Designing Algorithms for Highly Dynamic Networks

John Augustine
IIT Madras

August 11, 2014
Fault Tolerant Distributed Algorithms — in the last 35 years

- Failures are common in large networks (like the Internet).
- Network should not be crippled by failures.

How can a ship survive decaying planks?

Theorem (Fisher, Lynch, and Paterson — JACM, 1985)

Distributed agreement is impossible in an asynchronous network with one faulty node.

6/13 Dijkstra Prize winning articles (incl. above) deal with fault tolerance.
Fault Tolerant Distributed Algorithms
— in the last 35 years

- Failures are common in large networks (like the Internet).
- Network should not be crippled by failures.

How can a ship survive decaying planks?

Theorem (Fisher, Lynch, and Paterson — JACM, 1985)

Distributed agreement is impossible in an asynchronous network with one faulty node.

6/13 Dijkstra Prize winning articles (incl. above) deal with fault tolerance.
Fault Tolerant Distributed Algorithms
— in the last 35 years

- Failures are common in large networks (like the Internet).
- Network should not be crippled by failures.

How can a ship survive decaying planks?

Theorem (Fisher, Lynch, and Paterson — JACM, 1985)

Distributed agreement is impossible in an asynchronous network with one faulty node.

6/13 Dijkstra Prize winning articles (incl. above) deal with fault tolerance.
The Ship of Theseus
— a paradox

Does a ship remain the same when each plank has been replaced?

— Ancient Greeks
Today’s Networks

**Internet:**
new users, new servers, new ISPs

**Mobile Networks:**
new areas, new sensors

**Peer-to-Peer Networks:**
new users, new sessions

**Data centres:**
virtualisation and server consolidation

Real networks highly dynamic with heavy churn, and yet work well.

Theory is lagging. How to make it lead?
Today’s Networks

Internet:
new users, new servers, new ISPs

Mobile Networks:
new areas, new sensors

Peer-to-Peer Networks:
new users, new sessions

Data centres:
virtualisation and server consolidation

Real networks highly dynamic with heavy churn, and yet work well.

Theory is lagging. How to make it lead?
Today’s Networks

Internet:
new users, new servers, new ISPs

Mobile Networks:
new areas, new sensors

Peer-to-Peer Networks:
new users, new sessions

Data centres:
virtualisation and server consolidation

Real networks highly dynamic with heavy churn, and yet work well.

Theory is lagging. How to make it lead?
Our Wishlist

Ad infinitum. Well, because life goes on . . .

- Lots of recent works, stark contrast with early works

Edge rewiring. Node mobility, restructuring induced by link failures

- Dell & Wattenhofer (DIAL-POMC 05); Avin, Koucky, & Lotker (ICALP 08); Clementi, Monti, Pasquale, & Silvestri (IPDPS 09); Baumann, Crescenzi, & Fraigniaud (PODC 09), Kuhn, Lynch, & Oshmann (STOC 11); Das Sarma, Molla, & Pandurangan (DISC 12); Dutta, Pandurangan, Rajaraman, Sun, & Viola (SODA 13).

Node churning. Area specific mobile networks, failures and recoveries, new users in p2p

- Rhea, Geels, Roscoe, & Kubiatowicz (ATEC 04); Kuhn, Schmid, & Wattenhofer (DC 10); —, Pandurangan, Robinson, & Upfal (SODA 12); —, Molla, Morsy, Pandurangan, Robinson, & Upfal (SPAA 13); —, Pandurangan, & Robinson (PODC 13); Klappenecker & Lee (ICPADS 13); —, Pandurangan, Robinson (submitted); —, Kulkarni, Sivasubramanian (submitted)

Adversarial. Need robust algorithms even if nature of dynamics change
Our Model — the setting

**Synchronous.** All nodes follow the same clock

**Adversarial Dynamism.** An oblivious adversary (knows algorithm, but not the coin toss outcomes) designs churn and edge dynamics

$$G = (G^0, G^1, \ldots, G^r, \ldots)$$

**Unique ID and single lifetime.** Each node comes in once and leaves at most once.

**Churn.** Up to $C(n)$ nodes leave/join the network per round

**Stable Network Size.** Number of nodes $n$ unchanged over time (In each round: $\#$ churned out = $\#$ churned in)

**Edge Dynamics.** Topology can change round to round
Our Model — the setting

**Synchronous.** All nodes follow the same clock

**Adversarial Dynamism.** An oblivious adversary (knows algorithm, but not the coin toss outcomes) designs churn and edge dynamics

\[ G = (G^0, G^1, \ldots, G^r, \ldots) \]

**Unique ID and single lifetime.** Each node comes in once and leaves at most once.

**Churn.** Up to \( C(n) \) nodes leave/join the network per round

**Stable Network Size.** Number of nodes \( n \) unchanged over time (In each round: \( \# \) churned out = \( \# \) churned in)

**Edge Dynamics.** Topology can change round to round
Our Model — the setting

**Synchronous.** All nodes follow the same clock

**Adversarial Dynamism.** An oblivious adversary (knows algorithm, but not the coin toss outcomes) designs churn and edge dynamics

\[ G = (G^0, G^1, \ldots, G^r, \ldots) \]

**Unique ID and single lifetime.** Each node comes in once and leaves at most once.

**Churn.** Up to \( C(n) \) nodes leave/join the network per round

**Stable Network Size.** Number of nodes \( n \) unchanged over time

(In each round: \( \# \) churned out = \( \# \) churned in)

**Edge Dynamics.** Topology can change round to round
Our Model — the setting

**Synchronous.** All nodes follow the same clock

**Adversarial Dynamism.** An oblivious adversary (knows algorithm, but not the coin toss outcomes) designs churn and edge dynamics

\[ \mathcal{G} = (G^0, G^1, \ldots, G^r, \ldots) \]

**Unique ID and single lifetime.** Each node comes in once and leaves at most once.

**Churn.** Up to \( C(n) \) nodes leave/join the network per round

**Stable Network Size.** Number of nodes \( n \) unchanged over time (In each round: \( \# \) churned out = \( \# \) churned in)

**Edge Dynamics.** Topology can change round to round
Our Model
— high connectivity assumption

Each $G^r = (V^r, E^r)$ is a $d$-regular $\alpha$-expander graph.

A graph $G = (V, E)$ is an $\alpha$-expander if

For every $S \subset V$ such that $|S| \leq |V|/2$,

$$|\Gamma(S)| \geq \alpha \cdot |S|,$$

where $\Gamma(S) = \{ u \in V \setminus S \mid \exists s \in S \text{ such that } (s, u) \in E \}$.

- Common assumption even in static fault tolerant networks
  E.g., Dwork, Peleg, Pippenger, & Upfal (SICOMP 88)

- There are attempts to maintain expansion, but not our concern today
Our Model
— high connectivity assumption

Each \( G^r = (V^r, E^r) \) is a \( d \)-regular \( \alpha \)-expander graph.

A graph \( G = (V, E) \) is an \( \alpha \)-expander if

For every \( S \subseteq V \) such that \(|S| \leq |V|/2\),

\[
|\Gamma(S)| \geq \alpha \cdot |S|,
\]

where \( \Gamma(S) = \{ u \in V \setminus S \mid \exists s \in S \text{ such that } (s, u) \in E \} \).

- Common assumption even in static fault tolerant networks
  E.g., Dwork, Peleg, Pippenger, & Upfal (SICOMP 88)

- There are attempts to maintain expansion, but not our concern today
Our Model
— high connectivity assumption

Each $G^r = (V^r, E^r)$ is a $d$-regular $\alpha$-expander graph.

A graph $G = (V, E)$ is an $\alpha$-expander if

For every $S \subset V$ such that $|S| \leq |V|/2$,

$$|\Gamma(S)| \geq \alpha \cdot |S|,$$

where $\Gamma(S) = \{u \in V \setminus S \mid \exists s \in S \text{ such that } (s, u) \in E\}$.

- Common assumption even in static fault tolerant networks
  E.g., Dwork, Peleg, Pippenger, & Upfal (SICOMP 88)
- There are attempts to maintain expansion, but not our concern today
Our Model
— the setting

Message Passing CONGEST. At most $O(\text{polylog}(n))$ bits per round per recipient.

Overlay Edges. When node enters, it is seeded with overlay neighbours.

Direct Communication. When recipient address is known and not churned out (not guaranteed).

(Not always required.)
Our Model
— the setting

**Message Passing CONGEST.** At most $O(\text{polylog}(n))$ bits per round per recipient.

**Overlay Edges.** When node enters, it is seeded with overlay neighbours.

**Direct Communication.** When recipient address is known and not churned out (not guaranteed).

(Not always required.)
Techniques

Flooding Based.

Churn limit \( C(n) = \epsilon n \)

\[ \rightarrow \text{Agreement} \quad (—, \text{Pandurangan, Robinson, Upfal (SODA 12)}) \]

\[ \rightarrow \text{Leader Election} \quad (—, \text{Kulkarni, Sivasubramaniam (submitted)}) \]

Random Walks Based Sampling.

Churn limit \( C(n) = n / \text{polylog}(n) \)

\[ \rightarrow \text{Byzantine Agreement} \quad (—, \text{Pandurangan, Robinson (PODC 13)}) \]

\[ \rightarrow \text{Byzantine Leader Election} \quad (—, \text{Pandurangan, Robinson, Roche, Upfal (submitted)}) \]

\[ \rightarrow \text{Storing and Searching} \quad (—, \text{Molla, Morsy, Pandurangan, Robinson, Upfal (SPAA 13)}) \]
Techniques

Flooding Based.

Churn limit $C(n) = \epsilon n$

$\rightarrow$ Agreement (—, Pandurangan, Robinson, Upfal (SODA 12))

$\rightarrow$ Leader Election (—, Kulkarni, Sivasubramaniam (submitted))

Random Walks Based Sampling.

Churn limit $C(n) = n/\text{polylog}(n)$

$\rightarrow$ Byzantine Agreement (—, Pandurangan, Robinson (PODC 13))

$\rightarrow$ Byzantine Leader Election (—, Pandurangan, Robinson, Roche, Upfal (submitted))

$\rightarrow$ Storing and Searching (—, Molla, Morsy, Pandurangan, Robinson, Upfal (SPAA 13))
Techniques

Flooding Based.

Churn limit $C(n) = \epsilon n$

\[\rightarrow\] Agreement (—, Pandurangan, Robinson, Upfal (SODA 12))

\[\rightarrow\] Leader Election (—, Kulkarni, Sivasubramaniam (submitted))

Random Walks Based Sampling.

Churn limit $C(n) = n/\text{polylog}(n)$

\[\rightarrow\] Byzantine Agreement (—, Pandurangan, Robinson (PODC 13))

\[\rightarrow\] Byzantine Leader Election (—, Pandurangan, Robinson, Roche, Upfal (submitted))

\[\rightarrow\] Storing and Searching (—, Molla, Morsy, Pandurangan, Robinson, Upfal (SPAA 13))
Flooding Based Agreement Protocol
Almost Everywhere Agreement Problem (Churning Network)

Consider a dynamic network \(\{G^0, G^1, \ldots, G^r, \ldots\}\)

\[ \text{Churn } C(n) = \epsilon n \]

\[ \text{No direct communication} \]

- Every \(v \in V^0\) is given an input bit value from \(\{0, 1\}\).
- Every node has a special output bit variable that can be written to exactly once.
  - Until the node writes, it is “undecided.”
  - When it writes, it “decides” on the value written.
Almost Everywhere Agreement Problem (Churning Network)

Consider a dynamic network \( \{ G^0, G^1, \ldots, G^r, \ldots \} \)

\( \xrightarrow{\text{Churn}} C(n) = \epsilon n \)

\( \xrightarrow{\text{No direct communication}} \)

- Every \( v \in V^0 \) is given an input bit value from \( \{0, 1\} \).
- Every node has a special output bit variable that can be written to exactly once.
  - Until the node writes, it is “undecided.”
  - When it writes, it “decides” on the value written.
Almost Everywhere Agreement Problem (Churning Network)

*Stable Agreement* is reached in $R$ rounds if

**Validity:** No node decides on a bit value other than some value in
{0, 1} that some node proposed in round 1.

**Almost Everywhere Agreement in round $r$:** At least $(1 - \beta)n$ nodes in $V^r$ decide on the same bit value for some fixed (but small) $\beta > 0$. Our algorithm requires $\beta > \frac{\epsilon(1 + \alpha)}{\alpha}$.

**Stability:** For every round $r \geq R$, almost everywhere agreement is maintained.
Why naive flooding does not work?
Why flooding does not work?
Why flooding does not work?
Why flooding does not work?
Why flooding does not work?
Why flooding does not work?
Why flooding does not work?
Why flooding does not work?
Why flooding does not work?
Why flooding does not work?
Why flooding does not work?
Some Definitions

Definition
The dynamic distance from a node \( u \in V^r \) to a node \( v \) starting at round \( r \) is the number of rounds it takes for flooded messages from \( u \) to reach \( v \) starting at round \( r \).

Definition
The influence set of \( u \) after \( R \) rounds starting at round \( r \) is the set of nodes in \( V^{r+R} \) whose dynamic distance from \( u \) starting at round \( r \) is at most \( R \).

Influence set of \( U \subseteq V^r \) = \( \bigcup_{u \in U} \) Influence set of \( u \)

Note: the influence set defined as \( \subseteq V^{r+R} \).
**Some Definitions**

**Definition**
The dynamic distance from a node \( u \in V^r \) to a node \( v \) starting at round \( r \) is the number of rounds it takes for flooded messages from \( u \) to reach \( v \) starting at round \( r \).

**Definition**
The influence set of \( u \) after \( R \) rounds starting at round \( r \) is the set of nodes in \( V^{r+R} \) whose dynamic distance from \( u \) starting at round \( r \) is at most \( R \).

\[
\text{Influence set of } U \subseteq V^r = \bigcup_{u \in U} \text{Influence set of } u
\]

*Note: the influence set defined as \( \subseteq V^{r+R} \).*
Some Definitions

Definition
The dynamic distance from a node $u \in V^r$ to a node $v$ starting at round $r$ is the number of rounds it takes for flooded messages from $u$ to reach $v$ starting at round $r$.

Definition
The influence set of $u$ after $R$ rounds starting at round $r$ is the set of nodes in $V^{r+R}$ whose dynamic distance from $u$ starting at round $r$ is at most $R$.

Influence set of $U \subseteq V^r = \bigcup_{u \in U} \text{Influence set of } u$

Note: the influence set defined as $\subseteq V^{r+R}$. 
Properties of Influence Sets

We establish that

A large fraction of the nodes can “influence” a (common) large fraction of nodes in $O(\log n)$ rounds.

1. Any reasonably sized fraction of the nodes (i.e., $\geq \beta n$ nodes) influence a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) in constant rounds.

2. Given any reasonably sized fraction of the nodes $U$ (i.e., $\geq \beta n$ nodes), there is a node $u \in U$ that influences $(1 - \beta)n$ nodes in $O(\log n)$ rounds.

3. There is a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) that influence a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) in $O(\log n)$ rounds.
Properties of Influence Sets

We establish that

A large fraction of the nodes can “influence” a (common) large fraction of
nodes in $O(\log n)$ rounds.

1. Any reasonably sized fraction of the nodes (i.e., $\geq \beta n$ nodes) influence
a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) in constant rounds.

2. Given any reasonably sized fraction of the nodes $U$ (i.e., $\geq \beta n$
nodes), there is a node $u \in U$ that influences $(1 - \beta)n$ nodes in
$O(\log n)$ rounds.

3. There is a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) that
influence a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) in
$O(\log n)$ rounds.
Properties of Influence Sets

We establish that

A large fraction of the nodes can “influence” a (common) large fraction of nodes in $O(\log n)$ rounds.

1. Any reasonably sized fraction of the nodes (i.e., $\geq \beta n$ nodes) influence a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) in constant rounds.

2. Given any reasonably sized fraction of the nodes $U$ (i.e., $\geq \beta n$ nodes), there is a node $u \in U$ that influences $(1 - \beta)n$ nodes in $O(\log n)$ rounds.

3. There is a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) that influence a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) in $O(\log n)$ rounds.
Properties of Influence Sets

We establish that

A large fraction of the nodes can “influence” a (common) large fraction of nodes in $O(\log n)$ rounds.

1. Any reasonably sized fraction of the nodes (i.e., $\geq \beta n$ nodes) influence a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) in constant rounds.

2. Given any reasonably sized fraction of the nodes $U$ (i.e., $\geq \beta n$ nodes), there is a node $u \in U$ that influences $(1 - \beta)n$ nodes in $O(\log n)$ rounds.

3. There is a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) that influence a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) in $O(\log n)$ rounds.
Properties of Influence Sets

We establish that a large fraction of the nodes can “influence” a (common) large fraction of nodes in $O(\log n)$ rounds.

1. Any reasonably sized fraction of the nodes (i.e., $\geq \beta n$ nodes) influence a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) in constant rounds.

2. Given any reasonably sized fraction of the nodes $U$ (i.e., $\geq \beta n$ nodes), there is a node $u \in U$ that influences $(1 - \beta)n$ nodes in $O(\log n)$ rounds.

3. There is a large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) that influence a *common* large fraction of the nodes (i.e., $(1 - \beta)n$ nodes) in $O(\log n)$ rounds.
Algorithmic Tools

**Globally Representative Value:** A large fraction of the nodes (in unison) choose a value held by some node

- Why does this not suffice in the first place?

**Support Estimation:** Count the number of nodes currently "leaning towards" 1.

- Let $X_1, X_2, \ldots, X_k$ be $k$ exponential r.v's with parameter 1. Then,

$$E[\min_{i \in [k]} X_i] = 1/k.$$ 

- Each node estimates

- Guarantee: large fraction of the nodes estimate within a small margin of error
Algorithmic Tools

**Globally Representative Value:** A large fraction of the nodes (in unison) choose a value held by some node

- Why does this not suffice in the first place?

**Support Estimation:** Count the number of nodes currently “leaning towards” 1.

- Let $X_1, X_2, \ldots, X_k$ be $k$ exponential r.v’s with parameter 1. Then,

$$E[\min_{i \in [k]} X_i] = 1/k.$$  

- Each node estimates

- Guarantee: large fraction of the nodes estimate within a small margin of error
Algorithmic Tools

Globally Representative Value: A large fraction of the nodes (in unison) choose a value held by some node

Why does this not suffice in the first place?

Support Estimation: Count the number of nodes currently “leaning towards” 1.

- Let $X_1, X_2, \ldots, X_k$ be $k$ exponential r.v’s with parameter 1. Then,

$$E[\min_{i \in [k]} X_i] = 1/k.$$ 

- Each node estimates

- Guarantee: large fraction of the nodes estimate within a small margin of error
Intuition Behind Solution

0

n
Intuition Behind Solution

Decide on 0
Intuition Behind Solution

Decide on 1
What now?
Use a “Globally Representative Value”
What now?
Storing/Searching Via Random Walks Based Sampling
Consider a single data item (think \(\langle \text{key}, \text{value} \rangle\) pair) generated by some node

\[ C(n) = \frac{n}{\text{polylog}(n)} \]

Direct communication allowed

How to store/search despite churn?
Problem Statement

- Consider a single data item (think ⟨key, value⟩ pair) generated by some node
- \( C(n) = n / \text{polylog}(n) \)
- Direct communication allowed

How to store/search despite churn?
Results

Storage & Maintenance

- *Store* $O(\log n)$ copies
- *Maintain* the data in the network for polynomial in $n$ number of rounds

Search

- *Construct* a search facilitating structure of size $\tilde{O}(\sqrt{n})$
- *Most* nodes (at least $n - o(n)$) can *Search* for a particular key in $O(\log n)$ time

(Results hold with high probability)
A Long Term Task in a Short Lived Life
— the committee

**Issue:** Suppose a node is entrusted with a task.
The node may be churned out before task completed.

**Solution:** Form a completely connected committee of $\Theta(\log n)$ *random* nodes.

- Small enough, low communication cost
- Easy to behave in unison
- Hard for oblivious adversary to disrupt
A Long Term Task in a Short Lived Life
— the committee

**Issue:** Suppose a node is entrusted with a task.

    The node may be churned out before task completed.

**Solution:** Form a completely connected committee of $\Theta(\log n)$ random nodes.

- Small enough, low communication cost
- Easy to behave in unison
- Hard for oblivious adversary to disrupt
Creating a Committee

Context. A node $u$ has some “task” to perform.

1: Node $u$ sends an invitation (along with list of invitees) to $\Theta(\log n)$ random nodes.

2: Nodes that receive the invitation (if alive) connect with the other invitees and form a committee.
Committee Maintenance
— the algorithm

Why maintain? With $n/\polylog(n)$ churn, the committee will be decimated in $O(\polylog(n))$ round.

Every $\Theta(\log n)$ rounds

1: The committee elects a leader $\ell$

2: The node $\ell$ elects a new committee

3: The task is handed over to the new committee

4: The old committee disbands itself after the new committee can fully take over.
A Good Committee
Good if at least $\Omega(\log n)$ nodes.

Theorem
With high probability, a committee will be good for a number of rounds polynomial in $n$. 
Committee Landmarks
— useful in storing and maintaining data

{Everytime a new committee is (re)formed, the following is executed.}

Each node in the new committee spawns a pair of landmarks by selecting two samples and passes all committee ids and initiates level $\leftarrow 0$.

Each new landmark spawns another pair in turn and sends committee ids and incremented level. Repeat until $\tilde{O}(\sqrt{n})$ landmarks are created.

Storing and Maintaining Data

To store an identifiable data item,

- node $u$ creates a committee and the member store the data
- Landmarks are created
- Data is passed on to new committee members chosen for maintenance.
Committee Landmarks
— useful in storing and maintaining data

{Everytime a new committee is (re)formed, the following is executed.}

Each node in the new committee spawns a pair of landmarks by selecting two samples and passes all committee ids and initiates level ← 0.

Each new landmark spawns another pair in turn and sends committee ids and incremented level. Repeat until $\tilde{O}(\sqrt{n})$ landmarks are created.

---

Storing and Maintaining Data

To store an identifiable data item,

1. node $u$ creates a committee and the member store the data

2. Landmarks are created

3. Data is passed on to new committee members chosen for maintenance.
Searching For Identifiable Data
— using the birthday paradox

1. Create a committee and entrust with task of finding data

2. The committee creates landmarks (similar to before)

3. When these landmarks collide with landmarks of searched data, the data can be retrieved.

Theorem
At any round $r$,

- There are $n - o(n)$ nodes that can store a data item,
- The data item can be maintained for a polynomial in $n$ number of rounds, and
- The data item can be searched by $n - o(n)$ nodes in $\tilde{O}(\log n)$ rounds.
Searching For Identifiable Data
— using the birthday paradox

1. Create a committee and entrust with task of finding data
2. The committee creates landmarks (similar to before)
3. When these landmarks collide with landmarks of searched data, the data can be retrieved.

**Theorem**

At any round $r$,

- There are $n - o(n)$ nodes that can store a data item,
- The data item can be maintained for a polynomial in $n$ number of rounds, and
- The data item can be searched by $n - o(n)$ nodes in $\tilde{O}(\log n)$ rounds.
How to find Random Nodes?
— the soup

1: for round \( r = 1, 2, \ldots \) at every node \( v \) do
2:   **Initiate** \( \Theta(\log n) \) random walk tokens
    \( \rightarrow \) with node \( v \)'s id and
    \( \rightarrow \) a timer set for \( \Theta(\log n) \) rounds
3:   **Forward** every unexpired random walk tokens
4:   **Consume** expired random walks (right away) as node samples
5: end for

**Theorem (paraphrased)**

Most nodes will get \( \Theta(\log n) \) node samples every round chosen (almost) uniformly at random from most nodes.
How to find Random Nodes?
— the soup

1: for round $r = 1, 2, \ldots$ at every node $v$ do
2: \hspace{1em} Initiate $\Theta(\log n)$ random walk tokens
   \hspace{1em} $\rightarrow$ with node $v$’s id and
   \hspace{1em} $\rightarrow$ a timer set for $\Theta(\log n)$ rounds
3: \hspace{1em} Forward every unexpired random walk tokens
4: \hspace{1em} Consume expired random walks (right away) as node samples
5: end for

Theorem (paraphrased)

Most nodes will get $\Theta(\log n)$ node samples every round chosen (almost) uniformly at random from most nodes.
How to find Random Nodes?
— the soup

1: **for** round \( r = 1, 2, \ldots \) at every node \( v \) **do**
2: **Initiate** \( \Theta(\log n) \) random walk tokens
   \( \rightarrow \) with node \( v \)'s id and
   \( \rightarrow \) a timer set for \( \Theta(\log n) \) rounds
3: **Forward** every unexpired random walk tokens
4: **Consume** expired random walks (right away) as node samples
5: **end for**

**Theorem (paraphrased)**

Most nodes will get \( \Theta(\log n) \) node samples every round chosen (almost) uniformly at random from most nodes.
How to find Random Nodes?
— the soup

1. **for** round $r = 1, 2, \ldots$ at every node $v$ **do**
2. **Initiate** $\Theta(\log n)$ random walk tokens
   - with node $v$’s id and
   - a timer set for $\Theta(\log n)$ rounds

3. **Forward** every unexpired random walk tokens

4. **Consume** expired random walks (right away) as node samples

5. **end for**

**Theorem (paraphrased)**

Most nodes will get $\Theta(\log n)$ node samples every round chosen \((almost)\) uniformly at random from most nodes.
How to find Random Nodes?

— the soup

1: **for** round $r = 1, 2, \ldots$ at every node $v$ **do**
2: **Initiate** $\Theta(\log n)$ random walk tokens
   $\longrightarrow$ with node $v$’s id and
   $\longrightarrow$ a timer set for $\Theta(\log n)$ rounds
3: **Forward** every unexpired random walk tokens
4: **Consume** expired random walks (right away) as node samples
5: **end for**

**Theorem (paraphrased)**
Most nodes will get $\Theta(\log n)$ node samples every round chosen (almost) uniformly at random from from most nodes.
How to find Random Nodes?

— the soup

1: **for** round \( r = 1, 2, \ldots \) at every node \( v \) **do**
2: **Initiate** \( \Theta(\log n) \) random walk tokens
   \[ \text{−→} \text{ with node } v \text{'s id and} \]
   \[ \text{−→} \text{ a timer set for } \Theta(\log n) \text{ rounds} \]
3: **Forward** every unexpired random walk tokens
4: **Consume** expired random walks (right away) as node samples
5: **end for**

**Theorem (paraphrased)**

*Most nodes will get \( \Theta(\log n) \) node samples every round chosen (almost) uniformly at random from most nodes.*
What next?

Can we eliminate the need for expansion?

- More realistic
- Need models amenable to theoretical analysis
- What techniques?
Thank You