applied sciences

MDPI

Article

AccFlow: Defending against the Low-Rate TCP DoS Attack in Drones

Yuan Cao [1], Haotian Li [1], Lijuan Han [1], Xiaojin Zhao [2], Xiaofang Pan [3] and Enyi Yao [4,*]

[1] The College of Information Science and Engineering, Hohai University, Changzhou 213022, China; caoyuan0908@gmail.com (Y.C.); 1862510127@hhu.edu.cn (H.L.); hlj20192611@gmail.com (L.H.)
[2] The College of Electronic Science and Technology, Shenzhen University, Shenzhen 518060, China; eexjzhao@szu.edu.cn
[3] The College of Information Engineering, Shenzhen University, Shenzhen 518060, China; eexpan@szu.edu.cn
[4] School of Microelectronics, South China University of Technology, Guangzhou 511442, China
* Correspondence: yaoenyi@scut.edu.cn

Abstract: As drones are widely employed in various industries and daily life, concerns regarding their safety have been gradually emerging. Denial of service (DoS) attacks have become one of the most significant threats to the stability of resource-constrained sensor nodes. Traditional brute-force and high-rate distributed denial of service (DDoS) attacks are easily detectable and mitigated. However, low-rate TCP DoS attacks can considerably impair TCP throughput and evade DoS prevention systems by inconspicuously consuming a small portion of network capacity, and whereas the literature offers effective defense mechanisms against DDoS attacks, there is a gap in defending against Low-Rate TCP DoS attacks. In this paper, we introduce AccFlow, an incrementally deployable Software-Defined Networking (SDN)-based protocol designed to counter low-rate TCP DoS attacks. The main idea of AccFlow is to make the attacking flows *accountable* for the congestion by dropping their packets according to their loss rates. AccFlow drops their packets more aggressively as the loss rates increase. Through extensive simulations, we illustrate that AccFlow can effectively safeguard against low-rate TCP DoS attacks, even when attackers employ varying strategies involving different scales and data rates. Furthermore, whereas AccFlow primarily addresses low-rate TCP DoS attacks, our research reveals its effectiveness in defending against general DoS attacks. These general attacks do not rely on the TCP retransmission timeout mechanism but rather deplete network resources, ultimately resulting in a denial of service for legitimate users. Additionally, we delve into the scalability of AccFlow and its viability for practical deployment in real-world networks. Finally, we demonstrate the effectiveness of AccFlow in safeguarding network resources.

Keywords: drones security; DoS attack; Software-Defined Networking

check for updates

Citation: Cao, Y.; Li, H.; Han, L.; Zhao, X.; Pan, X.; Yao, E. AccFlow: Defending against the Low-Rate TCP DoS Attack in Drones. *Appl. Sci.* **2023**, *13*, 11749. https://doi.org/10.3390/app132111749

Academic Editors: Lip Yee Por and Abdullah Ayub Khan

Received: 14 September 2023
Revised: 24 October 2023
Accepted: 24 October 2023
Published: 27 October 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Drones or Unmanned Aerial Systems (UAS) are unmanned aerial vehicles operated by radio remote control devices [1]. Due to their high degree of flexibility and adaptability, UAS are widely used in both military and civilian applications. However, a number of security concerns have arisen [2,3] pertaining to the confidentiality, integrity, and availability of drones. One emerging threat is the "Low-rate TCP DoS Attack". In this type of attack, assailants exploit the TCP retransmission timeout mechanism [4]. To launch such an attack, the attackers set up periodic on–off "square-wave" traffic patterns with a peak transmission rate that is significant enough to exhaust the network bandwidth. When subjected to this attack, legitimate TCP flows experience severe packet losses, resulting in retransmission timeouts. If the period of the attacking flow coincides with the retransmission timeouts, the legitimate TCP flows encounter another peak as they attempt to recover from these timeouts, leading to further packet losses and longer retransmission timeouts. This cycle continues, ultimately throttling the legitimate TCP flows to nearly zero throughput.

Compared to general DoS attacks in which malicious users cause a denial of service by sending continuous high-rate flows, rather than relying on the TCP retransmission timeout mechanism, the time-averaged bandwidth usage of the low-rate TCP DoS attacking flow is small. In some cases, it can even be much less than the total available bandwidth. This is why we refer to such an attack as the *low-rate* TCP DoS attack.

Another noteworthy characteristic of the low-rate TCP DoS attacking flow is its periodic traffic pattern, which bears a resemblance to legitimate TCP periodic traffic, such as video traffic adopting the DASH (Dynamic Adaptive Streaming over HTTP) [5] standard. Despite the apparent similarity in traffic patterns, a fundamental distinction exists between benign TCP periodic flows and the low-rate TCP DoS attacking flow.

The key difference lies in how they respond to the packet loss. Legitimate TCP flows follow a protocol that involves backing off and entering retransmission timeout when their packets are lost. In contrast, the low-rate TCP DoS attacking flow does not exhibit this behavior.

Although the low-rate TCP DoS attack has been proposed for nearly ten years, it has not been fully addressed. Sun et al. [6] use signal processing (autocorrelation of the traffic) to detect the periodic burst attack at the congested router. Whenever attacks have been detected, the router traces back to its upstream routers to find the attack source. Such a solution may not work if the congested router has multiple upstream routers so that the bursty traffic it detects consists of the aggregate traffic from these upstream routers. Therefore, it is possible that the upstream routers cannot detect the bursty attacking traffic, which disrupts or halts the tracing back process. The work by Chang et al. [7] addresses this problem by assigning high priorities to the packets that are destined for high loss rate TCP application ports. However, such a defense mechanism can be breached if the attackers send large volumes of traffic to a specific protected port to cause a high loss rate at this port. Consequently, the attackers' traffic will be marked as high-priority traffic. Wang et al. [8] propose a defense method based on sFlow and an improved Self-Organizing Map (SOM) model in Software-Defined Networking (SDN). This approach defines a metric to differentiate between genuine normal traffic and large-scale attack traffic, allowing for the issuance of drop or rate-limiting rules near the source of the attack. However, when two types of traffic are very similar, stricter restrictions may affect the experiences of normal users. Bhuyam et al. [9] proposed a mechanism based on correlation coefficients to detect low-rate and high-rate DDoS attacks. Partial rank correlation is used to detect low-rate attacks. However, this mechanism may not be effective when only one macro traffic attacks the network. Ruchi Vishwakarma et al. [10] present a honeypot-based approach that can be taken as a productive outset towards combatting Zero-Day DDoS Attacks to analyze ways to prevent SSH and Telnet Protocol Attacks [11]. Tang [12] proposed a detection method to identify LDOS attacks at the transport layer and used data mining technology to analyze network anomalies under LDOS attacks to complete the detection. Furthermore, because all of the aforementioned solutions target solely the ideal low-rate TCP DoS attack, one alternative strategy for attackers to circumvent the defense could be splitting their traffic into multiple attacking flows to trigger distributed DoS attacks. Additionally, they do not illustrate how their defending protocols will impact the benign periodic flows, such as the aforementioned video traffic. In this paper, we introduce an Accountable Flow (AccFlow) protocol to offer effective defense against low-rate TCP DoS attacks. AccFlow has been developed to offer effective defense against low-rate TCP DoS attacks while ensuring that there is no performance degradation for legitimate periodic flows. Additionally, AccFlow offers robust protection against general DoS or DDoS attacks.

In contrast to previous literature, our approach to designing AccFlow distinguishes itself by integrating the principles of Software-Defined Networking (SDN) [13] and emphasizing flow accountability. Within the SDN architecture, decisions concerning packet routing and forwarding are centralized, distinguishing it from conventional networking approaches. This centralized approach empowers explicit execution of distinct policies for various flows, thereby offering flexibility and innovation in network configuration.

For instance, one notable advantage of this centralized architecture is that it empowers network operators to effectively manage traffic, resulting in the creation of low-latency and congestion-free networks. This is particularly advantageous in data center environments [14,15]. Although originally not designed to address security issues in computer networking, the concept of SDN, a network architecture that centralizes control and offers innovative avenues for reevaluating and mitigating such challenges [16]. Notably, the centralized network architecture enables the controller to perform real-time traffic monitoring and analysis. When attacking flows are identified, the controller can promptly block them, preserving network resources for legitimate flows. This ability to quickly respond to threats and prioritize legitimate traffic is one of the key advantages of SDN-based defense techniques. These techniques offer immediate benefits to the deployed routers or Autonomous Systems (ASes) since they do not necessitate reconfiguration in other parts of the network. Despite these advantages, flow-based security protocols must be scalable to effectively handle a large volume of attacking flows, particularly when considering deployment in Wide Area Networks (WANs). In this paper, we propose to use flow aggregation (we use source IP address-based flow aggregation; the detailed explanation is in Sections 4.1 and 5.2) and virtual centralized controller (the concept of virtual centralized controller is that we use multiple coordinated processors to serve as the centralized controller; the details are in Section 4.2). to solve the scalability problem so that our protocol can be deployed in both Local Area Networks (LANs) and WANs.

The reason why the low-rate TCP DoS attack and other kinds of DoS attacks work well is that whenever a congestion happens, the router drops the packets from all flows regardless of who causes the congestion. In other words, *accountability* for the congestion is not considered for packets dropping [17]. As a result, even legitimate TCP flows that diligently adhere to the congestion avoidance protocol and transmit at reasonable rates are erroneously affected by congestion. This congestion is primarily caused by attacking flows that flood the network with large volumes of traffic, depleting available bandwidth. Therefore, in order to effectively combat DoS attacks, AccFlow considers congestion accountability when implementing packet drops. To be precise, the more accountable flows are to contribute to the congestion, the more assertively AccFlow drops their packets. We establish a connection between each flow's accountability for congestion and its loss rate. Essentially, the higher the loss rate, the greater the flow's accountability to cause congestion. This connection is rooted in the behavior of attacking flows, which consistently exhibit high loss rates due to their higher accountability for congestion. They do so because they must continually send an excessive number of packets to overwhelm the network. In contrast, legitimate TCP flows, with their lower accountability for congestion, rarely experience consistently high loss rates. Instead, they adapt by reducing their transmission rates and may even enter timeouts when their packets are dropped. As a result, there is a positive correlation between a flow's loss rate and its accountability for causing congestion. AccFlow safeguards the network by prioritizing the dropping of packets from flows with higher loss rates and corresponding higher probabilities. Conversely, packets from flows with lower loss rates are dropped with lower probabilities.

Through extensive simulations using ns-3 [18], we illustrate that AccFlow proves effective in defending against both the low-rate TCP DoS attack and general DoS attacks. In summary, this paper makes the following contributions.

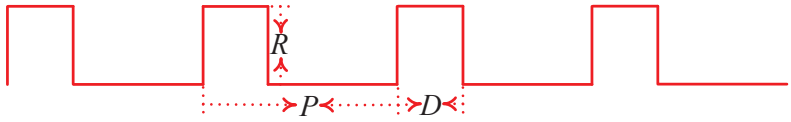
- AccFlow is the first SDN-based security protocol that considers flow accountability when defending against DoS or DDoS attacks.
- We demonstrate that AccFlow, which does not cause any performance degradation to benign flows, can effectively defend against both the low-rate TCP DoS attack and general DoS attacks even if attackers are able to vary their strategies.
- We use flow aggregation and virtual centralized controller to solve the scalability problem of AccFlow and make it deployable in real networks.

The rest of the paper is organized as follows. In Section 2, we give a brief introduction to the low-rate TCP DoS attack. In Section 3, we elaborate on the AccFlow protocol. In

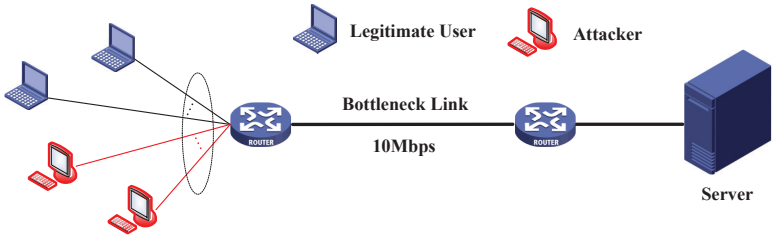
Section 4, we consider the deployment of AccFlow in real networks and its interaction with other security protocols. In Section 5, we thoroughly study the effectiveness of AccFlow in different simulation settings. Finally, we conclude in Section 6.

2. Low-Rate TCP DoS Attack

In this section, we provide a brief overview of the low-rate TCP DoS attack and its effectiveness in causing denial of service to legitimate TCP flows. The characteristics of an ideal low-rate TCP DoS attacking flow can be represented by a triple $\{R, P, D\}$, where R indicates the peak data rate, P indicates the attacking period and D indicates the burst duration within one period, as illustrated in Figure 1a. In order to overflow the network, R needs to be larger than the bottleneck link bandwidth. P should be small enough, compared to the RTTs of legitimate TCP flows, to attack most of the traversing flows. D is negatively correlated with R if the amount of traffic generated by the attackers in one period is fixed. A detailed discussion on the choice of R, P, and D can be found in [19].



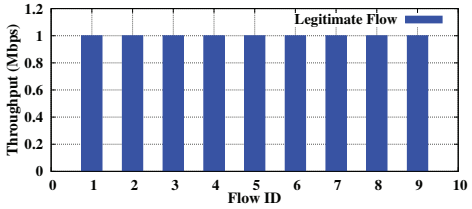
(a) Low-rate TCP DoS attacking flow.



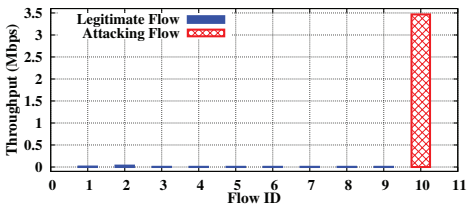
(b) Network topology.

Figure 1. Attacking traffic and network topology.

We set up simulations on the ns-3 platform to illustrate the effectiveness of low-rate TCP DoS attacks. We create a "dumbbell" network topology whose bottleneck link bandwidth is 10 Mbps , as illustrated in Figure 1b. In this simulation setup, nine legitimate TCP flows and one attacking flow are traversing the bottleneck link. We configure the network so that each legitimate TCP flow is transmitting at 1 Mbps and the attacking flow triple $\{R, P, D\}$ is {30 Mbps, 200 ms, 67 ms}. Note that we scale down the bottleneck link bandwidth and flow rates in order to accelerate the simulation. As illustrated in Figure 2a, without being attacked, all legitimate TCP flows fairly share the bottleneck link bandwidth and achieve their desired data rates. However, they are throttled to nearly zero throughput under attack, as illustrated Figure 2b. Note that the time-averaged data rate of the attacking flow (around 3.5 Mbps) is far less than the bottleneck link bandwidth. Therefore, attackers use much less resources to achieve very effective DoS attacks.



(a) Throughput without being attacked.



(b) Throughput under attack.

Figure 2. Effectiveness of the low-rate TCP DoS attack.

3. AccFlow Design

In this section, we provide an in-depth look at AccFlow. AccFlow is a transport layer protocol deployed on the SDN centralized controller. The controller's role encompasses monitoring all traversing flows, conducting flow analysis, and instructing the switches or routers to implement different routing and forwarding policies for various flows, as illustrated in Figure 3. The architecture of AccFlow is depicted in Figure 3 as well. AccFlow comprises two primary modules: Aggressive Detection and Early Drop. Aggressive Detection is responsible for blocking attacking flows that exhibit detectably aggressive behavior, whereas Early Drop is employed to safeguard the network when attackers attempt to evade detection by intelligently altering their attacking strategies. As Flow Data, computed by other layers, traverses the Transport Layer, AccFlow, equipped with these two modules, effectively safeguards the network from attacks through Flow Prioritization and Flow Analysis.

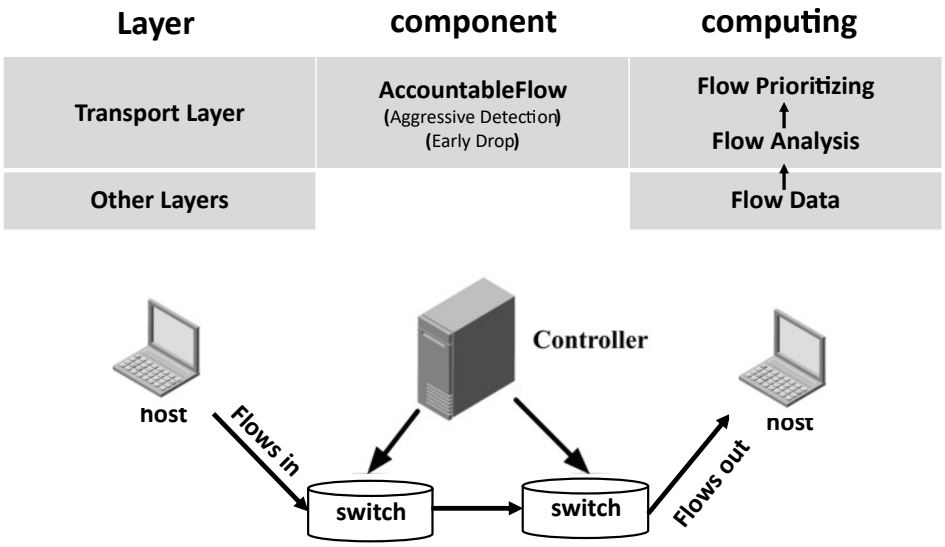
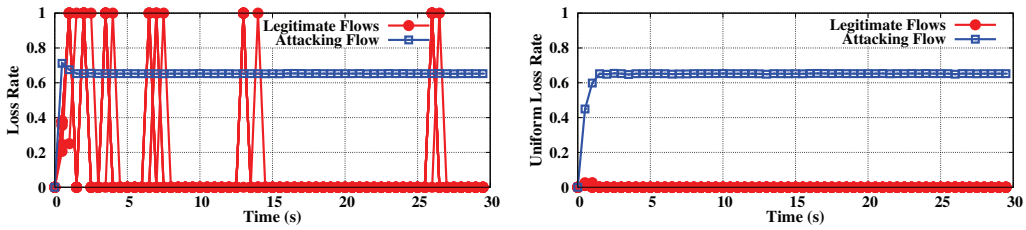


Figure 3. AccFlow architecture.

3.1. Aggressive Detection

To effectively detect attacking flows, it is essential to identify unique features that set them apart from legitimate ones. Given that attackers consistently generate high traffic volumes to overload the network, the loss rates of their flows (the ratio of lost packets to the total transmitted packets) are expected to be higher compared to legitimate flows. Consequently, a straightforward method to distinguish between legitimate flows and attacking flows is by using the loss rate as a differentiator. In order to validate the effectiveness of this intuitive approach, we conducted a study on the loss rates of each flow during the simulation, as discussed in the previous section. The centralized controller closely monitors the traffic and performs periodic statistical analysis on all traversing flows. The flow analysis period of the controller should be set to two or three times the typical Round-Trip Times (RTTs) of the traversing flows. This enables the controller to gain accurate insights into the behaviors of these flows and respond rapidly to attacks. In our simulation, we configured this period to be 0.5 s, which is approximately twice the typical RTTs of the traversing flows. In the rest of the paper, we use the term *detection period* to indicate the controller's analysis period. The results of the simulation are presented in Figure 4a.



(a) Loss rate of each flow. (b) Uniform loss rate of each flow.

Figure 4. Loss rate and uniform loss rate of each flow.

It is evident that the loss rate of attacking flows remains consistently high. However, we have observed that in certain detection periods, the loss rates for some legitimate flows are even higher. Consequently, relying solely on loss rate as a detection criterion may lead to false positives.

Upon closer examination of the traffic generated by each flow, we have identified the reason behind these higher loss rates in legitimate flows during specific detection periods. This phenomenon occurs when a legitimate flow sends only one packet within a detection period, and that single packet is dropped. In such cases, the legitimate flow reacts by stepping back and waiting for one retransmission timeout before entering the TCP slow start phase. At the outset of the slow start, the flow sends out one packet to probe the available bandwidth. In cases of extreme network congestion, this newly generated packet is highly likely to be dropped. Consequently, the legitimate flow is compelled to endure an even longer retransmission timeout. This explains why the loss rate of a legitimate flow is either 1 (the only packet is dropped) or 0 (waiting in retransmission timeout). In contrast, the attacking flow adopts a different strategy. It continuously dispatches a substantial volume of packets without any form of backing off, even when a significant number of preceding packets are dropped. As a result, the attacking flow maintains a consistently high loss rate.

In our framework, we propose to use *Uniform Loss Rate (ULR)* which is defined in Equation (1). We use it to differentiate the attacking flow from legitimate flows.

$$ULR = loss\,rate \times usage\,rate \tag{1}$$

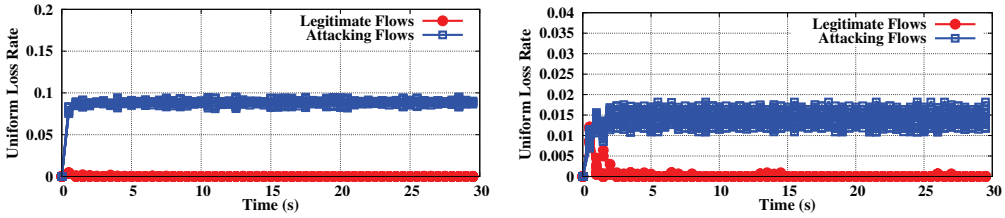
The usage rate of one flow is defined in Equation (2). In the equation, the usage rate of one flow is the ratio of the number of its transmitted packets in one detection period over the total number of arriving packets from all flows in this detection period.

$$usage\,rate = transmitted\,packets \div total\,packets \tag{2}$$

The attacking flow is featured with high ULR since it has to consistently send large numbers of packets (high usage rate) and never backs off even if its packets are dropped (high loss rate). As illustrated in Figure 4b, there is a notable gap between the ULR of the attacking flow and legitimate flows. Therefore, ULR is an effective feature to leverage to differentiate the attacking flow from legitimate ones. Whenever detecting a flow with excessively high ULR while other flows' ULRs are close to zero, the centralized controller will identify it as an attacking flow and completely block its traffic.

Although Aggressive Detection proves to be a straightforward and precise defense mechanism against the ideal low-rate TCP DoS attack, it necessitates a substantial ULR (Upstream Loss Rate) gap to effectively distinguish between the attacking flow and legitimate flows. Finding a reasonable ULR (Upstream Loss Rate) threshold can be a challenging task, particularly when attackers continuously alter their strategies to reduce the ULRs of their flows. For example, instead of launching a single attacking flow, attackers can divide their traffic into N flows and synchronize them to create periodic burst flows. The usage rate of each individual attacking flow decreases by a percentage of $\frac{N-1}{N}$, including its ULR. As the

number of synchronized attacking flows increases, the ULR gaps between legitimate flows and attacking flows diminish, rendering it more challenging for the controller to detect the attacking flows. However, it is important to note that even as the ULR gaps decrease, the network still experiences the same volume of attacking traffic, allowing the DoS attack to remain effective. We present the shortcoming of Aggressive Detection in Figure 5. Note that when the attackers split their traffic into 50 synchronized flows, the ULR differences between attacking flows and legitimate flows are close to zero. To tackle such distributed attacks, we design the Early Drop module in the next subsection.



(a) 10 attacking flows. (b) 50 attacking flows.

Figure 5. ULR of each flow under distributed DoS attack.

3.2. Early Drop

Early Drop is proposed to effectively deal with the aforementioned distributed attacks. The design of Early Drop is also based on consistent flow monitoring and periodic flow analysis by the centralized controller. However, Early Drop does not purely rely on the ULR to detect attacking flows. In fact, Early Drop is a heuristic algorithm illustrated in Algorithm 1 that conducts flow-based packet dropping according to each flow's loss rate without explicitly detecting attacking flows. Next, we elaborate on the Early Drop algorithm.

The algorithm is executed during every detection period. The first 4 lines of the code are executed only once at the beginning of each detection period, whereas the remaining lines are executed whenever a packet arrives within that detection period. The aggregate loss rate, denoted as $\mathcal{L}$ (line 2), represents the ratio of the number of dropped packets from all flows in the *previous* detection period to the total number of arriving packets in the *previous* detection period. Similarly, usage U_i of flow F_i (line 4) reflects the number of packets sent by F_i in the previous detection period. Loss rate L_i of F_i (line 4) is calculated as the ratio of the number of dropped packets from F_i in the previous detection period to U_i. It is important to note that these values, namely$\mathcal{L}$, U_i, and L_i, are computed based on statistics from the previous detection period. In the event that the current detection period is the first, the controller initializes all these values to zero. These values remain constant within the current detection period and are updated at the start of the subsequent detection period. The rest of the lines of the codes, starting from line 5 to the end, execute the packet-dropping policy based on these calculated values.

We add a condition $\mathcal{L} > Th_1$ in line 5 for packet dropping. This is because Early Drop initiates packet dropping before the network bandwidth is fully exhausted. Therefore, it is necessary to make sure that the network is being attacked before applying such an aggressive dropping policy. In other words, Early Drop is a self-protective mechanism that automatically begins dropping packets when the network shows a sign of being attacked. We use aggregate loss rate to verify whether the network is being attacked or not for the following reasons. This approach is grounded in the behavior of legitimate TCP flows, which adhere to congestion-avoidance protocols. When packets are lost due to substantial congestion, legitimate flows respond by throttling their transmission rates, aligning them with the available bandwidth. Consequently, even if these flows initially have high original transmission rates, they adjust their data rates to match the prevailing network conditions. Therefore, it is rare for the network to witness a consistently high aggregate loss rate under normal situations. However, when attackers are trying to cause denial of service to legitimate flows by exhausting the network bandwidth, they have to

continuously generate traffic even though many of their packets are dropped, i.e., they never back off when meant to do so. As a result, the network will experience a very high aggregate loss rate under attack. We verify our analysis by studying the aggregate loss rate in both normal and attacked scenarios. The experimental results, illustrated in Figure 6, show that even 40 legitimate flows each with the original 1 Mbps transmission rate are traversing the 10 Mbps bottleneck link, the aggregate loss rate is well below 10% (the real network is always bandwidth over-provisioned to tolerate the traffic burst caused by legitimate flows, which makes it rare for the network to have a large aggregate loss rate under normal situation). However, when the attackers are trying to launch an attack, the aggregate loss rate is more than 65%. Thus, aggregate loss rate is an effective feature to indicate whether the network is under DoS attack or not. The network operators can have different configurations for the threshold Th_1 based on their own policies and traffic characteristics. For instance, if they want to aggressively protect their network, they need to set a relatively low threshold for Th_1 and vice versa.

Algorithm 1: Early Drop

if *the beginning of kth detection period* **then**
  Calculate the aggregate loss rate $\mathcal{L}$;
  for *each traversing flow* F_i **do**
    Calculate its usage U_i and loss rate L_i;
    // Loss rate L_i is the ratio of the number of dropped packets from F_i in the previous detection period to U_i
if $\mathcal{L} > Th_1$ **then**
  // Th_1 is set up based on policies and traffic characteristics
  for *each arriving packet* **do**
    Find the flow F_j it belongs to;
    if $U_j > Th_2$ **then**
      // Th_2 is set up to make sure that Early Drop will not falsely blame the legitimate TCP flows
      if $L_j > 0.5 \times \mathcal{L}$ **then**
        Drop the packet with probability L_j;
        else if $N_{packet} > Th_3$ **then**
          Drop the packet with probability L_j;

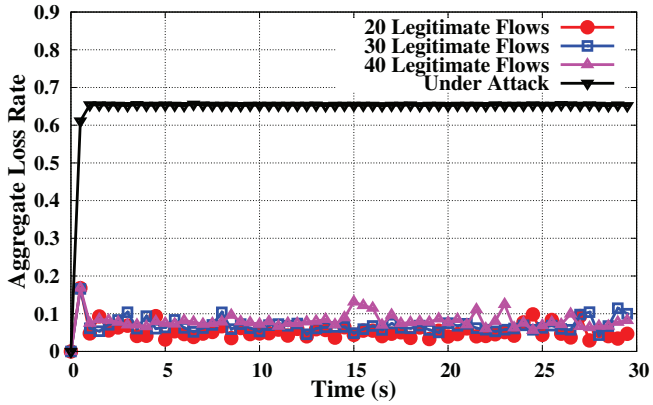


Figure 6. Aggregate loss rate for normal and attacked scenarios.

The main idea of the Early Drop algorithm is that it drops the packets from *accountable* flows with *reasonable* probabilities. By "accountable", we mean that Early Drop only blames the flows that are accountable for the congestion. By "reasonable", we mean

that Early Drop drops the packets of accountable flows according to their loss rates. We achieve accountability by line 8 of the algorithm. In particular, if a flow only sends one or two packets during one detection period, it is not accountable for the congestion so that Early Drop will not drop its packets. We add the threshold Th_2 to make sure that Early Drop will not falsely blame the legitimate TCP flows who have just recovered from retransmission timeouts and send one packet in the beginning of the TCP slow start process to probe the available bandwidth. Th_2 should be small and increase with the duration of the detection period since a longer detection period may contain more TCP slow start processes. We set Th_2 as 5 in our simulations when we test the effectiveness of AccFlow in the next section. Furthermore, we accomplish reasonability by considering its loss rate while dropping packets from a particular flow (lines 9 to 12). Specifically, Early Drop divides all flows into two groups, i.e., the high loss rate group and the low loss rate group. All flows whose loss rates are above half of the aggregate loss rate $\mathcal{L}$ will be categorized into the high loss rate group and Early Drop immediately drops their packets according to their loss rates. On the contrary, flows whose loss rates are no greater than half of the $\mathcal{L}$ will be assigned to the low loss rate group and Early Drop applies packet dropping to these flows only when the number of queueing packets N_{packet} in the router is larger than Th_3. The threshold Th_3 is used to indicate that the network is slightly congested so that it is positively related to the router's buffer size. We set Th_3 to be 10% of the router's buffer size in our simulations. Again, the network operators can have different configurations for Th_3 according to their policies. To sum up, Early Drop blames the flows that are accountable for the congestion, and the higher their loss rates, the more aggressively it drops their packets. The fundamental difference between Early Drop and other Active Queue Management disciplines such as RED and WRED is that Early Drop selectively drops packets from more accountable flows (often the attacking flows) *early* before the router buffer is exhausted so that the packets from less accountable flows (often the legitimate flows) can be enqueued. However, RED and WRED simply drop all arriving packets when the buffer is full so that the legitimate flows will suffer from denial of service.

Note that we can use Aggressive Detection as a supplement to Early Drop. In particular, Early Drop is always active to protect the network whereas Aggressive Detection will be applied to completely block the attacking flows if they perform aggressively enough to be detected by the controller. In the next section, we thoroughly test the effectiveness of AccFlow through substantial amounts of simulations.

4. Deployment of AccFlow

In this section, we address the deployment of AccFlow in real networks and its interaction with other security protocols. Although AccFlow offers advantages like flexible control and real-time response to attacks, it also faces scalability challenges due to the centralized network architecture. The centralized controller must create an entry in the routing table for each distinct flow, which can strain CPU and storage resources. To overcome this scalability issue, we propose the use of flow aggregation and a virtual centralized controller. These solutions aim to optimize AccFlow's performance and ensure its practicality in large-scale networks.

Flow aggregation involves grouping similar flows together, reducing the number of distinct flows and, consequently, the load on the controller. The virtual centralized controller concept employs multiple coordinated processors to serve as the centralized controller, allowing for distributed handling of flow management. Together, these measures enhance AccFlow's scalability, making it suitable for deployment in both local area networks (LANs) and wide area networks (WANs).

By addressing scalability concerns and optimizing AccFlow's performance, we aim to make it a viable and effective solution for enhancing network security and mitigating DoS and DDoS attacks.

4.1. Flow Aggregation

Flow aggregation can prevent the attackers from amplifying their attacking scale by faking huge numbers of flows. Furthermore, with protocols like ingress filter [20] and Passport [21], ASes can limit the range of their acceptable source IP addresses. As a result, the number of distinct attacking flows that can be used to attack the bottleneck link is limited. Moreover, the existing security protocols, such as MiddlePolice [22,23], Mirage [24], Phalanx [25], Pushback [26], and DoS-limiting architecture [27], can be applied to further limit the attacking scales. For instance, Mirage adopts the concept of frequency hopping in wireless networks to "hop" the destination IP addresses among all available addresses. Each time a user wants to send traffic to this site, it has to solve a computational puzzle to obtain the new IP address, which will limit the volumes of traffic that the computationally limited attackers can send. MiddlePolice, on the other hand, allows the destination to determine which source IPs are allowed through self-defined traffic control policies. Since AccFlow does not cause disruption to the existing network infrastructure, it can effectively interact with these security protocols to defend the network against extremely large-scale DDoS attacks.

4.2. Virtual Centralized Controller

In order to deal with large numbers of flows, we can also adopt the concept of a virtual centralized SDN controller. Specifically, multiple processors can serve as the conceptual centralized controller. Each processor keeps its routing table and tackles a certain number of flows. In order to tolerate individual processor failures within the distributed virtual centralized controller system, we can adopt the Paxos protocols [28]. In fact, the B4 architecture [14], a worldwide large-scale SDN data center network built by Google, also adopts the concept of virtual centralized controller by clustering their networks to deal with millions of flows traversing Google's data centers. By embracing these techniques to solve the scalability problem, AccFlow can be deployed in real networks. We present a straightforward deployment architecture in Figure 7.

We consider deploying AccFlow on both core routers and border routers. Typically, border routers are responsible for dealing with the inter-domain flows such as BGP sessions [29] whereas core routers carry the traffic across the AS and may execute a particular traffic engineering policy such as MPLS [30]. Consider the situation where a remote legitimate client and the attacker are sending their traffic to the AS through an undeployed border router R_1. Since the attacking flow D exhausts the bandwidth of the victim border router R_1, the client's flow A is throttled to zero throughput (flow A is not able to traverse R_1 in Figure 7). Attacking flow D continues to propagate in the AS until it counters a deployed core router R_2 which drops its packets to save the network resources for the legitimate flow C (attacking flow D is not able to traverse R_2 in Figure 7). When the AS deploys AccFlow on its border router R_3, it protects the inter-domain traffic B from being attacked so that B can safely enter the AS. Apart from launching inter-domain attacks, the attacker can also compromise the nodes within the AS to generate local DoS attacking flows, such as the attacking flow E. The deployed core router R_4 can protect the network from such an attack. Furthermore, the deployed routers can stop the local attacking flows from leaving the AS to attack remote sites. To sum up, AccFlow is compatible with the existing security protocols and is incrementally deployable without disruption to the existing network infrastructure.

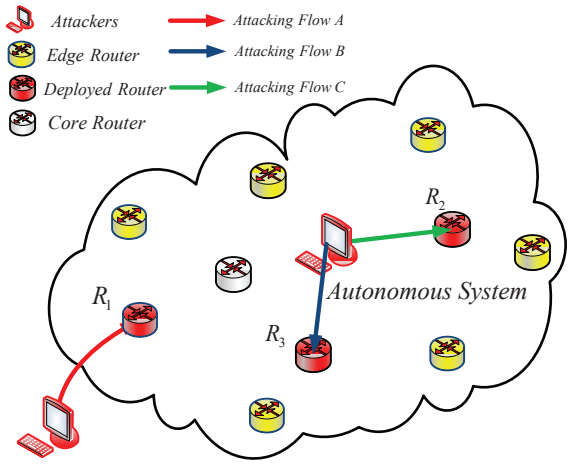


Figure 7. Deployment of AccFlow.

5. Experimental Results

In this section, we thoroughly study the effectiveness of AccFlow on the ns-3 platform in four major simulation setups. We use the network topology illustrated in Figure 1b in all simulation setups whereas the traversing flows are different under different setups (AccFlow is not limited to the simple dumbbell network topology. It is effective to protect both the inter-domain and intra-domain traffic). The first setup regards the distributed low-rate TCP DoS attack, in which we consider different types of legitimate traffic and different attacking strategies. The second setup is about another DoS attack derived from the low-rate TCP DoS attack. We call it the Short Selfish TCP Flow (SSTF) attack because the attackers selfishly consume nearly the whole network resources by generating excessive numbers of short TCP flows. The third setup is designed to verify that AccFlow does not falsely drop packets from legitimate periodic flows. Finally, we consider the general DoS attacks in the fourth setup.

5.1. Distributed Low-Rate TCP DoS Attack

In this setup, we design three different simulation settings. In setting one, we have five legitimate TCP flows each with 1 Mbps transmission rate to simulate drone activities, such as Data Transfer and Telemetry and Remote Control Signals. In a real network, the attackers are able to launch different scales of distributed attacks by splitting their traffic into different numbers of synchronized subflows which refer to the individual streams of network traffic that are coordinated or synchronized by the attacker. By dividing their traffic into multiple subflows, attackers can make the attacks more challenging to detect and mitigate, as the traffic may appear more like legitimate network activity. In our simulations, we varied the number of synchronized subflows from 1 to 50 to replicate real-world attacks. (Note that we scale down the number of flows in order to accelerate the simulation. As you can see in our experiment results, the performance of AccFlow is not impacted by the scale of DDoS attacks.) The total aggregate rate of these synchronized attacking flows reached approximately 30 Mbps, which is three times the bandwidth of the bottleneck link. This setup effectively mimics attacks that surpass the network's capacity, resulting in network congestion and potential service disruptions. The attacking period P is 200 ms and the attacking duration D in one period is 67 ms to simulate the real attacks. In setting two, we use the same attacking traffic as that of setting one but we have nine legitimate TCP flows each with a different transmission rate, ranging from 0.3 Mpbs to 1.1 Mpbs. This setup mimics diverse drone activities, including real-time adaptive data transmission and adaptive flight planning. The reason we have both setting one and setting two is that TCP uses the max–min fairness [31], where the network first satisfies the flows with smaller demands (lower transmission rates) and then evenly distributes the bandwidth to

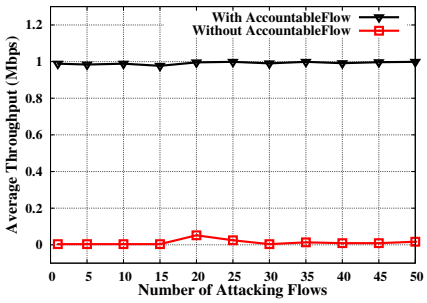
flows with larger demands if the network resources are limited. As a result, in a congested network, flows with smaller transmission rates can obtain their fair bandwidth shares more easily than flows with higher rates. Therefore, we need to consider both the two settings in our simulation. In the third setting, we test the effectiveness of AccFlow when attackers are varying their attacking rates from 20 Mbps to 60 Mbps which is more varied. Without loss of generality, we assume that attackers split their traffic into five attacking flows in this setting and we use the same legitimate TCP traffic as that of setting two. We summarize the three simulation settings in Table 1.

Table 1. Simulation settings .

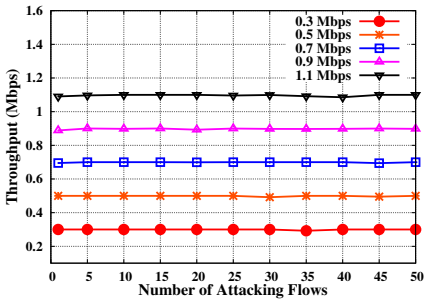
Simulation Setting	Legitimate TCP Flows	Attack Scales	Aggregate Attacking Rate
Setting One	5 flows with same rate 1 Mbps	1 to 50 attacking flows	30 Mbps
Setting Two	9 flows with different rates ranging from 0.3 Mbps to 1.1 Mbps	1 to 50 attacking flows	30 Mbps
Setting Three	9 flows with different rates ranging from 0.3 Mbps to 1.1 Mbps	5 attacking flows	20 Mbps to 60 Mbps

The simulation results are illustrated in Figure 8. Figure 8a illustrates the results for setting one when five 1 Mbps legitimate TCP flows are traversing the network. Since the whole bandwidth of the bottleneck link is 10 Mbps, which is large enough to hold all the legitimate traffic, all the five flows should not experience any packet losses (in this paper, we only consider packet losses caused by network congestions). and achieve their ideal throughput when they are not attacked. As we can see in Figure 8a, AccFlow can effectively protect the legitimate flows from being attacked since the average throughput of the legitimate flows is close to the desired transmission rate. Furthermore, the performance of AccFlow does not degrade as the number of attacking flows increases. Figure 8b illustrates the simulation results for setting two. Note that we set up nine legitimate flows in this setting but we only plot the results for five of them in the figure for a concise presentation. With AccFlow, all nine legitimate flows are able to achieve their desired data rates even under large-scale attacks. In setting three, we vary the aggregate attacking rates from 20 Mbps to 60 Mbps. Again, we plot the simulation results for 5 legitimate flows in the figure. The results show that AccFlow can also effectively defend the network even if attackers are able to change their attack rates.

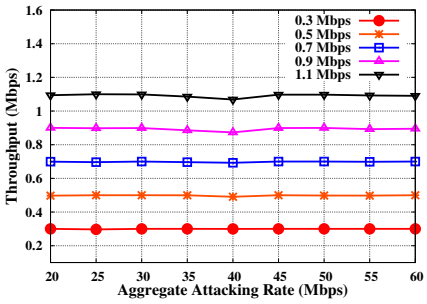
Here, we make a detailed explanation of the effectiveness of AccFlow. Consider one legitimate TCP flow and one attacking flow in our simulations. Assume that attackers launch DoS attacks in the detection period T_k. Due to severe packet losses in T_k, the legitimate TCP flow envisions a heavily congested network and enters a retransmission timeout. Therefore, in the next detection period T_{k+1}, either its loss rate is zero if it is still waiting in the timeout or its usage is very low if it just recovers from the timeout and sends small amounts of packets to probe the available bandwidth. Under both scenarios, AccFlow will not further blame the legitimate flow. However, the attackers have to continuously send high volumes of traffic in order to overflow the bottleneck link. Thus, the attacking flow still has a large usage in the next detection period T_{k+1} in spite of its high loss rate in T_k. Under such a situation, AccFlow will Early Drop its packets according to its loss rate. Furthermore, if the network is still congested after Early Drop, the router itself will also drop packets since it cannot deal with so much traffic. Therefore, the attacking flow will experience an even larger loss rate in the detection period T_{k+1}. This cycle repeats so that the loss rate of the attacking flow increases in each detection period until the network is not congested or its loss rate equals one. Under both scenarios, the attacking flow can no longer harm the network.



(a) Setting one.



(b) Setting two.



(c) Setting three.

Figure 8. Under all three settings, AccFlow can effectively defend legitimate TCP flows from being attacked. The achieved throughput for each legitimate flow is almost the same as its original transmission rate.

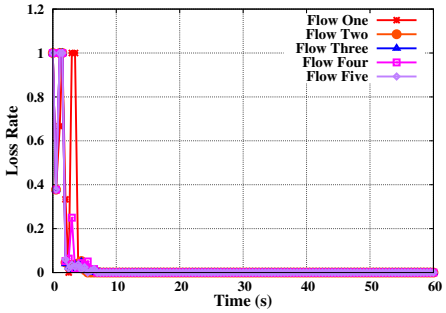
In order to be a real-time defending technique, AccFlow needs to react to attacks very fast. Here, we study the convergence time of AccFlow, i.e., how long it takes AccFlow to clear up the attacks. We define the convergence time as the time when all legitimate flow loss rates are zero. Without loss of generality, we randomly pick up one simulation setup in each of the three settings listed in Table 1 to study its convergence time. Specifically, we test the scenarios where attackers set up 20 and 30 attacking flows in setting one and setting two, respectively. As for setting three, we use the case when attackers are generating traffic at a rate of 40 Mbps. Our simulation results are illustrated in Figure 9. Under all three scenarios, AccFlow can react to the attack quickly and the convergence time is in the order of seconds or tens of seconds.

As a flow-based defending technique, AccFlow needs to be scalable to deal with large numbers of attacking flows. Although the performance of AccFlow does not decline as the number of attacking flows increases (as illustrated in our simulation results), huge numbers of flows will exhaust the CPU and storage of the centralized controller. We discuss the scalability problem and propose our solutions in Section 4 where we consider the deployment of AccFlow in real networks.

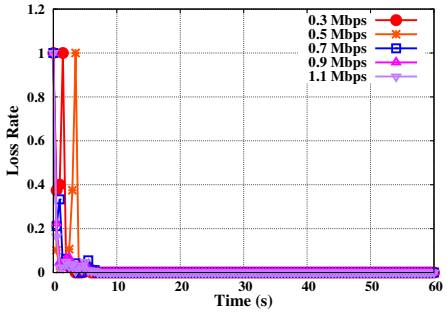
5.2. Short Selfish TCP Flow Attack

In this section, we discuss a very effective DoS attack which is similar to but still fundamentally different from the low-rate TCP DoS attack. The attacking technique is that malicious users periodically set up many short TCP flows to gain unfair share of network resources. Specifically, the early coming short TCP flows congest the network and cause all flows (including both legitimate TCP flows and themselves) to enter retransmission timeouts. Then, the attackers selfishly start new short TCP flows to occupy the whole network bandwidth since no one apart from attackers is transmitting now. The interesting point of such DoS attacks is that it seems that the attackers never deviate from the TCP protocol since these short TCP flows will back off when their packets are lost. However, the attackers are able to cause very severe denial of service to legitimate users simply by

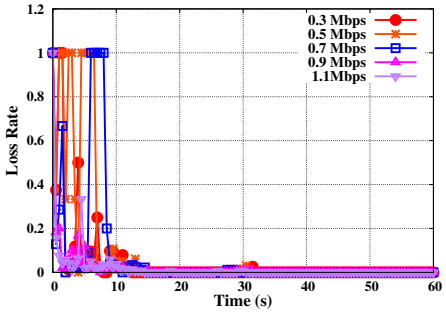
breaking their traffic into small short flows. We name such an attack the Short Selfish TCP Flow (SSTF) attack. Note that the difference between the SSTF attack and low-rate TCP DoS attack is that the former cannot synchronize all these short TCP flows to create the regular periodic burst traffic since transmitters have to wait for the ACKs before sending new packets. We set up simulations to illustrate the effectiveness of the SSTF attack. In our simulation, we have 10 attackers and each of them sends one short TCP flow with 3 Mbps data rate every 200 ms. Furthermore, we have nine legitimate TCP flows with 1 Mbps transmission rate each. The results, illustrated in Figure 10, show that the SSTF attack is able to effectively throttle the legitimate users to almost zero throughput.



(a) 20 attacking flows in setting one.



(b) 30 attacking flows in setting two



(c) 40 Mbps aggregate attacking rate in setting three

Figure 9. Convergence time of AccFlow.

Now, we explain why the SSTF attack can cause such an effective denial of service to legitimate users even though each individual short flow itself behaves exactly the same as a legitimate TCP flow (here, we mean each short TCP flow complies with the TCP protocol and will back off when a congestion happens). First, let us reconstruct the attacking procedure. Assume that attackers first set up n short TCP flows $\mathcal{A} = \{A_1, A_2, ..., A_n\}$ to cause congestion. Then all the legitimate flows and attacking flows in $\mathcal{A}$ will suffer from severe packet losses and enter retransmission timeouts. After a short period, attackers set up another set of short TCP flows $\mathcal{B} = \{B_1, B_2, ..., B_m\}$. Since no one is transmitting now, $\mathcal{B}$ will occupy the whole network resources. When the legitimate flows try to recover from timeouts, they may face another set of short attacking flows started by the attackers after flows in $\mathcal{B}$ finish. Again, congestion happens and the legitimate flows are forced to enter even longer retransmission timeouts. The cycle repeats and the attackers are able to selfishly utilize nearly the whole network resources. In a word, by sacrificing a small fraction of their traffic, the attackers create a "clear" network environment for most of their traffic and cause severe denial of service to the legitimate users.

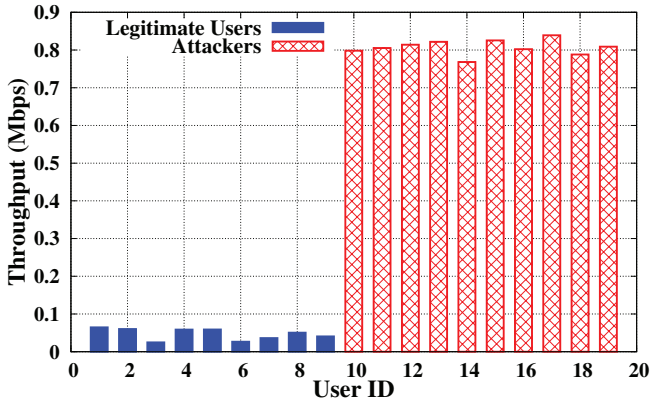


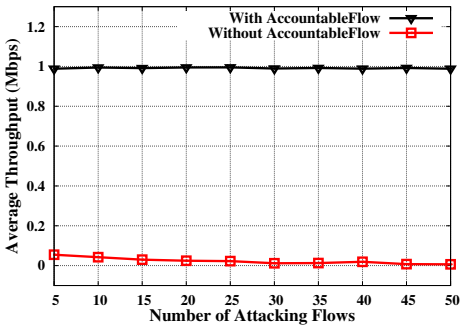
Figure 10. Effectiveness of the SSTF attack.

The trick played by the attackers is to evade accountability by continuously generating fresh short flows. In particular, it is the flows in $\mathcal{A}$ that cause the congestion so that we have no reason to blame the flows in $\mathcal{B}$. However, when flows in $\mathcal{A}$ experience severe packet losses, the source should realize that the network is congested and should not start new flows. Thus, flow set $\mathcal{B}$ should not be generated because both $\mathcal{A}$ and $\mathcal{B}$ come from the same source (attackers). Therefore, although each individual short TCP flow complies with the TCP protocol, the attackers still behave maliciously by periodically setting up new TCP flows even though the previous flows experience high loss rates.

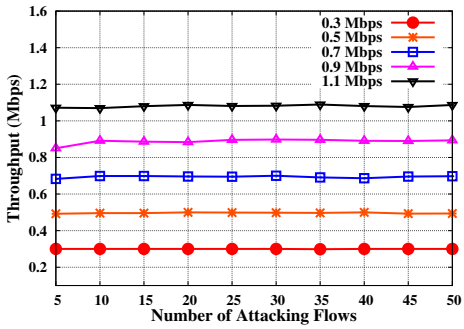
We propose to use *flow aggregation* to defend against the SSTF attack. Specifically, all flows with the same source IP address will be aggregated as one flow. (It is possible to conduct flow aggregation based on other properties such as source IP address and application port pair. We leave the discussion of different aggregating properties in future works.) Consequently, although flow $A_i \in \mathcal{A}$ and flow $B_j \in \mathcal{B}$ are different flows, they may have the same source IP address since they are both generated by the attackers. As a result, AccFlow will aggregate them as one flow. Therefore, flow B_j will be blamed for the congestion caused by flow A_i. Similarly, the subsequent flows will be accountable for the congestion caused by their precedent flows as long as they have the same source IP address. Thus, the attackers are not able to selfishly over-utilize the network resources by creating new flows. A potential problem for conducting such flow aggregation is that attackers can spoof their source IP addresses to keep generating new flows. However, the network security community has proposed effective mechanisms such as Stackpi [32] and packet filters [33] to prevent source IP spoofing. AccFlow can embrace such security protocols to prevent attackers from faking flows. Another potential problem is that the Network Address Translation (NAT) router translates the IP addresses of the hosts within its LAN to its public IP address. Then, all the flows from different hosts will have the same source IP address after they leave the LAN. When they reach other remote sites that deploy AccFlow, they will be aggregated as one flow. Thus, a single compromised host within the LAN may cause denial of service to all the legitimate hosts within the LAN since their flows are aggregated as the same flow. To solve the problem, we can deploy AccFlow on the NAT router so that it will drop the packets from the local attacking flow and save bandwidth for legitimate flows. Therefore, the local attacking flow will not be able to leave the LAN to attack the remote sites.

We test the effectiveness of AccFlow when the network is faced with the SSTF attack under similar simulation settings in Table 1. As illustrated in Figure 11, Accountable Flow can effectively defend against the SSTF attack. Note that we set the minimum number of attacking flows to five since the SSTF attack needs to be distributed in order to be effective. The convergence time is also in the order of seconds to tens of seconds. We do not present the results for the convergence time in the paper due to space constraints. Note that the achieved data rates (throughput) for higher rate flows, i.e., ones with rates of 0.9 Mbps

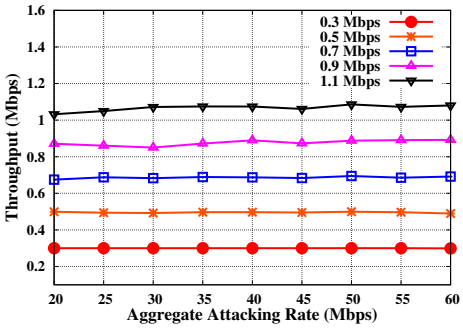
and 1.1 Mbps, are slightly less than their desired data rates. We attribute such slight performance degradation to the fact that TCP max–min fairness serves the low rate flows first in congested networks.



(a) Setting one.



(b) Setting two.



(c) Setting three.

Figure 11. AccFlow effectively defends against the SSTF attack.

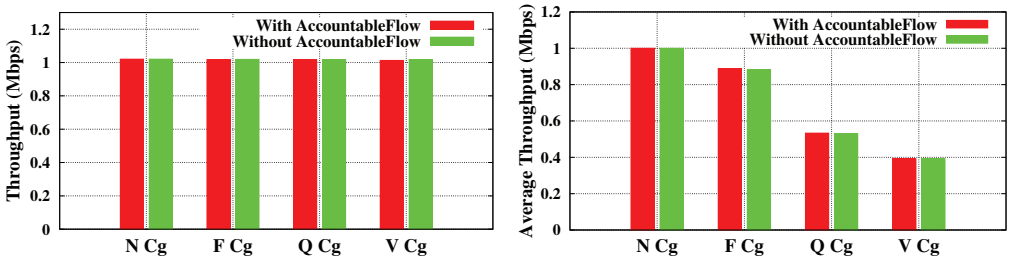
5.3. Benign Periodic Flow

Real-life networks, such as the Internet, also carry periodic or bursty flows whose traffic patterns are similar to that of the low-rate TCP DoS attacking flows. One example is that YouTube generates periodic traffic by loading chunks of a video with pauses between each chunk [34]. In this subsection, we show that AccFlow does not falsely drop the packets from benign periodic flows through the following four experiments. In the first experimental setup, five normal TCP flows each transmitting at a rate of 1 Mbps and one benign periodic flow are sharing the network resources. The analysis in [34] reveals that the peak rate for a typical YouTube flow ranges from hundreds of kilobytes to several megabytes. The interval of each video chunk ranges from hundreds of milliseconds to seconds. We set the peak rate and period of the periodic flow in our simulation to be 3 Mbps and 200 ms so that it can represent real video traffic. Since the bottleneck link bandwidth is 10 Mbps which is large enough to hold all the traffic, the first setup is congestion free. In the second setup, we create a fairly congested network by generating nine normal TCP flows and the same periodic flow. In the third setup, the network becomes quite congested by carrying 15 normal TCP flows and the same periodic flow. Finally, in the fourth setup, the network is very congested as 20 normal TCP flows and the same periodic flow traverse the bottleneck link.

The simulation results are illustrated in Figure 12. For clear presentation, we use characters “N”, “F”, “Q”, and “V” to represent words “Not”, “Fairly”, “Quite”, and “Very”, respectively. The character “Cg” represents the word “Congested”. Thus “F Cg” means that the network is fairly congested, which corresponds to the second setup. As illustrated in Figure 12a, the benign periodic flow achieves its desired throughput in all four setups no matter whether AccFlow is applied or not. Furthermore, AccFlow also does not have any negative effect on the normal TCP flows, as illustrated in Figure 12b. The results indicate

that AccFlow seamlessly coexists with benign flows, maintaining their performance without degradation. This successful coexistence can be attributed to the fundamental difference between benign periodic flows and attacking flows.

In benign periodic flows, the objective is not to overwhelm congested links, unlike attacking flows. Therefore, the network does not experience a high aggregate loss rate, and as a result, AccFlow does not implement its aggressive dropping policy. In situations where a legitimate bursty flow momentarily reaches a peak rate that might cause a high aggregate loss rate, triggering AccFlow to drop a few packets, the adverse impact does not propagate. Legitimate flows naturally adjust by entering retransmission timeouts after encountering packet losses. Consequently, the network becomes less congested, and the aggregate loss rate falls below the threshold Th_1.



(a) Benign Periodic Flow (b) Normal Flows

Figure 12. AccFlow harmoniously coexists with the benign periodic flow and normal flows without causing performance degradation.

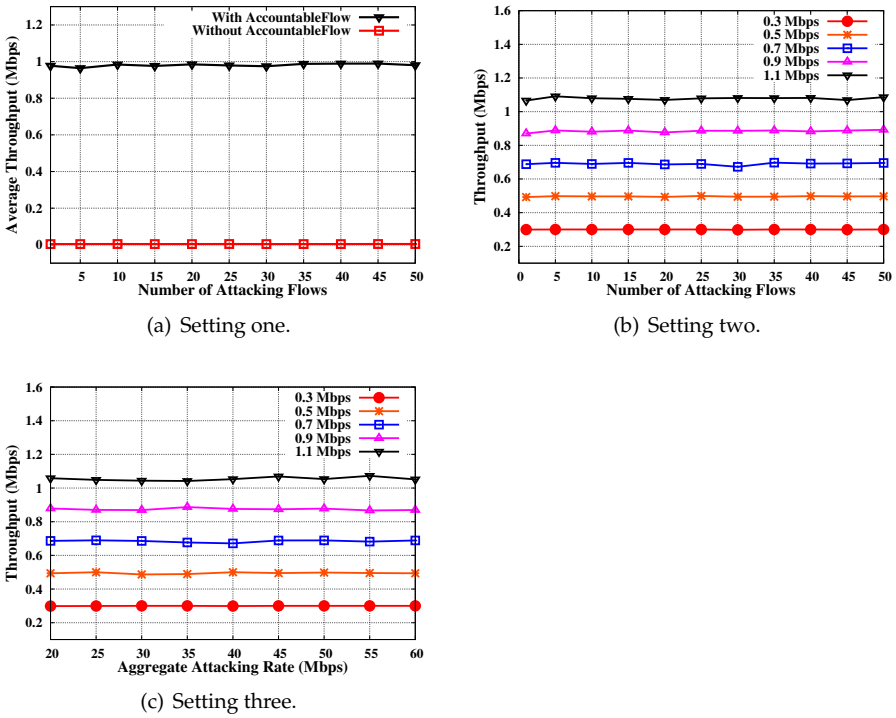
5.4. General (D)DoS Attack

Although AccFlow is primarily designed to address low-rate TCP DoS attacks, it also serves as a defense mechanism against general DoS attacks. The distinction between low-rate TCP DoS attacks and general DoS attacks lies in their respective methods of operation. The former relies on the TCP retransmission timeout mechanism to launch attacks, whereas the latter disrupts legitimate users' services by inundating the network with a high volume of traffic.

As previously mentioned, the central concept behind AccFlow is to hold attacking flows accountable for network congestion by promptly dropping their packets based on their loss rates. Consequently, any attacking flows that contribute to congestion are unable to disrupt legitimate flows by excessively utilizing network resources.

In fact, the effectiveness of AccFlow against general DoS attacks stems from our deliberate decision not to rely on the periodic nature of low-rate TCP DoS attacking flows when designing the algorithm.

To test the effectiveness of AccFlow when the network is faced with general DoS attacks, we use similar simulation settings listed in Table 1 except that the attackers consistently generate traffic without pause. As illustrated in Figure 13, AccFlow can effectively defend against the general DoS attacks. Furthermore, AccFlow also has a quick convergence time under such attacks, which is in the order of tens of seconds.



(a) Setting one.

(b) Setting two.

(c) Setting three.

Figure 13. AccFlow provides a strong defense against the general DoS attacks.

6. Conclusions

In recent years, the defense against low-rate TCP DoS attacks in SDN has garnered significant attention from researchers. However, the application of such defenses to drones has not been a focal point of research, and there are currently no related studies. This paper introduces AccFlow, an incrementally deployable SDN-based protocol, designed to counter both low-rate TCP DoS attacks and general DoS attacks in the context of drones. Consistent with previous studies of defending DDoS attacks where they propose an SDN-assisted DDoS attack defense method [35,36] and the DDoS Attack Detection System using Apache Spark [37], we concentrate on defending low-rate TCP DoS attacks on drones and still demonstrated effectiveness for normal DDoS. We deploy AccFlow on the SDN-centralized controller and use Aggressive Detection and Early Drop to protect the network when attackers try to evade detection by varying their attacking strategies. The main idea of AccFlow is to make the attacking flows accountable. Through substantial amounts of simulations in each setup, we demonstrate that AccFlow, which does not cause any performance degradation to benign flows, can effectively defend against both the low-rate TCP DoS attacks and the general DoS attacks even if attackers are able to vary their strategies.

Author Contributions: Conceptualization, Y.C. and X.Z.; Methodology, Y.C.; Software, X.P.; Validation, L.H.; Investigation, H.L.; Supervision, E.Y.; Funding acquisition, E.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by National Natural Science Foundation of China (62274056), Guangdong Basic and Applied Basic Research Foundation (2022A1515110045 and 2023A1515011241), the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province (SKLACSS-202209), Key Research and Development Program of Jiangsu Province (BE2022098), Postdoctoral Science Foundation of Jiangsu Province (2021K605C).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pärlin, K.; Alam, M.M.; Moullec, Y.L. Jamming of uav remote control systems using software defined radio. In Proceedings of the 2018 International Conference on Military Communications and Information Systems (ICMCIS), Warsaw, Poland, 22–23 May 2018; pp. 1–6.
2. Kardasz, P.; Doskocz, J.; Hejduk, M.; Wiejkut, P.; Zarzycki, H. Drones and possibilities of their using. *J. Civ. Environ. Eng.* **2016**, *6*, 1000233. [CrossRef]
3. Javaid, A.Y.; Sun, W.; Devabhaktuni, V.K.; Alam, M. Cyber security threat analysis and modeling of an unmanned aerial vehicle system. In Proceedings of the 2012 IEEE Conference on Technologies for Homeland Security (HST), Waltham, MA, USA, 13–15 November 2012; pp. 585–590.
4. Kuzmanovic, A.; Knightly, E.W. Low-rate tcp-targeted denial of service attacks and counter strategies. *IEEE/ACM Trans. Netw.* **2006**, *14*, 683–696. [CrossRef]
5. Stockhammer, T. Dynamic adaptive streaming over http–: Standards and design principles. In Proceedings of the Second Annual ACM Conference on Multimedia Systems, ACM, San Jose, CA, USA, 23–25 February 2011; pp. 133–144.
6. Sun, H.; Lui, J.C.S.; Yau, D.K.Y. Defending against low-rate tcp attacks: Dynamic detection and protection. In Proceedings of the 12th IEEE International Conference on Network Protocols, Berlin, Germany, 8 October 2004.
7. Chang, C.-W.; Lee, S.; Lin, B.; Wang, J. The taming of the shrew: Mitigating low-rate tcp-targeted attack. In Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems, Montreal, QC, Canada, 22–26 June 2009.
8. Wang, M.; Lu, Y.; Qin, J. Source-based defense against ddos attacks in sdn based on sflow and som. *IEEE Access* **2021**, *10*, 2097–2116. [CrossRef]
9. Bhuyan, M.; Kalwar, A.; Goswami, A.; Bhattacharyya, D.; Kalita, J. Low-rate and high-rate distributed dos attack detection using partial rank correlation. In Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, India, 4–6 April 2015; pp. 706–710.
10. Vishwakarma, R.; Jain, A.K. A honeypot with machine learning based detection framework for defending iot based botnet ddos attacks. In Proceedings of the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 23–25 April 2019; pp. 1019–1024.
11. Başer, M.; Güven, E.Y.; Aydın, M.A. Ssh and telnet protocols attack analysis using honeypot technique: Analysis of ssh and telnet honeypot. In Proceedings of the 2021 6th International Conference on Computer Science and Engineering (UBMK), Ankara, Turkey, 15–17 September 2021; pp. 806–811.
12. Tang, D.; Chen, J.; Wang, X.; Zhang, S.; Yan, Y. A new detection method for ldos attacks based on data mining. *Future Gener. Comput.* **2022**, *128*, 73–87. [CrossRef]
13. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. Openflow: Enabling innovation in campus networks. *ACM SIGCOMM* **2008**, *38*, 69–74. [CrossRef]
14. Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M.; et al. B4: Experience with a globally-deployed software defined wan. *ACM SIGCOMM* **2013**, *43*, 3–14. [CrossRef]
15. Hong, C.-Y.; Kandula, S.; Mahajan, R.; Zhang, M.; Gill, V.; Nanduri, M.; Wattenhofer, R. Achieving high utilization with software-driven wan. In Proceedings of the SIGCOMM '13: ACM SIGCOMM 2013 Conference on SIGCOMM, Hong Kong, China, 12–16 August 2013.
16. Shin, S.; Porras, P.; Yegneswaran, V.; Fong, M.; Gu, G.; Tyson, M. Fresco: Modular composable security services for software-defined networks. In Proceedings of the Network and Distributed Security Symposium, San Diego, CA, USA, 25–27 February 2013.
17. Liu, Z. FlowPolice: Enforcing Congestion Accountability to Defend against DDoS Attacks. Ph.D. Dissertation, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 2015.
18. Ns-3: A Discrete-Event Network Simulator. Available online: https://www.nsnam.org (accessed on 1 June 2021).
19. Aleksandar, K.; Edward, W.K. Low-rate TCP-targeted denial of service attacks: The shrew vs. the mice and elephants. In Proceedings of the SIGCOMM '03: 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Karlsruhe, Germany, 25–29 August 2003.
20. Ferguson, P.; Senie, D. *Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing*; RFC Editor: Marina del Rey, CA, USA, 2000.
21. Liu, X.; Li, A.; Yang, X.; Wetherall, D. Passport: Secure and adoptable source authentication. In Proceedings of the 5th USENIX Symposium on Networked Systems Design & Implementation, NSDI 2008, San Francisco, CA, USA, 16–18 April 2008.
22. Liu, Z.; Jin, H.; Hu, Y.-C.; Bailey, M. MiddlePolice: Toward Enforcing Destination-Defined Policies in the Middle of the Internet. In Proceedings of the CCS '16: 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016.
23. Liu, Z.; Jin, H.; Hu, Y.-C.; Bailey, M. MiddlePolice: Fine-Grained Endpoint-Driven In-Network Traffic Control for Proactive DDoS Attack Mitigation. *arXiv* **2017**, arXiv:1709.05710.
24. Mittal, P.; Kim, D.; Hu, Y.-C.; Caesar, M. Mirage: Towards deployable ddos defense for web applications. *arXiv* **2011**, arXiv:1110.1060.

25. Dixon, C.; Anderson, T.; Krishnamurthy, A. Phalanx: Withstanding multimillion-node botnets. In Proceedings of the NSDI 08: 5th USENIX Symposium on Networked Systems Design USENIX Association and Implementation, San Francisco, CA, USA, 16–18 April 2008.
26. Mahajan, R.; Bellovin, S.M.; Floyd, S.; Ioannidis, J.; Paxson, V.; Shenker, S. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM* **2002**, *32*, 62–73. [CrossRef]
27. Yang, X.; Wetherall, D.; Anderson, T. A dos-limiting network architecture. *ACM SIGCOMM* **2005**, *35*, 241–252. [CrossRef]
28. Chandra, T.; Griesemer, R.; Redstone, J. Paxos made live-an engineering perspective (2006 invited talk). In Proceedings of the 26th ACM Symposium on Principles of Distributed Computing-PODC, Portland, OR, USA, 12–15 August 2007; Volume 7.
29. Kent, S.; Lynn, C.; Seo, K. Secure border gateway protocol (s-bgp). *IEEE JSAC* **2000**, *18*, 582–592. [CrossRef]
30. Awduche, D.O.; Malcolm, J.; Agogbua, J.; O'Dell, M.D.; McManus. Requirements for Traffic Engineering over MPLS. *RFC 2702* **1999**, 1–29.
31. Hahne, E.L. Round-robin scheduling for max-min fairness in data networks. *IEEE J. Sel. Areas Commun.* **1991**, *9*, 1024–1039. [CrossRef]
32. Yaar, A.; Perrig, A.; Song, D. Stackpi: New packet marking and filtering mechanisms for ddos and ip spoofing defense. *IEEE J. Sel. Areas Commun.* **2006**, *24*, 1853–1863. [CrossRef]
33. Duan, Z.; Yuan, X.; Chandrashekar, J. Controlling ip spoofing through interdomain packet filters. *IEEE Trans. Dependable Secur. Comput.* **2008**, *5*, 22–36. [CrossRef]
34. Ameigeiras, P.; Ramos-Munoz, J.J.; Navarro-Ortiz, J.; Lopez-Soler, J.M. Analysis and modelling of youtube traffic. *Trans. Emerg. Telecommun. Technol.* **2012**, *23*, 360–377. [CrossRef]
35. Kiwon, H.; Youngjun, K.; Hyungoo, C.; Jinwoo, P. SDN-Assisted Slow HTTP DDoS Attack Defense Method. *IEEE Commun. Lett.* **2018**, *22*, 688–691.
36. Wisam, H.A.M. A hybrid scheme for detecting and preventing single packet Low-rate DDoS and flooding DDoS attacks in SDN. In Proceedings of the 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA), Benghazi, Libya, 21–23 May 2023; pp. 707–712
37. Heena, K.; Moin, M.M.; Pooja, S.; Narayan, D.G. DDoS Attack Detection System using Apache Spark. In Proceedings of the 2021 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 27–29 January 2021; pp. 1–5.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.