applied sciences

MDPI

Article

Adaptive TCP Transmission Adjustment for UAV Network Infrastructure

Joon Yeop Lee, Woonghee Lee, Hyunsoon Kim and Hwangnam Kim *

School of Electrical Engineering, Korea University, Seoul 136-713, Korea; charon7@korea.ac.kr (J.Y.L.); tgorevenge@korea.ac.kr (W.L.); gustns2010@korea.ac.kr (H.K.)
* Correspondence: hnkim@korea.ac.kr; Tel.: +82-2-3290-4821

Received: 27 December 2019; Accepted: 5 February 2020; Published: 9 February 2020

Abstract: A UAV network composed of multiple UAVs allows a wide operating radius and various tasks to be performed. However, a UAV network mostly suffers from high probability of transmission failure due to interference or mobility. Also, nodes connected to the network often experience connection loss and segment loss caused by frequent node mobility and routing update. Since congestion is not the only cause of data loss in UAV networks, the TCP congestion control should not be run if there is a possibility of transient link instability unless a reduction in transmission speed is required. For this reason, we propose an algorithm to improve the transmission performance of UAV network through TCP with Slow-Start threshold (Ssthresh) value adjusted. The adjustment algorithm is called Adaptive Ssthresh Reviser for flying Ad hoc Network (*ASRAN*) that quickly restores unnecessary decrease of transmission speed in UAV network.

Keywords: transmission control protocol; UAV network; ad hoc; wireless network

1. Introduction

With the development of wireless network devices, it has become possible to collect a wide range of information and form flexible topologies through various small network devices. However, due to the wide range of activities or mobility, it is difficult for all equipment to be directly connected to the Internet's backbone. In environments where it is difficult to connect directly to the Internet, many devices can connect to the Internet with a wireless ad hoc network. There are various techniques based on wireless multi-hop environments; for example, Unmanned aerial vehicle (UAV)-based network system [1–3], multiple Internet of Things (IoT) devices [4,5], 5G network with small mesh networks for fast data transmission speed [6], VANET technology for vehicles [7,8], and MANET technology [9,10]. However, various wireless multi-hop networks including UAV network (e.g., flying ad hoc network) often suffers from segment loss [11]. UAVs have an unstable network because they are difficult to load high-performance device due to limited payload size and moves around frequently to carry out various tasks. As a result, UAV networks often suffer from segment losses due to unstable networks. In this situation, since the formation information and the control message necessary for UAV must be received, the transmission control protocol (TCP), which is a reliable communication protocol, is used to recover the segment loss [12]. Based on this observation, this paper suggests an improvement technique of TCP to enhance the performance of the overall UAV network.

If the segment loss occurs, TCP considers the network to be congested and decreases the congestion window (CWND) to reduce the transmission speed. The reason for reducing the transmission speed is that segment losses are caused by heavy traffic exceeding link capacity in most of wired environment. However, in wireless multi-hop environments such as UAV networks, segment losses are caused by not only buffer overflow but also transient link instability. For example, if power

drained UAV is dropped from the network, a routing change occurs. In this case, data transmission is disconnected until the transmission path is reconstructed. Also, the mobility of UAV in most wireless multi-hop networks causes connection loss, resulting in time delay to find an alternative path to handle packet delivery. The delay misleads the end nodes to understand the phenomenon as a buffer overflow due to heavy traffic and will make the sender reduce traffic. Indeed, we frequently experienced the aforementioned phenomena in UAV network performance experiments based on ad hoc networks. However, aforementioned transient link interference should not be a reason to reduce the transmission throughput of TCP link because a path that recovered from the transient problems may have or tends to have enough capacity to support existing transmission throughput. Therefore, it is necessary to adjust TCP, which is characterized by a decrease in throughput when segment loss occurs.

End-to-end controls including TCP cannot distinguish between whether segment loss occurs due to excessive traffic or transient link instability [13]. TCP therefore should have an additional solution for congestion incurred by reasons other than buffer overflow. To propose the solution, this paper suggests Adaptive Ssthresh Reviser for flying Ad hoc Network (*ASRAN*) that is designed to quickly recover the reduced throughput caused by transient link instability, which is suitable for UAV network. The contribution of *ASRAN* is listed as follows:

- The *ASRAN* modifies the TCP to be suitable for UAV network.
- The *ASRAN* deals with the transient link instability, separated from heavy traffic, which is a factor of reduced transmission performance in a UAV network.
- In transient link instability situation, *ASRAN* quickly recovers unnecessarily reduced throughput.

This paper is organized as follows. Section 2 shows related work on enhancing performance and analyzing TCP in wireless network or ad hoc network. Details of *ASRAN* are described in Section 3 and Section 4 proposes a mathematical model to predict the effect of *ASRAN*. To prove the performance of *ASRAN*, experimental setup and results are shown in Section 5. At the end, Section 6 concludes the paper and lists several future works to be done.

2. Related Work

This section lists several related work focusing on TCP performance enhancement in ad hoc network. There is some research about modifying TCP for ad hoc networks or wireless networks. Khurshid et al. [14] suggested new congestion control algorithm for better performance in wireless networks. They argued that the major problem of random segment loss in the wireless networks is the bit error. Therefore, the arrival of duplicate acknowledgement or even the retransmission timeout is not enough to denote actual reason of congestion. They provided fast transmission solution by adjusting CWND size according to count of timeouts, 3 dup ACKs and timeout interval of the retransmission timer. However, this algorithm was only implemented for TCP-NewReno and was not considered about any other TCP protocols. Zhang et al. [15] provided a solution named AFStart which acts like a slow-start algorithm for the network environment with long-distance and high end-to-end latency. They proposed a solution that finds available bandwidth to set the *ssthresh* and adjust the congestion window. To estimate the available bandwidth, they used ImTCP [16] which derives the available bandwidth based on round trip time (RTT). If there is enough bandwidth to be used, this algorithm ramps up CWND aggressively which is faster than traditional slow-start algorithm for fast transmission. Jude et al. [17] said that TCP performs poorly under Vehicular Ad hoc Network (VANET) due to its high mobility and wireless errors such as transmission path change (fading) and interferences that lead to unpredicted packet drop. Therefore, they analyzed various TCP such as Vegas, Compound, Westwood, NewReno, BIC and CUBIC under vehicular environment for variable speed and traffic. They argued that TCP-CUBIC is the best algorithm in the aspect of throughput and TCP-Vegas has strong aspects of delay control. Although improvement of the algorithm was not included, they showed that various aspects of congestion avoidance should be considered in the wireless environment with mobility. Gururaj et al. [18] suggested High speed

TCP to improve the performance of TCP connections with large congestion windows for Ad hoc environment with no guaranteed bandwidth and node mobility. This algorithm was based on the relation between an average congestion window and packet drop rate. The characteristic of HSTCP algorithm is increasing and decreasing CWND abruptly to improve the retransmission time and to avoid congestion in dynamic conditions where node mobility is completely random. Using CWND increase and decrease factor seems flexible and suitable for an unstable ad hoc network. However, they implemented in experiment with assumed value of CWND increase factor and decrease factor with lack of explaining. Also, their evaluation study was performed with fixed CWND increase and decrease factors, which cannot satisfy the various network environments.

Also, there is some research about TCP-CUBIC for an ad hoc network. Tomita et al. [19] presented an algorithm that solves a problem of significant packet losses due to small buffer size in Linux TCP. This problem especially occurs when small memory devices (such as digital cameras and digital camcorders) upload photos or movies via long distance network. The proposed solution analyzed the shape of the throughput curve of TCP-CUBIC in order to decide whether or not to increase the sender buffer. Bao et al. [20] proposed a Markovian model to determine the steady state throughput of TCP-CUBIC in wireless environment. The proposed model considered both congestion loss and random packet loss due to fading. Also, they proposed a solution which increases the throughput performance of TCP-CUBIC by moderately increasing the window growth factor and the multiplicative decrease factor. However, they presented normalized throughput only, but they did not show the total throughput or comparison with original TCP-CUBIC. Evaluation of normalized throughput showed that throughput is improved by the reduced packet loss. However, reduced packet loss can be achieved with lowered throughput, but it cannot be easily recognized in evaluating normalized throughput. Therefore, this evaluation cannot suggest that throughput is improved. Pilimon et al. [21] showed a simulation-based analysis of the robustness of high-speed TCP-CUBIC connections when random packet losses increase in a network with large bandwidth-delay product and different TCP connections. They showed strong point of TCP-CUBIC compared with other TCP algorithm under high packet loss environment.

3. *ASRAN*

This section describes the development motivation and design of the *ASRAN*. Section 3.1 describes motivation and Section 3.2 describes the detailed design and effects of the *ASRAN*. Then, Section 3.3 describes how *ASRAN* distinguishes transient link instability from buffer overflow.

3.1. Motivation

The UAV network is flexible and highly scalable. However, UAV network lacks network transmission stability. We experienced network instability while experimenting with UAV network. We constructed a flying ad hoc network based on multiple UAVs and experimented to transmit data. The throughput measurement data is set to be transmitted at 1000 kbps, and no additional traffic was generated in the UAV network. The cumulative distribution function (CDF) of the measured throughput in this environment is shown in Figure 1. As shown in Figure 1, the rate at which the network maintains the maximum transmission rate is less than 50% of total transmission time. In addition, the interval measured at 0 Kbps was 19% of the total transmission time because data transmission was impossible. In other words, transmission via UAV network has also been temporarily disconnected.

As shown in our previous experiments, transient link instability occurs frequently in UAV networks. In other words, in a UAV network environment, data loss can occur frequently without buffer overflow due to heavy traffic. To cope with UAV network environment, we tried to improve the throughput of UAV network by improving TCP. Since the original TCP assumes that data loss is caused by buffer overflow due to heavy traffic, TCP reduces throughput to reduce network traffic and avoid congestion. However, since the transient link instability that occurs in a UAV network is not

caused by heavy traffic, it is unnecessary to reduce the throughput. Therefore, we propose an *ASRAN* that improves TCP to UAV network. *ASRAN* operates by distinguishing transient link instability from buffer overflow due to heavy traffic. With this distinction, *ASRAN* quickly recover throughput in case of transient link instability and acts as on original TCP to avoid congestion in case of heavy traffic.

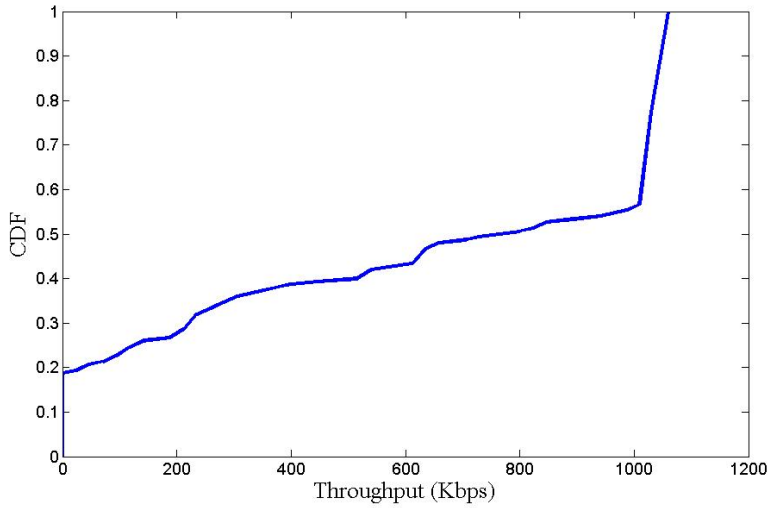


Figure 1. Throughput measurement result of multi-hop UAV network.

3.2. Design for ASRAN

The main concept of *ASRAN* is adjusting TCP parameter of *ssthresh* to decide CWND increase rate according to the main reason of congestion. If the congestion is due to heavy traffic, CWND is reduced, and then it should increase slowly, like original TCP, not to incur frequent congestion later on. On the other hand, CWND should increase fast in order to fully use the wireless link which was momentarily unstable. For this purpose, *ASRAN* adjusts *ssthresh* value as shown in Algorithm 1.

Algorithm 1 Modified *ssthresh* calculator.

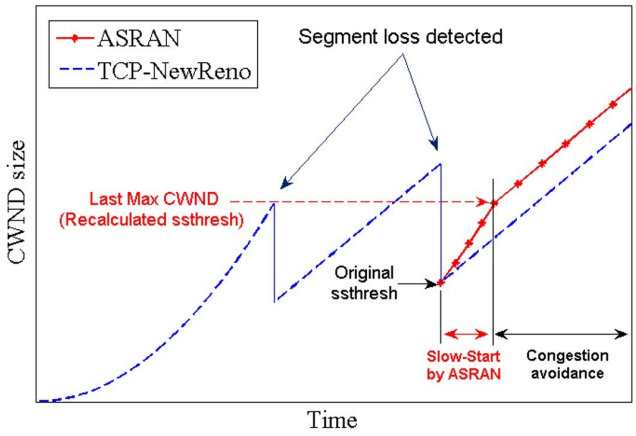
```
1: while segment_loss = true do
2:     function ssthresh_calculator(current_max_cwnd)
3:         original_ssthresh = current_max_cwnd/2
4:         recalculated_ssthresh = max(original_ssthresh, last_max_cwnd)
5:         last_max_cwnd = current_max_cwnd
6:         return recalculated_ssthresh
7:     end function
8: end while
```

When a segment loss occurs, the data sender receives duplicated ACKs (Retransmission timeout may also occur, but this means that a more serious problem occurred with the network, so *ASRAN* behaves the same as before). In this case, a typical TCP (TCP-NewReno) reduces the CWND size in half. Also, the *ssthresh* value is updated to half of the CWND size. If CWND size is larger than *ssthresh* value, TCP switches from Slow-Start phase to congestion avoidance phase. In other words, after segment loss, TCP automatically activates congestion avoidance phase. Using the TCP feature of switching phase according to *ssthresh*, the *ASRAN* adjusts the *ssthresh* value to induce Slow-Start rather than congestion avoidance when a segment loss occurs due to transient link failure. To choose appropriate *ssthresh* value, *ASRAN* uses the latest CWND size prior to the previous segment loss event, named *last_max_cwnd*. Because *last_max_cwnd* is the CWND size of successful transmission until the segment loss occurs, *ASRAN* exploits the presumption that *last_max_cwnd* reflects the available capacity of the network for transmission. When a segment loss occurs, *last_max_cwnd* is compared

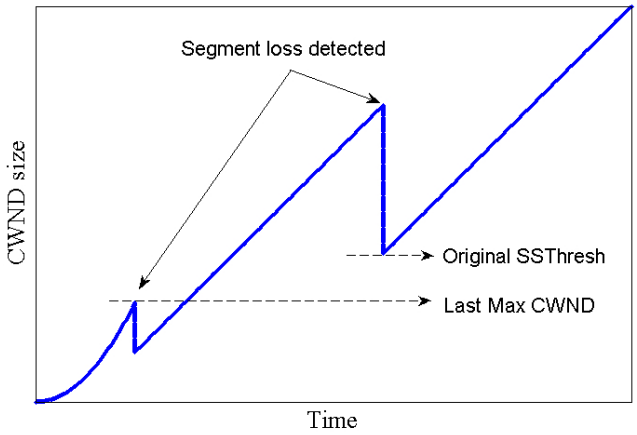
to the originally calculated *ssthresh* value, *original_ssthresh*. The bigger value will be the new *ssthresh* value, named *recalculated_ssthresh*. After that, *last_max_cwnd* will be updated to the latest CWND size prior to the current segment loss for *ssthresh* decision of future segment loss. Also, as shown in Algorithm 1, every segment loss event updates the *last_max_cwnd* value which means that *ASRAN* always reflects fluctuation of network environment.

3.3. How to Distinguish Transient Link Instability from Buffer Overflow

The TCP cannot distinguish whether the segment loss occurs due to transient link instability or buffer overflow. Unlike original TCP, the *ASRAN* distinguishes between transient link instability and buffer overflow based on the *last_max_cwnd* presented in Section 3.2. *Last_max_cwnd* represents the most recently available transmission capacity of network. Therefore, if the size of the CWND is smaller than *last_max_cwnd* after the segment loss occurs, the *ASRAN* determines that transient link instability occurs in the network and restores the throughput quickly. If the size of the CWND is larger than *last_max_cwnd* after the segment loss occurs, the *ASRAN* determines that the buffer overflow is caused by the data transmission exceeding the transmission capacity. Figure 2a,b shows CWND fluctuation modeling of TCP using *ASRAN*. CWND fluctuation modeling follows the TCP-NewReno algorithm, and each figure shows how *ASRAN* works with CWND fluctuations when transient link instability and buffer overflow occur. Detailed description of each situation is given in the following subsections.



(**a**) CWND fluctuation in *ASRAN* activated scenario



(**b**) CWND fluctuation in buffer overflow scenario with *ASRAN*

Figure 2. CWND fluctuation of ASRAN in transient link instability and buffer overflow scenario.

3.3.1. Dealing with Transient Link Instability

Figure 2a shows the effectiveness of the *ASRAN* with CWND size fluctuations between original TCP-NewReno and *ASRAN* in transient link instability situation. The initial rise of CWND is the result of TCP slow-start. In addition, it takes the first segment loss which results in halving CWND value. At this moment, original TCP-NewReno updates its current ssthresh, *original_ssthresh*, as the halved CWND in order to enter congestion avoidance phase. For the first time, *ASRAN* also enters congestion avoidance phase because *last_max_cwnd* is not updated before. Assuming *last_max_cwnd* is 0 at this moment, maximum *ssthresh* value becomes *original_ssthresh*, resulting in congestion avoidance phase after all. Then *last_max_cwnd* is updated to CWND size before halved, which will be used for the next segment loss event. At the second segment loss event in Figure 2a, updated *last_max_cwnd* is larger than *original_ssthresh* value. Because the size of CWND is less than the recalculated *ssthresh* value(*recalculated_ssthresh* value in Algorithm 1), TCP flow quickly recovers CWND by using the slow-start algorithm rather than congestion avoidance. After the second segment loss, CWND rising curve of *ASRAN* seems like a linear curve. However, it is a quadratic curve that is increased by slow-start. This is due to the method of CWND size increases in Slow-Start algorithm. Slow-Start algorithm increases CWND size per ACK received. This method results in a quadratic increase in CWND size, unlike congestion avoidance which CWND size increases linearly. Since slow-start phase increases CWND size quadratically, the transmission speed of *ASRAN* recovers rapidly, resulting in better performance than original TCP-NewReno.

3.3.2. Dealing with Buffer Overflow

If CWND always rises rapidly in the slow-start phase, wireless link can frequently experience buffer overflow due to excessively large amount of data stream generated by overgrown CWND size. If *ASRAN* uses *current_cwnd* to calculate *recalculated_ssthresh*, TCP will always increase CWND using the Slow-Start algorithm after segment loss even in the case of congestion due to buffer overflow. In order to prevent such the case, *ASRAN* uses the *last_max_cwnd*. If *last_max_cwnd* is larger than currently reduced CWND size after segment loss, the *ASRAN* recovers the CWND size using slow-start until it is the same size as *last_max_cwnd*. If not, the *ASRAN* uses the *original_ssthresh* as the *recalculated_ssthresh*, so the TCP flow executes the congestion avoidance phase as usual TCP congestion control does. Figure 2b shows such the scenario that lets *ASRAN* perform congestion avoidance. In Figure 2b, on the second CWND drop, the saved *last_max_cwnd* of the previous segment loss is smaller than *original_ssthresh*. In this case, the value of the *original_ssthresh* will be the *recalculated_ssthresh* value. Because *recalculated_ssthresh* is smaller than reduced CWND size, congestion avoidance algorithm is executed.

4. *ASRAN* throughput Model

This section shows the throughput model of *ASRAN*. *ASRAN* modeling has two purposes. The first purpose is to verify the result of real experiment. Under environment of wireless multi-hop network, packet arrival period and segment loss event are unstable and aperiodic. Therefore, it is difficult to figure out whether experiment result is affected by *ASRAN* or just an accidental result caused by unstable environment. To decide whether the experiment result is credible or not, modeling the algorithm and calculating the expected throughput gain of *ASRAN* are necessary. The second purpose is to prove mathematically that *ASRAN* outperforms TCP-CUBIC. Please note that TCP-CUBIC is faster than TCP-NewReno and TCP-Reno to recover communication when segment loss occurred, which is shown in Section 5. TCP-CUBIC is developed in consideration of the wireless communication environment with both high bandwidth and high latency [22]. The congestion avoidance of TCP-CUBIC increases CWND size as cubic [23]. We confirmed through modeling that the *ASRAN* can show better throughput than the TCP-CUBIC.

As shown in Figure 3, CWND size of the *ASRAN* reaches to the *last_max_cwnd* faster than TCP-CUBIC. After that, even if *ASRAN* increases with cubic curve as the same as TCP-CUBIC, difference of CWND size between *ASRAN* and TCP-CUBIC is getting larger. Because of CWND size of *ASRAN* reaches *last_max_cwnd* faster than TCP-CUBIC, *ASRAN* obtains more time to increase CWND with a cubic curve, compared to TCP-CUBIC. In the subsequent modeling, these effects are proved by calculating total throughput gain of *ASRAN*. To obtain the expected throughput gain of *ASRAN*, simplified modeling was used to calculate the throughput ratio between throughput of TCP-CUBIC and throughput of *ASRAN*.

Total throughput can be calculated by integrating graph of CWND size fluctuation shown in Figure 3. Predicted total throughput of *ASRAN* in Figure 3 will be the following equation:

$$ASRAN\_total\_throughput = \int_0^K (x+i)^2 dx + \int_K^T \{(x-k)^3 + last\_max\_CWND\} dx. \quad (1)$$

In Equation (1), x indicates the time variable, i indicates the initial CWND size, T indicates the time until the next segment loss occurs, and K represents the time that CWND reaches the *last_max_cwnd*. In *ASRAN*, before x reaches K, CWND will be increased in Slow-Start phase. After that, CWND is increased by the congestion avoidance algorithm of TCP-CUBIC in probing phase which is cubic curve. For calculating K value, CWND size will be recovered with quadratic curve of slow-start until CWND reaches the *last_max_cwnd*. At this moment, x value will be K value. Part of the quadratic CWND increase equation is shown in the front part of Equation (1). Calculating K value is:

$$\begin{aligned} last\_max\_CWND &= (K+i)^2 \\ K &= \sqrt{last\_max\_CWND} - i. \end{aligned} \quad (2)$$

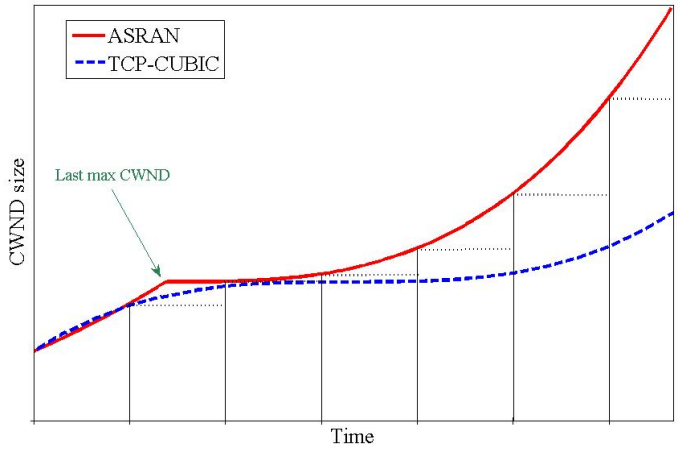


Figure 3. Discrete time division of *ASRAN* and TCP-CUBIC.

Since CWND goes up every time an ACK arrives, the rise of CWND is discrete event. Therefore, the total throughput should be calculated by summing CWND value at each discrete time, instead of calculating predicted total throughput obtained by integrating graph of modeling. Figure 3 shows discrete time at regular interval. Sum of each CWND size at each discrete x value represents the total throughput.

The gain of *ASRAN* compared to TCP-CUBIC ratio value can be calculated by dividing total throughput obtained from *ASRAN* by the total throughput of TCP-CUBIC. To show the difference of throughput between *ASRAN* and TCP-CUBIC, we set the example model to calculate the gain value. $i = 1$, $last\_max\_cwnd = 2$, $T = 2.5$ were arbitrarily assigned in example model, and Figure 3 is drawn

based on the example model. Based on example model, the throughput gain of *ASRAN* is calculated as follows:

$$\begin{aligned} Gain\_of\_ASRAN &= \frac{ASRAN\_total\_throughput}{CUBIC\_total\_throughput} \\ &= \frac{\sum y_{ASRAN}}{\sum y_{CUBIC}} = \frac{101.162}{62.375} = 1.622. \end{aligned} \tag{3}$$

In the general case, to see the CWND fluctuation of TCP-CUBIC transmission, congestion avoidance phase is shown continuously over the whole transmission phase regardless of throughput increase phase or throughput decrease phase [22]. Considering unstable network performance of wireless multi-hop network, congestion avoidance occurs in almost every time and *ASRAN* can be activated in whole transmission time. Therefore, in the recovery phase right after segment loss, the throughput of *ASRAN* can be expected 1.622 times larger than that of TCP-CUBIC. In the evaluation section, the real experiment result will show similar value to the result calculated by the proportion from modeling.

5. Performance Evaluation

This section describes experiments and results to demonstrate the performance of *ASRAN*. Empirical experiment to verify the effectiveness of *ASRAN* is described in the Section 5.1. In addition, the result of simulation experiment to increase the number of UAVs and add UAV mobility is written in Section 5.2.

5.1. Empirical Study

For empirical experiment, Section 5.1.1 describes details of the experimental setup. In order to emphasize the strength of the proposed algorithm on any condition, we conducted experiments for both transient link instability case and buffer overflow case. The details are described in Sections 5.1.2 and 5.1.3.

5.1.1. Experimental Setup

Figure 4 shows the topology of the empirical experiment. To construct the network with UAV, an Odroid device was mounted on every UAVs. Odroid device is a single board computer with heterogeneous multi-processing, and support Ubuntu OS [24]. Imagining that Operating_UAVs give feedback to Ground Control System (GCS) through an access network, Routing_UAV 1 acted like an Access Point (AP) to create a subnet which enables Operating_UAVs to access the network through Routing_UAV 1. The UAV network, which includes the Routing_UAVs, communicates in the 5 GHz bandwidth so as not to affect the network communication environment of the Operating_UAVs using the 2.4 GHz bandwidth Wi-Fi. In addition, the UAV network is based on the commonly used Destination-Sequenced Distance Vector (DSDV) routing protocol [25]. All Operating_UAVs act as TCP senders through Wi-Fi in the 2.4 GHz bandwidth. Operating_UAV 1 transmits data through default algorithm: TCP-CUBIC, and Operating_UAV 2 transmits data through TCP equipped with *ASRAN*. Since TCP related implementation is held on the device kernel, *ASRAN* was implemented into the kernel of Ubuntu. In the kernel layer, there is a function to calculate the *ssthresh* of TCP after segment loss. *ASRAN* compares the previously stored *last_max_CWND* with newly calculated *ssthresh* and returns a larger value. To form a wireless multi-hop network, two Routing_UAVs acted like wireless intermediary nodes that forward data from Operating_UAVs to GCS. Role of the GCS is a receiver of TCP data transmitted by the Operating_UAVs in the experiment configuration. GCS is a laptop with Linux OS which acted like a gateway to an ad hoc network and assigned IP addresses to Routing_UAV to form a UAV network topology.

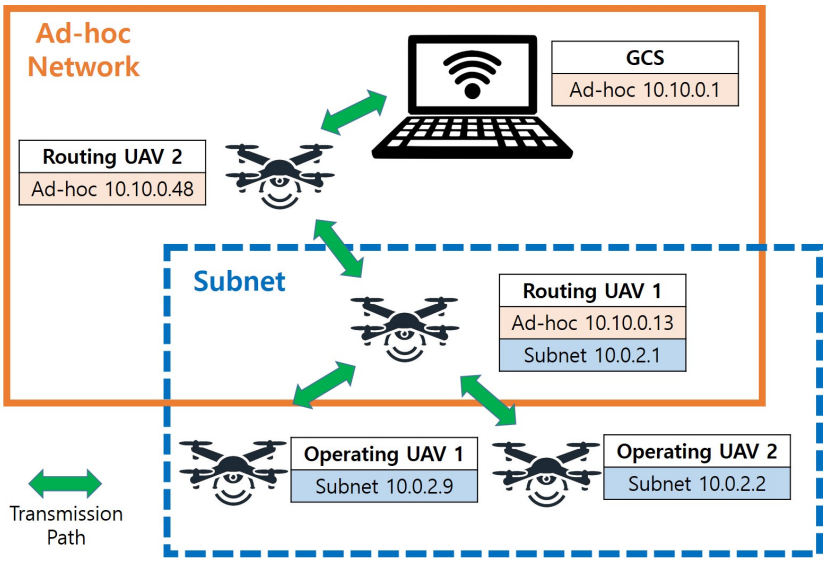


Figure 4. Network topology for experiment.

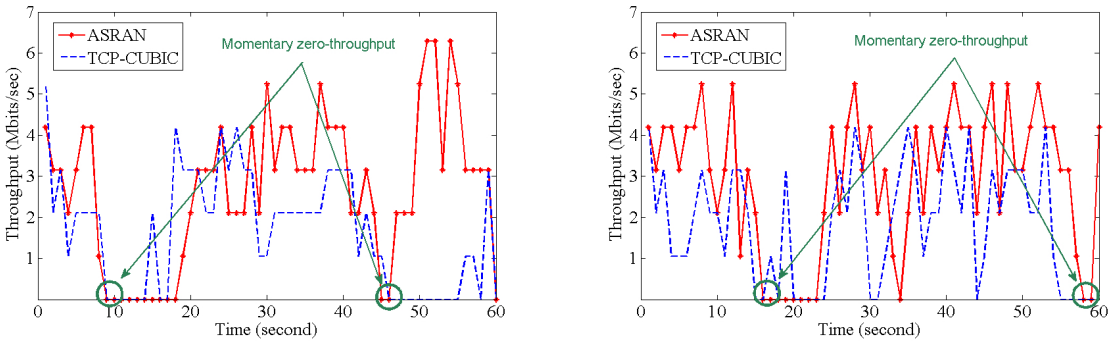
With this multi-hop environment, Operating_UAVs are connected to Routing_UAV 1 with Wi-Fi. As aforementioned, Operating_UAV 2 uses original kernel, and Operating_UAV 1 uses modified kernel on which *ASRAN* is implemented. After establishing connection between Operating_UAVs and Routing_UAV 1, Operating_UAVs simultaneously transmit TCP data to GCS.

5.1.2. In Case of Transient Link Instability Only

In this experiment, by transmitting low throughput of TCP flows with only two Operating_UAVs, it prevents buffer overflow caused by heavy traffic that make segment loss events occur by transient link instability only. For transmission, Iperf application was used to create TCP flows and data log for throughput measurement. Two Operating_UAVs simultaneously sent data to GCS via Routing_UAV 1. We evaluated TCP-CUBIC and *ASRAN* at the same time to obtain throughput under fair environment. For performance comparison, the throughput was measured every second and the average throughput of all experimental data for 600 s was obtained. Table 1 compares the average throughput of TCP-CUBIC, *ASRAN* and theoretical model (shown in Section 4). As shown at the table, the result of *ASRAN* shows higher throughput compared to TCP-CUBIC. In addition, *ASRAN* has similar results as expected throughput (1.028 Mbits/s) calculated through theoretical model. The high standard deviation shows that the variation of throughput of the experiment was severe. This result shows that the throughput of the UAV network can be very unstable. In this experimental result, *ASRAN* shows a higher standard deviation than the TCP-CUBIC. This is because *ASRAN* aggressively increases the throughput and is likely to reach the bandwidth limit. Since the *ASRAN* is faster than the existing TCP, we analyzed how *ASRAN* violated fairness through Jain's fairness index [26]. The Jain's fairness index was 0.925, which indicates that ASRAN does not significantly impair fairness. Figure 5 shows the typical throughput fluctuation of experiments. It is possible to observe the point where the transmission becomes impossible (zero throughput) due to the transient link instability of wireless multi-hop network. In most cases, the throughput of experiments were unstable. In this environment, most of experiment time, *ASRAN* shows the higher throughput compared to original TCP-CUBIC. This is because *ASRAN* recovers CWND more quickly most of time by using the Slow-Start algorithm when segment loss occurs.

Table 1. Comparison of experiment when transient link instability appears.

	TCP-CUBIC	*ASRAN*
Average Throughput (Mbits/s)	0.634	1.140
Standard Deviation	0.945	1.101



(**a**) Test Set 1 (**b**) Test Set 2

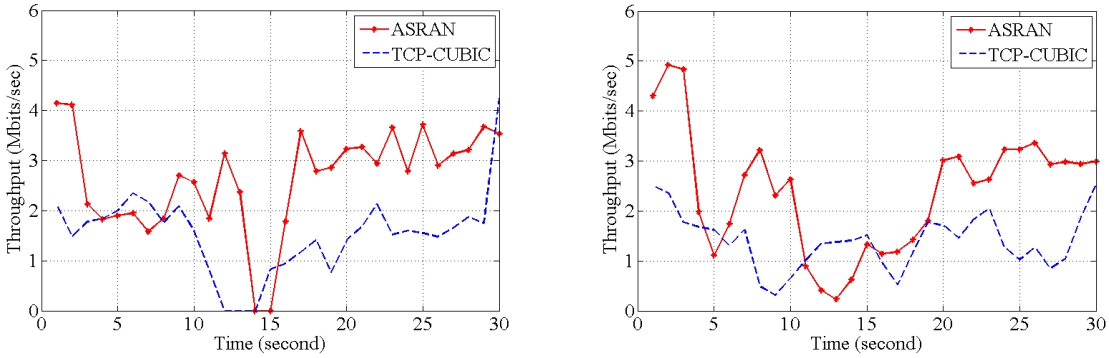
Figure 5. Throughput comparison of simultaneous experiment when transient link instability appears.

5.1.3. In Case of Heavy Traffic

UAV networks often have transient link instability, but heavy traffic can also occur. In this section, we assumed that a heavy traffic is generated by a particular service user. To make segment losses incurred by heavy traffic, the third UAV (Operating_UAV 3) which transmitted heavy TCP traffic to GCS via Routing_UAV 1 was added. While Operating_UAV 1, 2, and 3 co-uses Routing_UAV 1, Routing_UAV 1 suffers from frequent buffer overflow. In this experiment, throughput of Operating_UAV 1 and 2 were measured for 600 s long, and the average throughput of measured throughput per second was obtained. Table 2 shows experimental results. Similar to the previous experiment, *ASRAN* still shows better performance than TCP-CUBIC even though heavy background traffic appears in the network. However, the average throughput of heavy traffic experiment was lower than that of environment where only transient link instability appears, which means the network transmission environment became worse than the prior experiment. Also, the deterioration of the overall transmission environment caused real *ASRAN* throughput to be lower than expected throughput (0.900 Mbits/s) calculated through theoretical model. Since the *ASRAN* only operates during transient link instability, the experimental results show that *ASRAN* did not aggressively increase throughput in heavy traffic environment. The Jain's fairness index was 0.977, indicating that the fairness was better than in the previous experiment. Figure 6 shows that transient zero throughput and throughput fluctuations occurred in heavy traffic scenarios. Therefore, *ASRAN* enhances the average throughput, whereas throughput fluctuation graphs show the mixture of better throughput than and similar throughput to TCP-CUBIC.

Table 2. Comparison of experiment in heavy traffic environment.

	TCP-CUBIC	*ASRAN*
Average Throughput (Mbits/s)	0.555	0.757
Standard Deviation	0.860	0.984



(**a**) Test Set 1 (**b**) Test Set 2

Figure 6. Throughput comparison of simultaneous experiment in heavy traffic environment.

5.2. Simulation Study

Simulation experiments were conducted to increase the number of UAVs constituting the UAV network and to implement the scenarios in which the network topology changes through mobility. In the simulation experiments, we verified the performance of *ASRAN* through network simulator NS3. Figure 7 shows the simulation topology. The Operating_UAV transmits data to the GCS through the UAV network. The UAV network constructed the transmission path based on the DSDV routing algorithm. During data transmission, Routing_UAV 2 and Routing_UAV 3 exchanged their position. In this environment, we compared the throughput of *ASRAN* and the most commonly used TCP algorithm, TCP-NewReno. The average transmission rates measured in this scenario are shown in Table 3. As shown in Table 3, the *ASRAN* has a higher throughput than the TCP-NewReno. Figure 8 shows throughput fluctuations of typical simulation experiments. Especially in Figure 8b, *ASRAN* and TCP-NewReno show the same disconnect time. This is because both experiments use the same routing algorithm and the routing algorithm changes the transmission path at approximately the same time after detecting the UAV2 mobility. After the transmission is disconnected, the transmission path was recovered via the UAV 4 and the transmission was resumed. Even in this environment, *ASRAN* showed higher average throughput. Also, Jain's fairness index was 0.996, indicating that the *ASRAN* generally maintained fairness. As a result, *ASRAN* has overall better performance than TCP-NewReno, even in environments where data is transmitted over unstable UAV networks due to mobility.

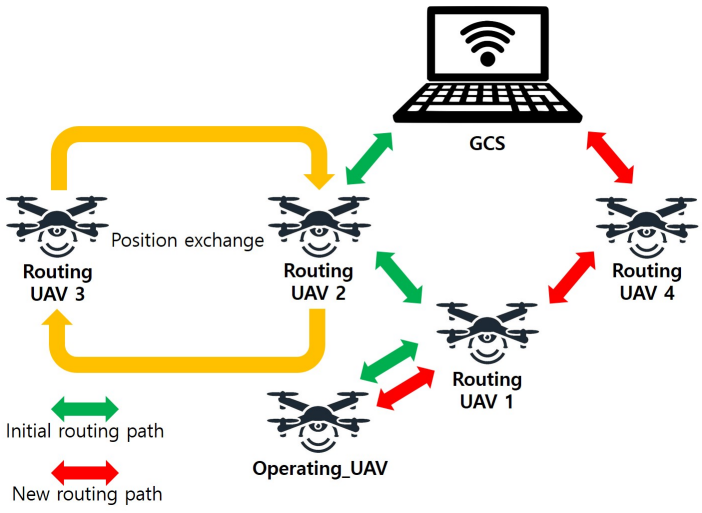
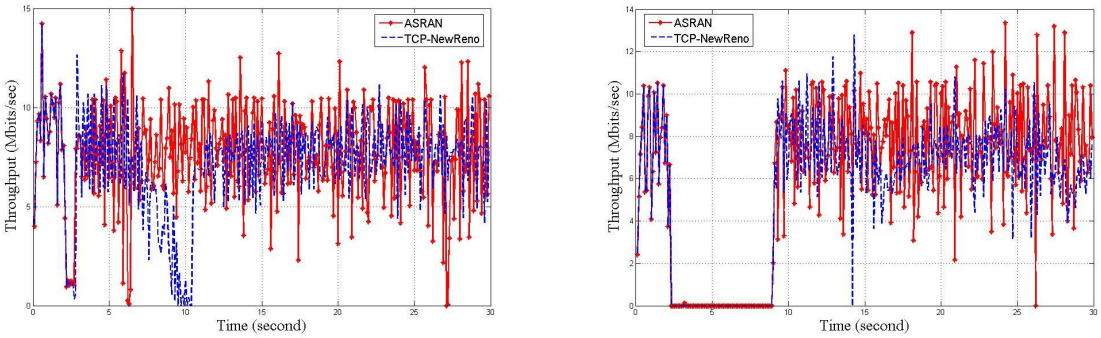


Figure 7. Simulation topology.

Table 3. Average throughput comparison of simulation experiment.

	TCP-NewReno	***ASRAN***
Average Throughput (Mbits/s)	5.805	6.582
Standard Deviation	3.692	4.094



(**a**) Test Set 1 (**b**) Test Set 2

Figure 8. Throughput fluctuation of simulation experiments.

6. Conclusions

This paper presents the *ASRAN* for TCP that is suitable for UAV network environment. When a network is deployed through multiple UAVs, transient link instability can occur due to mobility and transient link failure, which in turn triggers frequent changes in network topology. Since transient link instability is not caused by heavy traffic, there is no need to reduce the transmission speed, like the original TCP. In order to cope with an unstable network environment, *ASRAN* adjusts the TCP parameter to improve the network throughput by recovering unnecessarily reduced throughput when transient instability occurs in the UAV network. As a result, if the *ASRAN* is applied to UAV, the transmission speed of the overall UAV network can be improved. In addition, since *ASRAN* uses the Slow-Start algorithm as the basic technique of TCP to increase the throughput, even if there is data loss or buffer overflow, *ASRAN* can operate in similar way to the original TCP.

As future work, we will apply *ASRAN* to multi-path TCP (MPTCP). Our goal is to improve the throughput of each subsocket through *ASRAN*, thereby improving the overall throughput of MPTCP. In addition, we plan to enhance the throughput model of *ASRAN* when it is applied to MPTCP.

Author Contributions: Conceptualization, J.Y.L.; methodology, J.Y.L.; software, J.Y.L. and W.L.; validation, J.Y.L., H.K. (Hyunsoon Kim) and H.K. (Hwangnam Kim); formal analysis, J.Y.L.; investigation, J.Y.L.; resources, H.K. (Hwangnam Kim); data curation, J.Y.L.; writing—original draft preparation, J.Y.L.; writing—review and editing, W.L., H.K. (Hyunsoon Kim) and H.K. (Hwangnam Kim); visualization, J.Y.L.; supervision, H.K. (Hwangnam Kim); project administration, H.K. (Hwangnam Kim); funding acquisition, H.K. (Hwangnam Kim). All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by "Human Resources program in Energy Technology" of the Korea Institute of Energy Technology Evaluation and Planning(KETEP) granted financial resource from the Ministry of Trade, Industry & Energy, Republic of Korea(No. 20174030201820).

Acknowledgments: Preliminary version of this paper appeared at Local Computer Networks (LCN), 2016 IEEE International Conference on. IEEE, 2016 [27].

Conflicts of Interest: The authors declare that they have no conflicts of interest.

References

1. Qi, F.; Zhu, X.; Mang, G.; Kadoch, M.; Li, W. UAV Network and IoT in the Sky for Future Smart Cities. *IEEE Netw.* **2019**, *33*, 96–101. [CrossRef]
2. Luo, Y.; Liu, X.; Yu, R.; Meng, J.; Li, R. UAV Ad Hoc Network System Design. In Proceedings of the 2018 3rd International Workshop on Materials Engineering and Computer Sciences (IWMECS 2018), Jinan, China, 27–28 January 2018.
3. Park, S.; Kim, K.; Kim, H.; Kim, H. Formation control algorithm of multi-UAV-based network infrastructure. *Appl. Sci.* **2018**, *8*, 1740. [CrossRef]
4. Theodoridis, E.; Mylonas, G.; Chatzigiannakis, I. Developing an iot smart city framework. In Proceedings of the 2013 Fourth International Conference on Information, Intelligence, Systems and Applications (IISA), Piraeus, Greece, 10–12 July 2013; pp. 1–6.
5. Mekki, K.; Bajic, E.; Chaxel, F.; Meyer, F. A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express* **2019**, *5*, 1–7. [CrossRef]
6. Qiao, J.; Shen, X.S.; Mark, J.W.; Shen, Q.; He, Y.; Lei, L. Enabling device-to-device communications in millimeter-wave 5G cellular networks. *IEEE Commun. Mag.* **2015**, *53*, 209–215. [CrossRef]
7. Al-Sultan, S.; Al-Doori, M.M.; Al-Bayatti, A.H.; Zedan, H. A comprehensive survey on vehicular ad hoc network. *J. Netw. Comput. Appl.* **2014**, *37*, 380–392. [CrossRef]
8. Park, S.; Kim, H.T.; Kim, H. Energy-Efficient Topology Control for UAV Networks. *Energies* **2019**, *12*, 4523. [CrossRef]
9. Amouri, A.; Morgera, S.; Bencherif, M.; Manthena, R. A Cross-Layer, Anomaly-Based IDS for WSN and MANET. *Sensors* **2018**, *18*, 651. [CrossRef] [PubMed]
10. Jung, J.; Yoo, S.; La, W.G.; Lee, D.R.; Bae, M.; Kim, H. Avss: Airborne video surveillance system. *Sensors* **2018**, *18*, 1939. [CrossRef] [PubMed]
11. Chowdhury, K.R.; Di Felice, M.; Akyildiz, I.F. TCP CRAHN: A transport control protocol for cognitive radio ad hoc networks. *Mob. Comput. IEEE Trans.* **2013**, *12*, 790–803. [CrossRef]
12. Lee, W.; Kim, H.; Lee, J.Y.; Kim, H. Improving quality of multimedia services through network performance isolation in a mobile device. *Multimed. Tools Appl.* **2017**, *76*, 5317–5346. [CrossRef]
13. Chen, X.; Zhai, H.; Wang, J.; Fang, Y. TCP performance over mobile ad hoc networks. *Electr. Comput. Eng. Can. J.* **2004**, *29*, 129–134. [CrossRef]
14. Khurshid, A.; Kabir, M.H.; Das, R. Modified TCP NewReno for wireless networks. In Proceedings of the 2015 International Conference on Networking Systems and Security (NSysS), Dhaka, Bangladesh, 5–7 January 2015; pp. 1–6.
15. Zhang, Y.; Ansari, N.; Wu, M.; Yu, H. AFStart: An adaptive fast TCP slow start for wide area networks. In Proceedings of the 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012; pp. 1260–1264.
16. Man, C.L.T.; Hasegawa, G.; Murata, M. A new available bandwidth measurement technique for service overlay networks. In *Management of Multimedia Networks and Services*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 436–448.
17. Jude, J.A.; Ganesan, R. A comprehensive experimental analysis of standard TCP variants in vehicular environment. In Proceedings of the 2015 International Conference on Computing and Communications Technologies (ICCCT), Chennai, India, 26–27 February 2015; pp. 350–355.
18. Gururaj, H.; Ramesh, B. Performance Analysis of HSTCP for Optimizing Data Transfer rate in Mobile Ad-hoc Networks. *Int. J. Comput. Appl.* **2015**, *123*, 40–43.
19. Tomita, N.; Valaee, S. Data uploading time estimation for CUBIC TCP in long distance networks. *Comput. Netw.* **2012**, *56*, 2677–2689. [CrossRef]
20. Bao, W.; Wong, V.W.; Leung, V. A model for steady state throughput of TCP CUBIC. In Proceedings of the 2010 IEEE Global Telecommunications Conference (GLOBECOM 2010), Miami, FL, USA, 6–10 December 2010; pp. 1–6.
21. Pilimon, A.; Ruepp, S.; Berger, M.S. Robustness of Multiple High Speed TCP CUBIC Connections under Severe Operating Conditions. In Proceedings of the 2015 IEEE 14th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 28–30 September 2015; pp. 76–80.

22. Ha, S.; Rhee, I.; Xu, L. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Oper. Syst. Rev.* **2008**, *42*, 64–74. [CrossRef]
23. Marrone, L.A.; Barbieri, A.; Robles, M. TCP Performance. *J. Comput. Sci. Technol.* **2013**, *13*, 1–8.
24. Morabito, R. Virtualization on internet of things edge devices with container technologies: A performance evaluation. *IEEE Access* **2017**, *5*, 8835–8850. [CrossRef]
25. Wu, C.H.; Qin, K.Y.; Li, P.; Wu, G.H. The Dynamic Routing Protocol Implementation Strategy of the Cooperative Control Awareness Combat Network. *Appl. Sci.* **2018**, *8*, 948. [CrossRef]
26. Jain, R.K.; Chiu, D.M.W.; Hawe, W.R. *A Quantitative Measure of Fairness and Discrimination*; Eastern Research Laboratory, Digital Equipment Corporation: Hudson, MA, USA, 1984.
27. Lee, J.Y.; Kim, H.; Lee, W.; Kim, H. Adaptive Transmission Scheme for TCP in Wireless Multi-Hop Network. In Proceedings of the 2016 IEEE 41st Conference on Local Computer Networks (LCN), Dubai, UAE, 7–10 November 2016; pp. 503–506.