Practical Project *on Distributed Data-Centric Networks*

Dependable Distributed Systems
Year 2022/2023

Brunetti Jacopo, Carmignani Federico and Caruso Paolo

Abstract

This paper is going to explain a historical paper named **"The price of validity in dynamic networks"**.
It was presented in 2006 in the Journal of Computer And System Sciences by researchers from USA and Finland.

Introduction

The authors concentrate on networks with specific properties:

1. massive-scale;
2. self-administered;
3. dynamic;
4. full of short-lived hosts;
5. distributed;
6. data-centric.

The networks analyzed are *Peer-to-Peer* and *Sensor Networks*.

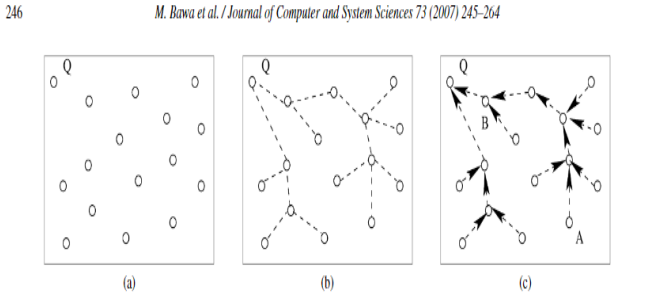

Fig. 1. *Ill-Defined Semantics*: (a) Sensor Network with 16 sensors, (b) Broadcast, and (c) Convergecast. Failure of sensors *A* and *B after* Broadcast leads to counts of 15 and 6, respectively. Which of these is correct and why?

Figure 1. ConvergeCast

Until now the best effort on aggregate queries (sum, max..).
They claim that query semantics are ill-defined and that there are not enough guarantees about them. Therefore, they propose a correctness condition named *Single-Site Validity* applying it to new algorithms satisfying them. Performances are tested on real-life synthetic networks and they show at the end the reason why the paper is called in this way.
Aggregate queries with N hosts are considered and they talk about 2 approaches:

1. *The warehousing approach*: central databases shipping data from hosts where query processing is done and steps are controlled (e.g. concurrency). It reflects data for a single snapshot of the network.
2. *The in-network processing*: with a distributed query plan in which only relevant data are shipped.

The first one has a high communication cost in the network and the central database host. The second one deals with dynamism, like host failures and aborts as in traditional distributed database systems.
They present an algorithm called **ConvergeCast** in which there 2 phases (see Fig.1), the Broadcast phase in which the querying host h_q sends the query q to all the participants of the network and then a Converge phase in which they answer with their values for the query and usually this is sent back to the querying host creating a spanning tree to return from the leaves to the root aggregating the result in each hop.
They want to define why and how in terms of performance and technologies the query semantics and processing are leading to concrete results of the query.
They show also show an example of semantics talking about the result of an online progressive query and their properties of "confidence" and "interval bounds" in terms of approximation of the real final result.
They defined the following concept:

Correctness condition: a query result is *"single-site" valid* if equivalent to a legal failure-free computation as observed from the querying host.

The novelty of the paper is the in-network processing **(WildFire)** scheme for aggregate queries. Continuous and approximate domains are covered by their arguments. They want to focus on validity semantics and they prove that the costs are similar for min and max functions, whereas 5 times higher for count and sum w.r.t. best-effort algorithms, thus denoting the so-called "price of validity".

Related work

Aggregate queries are considered as summarizing the state of large-scale networks deducing 'trends' in P2P and Sensor Networks (load, lifetimes, measures..), and reducing also communication costs in terms of bandwidth.
They resumed a work by Imielinsky and Badrinath on mobile networks for data changing during query evaluations, arising the following questions: how to deal with the meaning of queries in this situation? And in such case, how to augment them?
Until this moment only has 3 related works' models to be analyzed:

- *Append-only continuous query semantics*: Bonnet et al. proposed a data model, where host failures were not handled, for sensor networks; there is 1 central relational host for a persistent view and data from the network arrives as time series in an append-only mode giving a time duration to the specific view in the host.
- *Eventual consistency*: Gossip and epidemic algorithms are fault-tolerant, multiple rounds in which each host exchange information with some hosts: this means "gossiping", giving an eventual consistency to the results since failures and updates are possible. The convergence time is lower bounded by the mixing time of Markov chain representing the algorithm flow [Boyd et al.].
- *Best-effort algorithms*: it deals with lossy query processing, where failures are not handled and in this algorithm, the ConvergeCast is applied. The first phase of Broadcast is where spanning trees are created and then a Convergence until aggregation is done.

The validity metrics of query results are:

1. *Completeness*: % of hosts contributing to the final result;
2. *Relative Error*: $|y/x - 1|$: where y is the reported result and x is the true one.

The probabilistic counting scheme of Flajolet and Martin is used for aggregation operators.

Problem setting

In the problem they are analyzing there is 1 querying host h_q, H hosts in total and aggregate queries that are sent to them,

$$V = q(H)$$

is the result and each host contributes to it having $< a, v >$ pairs as distributed data.

Network model

An undirected graph is a model: $G = (H, E)$, where the number of hosts and symmetric neighbour relations, or edges, are the parameters.
The messages are only sent to their neighbours, and queries are ad-hoc meaning that they are not predictable a priori but exchanged and that there are no cached values or replicated ones.
They adopt an **relaxed asynchronous model** for distributed systems: a max delay is set equal to D and the interval is $(t, t + D)$.
Periodical heartbeats are sent between the hosts.
In P2P networks, each host has a list of IP_s of its neighbours, where TCP connections and synchronized clocks via NTP are present.
While in Sensor networks, unique MAC addresses identify hosts and they communicate via wireless radio for short-range messages (to reduce power consumption), even here clocks are synchronized and asymmetric links are ignored by routing protocols.
All hosts are unaware of the values of the others, so we can say that "all database is needed". It is possible that the value needed is not the one kept by the host itself but computation on it is done before, not known before receiving the query.

Dynamism model

Dynamism is fundamental in the paper, failures (e.g batteries turn-off in sensors) or new hosts are considered part of the algorithms and so how the graph and the neighbourhood sets are updated:

- H_t: hosts of the networks at time t;
- D_t: diameter of the network at time t.

Messages are reliably delivered and overlay network partitions are possible due to failures.
The "small world phenomenon", where D grows slowly with H is applied to the networks we are considering.

Aggregate queries semantics

In the model, the query processing is dynamic and it affects semantics and results in terms of computability and domains.

One-time queries

In the ConvergeCast (see ALLREPORT algorithm in Fig.2), they tried to apply the strictest semantics but they understood it was infeasible and so then it was relaxed, because of the dynamism and distribution, leading to uncertainty in time and operations. They propose the concept of **Snapshot Validity**: an algorithm that computes an aggregate query must guarantee that the value returned is the exact value of the aggregate query at some instant t in the interval considering all hosts were at the beginning of the processing.
They showed it is impossible to prove it in our setting with a relaxed asynchronous model and reliable order communication, because of the time of information flowing and because of failures.

PROTOCOL ALLREPORT $(q, \widehat{D})$
▷ On issue of query q at host h_q
Send Broadcast message $[h_q, q]$ to all neighbors
$M := \{h_q\}$
Set Timer to expire after $2\widehat{D}\delta$
while (Timer not expired) {
▷ On receipt of attribute value a from h
$M := M \cup \{a\}$
}
Output $v = q(M)$
Terminate

▷ On receipt of query q at host $h \neq h_q$
Send Broadcast message $[h_q, q]$ to all neighbors
Send attribute value to h_q
Terminate

Fig. 2. ALLREPORT achieves Single-Site Validity.

Figure 2. ALLREPORT algorithm for ConvergeCast

Then they considered **Interval Validity**, meaning that an algorithm that computes an aggregate query must guarantee that the value is $v = q(H)$ for some set H s.t. $H_I \subseteq H \subseteq H_U$, where the other two parameters are the intersection of the hosts in the interval and the union.
However, they understood there is no algorithm that can guarantee these semantics.
This is because the hosts can be always alive but at a certain moment they can be in a disconnected part of the graph due to failures and so H is s.t. $H \subseteq H_I \subseteq H_U$.
This introduced the problem of **reachability** in the graph, so H_I became H_C, all alive hosts having a stable path towards the querying host. Thus, having $v = q(H)$ for some set H s.t. $H_C \subseteq H \subseteq H_U$, also known as **Single-site validity**.
Considering reachability, they defined D_h as the diameter considered from a host h, and D as the max stable diameter. They are smaller in general than the size of the network and can be overestimated by a constant. They proved that this validity can be applied and it is about the result as based on a view from only h_p and all failures "seen" from it and results are constructed progressively.

Continuous queries

Until now we considered one-time queries, but also long-running and periodic ones are possible in a continuous domain. In this case, just applying what is found for one-time queries, the semantics could degenerate since H_C can be an empty set also in a long time. All parameters in the continuous domain are in an interval and the maximum delay should be large enough to be awaited in the worst case.

Approximate queries

Also, approximate queries are considered because it can happen that quick and dirty answers are acceptable in some scenarios. The ALLREPORT algorithm is modified as RANDOMIZEDREPORT. An example problem is the estimation of the size of a network: h_q floods a message containing an additional parameter p, and each host sends a 1 to h_q with probability $\geq (4/(e^2 * n))ln(2/q)$, aggregation happens until a fixed interval and the values is the sum divided by p. It is going to use fewer messages than the ALLREPORT and it satisfies the **Approximate Single-Site Validity**: an algorithm that computes an aggregate query must guarantee that the answer is $(1-\epsilon) * q(H) \leq v \leq (1+\epsilon) * q(H)$ with probability 1-q for some set H s.t. $H_C \subseteq H \subseteq H_U$ and $0 \leq \epsilon \leq 1$, where the other two parameters are the intersection of the hosts in the interval and the union.

For more information about the theorems check them on the paper.
The problem of this algorithm with a *Direct Delivery* is that the number of messages is high and so the bandwidth should be high. It is obviously possible to achieve Single-Site Validity by spending less and increasing the degree of in-network processing but also dividing hosts into edge-subset (in best-effort algorithms) networks during Broadcast for efficiency. This can be done in 2 ways:

1. *Spanning Trees*: sensitive to failures;
2. *Directed Acyclic Graphs*: not sensitive to failures.

Message sizes are lower and the communication cost is greater for the second one. However, both for dynamism may return results arbitrarily bad, as shown in one of their theorems.

Achieving Single-Site Validity

In this passage, the authors introduce a new in-network algorithm called WILDFIRE that achieves Single-Site Validity for aggregation queries (*minimum, maximum, count, sum, and average*).

WILDFIRE protocol

The paper starts the discussion by analyzing the case of maximum and minimum aggregation queries. Other queries will be discussed later. The protocol has two phases: *Broadcast* and *Convergecast*. A formal description of these two phases is given in Figures 3 and 4, respectively. We say that a host is *inactive* at time *t* if it is not participating in the protocol at that time; otherwise, it is *active*. At the start of the protocol, h_q is active, while all other hosts are inactive. Each active host h maintains a partial aggregate A_h which is initiated on the transition to the active state. The initial value of A_h is set to be the relevant attribute value at the host h for

minimum and maximum queries.

During the Broadcast phase, the host h_q initiates the protocol by sending a message containing the query q, the initiating time 0, and an overestimate of network stable diameter $\widehat{D}$ to all its neighbours. A host which receives the Broadcast message checks if it is inactive. If so, the host changes state to active, initializes Ah and sends the message to all its neighbours. Otherwise, the host drops the message. The Broadcast phase completes when all hosts have received the query and the two parameters.

When a host h become active, switching to the Convergecast phase, it initializes its partial aggregate A_h and sends the message to all its neighbours. An active host h that receives a partial aggregate $A_{h'}$ from its neighbour h' recomputes its own partial aggregate A_h using a query-dependent "combine" function. The combined function for *minimum* and *maximum* queries is the query itself. If the host detects a change in its partial aggregate, it sends the new A_h to all its neighbours. Each host continues to participate in Convergecast until $2\widehat{D}\delta$ time. At the end of Convergecast, h_q declares its partial aggregate to be the query result.

The authors prove that WILDFIRE guarantees Single-Site Validity for duplicate-insensitive aggregate operators.

PROTOCOL WILDFIRE-PHASE I
- ▷ On receipt of Broadcast message $[q, 0, \widehat{D}]$
- ▷ at $h \neq h_q$ from h' at time t
- **if** (h is inactive & $t < 2\widehat{D}\delta$) {
  - Change state to *active*
  - Initialize A_h
  - Send Broadcast message $[q, 0, \widehat{D}]$ to neighbors
- }

Figure 3. Broadcast phase in WILDFIRE protocol.

PROTOCOL WILDFIRE-PHASE II
- ▷ On receipt of Convergecast message $[q, A_{h'}]$
- ▷ at h from h' at time t
- **if** ($t \leqslant 2\widehat{D}\delta$) {
  - $A_h^{new} := Combine(q, A_h, A_{h'})$
  - ▷ Propagate A_h^{new} in the network
  - **if** ($A_h^{new} \neq A_h$) {
    - $A_h := A_h^{new}$
    - Send Convergecast message $[q, A_h]$ to neighbors
  - } **else if** ($A_h \neq A_{h'}$) {
    - Send Convergecast message $[q, A_h]$ to h'
  - }
- } **else** Terminate

Figure 4. Convergecast phase in WILDFIRE protocol.

Duplicate-insensitive AGGREGATE operators

We focus now on the case of computing *count* and *sum* aggregates in which the conventional combine function $(+)$ for both is duplicate sensitive. The authors make use of a modified version of an algorithm by Flajolet and Martin. This algorithm ensures that for every $c > 2$, given a set M of elements drawn from a set V of size n, it outputs an estimate $\widehat{m}$ of the number of distinct elements m in M such that $Pr(\frac{1}{c} \leq \frac{\widehat{m}}{m} \leq c) \geq 1 - \frac{2}{c}$.

The authors' algorithm involves the creation of c vectors $B_1^v, B_2^v, ..., B_c^v$ (each with a specific bit set to 1) for each element v in M, and then logically OR the $|M|$ vectors to obtain the final vectors $B_1, B_2, ..., B_c$.

The input to distributed procedure consists of a set M of values distributed across $|M|$ hosts. Host h_q initiates *Broadcast* and includes parameter c in its message. On receipt of the Broadcast message, each host creates c vectors $B_1^h, B_2^h, ..., B_c^h$. Each host now pretends to have an element distinct from other hosts by simulating mapping functions $map_1, map_2, ..., map_c$ as follows. A total of c fair coin toss sequences are generated each of which ends when the first *Head* in the sequence is observed. Host h sets b_i bit in B_i^h 1 where b_i is the index of the last Tail in the ith sequence.

The final vectors $B_1, B_2, ..., B_c$ are to be obtained by performing a logical OR of the corresponding vectors across the M hosts. Since OR is a duplicate-insensitive operator, we can use WILDFIRE to assure the Single-Site Validity of the final vectors. Each active host initializes its partial aggregate $A_h := (B_1^h, B_2^h, ..., B_c^h)$.

On transition into the *Convergecast* phase, each host sends its A_h to its neighbours. On receiving a partial aggregate from its neighbour, a host recomputes its new A_h using the logical OR of the vectors as the combined function. At the end of Convergecast, h_q identifies the lowest-order bits z_i in B_i that are still 0. The average value $\overline{z} = \sum_{i=1}^{c} \frac{z_i}{c}$ is computed and $2^{\overline{z}}/0.78$ is returned as the count.

Our distributed *sum* procedure is similar to our *count* adaptation of the initial algorithm. The *sum* takes as input a set M of values drawn from $[0, V]$ distributed across $|M|$ hosts, with one value h at each host. The final output is an estimate of the sum of elements in M.

Host h_q initiates *Broadcast* and includes parameters c and $\log M$ in its message. On receipt of the message, a host creates c vectors $B_1^h, B_2^h, ..., B_c^h$ of size $O(\log |M||V|)$ each. Each host now pretends to have h elements distinct from other hosts and runs the counting procedure h times. Let the count procedure for the ith ($1 \leq i \leq h$) element generate vectors $B_1^{h,i}, B_2^{h,i}, ..., B_c^{h,i}$. The host then sets its vectors $B_1^h, B_2^h, ..., B_c^h$ to be a logical OR of the count vectors: $B_j^h = \bigvee_{i=1}^{h} B_j^{h,i}$ where $1 \leq j \leq c$.

Once the vectors are initialized, hosts participate in *Convergecast* as before. At the end of Convergecast, h_q computes $\overline{z} = \sum_{i=1} * c\frac{z_i}{c}$ as before and reports $2^{\overline{z}}/0.78$ as the sum.

So, the authors end the explanation of the algorithm asserting that for every $c > 2$, given a set H, our count and sum procedures output an estimate $\widehat{v}$ of the actual value v such that $Pr[\frac{1}{c} \leq \frac{\widehat{v}}{v} \leq c] \geq 1 - \frac{2}{c}$. Furthermore, the WILDFIRE($q, \widehat{D}$) algorithm guarantees Approximate Single-Site Validity within a factor c with probability at least $1 - \frac{2}{c}$ for the class of *count, sum or average* queries.

Discussion

At first glance, it appears that the WILDFIRE protocol is inefficient.
In the worst case, every host will observe an update to its partial aggregate at every time instant during the query processing interval. Each update causes a host to propagate its new partial aggregates to its neighbours causing a worst-case traffic of $2\widehat{D}|E|$ messages as opposed to $|E| + |H|$ in SPANNINGTREE.
Experiments on real-life networks, however, demonstrate that such worst-case behaviour is rarely observed as early aggregation reduces updates at hosts. In addition, WILDFIRE can be engineered to improve efficiency. WILDFIRE can also leverage domain capabilities: the broadcast ability of the wireless medium in Sensor Networks allows a host to send its partial aggregate to all its neighbors by a single message, reducing the worst-case cost.

Continuous approximate count queries

The authors consider now the problem of estimating the size of a dynamic network by trying to find a schema that is not protocol-specific. They present one that enables Continuous Approximate Single-Site Validity for estimates of $|H|$ and makes minimal assumptions on the network protocol. Specifically, the scheme makes the following assumptions:

1. Every host in the network has the same probability of occurring in a sample.
2. Sampling is "instantaneous" with respect to host lifetimes.
3. Every host in the network has the same probability of leaving the network at each instant.

The scheme samples hosts at periodic intervals. Let the sample sets of hosts be $N_1, N_2, ...$, where N_t is the sample set taken at the t-h interval. Let M_t denote a subset of hosts that are *alive* during the time in which the sample set N_t is obtained; we refer to the hosts in M_t as *marked* hosts. We assume that the querying host maintains such a subset of alive hosts. Now let m_t be the number of marked hosts that are found in the sample N_t, that is, $m_t = |Mt \cap Nt|$.
The authors provide an empirical estimate for the size of the network as $\widehat{H_t} = \frac{|Mt||Nt|}{m_t}$.
The above approximation holds with such probability if the number of samples satisfies $|N_t| \geq \frac{4}{\epsilon^2 \rho_t} \ln \frac{2}{\delta}$, where $\rho_t = \frac{|M_t|}{|H_t|}$ is the fraction of the marked hosts in the population at time interval t. The parameters ϵ and δ are selected by the user according to the specific requirements on accuracy and probability of success.
Of course, the quantity $|H_t|$ is unknown and thus one cannot calculate precisely the required sizes of the sets M_t and N_t. However, in practice, a very crude estimate on $|H_t|$ can be very useful in setting the parameters of the algorithm, for instance, the estimate $\widehat{H}_{t-1}$ from the previous time instance can be used.

Paper's Empirical Evaluation

At this point, the paper compares the performance of the protocols used, analysing their efficiency, accuracy and cost, these are Wildfire, SpanningTree and DirectedAcyclicGraph. Before defining the topologies used, each host in the graph must be assigned an attribute value, which is drawn from a Zipfian distribution with values between 10 and 500 (including the endpoints of this interval) in order to evaluate the accuracy of the studied topology.
The concept of topology refers to how the hosts are distributed in the graph, in this scenario are used two types of topology, these are *real-life network topology*, which is a topology that reflects a real topology, and *synthetic network topology*, which is a topology created ad-hoc for network simulation. Four topologies are described here:

- **Gnutella**, this is a real-life network topology, with a number of hosts near to 40K.
- **Random**, this is a synthetic network topology, with a number of hosts equal to 40K, this is built by placing edges between nodes with a uniform distribution and in such a way that the average degree of the nodes is 5.
- **Power-law**, this is a synthetic network with a number of hosts equal to 40K, this is built in order to have a power-law distribution, with $\lambda = 2.9$ in host degrees
- **Grid**, this is a synthetic topology, with a number of hosts equal to 10K, this is built by placing the hosts in a 100×100 grid.

The performance of the protocol is evaluated using three measurements that describe its efficiency: communication cost, computation cost, and time cost. The *communication cost* of the protocol is obtained by the sum of the number of messages sent between any pair of hosts. The *computation cost* of a host is the number of messages that it processes, and the *computation cost* of the protocol is the maximum among the computation cost of the hosts in the network. The *time cost* of the protocol is the length of the longest chain of messages that occurs before the termination of the protocol.

Accuracy evaluation

The paper focuses on studying the accuracy of the *sum* and *count* operators. The first step was to generate a set

of elements with a Zipf distribution called M with variable cardinality $(2^{10}, 2^{12}, ...)$. On this set the operation *count* is used to estimate the number of elements and the operation *sum* to estimate the total sum of elements. The ratio between the estimate $\bar{m}$ and the computed answer m represents the accuracy of the operators. If the ratio is $\bar{m}/m < 1$, this represents an underestimate, if it is $\bar{m}/m > 1$, this represents an overestimate, and if it is $\bar{m}/m = 1$, this represents perfect accuracy. Figure 5 is a graphic showing the relationship between the ratio of operators (y-axis) and the number of repetitions performed (x-axis). The analysis shows that the ratio quickly converges to 1 as the number of repetitions increases, which means that WildFire performs very well even when the number of repetitions is limited.

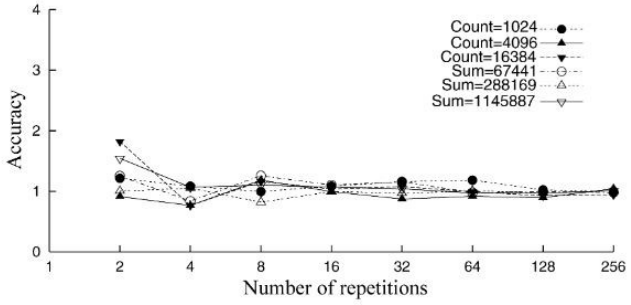

Figure 5. The legend on the top-left indicates the curve related to a specific scenario defined by the operation and the size of the set M.

Achieving Single-Site Validity

The focus now shifts to how SpanningTree, DirectedAcyclicGraph and WildFire protocols behave in different degrees of dynamism in the network. Figure 6 is a graphic showing the relationship between the query result in v (y-axis) and the number of hosts leaving the computation (x-axis). The shown analysis is performed over the Gnutella topology. In the Figure the curves related to Oracle show the upper bound and lower bound for Single-Site Validity, this oracle detects the reachability of each host from the one where the computation starts and uses this information to compute the bounds. In a real scenario, ORACLE is not feasible.

If the result of the query v is less than the lower bound, it means that there are hosts that were not included in the query process even though they were alive in G. Observing the graph, it is possible to state that all the protocols behave in a similar way in situations of low dynamism (low R value). As the dynamism increases, both SpanningTree and DirectedAcyclicGraph rapidly fall below the lower bound for Single-Site Validity. WILDFIRE, on the other hand, continues to return values within the Single-Site Validity bounds even for high dynamism rates. DirectedAcyclicGraph shows improvement over SpanningTree but becomes unable to provide valid answers as dynamism increases.

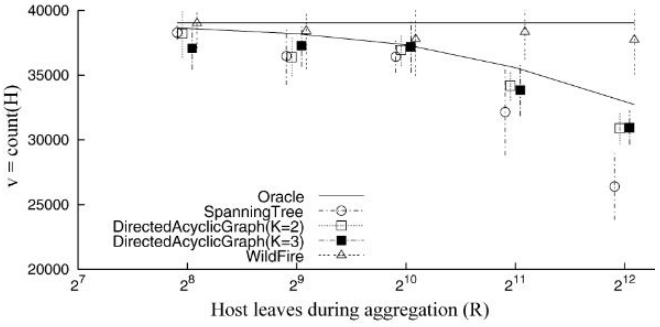

Figure 6. Analysis for *Count* query.
Similar analysis can be done over *Sum* query

Price of Single-Site Validity

From the previous section, it is possible to state that WildFire is the only protocol, among those studied, which guarantees Single-Site Validity when the environment is highly dynamic. To ensure this, the protocol pays a price in terms of the number of messages to be exchanged, and the communication cost.

Two scenarios on which the costs are studied will be presented below: count query on Random topology and some aggregation queries on Grid topology.

Figure 7 shows the results of the count query performed on Random topology, these are represented on a graph that shows the relationship between the number of exchanged messages (y-axis) and the number of hosts in the computation (x-axis). WildFire algorithm requires as a parameter the stable diameter $\bar{D}$ and it operates for a duration approximately equal to $2\bar{D}$t. The Figure provides the result for different values of $\bar{D}$, and from these, it is possible to see that the related curves overlap. This observation leads to the conclusion that the communication costs remain unaffected by the value of D. Also, another relevant observation is that the communication costs of SpanningTree and DirectedAcyclicGraph are 4 times lower than WildFire, finally one can draw the conclusion that to ensure the Single-Site Validity property one has to pay 4 times the communication cost.

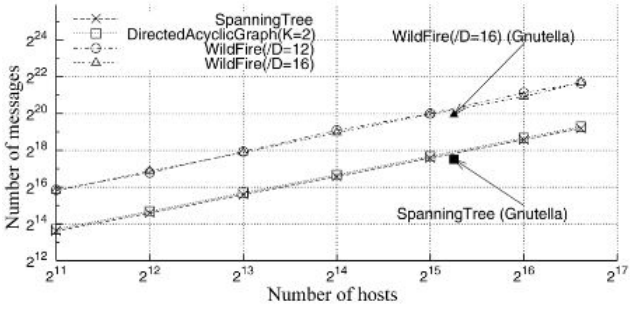

Figure 7. Count query on Random topology

Figure 8 shows the results of the count query performed on Grid topology, these results are shown on a graph with axes the same as Figure 7. The Grid topology used for this experiment is based on the idea that each host in the grid communicates via broadcast protocol with its neighbours. The performed experiments on SpanningTree and DirectedAcyclicGraph are about the Count query, which produces the same communication cost as minimum query or sum query, while on WildFire are performed experiments about the aggregation queries Minimum, Maximum and Count.
The Figure provides the observation that the WildFire for Count query pays 5 times the communication cost paid by the SpanningTree and DirectedAcyclicGraph protocols. Another factor that catches the attention is the cost of the maximum and minimum queries for WildFire, in fact, the maximum query pays the same communication cost as the aggregation queries performed in SpanningTree and DirectedAcyclicGraph, while for the minimum query, the cost is lower than these operations this is due by the early aggregation performed in WildFire. Early aggregation is a technique that enhances the speed of aggregation queries.

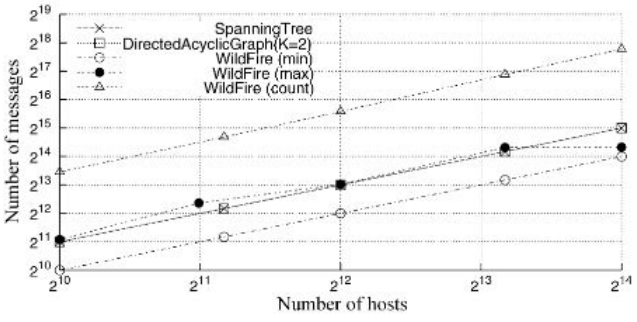

Figure 8. Different queries on Grid topology

Computation cost and Time cost

The study of computation cost consists in finding the maximum number of messages processed by a single host. Figure 9 shows the results of computation costs, performed on a Power-Law topology and on a Grid topology, through a graphic, which shows the relationship between the number of hosts (y-axis) and the computation costs of a single host (x-axis). Study of the two topologies:

- **Power-Law topology** ⇒ WildFire pays a computation cost that is 2 times higher than the cost paid by SpanningTree
- **Grid topology** ⇒ WildFire pays a computation cost that is 44 times higher than the cost paid by SpanningTree

The bad performances of WildFire on Grid topology are due to the fact that while in Spanning Tree protocol each message

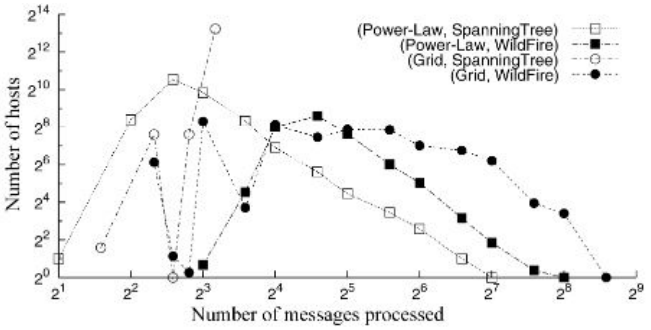

Figure 9. Evaluation of Computation cost

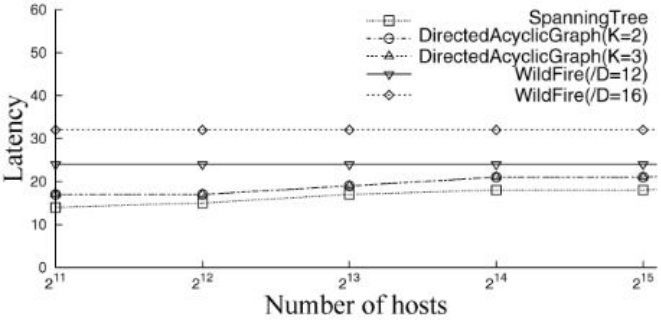

Figure 10. Evaluation of Time cost

is directed to the parents of the current host, in WildFire each message is sent to all the 8 neighbours of the host, so the exchange of messages becomes more expensive. The study of time cost consists of finding the length of the longest chain of messages that occurs before the termination of the protocol. Figure 10 shows a graphic which shows the relationship between the time cost of protocol (y-axis) and the network size (x-axis). According to WildFire's definition, its result is returned $2\hat{D}t$ time after the start time. This time is a constant for a given value of $\hat{D}$ regardless of the number of hosts.

Our Dependability Evaluation

Dependability is the ability of a system to deliver a service that can justifiably be trusted, it is the ability to avoid service failures that are more frequent and more severe than is acceptable.
A service failure is an event that occurs when the delivered service deviates from the correct service. A correct service is delivered when the service implements its functional specifications in terms of

- **functionality**
- **performance**

Our study is focused on performance evaluation because, regarding functionality evaluation, this is guaranteed by the studies and demonstrations done by the authors of the paper.
There are three ways to evaluate the performance of a system:

Measurement through benchmarking, Analytical modelling or Simulation modelling. Measurements are possible only if the system already exists. If it is a new concept, analytical modelling and simulation are the only techniques from which to choose.
A model is an abstraction of a generalized overview of a real system. The level of detail of the model and the specific aspects of the real system that are considered in the model depend on the purpose of the model. The analytical models are a set of formulas and/or computational algorithms they are characterized by a lower level of detail and so they are less accurate but more efficient. Simulation models, instead, are computer programs where all resources and the data flow are simulated. These models have a higher level of detail but they are expensive to run, develop and validate.
In our case, we define a simulation model evaluating the parameters defined by the authors of the paper trying to compare them with theirs. More precisely we'll simulate a continuous system in which the state variables change continuously with respect to time. In fact, we simulate a system composed of hosts linked in specific topologies (Random and Power-Law) and each host can fail in order to simulate a real failure.
The dependability attributes taken in analysis in our work are:

- The **communication cost** of the protocol: obtained by the sum of the number of messages sent between any pair of hosts.
- The **computation cost** of a host: the number of messages that it processes, the computation cost of the protocol is the maximum among the computation cost of the hosts in the network.
- The **time cost** of the protocol: the length of the longest chain of messages that occurs before the termination of the protocol.

Our simulation model is composed of Docker containers linked on the network created by Docker exchanging messages through a message broker in cloud. Externally we can choose the topology to use for the tests and the will to make the containers fail. This will create the structure and the links between containers.

Other dependability properties

In this section, we discuss if the system guarantees or not the main dependability properties:

- **Availability**: readiness of correct service ⇒ The system described in the paper follows this property when we have at least a host on which we can query the system.
- **Reliability**: continuity of correct service ⇒ The system described in the paper follows this property when we have at least a host on which we can query the system.
- **Fault Tolerance**: avoid service failures in the presence of faults. We can divide it into two cases:
  - **Crashes**: a host in the system fails ⇒ Following the definition of Single-Site Validity, the system is tolerant of failures of hosts. The system doesn't recover the host failure and the network but the property cited before guarantees that the WildFire algorithm gives a good result.
  - **Byzantine failures**: a host in the system alters his behaviour ⇒ The system is not tolerant of Byzantine failures because the WIldFire algorithm doesn't provide checks on Byzantine behaviours.

Limitations

In order to simulate the system we are facing some limitations that we have to take into consideration such as the number of hosts in the topology. In fact, the authors of the paper use different topologies composed of thousands of hosts that communicate with each other.
In our development system, replicating the same conditions present in the paper would mean trying to support too many resources for the high number of hosts and messages exchanged. For this reason, we emulate the work in the paper with a system scaled to our availabilities. Furthermore, we choose to exchange messages through hosts using a message broker in cloud. This solution limits our system to a maximum of 20 nodes that will establish a connection to the message broker. Of these 20 connections, one will be used to interface with the host network.
Another limitation, this time regarding the accuracy of the results, that we faced is in the algorithm used by the writers of the paper. The latter has chosen to modify the Flajolet-Martin algorithm using a sequence of coin tosses as a hash function. The Flajolet-Martin algorithm requires a hash function that works properly to ensure that for every c > 2 (where c is the number of vectors used in the algorithm), given a set M of elements drawn from a set V of size n, it outputs an estimate $\widehat{m}$ of the number of distinct elements m in M such that $Pr(\frac{1}{c} \leq \frac{\widehat{m}}{m} \leq c) \geq 1 - \frac{2}{c}$. Due to the fact that we are using a hash function that loses its property of Collision resistance: multiple sequences of coin tosses could lead to the same results and consequently the definition of equal vectors. Furthermore, the estimation is strictly related to the number of trailing zeros in these vectors but this number is highly dependent on the construction of the vectors, and then on the probabilities that occurred in the construction of different vectors with trailing zeros.
From the analysis of the algorithm, we conclude that a good result can be obtained using a high number of vectors and a lower number of hosts. In fact, with the exchange of fewer messages, fewer logical OR will be performed and there will be less probability that many vectors have a 1 as the least

significant bit and therefore have the maximum number of trailing zeros wrong.

Obtained Results

We conducted a series of experiments in a distributed environment to evaluate the performance of different protocols under two network topologies: Power-Law and Random. The protocols under investigation were Spanning Tree and Wildfire. Each experiment was run three times, and the average results were collected for analysis.

Power-Law Topology: This topology follows a power-law distribution, where a few nodes have significantly higher degrees compared to the majority of nodes. In this topology, the network exhibits a scale-free characteristic, where a small number of highly connected nodes (hubs) play a crucial role in information dissemination.

Random Topology: This topology is generated by randomly connecting the nodes in the network. It lacks the characteristic structure of power-law distribution and offers a more homogeneous distribution of connections among nodes. For each combination of topology and protocol, we conducted two experiments to simulate different scenarios:

1. No-Failure Case: In this experiment, we assumed that all nodes in the network were operational, and no failures occurred during the simulation. This scenario allowed us to assess the protocol's performance under normal operating conditions.
2. Failure Case: In this experiment, we introduced a failure scenario by simulating node failures in the network. Each node had a probability of 1/7 to fail, meaning that approximately one-seventh of the nodes experienced failure during the simulation. This scenario allowed us to evaluate the robustness and resilience of the protocols in the face of failures.

By running these experiments and collecting the average results, we were able to analyze the performance of the protocols under different conditions. The collected data provided insights into the protocols' efficiency, effectiveness, and ability to handle failures in the network.

The obtained results from the experiments conducted in the distributed environment provide valuable insights into the performance of the Spanning Tree and Wildfire protocols under different network topologies (Power-Law and Random). The analysis primarily focused on the following performance metrics:

- Total Number of Processed Messages: This metric quantifies the overall communication load in the system. By comparing the total number of processed messages for each protocol and topology, we can assess the efficiency and effectiveness of the protocols in transmitting and disseminating information within the network ⇒ **Communication cost**
- Highest Number of Messages Processed by a Node: This metric identifies nodes that experience a higher message load compared to others, highlighting potential bottlenecks in the system ⇒ **Computation cost**
- Computation Time: This metric measures the time taken by the protocols to complete their operations. By comparing the computation times of the Spanning Tree and Wildfire protocols in different topologies, we can evaluate their performance in terms of speed and efficiency ⇒ **Time cost**

Analyzing these performance metrics for each protocol-topology combination allows us to draw conclusions about the features of each protocol under different network scenarios.

In the first type of experiment conducted, we aimed to evaluate the system's performance across all possible topology-protocol combinations while varying the dimension of the exchanged messages. This allowed us to assess the impact of message size on the performance metrics mentioned earlier.

In our experiments, we performed evaluations by adjusting the number of vectors (c) in each node. We began with an initial configuration where each node contained 8 vectors, and then gradually increased it to 12 vectors. Throughout the experiments, the number of nodes in the network remained constant and equal to 20.

Based on the obtained results from the evaluation of the system's performance metrics, we have the following graphics showcasing the outcomes for each metric:

- Total Number of Processed Messages - Figure 11
- Total Number of Processed Messages (Failure Case) - Figure 12
- Total Highest Number of Messages Processed by a Node - Figure 13
- Total Highest Number of Messages Processed by a Node (Failure Case) - Figure 14
- Computation Time - Figure 15
- Computation Time (Failure Case) - Figure 16

The first four graphics (11,12,13,14) mentioned depict the relationship between the chosen strategy (x-axis) and the number of messages (y-axis). Each strategy is associated with four points on the graph due to the consideration of two factors: the value of messages when the number of vectors (c) is 8 or 12, and the number of messages when the query is initiated from the node with the highest number of neighbours and the node with the lowest number of neighbours. This results in a total of four combinations per strategy.

The remaining graphics (15,16) focus on illustrating how the variation in the number of vectors affects the timing of the system. In these graphics, the curves represent the four possible strategies, and each data point is computed as the average time taken when the query is initiated from both

the node with the highest number of neighbours and the node with the lowest number of neighbours. The x-axis in these graphics represents the number of vectors (c), while the y-axis represents the computation time. By analyzing the curves for each strategy, we can observe how the timing changes with different numbers of vectors in the system. The computation time is determined by averaging the time taken for the query to be processed in both scenarios: when it is initiated from the node with the highest number of neighbours and when it is initiated from the node with the lowest number of neighbours.

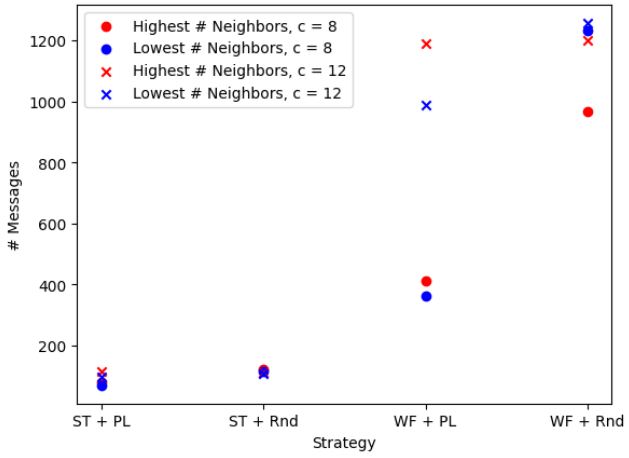

Figure 11. Total number of messages without failures

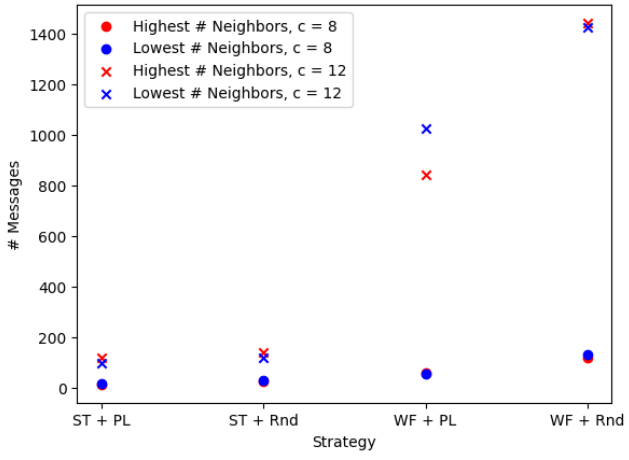

Figure 12. Total number of messages with failures

As evident from the graphics, our experiments on the variation of ”c” confirm the claims made in the original paper. The introduction of the WildFire algorithm leads to a decrease in performance in terms of both the quantity of messages and the execution time. Additionally, the graphics reveal some important observations that contribute to the deterioration of performance with WildFire.

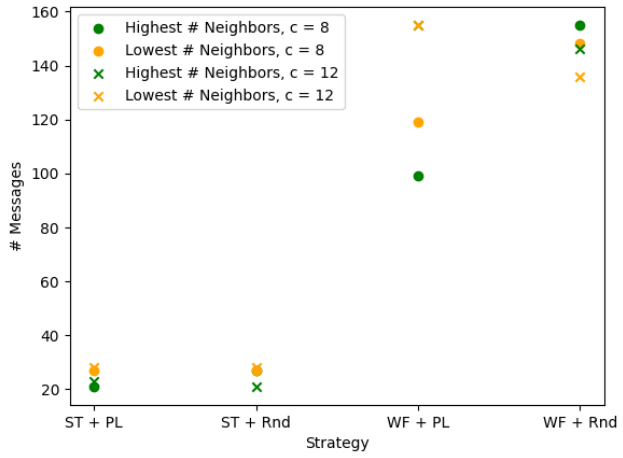

Figure 13. Maximum number of messages per node without failures

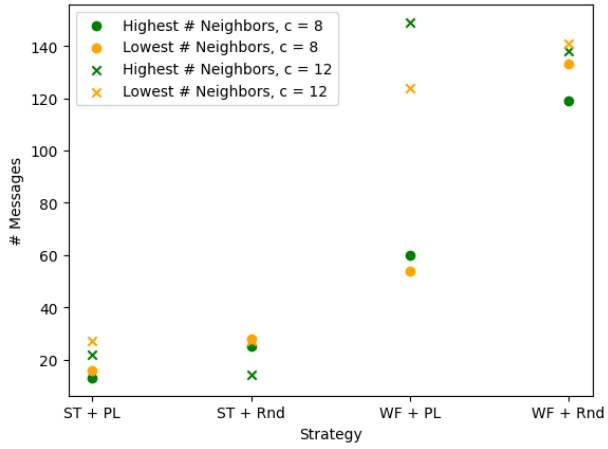

Figure 14. Maximum number of messages per node with failures

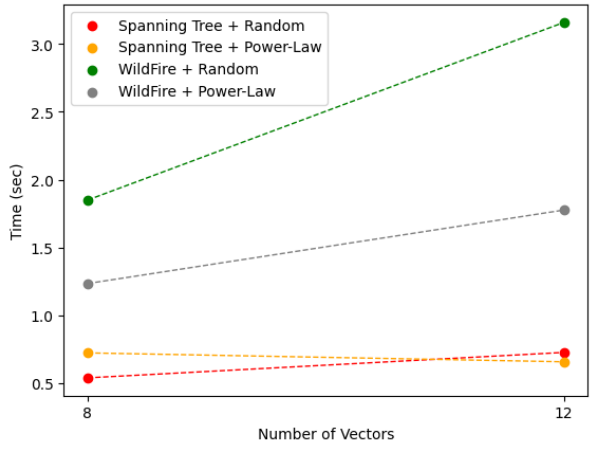

Figure 15. Computation Time without Failures

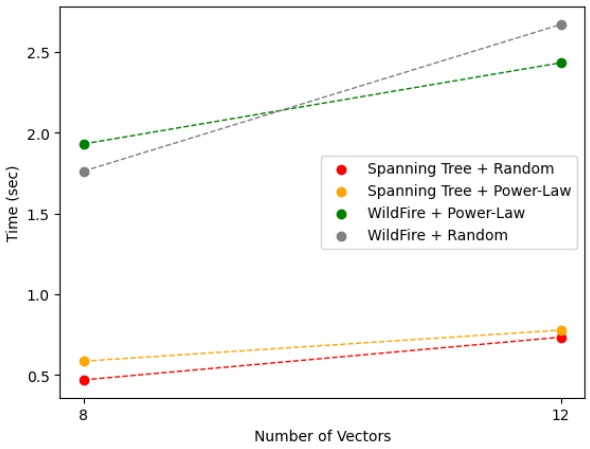

Figure 16. Computation Time with Failures

For instance, when WildFire is employed, the performance in terms of the total number of messages (Figures 11 and 12) and the highest number of processed messages (figures 13 and 14) tends to be higher in most cases when the value of "c" is higher.

These findings highlight the impact of WildFire on the performance metrics, indicating that increasing the value of "c" may exacerbate the performance degradation observed in terms of the total number of messages and the highest number of processed messages.

In addition to the experiments conducted on the variation of the number of vectors, we also performed experiments to evaluate the impact of the number of nodes in the network on the system's metrics. In these experiments, we kept the number of vectors (c) equal to 8 while varying the number of nodes in the network.

Specifically, we conducted experiments for three different configurations: 19 nodes, 15 nodes, and 10 nodes. By altering the number of nodes while keeping the number of vectors constant, we aimed to investigate how the network size affects the performance metrics of the system.

Based on the obtained results from the evaluation of the system's performance metrics, we have the following graphics showcasing the outcomes for each metric:

- Total Number of Processed Messages - Figure 17
- Total Number of Processed Messages (Failure Case) - Figure 18
- Total Highest Number of Messages Processed by a Node - Figure 19
- Total Highest Number of Messages Processed by a Node (Failure Case) - Figure 20
- Computation Time - Figure 21
- Computation Time (Failure Case) - Figure 22

The first four graphics (17,18,19,20) illustrate the relationship between the number of nodes (x-axis) and the number of messages (y-axis). The first two graphics represent the relationship between the number of nodes and the total number of messages processed during the computation, both in cases with and without failures. The remaining two graphics represent the relationship between the number of nodes and the highest number of messages processed by a single node in the network. The remaining two graphics (21,22) depict the relationship between the number of nodes in the network (x-axis) and the execution time (y-axis). These graphics model how the execution time varies as the number of nodes in the network changes. Based on the graphics obtained from

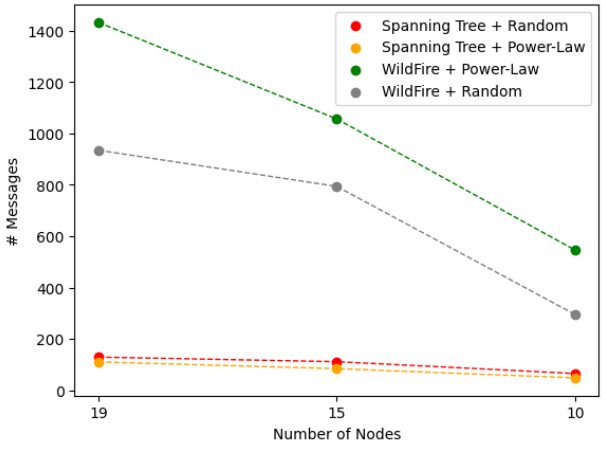

Figure 17. Total number of messages without failures

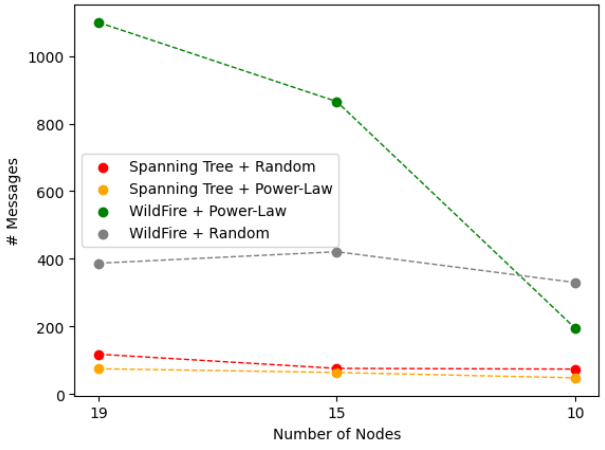

Figure 18. Total number of messages with failures

the experiments conducted in this study case, we can indeed confirm that the use of WildFire leads to an increase in the number of messages, as well as the computation time, as desired. These observations align with the intended outcome of using WildFire, which aims to increase the number of messages and extend the computation time.

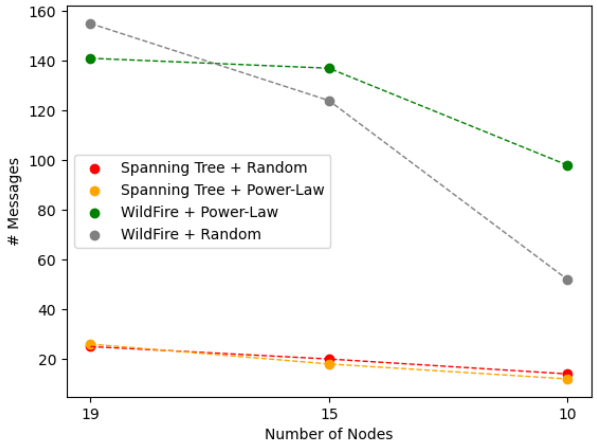

Figure 19. Maximum number of messages per node without failures

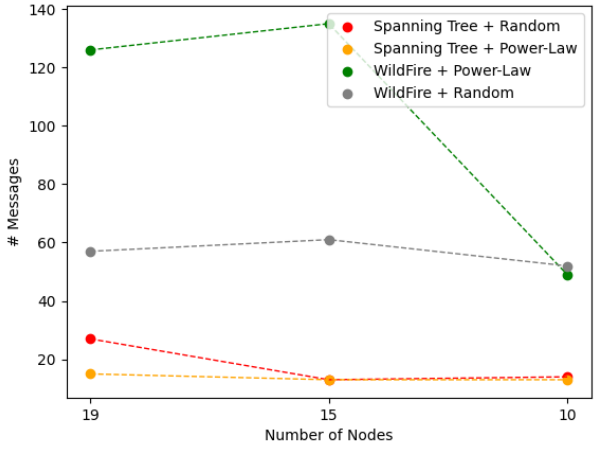

Figure 20. Maximum number of messages per node with failures

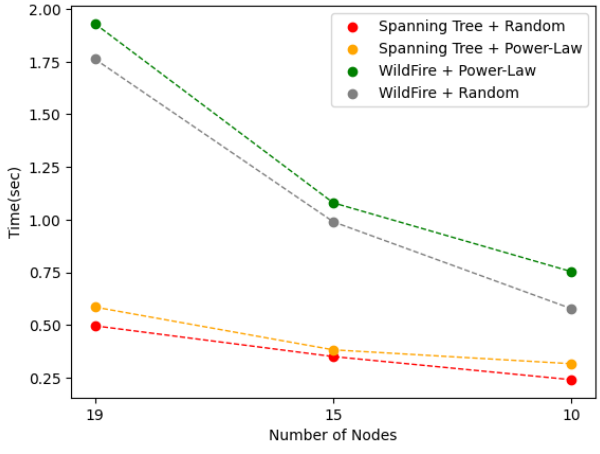

Figure 21. Computation Time without Failures

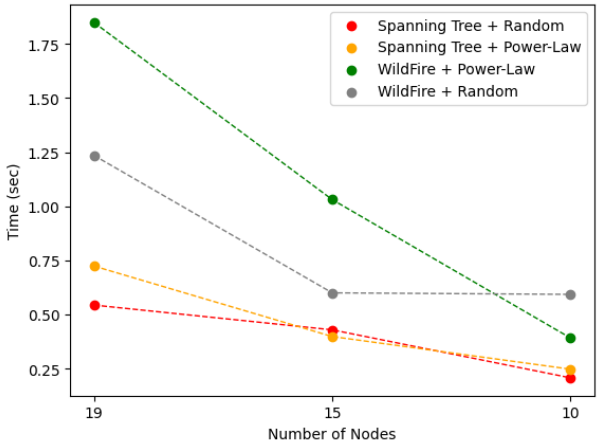

Figure 22. Computation Time with Failures

Conclusions

In conclusion, this essay has presented our findings and analysis from conducting experiments on the effects of the WildFire algorithm in terms of the number of messages and computation time. Through the graphics and results obtained, we have observed a clear increase in both metrics when WildFire is utilized, showing what the original paper states.

It is possible to access the source code, data, and additional materials related to our experiments, through the repository https://github.com/fed21/the-price-of-validity.