Efficient IoT Network Congestion Management using Genetic Algorithm-based RTO Calculation on Server Nodes

Mojtaba Najafi Moghadam
Department of Computer Engineering ImamReza International University
Mohammad Hossein jafari
Department of Computer Engineering ImamReza International University
Mohammad Nazari Manzari
Department of Computer Engineering ImamReza International University
Mohammad Hossein Mahrooghi
Department of Computer Engineering ImamReza International University

Abstract

This document presents a novel approach to alleviate congestion in IoT networks using Genetic Algorithm (GA) which implemented in server node. The GA adapts to varying traffic loads which gradually leads to mitigate the overload on client nodes . The integration of GA within server nodes ensures scalability and real-time responsiveness, satisfying the need of centralized control. The study explores the synergy between GA principles and IoT network congestion, demonstrating improved performance and adaptability through simulations. This GA-based approach contributes to efficient IoT network management, offering a distributed solution to congestion challenges within dynamic IoT environments

***Keywords*— congestion control, Genetic Algorithm, Internet of Things(IOT).**

I. INTRODUCTION

Managing network congestion is a tough challenge in the world of communication rules, and the Constrained Application Protocol (CoAP) is no exception. As the Internet of Things (IoT) keeps growing, CoAP has become crucial for making devices and servers communicate well, especially when devices have limited resources. One popular way to deal with congestion is to use the Genetic Algorithm (GA). This is a flexible method that can change according to how the network behaves. But previous attempts [1] had problems. These challenges have been solved by a innovative yet problematic approach which was the implementation of the GA on client nodes. While enhancing performance, this strategy inadvertently triggered issues like escalated power consumption.

So, here's a new idea. We're looking at using the Genetic Algorithm in a different way within the CoAP protocol. We realized need for a solution that can works well and keeps going as the IoT grows. Instead of putting the GA on client devices, we're trying it on servers. This change could solve the power problems we've seen before and enhance the throughput of the network.

II. RELATED WORK

Now we will delve into the research background in this area and take a closer look at the challenges these studies have encountered.

Congestion in IoT networks is one of the major work-related obstacles in this field. Two issues contribute to this problem: firstly, the transmission of a significant volume of packets towards server nodes, and secondly, the retransmission of packets when ACK is not received at client nodes.

In the realm of IoT communications, packet retransmission is an inevitable and regular occurrence. As a result, avoiding this process is not feasible.The fundamental principles of CoAP for congestion control involve constraining the number of concurrent messages (parallel CON messages) and regulating the outgoing message rate [2]. This approach begins by imposing a cap of one active interaction per destination. An active interaction refers to any request, whether CON or NON, that awaits an ACK or reply. Should an ACK not be received, a CON message can be retransmitted up to four times before considering it a failure. The initial Round Trip Time (RTT) value is randomly selected from the range of [2,3] seconds. Subsequent retransmissions utilize Binary Exponential Backoff (BEB), entailing doubling of the RTT value [2], To enhance responsiveness and adaptability to Round-Trip Time (RTT) information, a more refined congestion control mechanism is presented by August B. and colleagues, denoted as CoCoA/CoCoA+ in [2] and [3].

CoCoA is an advanced congestion management mechanism tailored to optimize the CoAP protocol's [4]performance within IoT networks. It revolves around refining the retransmission strategy based on Round Trip Time (RTT) measurements, ultimately enhancing communication reliability and mitigating congestion-related challenges.

CoCoA introduces two distinct RTO estimators: the Strong RTO Estimator (SRTT) and the Weak RTO Estimator (WRTT). SRTT calculates the RTO from the RTT measurements of successfully transmitted packets during the initial transmission, while WRTT is based on RTT data from packets that required retransmission (with up to two retransmissions).

The RTO calculation is a balanced combination of SRTT and WRTT, where the resulting RTO value is a weighted average, determined by αs (alpha strong) and αw (alpha weak):

$$RTO = \alpha s * SRTT + (1 - \alpha s) * RTO \qquad (1)$$

CoCoA introduces Variable Backoff Factor (VBF) to replace conventional Binary Exponential Backoff (BEB) for retransmitted packets, adapting RTO based on initial values. An RTO aging mechanism adjusts RTO when RTT measurements are lacking. This comprehensive approach enhances CoAP's congestion control, improving reliability and efficiency in IoT network communication.

"Genetic CoCoA++: Genetic Algorithm based Congestion Control in CoAP" introduces an innovative approach implemented in client nodes. This approach combines genetic algorithms with the CAIA Delay-Gradient (CDG) [5] mechanism to enhance congestion control within the UDP network. Genetic algorithms, drawing inspiration from natural evolution, employ techniques such as crossover and mutation[6, 7] to optimize solutions. The CDG mechanism, initially designed for TCP congestion control, undergoes modifications tailored to the specific characteristics of the UDP network.

The core focus of this method is to enhance congestion detection by leveraging RTTmin and RTTmax metrics over predefined intervals, further refined through moving averages. Repetitive crossovers and calculations of backoff probabilities are incorporated into the Genetic Algorithm. These calculations are influenced by variations in RTTmin, ultimately leading to adjustments in Round

Trip Timeout (RTO) values. The integration of genetic algorithms and CDG aims to improve network performance by effectively managing congestion, thereby enhancing the reliability of data transmission.

III. PROPOSED METHOD

As previously mentioned, the authors successfully maintained the RTO within its optimal range by employing the genetic algorithm on client nodes [1]. However, this implementation method poses its own set of challenges. Client nodes face issues such as resource scarcity and energy constraints when carrying out the necessary calculations for determining RTO, resulting in excessive resource consumption. This consumption increases with the growing count of network nodes.

Our proposed solution to address this problem involves implementing the genetic algorithm for RTO calculations on server nodes instead of client nodes. This decision is based on the absence of the aforementioned problems and constraints faced by server nodes, allowing for more efficient network congestion control while minimizing resource usage in the context of IoT.

To conduct RTO calculations on server nodes, specific time intervals are taken into account. The genetic algorithm optimizes the RTO value within these intervals. In our approach, we consider a time interval of 10 seconds. During this 10-second period, the server collects information from client nodes, including packet transmission times and the reception times of previous packets. This data is associated with the IP addresses of the client nodes, enabling the calculation of Round Trip Time (RTT) using these variables.

Within the genetic algorithm's functionalities, chromosomes are initially generated with random data. Subsequently, operations such as mutation, evolution, crossover [6, 7], and fitness function evaluation are performed on this data to select the most optimal RTO value.

This approach aims to effectively manage network congestion by leveraging the capabilities of the genetic algorithm within server nodes, thereby optimizing the performance of IoT networks.

For a more detailed workflow, as IoT client nodes continually transmit packets within the network, the server node receives this data and identifies the client node that sent each packet. All information is then categorized based on IP addresses to prevent disruptions and interference in computations caused by specific client nodes' data.

The categorized data is utilized in an algorithm, specifically a genetic algorithm, to calculate the optimal Round-Trip Time Out (RTO) and regulate network congestion. The subsequent steps outline the RTO calculation process:

Initially, a number of chromosomes equal to the genome size (x) are created. For each of these genomes, a random value between the minimum Round-Trip Time (RTT) and a slightly higher value than the maximum RTT is assigned.

$$\mathrm{RTO}_i = \mathrm{RTT}_{min} + (\mathrm{RTT}_{max} - \mathrm{RTT}_{min}) * RN \quad (1)$$

Where RN is a uniform random value between 0 and 1.

Next, the average of received client RTTs is computed (2), and this value is used in an evaluation function to select the most optimal chromosome.

$$\mathrm{RTT}_{\mathrm{avg}} = \frac{\sum_{\mathrm{i}}^{\mathrm{n}} \mathrm{RTT}_{\mathrm{i}}}{\mathrm{n}} \quad (2)$$

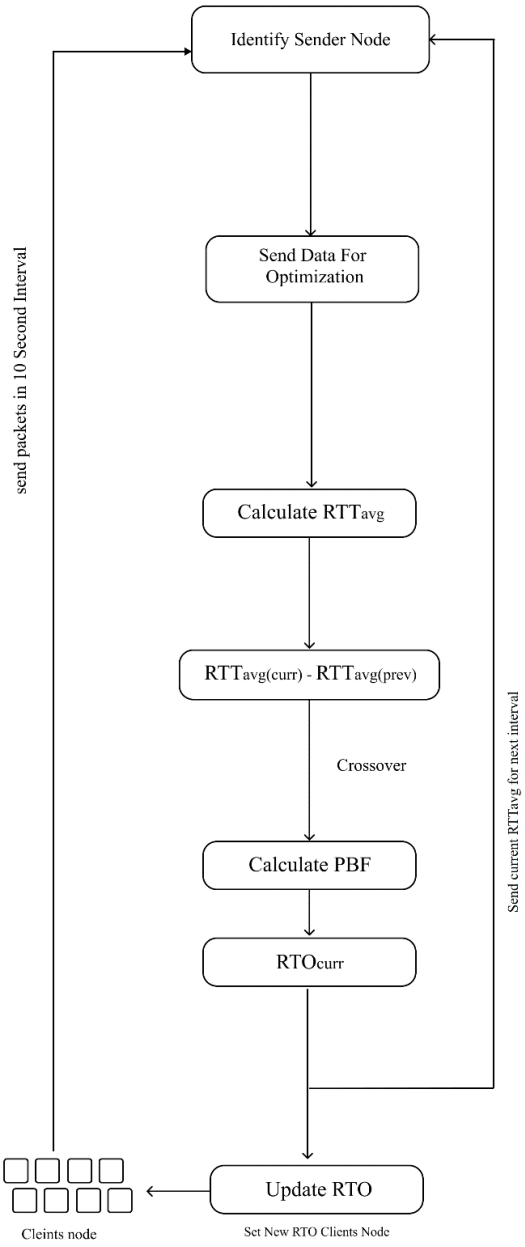

Fig. 1 . GA in Server Node Flowchart

In the subsequent step, the selected chromosomes undergo pairwise crossover, resulting in a new generation of diverse RTOs.

From the new generation of chromosomes, a parent chromosome is chosen for evolution based on its performance, and it replaces the chromosome with the lowest fitness.

In the final step of the genetic algorithm, mutations are introduced to the chromosomes of the new generation, aiming to obtain the best RTO suited to the current network conditions, the workflow illustrated in Fig. 1

The obtained RTO from the previous stages is transmitted to client nodes that receive ACKs during the next interval. This process ensures that the previous RTO value adapts to the current network conditions. In each time interval, the RTO in client nodes

is updated accordingly which presented in Fig. 2. As the network experiences congestion, client nodes delay the retransmission of their packets based on the RTO value. This action prevents network congestion and optimizes resource utilization in client nodes, given that these operations take place in the server node.

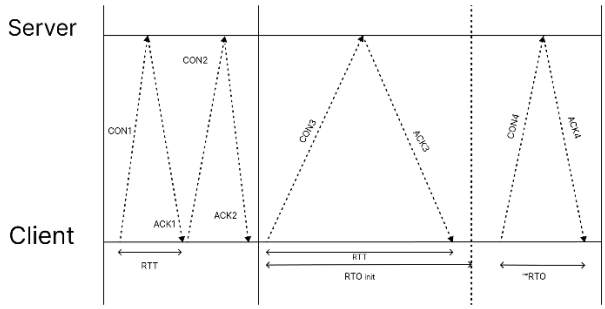

Fig. 2 . Message communication between a client and server.

IV. SIMULATION SETUP

This section presents the evaluation of the proposed genetic algorithm and the steps of a simulated wireless sensor network using the Contiki Cooja simulator which is an suitable open-source operating system for constrained networks [8]. Using former method with new modifications we conducted a simulation in cooja which is a tool of Contiki.

Our setup consists of a Grid topology with 24 sky mote client nodes and one server node (labeled as node 1) deployed in the center. The Sky mote type was utilized, it's a system designed to support high data rate sensors and low power networks in various application further network characteristics are summarized in Table I.

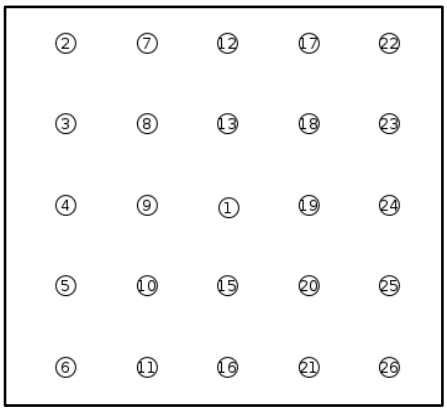

Fig. 3 .Grid topology.

Table I

Parameters	Values
Operating System	Contiki-3.0
Network Simluator	Cooja
Mote Type	Sky
Number of Nodes	24 Client & 1 Server
Interference Range of Nodes	20m
Transmission Range of Nodes	10m
Simulatioin Duration	30 mintures
Retransmission Limit	4

V. EVALUATION

The evaluations were conducted within a 30-minute timeframe, consistent with a previous study (GA CoCoA++) where simulations were completed. The simulation clock is plotted on the x-axis, and RTO values are depicted on the y-axis. The figures portray the fluctuation of RTO values during simulations in both Genetic CoCoA++ scenarios, including one involving a server node. Each data point signifies the RTO value initiated at a specific

time as denoted on the simulation clock. Additionally, we observed the incidence of failed packets, resulting from inaccurate RTO estimates. This data allows for the computation of the Packet Failure Rate (PFR), representing the ratio of failed packets to the total generated in the simulation. The results, encompassing average and maximum RTO values and PFR, are detailed in Table III. This table serves to compare both approaches across diverse topologies.

Method	Genetic CoCoA++	Server side GA
Topologies	Grid	Grid
Max RTO (s)	246.17	251.89
Avg. RTO(s)	15.78	21.73
PFR	0.0945	0.1104

Table II

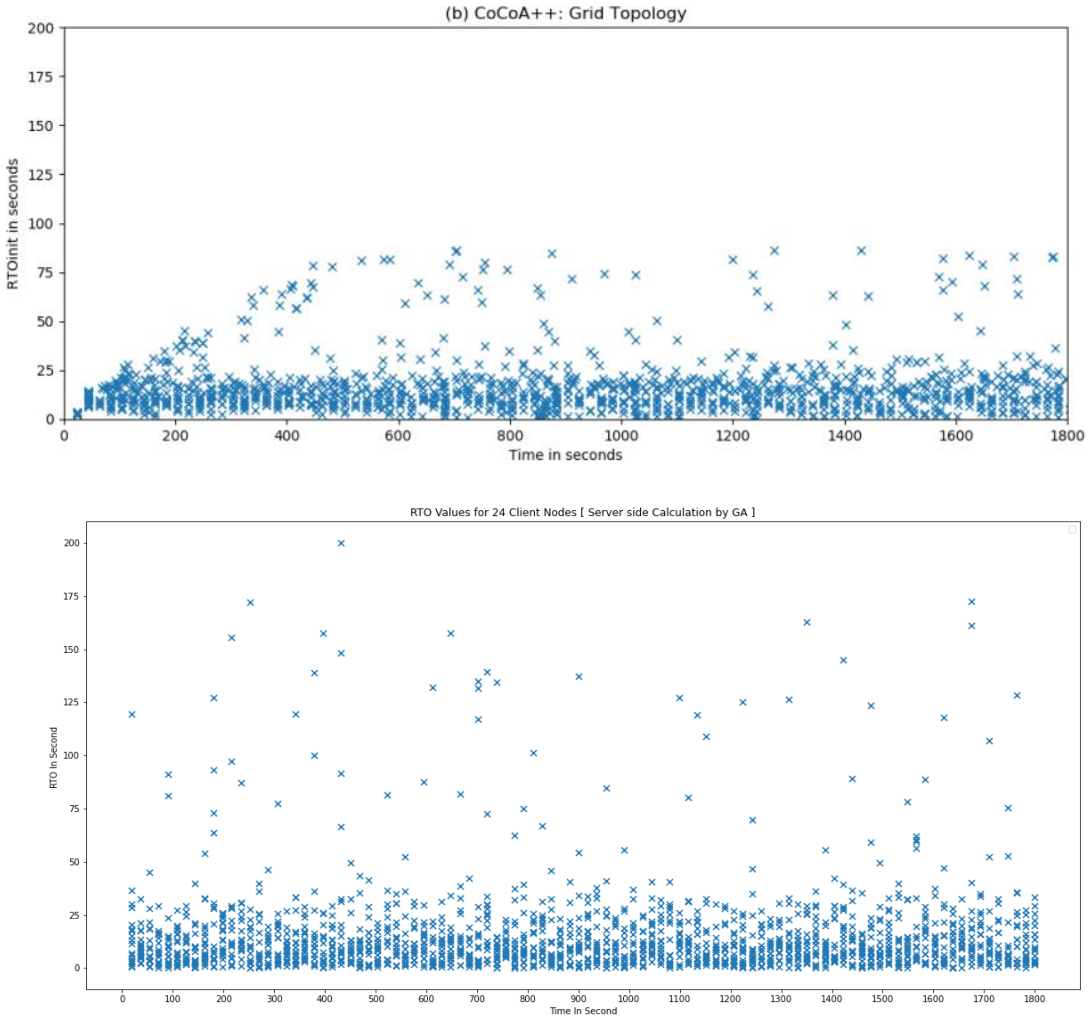

Fig. 4 . Comparing (a) Genetic CoCoA++ & (b) Genetic CoCoA++ in server node using Grid Topology.

VI. IN CONCLUSION

In this article, we addressed the challenges presented in previous studies and offered our solution to resolve these issues. Ultimately, our solution leads to reduced energy consumption, resource utilization, and consequently, cost reduction. These issues

primarily revolve around Round Trip Timeout (RTO) calculations performed extensively by client nodes in previous articles, which lead to excessive resource consumption in IoT networks. Additionally, these calculations in client nodes are condition-specific, making it impractical to perform them under changing network conditions. Hence, our approach initially involves aggregating these calculations on server nodes to prevent unnecessary resource consumption. Subsequently, an algorithm or method for scheduling the processing of all data sent to the server node needed to be devised.

Sent packets towards the server are managed by a buffer in the form of a queue. However, when the number of packets sent to the server is considerably high, there arises a need for a better data structure and method to manage the scheduling of processing these packets. In future research, an alternative data structure like a matrix and scheduling algorithms could be employed instead of the buffer data structure.

REFERENCES

1. Yadav, R.K., N. Singh, and P. Piyush. *Genetic CoCoA++: genetic algorithm based congestion control in CoAP*. in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*. 2020. IEEE.
2. Betzler, A., et al., *CoAP congestion control for the internet of things.* IEEE Communications Magazine, 2016. **54**(7): p. 154-160.
3. Matt Sargent , J.C., Dr. Vern Paxson , Mark Allman *Computing TCP's Retransmission Timer.* 2011.
4. Betzler, A., et al., *CoCoA+: An advanced congestion control mechanism for CoAP.* Ad Hoc Networks, 2015. **33**: p. 126-139.
5. David A. Hayes, G.A., *Revisiting TCP Congestion Control Using Delay Gradients.* 2011.
6. David E. Goldberg, J.H.H., *Genetic Algorithms and Machine Learning.* 1988.
7. Eiben, A. and J. Smith, *Introduction to Evolutionary Computing (Natural Computing Series) Springer.* Berlin/Heidelberg, Germany, 2008.
8. F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, Cross-level sensor network simulation with COOJA, in: Proceedings 2006 31st IEEE Conference on Local Comput er Networks, 2006, pp. 641–648.