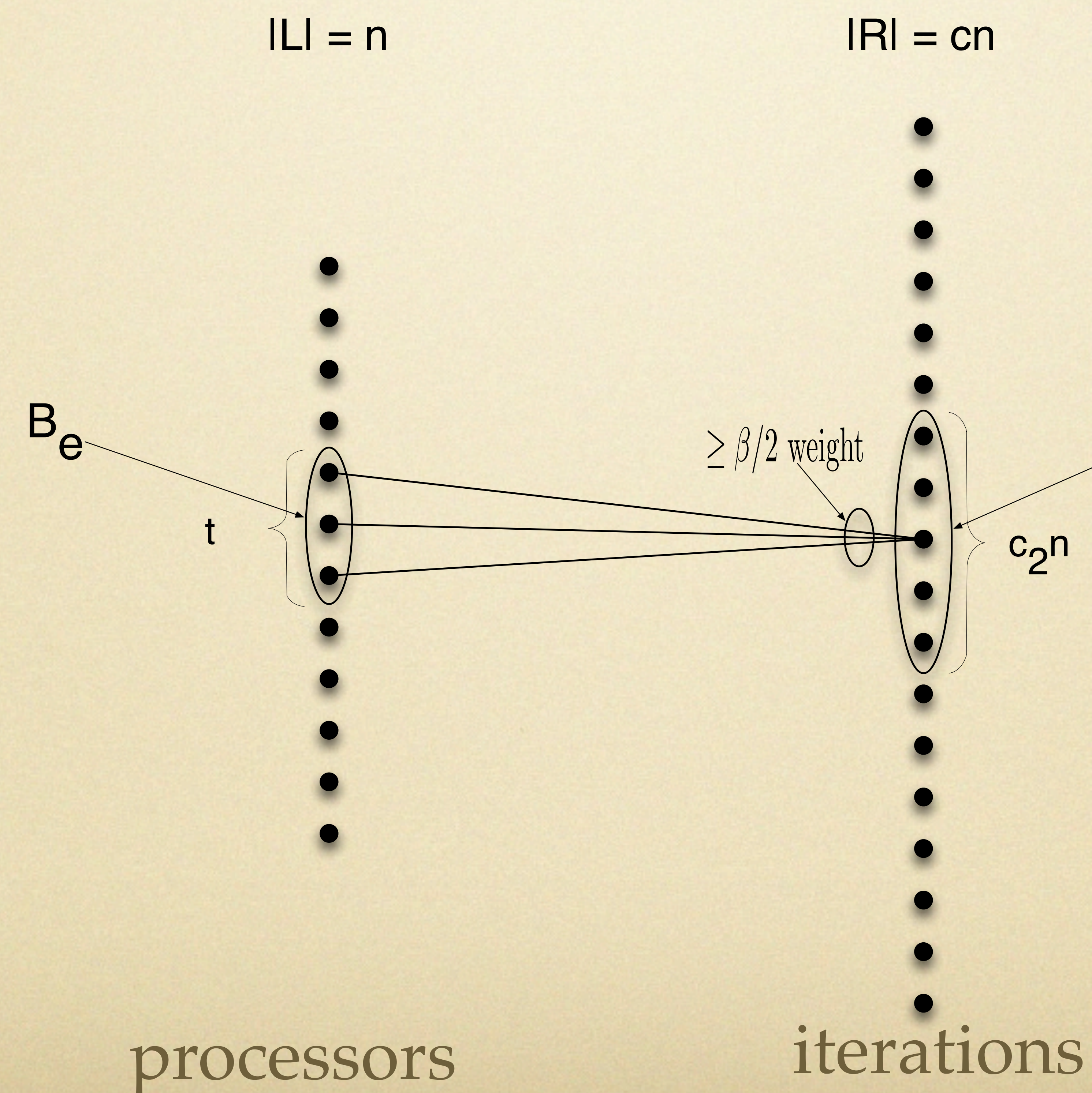
Bipartite Graph

edge between
each proc p and
each iter i with
weight = dvtn of
p in iter i

processors

iterations

Bipartite Graph



edge between each proc p and each iter i with weight = dvtn of p in iter i

cumdev(p)

- **cumdev(p)** starts at 0
- In each epoch, for every proc p in B_e, **cumdev(p) +=** "deviation of each processor p in direction of B_e, summed over all iterations I_e"
- We blacklist any proc p when **cumdev(p)** exceeds $n^{1.5}(\ln n)$

Algorithm

Algorithm

- Run an epoch (cn iterations of modified Ben-Or)
- If the epoch fails, find sets B_e and I_e
- Increase **cumdev** scores for every proc in B_e by amount they contributed to dev in each iter in I_e
- Blacklist blacklist any proc p when **cumdev(p)** exceeds $n^{1.5}(\ln n)$
- (We also blacklist any proc which has highly unlikely dev in any iteration (> $c' n^{.5} \log n$))

cumdev facts

- Let X be the sum of cumdev(p) for all procs p
- Fact 1: X is upper bounded by $\sim n^{2.5}(\ln n)$
- Fact 2: X increases by $(B/2)c_2 n \sim n^2$ in every epoch

Thus there are $\sim n^{.5}(\ln n)$ epochs

Lemma 12

Lemma 12: Assume the number of blacklisted procs is $\leq t$. Then in every non-terminating epoch e, there is a set of $c_2 n$ iterations $\mathbf{I_e}$ and a set of $\leq t$ processors $\mathbf{B_e}$, such that for all i in I_e:

$$\mathbf{idev(B_e,i)} \geq B/2$$

Lemma 14

- **Lemma 14:** The number of blacklisted good procs is no more than t (whp).

Sketch of Lemma 14

- Let GOOD be the set of good procs, BAD be set of bad procs; **cumdev(S,**e) = amount added to **cumdev** for all procs in S in epoch e

Sketch of Lemma 14

- Let GOOD be the set of good procs, BAD be set of bad procs; **cumdev(S,**e) = amount added to **cumdev** for all procs in S in epoch e

- **Lemma 13:** Whp, for any epoch e, **cumdev(**GOOD**,**e**)** $\leq (\beta/5)c_2 n$

Sketch of Lemma 14

- Let GOOD be the set of good procs, BAD be set of bad procs; **cumdev(S,**e) = amount added to **cumdev** for all procs in S in epoch e

- **Lemma 13:** Whp, for any epoch e, **cumdev(**GOOD**,**e**)** ≤ $(\beta/5)c_2n$

- Fact 2: X increases by $(\beta/2)c_2n$ in every epoch

- Thus: **cumdev(**BAD$\cap$$B_e$,e**)** ≥ $(3\beta/10)c_2n$

Sketch of Lemma 14

- **cumdev(**GOOD**,**e**)** increases by $\leq$ (2 ß/10)c_2n
- **cumdev(**BAD∩B_e,e**)** increases by $\geq$ (3 ß/10)c_2n
- You can see where this is going!

Proof of Lemma 13

- Let $G = GOOD \cap B_e$
- Fix a set G, a set I_e and a mapping, d, from iterations in I_e to {-1,+1}
- Let Y = sum of coins generated by G in iterations I_e in directions given by d

Proof of Lemma 13

Lemma 13: Whp, for any epoch e, **cumdev(**GOOD,e**)** $\leq (\beta/5)c_2n$

- Let $G = \text{GOOD} \cap B_e$
- Fix a set G, a set I_e and a mapping, d, from iterations in I_e to {-1,+1}
- Let Y = sum of coins generated by G in iterations I_e in directions given by d

Proof of Lemma 13

Lemma 13: Whp, for any epoch e, **cumdev(**GOOD,e**)** $\leq (\beta/5)c_2 n$

- Let $G = \text{GOOD} \cap B_e$
- Fix a set G, a set I_e and a mapping, d, from iterations in I_e to {-1,+1}
- Let Y = sum of coins generated by G in iterations I_e in directions given by d

We use Y to bound amount added to **cumdev**(G) in epoch e

Bounding Y

Chernoff bound

$$Pr(Y \geq (\beta/6)(c_2 n)) \leq e^{-(c_2 n \beta/6)^2/2(|G|c_2 n^2)}$$
$$\leq e^{-.026 c_2 n^2/t}$$

Bounding Y

Chernoff bound

$$Pr(Y \geq (\beta/6)(c_2 n)) \leq e^{-(c_2 n\beta/6)^2/2(|G|c_2 n^2)}$$
$$\leq e^{-.026 c_2 n^2/t}$$

Let ξ be event that $Y \geq (\beta/6)(c_2 n)$ for *any* G, I_e, and mapping d

- $\binom{cn}{c_2 n} \leq (ce/c_2)^{c_2 n}$ ways to pick the iterations I_e
- $\sum_{i=1}^{t} \binom{n}{i} \leq 2^n$ ways to pick the set G
- $2^{c_2 n}$ ways to pick the mapping d

Bounding Y

Chernoff bound

$$\begin{aligned} Pr(Y \geq (\beta/6)(c_2 n)) &\leq e^{-(c_2 n\beta/6)^2/2(|G|c_2 n^2)} \\ &\leq e^{-.026 c_2 n^2/t} \end{aligned}$$

Let ξ be event that $Y \geq (\beta/6)(c_2 n)$ for *any* G, I_e, and mapping d

- $\binom{cn}{c_2 n} \leq (ce/c_2)^{c_2 n}$ ways to pick the iterations I_e
- $\sum_{i=1}^{t} \binom{n}{i} \leq 2^n$ ways to pick the set G
- $2^{c_2 n}$ ways to pick the mapping d

Union Bound

$$
\begin{aligned}
Pr(\xi) &\leq (ce/c_2)^{c_2 n} 2^n 2^{c_2 n} e^{-.026 c_2 n^2/t} \\
&\leq e^{11 c_2 n - (.026 c_2 n^2/t)} \\
&\leq e^{-\Omega(n)} \qquad \text{Setting } n/t \geq 500
\end{aligned}
$$

$\blacksquare$

Union Bound

$$\begin{aligned} Pr(\xi) &\leq (ce/c_2)^{c_2 n} 2^n 2^{c_2 n} e^{-.026 c_2 n^2/t} \\ &\leq e^{11 c_2 n - (.026 c_2 n^2/t)} \\ &\leq e^{-\Omega(n)} \qquad \text{Setting } n/t \geq 500 \end{aligned}$$

Another union bound over the polynomial number of epochs and (views of) all good procs completes the proof. ∎

Conclusion

- First expected polynomial time algorithm for traditional Byzantine agreement
- Previous best algorithm (Ben-or's) was expected exponential time
- New technique: design of algorithms that force attackers into **statistically** deviant behavior that is detectable

Open Problems

- Can we improve resilience (currently we must have $t \leq n/500$)
- Our algorithm requires exponential computation. Can we reduce this to polynomial computation?
  - Computational problem is similar to finding a hidden high-weight subgraph
- Can we improve other randomized algorithms by forcing bad procs into detectably deviant behavior?

Questions