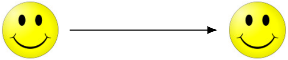
The Adversarial Noise Threshold for Distributed Protocols

William M. Hoza and Leonard J. Schulman
Caltech

January 10, 2016

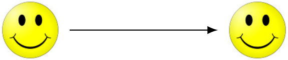
Coding for interactive communication

- Classic coding theory: Alice sends a message to Bob

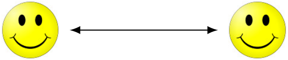


Coding for interactive communication

- Classic coding theory: Alice sends a message to Bob

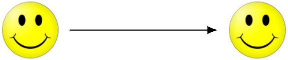


- Coding for *interactive* communication: Alice and Bob have a conversation

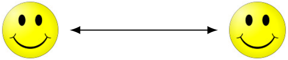


Coding for interactive communication

- Classic coding theory: Alice sends a message to Bob



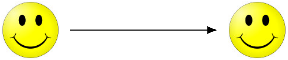
- Coding for *interactive* communication: Alice and Bob have a conversation



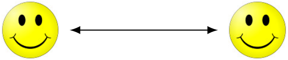
- E.g. chess over the phone

Coding for interactive communication

- Classic coding theory: Alice sends a message to Bob



- Coding for *interactive* communication: Alice and Bob have a conversation



- E.g. chess over the phone
- "What?"

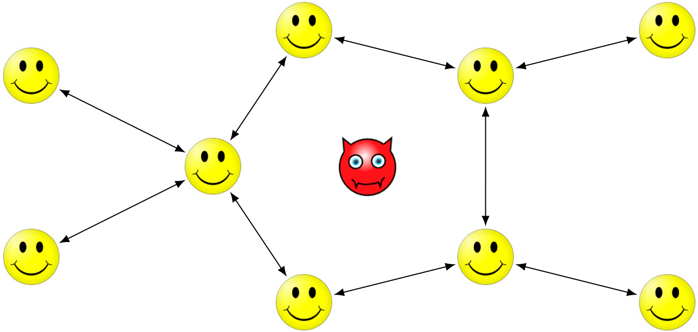
Topic of this talk

- Challenge: Protect distributed computation from channel noise
- Coding for interactive *multiparty* communication

Outline

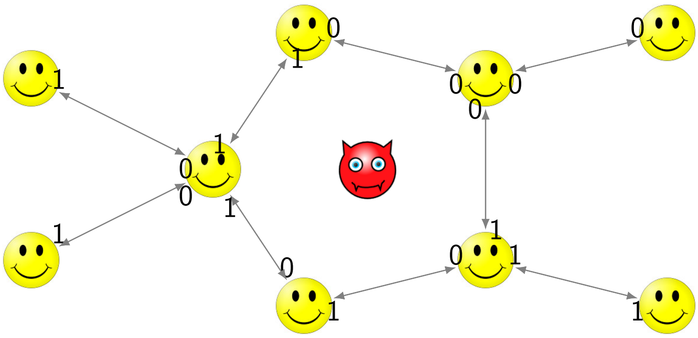
1. The model
2. Related work
3. Main result
4. Proof sketch
5. Directions for further research
6. Acknowledgements

The model



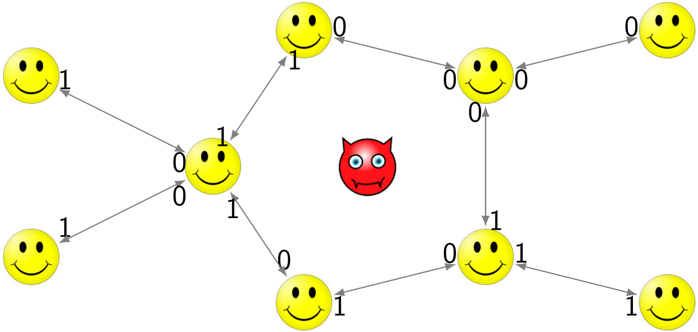
- Parties $P_1, \ldots, P_n$ connected by m two-way communication channels (arbitrary, static topology)
- Input to a computational problem split up as $x = (x_1, \ldots, x_n)$, with P_i receiving x_i

The model (2)



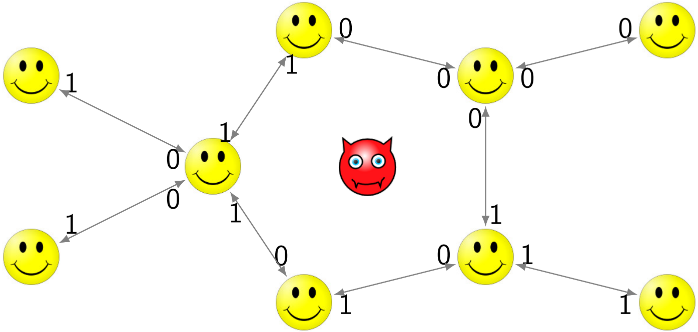
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



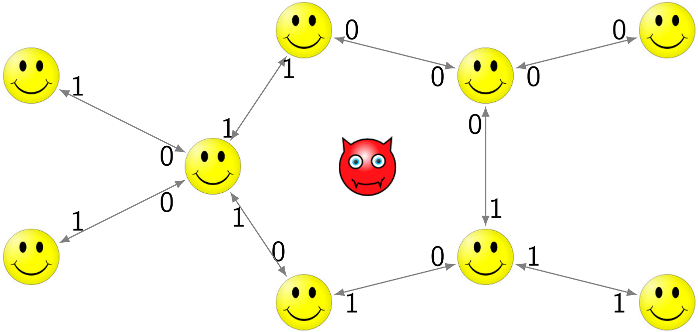
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



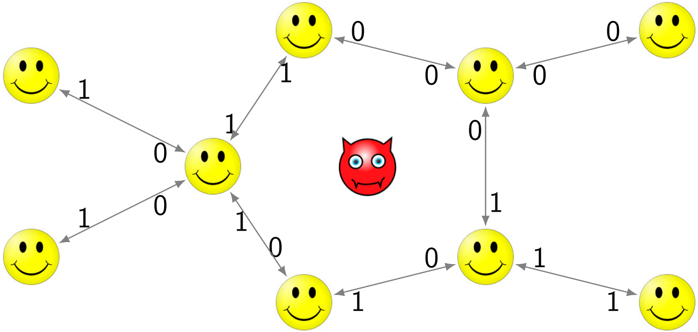
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



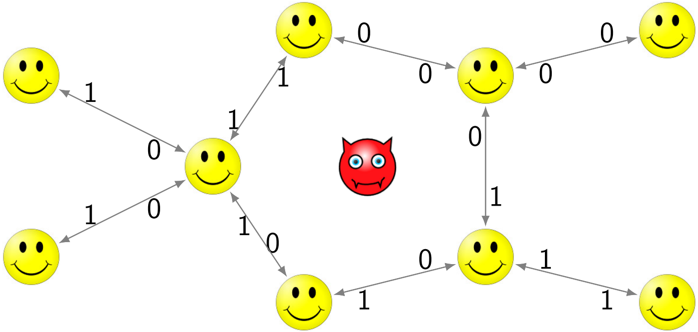
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



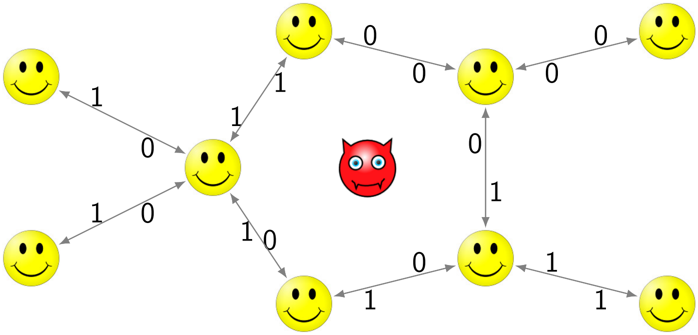
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



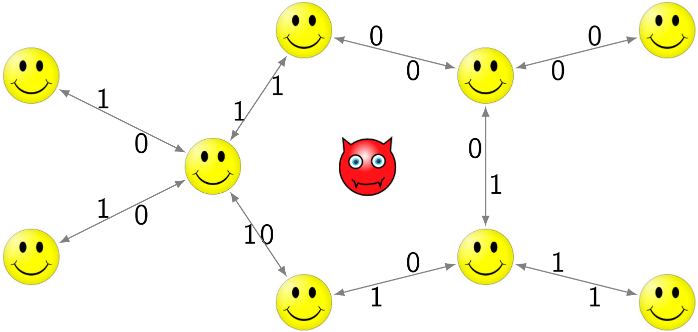
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



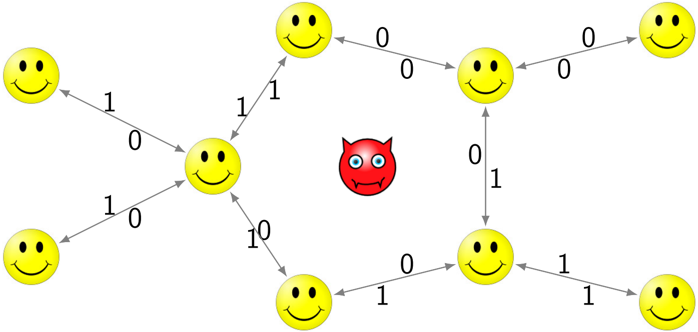
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



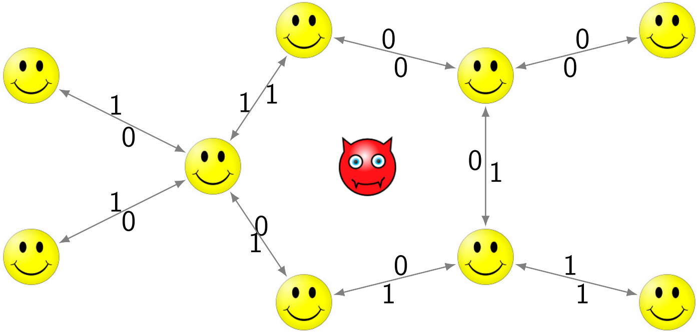
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



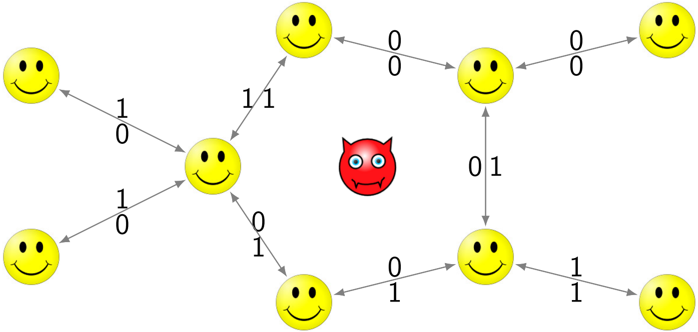
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



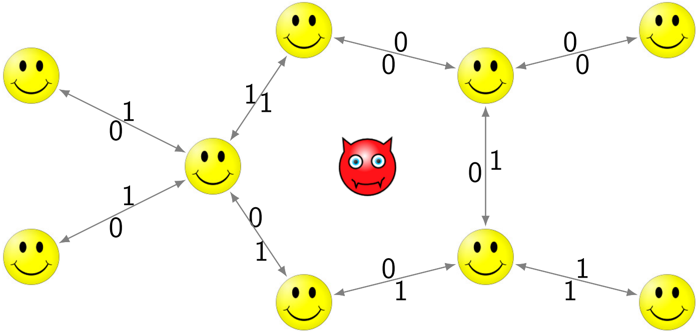
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



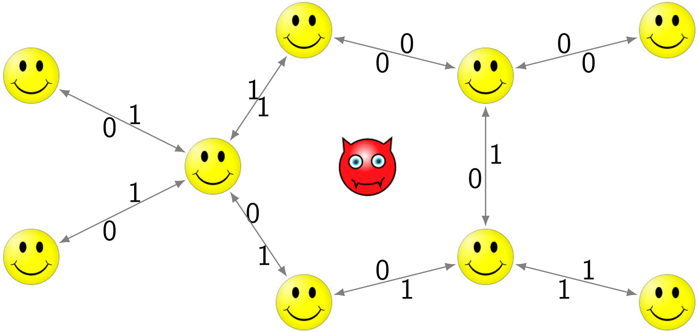
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



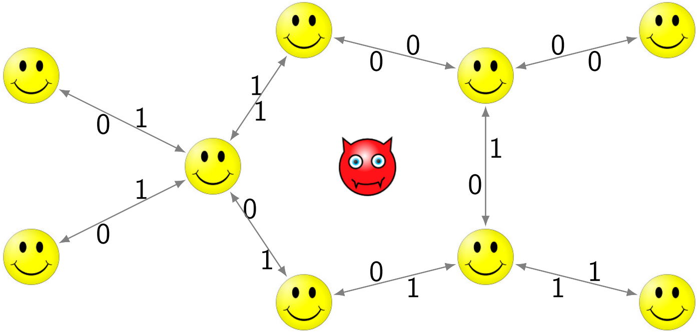
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



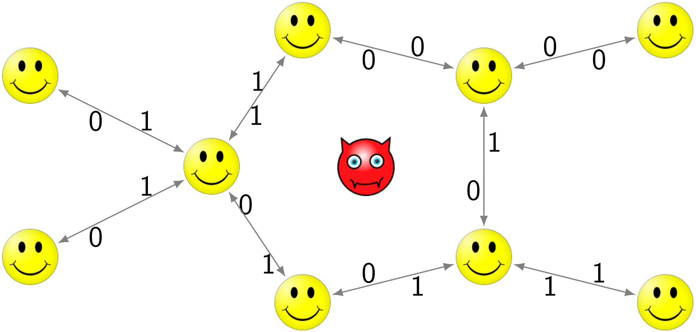
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



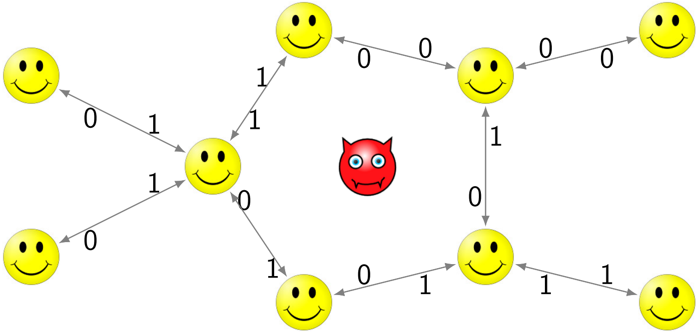
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



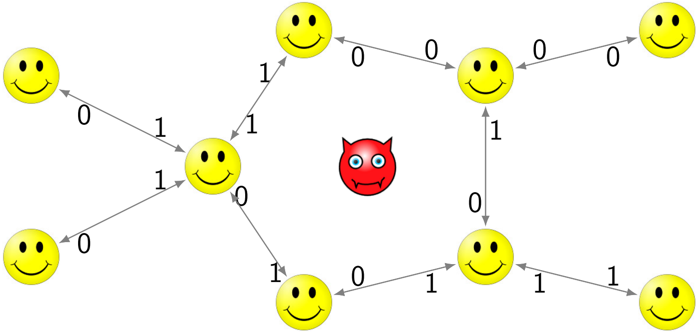
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



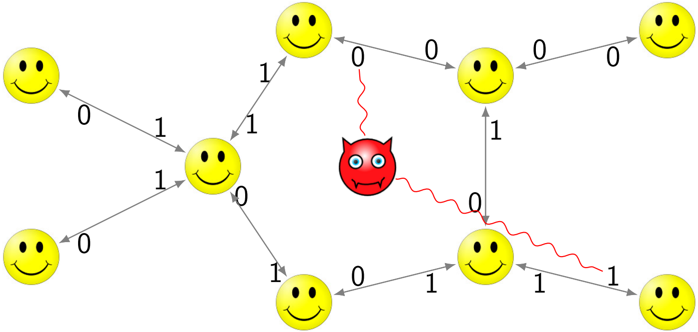
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



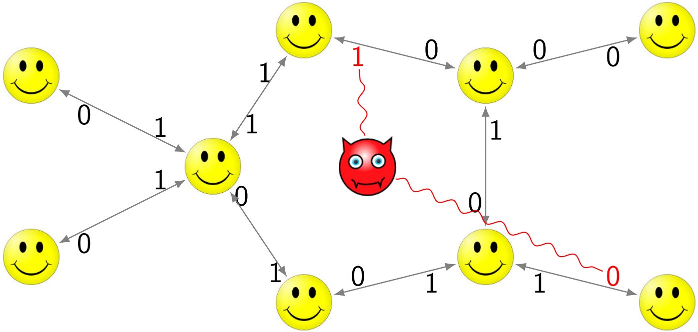
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



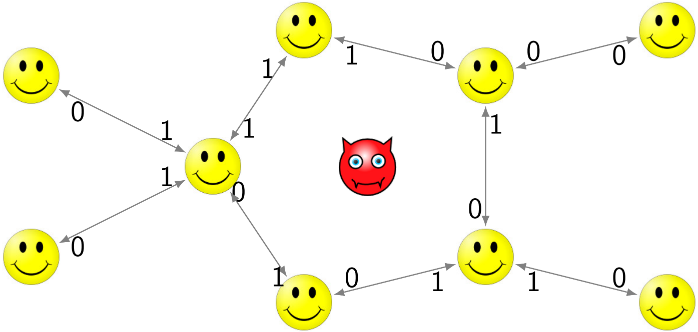
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



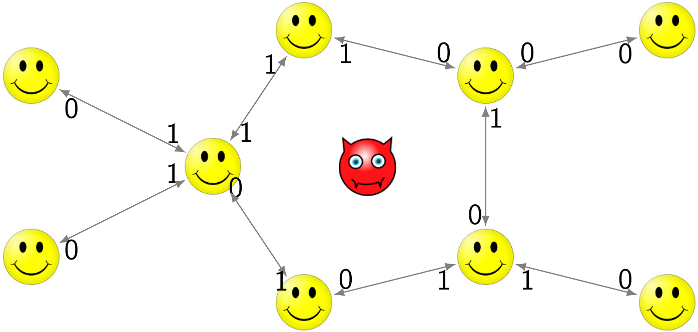
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



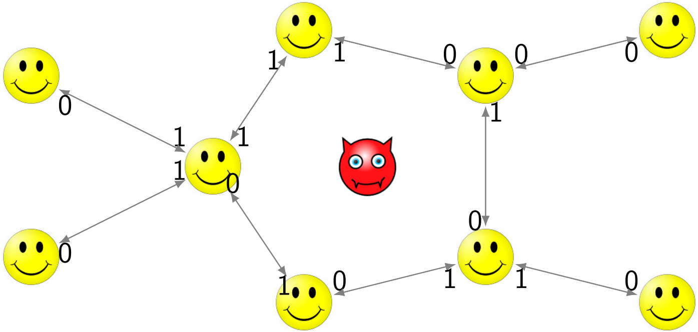
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



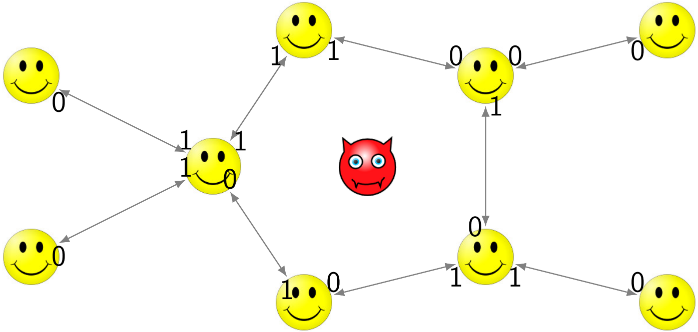
- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (2)



- Synchronous messaging: Two bits per edge per round (one in each direction)
- Adversary sees all, flips bits as she sees fit

The model (3)

- *Protocol*: n-tuple of (possibly probabilistic) algorithms that the parties use to decide what bits to transmit
- After T rounds, each party gives an output
- T is the *round complexity* of the protocol (known by all parties)

The model (4)

- Goal: Design *compiler* C: transforms protocol π into simulation protocol $\widetilde{\pi} = C(\pi)$ which tolerates a high error rate

$$\text{Error rate} = \frac{\text{Total number of bits flipped}}{\text{Total number of bits transmitted}}$$

The model (4)

- Goal: Design *compiler* C: transforms protocol π into simulation protocol $\widetilde{\pi} = C(\pi)$ which tolerates a high error rate

$$\text{Error rate} = \frac{\text{Total number of bits flipped}}{\text{Total number of bits transmitted}}$$

- Secondary goal: $\widetilde{\pi}$ should have low round complexity

Related work: Stochastic errors

- Variant model: Channels are independent BSCs with capacity $c > 0$

Related work: Stochastic errors

- Variant model: Channels are independent BSCs with capacity $c > 0$
- Rajagopalan and Schulman '94: Every π can be compiled into simulation $\widetilde{\pi}$ with

Related work: Stochastic errors

- Variant model: Channels are independent BSCs with capacity $c > 0$
- Rajagopalan and Schulman '94: Every π can be compiled into simulation $\widetilde{\pi}$ with
  - Round complexity $\mathcal{O}(T \log(\text{max degree} + 1))$

Related work: Stochastic errors

- Variant model: Channels are independent BSCs with capacity $c > 0$
- Rajagopalan and Schulman '94: Every π can be compiled into simulation $\widetilde{\pi}$ with
  - Round complexity $\mathcal{O}(T \log(\text{max degree} + 1))$
  - Failure probability $e^{-\Omega(T)}$

Related work: Stochastic errors

- Variant model: Channels are independent BSCs with capacity $c > 0$
- Rajagopalan and Schulman '94: Every π can be compiled into simulation $\widetilde{\pi}$ with
  - Round complexity $\mathcal{O}(T \log(\text{max degree} + 1))$
  - Failure probability $e^{-\Omega(T)}$
- Still open: Optimal asymptotic round complexity?

Related work: Stochastic errors

- Variant model: Channels are independent BSCs with capacity $c > 0$
- Rajagopalan and Schulman '94: Every π can be compiled into simulation $\widetilde{\pi}$ with
  - Round complexity $\mathcal{O}(T \log(\text{max degree} + 1))$
  - Failure probability $e^{-\Omega(T)}$
- Still open: Optimal asymptotic round complexity?
  - But see Alon et. al '15, Braverman et. al '16

Related work: Stochastic errors

- Variant model: Channels are independent BSCs with capacity $c > 0$
- Rajagopalan and Schulman '94: Every π can be compiled into simulation $\widetilde{\pi}$ with
  - Round complexity $\mathcal{O}(T \log(\text{max degree} + 1))$
  - Failure probability $e^{-\Omega(T)}$
- Still open: Optimal asymptotic round complexity?
  - But see Alon et. al '15, Braverman et. al '16
- See also Gelles, Moitra, Sahai '11, '14

Related work: Aynchronous communication

- Variant model: One message in-flight at a time

Related work: Aynchronous communication

- Variant model: One message in-flight at a time
- Jain, Kalai, Lewko '15: In graphs where one party is connected to all others, every "semi-adaptive" π can be compiled into simulation $\widetilde{\pi}$ with properties:

Related work: Aynchronous communication

- Variant model: One message in-flight at a time
- Jain, Kalai, Lewko '15: In graphs where one party is connected to all others, every "semi-adaptive" π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/n)$

Related work: Aynchronous communication

- Variant model: One message in-flight at a time
- Jain, Kalai, Lewko '15: In graphs where one party is connected to all others, every "semi-adaptive" π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/n)$
  - Round complexity $\mathcal{O}(T)$

Related work: Aynchronous communication

- Variant model: One message in-flight at a time
- Jain, Kalai, Lewko '15: In graphs where one party is connected to all others, every "semi-adaptive" π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/n)$
  - Round complexity $\mathcal{O}(T)$
- See also Lewko and Vitercik '15

Context: Elementary negative result

- Say simulation runs on $G = (V, E)$ with edge connectivity k

Context: Elementary negative result

- Say simulation runs on $G = (V, E)$ with edge connectivity k
- **Cannot** tolerate error rate $k/|E|$

Context: Elementary negative result

- Say simulation runs on $G = (V, E)$ with edge connectivity k
- **Cannot** tolerate error rate $k/|E|$
  - On some graphs, this is only $O(1/n^2)$!

Context: Elementary negative result

- Say simulation runs on $G = (V, E)$ with edge connectivity k
- **Cannot** tolerate error rate $k/|E|$
  - On some graphs, this is only $O(1/n^2)$!
- Proof: Adversary attacks k edges to effectively disconnect graph

Figure: "Concentrate all your fire on the nearest starship."

Our main result

- Theorem: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:

Our main result

- Theorem: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
    - Tolerates adversarial error rate $\Omega(1/n)$

Our main result

- Theorem: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/n)$
  - Runs on *subgraph* $\widetilde{G} = (V, \widetilde{E})$ of original graph $G = (V, E)$

Our main result

- Theorem: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/n)$
  - Runs on *subgraph* $\widetilde{G} = (V, \widetilde{E})$ of original graph $G = (V, E)$
  - Round complexity $\mathcal{O}\left(\frac{m \log n}{n} T\right)$

Our main result

- Theorem: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/n)$
  - Runs on *subgraph* $\widetilde{G} = (V, \widetilde{E})$ of original graph $G = (V, E)$
  - Round complexity $\mathcal{O}\left(\frac{m \log n}{n} T\right)$
- Error rate within constant factor of optimal

Our main result

- Theorem: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/n)$
  - Runs on *subgraph* $\widetilde{G} = (V, \widetilde{E})$ of original graph $G = (V, E)$
  - Round complexity $\mathcal{O}\left(\frac{m \log n}{n} T\right)$
- Error rate within constant factor of optimal
- $\widetilde{G}$ is sparse ($\mathcal{O}(n)$ edges)

Our main result

- Theorem: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/n)$
  - Runs on *subgraph* $\widetilde{G} = (V, \widetilde{E})$ of original graph $G = (V, E)$
  - Round complexity $\mathcal{O}\left(\frac{m \log n}{n} T\right)$
- Error rate within constant factor of optimal
- $\widetilde{G}$ is sparse ($\mathcal{O}(n)$ edges)
- Round complexity within $O(k \log n)$ of optimal

Our main result

- Theorem: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/n)$
  - Runs on *subgraph* $\widetilde{G} = (V, \widetilde{E})$ of original graph $G = (V, E)$
  - Round complexity $\mathcal{O}\left(\frac{m \log n}{n} T\right)$
- Error rate within constant factor of optimal
- $\widetilde{G}$ is sparse ($\mathcal{O}(n)$ edges)
- Round complexity within $O(k \log n)$ of optimal
  - $k =$ edge connectivity of G

Coding-theoretic ingredient of main result

- Lemma 1: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:

Coding-theoretic ingredient of main result

- Lemma 1: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/m)$

Coding-theoretic ingredient of main result

- Lemma 1: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/m)$
  - Round complexity $\mathcal{O}(T)$

Coding-theoretic ingredient of main result

- Lemma 1: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/m)$
  - Round complexity $\mathcal{O}(T)$
- Proof: Make small tweaks to construction/analysis in Rajagopalan and Schulman '94

Coding-theoretic ingredient of main result

- Lemma 1: Every π can be compiled into simulation $\widetilde{\pi}$ with properties:
  - Tolerates adversarial error rate $\Omega(1/m)$
  - Round complexity $\mathcal{O}(T)$
- Proof: Make small tweaks to construction/analysis in Rajagopalan and Schulman '94
- "RS compiler"

Proof outline of main result

1. Use multicommodity flow methods to route messages of π through cut sparsifier

No error correction

Proof outline of main result

1. Use multicommodity flow methods to route messages of π through cut sparsifier
2. Restore a few of the removed edges to improve efficiency

No error correction

Proof outline of main result

1. Use multicommodity flow methods to route messages of π through cut sparsifier
2. Restore a few of the removed edges to improve efficiency
3. Apply RS compiler to this new protocol

No error correction

Proof outline of main result

1. Use multicommodity flow methods to route messages of π through cut sparsifier
2. Restore a few of the removed edges to improve efficiency

No error correction

3. Apply RS compiler to this new protocol

- Second sparse subnetwork has $\widetilde{m} \in \mathcal{O}(n)$ edges

Proof outline of main result

1. Use multicommodity flow methods to route messages of π through cut sparsifier
2. Restore a few of the removed edges to improve efficiency
3. Apply RS compiler to this new protocol

No error correction

- Second sparse subnetwork has $\widetilde{m} \in \mathcal{O}(n)$ edges
- So final protocol tolerates error rate $\Omega(1/\widetilde{m}) = \Omega(1/n)$

Proof of main result: Step 1

1. Use multicommodity flow methods to route messages of π through cut sparsifier

---

- Special case of spectral sparsification theorem by de Carli Silva, Harvey Sato '15:

Proof of main result: Step 1

1. Use multicommodity flow methods to route messages of π through cut sparsifier

---

- Special case of spectral sparsification theorem by de Carli Silva, Harvey Sato '15:
- Every connected, undirected $G = (V, E)$ has a subgraph $\widetilde{G} = (V, \widetilde{E})$ with $\mathcal{O}(n)$ edges s.t. for each $U \subseteq V$,

$$\frac{10m}{n} \left|\widetilde{\delta}(U)\right| \geq |\delta(U)|. \tag{1}$$

  - $\delta(U)$ is the set of edges in G crossing U
  - $\widetilde{\delta}(U)$ is the set of edges in $\widetilde{G}$ crossing U

Proof of main result: Step 1

1. Use multicommodity flow methods to route messages of π through cut sparsifier

---

- Special case of spectral sparsification theorem by de Carli Silva, Harvey Sato '15:
- Every connected, undirected $G = (V, E)$ has a subgraph $\widetilde{G} = (V, \widetilde{E})$ with $\mathcal{O}(n)$ edges s.t. for each $U \subseteq V$,

$$\frac{10m}{n} \left| \widetilde{\delta}(U) \right| \geq |\delta(U)|. \tag{1}$$

  - $\delta(U)$ is the set of edges in G crossing U
  - $\widetilde{\delta}(U)$ is the set of edges in $\widetilde{G}$ crossing U
- Builds on Batson et. al '09, Benczúr and Karger '96

Proof of main result: Step 1 (cont.)

1. Use multicommodity flow methods to route messages of π through cut sparsifier

---

- Apply approximate multicommodity max-flow min-cut theorem (Linial, London, Rabinovich '95) + randomized rounding

Proof of main result: Step 1 (cont.)

1. Use multicommodity flow methods to route messages of π through cut sparsifier

---

- Apply approximate multicommodity max-flow min-cut theorem (Linial, London, Rabinovich '95) + randomized rounding
- $\Rightarrow$ There exists a set $\mathcal{P}_0$ of $2m$ simple paths through G such that

Proof of main result: Step 1 (cont.)

1. Use multicommodity flow methods to route messages of π through cut sparsifier

---

- Apply approximate multicommodity max-flow min-cut theorem (Linial, London, Rabinovich '95) + randomized rounding
- $\Rightarrow$ There exists a set $\mathcal{P}_0$ of $2m$ simple paths through G such that
  - $\mathcal{P}_0$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$

Proof of main result: Step 1 (cont.)

1. Use multicommodity flow methods to route messages of π through cut sparsifier

---

- Apply approximate multicommodity max-flow min-cut theorem (Linial, London, Rabinovich '95) + randomized rounding
- $\Rightarrow$ There exists a set $\mathcal{P}_0$ of $2m$ simple paths through G such that
  - $\mathcal{P}_0$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$
  - $\mathcal{P}_0$ uses $\mathcal{O}(n)$ edges in total

Proof of main result: Step 1 (cont.)

1. Use multicommodity flow methods to route messages of π through cut sparsifier

---

- Apply approximate multicommodity max-flow min-cut theorem (Linial, London, Rabinovich '95) + randomized rounding
- $\Rightarrow$ There exists a set $\mathcal{P}_0$ of $2m$ simple paths through G such that
  - $\mathcal{P}_0$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$
  - $\mathcal{P}_0$ uses $\mathcal{O}(n)$ edges in total
  - $\mathcal{P}_0$ has congestion at most $\mathcal{O}\left(\frac{m \log m}{n}\right)$

Proof of main result: Step 2

2. Restore a few of the removed edges to improve efficiency

- Lemma 2: For any $G = (V, E)$, there exists a set $\mathcal{P}$ of $2m$ simple paths through G such that

Proof of main result: Step 2

2. Restore a few of the removed edges to improve efficiency

- Lemma 2: For any $G = (V, E)$, there exists a set $\mathcal{P}$ of $2m$ simple paths through G such that
  - $\mathcal{P}$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$

Proof of main result: Step 2

2. Restore a few of the removed edges to improve efficiency

- Lemma 2: For any $G = (V, E)$, there exists a set $\mathcal{P}$ of $2m$ simple paths through G such that
  - $\mathcal{P}$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$
  - $\mathcal{P}$ uses $\mathcal{O}(n)$ edges in total

Proof of main result: Step 2

2. Restore a few of the removed edges to improve efficiency

- Lemma 2: For any $G = (V, E)$, there exists a set $\mathcal{P}$ of $2m$ simple paths through G such that
  - $\mathcal{P}$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$
  - $\mathcal{P}$ uses $\mathcal{O}(n)$ edges in total
  - $\mathcal{P}$ has congestion at most $\mathcal{O}\left(\frac{m \log m}{n}\right)$

Proof of main result: Step 2

2. Restore a few of the removed edges to improve efficiency

- Lemma 2: For any $G = (V, E)$, there exists a set $\mathcal{P}$ of $2m$ simple paths through G such that
  - $\mathcal{P}$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$
  - $\mathcal{P}$ uses $\mathcal{O}(n)$ edges in total
  - $\mathcal{P}$ has congestion at most $\mathcal{O}\left(\frac{m \log m}{n}\right)$
  - $\mathcal{P}$ has dilation (max length) at most $\frac{m \log m}{n}$

Proof of main result: Step 2

2. Restore a few of the removed edges to improve efficiency

- Lemma 2: For any $G = (V, E)$, there exists a set $\mathcal{P}$ of $2m$ simple paths through G such that
  - $\mathcal{P}$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$
  - $\mathcal{P}$ uses $\mathcal{O}(n)$ edges in total
  - $\mathcal{P}$ has congestion at most $\mathcal{O}\left(\frac{m \log m}{n}\right)$
  - $\mathcal{P}$ has dilation (max length) at most $\frac{m \log m}{n}$
- Proof: Start from $\mathcal{P}_0$ (last slide)

Proof of main result: Step 2

2. Restore a few of the removed edges to improve efficiency

- Lemma 2: For any $G = (V, E)$, there exists a set $\mathcal{P}$ of $2m$ simple paths through G such that
  - $\mathcal{P}$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$
  - $\mathcal{P}$ uses $\mathcal{O}(n)$ edges in total
  - $\mathcal{P}$ has congestion at most $\mathcal{O}\left(\frac{m \log m}{n}\right)$
  - $\mathcal{P}$ has dilation (max length) at most $\frac{m \log m}{n}$
- Proof: Start from $\mathcal{P}_0$ (last slide)
- Replace every path that is too long with a single edge from start to finish

Proof of main result: Step 2

2. Restore a few of the removed edges to improve efficiency

- Lemma 2: For any $G = (V, E)$, there exists a set $\mathcal{P}$ of $2m$ simple paths through G such that
  - $\mathcal{P}$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$
  - $\mathcal{P}$ uses $\mathcal{O}(n)$ edges in total
  - $\mathcal{P}$ has congestion at most $\mathcal{O}\left(\frac{m \log m}{n}\right)$
  - $\mathcal{P}$ has dilation (max length) at most $\frac{m \log m}{n}$
- Proof: Start from $\mathcal{P}_0$ (last slide)
- Replace every path that is too long with a single edge from start to finish
  - Only adds $\mathcal{O}(n)$ edges

Proof of main result: Step 2

2. Restore a few of the removed edges to improve efficiency

- Lemma 2: For any $G = (V, E)$, there exists a set $\mathcal{P}$ of $2m$ simple paths through G such that
  - $\mathcal{P}$ contains two paths $P_i \leftrightsquigarrow P_j$ for each $\{P_i, P_j\} \in E$
  - $\mathcal{P}$ uses $\mathcal{O}(n)$ edges in total
  - $\mathcal{P}$ has congestion at most $\mathcal{O}\left(\frac{m \log m}{n}\right)$
  - $\mathcal{P}$ has dilation (max length) at most $\frac{m \log m}{n}$
- Proof: Start from $\mathcal{P}_0$ (last slide)
- Replace every path that is too long with a single edge from start to finish
  - Only adds $\mathcal{O}(n)$ edges
  - Only increases congestion by 1 $\square$

Proof of main result: Step 3

3. Apply RS compiler to this new protocol

- Theorem (Leighton, Maggs, Rao '94): For any set $\mathcal{P}$ of simple paths, there is a schedule for sending one packet along each path in $\mathcal{P}$ in a total of $\mathcal{O}(\text{congestion} + \text{dilation})$ time steps.

Proof of main result: Step 3

3. Apply RS compiler to this new protocol

- Theorem (Leighton, Maggs, Rao '94): For any set $\mathcal{P}$ of simple paths, there is a schedule for sending one packet along each path in $\mathcal{P}$ in a total of $\mathcal{O}(\text{congestion} + \text{dilation})$ time steps.
- Proof of main result:

$$\pi \quad \overset{\text{Sparsifying compiler}}{\longmapsto} \quad \pi' \quad \overset{\text{RS compiler}}{\longmapsto} \quad \widetilde{\pi}$$

Proof of main result: Step 3

3. Apply RS compiler to this new protocol

- Theorem (Leighton, Maggs, Rao '94): For any set $\mathcal{P}$ of simple paths, there is a schedule for sending one packet along each path in $\mathcal{P}$ in a total of $\mathcal{O}(\text{congestion} + \text{dilation})$ time steps.
- Proof of main result:

$$\pi \xmapsto{\text{Sparsifying compiler}} \pi' \xmapsto{\text{RS compiler}} \widetilde{\pi}$$

- Each round of π is simulated by $\mathcal{O}(\frac{m \log n}{n})$ steps in π' by using the paths of Lemma 2 $\square$

Directions for further research

- Computational efficiency?

Directions for further research

- Computational efficiency?
- Open question: Can round complexity be improved by factor of $k \log n$?

Directions for further research

- Computational efficiency?
- Open question: Can round complexity be improved by factor of $k \log n$?
- Directed graphs

Directions for further research

- Computational efficiency?
- Open question: Can round complexity be improved by factor of $k \log n$?
- Directed graphs
  - Optimal error rate is $\Theta(\frac{1}{s})$, where s is the smallest number of edges in any subgraph with same reachability relation

Directions for further research

- Computational efficiency?
- Open question: Can round complexity be improved by factor of $k \log n$?
- Directed graphs
  - Optimal error rate is $\Theta(\frac{1}{s})$, where s is the smallest number of edges in any subgraph with same reachability relation
  - Open question: How to avoid large round complexity blowup?

Acknowledgements

- Thanks, Caltech SURF program!
- Thanks, Nellie Bergen and Adrian Foster Tillotson!
- Thanks, Achievement Rewards for College Scientists Foundation!
- Thanks, ACM!
- Thanks, SIAM!
- **Thanks, listeners!**