DOI: 10.18372/2310-5461.66.20311
УДК 621.396:004.75:004.78

I. Syvolovskyi, Post-Graduate Student
Ukrainian State University of Railway Transport
https://orcid.org/0000-0002-4592-0965
e-mail: ilyasvl95@gmail.com

V. Lysechko, Dr Sc., Professor
Scientific Center of the Air Force Ivan Kozhedub Kharkov
National University of Air Forces
https://orcid.org/0000-0002-1520-9515
e-mail: lysechkov@ukr.net

METHOD FOR LEADER NODE SELECTION AND PROCESSING PIPELINE FORMATION IN DISTRIBUTED TELECOMMUNICATION SYSTEMS

Introduction

Modern distributed telecommunication systems are constantly facing increasing demands for real-time processing of data streams. Such systems are widely used to ensure stable and fast information transfer between numerous nodes distributed over long distances.

With frequent changes in traffic intensity, packet loss, and limited resources, there is a need to implement methods that could ensure reliable and efficient coordination of computing processes. In real-world conditions, both the choice of a master node (cluster coordinator) responsible for operational resource management and the rapid formation of computing pipelines that minimize data transmission latency have become important aspects.

The existing solutions, such as the Bully Algorithm and its modifications, although providing basic stability of the systems, have significant drawbacks. In particular, they provoke periodic spikes in network traffic (election «storms») and do not respond effectively enough to sudden node failures. This negatively affects the response time and overall performance of the telecommunications system.

Therefore, there is a need to create an adaptive method that would allow for a quick response to changes in the state of nodes without explicit elections and ensure optimal use of available resources, especially in conditions of unstable communication and unpredictable loads.

Analysis of recent research and publications

Studies [1–4] have substantiated the feasibility of using consensus algorithms to ensure consistency in distributed computing environments, but these works mainly focus on formal models and classical protocols, without taking into account the specifics of clustered DTSs with latency and resource constraints, and the need for local response to load changes.

Various approaches to leader election are considered in [5–12]: from classical schemes in ring topologies to self-organizing and probabilistic mechanisms for anonymous or partially synchronous networks. However, they require global coordination, do not provide mechanisms for fast local recovery after a node failure, or have significant overhead in the election process.

Study [13] proposed an improved version of the ring election algorithm with a reduced number of messages, but it remains tied to a fixed topology and does not take into account adaptive mechanisms for distributed platforms with pipeline processing.

Works [6, 10, 14] focus on optimizing certain characteristics, such as energy consumption, predictability, or combining algorithms (e.g., Raft with PoW [14]), but they do not provide an effective solution to the problem of selecting a leader in conditions of limited computing resources, with delays and instability of connections typical to DTS.

Thus, the problem of developing a deterministic, scalable, and resource-efficient method for selecting a master node capable of adaptively functioning within clustered DTSs remains unresolved.

Problem Statement

Modern DTSs with step-by-step pipeline data processing operate in a dynamic environment characterized by variable traffic intensity, spontaneous load spikes, and periodic instability of network connections. This requires high adaptability in managing computing resources, guaranteed continuous data processing, and efficient routing of data streams between nodes.

Particularly relevant is the selection of the cluster's master node (coordinator), which performs local management functions and is responsible for collecting metrics, monitoring resources, and coordinating data streams.

In practice, classical leader selection algorithms show limited efficiency due to excessive network traffic in the election phase, sensitivity to message loss, delays in control restructuring after a coordinator failure, and the lack of mechanisms for prioritizing the delegation of coordinating functions to less productive nodes [1–16].

Based on this, the scientific and practical task of this study is to develop a method for selecting a master node that will be adapted to the functional and architectural features of DTS with a cluster structure and pipeline data processing and will ensure efficient use of computing resources, minimize delays in the transmission of management messages, and support a hierarchical structure with a distinction between intercluster and intracluster management.

The implementation of the proposed method requires the creation of a specialized algorithm that takes into account both the topological (delay between nodes) and resource (computing performance) characterristics of nodes, as well as mechanisms for instantly determining a new leader in case of failure of the current coordinator.

The purpose of the article

The aim of the study is to develop a method for selecting the master node (coordinator) in a DTS with a cluster architecture, using a multi-criteria evaluation of nodes and an improved Gossip leader selection algorithm in an unstable environment.

Summary of the main material

To implement the scientific and practical tasks of this study, a distributed processing system is considered, the main purpose of which is to provide continuous streaming data processing in a step-by-step mode [1, 2, 4]. The initial data is generated by end devices (sensors), and the processing results must be delivered to a cloud environment or data center for further storage or analysis.

Each stage of data processing in the system has a unique number, and the stages are performed in a strictly defined sequence that forms a processing pipeline. Such a pipeline is an ordered set of tasks (for example, a sequence of four processing operations) that are implemented in stages.

The architecture and functional features of the model of a distributed telecommunications system are determined through a set of provisions that reveal the logic of interaction of its components.

1. Cluster structure. The system consists of clusters of processing nodes. Nodes within the same cluster are located close to each other, although they are not necessarily located in the same network or geographic area. Each node in the cluster is capable of performing only one specific task from a particular processing pipeline [9, 13].

2. Node resource model. Each node is characterized by limited computing resources (processor, RAM, storage) and has established communication channels (tunnels) with other nodes in its cluster. It is assumed that there are no isolated or inaccessible nodes within a cluster [1, 2].

3. Data stream model. Data is transmitted through tunnels, which are defined as data streams. The stream is characterized by a steady intensity, which at certain moments can spontaneously increase, forming short-term bursts. At the same time, processing data from the stream consumes part of the node's resources, and these costs can also reach peak values synchronously with the increase in the intensity of the stream [4].

4. The role of the master node. Each cluster defines a master node that acts as a coordinator: it collects metrics from other nodes and manages the cluster. This node does not participate in the transmission of data streams, but it has full information about the structure of the streams of the nodes in its cluster.

5. Supervisor in the cloud. In the cloud part of the system, there is a supervisor node that acts as the final coordinator for inter-cluster decisions, although local, intra-cluster management is preferred [2, 3].

Considering the specifics of the architecture and functional purpose of the model, a method aimed at solving two scientific and practical tasks is proposed:

– selection of the master node (coordinator) within each cluster with the provision of potential backup options, with priority given to nodes with lower resources and lower average latency to other nodes;

– construction of processing pipelines with minimization of inter-cluster delays, provided that the sequence of tasks is maintained and the resource constraints of each cluster are met.

The system architecture presented in Fig. 1 shows the main structural and functional features that were considered during the development of the method. Based on this, additional assumptions are made in this study:

1. The clusters are small (5–60 nodes), which is ensured by preliminary grouping based on the method of hierarchical clustering of processing nodes in distributed telecommunication systems, using a modified Louvain algorithm that performs multi-step clustering of the graph with dynamic adjustment of parameters [15].

2. Low latency is ensured within a single cluster, but partial packet loss is possible due to connection instability or nodes being located in different WAN segments.

3. The key requirement is to minimize the time to restore cluster control in the case of a master node failure.

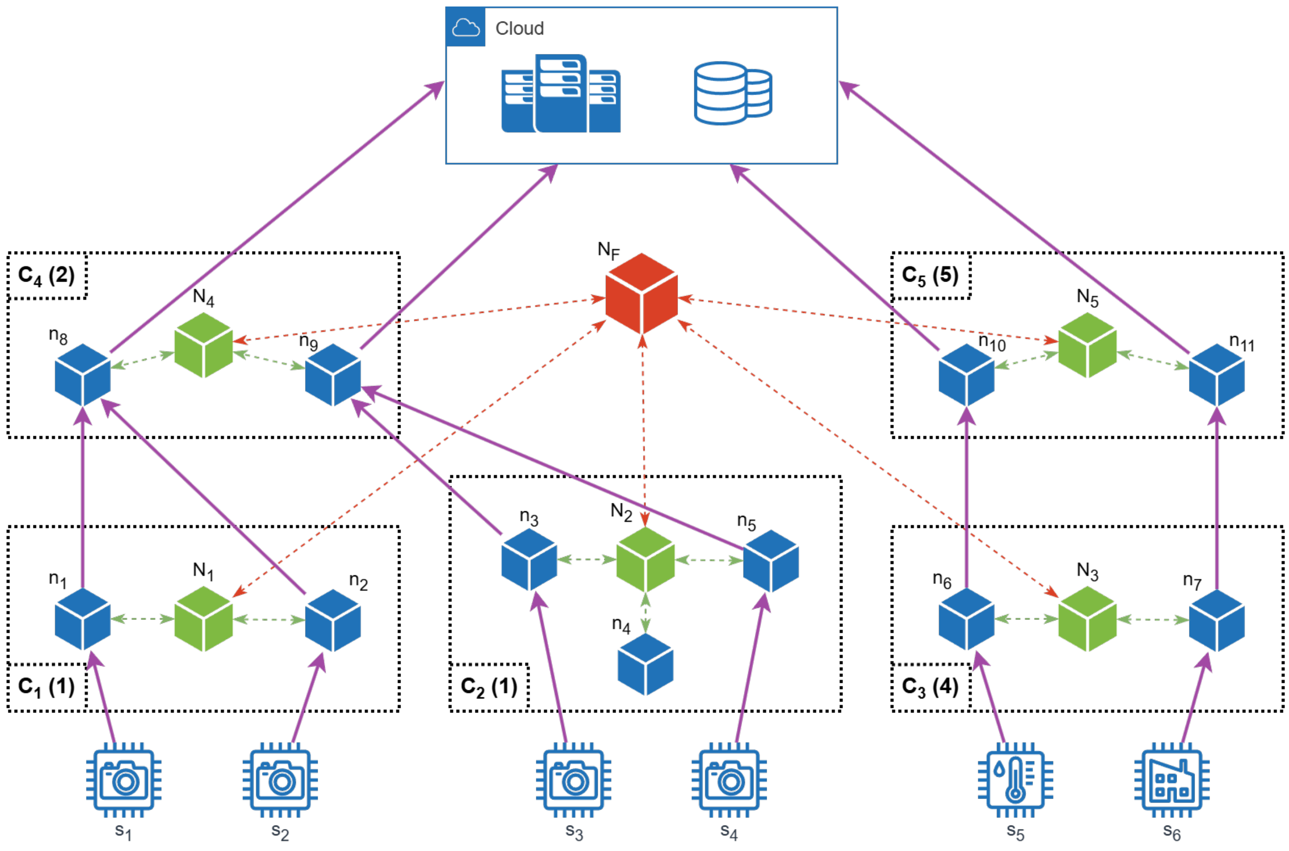


Fig. 1. DTS architecture with node clustering and centralized supervisor

The designation of the elements shown in Fig. 1:

Blue squares with an icon are sensors $S_1 - S_6$. They initiate data streams (continuously generate data).

Small dotted lines are the boundaries of clusters of processing nodes. The designation of the cluster indicates its number and, in parentheses, the number of the pipeline task it can perform.

Blue cubes are processing nodes that perform certain stages of the pipeline.

Green cubes are local master nodes (cluster coordinators $\mathrm{C}_1 - \mathrm{C}_5$).

The red cube N_F is the central supervisor responsible for inter-cluster coordination.

Purple solid arrows – direction of data streams (pipelines) from sensors $S_1 - S_6$.

Green dotted arrows – connections between processing nodes and the master node within the cluster.

Red dashed arrows – connections between the master nodes of the clusters and the central supervisor.

The assumptions formulated in the study outline the limitations and conditions of functioning within which the proposed method is implemented. In this context, the first of the defined scientific and practical research objectives is detailed as follows.

1. Determining the criteria for selecting a master node from a set of candidates.

2. Development of an algorithm for master node election.

Furthermore, when developing a method, it is necessary to take into account possible variants of cluster operation (functioning) that significantly affect its effectiveness, namely.

1. The method for selecting the master node should ensure stable operation both in the absence of the coordinator (for example, during initialization or after a failure) and in its presence (to maintain balance when new nodes are added).

2. It is necessary to define a «substitute» for the master node to ensure a quick and uninterrupted transfer of leadership in the case of a master node failure.

To determine the priority of the node in the process of selecting the master node, this study proposes an indicator - the Scoring formula - that allows ranking candidates for the role of cluster coordinator. This function is developed based on three main criteria that take into account the topological and resource characteristics of nodes and allow deterministic solutions to situations with the same metrics.

Each criterion in the proposed formula is normalized to a unified scale and weighted by the corresponding significance coefficient w_i, depending on the characteristics of the distributed architecture.

The general view of the evaluation function of node i is described by the formula:

$$Score(i) = w_1 \cdot AvgLat(i) + \\ + w_2 \cdot NormPower(i) + \\ + w_3 \cdot NormID(i). \quad (1)$$

Where:

$$w_1, w_2, w_3 \in [0,1], w_1 + w_2 + w_3 = 1.$$

1. Average delay to other nodes. This criterion reflects the node's topological proximity to other cluster members, which is important for reducing coordination delays. It is calculated by the formula:

$$AvgLat(i) = \frac{1}{|N| - 1} \sum_{j \neq i} Latency(i, j),$$

where $|N|$ – number of nodes in the cluster.

2. Normalized resource capacity of a node. Estimates the total computing capacity of a node based on the total amount of primary resources:

$$Power(i) = CPU_i + RAM_i + STG_i,$$

The estimated power value of a node is normalized relative to the maximum power among all nodes in the cluster:

$$NormPower(i) = \frac{Power(i)}{max_k Power(k)},$$

In the proposed method of selecting the master node, preference is given to weaker nodes, i.e., those with lower computing power. This allows to keep powerful nodes for intensive computations, and to transfer coordinating functions to less productive participants.

3. Normalized value of the node identifier. This is an additional criterion that is used to resolve situations with equal scores, as an additional mechanism for resolving a «tie»:

$$NormID(i) = \frac{ID_i}{max_k ID_k},$$

The smaller the $Score(i)$ value (1), the higher the priority of the node for being elected as the master. The proposed evaluation function allows for the preliminary ranking of all candidates in the cluster without the need to launch a full-scale election procedure. However, to ensure the stable functioning of the system in the face of dynamic changes, it is necessary, in addition to preliminary ranking, to apply an effective consensus algorithm capable of quickly and without excessive delays determining a new master node in the case of a failure or in the case of changes in the topology.

Table 1 shows a comparative characteristic of the leader election algorithms that can be used to select the master node [1, 3, 7, 13]

Table 1

Comparative evaluation of leader election algorithms

Algorithm	The principle of operation	Advantages	Disadvantages
Bully algorithm	In case of failure of the master node, the node with the highest ID becomes the leader. Initialization occurs by sending «Elections» messages to all active nodes.	Simple, fast on a stable network.	High traffic with frequent use. Sensitive to message loss.
Fast Bully algorithm	Enhanced algorithm with fewer messages during elections, but with nodes caching more information.	Faster due to reduced messages; optimal for small clusters.	Requires a guarantee of the correct order of message transmission.
Raft (only at the leader election stage)	The nodes «start» voting with random timers, and the one who is the first to get the majority of votes wins.	Very resilient even to the loss of messages and partitions.	More complicated than the Bully algorithm and redundant for selecting only the master node.
Viewstamped Replication (VSR) election	A lightweight alternative to Raft: reconciliation of a new leader after changing the view.	Great for clusters with high availability requirements.	More difficult to implement; requires constant activity of all replicas.
Ring-based election (Chang and Roberts)	The nodes located in the logical ring transmit the message «Elections» in a circle. The node with the highest score wins.	Simple, guaranteed elections.	Slow in large clusters, not suitable for latency optimization.

The end of the table 1

Algorithm	The principle of operation	Advantages	Disadvantages
Gossip with deterministic selection *(*author's enhancement)*	The nodes exchange metrics and then independently and deterministically choose the best node based on common criteria.	Light traffic, no election storm, fast recovery thanks to predefined substitutions.	Requires efficient initial metric distribution. Does not handle long partitions on its own.

The analysis of the data in Table 1 allows us to conclude that, given the stated research goal of ensuring the effective selection of the master node and the stable functioning of clusters in distributed telecommunications systems, it is expedient to use the following algorithms based on their advantages.

1. Fast Bully algorithm:

– effective for small and medium-sized clusters with relatively stable connection conditions;

– provides fast convergence when using node identifiers;

– makes it easy to implement the mechanism of substitute nodes, which increases reliability in case of failures;

– performs better than the classic bully algorithm in the conditions of temporary network issues;

The disadvantage of this algorithm is that it requires the identifiers/scores of other nodes for continuous operation, which can lead to broadcast congestion («election storm») during the initialization phase.

2. Gossip algorithm with deterministic selection. The advantages of the algorithm include:

– the election is not initiated at all. If a master node fails, the other nodes or a substitute detect this and determine a new master node in a deterministic manner;

– constant but very light traffic;

– nodes know the full list of substitutes. Meets the need for instantaneous selection of substitutes;

– fast recovery after a failure;

– works in clusters whose topology is not a complete graph or in the presence of cycles. This allows the maximum cluster size to be significantly increased and the algorithm is able to handle more practical cases.

Given the advantages of a deterministic approach to selecting a master node without running conventional election procedures, this research has enhanced the Gossip algorithm with deterministic selection, with modifications in accordance to the architectural requirements and features of the operation of a distributed telecommunications system.

Let's take a more detailed look at the main characteristics of the Gossip algorithm with deterministic selection, the stages of its implementation and enhanced.

Gossip is a class of protocols (algorithms) based on the gradual spread of information between neighboring nodes, similar to an epidemic. Gossip is widely used in decentralized networks where there is no permanent coordinator, and it is an effective way to exchange data without centralized control [4, 9].

In the study, based on the Gossip approach, an enhanced algorithm for selecting the master node is proposed, which avoids broadcast messaging and implements the selection of the coordinator without an explicit election phase. The main principles of the algorithm implementation are:

1. Periodic exchange of basic metrics or pre-calculated scores between nodes.

2. Independent calculation by each node of the optimal candidate for the role of master node based on the collected metrics.

3. Automatic selection of a predefined substitute in case of failure of the current leader, which ensures almost instantaneous restoration of cluster management.

4. Limiting the size of messages to the minimum required parameters ({ID, score or metrics, timestamp}).

5. Exchange of heartbeat messages between nodes after determining the master node to confirm the status.

The main designations of the algorithm's indicators (metrics) are presented in Table 2.

Table 2

Designations (metrics) of the enhanced Gossip algorithm

Symbol	Value
ID_n	Unique identifier of node n. Defined during initialization and does not change
M_n	Metrics of node n: delay, total number of resources (or their utilization)
$Score(n)$	The score of a node n is calculated based on its metrics (formula 1)
C_n	A local list of known cluster nodes with up-to-date metrics obtained through the Gossip spreading mechanism

The end of the table 2

Symbol	Value
R_n	A ranked list of nodes based on the values of $Score(k)$, ordered in ascending order
$Master_n$	The node that is considered to be the master for node n
$Substitutes_n$	A list of alternatives (substitutes) of node n that can replace the coordinator in case of its failure

The flowchart of the enhanced Gossip algorithm during normal operation and when a master node failure is detected is shown in Fig. 2.

According to Fig. 2, the procedure of the algorithm in normal operation includes the sequence of the following actions.

Steps of the enhanced Gossip algorithm in normal operation

1. Each node periodically transmits («gossips») its current metrics, for example, at intervals of 2–5 seconds.

2. The node transmits the metrics received from its neighbors to other nodes (all or a certain number of them).

3. The node locally creates a set C_n (a set of current metrics of known nodes) and independently

– calculates $Score(n)$ for all nodes in C_n;

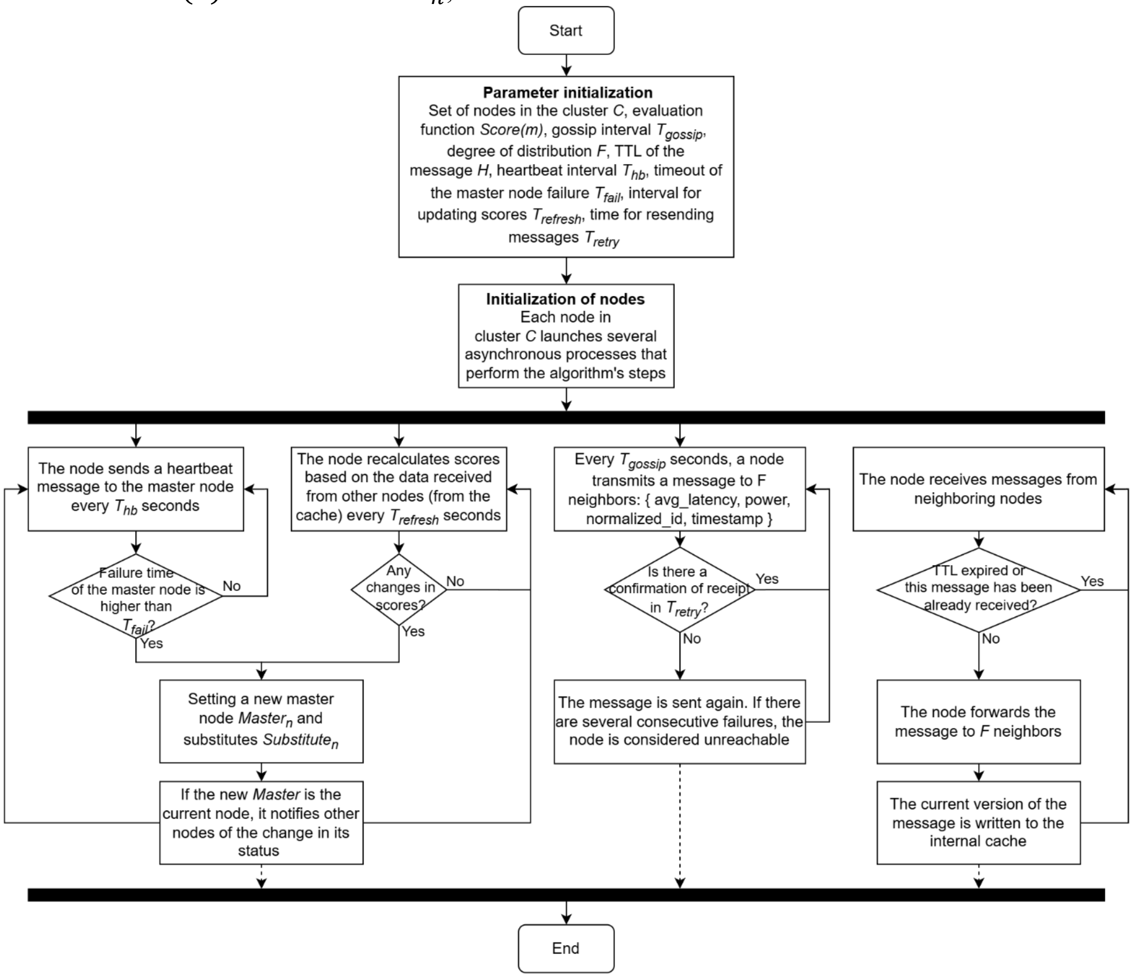
– sorts the nodes in R_n (from best to worst);

– sets the master node $Master_n = R_n[0]$ and substitutes $Substitutes_n = R_n[1:]$ (the rest of the list).

Thus, each node locally has information about the current master (coordinator), and in case of its failure, it instantly appoints a new leader from a pre-calculated list. If necessary, metrics can be replaced with scores, but this reduces the flexibility and versatility of the algorithm.

Steps of the enhanced Gossip algorithm when detecting a master node failure

In case of detecting the absence of coordinator activity (disappearance of periodic messages of the «heartbeat» type), in accordance with Fig. 2, the following sequence of actions is performed:



Start
Parameter initialization
Set of nodes in the cluster C, evaluation function $Score(m)$, gossip interval T_{gossip}, degree of distribution F, TTL of the message H, heartbeat interval T_{hb}, timeout of the master node failure T_{fail}, interval for updating scores $T_{refresh}$, time for resending messages T_{retry}
Initialization of nodes
Each node in cluster C launches several asynchronous processes that perform the algorithm's steps
The node sends a heartbeat message to the master node every T_{hb} seconds
Failure time of the master node is higher than T_{fail}?
No
Yes
The node recalculates scores based on the data received from other nodes (from the cache) every $T_{refresh}$ seconds
Any changes in scores?
No
Yes
Setting a new master node $Master_n$ and substitutes $Substitute_n$
If the new Master is the current node, it notifies other nodes of the change in its status
Every T_{gossip} seconds, a node transmits a message to F neighbors: { avg_latency, power, normalized_id, timestamp }
Is there a confirmation of receipt in T_{retry}?
Yes
No
The message is sent again. If there are several consecutive failures, the node is considered unreachable
The node receives messages from neighboring nodes
TTL expired or this message has been already received?
Yes
No
The node forwards the message to F neighbors
The current version of the message is written to the internal cache
End

Fig. 2. Flowchart of the enhanced Gossip algorithm

1. The node with the highest priority in the $Substitutes_n$list immediately starts to perform the functions of the master node $Master_n$.

2. (Optional) In case of critical requirements to the reaction time, the newly appointed coordinator can explicitly notify other nodes about the change of its status.

3. All nodes in the cluster:

– update the master node: $Master_n = Substitute$;

– shift the list of substitutes accordingly.

Thus, the change of the master node occurs without explicitly launching the election procedure, with minimal delays and without generating excessive message exchange. The state of the coordinator can be monitored either by all nodes in the cluster or only by its substitute candidates.

Similarly, when new nodes are dynamically added or when some nodes are lost, re-electing the coordinator does not require explicitly starting the election procedure. A new leader and an updated list of deputies are generated automatically based on the messages spread within the cluster (gossip metrics).However, to ensure the stable operation of the algorithm in a dynamic topology, it is important to take into account the potential risks of excessive message propagation and data consistency. To this end, the proposed algorithm implements special control mechanisms that ensure efficient cycle processing and avoid duplication.

One of the possible problems with Gossip algorithms is their operation in networks with internal loops. Without a mechanism to interrupt the cyclic transmission, a message can circulate in the system an unlimited number of times. In addition, there is support for the consistency of data circulating in the system (anti-entropy). The following techniques can be used to solve these problems:

1. Use of versions or timestamps. Each node keeps a cache of the last messages (origin, version, or time) that it has already received. If it receives a duplicate, it discards the message. This ensures that each update is propagated only once to each node.

2. Limiting the depth of propagation through TTL (Time-To-Live). Each «gossip» message contains a TTL (for example, 6–12 hops). Each node that forwards the message reduces the TTL by 1. If the TTL reaches 0, it deletes the message. Useful if it is necessary to limit the scope of action and avoid widespread distribution.

3. Anti-entropy checksums. Every few rounds (e.g., every 10th gossip), nodes send a state (a hash or version map) of what they have seen. Other nodes:

a. Compare their state with the received state.

b. Send only the updates that are missing.

This avoids unnecessary data exchange and synchronizes nodes that have fallen behind.

4. The «Push–Pull» gossip transmission protocol. Provides a two-phase model, instead of just transmitting messages:

a. Push: sending the latest updates to other nodes.

b. Fetching: asking a neighbor if it has any missing information.

The more techniques used in the system, the higher its stability and reliability, but this increases its complexity, resource consumption, and network load. The proposed algorithm uses techniques 1 and 2. For systems with increased requirements for accuracy and speed, techniques 3 and 4 or their combination are effective.

In Fig. 3, according to technique 1, an example of message propagation with metrics across the cluster is presented, followed by canceling the message transmission if it has already been received by a node.

White circles – the message has not yet been received by the node; green circles – the message has already been received by the node; green arrows – the direction of message distribution at the current stage; red crosses – the message is not distributed further because it has already been received; black arrows – transition between stages.

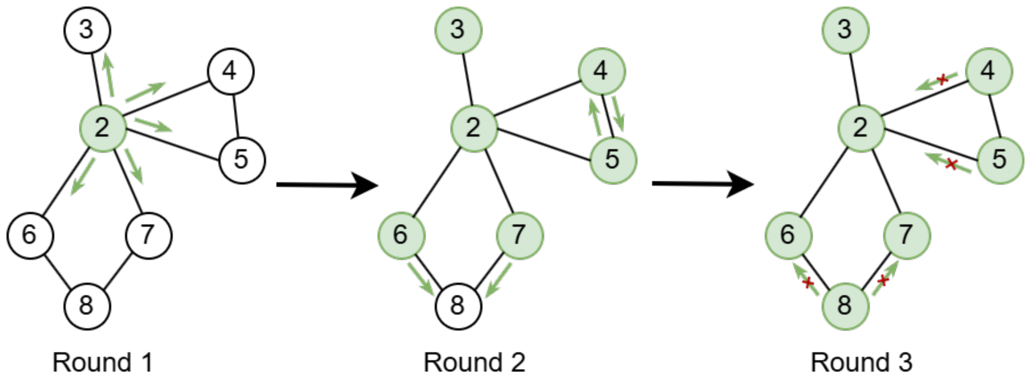


Fig. 3. An example of distributing a message with metrics across a cluster

To ensure the efficient functioning of the algorithm, it is necessary to correctly configure the parameters, examples and recommended value ranges of which are shown in Table 3.

Table 3

The main parameters of the algorithm and their recommended values

Parameter	Symbol	Example value	Purpose
Gossip interval	T_{gossip}	1,0–5,0 s	How often a node reports its metrics/status
Gossip fanout	F	2-6 nodes or «all neighbors» per round	How many nodes communicate (gossip) with the node
Message TTL	H	6-12 hops	Maximum propagation depth (if TTL is used)
Convergence time	T_{conv}	$Depth_{max} \times T_{gossip}$	The time it takes for data to reach all nodes
Heartbeat interval	T_{hb}	0,5–1,0 s	Leader/substitute heartbeat interval
Master failure timeout	T_{fail}	$2 \times T_{hb}$ = 1,5–2,0 s	Timeout before the master node is considered to be down
Score refresh interval	$T_{refresh}$	10–30 s	Redefine candidates based on the latest messages
Message resending interval	T_{retry}	100–1000 ms	Delay before re-distributing the message if there is no confirmation/distribution

For the experimental modeling, a set of conditions was defined that correspond to typical scenarios of distributed telecommunication systems. The relevant test data are summarized in Table 4.

Table 4

Input parameters for experimental modeling

Parameter	Assumed value
Cluster size	5–100 nodes
Latency between nodes	10–50 ms
Message loss rate	Low (≤ 1 %)
Goal of application	Regular synchronization of metrics and quick selection of a leader
Transport	TCP or UDP

Based on the parameters from Table 4, an example of modeling for a cluster of 50 nodes was calculated under the following conditions:

1. Gossip spreading interval $T_{gossip} = 1{,}0$ c.

2. Number of nodes receiving messages in one round $F = 3$.

The propagation of messages within the algorithm is exponential:

$$Nodes\ reached\ in\ R\ rounds\ \approx\ F^R$$

Accordingly:

– After round 1: 3 nodes.

– After round 2: 9 nodes.

– After round 3: 27 nodes.

– After round 4: 81 nodes (the entire cluster).

Conclusion: full convergence (information dissemination) in a cluster of 50 nodes takes 4–6 seconds, which ensures quick state coordination and prompt selection of a new coordinator in case of a master node failure.

Considering the given time characteristics and robustness requirements, it is also important to assess the overall effectiveness of the proposed method. Table 5 presents a comparative analysis of the enhanced Gossip algorithm with one of the most commonly used in practice, the Fast Bully algorithm that uses substitutes.

Table 5

Comparison of the proposed algorithm with the Fast Bully algorithm

Aspect	Fast Bully algorithm with substitutes	Enhanced Gossip with deterministic selection
Leader election trigger	Explicit identification of failures and then active elections	No election; nodes calculate the master node independently
Recovery from a failure	The pre-selected substitute becomes the leader after the current leader refuses (only by sending a message to all others)	The substitute becomes the leader. The others find out either after the substitute informs them or through gossip.
Initial election overhead	Significant spike (may cause a storm)	No overhead costs, equivalent to regular operation
Normal operation overhead	Almost zero (only a heartbeat) until the next election	Low (periodic status/metrics gossip), customizable
Convergence time	Fast (using identifiers) or medium (using scores)	Slower – converges in several rounds of gossip, depending on the depth of the cluster. Deterministic when data is distributed throughout the system
Determinism	Deterministic outcome (the same scores give the same result)	They may diverge for a short time if the view hasn't yet been coordinated

The end of the table 2

Aspect	Fast Bully algorithm with substitutes	Enhanced Gossip with deterministic selection
Suitability for unstable networks	Very sensitive to message loss during elections	Resilient (gossip can withstand partial loss of messages)
Scalability (cluster size)	Best suited for small and medium-sized clusters (≤ 50 nodes)	Easily scalable to large clusters (100-500 nodes)
Consistency of data in the selection of the master node	Consistent representation after the election	May be inconsistent during the negotiation process
Customizability of score	Uses an identifier or score	Can use any score
Risk of split-brain situation (2 masters)	Very low (stable representation)	Medium, possible during delayed gossip messages
Traffic spike in case of a malfunction of the master node	Gathering of scores (only when using scores) and transmission of election results	No spike, information spreads gradually
Total bandwidth usage	Low (with significant spikes)	Higher (periodic messaging), but without spikes

As shown in Table 5, both algorithms have their advantages depending on the system operating conditions. The Fast Bully Algorithm provides high performance and a clear leadership mechanism through the use of identifiers and substitutes, which makes it effective in stable networks with limited latency.

On the other hand, the improved Gossip algorithm demonstrates better resilience in a dynamic environment with intermittent connectivity, avoids excessive traffic, and is easily scalable, which is especially important for large or unstable clusters. Thus, the choice of algorithm should be based on the requirements for scalability, resilience, and the characteristics of changes in the network environment.

Conclusions

The article proposes a method for efficiently selecting the master node (coordinator) and forming optimal computing pipelines in DTS. To implement the method, we have improved the Gossip leader election algorithm with deterministic selection, which avoids election storms, reduces the overhead of messaging, and ensures fast reassignment of the coordinator.

One of the features of the proposed method is the use of an evaluation function that takes into account the average delay between nodes, the resource capacity of nodes, and normalized identifiers. This allows for flexible and efficient selection of cluster coordinators, taking into account topological and resource constraints.

Enhanced Gossip leader election algorithm provides:

– no need for an explicit procedure for electing a coordinator;

– fast automatic assignment of a new coordinator in case of failure of the previous one with the help of backup candidates (substitutes);

– stable operation in conditions of dynamically changing network topology and partial loss of messages;

– ease of implementation through periodic exchange of lightweight messages (gossip metrics).

Experimental modeling has confirmed the fast convergence (4–6 seconds for 50 nodes) and high stability of the proposed algorithm under the conditions of typical DTS operation. Comparative analysis has shown that the proposed improvement of the Gossip leader detection algorithm scales effectively for large clusters and provides stability in unstable networks, unlike traditional algorithms.

Thus, the results of the study confirm the feasibility and effectiveness of applying the proposed method and the improved algorithm for selecting a leader to ensure the reliable functioning of the DTS.

REFERENCES

[1] Attiya H., & Welch J. (2004). Distributed Computing: Fundamentals, Simulations, and Advanced Topics. A JOHN WILEY & SONS, INC., PUBLICATION, P. 414 http://lib.ysu.am/disciplines_bk/c95d04e111f3e28ae4cc589bfda1e18b.pdf

[2] Chandrakant K., Piwowarek G. (2024) Consensus Algorithms in Distributed Systems, https://www.baeldung.com/cs/consensus-algorithms-distributed-systems

[3] Ongaro D., Ousterhout J. In Search of an Understandable Consensus Algorithm, 2014 //https://web.stanford.edu/~ouster/cgi-bin/papers/raft-atc14.pdf

[4] Tel, G. Introduction to Distributed Algorithms. Cambridge University Press, 2000 https://doi.org/10.1017/CBO9781139168724

[5] Bakhshi, R., et al. Leader Election in Anonymous Rings: Franklin Goes Probabilistic // IFIP International Federation for Information Processing, 2008. Volume 273; PP. 57–72, https://satoss.uni.lu/members/jun/papers/TCS08.pdf
[6] Chang Y.-J., & Jiang S. The Energy Complexity of Las Vegas Leader Election. Computer Science – Data Structures and Algorithms, 2022. DOI: 10.48550/arXiv.2205.08642
[7] Dolev S., Israeli A., & Moran S. Uniform dynamic self-stabilizing leader election. IEEE Transactions on Parallel and Distributed Systems (Volume: 8, Issue: 4, April 1997), PP. 424–440, DOI: 10.1109/71.588622
[8] Gilbert S., Robinson P., & Sourav S. Leader Election in Well-Connected Graphs. Computer Science. Distributed, Parallel, and Cluster Computing, 2019. https://doi.org/10.48550/arXiv.1901.00342
[9] Khan, M. S., & Ahmad, I. (2016). A Dynamic Leader Election Algorithm for Decentralized Networks, Journal of Computer and Communications, Vol.4 No.1, January 2016, DOI: 10.4236/jcc.2016.41001.
[10] Kim T. W., & Kim T. Y. Predictable Leader Election Algorithm, Concurrent Engineering, 1995. Volume 3, Issue 3. https://doi.org/10.1177/1063293X9500300305
[11] Kowalski D. R., & Mosteiro M. A. Time and Communication Complexity of Leader Election in Anonymous Networks. Computer Science. Distributed, Parallel, and Cluster Computing, 2021. https://doi.org/10.48550/arXiv.2101.04400
[12] Lavault C., & Louchard G. Asymptotic Analysis of a Leader Election Algorithm. Computer Science. Distributed, Parallel, and Cluster Computing, 2006. https://doi.org/10.48550/arXiv.cs/0607032
[13] BeaulahSoundarabai P., Thriveni J., Venugopal K. R., & Patnaik L. M. An Improved Leader Election Algorithm for Distributed Systems. International Journal of Next-Generation Networks (IJNGN) Vol. 5, No. 1, March 2013, https://airccse.org/journal/ijngn/papers/5113ijngn02.pdf
[14] Ulukök Mehtap Köse, Sarıyıldız İrfan Evri Vesile. Hybrid Raft-PoW Blockchain Consensus Algorithm, 2024. // IEEE Access, P.P. (99):1–1, DOI:10.1109/ACCESS.2025.3562725
[15] Syvolovskyi I. M., Lysechko V. P. A method of hierarchical clustering of nodes in distributed telecommunication systems using graph algorithms. 2025. National University «Yuri Kondratyuk Poltava Polytechnic». Control, Navigation and Communication Systems, 2(80).
[16] Hirschberg D. S., & Sinclair J. B. Decentralized Extrema-Finding in Circular Configurations of Processors. Communications of the ACM, 1980, Vol 23, Issue 11, P. 627. https://dl.acm.org/doi/pdf/10.1145/359024.359029

Сиволовський І. М., Лисечко В. П.

МЕТОД ВИБОРУ ГОЛОВНОГО ВУЗЛА ТА ФОРМУВАННЯ КОНВЕЄРІВ ОБРОБКИ У РОЗПОДІЛЕНИХ ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМАХ

У статті запропоновано метод вибору головного вузла в РТС з кластерною архітектурою та конвеєрною обробкою даних. Метод спрямований на забезпечення стійкого керування інформаційними потоками у кластері за умов динамічного змінення навантаження, нестабільності мережевих з'єднань і обмеженості обчислювальних ресурсів. На відміну від класичних процедур вибору лідера, які базуються на глобальній синхронізації або широкомовних виборчих алгоритмах, запропонований підхід реалізує детермінований вибір координатора на основі локального ранжування вузлів з урахуванням метрик затримки, обчислювальної потужності та унікальних ідентифікаторів.

У межах запропонованого методу удосконалено алгоритм вибору лідера Gossip, за рахунок доповнення механізмами періодичного обміну метриками, локального ранжування кандидатів, призначення резервних вузлів і автоматичного перемикання керування у разі виявлення відмови. Алгоритм передбачає збереження актуального стану вузлів у вигляді локальних списків, а також використання контрольних повідомлень типу heartbeat для підтвердження активності головного вузла. Розроблено механізми обмеження надлишкового поширення інформації через введення TTL (Time-To-Live) і маркерів ітерацій, що унеможливлює циркуляцію застарілих повідомлень.

Експериментальне моделювання показало, що удосконалений алгоритм забезпечує повну збіжність даних у кластері з 50 вузлів за 4–6 секунд, демонструє високу стійкість до втрат повідомлень (≤1 %) та мінімальні затримки при автоматичному перепризначенні координатора. У порівнянні зі швидким алгоритмом хулігана, запропонований підхід зменшує загальний мережевий трафік у фазі відновлення керування до 18 % і підвищує стабільність роботи кластеру в умовах частих змін топології.

Таким чином, запропонований метод дозволяє ефективно управляти кластерними РТС з конвеєрною обробкою даних без потреби у запуску виборчих процедур, що робить його придатним для впровадження у масштабовані та критично навантажені телекомунікаційні середовища.

Ключові слова: розподілені телекомунікаційні системи (РТС); самоорганізація; оптимізація; мережа; вузли; топологія; відмовостійкість; затримка; алгоритми; потужність; кластер.

Syvolovskyi I., Lysechko V.

METHOD FOR LEADER NODE SELECTION AND PROCESSING PIPELINE FORMATION IN DISTRIBUTED TELECOMMUNICATION SYSTEMS

The paper proposes a method for selecting the master node (coordinator) in distributed telecommunication systems (DTS) with a clustered architecture and pipeline-based data processing. The method is aimed at ensuring stable data flow management within a cluster under conditions of dynamically changing workloads, network instability, and limited computational resources. Unlike classical leader election procedures that rely on global synchronization or broadcast-based voting algorithms, the proposed approach implements deterministic coordinator selection based on local node ranking, taking into account latency metrics, computational capacity, and unique identifiers.

As part of the proposed method, the Gossip-based leader election algorithm has been enhanced by integrating mechanisms for periodic metric exchange, local candidate ranking, pre-assignment of backup nodes, and automatic control transfer in case of coordinator failure. The algorithm maintains the current state of nodes as local lists and uses heartbeat-type control messages to confirm the coordinator's activity. To prevent redundant message propagation, mechanisms such as Time-To-Live (TTL) and iteration markers have been introduced, which eliminate the circulation of outdated data.

Experimental modeling shows that the improved algorithm achieves full data convergence in a 50-node cluster within 4–6 seconds, demonstrates high resilience to message loss (≤1 %), and ensures minimal delay during automatic coordinator reassignment. Compared to the fast Bully algorithm, the proposed approach reduces total control recovery traffic by up to 18% and significantly improves cluster stability under frequent topology changes.

Thus, the proposed method enables effective management of clustered DTS with pipeline processing without initiating explicit election procedures, making it suitable for deployment in scalable and high-load telecommunication environments.

Keywords: distributed telecommunication systems (DTS); self-organization; optimization; network; nodes; topology; fault tolerance; latency; algorithms; computational capacity; cluster.

Стаття надійшла до редакції 19.05.2025 р.
Прийнято до друку 11.06.2025 р.