Breaking the $O(n^2)$ Bit Barrier: Scalable Byzantine Agreement

Jared Saia, U. New Mexico

Joint with Valerie King, U. Victoria

Unreliable Components

- Imagine have a collection of chips, some of which are unreliable
- Goal: build a reliable computer

Unreliable Components

- Imagine have a collection of ~~chips~~, some of which are unreliable

- Goal: build a reliable ~~computer~~

Unreliable Components

- Imagine have a collection of ~~chips~~ computers, some of which are unreliable
- Goal: build a reliable ~~computer~~

Unreliable Components

- Imagine have a collection of ~~chips~~ computers, some of which are unreliable
- Goal: build a reliable ~~computer~~ network

Unreliable Components

- Imagine have a collection of ~~chips~~, some of which are unreliable

- Goal: build a reliable ~~computer~~ network

Unreliable Components

- Imagine have a collection of ~~chips~~, some of which are unreliable

- Goal: build a reliable ~~computer~~

Unreliable Components

- Imagine have a collection of ~~chips~~ people, some of which are unreliable

- Goal: build a reliable ~~computer~~

Unreliable Components

- Imagine have a collection of ~~chips~~ people, some of which are unreliable

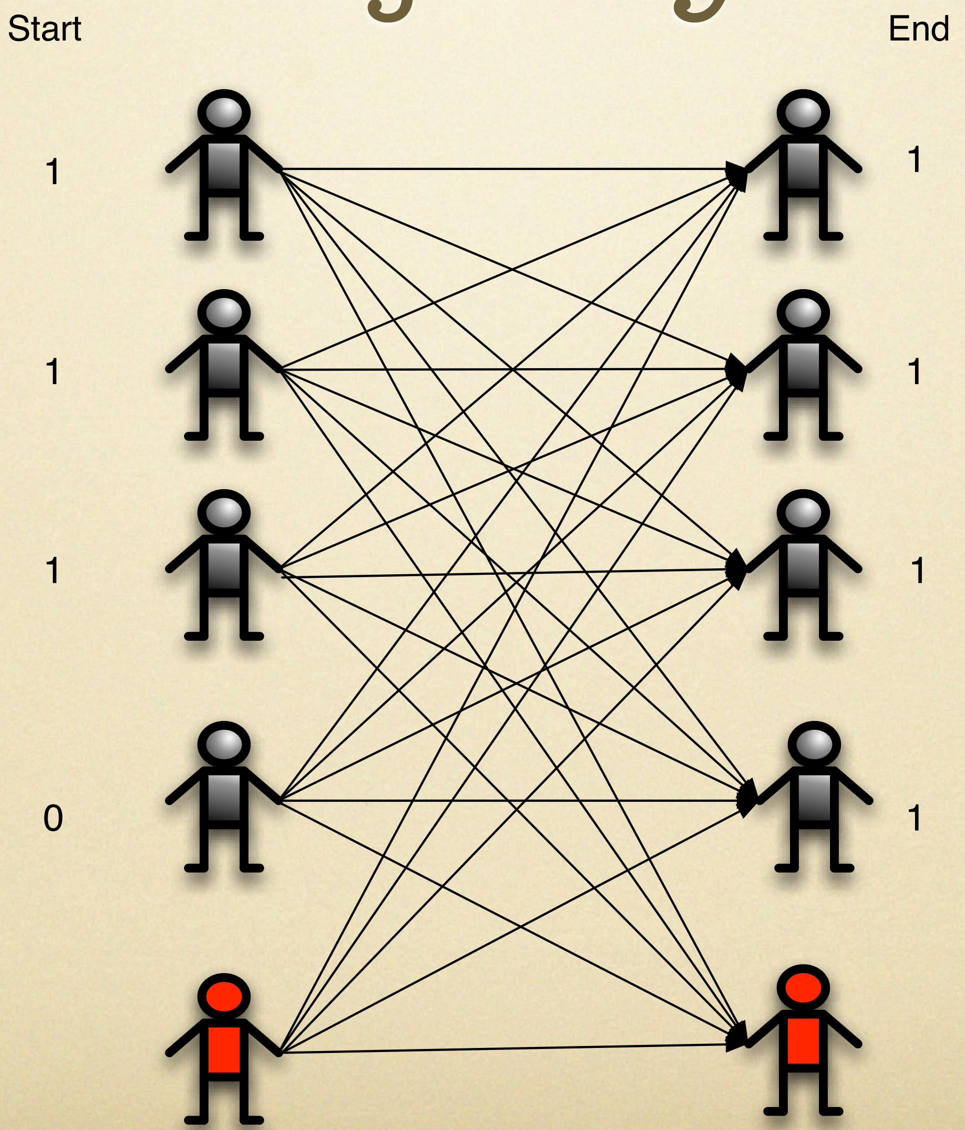
- Goal: build a reliable ~~computer~~ system

Components Fail, Group Functions

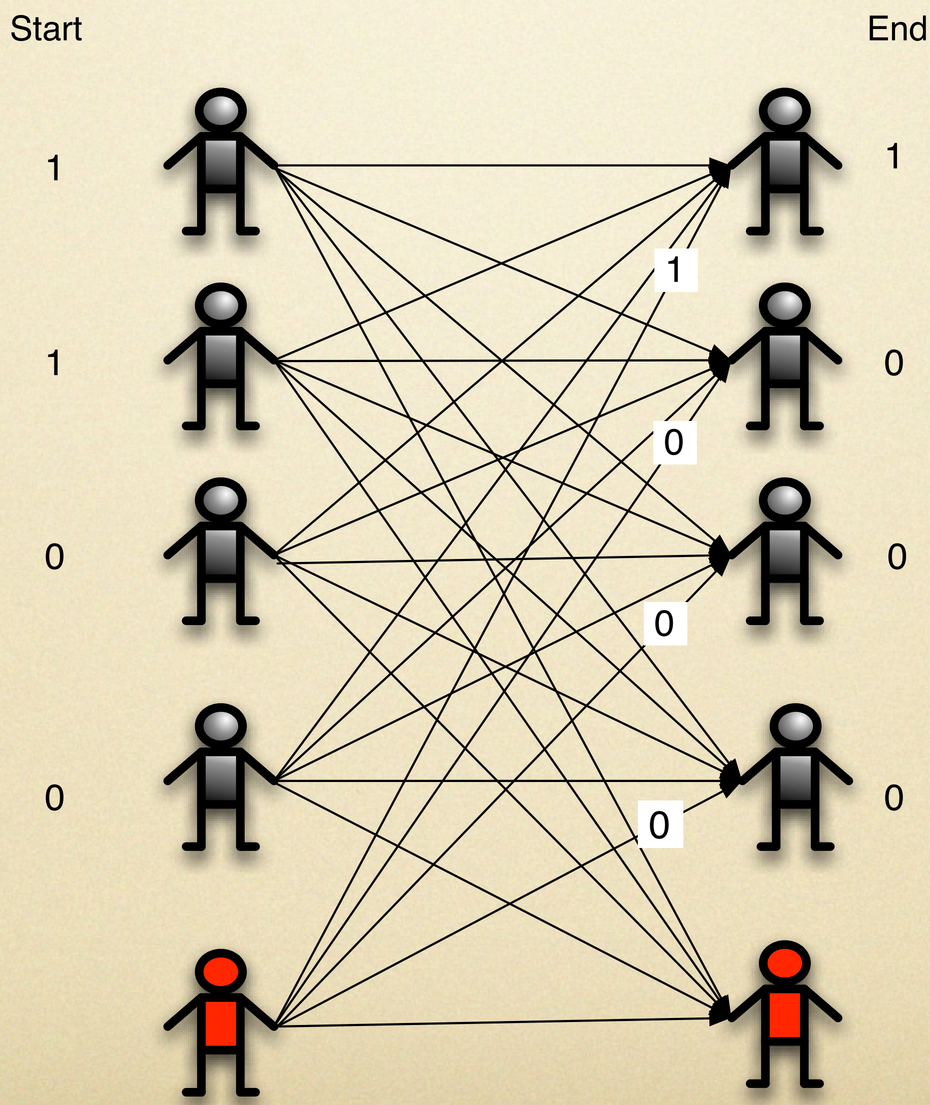
Group Synchronization

- Periodically, all components must unite in action
- How? Idea: components vote on correct action
- Problem: How to count the votes?

Idea: Majority Voting



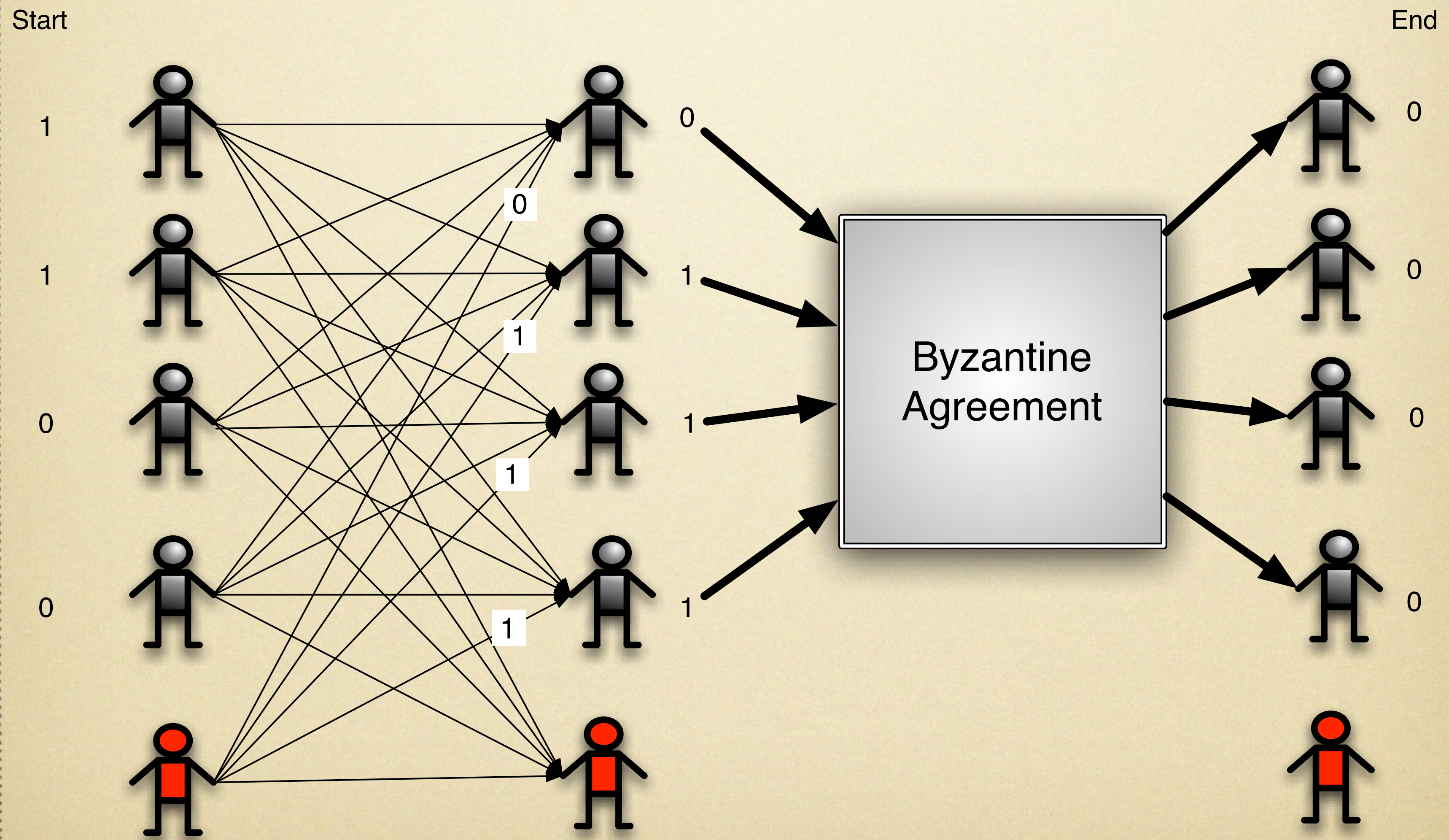
A Problem



Byzantine Agreement

- Each processor starts with a bit
- Goal: All good processors output a bit, that is the same as one of their initial bits
- t = # bad processors controlled by an adversary

Problem Solved



Importance

- BA is ***synchronization in complex systems***
  - How do fireflies, economic markets, ants, computer networks, bees, brains, immune systems function without a leader?
- Sine qua non of robust computation

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"

2,800 Cites Later

- Deterministic, Randomized
- Cryptography, No cryptography
- Synchronous, Asynchronous
- Adaptive, non-adaptive adversary
- Quantum, Shared Memory, Fault-Detectors, Sparse Network, Leader Election, Global Coin Toss, Etc., Etc,

Large-Scale BA

- Peer-to-peer networks (Oceanstore, Farsite)

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

- Rule Enforcement

  *"... requiring the manager set to perform a **Byzantine agreement protocol**"*

- Game Theory (Mediators)

  *"The proofs of the impossibility results bring out deep connections between implementing mediators and various agreement problems, such as **Byzantine agreement**"*

Scalability

- *"Unfortunately, Byzantine agreement requires a **number of messages quadratic** in the number of participants, so it is infeasible for use in synchronizing a large number of replicas"* [REGZK '03]

- *"Eventually batching cannot compensate for the **quadratic number of messages** [of Practical Byzantine Fault Tolerance (PBFT)]"* [CMLRS '05]

- *"The **communication overhead** of Byzantine Agreement is **inherently large**"* [CWL '09]

Our Model

- Synchronous w/ rushing adversary
- Private channels
- Resilience: $t < n(1/3-\varepsilon)$
- Unlimited messages for bad procs
- Adaptive adversary

Our Goal: Scalable BA

- Polylog bits sent per processor
- Polylog rounds

Impossibility

- Any BA (randomized) protocol which always uses $o(n^2)$ messages will fail with probability > 0
- Implication of [Dolev, Reischuk '85]

Our results

Theorem 1 (BA): For any constants c, ε, there is a constant d and a (1/3- ε)n resilient protocol which solves BA with prob. $1-1/n^c$ using

$\tilde{O}(n^{1/2})$ bits per processor in $O(\log^d n)$ rounds

Also

Theorem 2: (a.e.BA) For any constants. c, ε, there is a constant d and a (1/3- ε)-resilient protocol which brings

1-O(1/log n) fraction of good procs to agreement with prob. $1-1/n^c$ using

Õ(1) bits per proc in $O(\log^d n)$ rounds

Previous work

- An expected constant number of rounds suffice. (Feldman and Micali 1988)
- However, all previously known protocols use all-to-all communication

KEY IDEA: Short somewhat random stream S

- S= s_1 s_2 … s_k is a short stream of numbers.
- Some a.e. globally known random numbers, some numbers fixed by an adversary which can see the preceding stream when choosing.
- S can be generated w.h.p.

Algorithm Outline

I: Using S to get a.e. BA

II: Using S to go from a.e. BA to BA

III: Generating S

Rabin's BA with Global Coin, GC

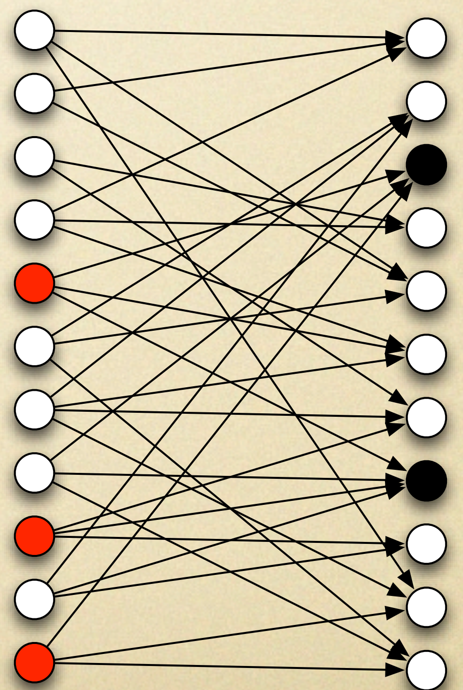
$vote \leftarrow b_i$; Repeat $c \log n$ rounds:

1. Send $vote$ to all procs;
2. $maj \leftarrow$ majority bit from others;
3. $fraction \leftarrow$ fraction of votes for maj;
4. If $fraction \geq 2/3$ then $vote \leftarrow maj$;
5. Else $vote \leftarrow GC$;

Scalable a.e.BA w/ GC

- Use **sampler** to assign neighbors to procs
- Ensures almost all neighbor sets contain a representative fraction of good procs
- Thus almost all procs have correct *maj* when "frac with majority bit" > $2/3 + \varepsilon/2$ and $t < n/3 - \varepsilon$

Sampler: Almost all nodes on right have majority good neighbors **no matter how bad distributed**



I: Using S to get a.e. BA

- Use S instead of GC --> a.e.BA whp
- For i=1,…,k, generate bit s_i
- Run a.e. BA using s_i for a.e.global coin
- It suffices that clogn bits of S are known a.e. and random

II: Using S to go from a.e. BA to BA

- Idea: Query random set of procs to ask bit. Since almost all good procs agree, majority should give correct answer.
- Problem: In our model, the adversary can flood all procs with queries!!
- Use s to decide which queries to answer.

II: Using S to go from a.e. BA to BA

Labels= $\{1,..,n^{1/2}\}$

FOR each number s of S=Labels^k :

- Each proc. p picks $\tilde{O}(n^{1/2})$ random queries <proc,label> and sends label to proc.
- q answers only if label= s (and not overloaded)
- if 2/3 majority of p's queries with the same label are returned and agree on v, then p decides v.

II: Using S to go from a.e. BA to BA

Labels= $\{1,..,n^{1/2}\}$

FOR each number s of S=Labels^k :

- Each proc. p picks $\tilde{O}(n^{1/2})$ random queries <proc,label> and sends label to proc.
- q answers only if label= s (and not overloaded)
- if 2/3 majority of p's queries with the same label are returned and agree on v, then p decides v.

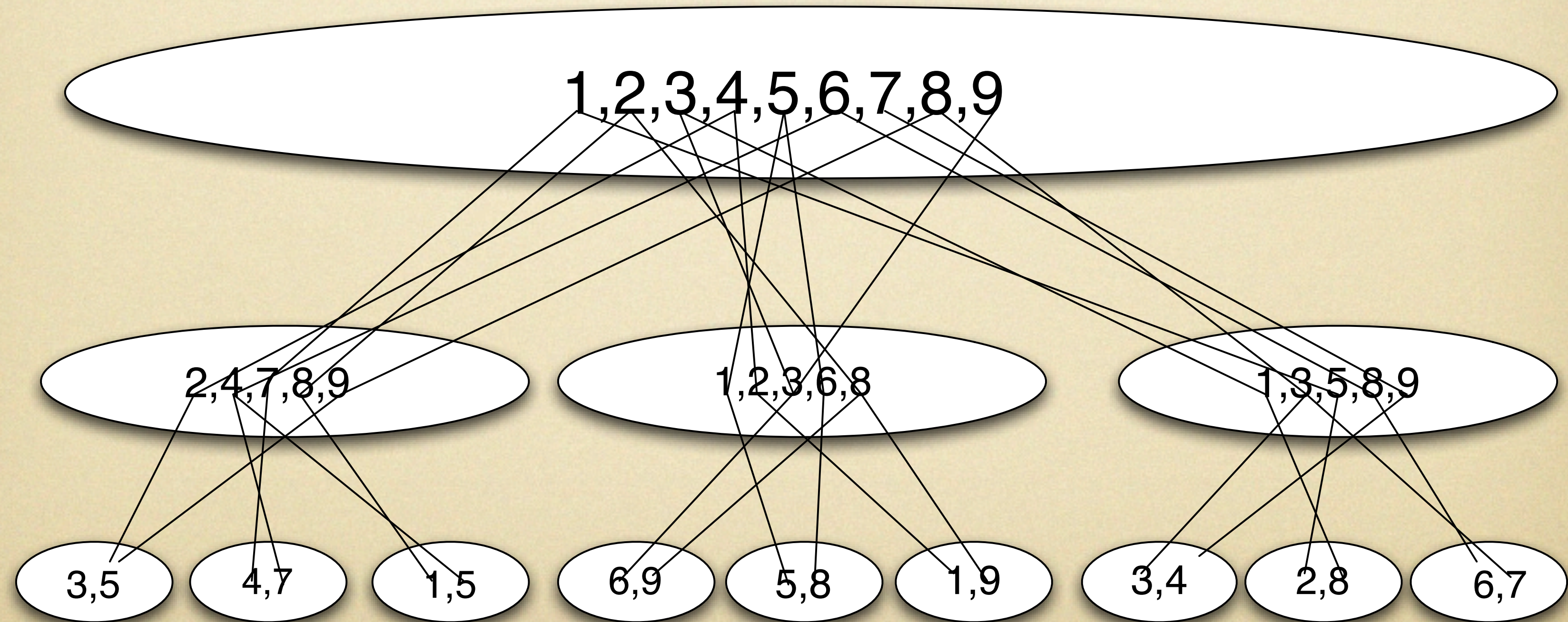
IT SUFFICES TO HAVE AN a.e. AGREED upon S with a RANDOM subsequence!

III Generating S

- Sparse Network
- Arrays of Random Numbers
- Lightest Bin Algorithm
- Secret Sharing

Sparse Network

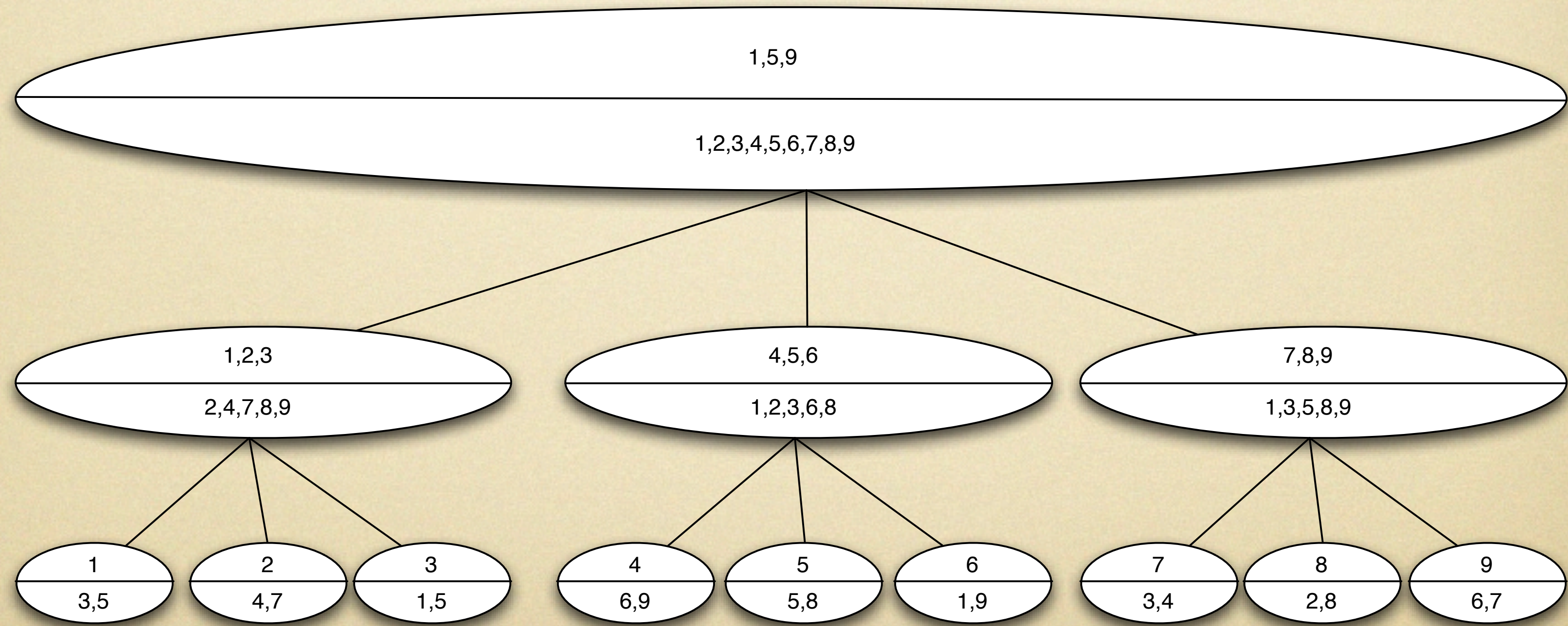
- Tree of supernodes of increasing size
- Linked: 1) child & parent; 2) parent & subtree leaves
- Links **and** Supernodes generated via samplers



Elections

- Each proc p generates array A_p of random numbers and **secret shares** it with its leaf node

- Numbers are revealed as needed to elect which parts of arrays will be passed on to parent node



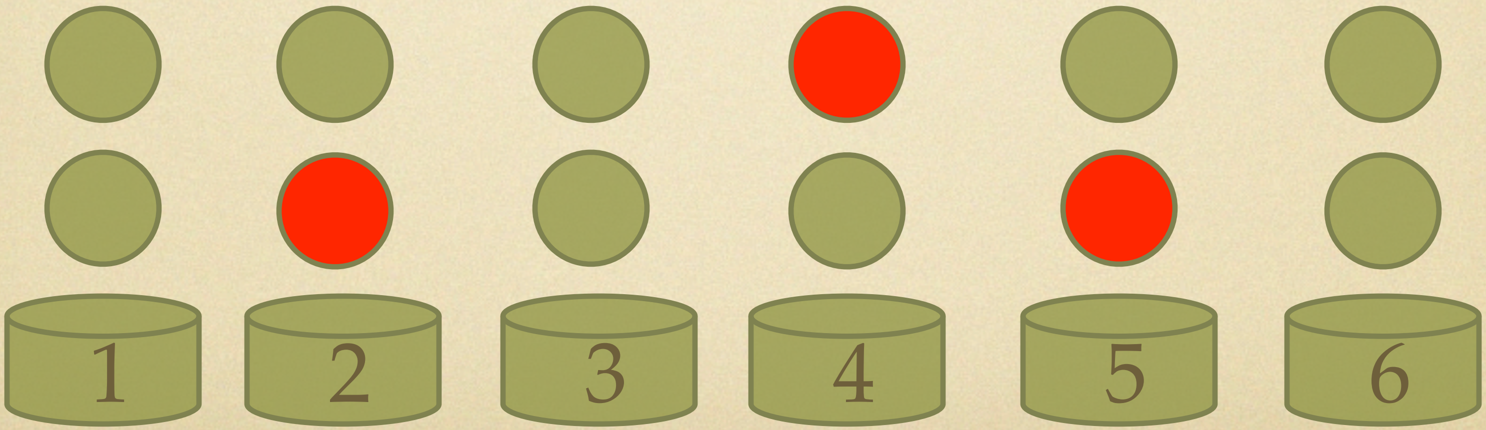
Election at a node

- Feige's algorithm:

  1. Each candidate picks a bin uniformly at random;

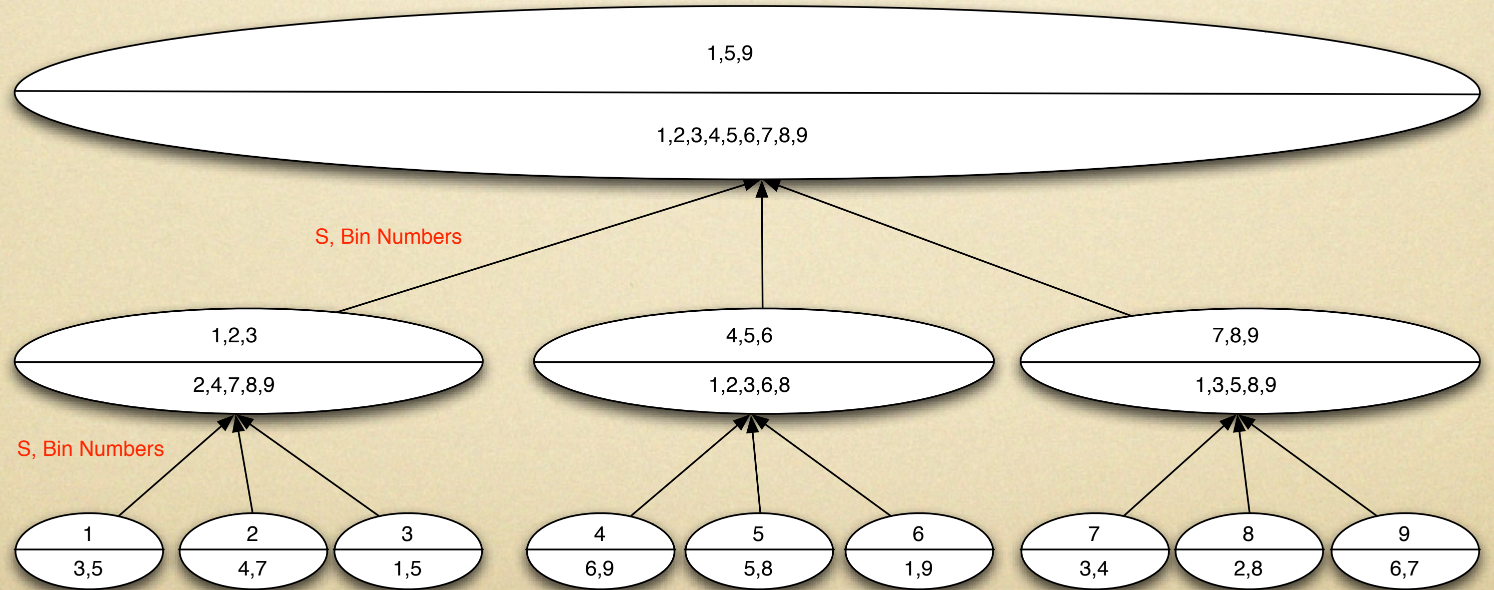
  2. Winners are candidates in lightest bin

- Requires Agreement on all bin choices



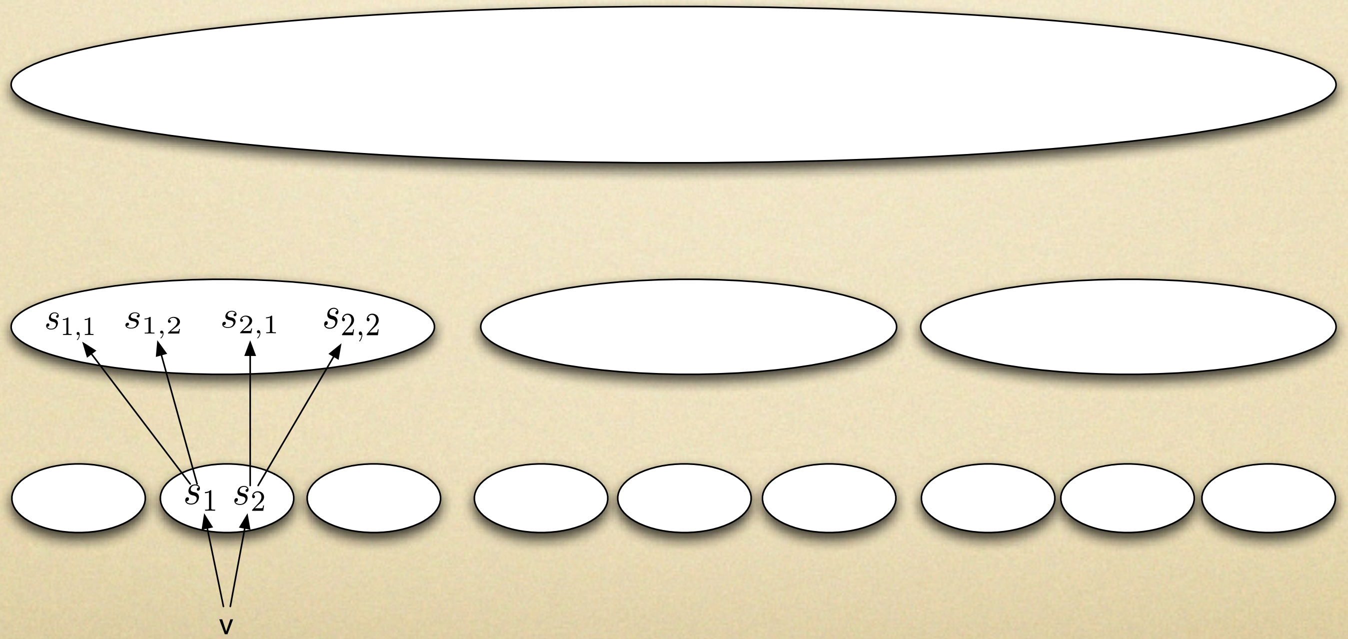
How to run Feige?

- We use scalable a.e. BA
- Bin numbers and S given by winning arrays of children supernodes.



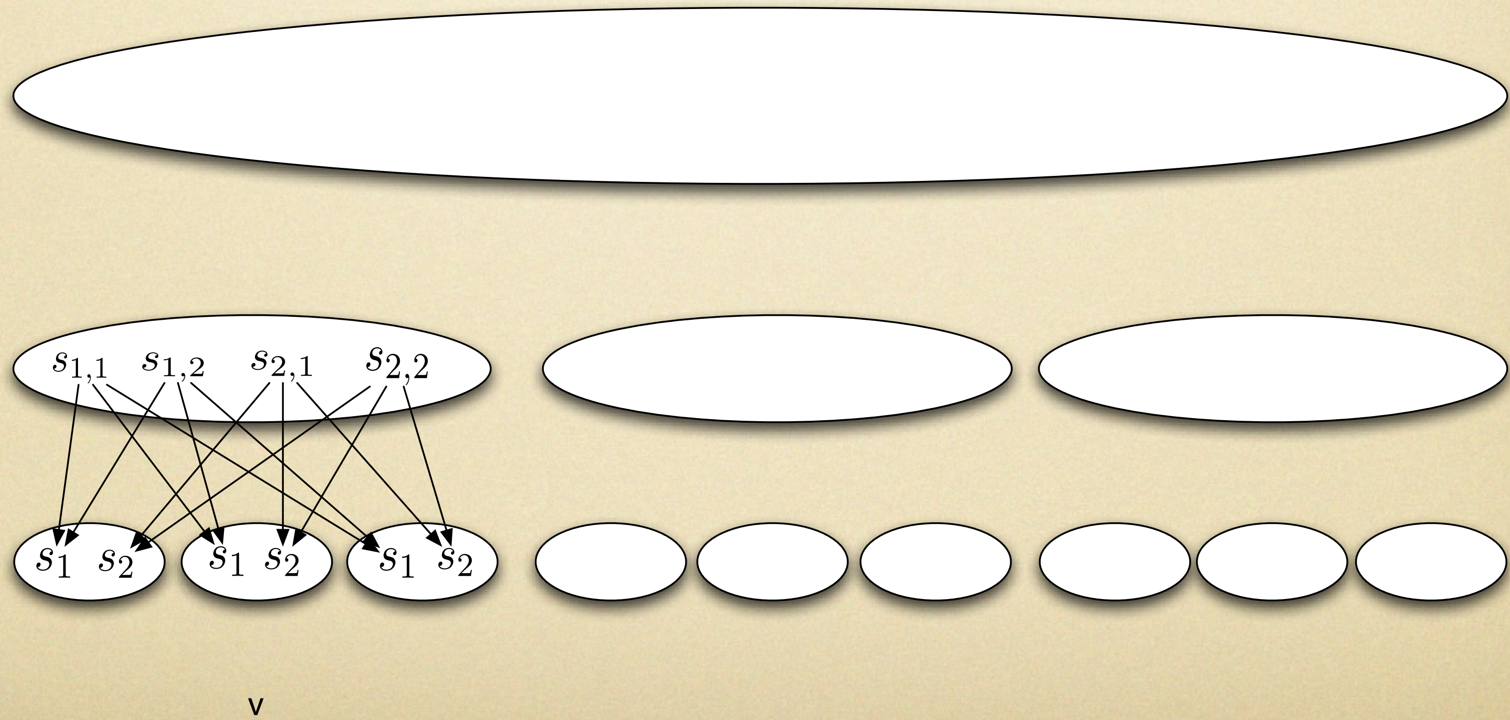
Splitting Secrets

- As winning array moves up, secret shares are split up among more and more procs on higher levels and erased from children
- Thus, adversary can't learn array by taking over small number of procs at lower levels



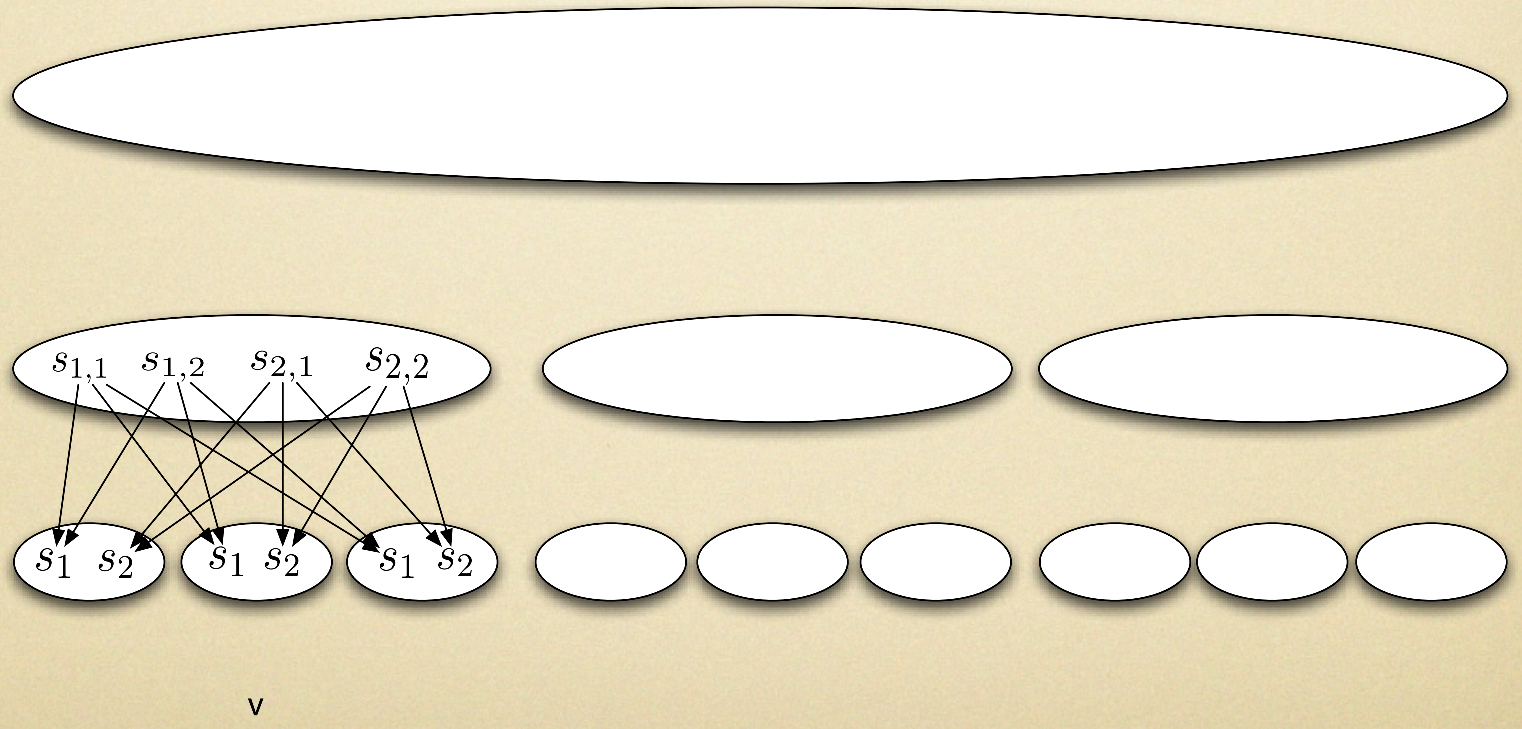
Revealing Secrets

- Secrets revealed as needed: by reversing communication downward, reassembling shares at subtrees and leaves
- Thus, adversary can't prevent secret from being exposed by blocking a single path



Revealing Secrets

- Leaves are sampled deterministically by procs in subtree root in order to learn the secret value



Generation of short S

- Only a polylog number of arrays are left at each of the polylog children of the root. These form S.
- When agreement on all of S is needed, a.e. BA can be run using supplemental bits.

Uses of S

- Easier to generate than a single random coinflip:
  - S can be generated w.h.p scalably in the full information nonadaptive adversary model
- A polylog size S has sufficient randomness to specify a set of n small quorums which are all good w.h.p
- Asynch alg w/nonadaptive adv

Past Scalable BA Results

- No crypto; Asynch communication; Non-adaptive Adv; o(1) prob. failure:
  - Algorithm for BA that requires $\tilde{O}(\sqrt{n})$ bits per proc and polylog latency
  - Algorithm for almost-everywhere BA (all but o(n) procs) that requires $\tilde{O}(1)$ bits per proc and polylog latency

Past Scalable Results (Same Assumptions)

Can solve following with $\tilde{O}(\sqrt{n})$ bits per proc and polylog latency

1. Leader Election: Leader good with constant prob
2. Quorum Selection

- A **good** quorum has a majority of good procs
- Can reach agreement on n good quorums
- Balanced: No proc in more than O(log n) quorums

Future work

- Scalable **asynchronous** BA with **adaptive** adversary?
- $\tilde{O}(\sqrt{n})$ bandwidth is fundamental?
- Practical scalable BA
  - Reducing constant factors and polylog terms; Relaxing fault model: e.g. bad procs have limited bandwidth

FW (Cont'd)

- Robust & Scalable for other problems
  - Done: global coin toss, leader election, frequency counts
  - Todo: SMPC type result
- Handle churn
  - Idea: Robust & Scalable mapping of n procs to distinct id in $[1, (1+\epsilon)n]$

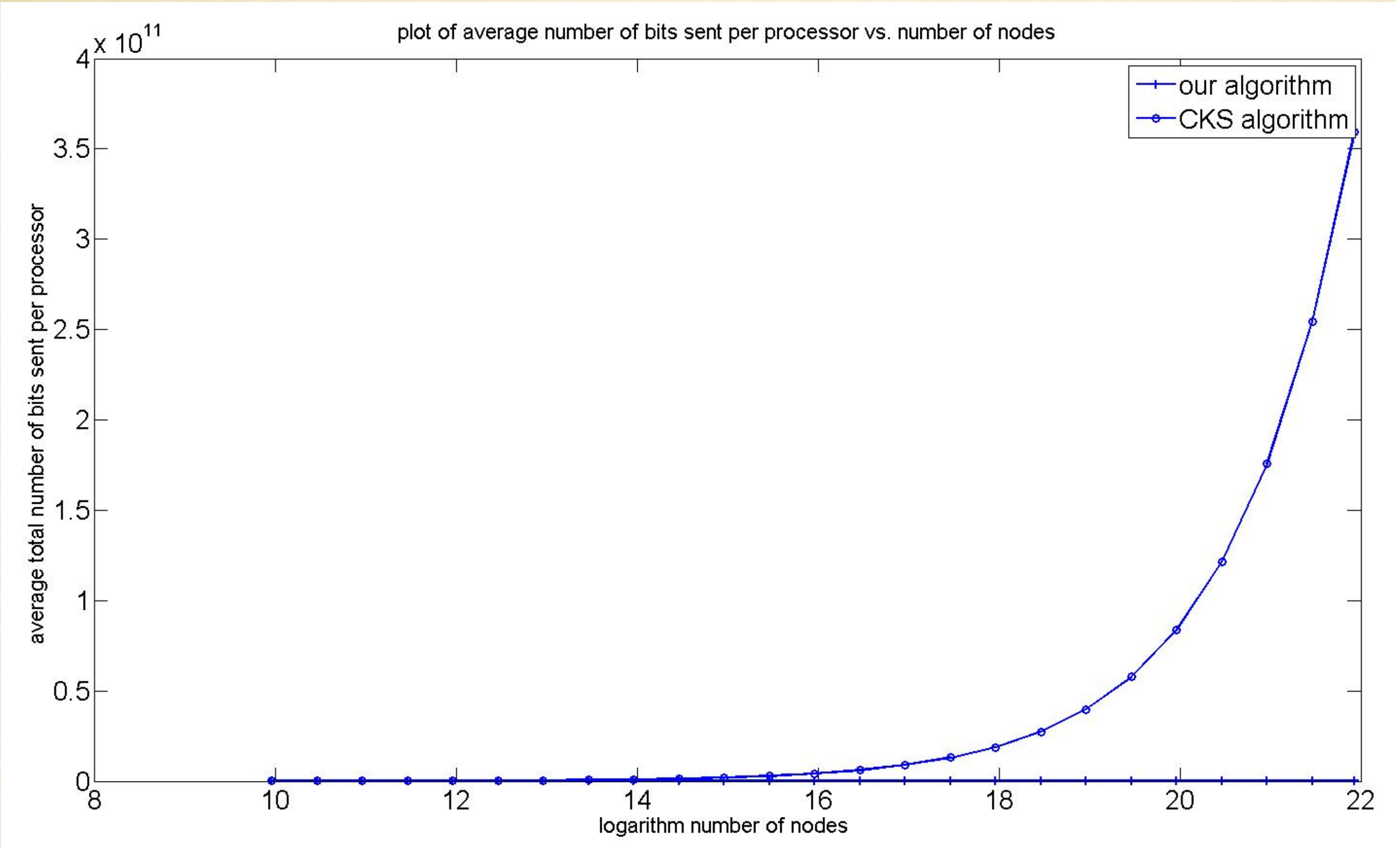
FW: HP/Cloud Computing

- Want: Many local error-corrections instead of one big one
- Idea: Error Correcting Algorithms
- ECA:Computation as ECC:Data

Related Work

- Practical BA
- Amortized Robustness
- Scalable, Rational Secret Sharing
- Scalable, Rational Data Dissemination

Practical BA



Olumuyiwa (M) Oluwasanmi

Amortized Robustness

- Fool me once, shame on you. Fool me $\omega(\log n)$ times, shame on me.
- **Goal:** Limit adversarial corruption of messages in a communication network where a majority of nodes are good
- **Problem:** assigning fault when communication involves multiple processors

IARPA
BE THE FUTURE

Jeffrey Knockel

Scalable Rational Secret Sharing

- **Q:** How to enable secret sharing when every player is selfish: wanting to learn the secret, but preferring for others not to learn it?
- **Known:** Achieve with O(n) bits per proc
- **Goal:** Achieve with O(log n) bits per proc
- Application: Mediation in game theory

Yamel Torres-Rodriguez

Rational Gossiping

- Want to disseminate a large file to large set of players
- File is broken into pieces, sent by a seeder
- Each player is selfish
  - Only shares pieces if in best interest
  - Leaves when it receives all the pieces

Nathan Hjelmn

Collaborators

- Current Students: Olumuyiwa Oluwasanmi, Jeffrey Knockel, Yamel Torres-Rodriguez, Nathan Hjelmn
- Former Students
  - PhD: Vishal Sanwalani (Waterloo/MSR), Amitabh Trehan (Technion), Navin Rustagi (Rice)
  - Masters: Maxwell Young (Waterloo), Bo Wu (Microsoft)
- Non-students: Valerie King (U. Victoria), Varsha Dasani (UNM), Jim Aspnes (Yale), David Kempe (USC), Erik Vee (Yahoo Research)

Questions?