mpQUAD: Multipath Quad TCP Congestion Control in FANETs

Neethu Subash*, B.Nithya, Vipul Patel, and Rahul Bangar
*Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, Tamilnadu, India

***Abstract*—AIMD (Additive Increase Multiplicative Decrease) and CUBIC (Cubic Congestion Control) are the two commonly used algorithms for network congestion control in the UAV (Unmanned Aerial Vehicle). AIMD and CUBIC can control the data transfer rate between the UAV and the ground station or other UAVs in a swarmed network. This is particularly important for real-time applications using flying adhoc networks (FANET), such as surveillance or monitoring, where timely data delivery is critical. Multiptah TCP utilizes individual subflows to implement congestion control. Nevertheless, the default congestion management mechanism for subflows in an MPTCP connection uses a linked increase adaptation technique to prevent the congestion window from rapidly expanding due to subflows independently developing their own congestion windows. The throughput of MPTCP connections may decline if fast algorithms like CUBIC TCP are employed in high speed congested network. This work proposes mpQUAD, a novel CUBIC TCP-based high-speed congestion management technique for MPTCP. It exposes specific control parameters of the algorithm to tweak the systems TCP congestion control behavior. The sender's congestion window can be controlled by changing the multiplicative factor and the rate at which it grows, by the user. The throughputs of MPTCP flows decrease in the conventional congestion control algorithms. The limited bandwidth and high mobility of FANETs can cause significant delay, which the proposed congestion control algorithm mpQUAD can mitigate.**

***Index Terms*—congestion control, mpCUBIC, mpQUAD, MPTCP, FANET**

I. Introduction

Controlling congestion is essential for ensuring efficient and effective use of FANET. FANET [1] is a type of wireless ad-hoc network consisting of unmanned aerial vehicles (UAVs) or drones that communicate wirelessly. FANETs are often used in disaster management, military operations, and environmental monitoring applications [2]. These networks operate in a dynamic and constantly changing environment, which makes it challenging to establish and maintain communication between the nodes. One of the major issues that FANETs face is congestion [3] which occurs when the network becomes overloaded with traffic, leading to delays, packet loss, and reduced network performance. In FANETs, congestion can be caused by several factors, including limited bandwidth, high node density, and mobility-induced changes in network topology. Controlling congestion in FANETs is critical to ensure reliable communication and efficient use of network resources. However, in FANETs TCP's congestion control mechanism can reduce network throughput and increase end-to-end delay due to several factors. One of the major challenges of using TCP in FANETs is the high packet loss rate due to interference, channel fading, and mobility-induced changes in network topology. TCP's congestion control algorithm interprets packet loss as an indication of network congestion and slows the transmission rate, leading to poor network performance. In addition, TCP's congestion control mechanism can lead to excessive delay in FANETs due to the high round-trip time and variable network conditions. TCP's congestion control algorithm slows down the transmission rate when packet loss occurs, and it takes time for TCP to increase the transmission rate back to the optimal level. This delay can be significant in FANETs due to the high round-trip time and variable network conditions, leading to reduced network throughput and increased end-to-end delay. To overcome these challenges, modifications have been proposed to adapt TCP's congestion control mechanism to FANETs. These modifications aim to improve TCP's performance in FANETs by accounting for the unique characteristics of wireless communication, such as high packet loss rate, high round-trip time, and variable network conditions. These modifications include cross-layer design, congestion control algorithms, window-based congestion control, and hybrid approaches. TCP uses congestion control algorithms CUBIC [4], mVeno [5] etc to regulate data flow and adjust the transmission rate based on network conditions. Single path TCP and MPTCP (Multi-path TCP) [6] are two different transmission protocols used in FANETs for data communication. Single path TCP is the traditional TCP protocol that uses a single path to transmit data between two endpoints, while Multi-path TCP uses multiple paths simultaneously to improve network performance. From a performance point of view, Multi-path TCP can provide several benefits compared to Single path TCP in FANETs, including:

- Improved network throughput: Multi-path TCP uses multiple paths to transmit data simultaneously, which can increase the network throughput and improve overall network performance. This is particularly beneficial in FANETs, where bandwidth is limited, and congestion is common.
- Better reliability: Multi-path TCP uses multiple paths to transmit data, which improves network reliability by reducing the impact of packet loss and network congestion. If one path is congested or fails, data can be rerouted through another available path, reducing the risk of data

Corresponding author: Neethu Subash (email: neethu.subash@gmail.com.

loss.
- Reduced end-to-end delay: Multi-path TCP can reduce end-to-end delay in FANETs by using multiple paths to transmit data, which can reduce the impact of network congestion and packet loss. This can improve real-time application performance, such as video streaming and VoIP.

However, there are also some challenges associated with using Multi-path TCP in FANETs, including:

- Increased complexity: Multi-path TCP is a more complex protocol than Single path TCP, which can make it more difficult to implement and maintain in FANETs.
- Increased energy consumption: Multi-path TCP uses multiple paths to transmit data, which can increase energy consumption in FANETs, particularly in resource-constrained nodes such as UAVs.
- Path selection: Multi-path TCP requires efficient path selection mechanisms to choose the best available paths for data transmission. In FANETs, the dynamic and fast-changing network topology can make path selection challenging.

CUBIC is a congestion control algorithm that aims to achieve fast and stable throughput. It is designed to support the transfer of large volumes of data and is considered a fast congestion control algorithm. In CUBIC, the sender adjusts the congestion window size based on receiving acknowledgments (ACKs) from the receiver. On receiving an ACK, CUBIC calculates a target window size and adjusts the congestion window towards this target. Multi path CUBIC supports multiple sub flows and uses a cubic function to estimate network conditions and adjusts the congestion window size accordingly. While mpCUBIC is effective congestion control algorithms [7] in FANETs, there are several limitations to using CUBIC and mpCUBIC over mpQUAD:

- Limited support for multi-path transmission: CUBIC was designed for use in single-path networks and does not support multi-path transmission. In contrast, mpQUAD is specifically designed for multi-path networks and can better handle the complexities of FANETs.
- Higher delay and packet loss: In FANETs, there can be significant variations in the delay and packet loss rates on different paths. CUBIC may not be able to adapt quickly enough to these variations, leading to higher delays and packet loss. mpQUAD can handle these variations better and provide better performance in FANETs.
- Limited scalability: As the number of UAVs in a FANET increases, the network becomes more complex and difficult to manage. CUBIC may not be able to scale up to handle large FANETs, while mpQUAD is designed to be scalable and can handle larger networks more effectively. The proposed work mpQUAD is based on CUBIC allows to achieve fast convergence to high throughput and responds quickly to changes in network conditions, leading to improved network performance compared to other congestion control algorithms.

To address these limitations, mpQUAD is designed to support the fast transfer of large volumes of data in FANETs. By dynamically adjusting the congestion window size based on the reception of ACKs and using a QUAD function to estimate network conditions, mpQUAD can achieve fast and stable throughput, improving network performance. The congestion window size as a function:

$$W(time) = Q(t-K)^4 + W_{max} \tag{1}$$

where Q is the Quadruple parameter, t is the elapsed time from the last congestion window size reduction refers to the time that has passed since the congestion window size was last decreased due to network congestion. This elapsed time is used to calculate the congestion window size. The idea behind using this elapsed time is to gradually increase the congestion window size over time, as long as the network conditions allow for it, to make better use of the available bandwidth and prevent congestion from occurring. W_{max} is when the most recent packet loss event occurred. K represents the time it takes for the congestion window size to reach its maximum value after a reduction due to network congestion. As the algorithm increases the window size as the network reaches saturation. This property makes mpQUAD highly scalable and efficient in networks with large capacity and delay products and multiple pathways to improve performance and increase redundancy. The paper discusses the implementation of the mpQUAD algorithm as a Linux kernel module and its analysis. The module's unique feature allows specific QUAD congestion window function control parameters to be exposed to user space through a graphical user interface (GUI). This allows users to manipulate these parameters in real-time and affect the behavior of TCP congestion control. A simulation environment was created to analyze the behavior of mpQUAD. The simulation creates a bottleneck link, and multiple TCP flows are started to share the link. The captured data is visualized to analyze the congestion window size, throughput, and queueing delays. This helps to evaluate the performance of the mpCUBIC algorithm, mpQUAD and its ability to achieve fast and stable throughput in computer networks. mpQUAD extends the mpCUBIC algorithm by incorporating a new estimator to improve network performance. The new estimator uses a quadratic function to more accurately reflect network conditions, resulting in improved congestion control and better overall network performance.

The paper presents experimental results demonstrating the superiority of the mpQUAD algorithm over mpCUBIC. The subsequent sections of this paper will be organized as, in section 2, related work is investigated and examine various solutions that have been previously proposed. Section 3 provides an in-depth explanation of the mpQUAD algorithm. Section 4 presents the results obtained by simulating mpQUAD. Section 5 is the conclusion, where the work is summarized with findings and future scope.

II. Related Work

The study of congestion control has been an active area of research for many years, with several algorithms developed to address the issue of network congestion. One such algorithm is the CUBIC congestion control algorithm, widely used in both wired and wireless networks. However, CUBIC has been shown to have some limitations, particularly in wireless networks with high loss rates. As a result of this, new algorithms have been created to enhance the effectiveness of CUBIC and other congestion control algorithms in wireless networks. This section summarizes the research on congestion control in FANET and explore its constraints. The article [5] suggests a new congestion control algorithm called mVeno, which aims to enhance the performance of MPTCP in wireless communication networks with random packet loss. MPTCP enables multiple paths to be used in a TCP connection simultaneously. However, the current congestion control algorithms used are loss-based and prefer routes with low failure rates, which can negatively impact wireless network performance. The mVeno approach dynamically adjusts the transmission rate of each subflow, utilizing congestion data from all subflows associated with the TCP connection. The mVeno technique decouples the congestion control for each subflow, enabling data to be transmitted simultaneously over multiple channels. The new congestion control algorithm is designed to overcome the limitations of the current loss-based algorithms, improving the performance of MPTCP in wireless networks with high levels of random packet loss. The algorithm [8] is a congestion control algorithm designed to improve the performance of TCP in high bandwidth-delay product networks. It extends the Cubic TCP congestion control mechanism to support multiple network paths. Balances traffic between multiple paths during bottlenecks to ensure fair sharing with other TCP variants such as standard TCP, Cubic, and MPTCP. This helps to improve resilience to link failures and reduce the time for restoring data rates. The algorithm is handy for high-bandwidth applications that require reliability, such as telemedicine conferencing in disaster-affected areas. However, one of the challenges with multipath transport protocols is the issue of sequence number space, which involves the flow or multipath link level. A single TCP stream may be divided over many pathways in MPTCP [9]. It offers clear advantages in terms of dependability and performance. Linux-based distributions have MPTCP implemented, which can be installed and utilised in both practical and hypothetical situations. The results reveal that the Balanced Linked Adaptation method outperforms the others when the pathways are shared with heavy traffic, which is not allowed by MPTCP. In contrast, Alias Linked Increase Congestion Control approach beats the others under average traffic load. The Linux operating system is used to measure the throughput of various situations. The results of all cases are compared, and the effectiveness of various MPTCP congestion control methods is evaluated. An Approach [10] to Reinforce Multipath TCP with Path-Aware Information Mp TCP, which allows multiple wireless connections, is a great way to create multi-homing devices on cellular networks. Suppose the throughput of MPTCP over a divergent channel is lower than ideal for single-pass TCP. In that case, there is a significant problem of negative aggregation benefit that MPTCP with PA can solve. The work demonstrates the effectiveness of the recommended strategy through the recent installation of the so-called MPTCP-LA (that is, loss-aware MPTCP). To keep the overall power positive, MPTCP-LA temporarily switches MPTCP transmissions on the route to sleep mode when the observed loss at the device exceeds a certain threshold. D-LIA [11] is a dynamic congestion control algorithm designed specifically for MPTCP, which can adaptively adjust the congestion window size and control the transmission rate based on the network congestion level. It achieves improved performance and fairness in multi-homed networks. Need to improve throughput in highly congested FANET scenarios. The paper on [12] proposes a bidirectional congestion control transport protocol (BCCTP) for the Internet of Drones (IoD) that utilizes a congestion window algorithm and a selective repeat mechanism to mitigate congestion and ensure reliable data delivery. BCCTP improves network utilization and reduces packet loss in IoD scenarios. However, when the number of UAVs are increased, the performance degrades drastically. mpCUBIC [4] is a congestion control algorithm for Multipath TCP that extends the well-known CUBIC algorithm to handle multiple network paths, improving overall throughput and fairness. It dynamically adapts its congestion window and pacing rate based on each path's available bandwidth and delay. But mpCUBIC may not perform optimally in networks with highly asymmetric paths or paths with different loss characteristics. Factors that lead to improvising congestion control algorithm for FANET include,

- Unresponsive to sudden network changes: Most of the congestion control algorithm may take longer to respond to sudden network changes, such as congestion or traffic spikes, which can result in reduced network performance.
- May result in long queue delays: Due to long queue delays, especially in high-speed networks, resulting in higher latency and reduced throughput.

III. Proposed work

The main aim of the proposed congestion control mechanism is to improve throughput by minimizing the congestion. A multi path flow should perform well as a single TCP flow that would be on the best of its available paths. An MPTCP connection should use individual sub-flow [13] dependent on the congestion on the route.The mpQUAD determines the congestion window value using the QUAD function of the last packet loss or detected congestion using equation 1.

A. Proposed Model of mpQUAD

The proposed algorithm uses the mpQUAD function to balance fast initial growth during the slow start phase and gradual development during the congestion avoidance phase. It works by controlling the rate at which data is sent, using a mathematical formula to adjust the sending rate based on the

congestion in the network. To do this, mpQUAD keeps track of the amount of data currently in transit in the network, and uses this information to adjust the sending rate accordingly. If the network is congested, mpQUAD will slow the sending rate to prevent further congestion. If the network is less crowded, mpQUAD will speed up the sending rate to maximize data transfer. In the initial phase of data transmission called "slow start", the size of the congestion window increases rapidly and exponentially, doubling after each round trip time. Once the window size reaches a specific limit, the algorithm shifts to the "congestion avoidance" phase. In this phase, the congestion window grows at a slower rate and is determined by a mathematical function that takes into account the time elapsed since the last congestion event. The QUAD TCP algorithm employs a quad function that exhibits a concave curve during slow start and a convex curve during congestion avoidance. The convex curve reflects the gradual growth of the congestion window as the network capacity gets fully utilized. Overall, the quad TCP algorithm is designed to provide efficient and stable congestion control in TCP networks, ensuring that the network does not become congested and packets are transmitted optimally. From equation 1, K can be equated as in equation 2

$$K = \sqrt[4]{\frac{W_{max}\beta}{Q}} \tag{2}$$

where Q is the quad parameter, K is the period that the above function takes to increase W to W_{max} when there is no other loss event, β is the reduction in the window at a quick fast retransmit event. To evaluate whether the protocol is in the TCP area after receiving an acknowledgment, the TCP window size is analyzed to the elapsed time t. This analysis involves the use of a multiplicative factor and an additive factor. The multiplicative factor increases the window size by a certain percentage every RTT. In contrast, the additive factor increases the window size by a fixed amount after every RTT.

$$W_{tcp(t)} = W_{max}\left[1 - \beta + \frac{3\beta}{2-\beta}\frac{t}{RTT}\right] \tag{3}$$

The cwnd setting is W_{tcp} at each ACK receipt if it is less than W_{tcp}. If cwnd is smaller than the W_{tcp} and the protocol is not in the TCP region, concave area of the protocol, then cwnd is increased by equation4.

$$\frac{W(t+RTT)}{cwnd} - 1 \tag{4}$$

If cwnd is more than W_{tcp}, then the protocol is in convex region and the increment in cwnd remains the same as above. In the event of a packet loss, cwnd is reduced by a factor of β. A value lower than 0.5 results in slow convergence so adaptive adjustment of β is an issue that the interactive GUI resolves. The congestion control strategy for a single-path CUBIC flow was covered in the introduction. Two CUBIC sub flows in an MPTCP connection is considered. It is essential to prevent the congestion window from expanding too quickly, which might happen if the sub-flows grow their congestion windows separately. A mpQUAD function whose cycle is one-half of a balanced QUAD growth function also determines the total congestion window size of an MPTCP connection i.e., $K/2$. The improvements considered are:

- For a multi path connection, the packet loss period is doubled.
- Overall cwnd size is set to W_{max}, the largest window size, shortly before a packet loss.

One sub flow's congestion window size in mpQUAD is given by:

$$W(t) = \frac{1}{F}\left[E(\frac{t}{2} - K)^4 + W_{max}\right] \tag{5}$$

The total window size [4] for one MPTCP connection for a single cycle is given by:

$$W_{max}(t) = \begin{cases} \frac{1}{F}\left[E(\frac{t}{2} - K)^4 + E(\frac{t+K}{2} - K)^4 + 2W_{max}\right], \\ when 0 < t < K \\ \frac{1}{F}\left[E(\frac{t}{2} - K)^4 + E(\frac{t-K}{2} - K)^4 + 2W_{max}\right], \\ when 0 < t < 2K \end{cases}$$

1) At t = 0, $W_{total}(-0)(1-\beta) = W_{total}(+0)$ we obtain,

$$\left[E(-\frac{1}{2}K)^4 + 2W_{max}\right](1-\beta)$$
$$E = \frac{16}{8+\beta}C$$

2) At t=$2K$, $W_{total}(2K-0)$=W_{max}, we obtain,

$$\frac{1}{F}\left[E(-\frac{-1}{2}K)^4 + 2W_{max}\right]$$
$$F = \frac{16}{8+\beta}$$

Upon substituting the values of E and F, for the cwnd size of an mpQUAD subflow:

$$W(t) = \left[Q(\frac{t}{2} - K)^4 + \frac{16}{8+\beta}W_{max}\right] \tag{6}$$

Here, Q, t, K, β and W_{max} are the mpQUAD parameters

In algorithm 1, $cwnd$ and $ssthresh$ represent the congestion window size and slow start threshold, respectively. Variables t, rtt, and ack represent the time elapsed since the start of the connection, the round-trip time of the last packet, and the number of acknowledged packets since the previous update, respectively. The target variable represents the target queue size in packets, while β, γ, and α represent the multiplicative decrease factor, additive increase factor, and control parameter for mpQUAD, respectively. Finally, w_{max} means the maximum congestion window size. The algorithm begins by calculating the scaling factor k and estimating the bottleneck

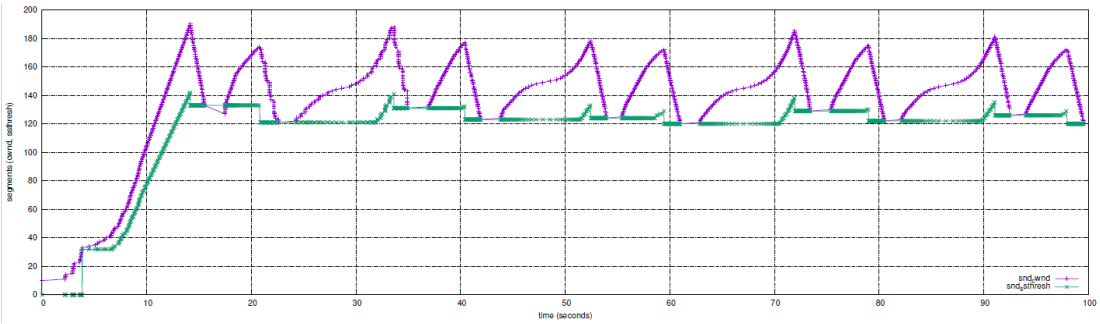

Fig. 1. cwnd, ssthresh for mpCUBIC

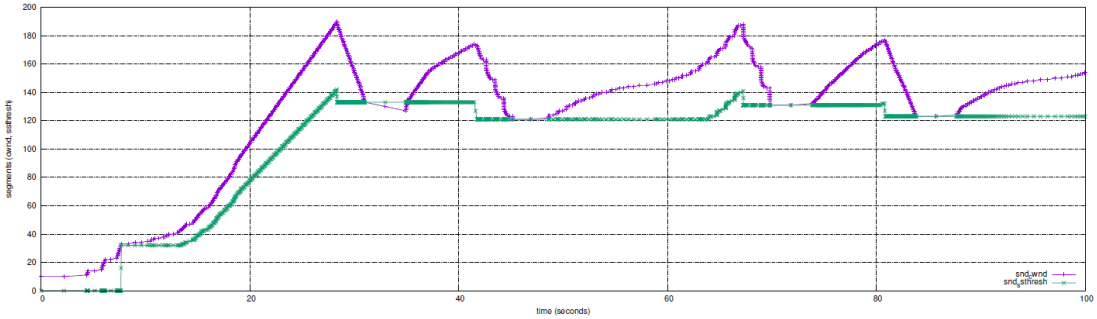

Fig. 2. cwnd, ssthresh for mpQUAD

Algorithm 1 mpQUAD Congestion Control Algorithm

Input: initial cwnd $cwnd$, initial slow start threshold $ssthresh$, time elapsed since the start of the connection t, round-trip time of the last packet rtt, number of acknowledged packets since the last update ack, target queue size in packets $target$, multiplicative decrease factor β, additive increase factor γ, control parameter for mpCUBIC $alpha$, maximum congestion window size w_{max}.

Output: $cwnd$, $ssthresh$

Initialisation :

FUNCTION *mpQUAD*($cwnd$, $ssthresh$, t, rtt, ack, $target$, $beta$, $gamma$, $alpha$, w_{max})

START

$k \leftarrow (\frac{w_{max}}{cwnd})^{\frac{1}{2}}$

$w_{est} \leftarrow cwnd \cdot (\frac{target}{ack})^{\frac{1}{2}}$

$QUAD \leftarrow (\frac{w_{est}}{k}) + \alpha \cdot t$

$\Delta cwnd \leftarrow QUAD - cwnd$

Check if $\Delta cwnd > 0$

  Check if $cwnd < ssthresh$

    $cwnd \leftarrow cwnd + \min(\Delta cwnd, ack \cdot \gamma)$

  Else

    $cwnd \leftarrow cwnd + \min(\Delta cwnd, \gamma)$

$ssthresh \leftarrow \max(cwnd \cdot \beta, 2)$

$cwnd \leftarrow ssthresh$

$cwnd, ssthresh$

END FUNCTION

bandwidth w_est. It then computes the QUAD function and the difference between the current congestion window size and the quad value. If the difference is positive, the algorithm enters the congestion avoidance phase and performs either an additive increase during a slow start or an additive rise during congestion avoidance. If the difference is negative, the algorithm enters the congestion control phase and performs multiplicative decrease by setting the cwnd and ssthresh to appropriate values. The updated cwnd and ssthresh values are returned at the end of the algorithm.

B. mpQUAD Behaviour

When two sub-flows are used in a multi path TCP connection, and both subflows use mpQUAD TCP, the cwnd behavior is designed to be balanced. When both sub-flows go through a bottleneck link and experience packet loss due to congestion, the cwnd of both subflows decreases similarly. The time variation graph's 4 black and red lines represent the cwnd of the two sub-flows. The mpQUAD behavior can be summarized as follows:

- At the beginning of the connection, the cwnd of both sub-flows starts with the ssthresh and increases linearly during the slow start phase.
- Following the initial slow start phase, cwnd of both subflows transitions into the congestion avoidance phase, during which it gradually increases at a slow pace through a minor linear function.
- When packet loss is detected due to congestion, the cwnd of both subflows is reduced similarly. However, due to the difference in RTT of the two subflows, the cwnd of

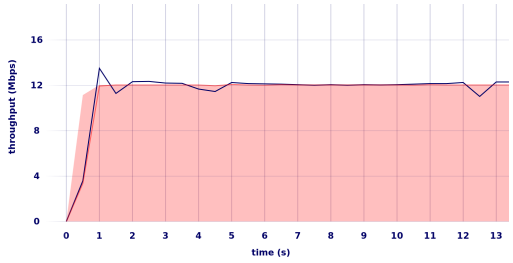

Fig. 3. Throughput for two sub flow in mpCUBIC

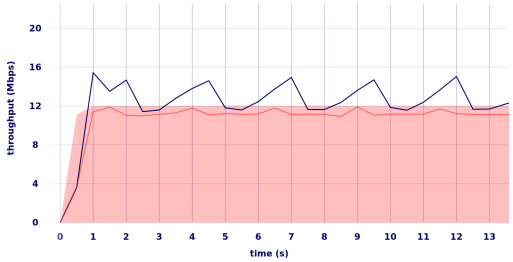

Fig. 4. Throughput for two sub flow in mpQUAD

the subflow with a shorter RTT reduces faster than the subflow with a longer RTT.

- As the cwnd of both subflows decreases, the congestion avoidance phase is re-entered, and the cwnd of both subflows starts to increase slowly again with a small linear function.
- When congestion occurs again, the cwnd of both subflows is reduced, and the process repeats. The aim is to balance the cwnd of both subflows, so that neither subflow dominates the network resources, and the performance of the multi path TCP connection is optimized.

IV. Performance Evaluation

The simulation of the FANET is carried out using Mahi Mahi simulator [14] for TCP flows and TCPTuner GUI [15]. The nodes chosen in the environment are dynamic. A 12Mbps bottleneck uplink with a 117Kib drop-tail queue is set up with an 80ms RTT. This arrangement enables to evaluate the throughput of various TCP flows operating on different protocols, as well as the throughput and delay of an individual flow. In this section, first, the impact of the cwnd and ssthresh values are discussed for mpCUBIC and mpQUAD and then

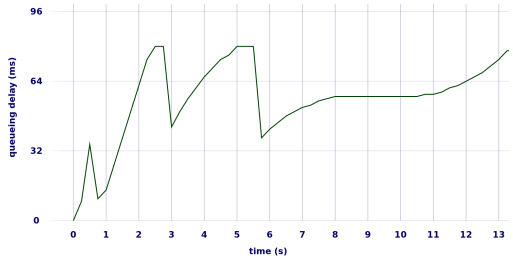

Fig. 5. Queuing Delay in mpCUBIC

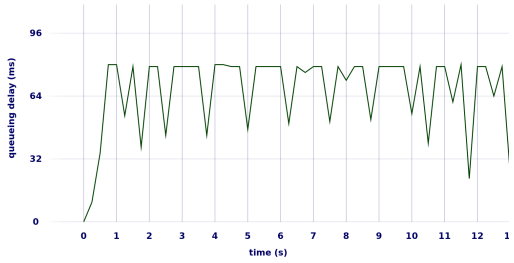

Fig. 6. Queuing Delay in mpQUAD

their performance is analyzed in terms of throughput and queueing delay.

A. Impact of Congestion Window Size and Slow Start Threshold

In mpQUAD, the cwnd and ssthresh are two critical parameters used to control the rate at which data is sent in the network. During the slow start phase of the algorithm, the cwnd and ssthresh are initialized to their default values. The cwnd is then increased linearly for each successful transmission until the ssthresh is reached. At this point, the algorithm enters the congestion avoidance phase, and the cwnd is increased more slowly, following a quad function. If congestion is detected, the cwnd is reduced, and the ssthresh is set to half the current cwnd. This reduction helps to prevent further congestion and to avoid TCP global synchronization. The relationship between the cwnd and ssthresh in mpQUAD is such that when the cwnd exceeds the ssthresh, the algorithm switches from slow start to congestion avoidance mode. In this mode, the cwnd is increased more slowly, and congestion is avoided by reducing the cwnd in response to congestion signals. When the cwnd is reduced due to congestion, the ssthresh is also reduced, which limits the growth of the cwnd in the future. This relationship between the cwnd and ssthresh helps to regulate the congestion control mechanism and avoid congestion collapse. Figure 1 and 2 depicts the nature of congestion window size and slow start threshold for cubic and mpQUAD respectively. The red traces indicates the the link capacity in the graphs. Congestion window grows more rapidly in cubic than mpQUAD, indicating network instability or congestion, leading to increased packet loss. The QUAD algorithm slows down and decrease the congestion window to prevent further congestion and improve overall network performance.

B. Throughput and Queueing-Delay

mpQUAD in fig4 has shown to have better throughput than mpCUBIC TCP in fig 3. This is because mpQUAD uses multiple paths and control parameters, which allows for better utilization of available network resources and faster recovery from congestion events. This results in more efficient use of network bandwidth and higher throughput. The relative performance of mpCUBIC and mpQUAD TCP indicate that,

under the dynamic nature of UAVs, mpQUAD outperforms mpCUBIC.

In terms of queuing delay performance mpCUBIC figure5 has shown to outperform mpQUAD TCP in figure6. mpQUAD uses multiple paths which allows for better utilization of available network resources and faster recovery from congestion events. This results in lower queuing delay, which is the time that a packet spends in a queue before being transmitted. On the other hand, although mpCUBIC TCP uses a multi path, but leads to suboptimal utilization of network resources and slower recovery from highly congested events. This results in higher queuing delay and can lead to longer response times and reduced throughput. Overall, mpQUAD's ability to efficiently use multiple paths and congestion control algorithms makes it a better choice for high-performance networking applications where low queuing delay is essential.

V. Conclusion

A congestion control algorithm for fast and long-distance FANET has been proposed, showing promising results. The algorithm has effectively managed network congestion and adaptability to different network scenarios in FANETs. With the time that has passed since the previous instance of congestion, the algorithm employs a quad window growth function. This real-time technique maintains TCP friendliness for short and long RTT routes by keeping the window expansion rate independent of RTT. The proposed mpQUAD is compared to mpCUBIC based on congestion window size, slow-start threshold, throughput, and queueing delay. To accommodate the multipath situation, mpQUAD is explored for multiple sub flows. The experimental analysis depicts the throughput enhancement and less queuing delays. Further, there is scope for implementing a machine learning model to learn how to change β according to a reward-based learning approach. This can result in dynamic and accurate congestion control, which would work to improve the throughput and RTT performance in an unstable and changing network environment.

Acknowledgment

No funds, grants, or other support was received.

References

[1] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (fanets): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[2] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on uavs for wireless networks: Applications, challenges, and open problems," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[3] M. Pakmehr, "Tcp congestion control characteristics and their impacts in qos over mobile broadband networks," Master's thesis, 2014.

[4] T. Kato, S. Haruyama, R. Yamamoto, and S. Ohzahata, "mpcubic: A cubic-like congestion control algorithm for multipath tcp," in *Trends and Innovations in Information Systems and Technologies: Volume 2 8*. Springer, 2020, pp. 306–317.

[5] P. Dong, J. Wang, J. Huang, H. Wang, and G. Min, "Performance enhancement of multipath tcp for wireless communications with multiple radio interfaces," *IEEE Transactions on Communications*, vol. 64, no. 8, pp. 3456–3466, 2016.

[6] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath tcp: Analysis, design, and implementation," *IEEE/ACM Transactions on networking*, vol. 24, no. 1, pp. 596–609, 2014.

[7] M. A. Al-Absi, A. A. Al-Absi, M. Sain, and H. Lee, "Moving ad hoc networks—a comparative study," *Sustainability*, vol. 13, no. 11, p. 6187, 2021.

[8] T. A. Le, R. Haw, C. S. Hong, and S. Lee, "A multipath cubic tcp congestion control with multipath fast recovery over high bandwidth-delay product networks," *IEICE transactions on communications*, vol. 95, no. 7, pp. 2232–2244, 2012.

[9] H. M. Hijawi and M. M. Hamarsheh, "Performance analysis of multipath tcp network," *Int. J. Comput. Netw. Commun*, vol. 8, no. 2, pp. 145–157, 2016.

[10] K. Nguyen, M. Golam Kibria, K. Ishizu, F. Kojima, and H. Sekiya, "An approach to reinforce multipath tcp with path-aware information," *Sensors*, vol. 19, no. 3, p. 476, 2019.

[11] T. Shreedhar, D. Zeynali, O. Gasser, N. Mohan, and J. Ott, "A longitudinal view at the adoption of multipath tcp," *arXiv preprint arXiv:2205.12138*, 2022.

[12] B. Sharma, G. Srivastava, and J. C.-W. Lin, "A bidirectional congestion control transport protocol for the internet of drones," *Computer Communications*, vol. 153, pp. 102–116, 2020.

[13] S. R. Pokhrel, J. Jin, and H. Le Vu, "Mobility-aware multipath communication for unmanned aerial surveillance systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6088–6098, 2019.

[14] R. Netravali, A. Sivaraman, K. Winstein, S. Das, A. Goyal, and H. Balakrishnan, "Mahimahi: A lightweight toolkit for reproducible web measurement," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 129–130, 2014.

[15] K. Miller and L. W. Hsiao, "Tcptuner: congestion control your way," *arXiv preprint arXiv:1605.01987*, 2016.