Randomized Rumor Spreading Algorithms

Keshav Raj Gupta (210508), Havi Bohra (210429) & Ankur Kumar (210152)

April 11, 2025

Abstract

This report analyzes and simulates three fundamental randomized rumor spreading algorithms: Push, Pull, and Push-Pull. We investigate their theoretical properties, focusing on the dynamics of information dissemination in a network, proving key lemmas and complexity bounds. We then outline an experimental setup to empirically validate these theoretical findings, discussing expected results and suitable visualizations. We prove that all three algorithms disseminate a rumor to all nodes in a network of size n in $O(\log n)$ rounds using $O(n \log n)$ messages, with high probability, and propose experiments to observe this behavior.

1 Model Definition

We define the model for rumor spreading as follows:

- **Network:** We consider a complete graph K_n with n nodes (vertices).
- **Nodes:** Each node v has a state $S(v) \in \{\text{informed, uninformed}\}$.
- **Time:** Time proceeds in synchronous rounds $t = 1, 2, 3, \ldots$.
- **Initialization:** At round $t = 0$, exactly one node s has $S(s) = \text{informed}$. For all $v \neq s$, $S(v) = \text{uninformed}$. Let $I_0 = \{s\}$, $U_0 = V \setminus \{s\}$.
- **Goal:** The process terminates when all n nodes are informed ($I_t = V$).
- **Randomness:** In each round, nodes choose other nodes uniformly and independently at random from $V = \{1, \ldots, n\}$.
- **Notation:**
  - V: Set of all nodes, $|V| = n$.
  - I_t: Set of informed nodes at the *start* of round t.
  - U_t: Set of uninformed nodes at the *start* of round t. $U_t = V \setminus I_t$.
  - $i_t = |I_t|$, $u_t = |U_t|$. $i_t + u_t = n$.

2 Algorithm Descriptions

2.1 Push Algorithm

In each round, informed nodes attempt to push the rumor to randomly chosen nodes.

Algorithm 1: Push Algorithm (Round t)

Input: I_t (set of informed nodes)
Output: I_{t+1} (updated set of informed nodes)

$I_{new} \leftarrow \emptyset$;
forall *node* $u \in I_t$ **do**
    $v \leftarrow$ Choose a node uniformly at random from V;
    if $v \in U_t$ **then**
        $I_{new} \leftarrow I_{new} \cup \{v\}$;
    ▷ `Node` u `sends 1 message`
$I_{t+1} \leftarrow I_t \cup I_{new}$;
return I_{t+1};

2.2 Pull Algorithm

In each round, uninformed nodes attempt to pull the rumor from randomly chosen nodes.

Algorithm 2: Pull Algorithm (Round t)

Input: I_t, U_t (sets of informed and uninformed nodes)
Output: I_{t+1}, U_{t+1} (updated set of informed and uninformed nodes)

$I_{new} \leftarrow \emptyset$;
forall *node* $u \in U_t$ **do**
    $v \leftarrow$ Choose a node uniformly at random from V;
    if $v \in I_t$ **then**
        $I_{new} \leftarrow I_{new} \cup \{u\}$;
    ▷ `Node` u `sends 1 message`
$I_{t+1} \leftarrow I_t \cup I_{new}$;
$U_{t+1} \leftarrow \bar{I}_{t+1}$;
return I_{t+1}, U_{t+1};

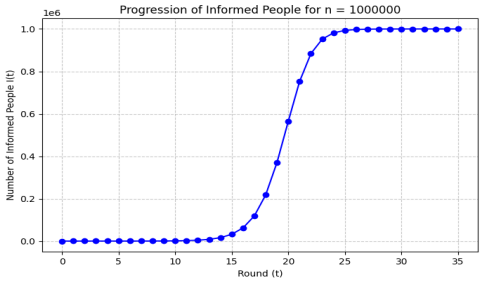
2.3 Push-Pull Algorithm

In each round, every node participates: informed nodes push, uninformed nodes pull.

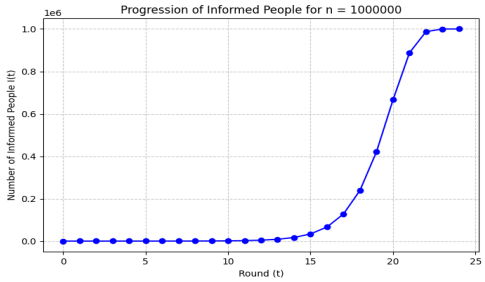
Algorithm 3: Push-Pull Algorithm (Round t)

Input: I_t, U_t (sets of informed and uninformed nodes)
Output: I_{t+1} (updated set of informed nodes)

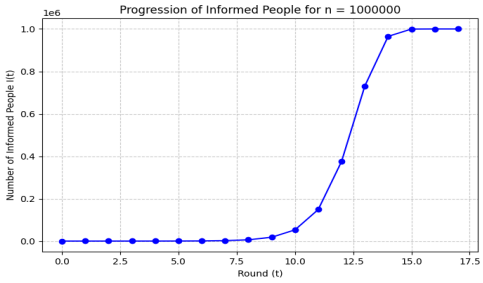
$I_{new} \leftarrow \emptyset$;
$NodesToInform \leftarrow \emptyset$;
▷ `Track who receives rumor this round`
forall *node* $u \in V$ **do**
  $v \leftarrow$ Choose a node uniformly at random from V;
  ▷ `Node` u `sends 1 message`
  if $u \in I_t$ **then**
    ▷ `Push Action`
    $NodesToInform \leftarrow NodesToInform \cup \{v\}$;
  else
    ▷ $u \in U_t$`, Pull Action`
    if $v \in I_t$ **then**
      $NodesToInform \leftarrow NodesToInform \cup \{u\}$;
forall *node* $w \in NodesToInform$ **do**
  if $w \in U_t$ **then**
    $I_{new} \leftarrow I_{new} \cup \{w\}$;
$I_{t+1} \leftarrow I_t \cup I_{new}$;
return I_{t+1};



(a) Push Algorithm



(b) Pull Algorithm



(c) Push-Pull Algorithm

Figure 1: Sample progressions (n=10^6) for (a) Push (b) Pull (c) Push-Pull Algorithm

3 Theoretical Analysis: Key Lemmas

Lemma 3.1 (Push Algorithm - Early Phase Growth)**.** *In the early rounds of the Push algorithm, while the number of informed nodes i_t is significantly smaller than n (e.g., $i_t \leq n/2$), the* expected *number of informed nodes approximately doubles in each round. Specifically, $\mathbb{E}[i_{t+1}|I_t] \approx 2i_t$.*

Proof. Let I_t be the set of informed nodes at the start of round t, with $i_t = |I_t|$. Consider an uninformed node $v \in U_t$. Node v remains uninformed at the end of round t if and only if *no* informed node $u \in I_t$ chooses v as its target for the push operation. For a specific informed node u, the probability that it does *not* choose v is $(n-1)/n$. Since the choices of the i_t informed nodes are independent, the probability that *none* of them choose v is:

$$\mathbb{P}(v \text{ remains uninformed}) = \left(\frac{n-1}{n}\right)^{i_t} = \left(1 - \frac{1}{n}\right)^{i_t}$$

The probability that v becomes informed during round t is therefore:

$$\mathbb{P}(v \text{ becomes informed}) = 1 - \left(1 - \frac{1}{n}\right)^{i_t}$$

Let X_v be the indicator random variable that node $v \in U_t$ becomes informed in round t. The expected number of *newly* informed nodes in round t is:

$$\begin{aligned}
\mathbb{E}[\#\text{ new informed}|I_t] &= \mathbb{E}\left[\sum_{v \in U_t} X_v\right] = \sum_{v \in U_t} \mathbb{E}[X_v] \\
&= \sum_{v \in U_t} \mathbb{P}(v \text{ becomes informed}) \\
&= u_t \cdot \left(1 - \left(1 - \frac{1}{n}\right)^{i_t}\right) \\
&= (n - i_t) \cdot \left(1 - \left(1 - \frac{1}{n}\right)^{i_t}\right)
\end{aligned}$$

The total expected number of informed nodes at the start of round $t+1$ is:

$$\mathbb{E}[i_{t+1}|I_t] = i_t + \mathbb{E}[\#\text{ new informed}|I_t] = i_t + (n - i_t)\left(1 - \left(1 - \frac{1}{n}\right)^{i_t}\right)$$

Now, consider the phase where i_t is small compared to n. We use the approximation $(1-x)^k \approx 1 - kx$ for small x. Here $x = 1/n$.

$$\left(1 - \frac{1}{n}\right)^{i_t} \approx 1 - \frac{i_t}{n}$$

Substituting this into the expression for $\mathbb{E}[i_{t+1}|I_t]$:

$$\begin{aligned}\mathbb{E}[i_{t+1}|I_t] &\approx i_t + (n - i_t)\left(1 - \left(1 - \frac{i_t}{n}\right)\right) \\ &= i_t + (n - i_t)\left(\frac{i_t}{n}\right) \\ &= i_t + \frac{n \cdot i_t}{n} - \frac{i_t^2}{n} \\ &= i_t + i_t - \frac{i_t^2}{n} \\ &= 2i_t - \frac{i_t^2}{n}\end{aligned}$$

If i_t is small (e.g., $i_t \leq n/2$, making $i_t^2/n \leq i_t/2$), the term i_t^2/n is small relative to $2i_t$. Therefore, $\mathbb{E}[i_{t+1}|I_t] \approx 2i_t$. This shows the expected doubling behavior in the early phase. Alternatively, using the approximation $(1 - 1/n)^{i_t} \approx e^{-i_t/n}$, for small i_t/n, $e^{-i_t/n} \approx 1 - i_t/n$, yielding the same result. □

Lemma 3.2 (Pull Algorithm - Late Phase Convergence using Fractions)**.** *In the late rounds of the Pull algorithm, let $x_t = u_t/n$ be the fraction of uninformed nodes at the start of round t. When x_t is small, the expected fraction of uninformed nodes in the next round decreases quadratically. Specifically, $\mathbb{E}[x_{t+1}|x_t] = x_t^2$.*

Proof. Let U_t be the set of uninformed nodes at the start of round t, with $u_t = |U_t|$, and let $x_t = u_t/n$ be the fraction of uninformed nodes.

Consider a single node $u \in U_t$. In round t of the Pull algorithm, node u contacts a node v chosen uniformly at random from all n nodes. Node u remains uninformed if and only if the contacted node v is also uninformed ($v \in U_t$).

The probability that the randomly chosen node v is uninformed is equal to the fraction of uninformed nodes in the network:

$$\mathbb{P}(v \in U_t) = \frac{|U_t|}{n} = \frac{u_t}{n} = x_t$$

Therefore, the probability that a specific node u (which started in U_t) remains uninformed after round t is x_t.

$$\mathbb{P}(u \text{ remains uninformed} \mid u \in U_t) = x_t$$

Now, let u_{t+1} be the random variable representing the number of uninformed nodes at the start of round $t+1$. These are precisely the nodes that were in U_t and remained uninformed during round t. Let Y_u be the indicator variable for $u \in U_t$, where $Y_u = 1$ if u remains uninformed, and $Y_u = 0$ otherwise. We have $P(Y_u = 1) = x_t$.

The total number of uninformed nodes at time $t + 1$ is $u_{t+1} = \sum_{u \in U_t} Y_u$. By linearity of

expectation, conditioning on the state at time t (which is determined by u_t or equivalently x_t):

$$\begin{aligned}
\mathbb{E}[u_{t+1} \mid U_t] &= \mathbb{E}\left[\sum_{u \in U_t} Y_u\right] \\
&= \sum_{u \in U_t} \mathbb{E}[Y_u] \\
&= \sum_{u \in U_t} \mathbb{P}(Y_u = 1) \\
&= \sum_{u \in U_t} x_t \\
&= |U_t| \cdot x_t \\
&= u_t \cdot x_t
\end{aligned}$$

We are interested in the expected *fraction* of uninformed nodes at time $t+1$, which is $x_{t+1} = u_{t+1}/n$.

$$\begin{aligned}
\mathbb{E}[x_{t+1} \mid x_t] &= \mathbb{E}\left[\frac{u_{t+1}}{n} \,\middle|\, u_t\right] \\
&= \frac{1}{n}\mathbb{E}[u_{t+1} \mid u_t] \quad \text{(by linearity of expectation)} \\
&= \frac{1}{n}(u_t \cdot x_t) \quad \text{(substituting the result above)}
\end{aligned}$$

Now substitute $u_t = n \cdot x_t$:

$$\begin{aligned}
\mathbb{E}[x_{t+1} \mid x_t] &= \frac{1}{n}(n \cdot x_t \cdot x_t) \\
&= x_t^2
\end{aligned}$$

This result, $\mathbb{E}[x_{t+1} \mid x_t] = x_t^2$, shows that the expected fraction of uninformed nodes decreases quadratically. Since $0 \leq x_t \leq 1$, we have $x_t^2 \leq x_t$. The decrease is strict ($x_t^2 < x_t$) as long as $0 < x_t < 1$, which is true whenever the algorithm is running and has not yet terminated ($u_t > 0$) or started ($u_t < n$). When x_t becomes small (late phase), squaring it results in a much smaller value, indicating very rapid convergence towards $x_t = 0$. □

4 Time and Message Complexity Analysis

We now prove that all three algorithms (Push, Pull, Push-Pull) inform all n nodes in $O(\log n)$ rounds using $O(n \log n)$ messages, with high probability (w.h.p.). "With high probability" typically means with probability at least $1 - n^{-c}$ for some constant $c \geq 1$. Rigorous proofs using Chernoff Bounds is detailed in Appendix A.

Theorem 4.1 (Push Time Complexity)**.** *The Push algorithm terminates in $O(\log n)$ rounds w.h.p.*

Proof Sketch. We divide the process into two phases:

1. **Phase 1 (Exponential Growth):** $i_t \leq n/2$. From Lemma 3.1, $\mathbb{E}[i_{t+1}|I_t] \approx 2i_t - i_t^2/n$. As long as $i_t \leq n/2$, $\mathbb{E}[i_{t+1}|I_t] \geq 2i_t - i_t/2 = 1.5i_t$. The expected number grows by a constant factor greater than 1. Using concentration bounds (e.g., Chernoff) on the number of newly informed nodes (which can be shown to be concentrated around the expectation), i_t grows exponentially (at least by a factor $1+\delta$ for some constant $\delta > 0$) in each round w.h.p.Starting from $i_0 = 1$, reaching $i_t = \Omega(n)$ (e.g., $n/2$) takes $O(\log n)$ rounds w.h.p.

2. **Phase 2 (Convergence):** $i_t > n/2$ (meaning $u_t < n/2$). Consider an uninformed node $v \in U_t$. The probability it remains uninformed is $\mathbb{P}(\text{stay}) = (1 - 1/n)^{i_t}$. Since $i_t > n/2$, $\mathbb{P}(\text{stay}) < (1 - 1/n)^{n/2} \approx e^{-1/2} < 0.61$. Let $p \approx e^{-1/2}$. The expected number of uninformed nodes in the next round is $\mathbb{E}[u_{t+1}|U_t] = u_t \cdot \mathbb{P}(\text{stay}) < p \cdot u_t$. The number of uninformed nodes decreases by a constant factor less than 1 in expectation. Using concentration bounds again, u_t decreases by at least a constant factor (< 1) in each round w.h.p.Starting from $u_t < n/2$, reaching $u_t = 0$ takes $O(\log n)$ rounds w.h.p.

Combining Phase 1 and Phase 2, the total time for Push is $O(\log n) + O(\log n) = O(\log n)$ rounds w.h.p. □

Theorem 4.2 (Pull Time Complexity)**.** *The Pull algorithm terminates in $O(\log n)$ rounds w.h.p.*

Proof Sketch. Again, we divide into phases.

1. **Phase 1 (Initial Growth):** $i_t \leq n/2$. Consider the expected increase in informed nodes. An uninformed node v becomes informed if it pulls from an informed node. $\mathbb{P}(v \text{ gets informed}) = i_t/n$. The expected number of newly informed nodes is $\mathbb{E}[\#\text{ new}|I_t] = u_t \cdot (i_t/n) = (n - i_t)i_t/n$. The expected total number of informed nodes is $\mathbb{E}[i_{t+1}|I_t] = i_t + (n - i_t)i_t/n = 2i_t - i_t^2/n$. This is the *same expression* as for the Push algorithm. Therefore, by the same concentration arguments as in Theorem 4.1 Phase 1, the number of informed nodes reaches $\Omega(n)$ (e.g., $n/2$) in $O(\log n)$ rounds w.h.p.

2. **Phase 2 (Fast Convergence):** $i_t > n/2$ (meaning $u_t < n/2$). Consider an uninformed node $v \in U_t$. The probability it becomes informed is $\mathbb{P}(\text{get}) = i_t/n$. Since $i_t > n/2$, $\mathbb{P}(\text{get}) > 1/2$. The expected number of nodes *remaining* uninformed is $\mathbb{E}[u_{t+1}|U_t] = u_t \cdot (1 - \mathbb{P}(\text{get})) < u_t \cdot (1 - 1/2) = u_t/2$. The number of uninformed nodes decreases by a factor of at least 2 in expectation. Using concentration bounds, u_t decreases by a constant factor (< 1) w.h.p.in each round. Starting from $u_t < n/2$, reaching $u_t = 0$ takes $O(\log n)$ rounds w.h.p.(Lemma 3.2 shows even faster convergence once u_t is very small).

Combining Phase 1 and Phase 2, the total time for Pull is $O(\log n) + O(\log n) = O(\log n)$ rounds w.h.p. □

Theorem 4.3 (Push-Pull Time Complexity)**.** *The Push-Pull algorithm terminates in $O(\log n)$ rounds w.h.p.*

Proof Sketch. The Push-Pull algorithm combines the mechanisms of both Push and Pull. An uninformed node v becomes informed if *either* an informed node pushes to it *or* it successfully pulls from an informed node. Consider an uninformed node $v \in U_t$. Let E_{push} be the event that at least one informed node pushes to v. Let E_{pull} be the event that v pulls from an informed node. $\mathbb{P}(v \text{ becomes informed}) = \mathbb{P}(E_{\text{push}} \cup E_{\text{pull}})$. Since the events are not disjoint, we use the fact that $\mathbb{P}(A \cup B) = 1 - \mathbb{P}(\neg A \cap \neg B)$. More simply, $\mathbb{P}(A \cup B) \geq \max(\mathbb{P}(A), \mathbb{P}(B))$.

$$\mathbb{P}(v \text{ becomes informed}) \geq \max(\mathbb{P}(E_{\text{push}}), \mathbb{P}(E_{\text{pull}}))$$

We know $\mathbb{P}(E_{\text{push}}) = 1 - (1 - 1/n)^{i_t}$ (as in Push analysis) and $\mathbb{P}(E_{\text{pull}}) = i_t/n$ (as in Pull analysis).

1. **Phase 1 ($i_t \leq n/2$):** $\mathbb{P}(v \text{ becomes informed}) \geq \mathbb{P}(E_{\text{pull}}) = i_t/n$. The expected number of newly informed nodes $\mathbb{E}[\#\text{ new}|I_t] = u_t \cdot \mathbb{P}(v \text{ becomes informed}) \geq u_t \cdot (i_t/n) = (n - i_t)i_t/n$. $\mathbb{E}[i_{t+1}|I_t] = i_t + \mathbb{E}[\#\text{ new}|I_t] \geq i_t + (n - i_t)i_t/n = 2i_t - i_t^2/n$. Again, this leads to exponential growth in $O(\log n)$ rounds to reach $i_t = \Omega(n)$ w.h.p.

2. **Phase 2 ($i_t > n/2$, $u_t < n/2$):** $\mathbb{P}(v \text{ becomes informed}) \geq \mathbb{P}(E_{\text{push}}) = 1 - (1 - 1/n)^{i_t} > 1 - (1 - 1/n)^{n/2} \approx 1 - e^{-1/2} > 0.39$. Also, $\mathbb{P}(v \text{ becomes informed}) \geq \mathbb{P}(E_{\text{pull}}) = i_t/n > 1/2$. So, $\mathbb{P}(v \text{ becomes informed}) > 1/2$. The expected number of nodes *remaining* uninformed is $\mathbb{E}[u_{t+1}|U_t] = u_t \cdot (1 - \mathbb{P}(v \text{ becomes informed})) < u_t \cdot (1 - 1/2) = u_t/2$. The number of uninformed nodes decreases by a factor of at least 2 in expectation. Using concentration bounds, u_t decreases exponentially w.h.p., taking $O(\log n)$ rounds to reach 0.

Combining phases, the total time for Push-Pull is $O(\log n)$ rounds w.h.p. □

Theorem 4.4 (Message Complexity)**.** *The Push, Pull, and Push-Pull algorithms use $O(n \log n)$ messages in total w.h.p.*

Proof. Let T be the number of rounds until termination. From Theorems 4.1, 4.2, and 4.3, we know $T = O(\log n)$ w.h.p.

- **Push:** In round t, i_t messages are sent. Since $i_t \leq n$, the total number of messages is $\sum_{t=1}^{T} i_t \leq \sum_{t=1}^{O(\log n)} n = n \cdot O(\log n) = O(n \log n)$.
- **Pull:** In round t, u_t messages are sent. Since $u_t \leq n$, the total number of messages is $\sum_{t=1}^{T} u_t \leq \sum_{t=1}^{O(\log n)} n = n \cdot O(\log n) = O(n \log n)$.
- **Push-Pull:** In round t, *every* node n sends one message (either push or pull). The total number of messages is $\sum_{t=1}^{T} n = n \cdot T = n \cdot O(\log n) = O(n \log n)$.

Therefore, all three algorithms use $O(n \log n)$ messages w.h.p. □

5 Experimental Setup

To empirically validate the theoretical results, we propose the following experimental setup:

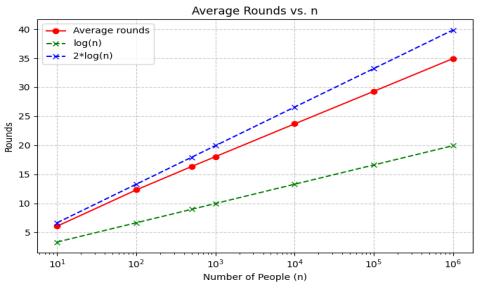
- **Implementation:** Implement the Push, Pull, and Push-Pull algorithms (Section 2) in a chosen programming language (Python).
- **Network Size (n):** Test a range of network sizes, n, e.g., $n = 10, 10^2, 5*10^2, 10^3, 10^4, 10^5, 10^6$.
- **Metrics:** For each trial: Record rounds (T), total messages, and informed count per round (i_t).
- **Averaging:** For each algorithm type and each chosen network size n, we perform N = 100000 independent runs (trials). Each trial should start with a single informed node.
  - Calculate the average termination time (average T) over the N trials.
  - Calculate the average total messages sent over the N trials.
- **Randomness:** Use python's random library to select a random node.

6 Experimental Results and Visualizations

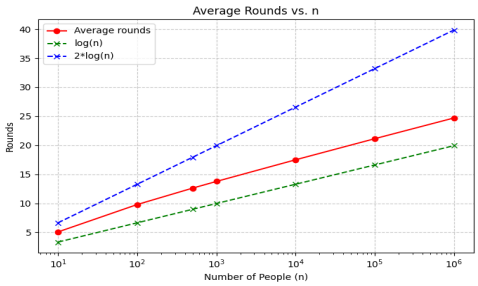
This section will present the results obtained from the simulations described in Section 2. We expect to observe behavior consistent with the theoretical analysis (Sections 3 and 4). Exhaustive list of plots is displayed in Appendix B.

6.1 Convergence Time (Rounds)

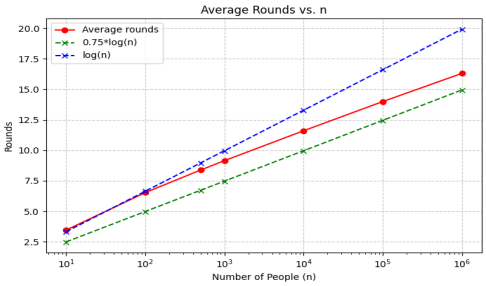
The primary goal here is to verify that the algorithms, namely Push, Pull, and Push-Pull, complete rumour spreading in $O(\log n)$ rounds with high probability (w.h.p.). To examine this empirically, we plot the average number of rounds required for termination ($\bar{T}$) against the network size (n) in Figure 2.



(a) Push Algorithm



(b) Pull Algorithm



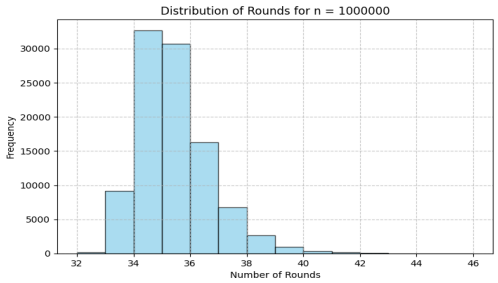
(c) Push-Pull Algorithm

Figure 2: Observed Average rounds vs n in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm

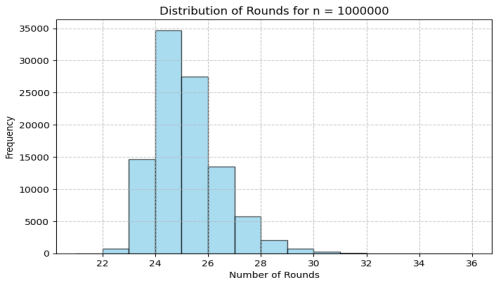
Figure 3 provides a closer look at the distribution of termination times for a fixed large network size, $n = 10^6$, based on the 10^5 simulation trials for each algorithm: (a) Push, (b) Pull, and (c) Push-Pull. These histograms show the frequency of runs terminating within specific round intervals. The key observation across all three plots is the strong concentration of results.

Figure 4 uses box plots to visualize how the distribution of termination rounds changes as the network size n increases. Each box plot summarizes the distribution for a given n over the 10^5 trials, showing the median (line inside the box), the interquartile range (IQR, the box itself), whiskers (typically extending to 1.5 times the IQR), and outliers (individual points beyond the whiskers).

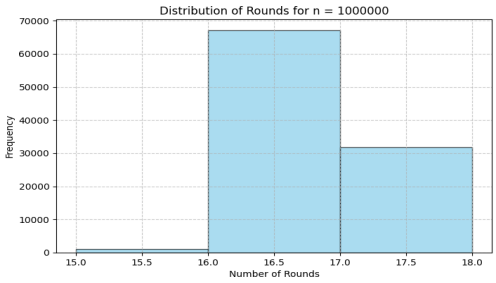
The distributions remain well-concentrated relative to the increasing median. Comparing the plots reaffirms the consistent speed advantage of Push-Pull, followed by Pull, then Push, across the tested network sizes.



(a) Push Algorithm

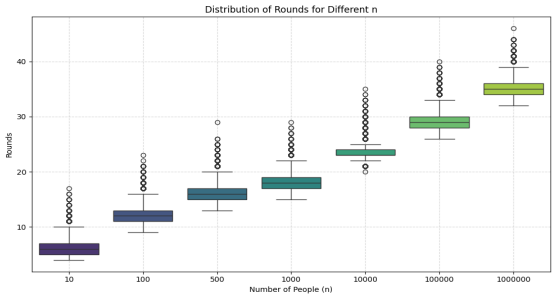


(b) Pull Algorithm

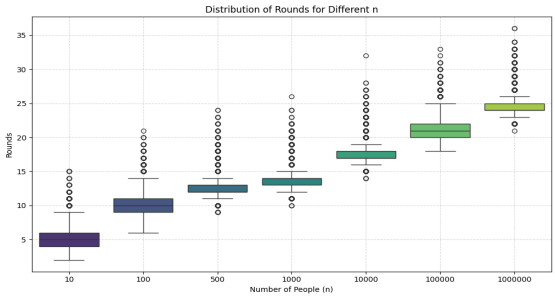


(c) Push-Pull Algorithm

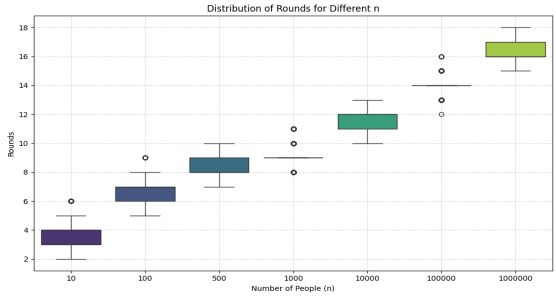
Figure 3: Distribution of Rounds (for n= 10^6) in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm



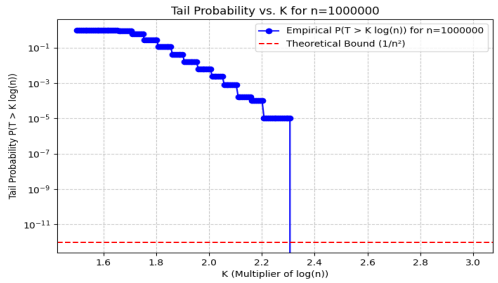
(b) Pull Algorithm



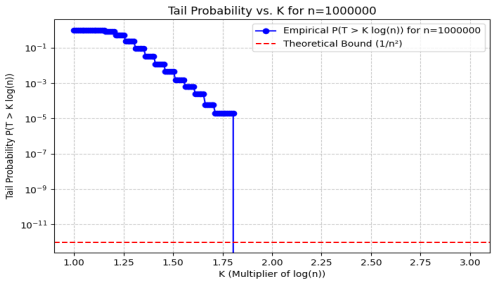
(c) Push-Pull Algorithm

Figure 4: Box-plots of observed rounds for different n in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm

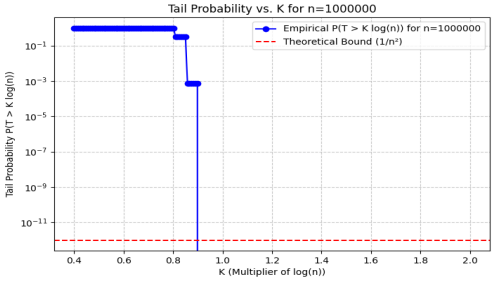
Figure 5 plots the empirically observed probability $\mathbb{P}(T > K \log_2 n)$ against the multiplier K for (a) Push, (b) Pull, and (c) Push-Pull algorithms, based on 10^5 trials. The plots show that the probability of exceeding $K \log_2 n$ rounds drops extremely quickly as K increases beyond a certain threshold. The empirical probabilities fall well below the theoretical $1/n^2$ bound (red dashed line) for moderate values of K, further supporting the claim that termination occurs within $O(\log n)$ rounds with high probability.



(a) Push Algorithm



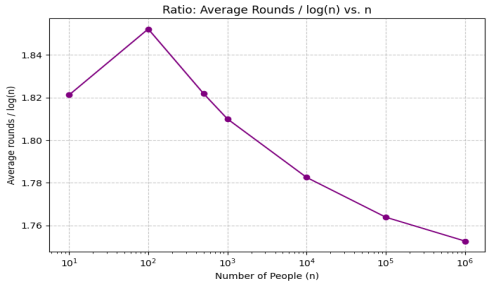
(b) Pull Algorithm



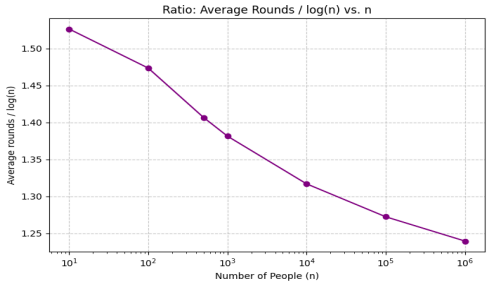
(c) Push-Pull Algorithm

Figure 5: Tail Probability P(#rounds> Klog(n)) vs n (for n= 10^6) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm

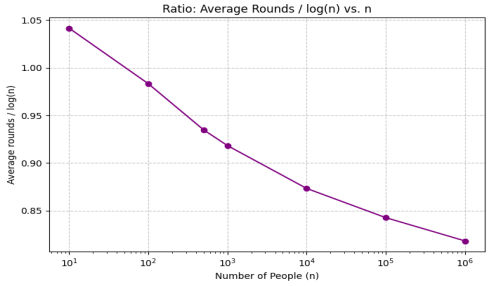
Figure 6 investigates the constant factor hidden in the $\Theta(\log n)$ time complexity. It plots the ratio of the average termination time to $\log_2 n$ (i.e., $\bar{T}/\log_2 n$) against the network size n. As n increases, the ratio appears to converge towards a constant value for each algorithm. This convergence provides strong empirical support for the $T = \Theta(\log n)$ complexity. Furthermore, comparing the limiting values allows for an empirical comparison of the algorithms' efficiency; Push-Pull converges to a ratio near 0.8, Pull near 1.25, and Push near 1.75, indicating that Push-Pull is asymptotically the fastest in terms of rounds, followed by Pull, then Push, consistent with observations in previous figures.



(a) Push Algorithm



(b) Pull Algorithm



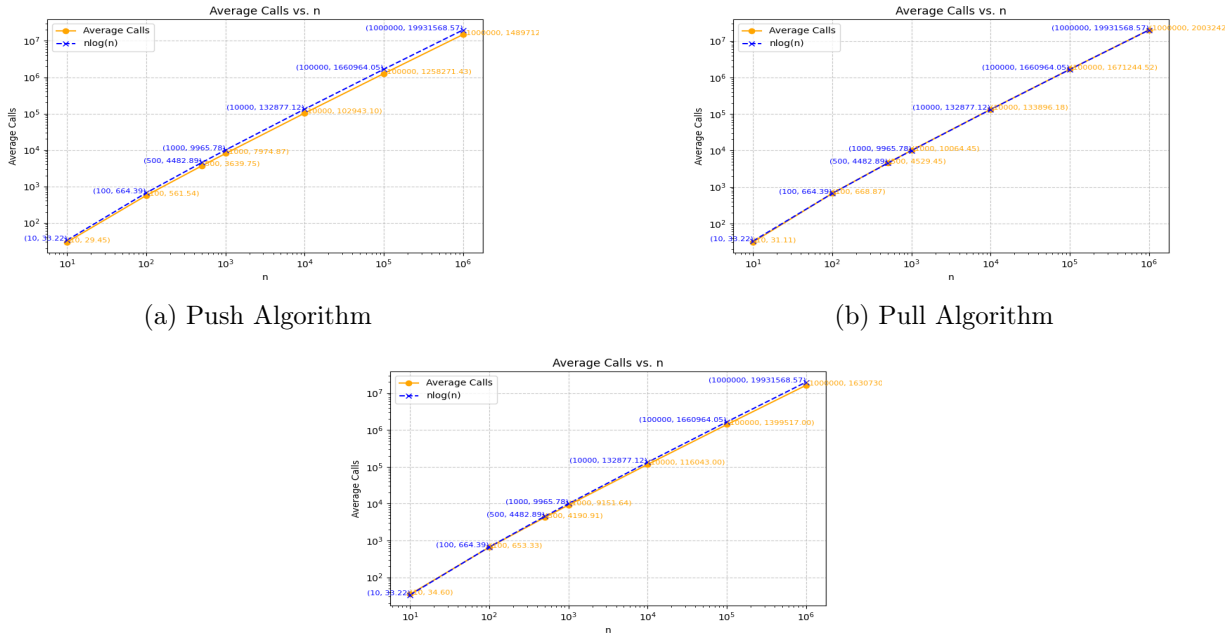
(c) Push-Pull Algorithm

Figure 6: Variation of Ratio = Average Rounds/ log(n) vs. n in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm

6.2 Message Complexity

Here, we verify the $\Theta(n \log n)$ message complexity theoretically established for all three algorithms (Theorem 4.4). We examine the growth of the average total number of messages/calls made during the simulation runs as the network size n increases.

Figure 7 plots the average total number of calls (messages) required for termination against the network size n for (a) Push, (b) Pull, and (c) Push-Pull algorithms, using data from 10^5 trials. The empirical average number of calls (orange line) closely tracks the $n \log_2 n$ reference across several orders of magnitude for all three algorithms. The near-parallel relationship on the log-log scale strongly suggests that the average message complexity scales as $\Theta(n \log n)$, providing empirical validation for the theoretical bound established in Theorem 4.4.



(a) Push Algorithm

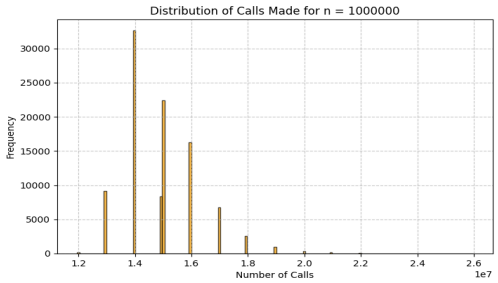
(b) Pull Algorithm

(c) Push-Pull Algorithm

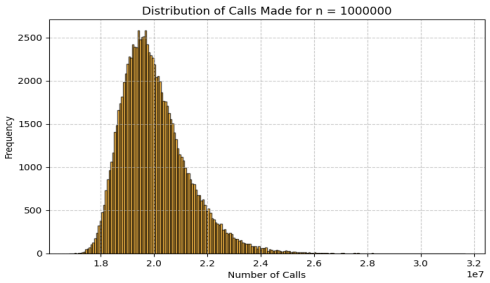
Figure 7: Avg. #total messages/calls made in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm

Similar to Figure 3 for rounds, Figure 8 displays histograms of the total number of messages/calls made for a fixed large network size demonstrating strong concentration around the mean total message count for each algorithm. The narrow peaks relative to the mean values indicate that runs with message counts deviating significantly from the average are rare, showing that the total communication cost is predictable and tightly centered.

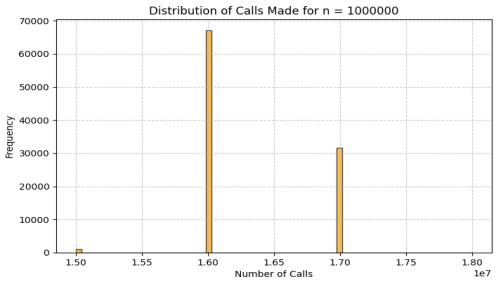
Figure 9 presents box plots illustrating the distribution of the total messages/calls made as the network size n varies. The overall trend is consistent with the $\Theta(n \log n)$ growth observed for the averages in Figure 7. The concentration, indicated by the relatively compact boxes and few outliers, persists across different network sizes.



(a) Push Algorithm

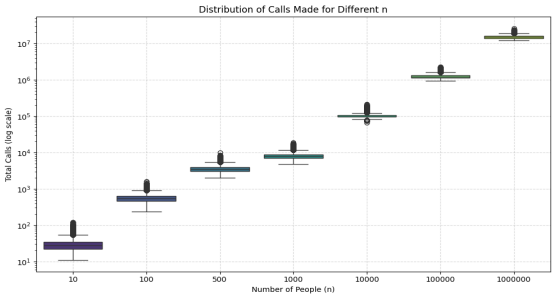


(b) Pull Algorithm

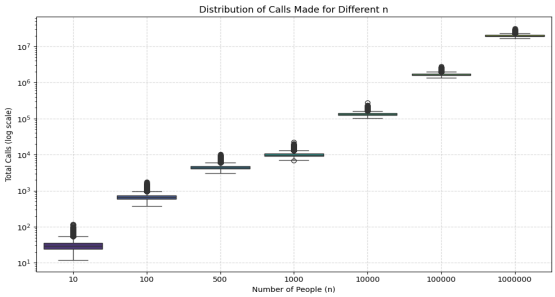


(c) Push-Pull Algorithm

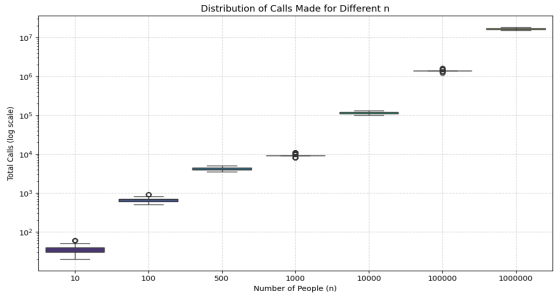
Figure 8: Distribution of #total messages/calls made (for n= 10^6) in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm



(b) Pull Algorithm



(c) Push-Pull Algorithm

Figure 9: Box-plots of observed #total messages/calls for different n in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm

6.3 Rumor Propagation Dynamics

These plots illustrate how the rumor spreading problem evolves over time within a typical run, focusing on the efficiency of the communication process for a large network size ($n = 10^6$). Figure 10 delves into the dynamics of communication efficiency per round for a fixed large network size $n = 10^6$. The figure presents results for the Push (a, b), Pull (c, d), and Push-Pull (e, f) algorithms.

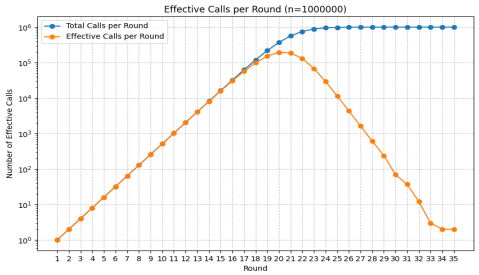
Subfigures 10a, 10c, and 10e plot the number of 'Total Calls per Round' and 'Effective Calls per Round' (calls resulting in a new node being informed) against the round number.

- For Push 10a, the total calls increase as more nodes become informed, peaking mid-process, while effective calls peak earlier and then decline as fewer uninformed targets remain.
- For Pull 10c, the total calls decrease as fewer nodes remain uninformed, while effective calls peak later when many potential targets are informed.
- For Push-Pull 10e, the total calls are constant ($n = 10^6$), while effective calls follow a curve peaking mid-process.

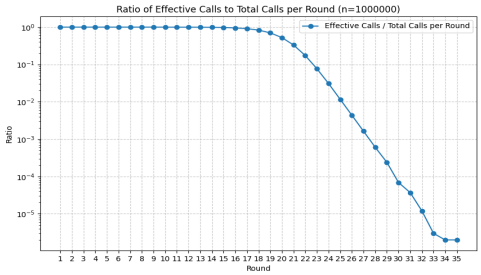
Subfigures 10b, 10d, and 10f show the ratio of 'Effective Calls / Total Calls per Round' versus the round number. This ratio represents the probability that a randomly chosen call in that round is effective.

- For Push 10b, efficiency is high initially (many uninformed targets) and decreases sharply as the network saturates.
- For Pull 10d, efficiency is extremely low initially (few informed targets) and increases dramatically towards the end.
- For Push-Pull 10f, the efficiency curve combines aspects of both, being moderately high throughout the middle phases.

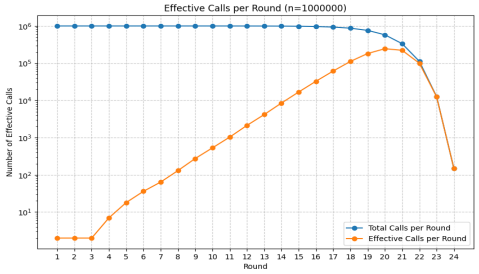
These plots highlight the different efficiency profiles of the algorithms during their execution phases.



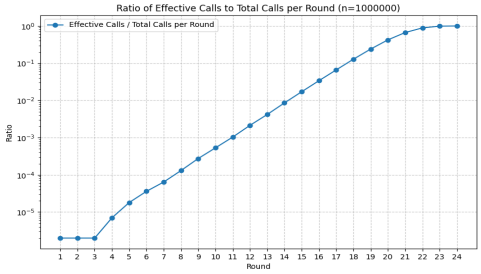
(a) Push Algorithm



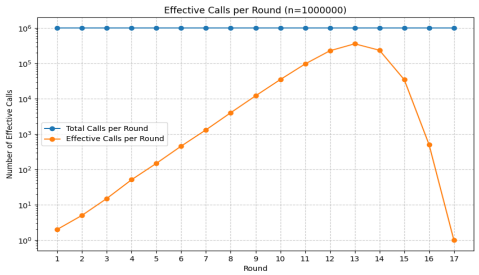
(b) Push Algorithm



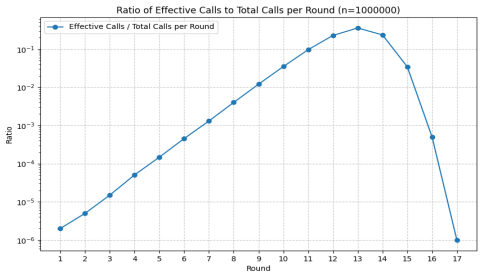
(c) Pull Algorithm



(d) Pull Algorithm



(e) Push-Pull Algorithm



(f) Push-Pull Algorithm

Figure 10: Sample progressions of Effective Calls (plots in left), Ratio of Effective calls/ Total Calls (plots in right) per round, (n=10^6) for each of (a) Push (b) Pull (c) Push-Pull Algorithm

7 Conclusion

We have theoretically analyzed three fundamental randomized rumor spreading algorithms: Push, Pull, and Push-Pull.

- The Push algorithm exhibits exponential growth in the number of informed nodes during its early phase.
- The Pull algorithm exhibits rapid (quadratic) convergence in its late phase when few nodes remain uninformed.
- Both Push and Pull, despite their differing strengths in different phases, achieve full dissemination in $O(\log n)$ rounds w.h.p.
- The combined Push-Pull algorithm also achieves full dissemination in $O(\log n)$ rounds w.h.p., potentially with better constants by leveraging the strengths of both methods throughout the process.
- All three algorithms achieve this logarithmic time complexity using a total of $O(n \log n)$ messages w.h.p.

These theoretical results provide a strong foundation for understanding rumor spreading dynamics. Empirical simulations can further explore the constants hidden in the O notation and the behavior under variations like network topology changes or node failures.

A Detailed Proofs using Chernoff Bounds

This appendix provides more detailed derivations for the time complexity bounds presented in Section 4, explicitly using Chernoff bounds.

Chernoff Bounds Used: Let $X = \sum_{i=1}^{N} X_i$ be a sum of N independent (or negatively associated) Bernoulli variables with $\mu = E[X]$.

- Lower Tail (Thm (b) form): For $0 < \delta < 1$, $\mathbb{P}(X \leq (1-\delta)\mu) \leq e^{-\mu\delta^2/2}$.
- Upper Tail (Thm (a), alt. form): For $0 < \delta$ s.t. $1+\delta < 2e \approx 5.43$, $\mathbb{P}(X \geq (1+\delta)\mu) \leq e^{-\mu\delta^2/4}$.

We require the relevant expectation μ to be $\Omega(\log n)$ for probabilities $\leq n^{-c}$ (we typically aim for $c = 2$).

A.1 Proof of Theorem 4.1 (Push Time Complexity)

Detailed Proof. Analyze in two phases for the $O(\log n)$ upper bound. Analyze growth rate for the $\Omega(\log n)$ lower bound. W.h.p. analyses assume constants C, C' are sufficiently large.

Part 1: Upper Bound $T = O(\log n)$ w.h.p.

Phase 1: Growth ($1 \leq i_t < n/2$)

- Goal: Show i_t reaches $n/2$ in $O(\log n)$ rounds w.h.p.
- Random Variable: $Y_t = \sum_{v \in U_t} X_v$, number newly informed. $i_{t+1} = i_t + Y_t$. Indicators X_v are negatively associated.
- Expectation: $\mu_t = E[Y_t] \geq 0.5 i_t$.
- Chernoff (Lower Tail): Use $\delta = 1/4$. $\mathbb{P}(Y_t \leq 0.75\mu_t) \leq e^{-\mu_t(1/4)^2/2} = e^{-\mu_t/32}$. To ensure this is $\leq n^{-2}$, we need $\mu_t \geq 64 \ln n$. Holds if $i_t \geq C \log n$ ($C = 128$). W.h.p. ($\geq 1 - n^{-2}$), $Y_t \geq 0.75\mu_t \geq (3/8) i_t \implies i_{t+1} \geq (11/8) i_t$.
- Duration: Initial rounds ($i_t < C \log n$) take $O(\log \log n)$ rounds w.h.p.The subsequent growth takes $O(\log n)$ rounds. Phase 1 is $O(\log n)$ w.h.p.

Phase 2: Convergence ($i_t \geq n/2$, so $u_t \leq n/2$)

- Goal: Show u_t goes to 0 in $O(\log n)$ rounds w.h.p.
- Random Variable: $Z_t = u_{t+1} = \sum_{v \in U_t} Y_v$, $Y_v = 1$ if v remains uninformed (negatively associated).
- Expectation: $\mu'_t = E[Z_t] \leq p u_t$, where $p = e^{-1/2} < 0.61$.
- Chernoff (Upper Tail): Use $\delta = 1/4$. $\mathbb{P}(Z_t \geq 1.25\mu'_t) \leq e^{-\mu'_t(1/4)^2/4} = e^{-\mu'_t/64}$. Requires $\mu'_t \geq 128 \ln n$, holds if $u_t \geq C' \log n$. W.h.p. ($\geq 1 - n^{-2}$), $u_{t+1} = Z_t \leq 1.25\mu'_t \leq 1.25 p u_t$. Let $p'' = 1.25p < 0.77$. Then $u_{t+1} \leq p'' u_t$.
- Duration: Decreasing by factor $p'' < 1$ takes $O(\log n)$ rounds. Phase 2 is $O(\log n)$ w.h.p.

Combining phases, $T = O(\log n)$ w.h.p.

Part 2: Lower Bound $T = \Omega(\log n)$ w.h.p.

- Goal: Show i_t cannot grow much faster than doubling.

- Random Variable: i_{t+1}. Expectation $\mu_{t+1} = \mathbb{E}[i_{t+1}] \leq 2i_t$.
- Chernoff (Upper Tail): Use $\delta = 1/2$. $\mathbb{P}(i_{t+1} \geq 1.5\mu_{t+1}) \leq e^{-\mu_{t+1}(1/2)^2/4} = e^{-\mu_{t+1}/16}$. Requires $\mu_{t+1} \geq 32 \ln n$, holds if $i_t \geq C'' \log n$. W.h.p. ($\geq 1 - n^{-2}$), $i_{t+1} \leq 1.5\mu_{t+1} \leq 1.5(2i_t) = 3i_t$.
- Conclusion: Growth factor is ≤ 3 w.h.p. Reaching n requires $T \geq \log_3 n = \Omega(\log n)$ rounds.

Combining bounds, $T = (\log n)$ w.h.p. □

A.2 Proof of Theorem 4.2 (Pull Time Complexity)

Detailed Proof. **Part 1: Upper Bound $T = O(\log n)$ w.h.p.**
Phase 1: Growth ($1 \leq i_t < n/2$)

- Goal: Show i_t reaches $n/2$ in $O(\log n)$ rounds w.h.p.
- Random Variable: $Y_t = \sum_{u \in U_t} X_u$, number newly informed (independent indicators X_u).
- Expectation: $\mu_t = E[Y_t] = u_t(i_t/n) \geq 0.5i_t$.
- Chernoff (Lower Tail): Use $\delta = 1/4$. $\mathbb{P}(Y_t \leq 0.75\mu_t) \leq e^{-\mu_t/32}$. Requires $\mu_t \geq 64 \ln n$, holds if $i_t \geq C \log n$. W.h.p., $i_{t+1} \geq (11/8)i_t$.
- Duration: Takes $O(\log n)$ rounds w.h.p.

Phase 2: Convergence ($i_t \geq n/2$, so $u_t \leq n/2$)

- Goal: Show u_t reaches 0 in $O(\log n)$ rounds w.h.p.
- Random Variable: $Z_t = u_{t+1} = \sum_{u \in U_t} Y_u$, $Y_u = 1$ if u's pull fails (independent indicators Y_u).
- Expectation: $\mu'_t = E[Z_t] = u_t^2/n \leq u_t/2$.
- Chernoff (Upper Tail): Use $\delta = 1/2$. $\mathbb{P}(Z_t \geq 1.5\mu'_t) \leq e^{-\mu'_t(1/2)^2/4} = e^{-\mu'_t/16}$. Requires $\mu'_t = u_t^2/n \geq 32 \ln n$, holds if $u_t \geq C'\sqrt{n \log n}$. W.h.p., $u_{t+1} = Z_t \leq 1.5\mu'_t \leq 0.75u_t$.
- Duration: Reaching $O(\sqrt{n \log n})$ takes $O(\log n)$. Final quadratic convergence is fast. Phase 2 is $O(\log n)$ w.h.p.

Combining phases, $T = O(\log n)$ w.h.p.
Part 2: Lower Bound $T = \Omega(\log n)$ w.h.p.

- Goal: Show i_t cannot grow much faster than doubling.
- Random Variable: i_{t+1}. Expectation $\mu_{t+1} = \mathbb{E}[i_{t+1}] \leq 2i_t$.
- Chernoff (Upper Tail): Use $\delta = 1/2$. $\mathbb{P}(i_{t+1} \geq 1.5\mu_{t+1}) \leq e^{-\mu_{t+1}/16}$. Requires $\mu_{t+1} \geq 32 \ln n$, holds if $i_t \geq C'' \log n$. W.h.p., $i_{t+1} \leq 3i_t$.
- Conclusion: Growth factor ≤ 3 w.h.p. Reaching n requires $T = \Omega(\log n)$ rounds.

Combining bounds, $T = (\log n)$ w.h.p. □

A.3 Proof of Theorem 4.3 (Push-Pull Time Complexity)

Detailed Proof. **Part 1: Upper Bound $T = O(\log n)$ w.h.p.**
Phase 1: Growth ($1 \le i_t < n/2$)

- Goal: Show i_t reaches $n/2$ in $O(\log n)$ rounds w.h.p.
- Random Variable: $Y_t = \sum_{v \in U_t} X_v$, number newly informed.
- Expectation: $\mu_t = E[Y_t] \ge u_t(i_t/n)$, so $E[i_{t+1}] \ge 1.5 i_t$.
- Chernoff (Lower Tail): Lower bound by Y'_{pull} (successful pulls only, independent indicators). $\mu'_{pull} = u_t(i_t/n) \ge 0.5 i_t$. Use $\delta = 1/4$. $\mathbb{P}(Y_t < 0.75\mu'_{pull}) \le \mathbb{P}(Y'_{pull} < 0.75\mu'_{pull}) \le e^{-\mu'_{pull}/32}$. Requires $\mu'_{pull} \ge 64 \ln n$, holds if $i_t \ge C \log n$. W.h.p., $Y_t \ge (3/8) i_t \implies i_{t+1} \ge (11/8) i_t$.
- Duration: Takes $O(\log n)$ rounds w.h.p.

Phase 2: Convergence ($i_t \ge n/2$, so $u_t \le n/2$)

- Goal: Show u_t reaches 0 in $O(\log n)$ rounds w.h.p.
- Random Variable: $Z_t = u_{t+1} = \sum_{v \in U_t} Y_v$, $Y_v = 1$ if v remains uninformed.
- Expectation: $\mu'_t = E[Z_t] \le u_t/2$.
- Chernoff (Upper Tail): Use $\delta = 1/2$. $\mathbb{P}(Z_t \ge 1.5\mu'_t) \le e^{-\mu'_t/16}$. Requires $\mu'_t \ge 32 \ln n$, holds if $u_t \ge C' \log n$. W.h.p, $u_{t+1} = Z_t \le 1.5\mu'_t \le 0.75 u_t$.
- Duration: Takes $O(\log n)$ rounds w.h.p.

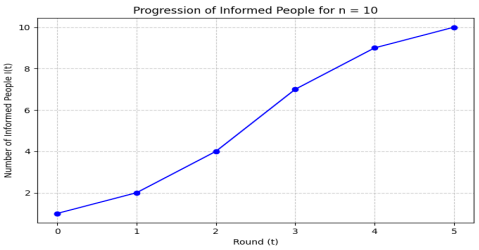
Combining phases, $T = O(\log n)$ w.h.p.
Part 2: Lower Bound $T = \Omega(\log n)$ w.h.p.

- Goal: Show i_t cannot grow much faster than doubling.
- Random Variable: i_{t+1}. Expectation $\mu_{t+1} = \mathbb{E}[i_{t+1}]$. Show $\mu_{t+1} \le c' i_t$ (e.g., $c' = 2$).
- Chernoff (Upper Tail): Use $\delta = 1/2$. $\mathbb{P}(i_{t+1} \ge 1.5\mu_{t+1}) \le e^{-\mu_{t+1}/16}$. Requires $\mu_{t+1} \ge 32 \ln n$, holds if $i_t \ge C'' \log n$. W.h.p., $i_{t+1} \le 1.5\mu_{t+1} \le (1.5c') i_t$.
- Conclusion: Growth factor is constant w.h.p. Reaching n requires $T = \Omega(\log n)$ rounds.

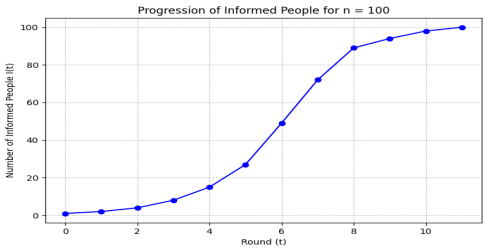
Combining bounds, $T = \Theta(\log n)$ w.h.p. □

B All Experimental Results

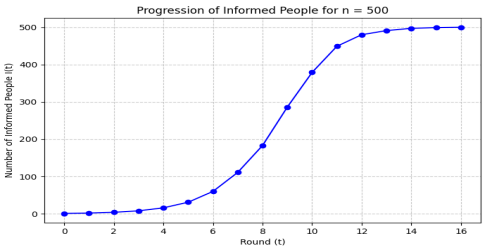
B.1 Sample Progression results



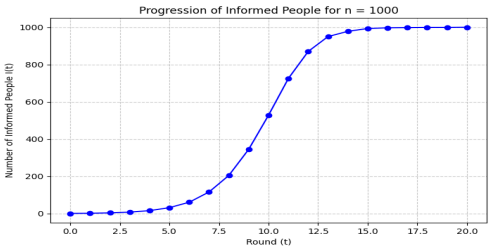
(a) n=10



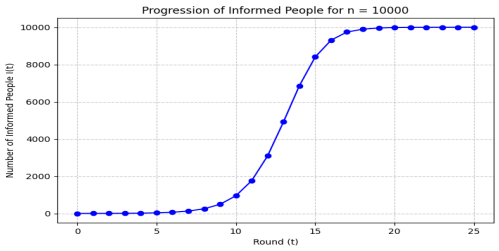
(b) n=100



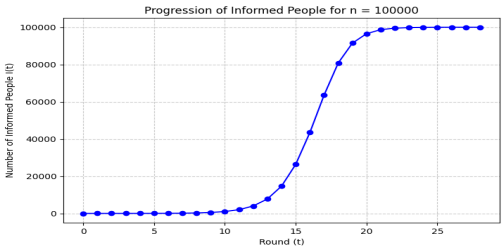
(c) n=500



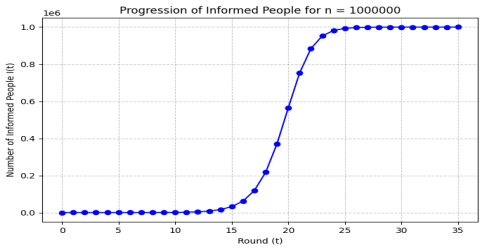
(d) n=1000



(e) n=10000

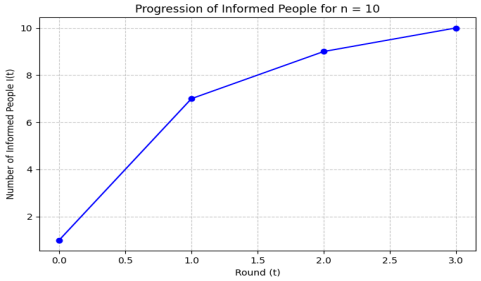


(f) n=100000

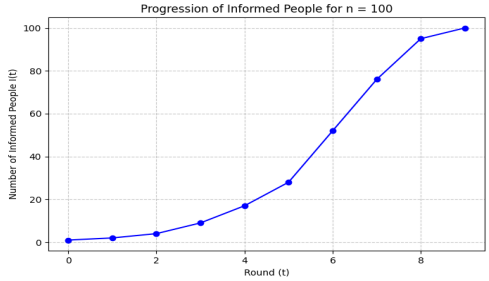


(g) n=1000000

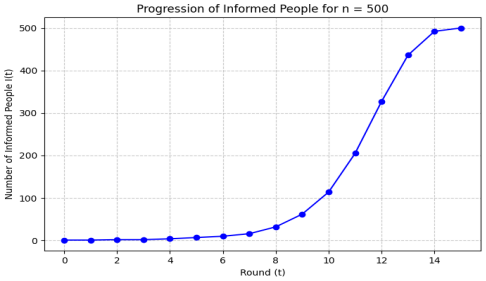
Figure 11: Sample Progressions of Informed People for different n using Push Algorithm



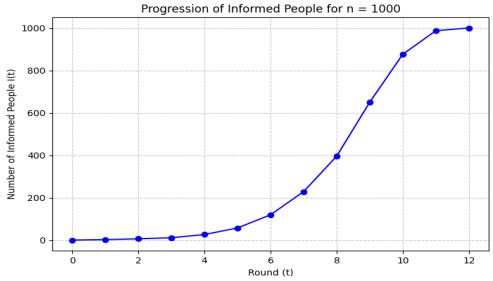
(a) n=10



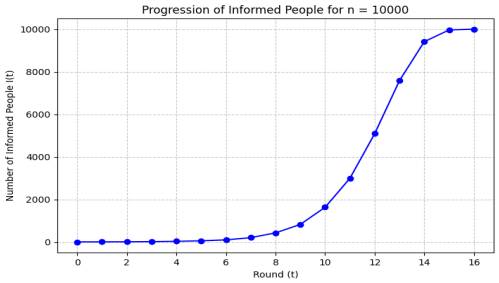
(b) n=100



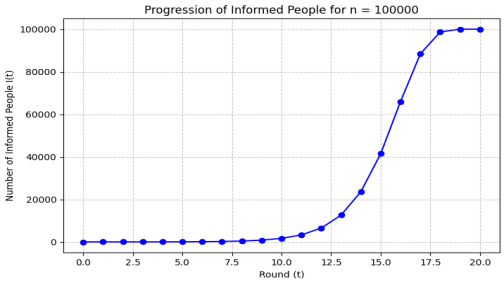
(c) n=500



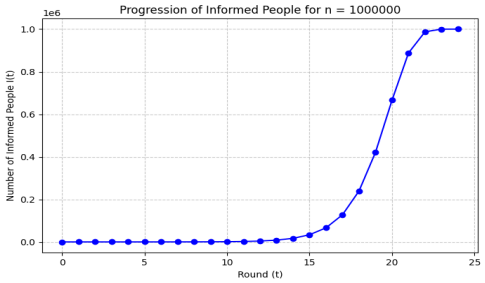
(d) n=1000



(e) n=10000

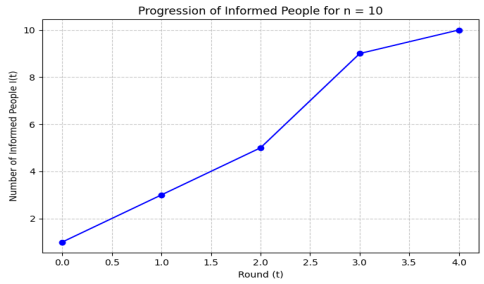


(f) n=100000

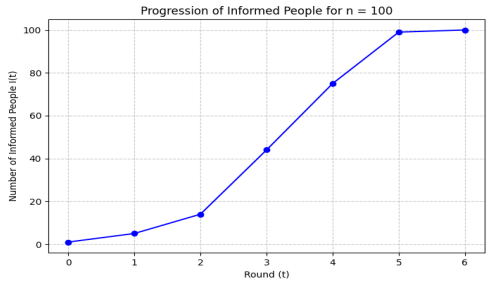


(g) n=1000000

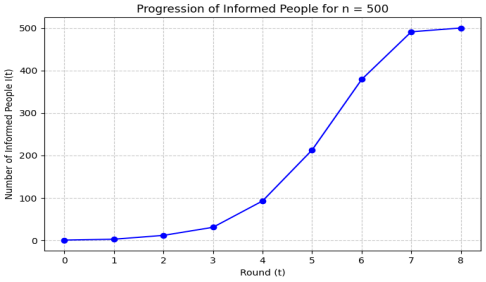
Figure 12: Sample Progressions of Informed People for different n using Pull Algorithm



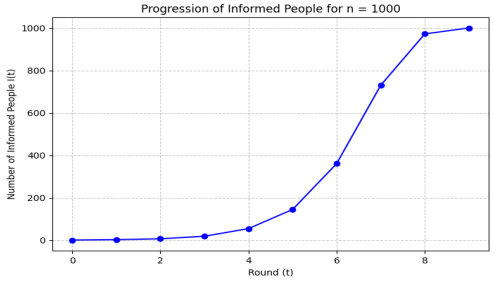
(a) n=10



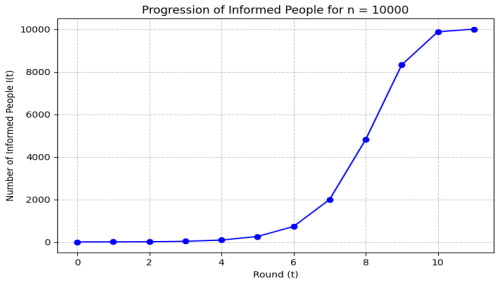
(b) n=100



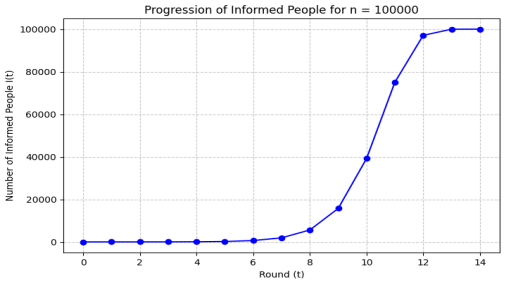
(c) n=500



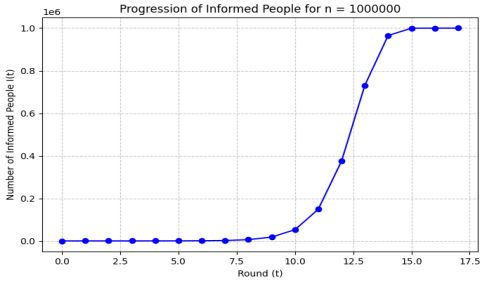
(d) n=1000



(e) n=10000

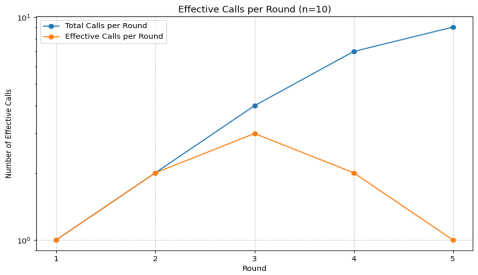


(f) n=100000

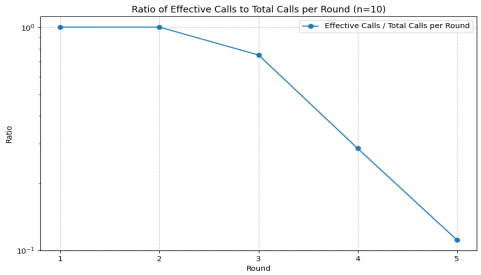


(g) n=1000000

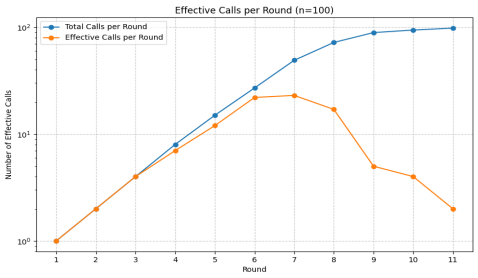
Figure 13: Sample Progressions of Informed People for different n using Push-Pull Algorithm



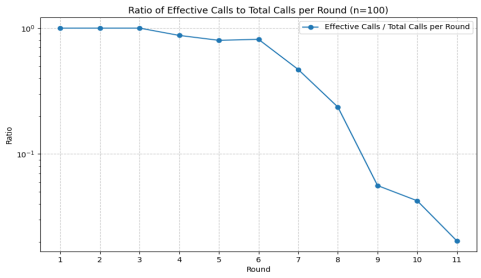
(a) n=10



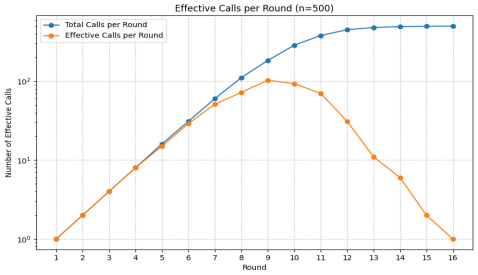
(b) n=10



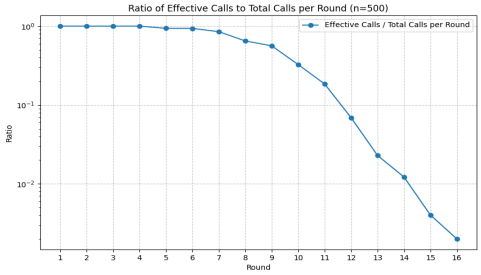
(c) n=100



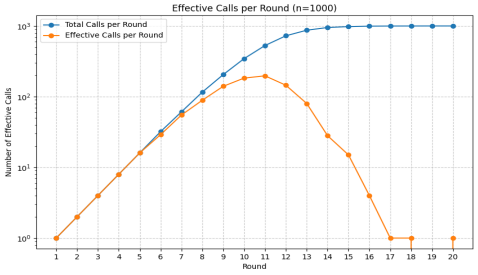
(d) n=100



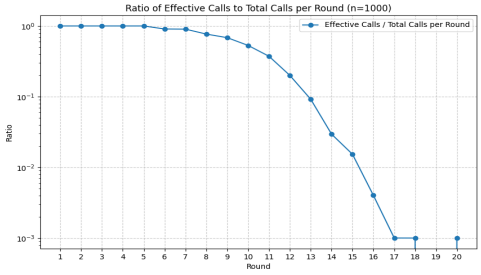
(e) n=500



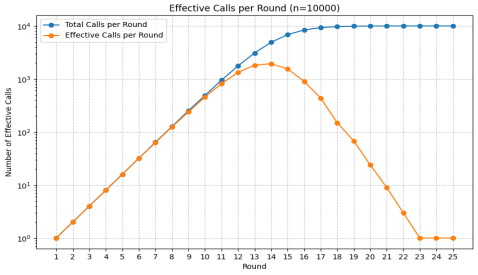
(f) n=500



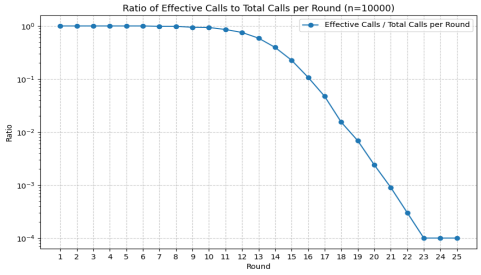
(g) n=1000



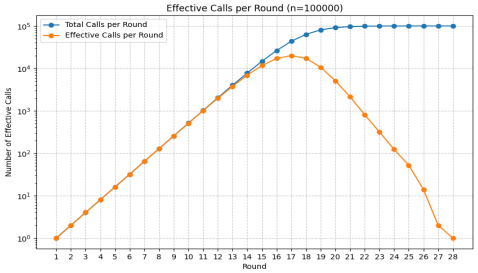
(h) n=1000



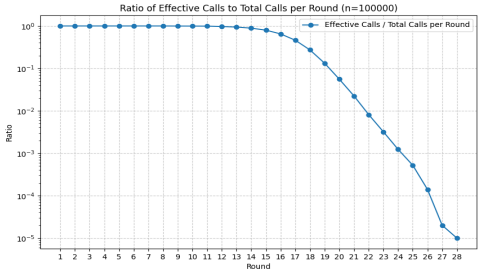
(i) n=10000



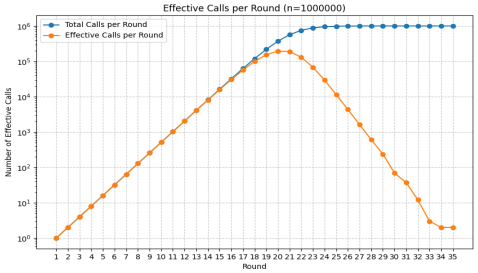
(j) n=10000



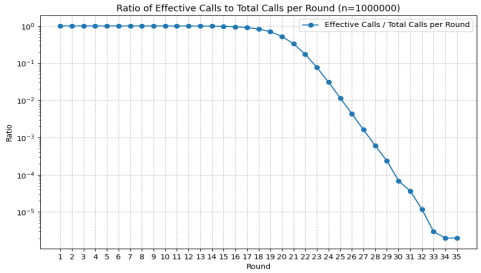
(k) n=100000



(l) n=100000

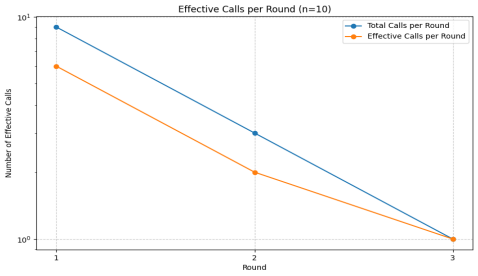


(m) n=1000000

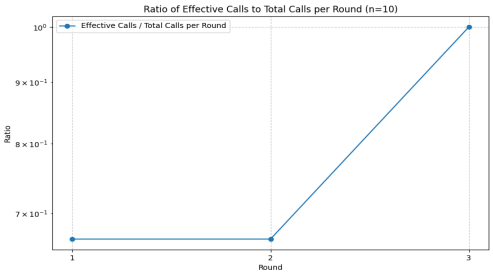


(n) n=1000000

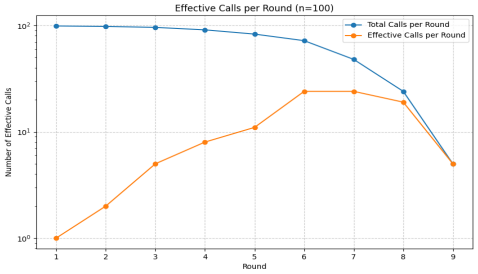
Figure 14: Sample progressions of Effective Calls (plots in left), Ratio of Effective calls/ Total Calls (plots in right) per round, for different n using Push Algorithm



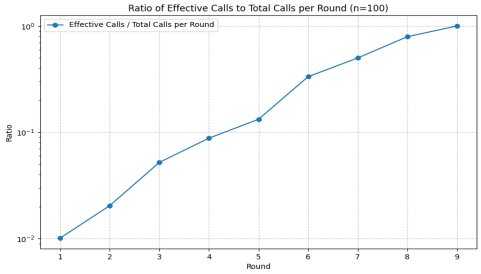
(a) n=10



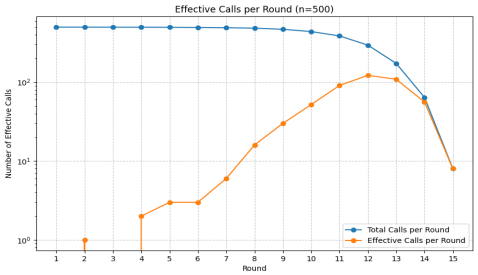
(b) n=10



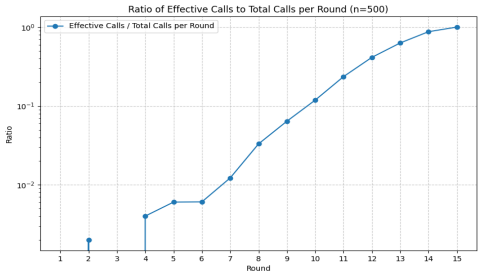
(c) n=100



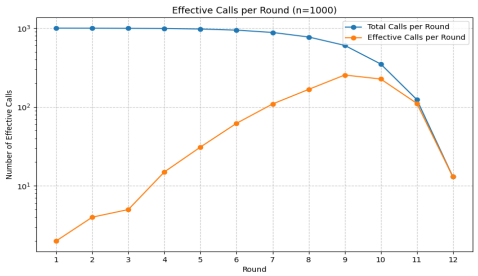
(d) n=100



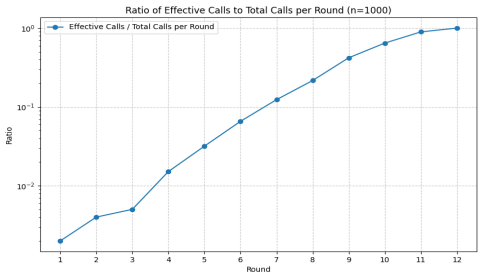
(e) n=500



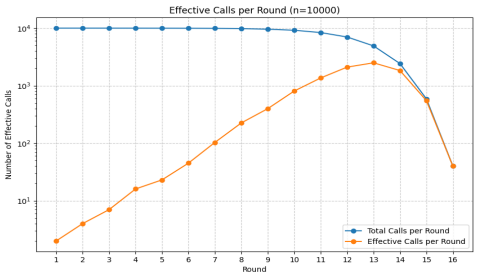
(f) n=500



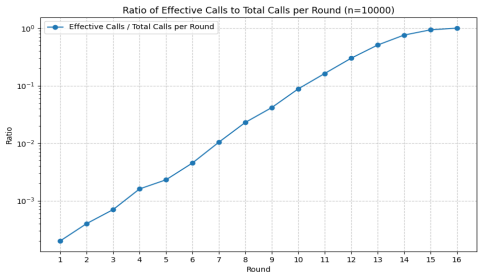
(g) n=1000



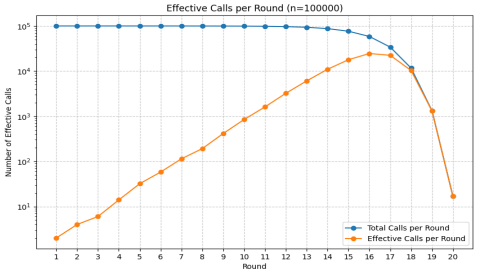
(h) n=1000



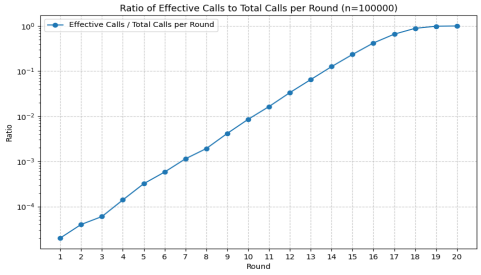
(i) n=10000



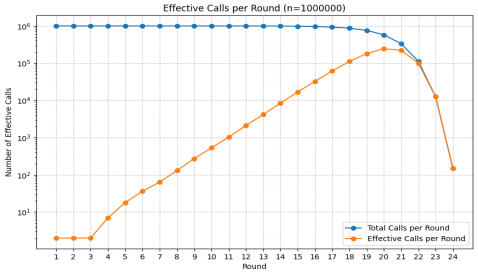
(j) n=10000



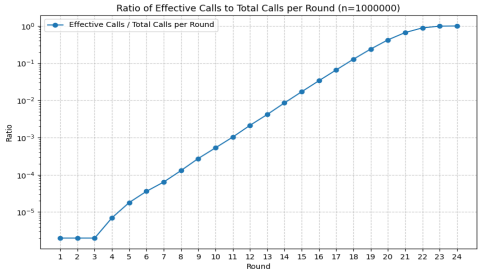
(k) n=100000



(l) n=100000

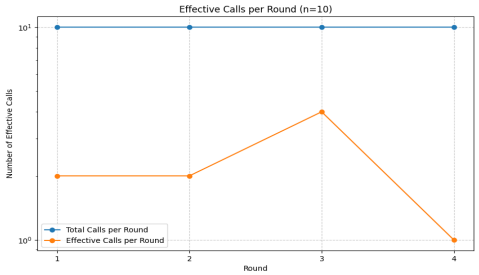


(m) n=1000000

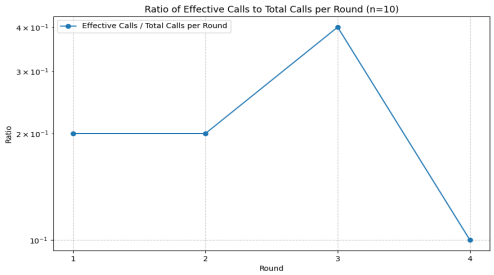


(n) n=1000000

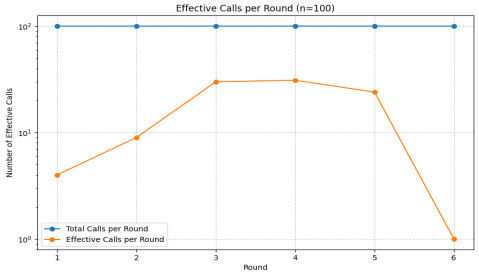
Figure 15: Sample progressions of Effective Calls (plots in left), Ratio of Effective calls/ Total Calls (plots in right) per round, for different n using Pull Algorithm



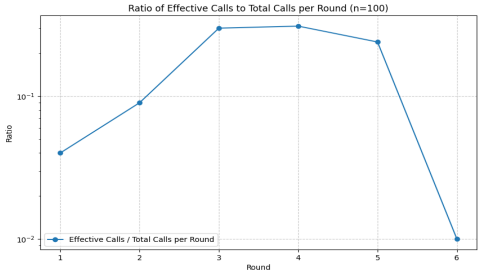
(a) n=10



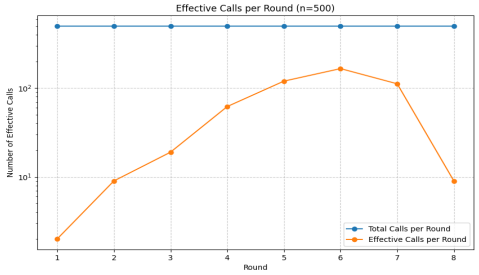
(b) n=10



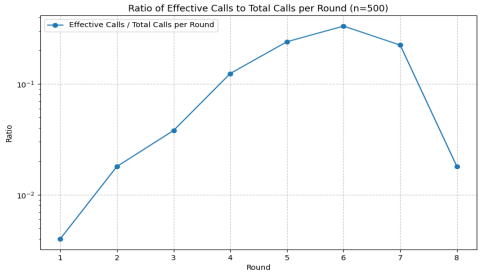
(c) n=100



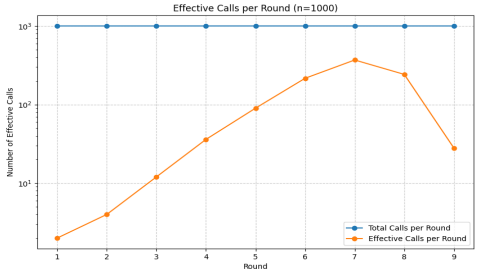
(d) n=100



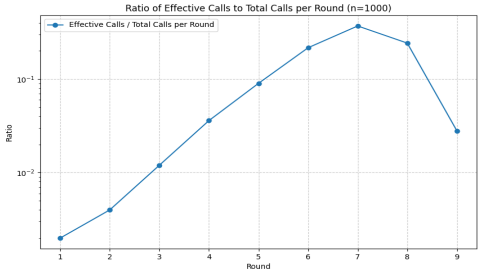
(e) n=500



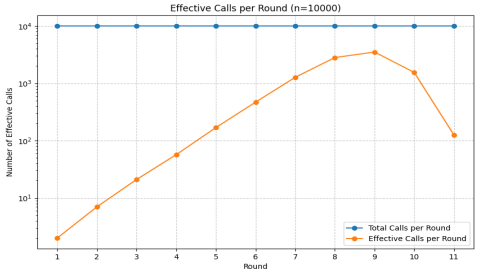
(f) n=500



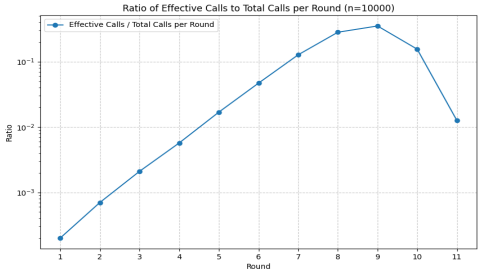
(g) n=1000



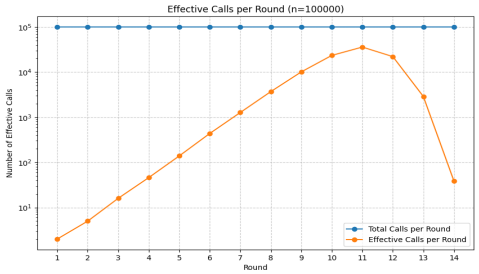
(h) n=1000



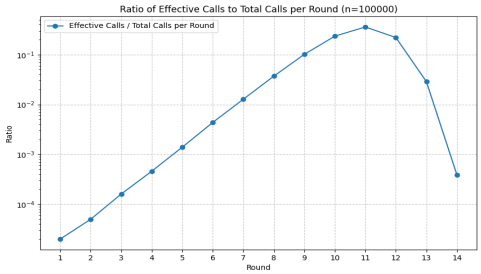
(i) n=10000



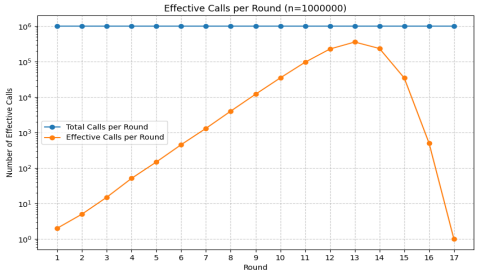
(j) n=10000



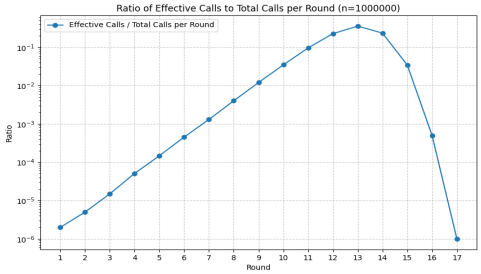
(k) n=100000



(l) n=100000



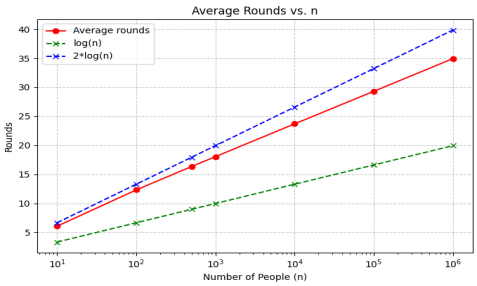
(m) n=1000000



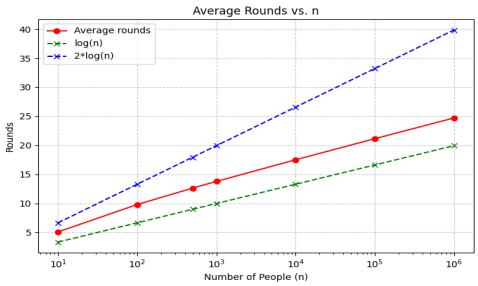
(n) n=1000000

Figure 16: Sample progressions of Effective Calls (plots in left), Ratio of Effective calls/ Total Calls (plots in right) per round, for different n using Push-Pull Algorithm

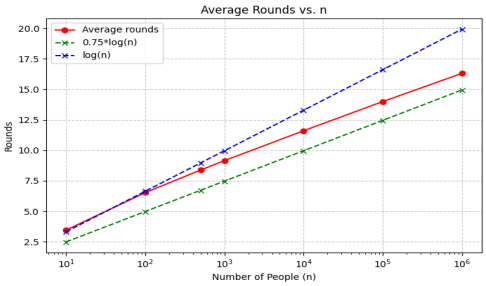
B.2 Results for #Rounds and #Messages



(a) Push Algorithm

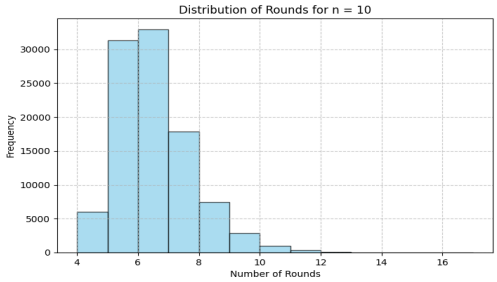


(b) Pull Algorithm

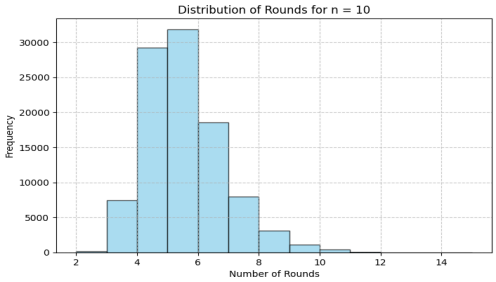


(c) Push-Pull Algorithm

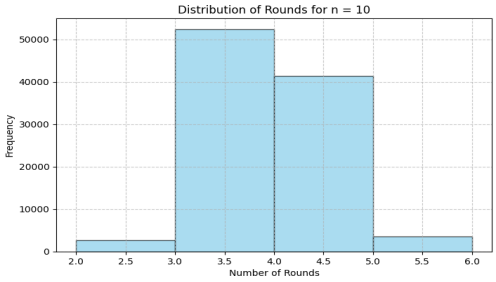
Figure 17: Observed Avg. rounds vs n in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

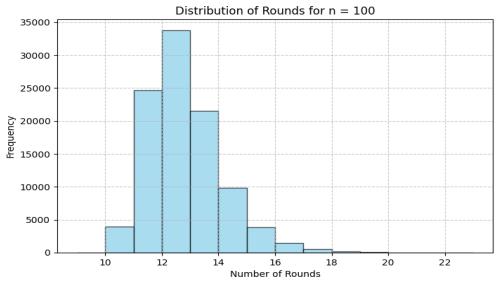


(b) Pull Algorithm

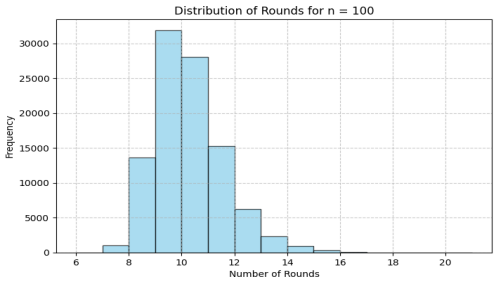


(c) Push-Pull Algorithm

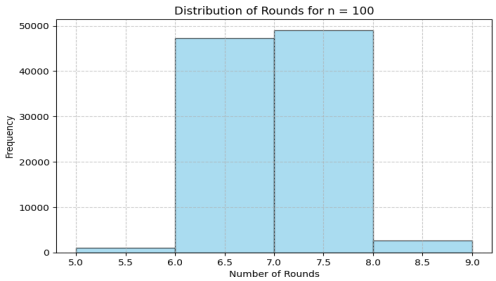
Figure 18: Dist. of Rounds (for n= 10) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

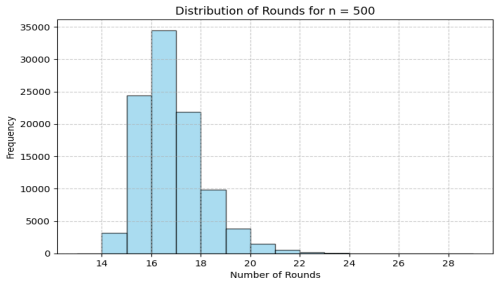


(b) Pull Algorithm

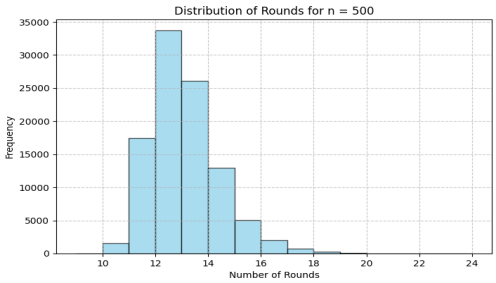


(c) Push-Pull Algorithm

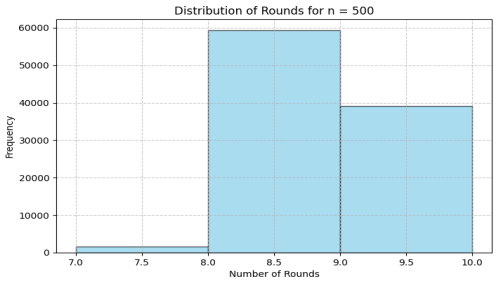
Figure 19: Dist. of Rounds (for n= 10^2) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

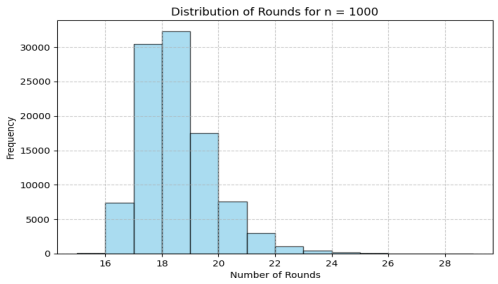


(b) Pull Algorithm

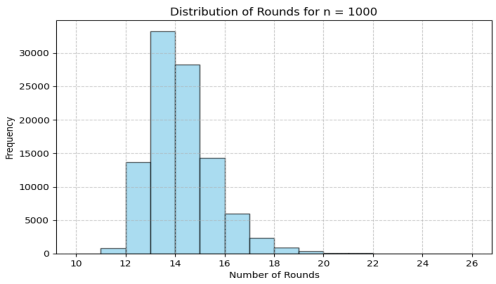


(c) Push-Pull Algorithm

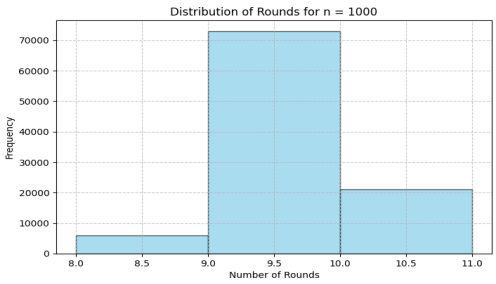
Figure 20: Dist. of Rounds (for n= 500) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

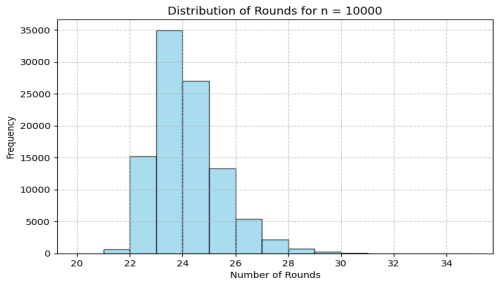


(b) Pull Algorithm

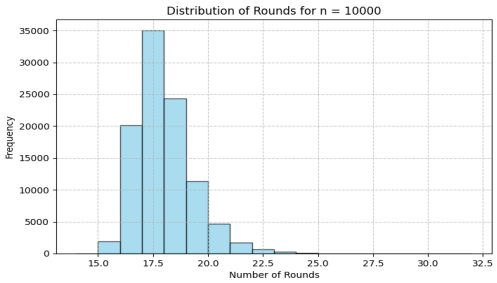


(c) Push-Pull Algorithm

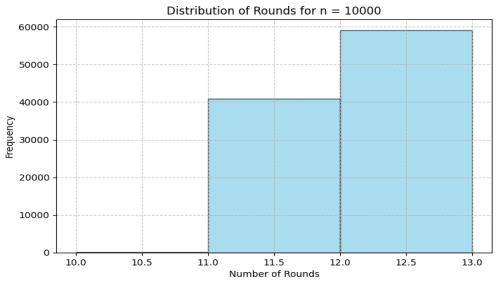
Figure 21: Dist. of Rounds (for n= 10^3) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

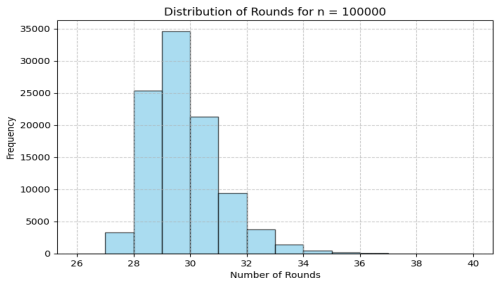


(b) Pull Algorithm

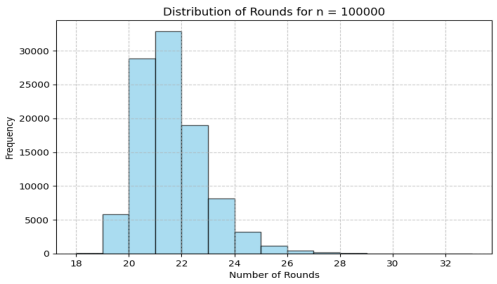


(c) Push-Pull Algorithm

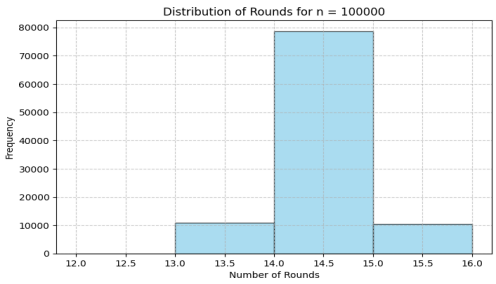
Figure 22: Dist. of Rounds (for n= 10^4) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

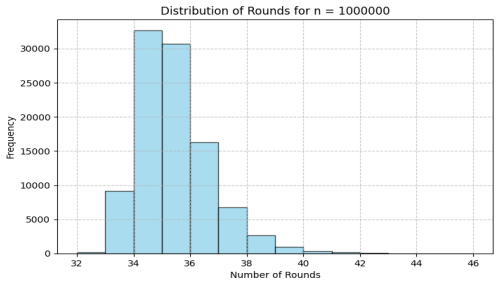


(b) Pull Algorithm

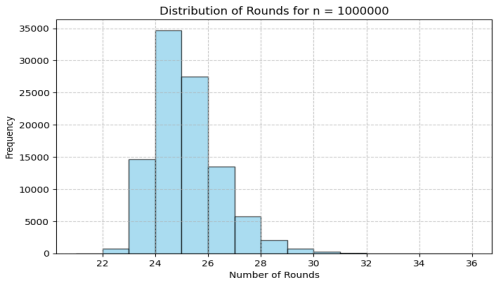


(c) Push-Pull Algorithm

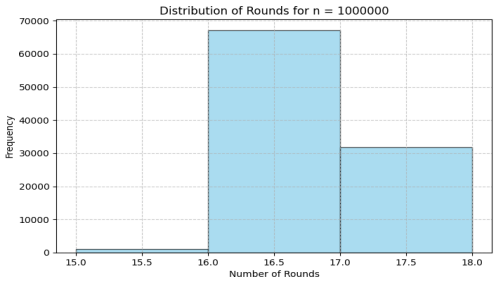
Figure 23: Dist. of Rounds (for n= 10^5) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

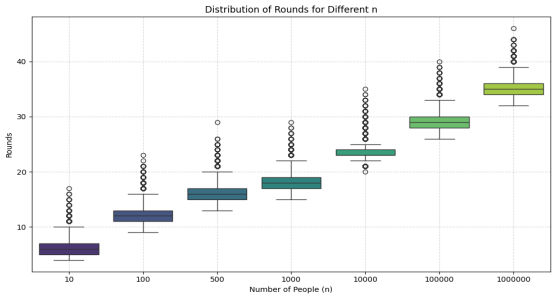


(b) Pull Algorithm

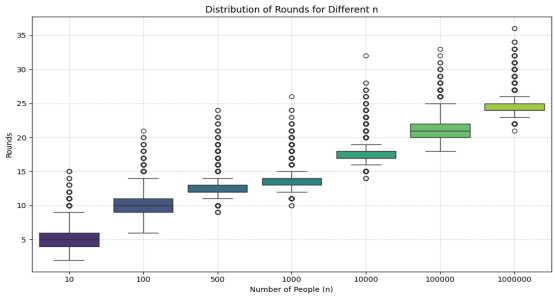


(c) Push-Pull Algorithm

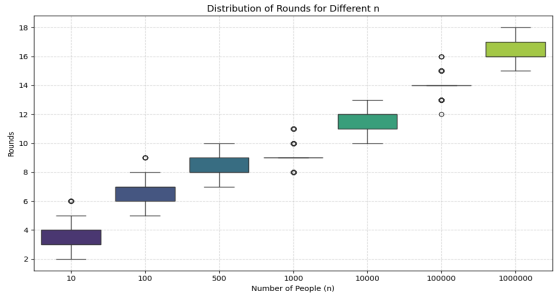
Figure 24: Dist. of Rounds (for n= 10^6) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

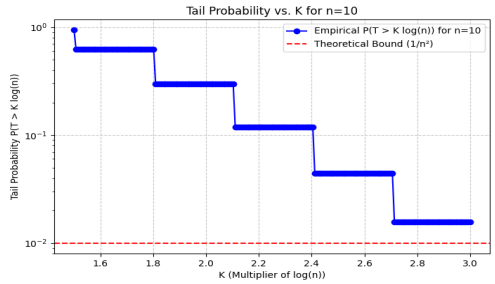


(b) Pull Algorithm

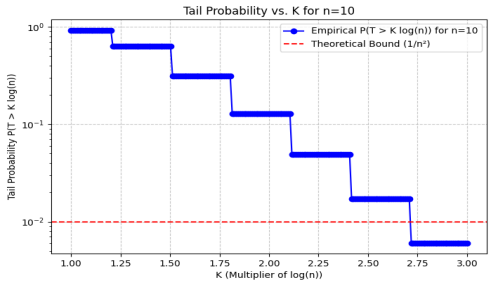


(c) Push-Pull Algorithm

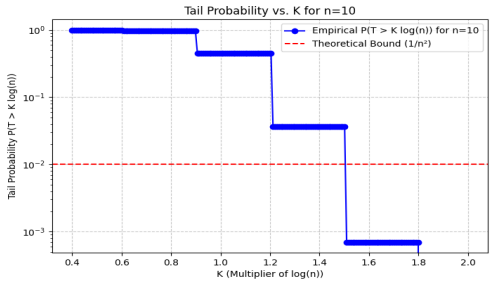
Figure 25: Box-plots of observed rounds for different n in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

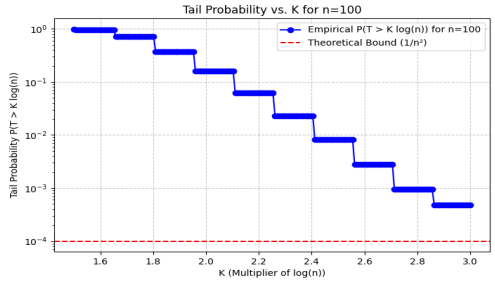


(b) Pull Algorithm

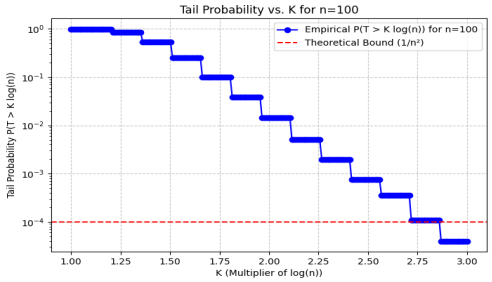


(c) Push-Pull Algorithm

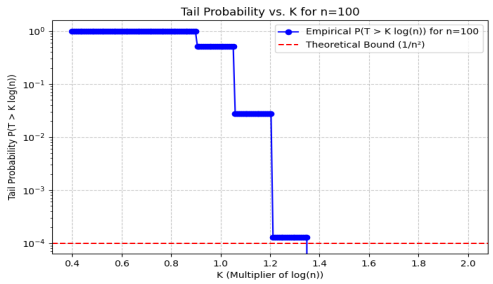
Figure 26: Tail Probability P(#rounds> Klog(n)) vs n (for n= 10) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

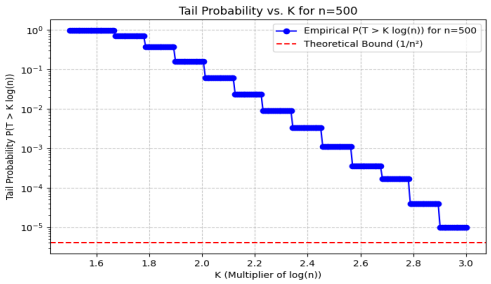


(b) Pull Algorithm

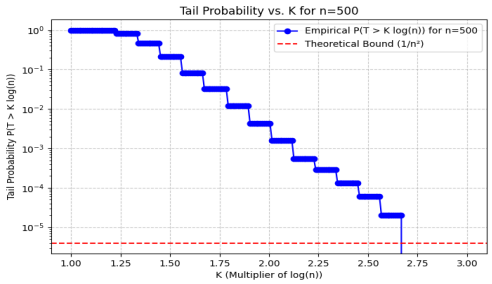


(c) Push-Pull Algorithm

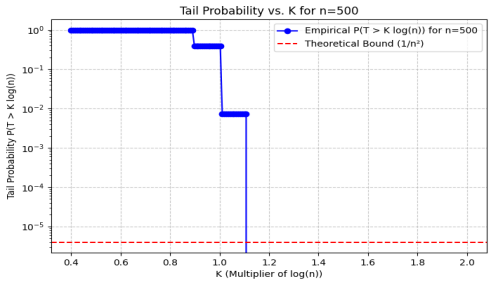
Figure 27: Tail Probability P(#rounds> Klog(n)) vs n (for n= 10^2) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

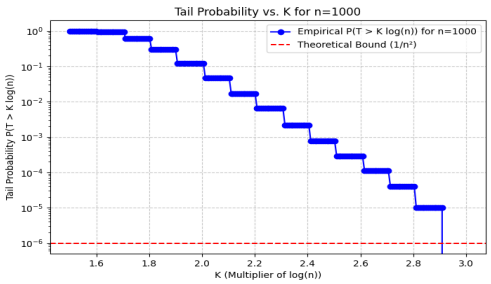


(b) Pull Algorithm

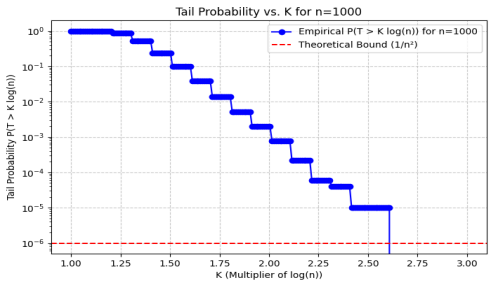


(c) Push-Pull Algorithm

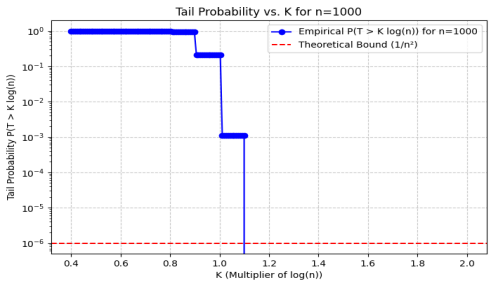
Figure 28: Tail Probability P(#rounds> Klog(n)) vs n (for n= 500) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

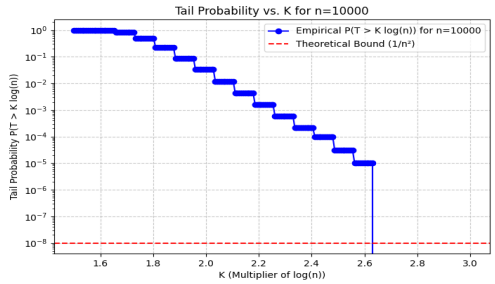


(b) Pull Algorithm

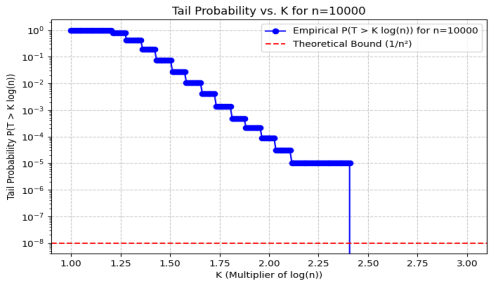


(c) Push-Pull Algorithm

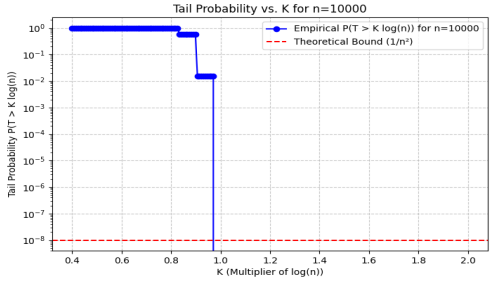
Figure 29: Tail Probability P(#rounds> Klog(n)) vs n (for n= 10^3) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

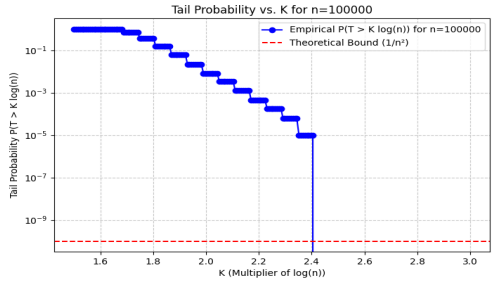


(b) Pull Algorithm

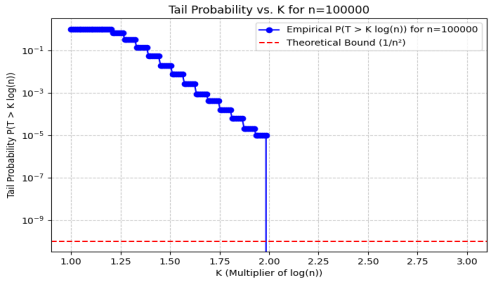


(c) Push-Pull Algorithm

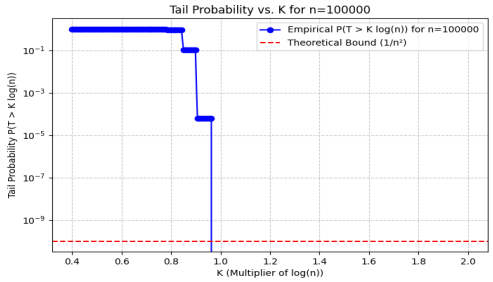
Figure 30: Tail Probability P(#rounds> Klog(n)) vs n (for n= 10^4) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

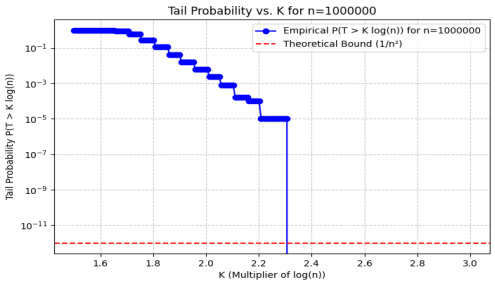


(b) Pull Algorithm

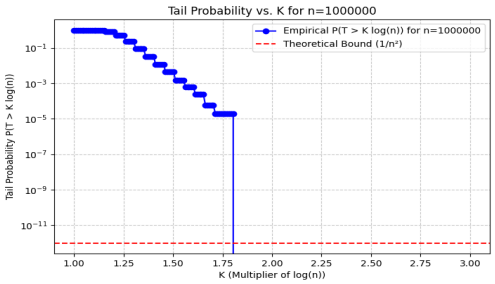


(c) Push-Pull Algorithm

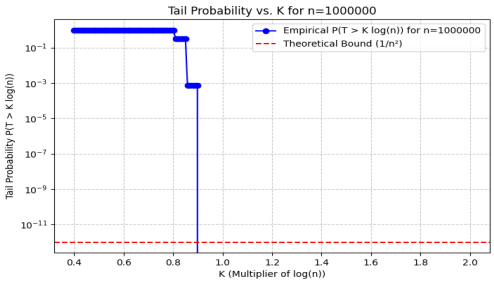
Figure 31: Tail Probability P(#rounds> Klog(n)) vs n (for n= 10^5) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

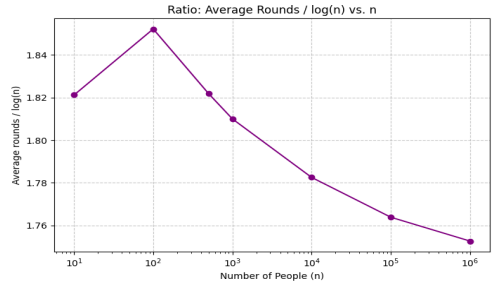


(b) Pull Algorithm

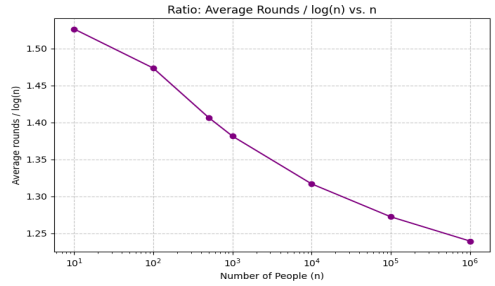


(c) Push-Pull Algorithm

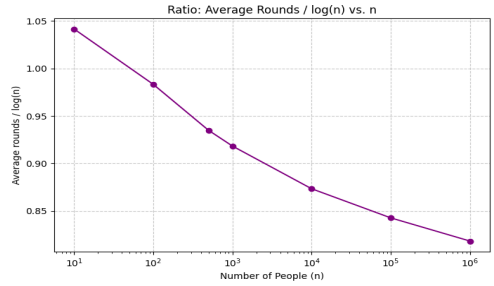
Figure 32: Tail Probability P(#rounds> Klog(n)) vs n (for n= 10^6) in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

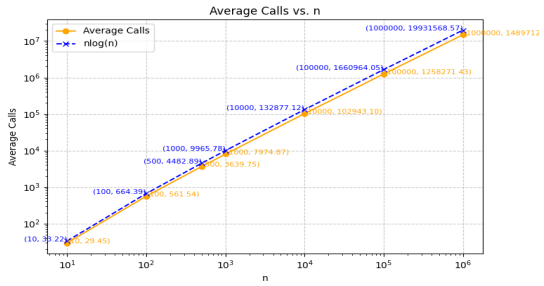


(b) Pull Algorithm

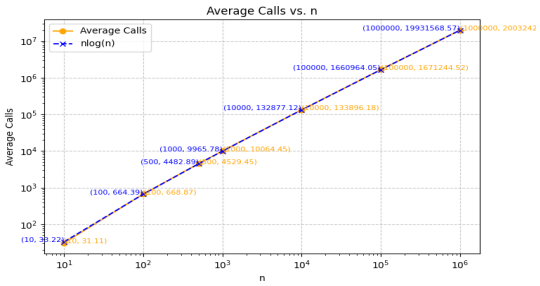


(c) Push-Pull Algorithm

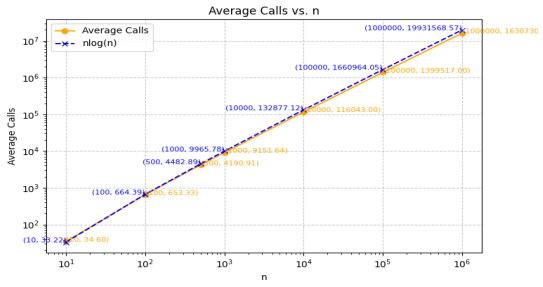
Figure 33: Variation of Ratio = Average Rounds/ log(n) vs. n in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

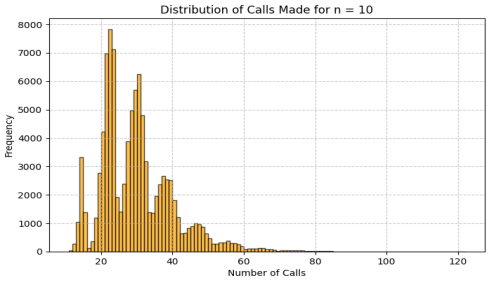


(b) Pull Algorithm

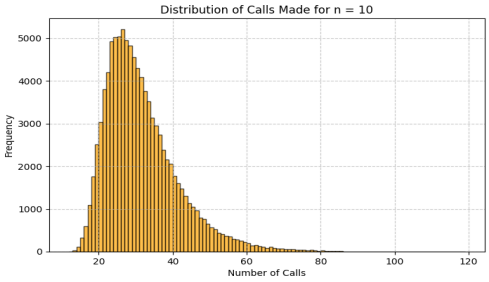


(c) Push-Pull Algorithm

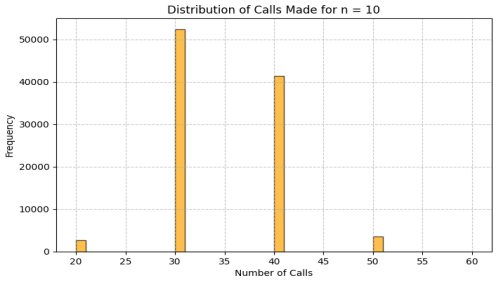
Figure 34: Avg. #total messages/calls made in 10^5 runs of each (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

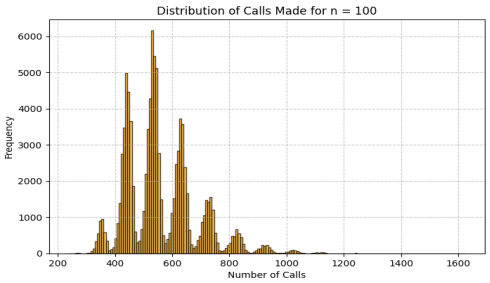


(b) Pull Algorithm

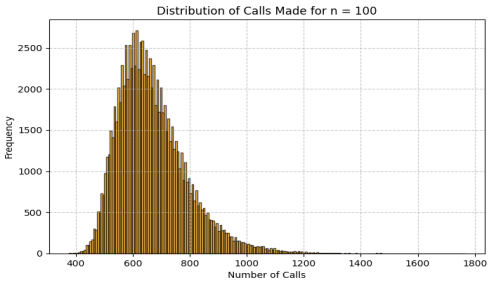


(c) Push-Pull Algorithm

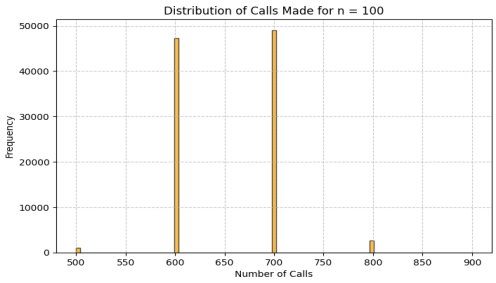
Figure 35: Distribution of #total messages/calls made (for n= 10) in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

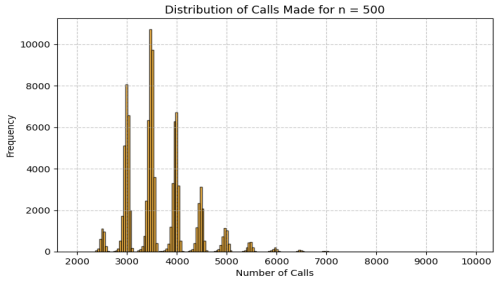


(b) Pull Algorithm

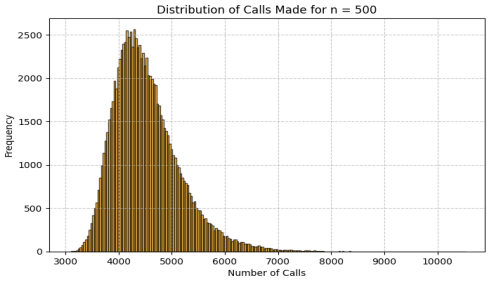


(c) Push-Pull Algorithm

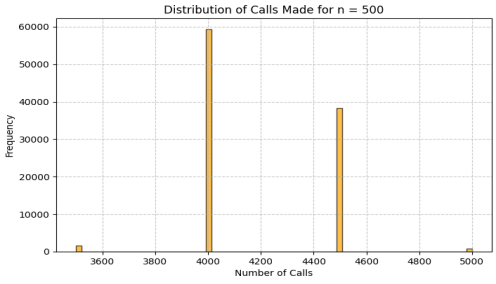
Figure 36: Distribution of #total messages/calls made (for n= 10^2) in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

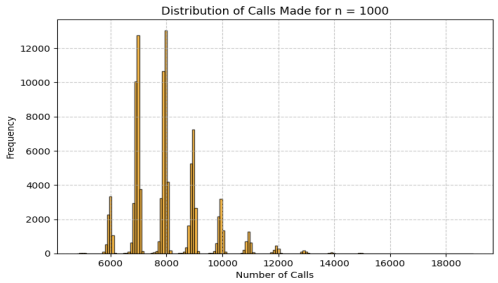


(b) Pull Algorithm

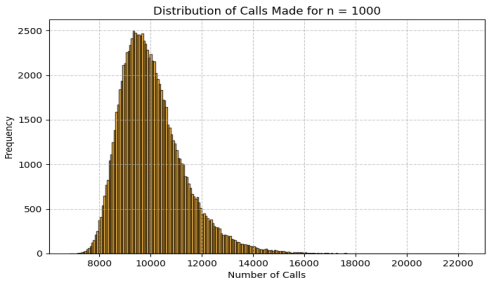


(c) Push-Pull Algorithm

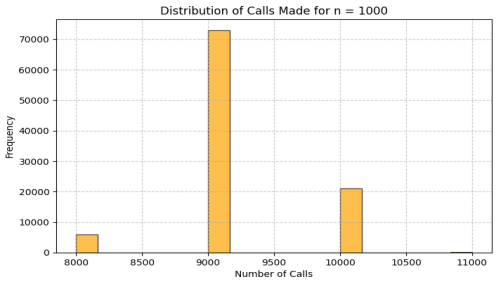
Figure 37: Distribution of #total messages/calls made (for n= 500) in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

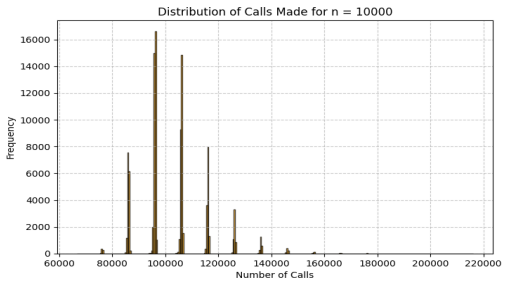


(b) Pull Algorithm

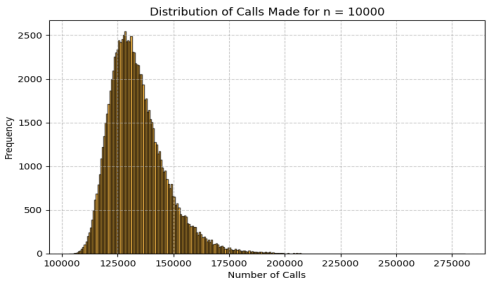


(c) Push-Pull Algorithm

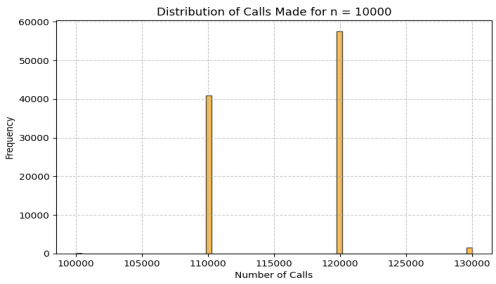
Figure 38: Distribution of #total messages/calls made (for n= 10^3) in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

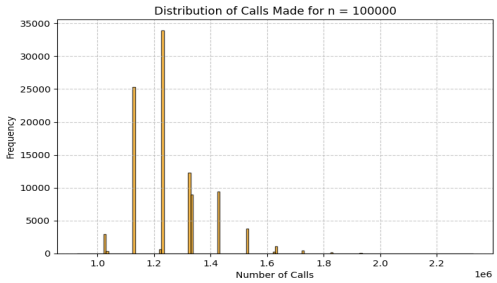


(b) Pull Algorithm

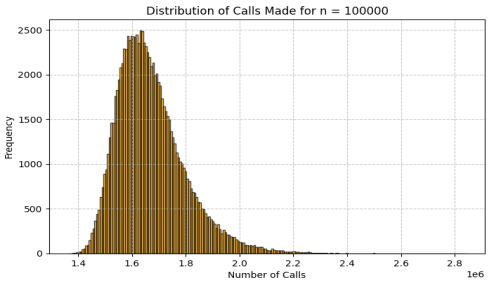


(c) Push-Pull Algorithm

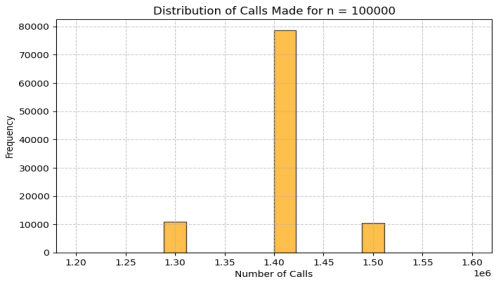
Figure 39: Distribution of #total messages/calls made (for n= 10^4) in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

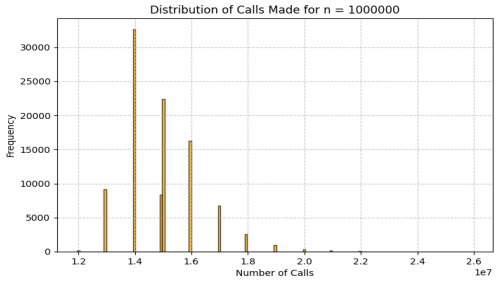


(b) Pull Algorithm

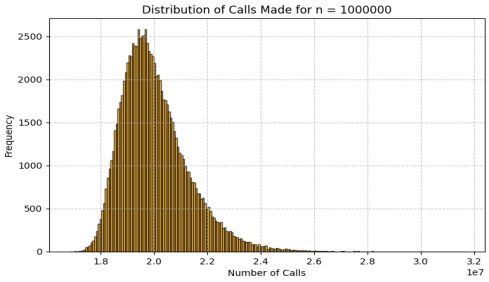


(c) Push-Pull Algorithm

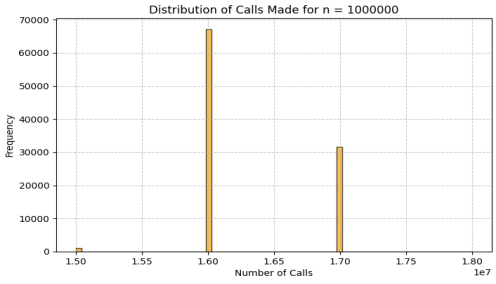
Figure 40: Distribution of #total messages/calls made (for n= 10^5) in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm

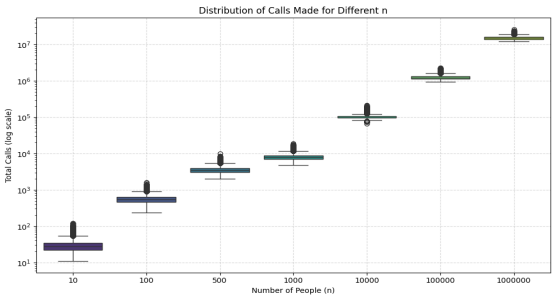


(b) Pull Algorithm

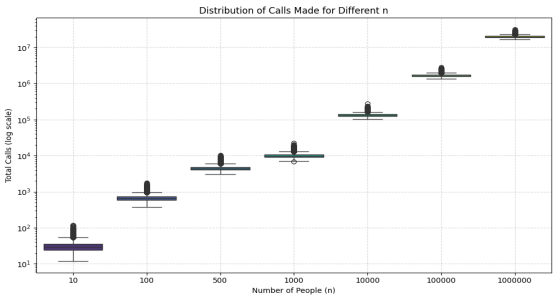


(c) Push-Pull Algorithm

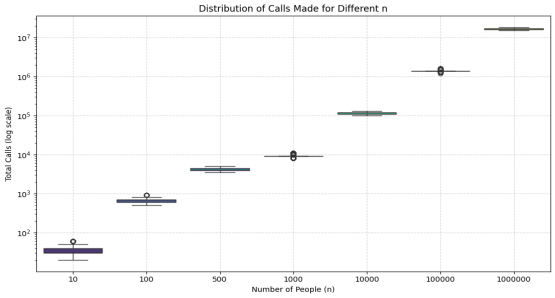
Figure 41: Distribution of #total messages/calls made (for n= 10^6) in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm



(a) Push Algorithm



(b) Pull Algorithm



(c) Push-Pull Algorithm

Figure 42: Box-plots of observed #total messages/calls for different n in 10^5 trials each for (a) Push (b) Pull (c) Push-Pull Algorithm