Distributed Computing in Fault-Prone Dynamic Networks

Philipp Brandes, Friedhelm Meyer auf der Heide

Introduction

- Moving nodes in a dynamic network with changing connections

Introduction

- Moving nodes in a dynamic network with changing connections
- Given highly dynamic network with n nodes
- But n unknown
- Needed for many basic tasks
  - all-to-all dissemination
  - determining median
- Counting important task by itself

Overview

- Introduction
- Model
- Impossibility of strong counting
- Weak counting
- Strong counting with upper bound N

Model

- $G_t = (V, E_t)$ with $V = |n|$
- Connected in every round, but no other restriction on E_t
- Nodes communicate via broadcast
- Each node has unique identifier (UID)
- T-interval dynamics: $\exists$stable, connected subgraph for the next T rounds at every round
- Solved by Kuhn et al. in $\mathcal{O}\left(n + \frac{n^2}{T}\right)$

Model

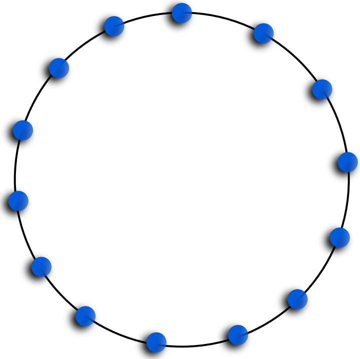
- $G_t = (V, E_t)$ with $V = |n|$
- Connected in every round, but no other restriction on E_t
- Nodes communicate via broadcast
- Each node has unique identifier (UID)
- T-interval dynamics: $\exists$stable, connected subgraph for the next T rounds at every round
- Solved by Kuhn et al. in $\mathcal{O}\left(n + \frac{n^2}{T}\right)$
- Random edge fault with probability p on top

Counting

Strong Counting: An algorithm for strong counting has a runtime bound $t(n)$ such that each node stops with the correct count n within $t(n)$ steps

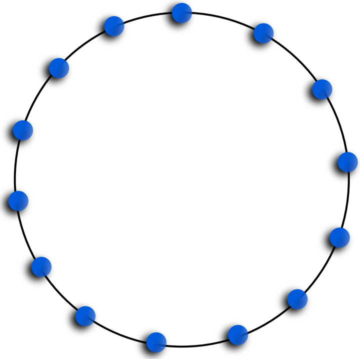
Weak Counting: An algorithm for weak counting has a runtime bound $t(n)$ such that each node has the correct count n after $t(n)$ steps, but the execution of the algorithm does not necessarily stop

Strong Counting with Random Edge Faults



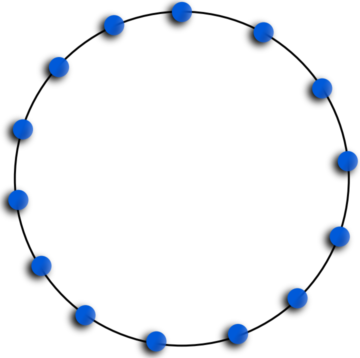
- Assume algorithm A with runtime bound $t(n)$
- Consider edges e_1, e_2 which create segments of length n

Strong Counting with Random Edge Faults



- Assume algorithm A with runtime bound $t(n)$
- Consider edges e_1, e_2 which create segments of length n
- Always faulty during the first $t(n)$ steps if size of the ring $T(n) \geq \left(\frac{1}{p}\right)^{2t(n)}$ with constant probability

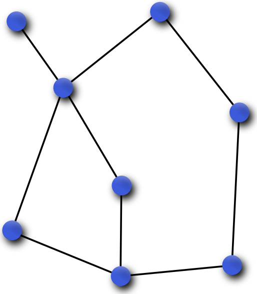
Strong Counting with Random Edge Faults



- Assume algorithm A with runtime bound $t(n)$
- Consider edges e_1, e_2 which create segments of length n
- Always faulty during the first $t(n)$ steps if size of the ring $T(n) \geq \left(\frac{1}{p}\right)^{2t(n)}$ with constant probability
- Strong Counting is not possible under random edge faults

Distributed Counting

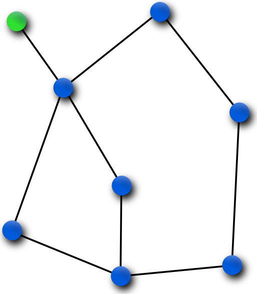
- Guess $k = 2, 4, 8, \ldots$
- Use T-dissemination to spread UIDs
- Count UIDs to obtain n



```
Disseminate(A, k)
S ← ∅
for i = 1, ..., k/T
  for r = 1, ..., 2T
    if S ≠ A
      b ← min (A \ S)
      broadcast b
      receive b_1, ..., b_y
      A ← A ∪ b_1, ..., b_y
      S ← S ∪ b
  S ← ∅
```

Distributed Counting

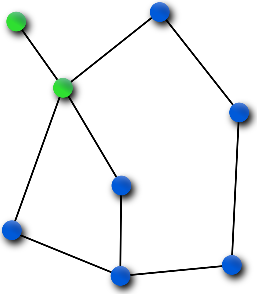
- Guess $k = 2, 4, 8, \ldots$
- Use T-dissemination to spread UIDs
- Count UIDs to obtain n



$\texttt{Disseminate}(A, k)$
$S \leftarrow \emptyset$
$\texttt{for } i = 1, \ldots, \frac{k}{T}$
  $\texttt{for } r = 1, \ldots, 2T$
    $\texttt{if } S \neq A$
      $b \leftarrow \min(A \setminus S)$
      $\texttt{broadcast } b$
      $\texttt{receive } b_1, \ldots, b_y$
      $A \leftarrow A \cup b_1, \ldots, b_y$
      $S \leftarrow S \cup b$
  $S \leftarrow \emptyset$

Distributed Counting

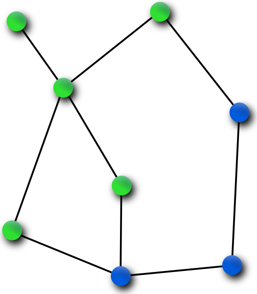
- Guess $k = 2, 4, 8, \ldots$
- Use T-dissemination to spread UIDs
- Count UIDs to obtain n



```
Disseminate(A, k)
S ← ∅
for i = 1, ..., k/T
  for r = 1, ..., 2T
    if S ≠ A
      b ← min (A \ S)
      broadcast b
      receive b_1, ..., b_y
      A ← A ∪ b_1, ..., b_y
      S ← S ∪ b
  S ← ∅
```

Distributed Counting

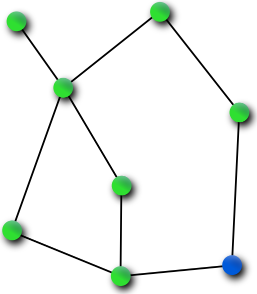
- Guess $k = 2, 4, 8, \ldots$
- Use T-dissemination to spread UIDs
- Count UIDs to obtain n



```
Disseminate(A, k)
S ← ∅
for i = 1, ..., k/T
   for r = 1, ..., 2T
      if S ≠ A
         b ← min(A \ S)
         broadcast b
         receive b_1, ..., b_y
         A ← A ∪ b_1, ..., b_y
         S ← S ∪ b
   S ← ∅
```

Distributed Counting

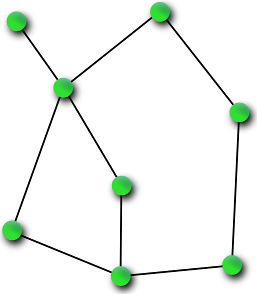
- Guess $k = 2, 4, 8, \ldots$
- Use T-dissemination to spread UIDs
- Count UIDs to obtain n



$\texttt{Disseminate}(A, k)$
$S \leftarrow \emptyset$
$\texttt{for } i = 1, \ldots, \frac{k}{T}$
  $\texttt{for } r = 1, \ldots, 2T$
    $\texttt{if } S \neq A$
      $b \leftarrow \min(A \setminus S)$
      $\texttt{broadcast } b$
      $\texttt{receive } b_1, \ldots, b_y$
      $A \leftarrow A \cup b_1, \ldots, b_y$
      $S \leftarrow S \cup b$
  $S \leftarrow \emptyset$

Distributed Counting

- Guess $k = 2, 4, 8, \ldots$
- Use T-dissemination to spread UIDs
- Count UIDs to obtain n



```
Disseminate(A, k)
S ← ∅
for i = 1, ..., k/T
  for r = 1, ..., 2T
    if S ≠ A
      b ← min (A \ S)
      broadcast b
      receive b_1, ..., b_y
      A ← A ∪ b_1, ..., b_y
      S ← S ∪ b
  S ← ∅
```

Dissemination under T-interval Dynamics and Edge Faults

- Adapt dissemination such that it can handle failures

```
Disseminate(A, l, x)
S ← ∅
for i = 1, ..., l
  for r = 1, ..., 2T/x
    if S ≠ A
      b ← min(A \ S)
      for q = 1, ..., x
        broadcast b
        receive b_1, ..., b_y
        A ← A ∪ b_1, ..., b_y
      S ← S ∪ b
  S ← ∅
```

Weak Counting

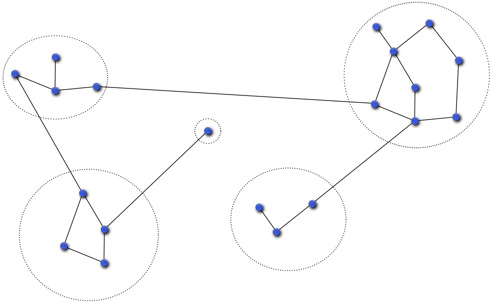
- Use Disseminate(A, l, x) to achieve s-dissemination.
  - If $p > \frac{1}{T}$, set $s = \frac{T}{2\log(T)} \log\left(\frac{1}{p}\right)$ and $l = 2 \cdot \frac{1}{1-p} \cdot e \cdot \frac{k}{s}$.
  - If $p \leq \frac{1}{T}$, set $s = \frac{T}{2}$, and $l = 2 \cdot \frac{1}{1-p} \cdot e \cdot \frac{k}{s}$.
  - Note that $s = \frac{T}{x}$

Theorem

The above procedure executes weak counting. If $p > \frac{1}{T}$, then all nodes output the correct count n after $\mathcal{O}\left(\frac{n^2}{T}\left(\frac{\log(T)}{\log\left(\frac{1}{p}\right)}\right)^2 \cdot \frac{1}{1-p}\right)$ steps. If $p \leq \frac{1}{T}$, they do so after $\mathcal{O}\left(\frac{n^2}{T}\right)$ steps. The bounds hold with probability at least $1 - e^{-\frac{n}{2T}}$.

Distributed Counting (2)

- *k*-Verification
  - Send committee ID or ⊥ if at least two committees are known



Strong Counting

- Use upper bound $N \geq n$ and reuse k-verification

Theorem

If an upper bound N on the number n of nodes is known to all nodes, then strong counting can be done. If $p > \frac{1}{T}$*, then it needs runtime* $\mathcal{O}\left(\frac{n^2}{T} \cdot \left(\frac{\log(T)}{\log\left(\frac{1}{p}\right)}\right)^2 \cdot \frac{1}{1-p} + \log\left(\frac{1}{p}\right) \cdot n \cdot \log N\right)$*. If* $p \leq \frac{1}{T}$*, then runtime* $\mathcal{O}\left(\frac{n^2}{T} + \log\left(\frac{1}{p}\right) \cdot n \cdot N\right)$ *suffices. The bounds hold with probability at least* $1 - n^{-\alpha}$*.*

p Unknown

- If p is unknown, strong counting is not possible

p Unknown

- If p is unknown, strong counting is not possible
- Weak counting with $\log n$ overhead
  - Let $k' = 2, 4, 8, \ldots$ be powers of 2 (upper bound on runtime)
  - Let $k = 2, 4, 8, \ldots$ be powers of 2 (estimation number of nodes)
  - Set p such that runtime bound is met

Conclusions

- Strong counting not possible without upper bound
- Strong counting possible with upper bound on n
- Weak counting possible with small overhead

Conclusions

- Strong counting not possible without upper bound
- Strong counting possible with upper bound on n
- Weak counting possible with small overhead

- Other connectivity models?

Conclusions

- Strong counting not possible without upper bound
- Strong counting possible with upper bound on n
- Weak counting possible with small overhead

- Other connectivity models?

Questions?