Strengthening Packet Loss Measurement from the Network Intermediate Point

Haoliang Lan[1*], Wei Ding[1] and YuMei Zhang[2]

[1]School of Cyber Science and Engineering, Southeast University

Nanjing, 211189 – CHINA

[email: hllan@njnet.edu.cn]

[2]School of Big Data and Information Engineering, Guizhou University

Guiyang, 550025 – CHINA

[email: yumeiz@yahoo.com]

*Corresponding author: Haoliang Lan

Received September 4, 2018; revised November 18, 2018; accepted May 18, 2019; published December 31, 2019

Abstract

Estimating loss rates with the packet traces captured from some point in the middle of the network has received much attention within the research community. Meanwhile, existing intermediate-point methods like [1] require the capturing system to capture all the TCP traffic that crosses the border of an access network (typically Gigabit network) destined to or coming from the Internet. However, limited to the performance of current hardware and software, capturing network traffic in a Gigabit environment is still a challenging task. The uncaptured packets will affect the total number of captured packets and the estimated number of packet losses, which eventually affects the accuracy of the estimated loss rate. Therefore, to obtain more accurate loss rate, a method of strengthening packet loss measurement from the network intermediate point is proposed in this paper. Through constructing a series of heuristic rules and leveraging the binomial distribution principle, the proposed method realizes the compensation for the estimated loss rate. Also, experiment results show that although there is no increase in the proportion of accurate estimates, the compensation makes the majority of estimates closer to the accurate ones.

Keywords: Packet loss rate, Capture rate, Estimation, Network performance management, Network measurement

1. Introduction

It is now common for network operators to perform ongoing performance measurements, the packet loss rate as a key performance metric is important for an ISP offering the services specified in Service-Level-Agreement (SLA). Currently, researchers mainly achieve packet loss estimation through active measurement or passive measurement. Active measurement [2] is the use of probe tools (such as Ping, BADABING, etc.) to inject probe packets, e.g., Internet Control Message Protocol (ICMP) echo packets, into the network and measure how many of them eventually reach the destination. The accuracy guarantee of the active measurement lies in selecting proper measurement point and injecting probe packets in an appropriate way without disturbing the target traffic. However, this is difficult in actual implementation as the active measurement is usually rate-based and does not share the sending pattern of the sender side. If the rate is too low, the real network performance may not be measured, while if the rate is too high, the target traffic will be disturbed by the probe traffic. Despite some random sampling models can reduce the effect of these disadvantages to some extent [3] [4] [5], active measurement is still difficult to cope with problems like discrete sampling nature of the probe process and dependence on the feedback loop. Therefore, although active measurement is simple and feasible, researchers are increasingly seeking to estimate packet loss through passive measurement.

All along, around passive packet loss measurement, research has focused on end-system measurement [6] [7] [8] [9] [10]. End-system methods have a common feature, viz., they are all based on the information available from the sender-side and/or receiver-side of a connection, which determines that they can only be used to evaluate the end-to-end performance of the individual connection. However, due to the traffic shaping and the events violating network neutrality [11] [12] [13], different connections may experience vastly different loss rates. From the perspective of network performance management [14], the packet loss rate obtained by end-system method is not suitable for evaluating the overall packet loss status of the monitored network. To this end, researchers explore to leverage the packet traces captured from the network intermediate point (e.g., at the boarder of an access network) to estimate packet losses. The intermediate-point methods can not only estimate the loss rates of individual end-to-end connection and aggregated traffic, but can also distinguish the packet losses within an ISP's management domain and that in the outside Internet. All these provide operators with the possibility to precisely grasp network performance to achieve fine-grained network management [15].

As connection-oriented and reliable transport protocol, TCP has a natural response to changes in network performance. Therefore, the research of intermediate-point methods has always been oriented towards TCP flows. For instance, Mellia *et al.* [16] built a tool Tstat for the first time to measure packet losses using TCP packet traces captured from the network intermediate point. For a TCP flow, Tstat simply takes retransmissions at the measurement point as the lost packets to estimate its packet losses. In the following years, due to its simplicity and practicability, Tstat is widely used in wired and wireless network performance evaluation and analysis [17] [18] [19] [20]. Meanwhile, due to the widespread application prospects of intermediate-point packet loss estimation, many scholars also embarked on the follow-up study of [16]. Benko *et al*. [21] used the traces captured from the network intermediate point to construct a series of heuristic rules to estimate packet losses before and after the measurement point. Favi *et al*. [22] analyzed the performance of the Benko-Veres algorithm [21] and found that the algorithm requires a considerable number of packets and lost-events to converge to a reasonable and accurate estimate. Jaiswal *et al*. [23] leveraged the information contained in the out-of-order packets to realize the estimation for packet losses. Collange *et al*. [24] estimated packet losses before and after the measurement point through the defined "Desequencements" data segment and retransmission mechanism. Moreover, with the proposed method, they also analyzed the relationship between packet loss rate and other traffic characteristics (e.g., connection time, connection size and the number of connection interruptions, etc.). Cheng *et al*. [25] extracted the information of fast retransmissions and timeout retransmissions contained in the data stream to realize the estimation for packet losses before and after the measurement point. With Retransmit TimeOut (RTO) and duplicate acknowledgments (ACKs), Ullah *et al*. [26] filtered the different types of retransmissions to estimate packet losses in the middle of the two ends. To troubleshoot packet losses, Cheng *et al*. [27] designed a lightweight efficient diagnosis tool TCPBisector to estimate packet losses in the middle of the network with relative error 3.5%-6.9%. In recent years, intermediate-point packet loss estimation has been introduced into Software Defined Network (SDN) applications, i.e., they combined with SDN to achieve packet loss monitoring [28] [29] [30], moreover, for good performance monitoring, Hark *et al*. [31] also evaluated the performance of these techniques to take the appropriate packet loss estimation technique in different network condition. In addition, Andrzej *et al*. [32] realized the packet loss estimation at the intermediate router by leveraging the stationary distribution of the queue size and the dropping function. While Sierra et al. [33] estimated packet losses in the network intermediate point by proposing a simplest approach of counting as a retransmission a packet whose sequence number is smaller than the previous one. Compared with the rich papers in this area, we only listed a part of them. The common feature of these methods is that they are all based on TCP packet traces captured from the boarder of the monitored network (typically Gigabit network). Therefore, the performance of the capturing system will directly affect the accuracy of such methods in practical applications. Actually, limited to the performance of current hardware (e.g., the processing power of the CPU) and software (e.g., the performance of the popular operating systems), capturing network traffic in a Gigabit environment is still a challenging task [34] [35] [36]. In other words, the capturing system is not able to capture everything. For instance, according to Schneider *et al*'s evaluation for the performance of capturing system under different combinations of hardware and software [37], the capture rate cannot reach 100% in most cases, and when the rate of traffic generation reaches 900 Mbit/s, even the best-performing FreeBSD/AMD combination still exhibited 10% to 20% packet losses. Although with the technological innovation of hardware and software, the capture rate has been improved in recent years, the problem of uncaptured packets in Gigabit environment is still inevitable as the rapid increment of the network traffic and transmission speed [38].

In our previous work [1], we enriched the body of intermediate-point estimation techniques by proposing an *A*lgorithm for *E*stimating *P*acket *L*osses on *N*etwork *P*ath (*AEPLNP*), which consists of two sub-algorithms. The first sub-algorithm estimates packet losses before the capture point by building a series of heuristic rules that aim to accurately reflect the state transitions of TCP congestion state machine in the sender-side associated with packet losses. At the same time, the factors like packet reordering, repeated packet losses, and so on were also considered to refine the first sub-algorithm in an effort to be more precise in estimation. While the second sub-algorithm realizes the estimate for packet losses after the capture point by distinguishing necessary retransmissions and spurious retransmissions with the information contained in the ACK stream. Therefore, besides the sender-side, the key for the second sub-algorithm is to leverage the state transitions in the receiver-side to locate the spurious retransmissions. Similar with other intermediate-point algorithms, for performance management reasons, the capture point of *AEPLNP* needs to be as close as possible to the boarder of an ISP's management domain (typically Gigabit network), and needs to capture all the TCP traffic that crosses the *entire* border destined to or coming from the Internet. Therefore, a problem, for *AEPLNP*, is that the TCP traffic cannot be fully captured and the missing packet information will affect: 1) the total number of captured packets; 2) the estimated number of packet losses before and after the capture point, which eventually affects the accuracy of the estimated loss rates. In view of this, this paper is intended to improve the robustness of *AEPLNP* in practical applications by detailing and validating a method with an eye towards overcoming the impact of uncaptured packets.

For packet capture, previous work [39] [40] [41] mainly focuses on how to improve the capture rate, and capture rate as an evaluation index is typically calculated by comparing the captured traffic with the original traffic that generated by different traffic generators [42] [43]. While about how to estimate the number of uncaptured packets with the captured packets, we looked up domestic and foreign literatures and found no relevant research about it. Therefore, in this work, we first explore leveraging the data and ACK streams to construct a series of heuristic rules to estimate the total number of uncaptured packets in a TCP connection. After that, we further leverage the estimated uncaptured packets combined with the binomial distribution principle to complete the compensation for the estimated number of packet losses before and after the capture point. Due to the impact of uncaptured packets, it is difficult to get an accurate loss rate with the packet traces captured in the actual network. Therefore in this paper, the simulations with practically relevant parameters are performed to evaluate the proposed method. And, the final experiment results show that although there is no increase in the proportion of accurate estimates, the compensation makes the majority of estimates closer to the accurate ones. In addition, what needs to be pointed out is that the previous work mainly focused on the intermediate-point algorithm itself and did not provide a solution to the problem of uncaptured packets in practical applications. From this angle, the contribution of this work not only lies in improving the robustness of *AEPLNP*, but also lies in providing a reference for other intermediate-point algorithms to overcome the impact of the uncaptured packets.

The remainder of this paper is organized as follow. The compensation strategy is introduced in Section 2. Next, Section 3 presents the algorithm for estimating the total number of uncaptured packets in a TCP connection. Then, Section 4 describes the compensation for the estimated number of packet losses. After detailing the proposed method, Section 5 presents the validation and analysis for the proposed method. Finally, Section 6 concludes the paper and gives the research forecast.

2. Compensation Strategy

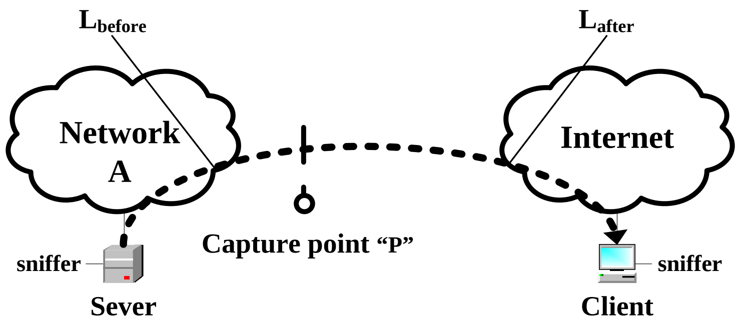


Fig. 1. Measurement scenario

The measurement scenario is shown in **Fig. 1**. As can be seen, the capture point "P" is located at the boarder of the monitored network A and divides each network path crossing through the boarder of A into two segments. The network path segments before "P" can be used to evaluate the packet loss status within A, while the network path segments after "P" can be used to evaluate the packet loss status in the outside Internet. Given that a TCP connection without compensation is denoted with *T*, then its packet loss rate on network path before "P" can be calculated as:

$$L_{before\_without} = \frac{N_{before}}{N_{before} + N_{capture}} \tag{1}$$

where N_{before} denotes the number of estimated packet losses of *T* before "P", and $N_{capture}$ denotes the number of data packets of *T* captured at "P".

Similarly, for *T*, the packet loss rate on network path after "P" can be calculated as:

$$L_{after\_without} = \frac{N_{after}}{N_{before} + N_{capture}} \tag{2}$$

where N_{after} denotes the number of estimated packet losses of T after "P".

It is obvious that the uncaptured packets will result in an underestimate for $N_{capture}$. Meanwhile, *AEPLNP* leverages the information extracted from the captured packets to estimate N_{before} and N_{after}. The uncaptured packets will also result in the loss of relevant information and thus result in an underestimate for N_{before} and N_{after}. Referring to equation (1) and equation (2), the accuracy of L_{before} and L_{after} will inevitably be affected. Therefore, to further improve the robustness of the algorithm *AEPLNP* in the Gigabit network environment, next the compensation for *AEPLNP* is detailed and validated. Concretely, the compensation has the following two steps:

Step1: Based on the seen sequence number pattern and the information contained in the ACK stream, the *A*lgorithm for *E*stimating *U*ncaptured *P*ackets (*AEUP*) consisting of a series of heuristic rules is first constructed to estimate the total number of uncaptured packets in a TCP connection, and thereby achieving the compensation for $N_{capture}$.

Step2: Combined with the binomial distribution principle, we complete the compensation for N_{before} and N_{after} with the estimated uncaptured packets in Step1.

3. Estimate the Number of Uncaptured Packets

In this section, the *A*lgorithm for *E*stimating *U*ncaptured *P*ackets (*AEUP*) is presented.

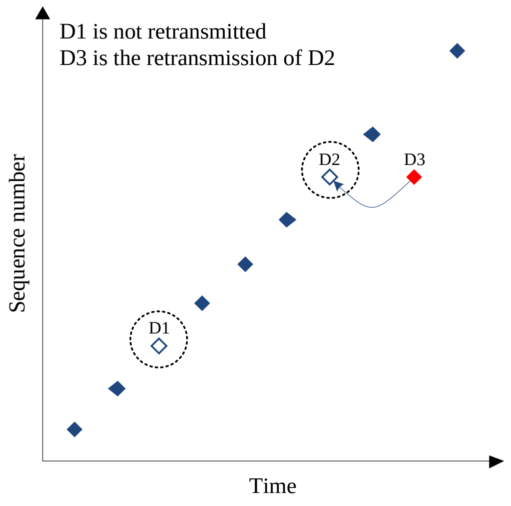


Fig. 2. Sample uncaptured packet pattern

Fig. 2 shows the data sequence sample of a TCP connection appearing at the capture point. Referring to **Fig. 2** and analyzing the type of TCP packets appearing at the capture point, we found that an uncaptured data packet will cause the following two cases:

Case 1: If a data packet appears at the capture point only once (e.g., D1 in **Fig. 2**) and is not captured, then *unfilled hole* will appear in the captured data sequence of a successfully ended TCP connection.

Case 2: Again, if a data packet appears at the capture point more than once, i.e., it is retransmitted (e.g., D2 in **Fig. 2**), in this case, even though it is not captured, we may not see any *visible hole* in the captured data sequence, viz., the hole caused by the uncaptured packet is refilled and becomes invisible (e.g., in **Fig. 2**, D2 is not captured, but D3 fills the hole caused by the lost D2).

Given that the data sequence of a TCP connection is described with set $S=\{<S_1, L_1>, \ldots, <S_n, L_n>\}$, where $\forall\ i \in N$, $<S_i, L_i>$ denotes the ith data segment, S_i denotes the sequence number of $<S_i, L_i>$, $S_i \leq S_{i+1}$, and L_i denotes the byte length of $<S_i, L_i>$. Then the unfilled hole and invisible hole in S are defined as follows.

Definition 1: (*Unfilled hole*) If $S_i+L_i<S_{i+1}$, then an unfilled hole appears between $<S_i, L_i>$ and $<S_{i+1}, L_{i+1}>$, we denote this unfilled hole with $H_{unfilled\_i}=<S_i+L_i, S_{i+1}>$.

Definition 2: (*Invisible hole*) If $<S_i, L_i>$ is not captured and appears at the capture point more than once, in this case, we denote the invisible hole caused by the loss of $<S_i, L_i>$ with $H_{invisible\_i}=<S_i, S_i+L_i>$.

According to the discussion above, if the actual number of data segments corresponding to the unfilled holes and invisible holes in S can be determined, then the number of uncaptured data packets can be calculated as:

$$N_{uncaptured} = N_{unfilled} + N_{invisible} \tag{3}$$

where $N_{unfilled}$ and $N_{invisible}$ represent the number of data segments corresponding to unfilled holes and invisible holes in S, respectively. Therefore, our goal is to seek $N_{unfilled}$ and $N_{invisible}$.

By definition, we can easily identity the unfilled holes in S by sorting the data packets in ascending order according to the sequence number. However, we also note that the byte range identified by the TCP sequence number is 0 to 2^{32}. Moreover, in a TCP connection, the sequence number of the first data segment is random. Thus, the sequence number may appear cyclic reuse during a TCP transfer, which can also cause unfilled hole in the data sequence. In order to exclude the effect of sequence number cyclic reuse, the maximum hole-size is set in this paper. Referencing the typical capture rate and default Maximum Segment Size (MSS) (typically 1460 bytes), the maximum unfilled hole-size is set to:

$$H_{max\_unfilled} = \frac{1}{10} N_{cap} * 1460 \tag{4}$$

where N_{cap} is the total number of captured data packets corresponding to a TCP connection. Although $N_{max\_unfilled}$ does not necessarily exclude the effect of sequence number cyclic reuse completely, it is expected to bound the error.

After identifying the unfilled holes in S, $N_{unfilled}$ can be determined with the default MSS and ACK stream.

For different TCP acknowledgement mechanisms, the form of ACK is different.

Accordingly, different techniques will be used to determine $N_{unfilled}$ in this paper. Currently there are three kinds of TCP acknowledgement mechanisms:

- ✦ **Standard Cumulative Acknowledgment (*SCA*):** According to RFC5681 [44], it cooperates with the basic TCP congestion control mechanisms to repair the packet losses (e.g., Reno, NewReno, BIC and CUBIC, etc.). This mechanism only acknowledges the data segments that arrive in order, i.e., the received segments that are not at the left edge of the receive window cannot be acknowledged. Therefore, for TCP connection taking this mechanism, extracting information contained in the ACK stream is not able to identify the edges of out-of-order arrivals. In this case, the default MSS can be used to assist in determining edges of uncaptured segments.
- ✦ **Selective Acknowledgment (*SACK*):** It combines with a selective retransmission policy at the sender-side to repair the packet losses and reduce the spurious retransmissions. This mechanism not only acknowledges the data segments that arrive in order, but can also acknowledge the out-of-order segments that have arrived at the receiver and not covered by the acknowledgement number. Therefore, for TCP connection taking this mechanism, the information contained in the SACK blocks is beneficial for determining the edges of partial out-of-order arrivals. More details about SACK are available from RFC2018 [45].
- ✦ **Duplicate Selective Acknowledgment (*D-SACK*):** This version, described in RFC2883 [46], is an extension to the SACK. It allows the receiver to inform the sender about segments that have already arrived more than once. Therefore, a spurious retransmission can be accurately identified with an ACK containing D-SACK information. As we know, since the flaws in TCP's retransmission schemes [47], spurious retransmissions are inevitable in a number of cases. On the basis of SACK, the information in DSACK blocks can not only specify the segments that arrive in order and out-of-order, but can also specify the redundant segments caused by spurious retransmissions. Therefore, DSACK combined with the various holes defined in this paper can comprehensively determine the edges of the uncaptured segments.

For easy discussion, we redefine the ACK number here. For SCA, the ACK number refers to acknowledgement number (hereinafter referred to as ack-number), while for SACK and D-SACK, the ACK number includes ack-number, SACK block left edge (hereinafter referred to as left-edge) and SACK block right edge (hereinafter referred to as right-edge). For ack-number A, when we speak of it falls into the unfilled hole $H_{unfilled\_i}$, we are referring to that $S_i+L_i<A\leq S_{i+1}$, and for left-edge L, when we speak of it falls into the unfilled hole $H_{unfilled\_i}$, we are referring to that $S_i+L_i\leq L<S_{i+1}$, while for right-edge R, when we speak of it falls into the unfilled hole $H_{unfilled\_i}$, we are referring to that $S_i+L_i<R<S_{i+1}$. According to whether or not an unfilled hole has an ACK number to fall into, it can be divided into the following categories.

Definition 3: (*Pure hole*) It has no any ACK number to fall into.

Definition 4: (*Impurity hole*) It has one or more ACK number(s) to fall into.

Definition 5: (*Overlapped hole*) It has duplicate ACK number to fall into.

Indeed, by definition, the overlapped hole belongs to impurity hole. In order to facilitate the calculation for $N_{unfilled}$, we further divide the impurity hole into disjoint categories.

Definition 6: (*Normal hole*) Among the impurity holes, except the overlapped holes, the remaining are normal holes.

According to the discussion above, eventually the unfilled hole is divided into the following three disjoint categories: pure hole, normal hole and overlapped hole.

$$\text{Unfilled hole}\begin{cases}\blacklozenge\ \text{Pure hole}\\ \text{Impurity hole}\begin{cases}\blacklozenge\ \text{Normal hole}\\ \blacklozenge\ \text{Overlapped hole}\end{cases}\end{cases}$$

For pure hole, because it has no any ACK number to fall into, so *AEUP* uses the default MSS to determine the number of data segments corresponding to the pure holes in *S*:

$$N_{pure} = \sum_{i=1}^{p}\left\lceil\frac{P_i}{1460}\right\rceil \tag{5}$$

where p is the number of pure holes in *S*, and P_i is the byte length of the ith pure hole.

For normal hole, it has no any duplicate ACK number to fall into. Thus, we just need to leverage the ACK numbers falling into it and the default MSS to determine the edges of the data segments corresponding to it. Accordingly, *AEUP* first uses ACK numbers falling into a normal hole to divide it into some small fragments. After the first round of dividing, if the byte length of the divided fragment is greater than 1460, then it will be further randomly divided into *F* smaller fragments whose byte length is greater than 1. *F* equals to the byte length of the divided fragment divided by 1460. Eventually, *AEUP* uses the number of final divided fragments as the number of data segments corresponding to the normal holes in *S*:

$$N_{nomal} = \sum_{i=1}^{n}\left[F_i + B_i\sum_{j=1}^{F_i}\left(\left\lceil\frac{L_j}{1460}\right\rceil - 1\right)\right] \tag{6}$$

where n is the number of normal holes in *S*, F_i is the number of fragments in the ith normal hole after the first round of dividing, L_j is the byte length of the jth divided fragment belonging to the ith normal hole after the first round of dividing, and B_i is a Boolean variable that reports the situation of the ith normal hole after the first round of dividing.

$$B_i = \begin{cases}1\text{:if the i}^{\text{th}}\text{ normal hole contains fragment that is greater than 1460 bytes}\\ 0\text{:otherwise}\end{cases}$$

For overlapped hole, compared with normal hole, the difference is that it has duplicate ACK numbers to fall into. For these duplicate ACK numbers, what do they represent is important for us to determine the number of data segments corresponding to the overlapped holes in *S* ($N_{overlapped}$). It should be noted that the sole function of the right-edge in this section is just to identify the edge of data segment, so when we discuss duplicate ACK number, the duplicate right-edge is not included. Since the duplicate ACK number may be duplicate ack-number or duplicate left-edge, we discuss the overlapped holes for SCA, SACK and D-SACK, respectively.

For SCA, about duplicate ack-number, we can prove the following:

Proposition 1: The duplicate ack-number is caused by either spurious retransmission or packet reordering.

Proof: In order to let TCP sender sent the data segment *D* that is suspected to be lost as soon as possible to avoid RTO expiration, the fast retransmission mechanism requires the receiver to immediately generate an ACK for expecting *D* upon receiving an out-of-order data segment. Therefore, if the received data is out-of-order, then a duplicate ack-number will be generated. In contrary, if the received data is orderly and non-retransmitted, then it will cause a new ACK. Else if the received data is orderly and necessary retransmission, it will also cause a new ACK, but if it is orderly and spurious retransmission, the duplicate ack-number will be generated since the buffer state of the receiver-side is not changed.

Therefore, for an overlapped hole, we first use the method dealing with the normal hole to determine the number of non-retransmitted data segments in it. Then the number of spurious retransmissions specified by some (not all) duplicate ACK numbers falling into the overlapped hole can be determined after excluding the effect of packet reordering. As we know, the essence that packet reordering can generate duplicate ACK is fast retransmission mechanism. Again, sufficient packet reordering will cause fast retransmission (typically over three duplicate ACKs). Therefore, for a duplicate ACK, if the number of times it appears at the capture point is greater than or equal to 4, we assume it is caused by packet reordering. Ultimately, for SCA, $N_{overlapped}$ can be calculated as:

$$N_{overlapped} = \sum_{i=1}^{o}\left[F_i + B_i \sum_{j=1}^{F_i}\left(\left\lceil\frac{L_j}{1460}\right\rceil - 1\right) + D_i - R_i\right] \tag{7}$$

where o is the number of overlapped holes in S, D_i is the number of duplicate ack-numbers falling into the ith overlapped hole, R_i is the number of duplicate ack-numbers caused by packet reordering in D_i.

For SACK, about duplicate ACK numbers, we can prove the following:

Proposition 2: The duplicate ack-number is caused by either spurious retransmission or packet reordering, but the duplicate left-edge is caused by spurious retransmission.

Proof: Just like the proof for SCA, for SACK, a duplicate ack-number is caused by either spurious retransmission or packet reordering. While for SACK block, the receiver always acknowledges the most recently transmitted ones [48]. Therefore, if the left-edge is duplicate, we can determine that it is caused by a spurious retransmission.

According to proposition 2, when determining $N_{overlapped}$, duplicate left-edge falling into the overlapped hole can be directly used to confirm a spurious retransmission. While for duplicate ack-number falling into overlapped hole, if the ACK where it is located doesn't contains duplicate left-edge, we can think it may be caused by a spurious retransmission and use it to determine a spurious retransmission, but need to exclude the effect of packet reordering, viz., we require the number of times it appears at "P" is less than or equal to 3. Therefore, for SACK, $N_{overlapped}$ can be calculated as:

$$N_{overlapped} = \sum_{i=1}^{o} \left[F_i + B_i \sum_{j=1}^{F_i} \left(\left\lceil \frac{L_j}{1460} \right\rceil - 1 \right) + D_i + E_i \right] \tag{8}$$

where D_i is the number of duplicate ack-numbers identifying spurious retransmissions and falling into the ith overlapped hole, and E_i is the number of duplicate left-edges falling into the ith overlapped hole.

Actually, for our work, the only valuable information that a duplicate ACK number can provide is whether it specifies a spurious retransmission or not. In this case, for D-SACK, we don't take the duplicate ACK numbers into account when determining $N_{overlapped}$. But we use the DSACK block to detect the spurious retransmission belonging to the overlapped hole, viz., if the left-edge falling into the overlapped hole is a DSACK block left edge, we add the number of spurious retransmissions belonging to this overlapped hole by 1.

Therefore, for D-SACK, $N_{overlapped}$ can be calculated as:

$$N_{overlapped} = \sum_{i=1}^{o} \left[F_i + B_i \sum_{j=1}^{F_i} \left(\left\lceil \frac{L_j}{1460} \right\rceil - 1 \right) + E_i \right] \tag{9}$$

where E_i is the number of DSACK block left edges falling into the ith overlapped hole.

Based on the discussion above, $N_{unfilled}$ is calculated as:

$$N_{unfilled} = N_{pure} + N_{normal} + N_{overlapped} \tag{10}$$

And, $N_{invisible}$ can be denoted as:

$$N_{invisible} = N_{rts} - N_{cap\_rts} \tag{11}$$

where N_{rts} denotes the total number of retransmissions appearing at the capture point, and N_{cap_rts} denotes the number of retransmissions actually captured by the capturing system.

According to equation (11), the key to obtain $N_{invisible}$ lies in determining N_{rts}. To this end, we leverage the conclusion in [49] to approximate N_{rts}. For illustrative purposes, we introduce the following metric:

Ratio$_{\text{data/ack}}$: the ratio of the number of data packets to the number of ACKs.

As Wu *et al*. [49] point out, for a TCP connection, Ratio$_{\text{data/ack}}$ is about 2:1 when there is no packet loss. On the contrary, Ratio$_{\text{data/ack}}$ will change from 2:1 to 1:1 until the lost packets are repaired. Therefore, we have:

$$\begin{cases} N_{ack} = N_{rts} + \dfrac{N_{non\_rts}}{2} \\ N_{non\_rts} = N_{non\_unfilled} + N_{non\_cap} \end{cases} \tag{12}$$

where N_{ack} is the number of ACKs, N_{non_trs} is the number of non-retransmitted segments, $N_{non\_unfilled}$ is the number of non-retransmitted segments in unfilled holes and N_{non_cap} is the number of captured non-retransmitted segments. Solve equation (12), we have:

$$N_{rts} = \left\lfloor N_{ack} - \frac{N_{non\_unfilled} + N_{non\_cap}}{2} \right\rfloor \tag{13}$$

Finally, the algorithm *AEUP* is given in Algorithm 1.

Algorithm 1: *AEUP*

```
//Preprocessing stage
Pure_h = Normal_h = Overlapped_h = Invisible_h = 0
for pkt in trace  of one TCP connection
    if pkt.is_Data() then
        H = pkt.form_Hole()
    else
        A = pkt.form_Ack()
    end if
end for
for hole in H
    if hole.pure_Check(A)
        Pure_h = hole.pure_Num()
    else if hole.normal_Check(A)
        Normal_h = hole.normal_Num(A)
    else
        Overlapped_h = hole.overlapped_Num(A)
end for
        Invisible_h = hole.invisible_Num(A)
Return Pure_h + Normal_h + Overlapped_h + Invisible_h
```

4. Compensate the Estimated Number of Packet Losses

To simplify discussion, we introduce the following identifiers:

S: the uncaptured data segments in a TCP connection

S_{non_r}: the non-retransmitted data segments in a TCP connection

S_r: the retransmitted data segments in a TCP connection

S_b: the data segments that are related to packet losses before the capture point

S_{n_b}: the data segments that are not related to packet losses before the capture point

S_a: the data segments that are related to packet losses after the capture point

S_{n_a}: the data segments that are not related to packet losses after the capture point

S_{b_s}: $S_b \cap S$

S_{b_n}: $S_b \cap \neg S$

$S_{n\_b\_s}$: $S_{n\_b} \cap S$

$S_{n\_b\_n}$: $S_{n\_b} \cap \neg S$

S_{a_s}: $S_a \cap S$

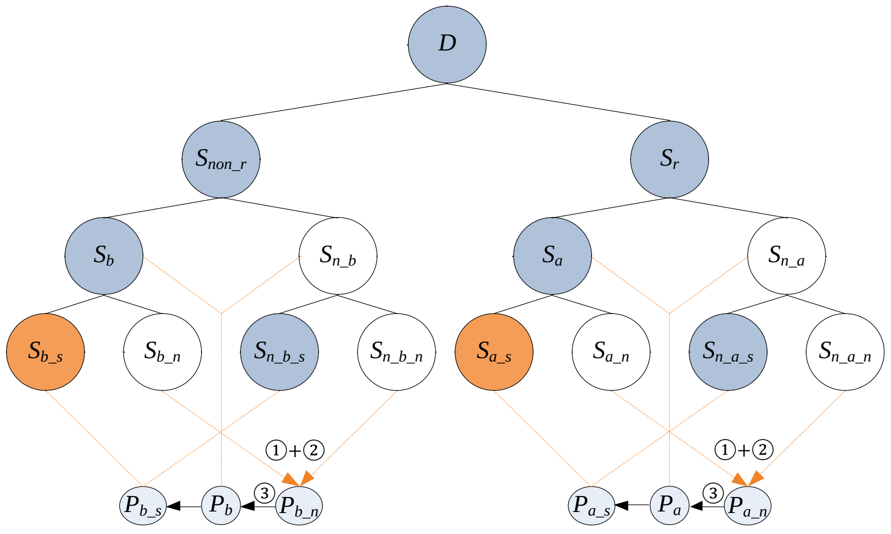
S_{a_n}: $S_a \cap \neg S$

$S_{n\_a\_s}$: $S_{n\_a} \cap S$

$S_{n\_a\_n}$: $S_{n\_a} \cap \neg S$

S_{non_s}: $S_{non\_r} \cap S$

S_{non_ns}: $S_{non\_r} \cap \neg S$
S_{r_s}: $S_r \cap S$
S_{r_ns}: $S_r \cap \neg S$
P_b: the probability that a segment in S_{non_r} belongs to S_b
P_{b_s}: the probability that a segment in S_{non_s} belongs to S_{b_s}
P_{b_n}: the probability that a segment in S_{non_ns} belongs to S_{b_n}
P_a: the probability that a segment in S_r belongs to S_a
P_{a_s}: the probability that a segment in S_{r_s} belongs to S_{a_s}
P_{a_n}: the probability that a segment in S_{r_ns} belongs to S_{a_n}
N_{non_s}: card(S_{non_s})
N_{non_ns}: card(S_{non_ns})
N_{r_s}: card(S_{r_s})
N_{r_ns}: card(S_{r_ns})
N_{b_s}: card(S_{b_s})
N_{b_n}: card(S_{b_n})
N_{a_s}: card(S_{a_s})
N_{a_n}: card(S_{a_n})



① Binomial distribution
② Maximum likelihood estimation
③ Large sample theory of maximum likelihood estimation

Fig. 3. Categories of data segments in a TCP transfer

With the defined identifiers, we divide the data segments in a TCP connection (denoted with D) into several categories shown in **Fig. 3**. As can be seen, N_{b_s} and N_{a_s} are the compensation numbers of N_{before} and N_{after}, respectively. Therefore, the goal becomes to seek N_{b_s} and N_{a_s}, and our methodology has the following two steps:

Step1: Leverage binomial distribution and maximum likelihood estimation to get P_{b_s} and P_{b_n} (P_{a_s} and P_{a_n}).

Step2: Let P_{b_s}=P_{b_n} (P_{a_s}=P_{a_n}) to obtain N_{b_s} (N_{a_s}).

In S_{non_ns}, one segment either belongs to S_{b_n} or to $S_{n\_b\_n}$. We use a random variable X to denote one segment belongs to S_{b_n} or to $S_{n\_b\_n}$. Let

$$\mathrm{X}=\begin{cases}0\text{: denotes one segment belongs to } S_{b\_n} \\ 1\text{: denotes one segment belongs to } S_{n\_b\_n}\end{cases},$$

then X ~ b (1, P_{b_n}). For a data flow, we take sample S_{non_ns} to observe how many segments belong to S_{b_n} and how many segments belong to $S_{n\_b\_n}$. Here, S_{non_ns} is recorded as (x_1, …, x_n), the probability of this observation event is:

$$P(\mathrm{X}_1 = x_1, \dots, \mathrm{X}_n = x_n; P_{b\_n}) = \prod_{i=1}^{n} P_{b\_n}{}^{x_i} * (1 - P_{b\_n})^{1-x_i}$$

$$= P_{b\_n}{}^{\sum x_i} * (1 - P_{b\_n})^{\mathrm{n}-\sum x_i} \tag{14}$$

According to the maximum likelihood principle, maximum likelihood principle we choose P_{b_n} to make equation (14) maximum. Equation (14) can be further expressed as:

$$\mathrm{L}(P_{b\_n}) = P_{b\_n}{}^{\sum x_i} * (1 - P_{b\_n})^{n-\sum x_i} \tag{15}$$

Equation (15) is transformed into the following form:

$$\ln \mathrm{L}(P_{b\_n}) = \ln\left[P_{b\_n}{}^{\sum x_i} * (1 - P_{b\_n})^{n-\sum x_i}\right] \tag{16}$$

Derive the derivative of equation (16) with respect to P_{b_n} and let its result be zero:

$$\frac{\partial \ln \mathrm{L}(P_{b\_n})}{\partial P_{b\_n}} = \frac{\partial \ln\left[P_{b\_n}{}^{\sum x_i} * (1 - P_{b\_n})^{n-\sum x_i}\right]}{\partial P_{b\_n}} = 0 \tag{17}$$

Solve equation (17) to get the maximum likelihood estimation of P_{b_n}:

$$\widehat{P_{b\_n}} = \widehat{P_{b\_n}}(x_1,\dots,x_n) = \frac{\sum x_i}{n} \tag{18}$$

Next, we prove $\widehat{P_{b\_n}}$ is unbiased:

$$\mathrm{E}(\widehat{P_{b\_n}}) = \mathrm{E}\left(\frac{\sum x_i}{n}\right) = \frac{1}{n}\mathrm{E}(\sum x_i) = \frac{1}{n} n\widehat{P_{b\_n}} = \widehat{P_{b\_n}} \tag{19}$$

Thus, the unbiasedness of $\widehat{P_{b\_n}}$ is proved. Similarly, P_{b_s} can be obtained.

As we know, $S_{non\_s} \subset S_{non\_r}$, so $\widehat{P_{b\_s}}$=P_b. Again, since $S_{non\_ns} \subset S_{non\_r}$ and card(S_{non_ns})>card(S_{non_r}), according to the large sample theory of maximum likelihood estimation, we have $\widehat{P_b}$=P_{b_n}. Because $\widehat{P_{b\_s}}$=P_b and $\widehat{P_b}$=P_{b_n}, so $\widehat{P_{b\_s}}$=P_{b_n}. After obtaining P_{b_s} and P_{b_n}, N_{b_s} can be calculated as follows:

$$\begin{cases} P_{b\_s} = \dfrac{N_{b\_s}}{N_{non\_s}} \\ P_{b\_n} = \dfrac{N_{b\_n}}{N_{non\_ns}} \\ P_{b\_s} = P_{b\_n} \end{cases} \xrightarrow{\text{yields}} N_{b\_s} = \left\lfloor \frac{(N_{uncaptured} - N_{r\_s}) * N_{before}}{N_{cap} - N_{cap\_rts}} \right\rfloor \tag{20}$$

where $N_{uncaptured}$ denotes the number of uncaptured segments estimated by *AEUP*. Similarly, N_{a_s} can be calculated as follows:

$$\begin{cases} P_{a\_s} = \dfrac{N_{a\_s}}{N_{non\_s}} \\ P_{a\_n} = \dfrac{N_{a\_n}}{N_{r\_ns}} \\ P_{a\_s} = P_{a\_n} \end{cases} \xrightarrow{\text{yields}} N_{a\_s} = \left\lfloor \frac{N_{after} * N_{r\_s}}{N_{cap\_trs} - N_{after}} \right\rfloor \tag{21}$$

5. Validation and Analysis

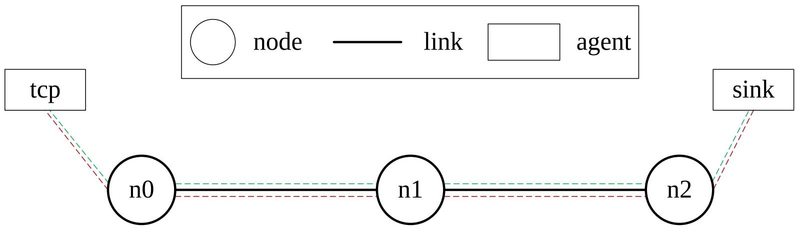


Fig. 4. Simulation scenario

Table 1. Simulation parameters

Protocols	NewReno, SACK, D-SACK
Proportion delay	10 ms
File size	100 Mbytes
Link rate	1 Mbps
Packet size	uniformly distributed between 0.2 and 1.5 Kbytes
Loss rates before/after “P”	gradually from 0.1% to 10%
Capture rates	99.9%, 99% and 90%

5.1 Validation for Compensation

To evaluate the proposed method, we implemented and tested it on packet traces obtained from simulation. The simulation was carried out using Network Simulator (NS-2) [50]. The simulated scenario, shown in **Fig. 4**, consists of three nodes: n0 (server), n1 (capture point) and n2 (client), and the common simulation parameters are shown in **Table 1**. During simulation, total 30 TCP transfers (10 for SCA, 10 for SACK and 10 for D-SACK) were scheduled between the server and the client to transfer a fixed-size file. For each transfer, packets were dropped independently and with equal probability on network path segments before and after “P”. During transfer, we first collected packet traces from n2 and then randomly discarded some ones to simulate the situation that the capturing system is not able to capture everything. This allows us to study how the proposed method behaves when capturing system is not able to capture everything and experiences different capture rates. To test the improvement effect of the proposed method on *AEPLNP*, we run *AEPLNP* with Compensation (*AEPLNP-C*) and pure *AEPLNP* on the remaining traces, respectively. Correspondingly, the relative error is used to evaluate the accuracy of the compensation, and it is calculated as the absolute difference between the estimated loss rate and the accurate loss rate divided by the accurate loss rate:

$$Error_{relative} = \frac{|L_{estimated} - L_{accurate}|}{L_{accurate}} \tag{22}$$

where $L_{estimated}$ denotes the estimated loss rate, and $L_{accurate}$ denotes the accurate loss rate.

After the compensation, the estimated loss rates of a TCP connection on network paths before and after “P” can be calculated as:

$$L_{before\_with} = \frac{N_{before} + N_{b\_s}}{N_{cap} + N_{before} + N_{uncaptured} + N_{b\_s}} \tag{23}$$

$$L_{after\_with} = \frac{N_{after} + N_{a\_s}}{N_{cap} + N_{before} + N_{uncaptured} + N_{b\_s}} \tag{24}$$

Finally, the simulation results of the compensation are shown in **Fig. 5**. Specifically, the first column consisting of **Fig. 5(a)**, **Fig. 5(c)** and **Fig. 5(e)** shows the compensation results for packet loss before the capture point, while the second column consisting of **Fig. 5(b)**, **Fig. 5(d)** and **Fig. 5(f)** shows that for after the capture point. In each row, the compensation effect under different capture rate can be seen. The first row is under capture rate of 99.9%, while the second and the third are under capture rate of 99% and 90%, respectively. In the plots, different colored lines represent the algorithms under different TCP acknowledgement mechanisms, while the solid and dotted lines of the same color represent algorithms *AEPLNP-C* and *AEPLNP*, respectively.

5.2 Result Analysis

As can be seen, on the whole, although there is no increase in the proportion of accurate estimates, the compensation makes the majority of estimates closer to the accurate ones. Comparing the graphs in the same column from top to bottom, we can see that with the decline of the capture rate, *AEPLNP-C* is more and more superior to *AEPLNP*. Specifically, when the capture rate is high, e.g., 99.9% of **Fig. 5(a)** and **Fig. 5(b)**, the relative errors of L_{before_with} and $L_{before\_without}$ are almost the same and stay below 10%, while the relative errors of L_{after_with} and $L_{after\_without}$ are also almost the same and around 10%. In this case, the compensation has no obvious effect on *AEPLNP*. With the further decline in capture rate, e.g., 99% of **Fig. 5(c)** and **Fig. 5(d)**, whether for L_{before_with} and $L_{before\_without}$ or L_{after_with} and

$L_{after\_without}$, their relative errors all increase slightly at the majority of the parameter space. However, even in this case, *AEPLNP-C* still exhibits better performance compared with pure *AEPLNP*. When the capture rate declines to a certain value, e.g., 90% of **Fig. 5(e)** and **Fig. 5(f)**, the relative errors of $L_{before\_without}$ and $L_{after\_without}$ are significantly higher than that of L_{before_with} and L_{after_with}, respectively. This illustrates that *AEPLNP-C* is more robust and therefore significantly superior to pure *AEPLNP* in the face of lower capture rate. After having a general description for the experiment results, we systematically analyze them from the following three aspects.

(a) Impact of capture rate

From **Fig. 5(a)** → **Fig. 5(c)** → **Fig. 5(e)**, and **Fig. 5(b)** → **Fig. 5(d)** → **Fig. 5(f)**, we can see that with the decrease in capture rate, the compensation appreciably improves the accuracy of loss estimation. Indeed, with the decrease in capture rate, the loss rate will be underestimated due to the increased uncaptured packets. In this case, the introduction of $N_{uncaptured}$, N_{b_s} and N_{a_s} made both the total number of captured packets and the estimated number of packet losses closer to their respective actual values to improve the underestimation obviously.

(b) Impact of packet loss rate

To fully verify the compensation effect, different packet loss rates were introduced. As can be seen, especially in **Fig. 5(e)** and **Fig. 5(f)**, the lower packet loss rates outperform the higher ones in terms of compensation effect when capture rate is fixed. Actually, it can be inferred that with the packet loss rate becomes higher, the more likely the uncaptured packet contains the information indicating packet losses. When the loss of such information increases as the increase of the packet loss rate, it will become difficult to conduct effective compensation, which skews our compensation results.

(c) Impact of TCP acknowledgement mechanism

As we can see, especially from **Fig. 5(e)** and **Fig. 5(f)**, on the whole, the compensation achieved better performance on SACK and D-SACK transfers. This can be attributed to the good performance of SACK and D-SACK in handling spurious retransmissions. D-SACK accurately identifies spurious retransmissions with the help of DSACK blocks, while the duplicate left edge also makes the SACK realize a good detection for spurious retransmissions. On the contrary, for SCA, setting the threshold to 3 may be too strict to effectively detect the spurious retransmissions. Moreover, for SACK and D-SACK, the left-edge and right-edge also provide more information to help accurately determine the edges of the uncaptured data segments.

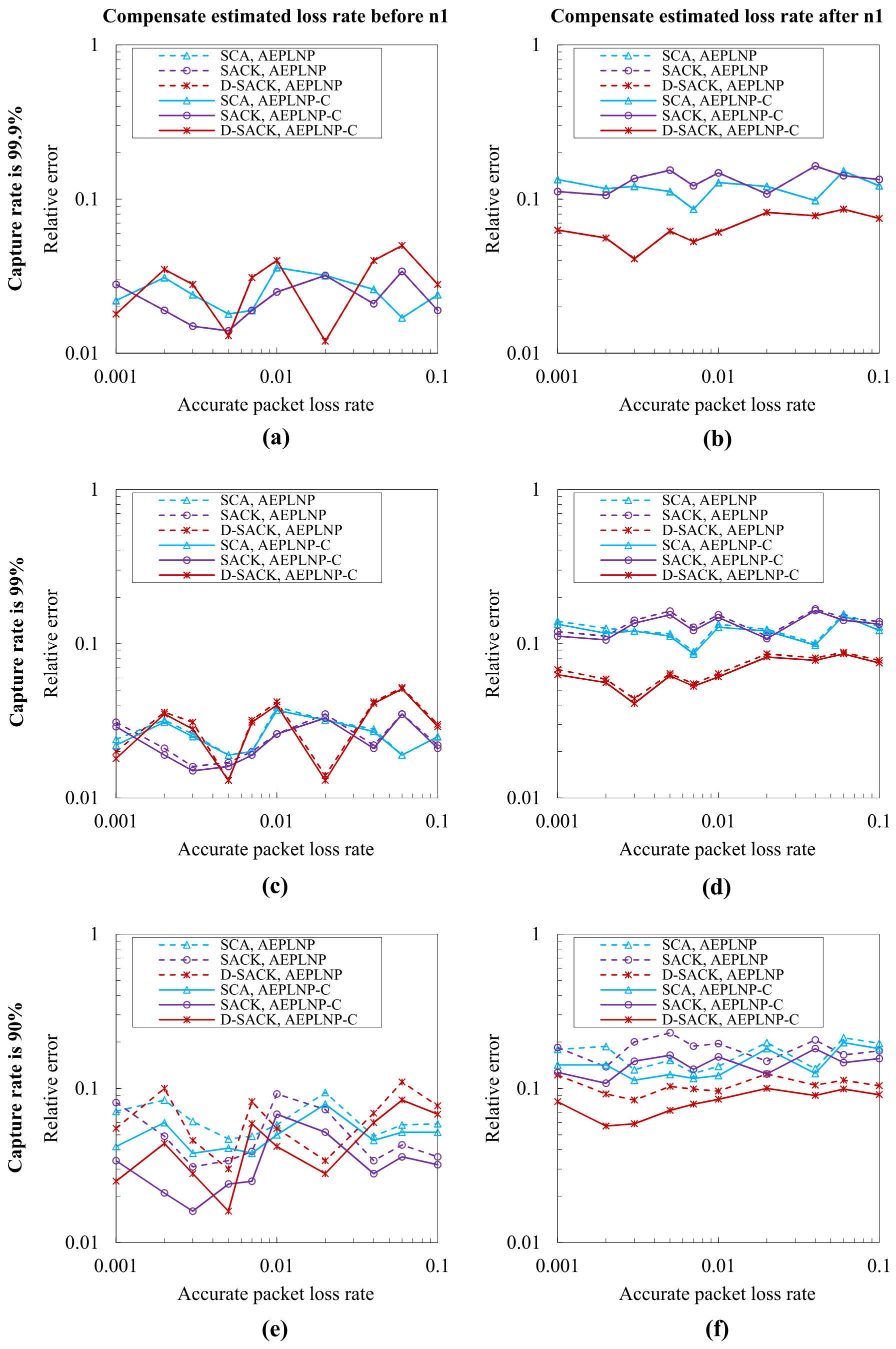


Fig. 5. Simulation results of the compensation

5.3 Error Sources

For the compensation errors, we found the following three factors can account for to some extent.

- **Strict segment size:** As we know, the byte length of data segment is not always equal to 1460. If it is less than 1460, the calculation accuracy of $N_{unfilled}$ will be affected.
- **Packet reordering:** Although rule was used to exclude the effect of packet reordering, but the effect can only be limited and cannot be completely eliminated.
- **Lost ACKs:** If an ACK is lost, the data segment specified by this ACK may not be identified.

According to the error sources listed above, one may be able to design a better algorithm by being more careful. That is, the error sources listed above provide a direction for further improving our algorithm. For instance, the calculation accuracy for $N_{unfilled}$ can be further improved by replacing the static segment size (1460 bytes) with a dynamic rule that depends on the distribution of the segment size in a TCP data sequence. Similarly, using a dynamic threshold obtained from the distribution of duplicate ACK numbers may also produce more complete and effective rules for detecting packet reordering.

6. Conclusion

The purpose of our study is to explore how to measure packet loss from the network intermediate point in a more robust way when facing with uncaptured packets that are inevitable in actual network. To this end, based on our previous work [1], a method of strengthening packet loss measurement from the network intermediate point is proposed and validated in this paper. Through constructing a series of heuristic rules and leveraging the binomial distribution principle, the proposed method realized the compensation for *AEPLNP*. And, the experimental results show that when capture rate is higher, the estimation accuracy of *AEPLNP-C* and *AEPLNP* is similar, but with the capture rate becomes lower, the former is obviously superior to the latter. Besides the capture rate, we also compared *AEPLNP-C* and *AEPLNP* by considering the packet loss rate and TCP acknowledgement mechanism. Similarly, although *AEPLNP-C* exhibited different performance under different packet loss rates and acknowledgement mechanisms, it is still clearly preferred over *AEPLNP* at the majority of parameter space. In addition, while *AEPLNP-C* enables relatively superior accuracy when facing with uncaptured packets, its accuracy is still expected to be further improved by overcoming the effect of the error sources listed in subsection 5.3.

Acknowledgements

This work was sponsored by the National High-Tech R&D Program of China (863) (2015AA015603); the National Nature Science Foundation of China (61602114).

References

[1] H. Lan, W. Ding and Y. Zhang, "Passive overall packet loss estimation at the border of an ISP," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 7, pp. 3150-3171, July, 2018. Article (CrossRefLink).

[2] F. E. Bustamante, D. Clark and N. Feamster, "Workshop on tracking quality of experience in the Internet: summary and outcomes," *ACM SIGCOMM Computer Communication Review*, vol. 47, no.1, pp. 55-60, January, 2017. Article (CrossRefLink).

[3] J. Sommers, P. Barford, N. Duffield and A. Ron, "A geometric approach to improving active packet loss measurement," *IEEE/ACM Transactions on Networking*, vol. 16, no. 2, pp. 307-320, April, 2008. Article (CrossRefLink).

[4] H. X. Nguyen and M. Roughan, "Rigorous statistical analysis of internet loss measurements," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 734-745, June, 2013. Article (CrossRefLink).

[5] C. Kocak and K. Zaim, "Performance measurement of IP networks using two-way active measurement protocol," in *Proc. of the 8th International Conference on Information Technology*, pp. 249-254, May 17-18, 2017. Article (CrossRefLink).

[6] S. Basso, M. Meo, A. Servetti and J. C. De Martin, "Estimating packet loss rate in the access through application-level measurements," in *Proc. of the 2012 ACM SIGCOMM workshop on Measurements up the stack*, pp. 7-12, August 17-17, 2012. Article (CrossRefLink).

[7] S. Basso, M. Meo, A. Servetti and J. C. De Martin, "Strengthening measurements from the edges: application-level packet loss rate estimation," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 45-51, July, 2013. Article (CrossRefLink).

[8] Dolk Victor and Maurice Heemels, "Event-triggered control systems under packet losses," *Automatica*, vol. 80, no. 13, pp. 143-145, June, 2017. Article (CrossRefLink).

[9] H. Fan, H. Wang and Y. Li, "Data-driven packet loss estimation for node healthy sensing in decentralized cluster," *Sensors*, vol. 18, no. 2, pp. 320, January, 2018. Article (CrossRefLink).

[10] Z. Hu and Q. Zhang, "A new approach for packet loss measurement of video streaming and its application," *Multimedia Tools and Applications*, vol. 77, no. 10, pp. 11589-11608, May, 2018. Article (CrossRefLink).

[11] T. B. Ma. Richard and V. Misra, "The public option: a nonregulatory alternative to network neutrality," in *Proc. of the Seventh COnference on emerging Networking EXperiments and Technologies*, 2011. Article (CrossRefLink).

[12] A. Antonopoulos, E. Kartsakli, C. Perillo and C. Verikoukis, "Shedding light on the internet: stakeholders and network neutrality," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 216-223, July, 2017. Article (CrossRefLink).

[13] P. Hosein, W. Choi and W. Seok, "Detecting network neutrality violations through packet loss statistics," in *Proc. of the 17th Asia-Pacific Network Operations and Management Symposium*, pp. 404–407, August 19-21, 2015. Article (CrossRefLink).

[14] B. Cho, K. J. Kim and J. W. Chung, "CBR-based network performance management with multi-agent approach," *Cluster Computing*, vol. 20, no. 1, pp. 757-767, March, 2017. Article (CrossRefLink).

[15] S. Okwir, S. S. Nudurupati, M. Ginieis and J. Angelis, "Performance measurement and management systems: a perspective from complexity theory," *International Journal of Management Reviews*, vol. 20, no. 3, pp.731-754, July, 2018. Article (CrossRefLink).

[16] M. Mellia, A. Carpani and R. L. Cigno, "TStat: TCP STatistic and analysis tool," in *Proc. of the International Workshop on Quality of Service in Multiservice IP Networks*, pp. 145-157, February 24-26, 2003. Article (CrossRefLink).

[17] A. Finamore, M. Mellia, M. Meo, M. M. Munafò and D. Rossi, "Live traffic monitoring with Tstat: capabilities and experiences," in *Proc. of the 8th International Conference on Wired/Wireless Internet Communications*, pp. 290-301, June 01-03, 2010. Article (CrossRefLink).

[18] Y. A. Limanto, J. Andjarwirawan and H. N. Palit, "Visualisasi bandwidth usage untuk universitas kristen petra menggunakan tools TSTAT," *Jurnal Infra*, vol. 4, no. 1, pp. 28-34, 2016. Article (CrossRefLink).

[19] Femminella, Mauro, Matteo Pergolesi and Gianluca Reali, "Performance evaluation of edge cloud computing system for big data applications," in *Proc. of the 5th IEEE International Conference on Cloud Networking*, pp. 170-175, October 03-05, 2016. Article (CrossRefLink).

[20] Q. De Coninck, M. Baerts, B. Hesmans and O. Bonaventure, "A first analysis of multipath TCP on smartphones," in *Proc. of the 17th International Conference on Passive and Active Network Measurement*, pp. 57-69, March 31-April 1, 2016. Article (CrossRefLink).

[21] P. Benko and A. Veres, "A passive method for estimating end-to-end TCP packet loss," in *Proc. of the Global Telecommunications Conference*, pp. 2609-2613, November 17-21, 2002. Article (CrossRefLink).

[22] Claudio Favi and Grenville Armitage, "Dynamic performance limits of the Benko-Veres passive TCP packet loss estimation algorithm," in *Proc. of the Australian Telecommunications Network and Applications Conference*, pp. 336-340, December 08-10, 2004. Article (CrossRefLink).

[23] S. Jaiswal, ""Measurements-In-The-Middle": inferring end-end path properties and characteristics of TCP connections through passive measurements," *Ph. D thesis, University of Massachusetts Amherst*, September, 2005. Article (CrossRefLink).

[24] Denis Collange and Jean-Laurent Costeux, "Correlation of packet losses with some traffic characteristics," in *Proc. of the 8th International Conference on Passive and Active Network Measurement*, pp. 233-236, April 05-06, 2007. Article (CrossRefLink).

[25] G. Cheng and Z. Gao, "Estimation packet loss ratios for the two segments of end-to-end path on the monitor," in *Proc. of the International Conference on Computer Science and Service System*, pp. 793-796, August 11-13, 2012. Article (CrossRefLink).

[26] Saeed Ullah, Imdad Ullah, Hassaan Khaliq Qureshi, Rim Haw, Sungman Jang and Choong Seon Hong, "Passive packet loss detection in Wi-Fi networks and its effect on HTTP traffic

characteristics," in *Proc. of the International Conference on Information Networking*, pp. 428-432, February 10-12, 2014. Article (CrossRefLink).

[27] G. Cheng, Y. Tang and T. Gyires, "A lightweight approach to manifesting responsible parties for TCP packet loss," in *Proc. of the 14th International Conference on Networks*, pp. 211-217, April 19-24, 2015. Article (CrossRefLink).

[28] N. L. M. Van Adrichem, C. Doerr and F. A. Kuipers, "OpenNetMon: network monitoring in openflow software-defined networks," in *Proc. of IEEE Network Operations and Management Symposium*, pp. 1-8, May 05-09, 2014. Article (CrossRefLink).

[29] M. Yu, L. Jose and R. Miao, "Software Defined Traffic Measurement with OpenSketch," in *Proc. of the 10th USENIX conference on Networked Systems Design and Implementation*, pp. 29-42, April 02-05, 2013. Article (CrossRefLink).

[30] R. Hark, D. Stingl, N. Richerzhagen, K. Nahrstedt and R. Steinmetz, "DistTM: collaborative traffic matrix estimation in distributed SDN control planes," in *Proc. of IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 82-90, May 17-19, 2016. Article (CrossRefLink).

[31] R. Hark, N. Richerzhagen, B. Richerzhagen, A. Rizk and R. Steinmetz, "Towards an adaptive selection of loss estimation techniques in software-defined networks," in *Proc. of IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 1-9, June 12-16, 2017. Article (CrossRefLink).

[32] Chydzinski Andrzej, Marek Barczyk and Dominik Samociuk, "The single-server queue with the dropping function and infinite buffer," *Mathematical Problems in Engineering*, vol. 2018, Article ID 3260428, 12 pages, October, 2018. Article (CrossRefLink).

[33] E. M. Sierra, D. Muelas, J. Ramos, J. E. L. de Vergara, D. Morató and J. Aracil, "Online Detection of Pathological TCP Flows with Retransmissions in High-speed Networks," *Computer Communications*, vol. 127, pp. 95-104, September, 2018. Article (CrossRefLink).

[34] V. Moreno, J. Ramos, P. M. S. del Río, J. L. García-Dorado, F. J. G. Arribas and J. Aracil, "Commodity packet capture engines: tutorial, cookbook and applicability," *IEEE Communications Surveys and Tutorials*, vol.17, no. 3, pp.1364-1390, May, 2015. Article (CrossRefLink).

[35] Paul Emmerich, Maximilian Pudelko and Sebastian Gallenmüller, "FlowScope: efficient packet capture and storage in 100 Gbit/s networks," in *Proc. of the 16th International IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 1-9, June 12-16, 2017. Article (CrossRefLink).

[36] G. J. Moreno, R. Leira, J. E. L. de Vergara, F. J. G. Arribas and I. González, "On the feasibility of 40 gbps network data capture and retention with general purpose hardware," in *Proc. of the 33rd Annual ACM Symposium on Applied Computing*, pp. 970-978, April 09-13, 2018. Article (CrossRefLink).

[37] F. Schneider and A. Feldmann, "Packet capture in 10-gigabit Ethernet environments using contemporary commodity hardware," in *Proc. of the International Conference on Passive and*

Active Network Measurement, pp. 207-217, April 05-06, 2007. Article (CrossRefLink).

[38] S. Gallenmüller, D. Scholz, F. Wohlfart, Q. Scheitle, P. Emmerich and G. Carle, "High-performance packet processing and measurements," in *Proc. of the 10th International Conference on Communication Systems and Networks*, pp. 1-8, January 3-7, 2018. Article (CrossRefLink).

[39] L. Braun, A. Didebulidze, N. Kammenhuber and G. Carle, "Comparing and improving current packet capturing solutions based on commodity hardware," in *Proc. of the 10th ACM SIGCOMM Conference on Internet Measurement*, pp. 206-217, November 01-30, 2010. Article (CrossRefLink).

[40] A. Papadogiannakis, M. Polychronakis and E. P. Markatos, "Stream-oriented network traffic capture and analysis for high-speed networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 10, pp. 1849-1863, September, 2014. Article (CrossRefLink).

[41] E. Papadogiannaki, L. Koromilas, G. Vasiliadis and S. Ioannidis, "Efficient software packet processing on heterogeneous and asymmetric hardware architectures," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1593-1606, June, 2017. Article (CrossRefLink).

[42] D. Smekal, J. Hajny and Z. Martinasek, "Packet generators on field programmable gate array platform," in *Proc. of the 40th IEEE International Conference on Telecommunications and Signal Processing*, pp. 97-100, July 5-7, 2017. Article (CrossRefLink).

[43] S. Srivastava, S. Anmulwar, A. M. Sapkal, T. Batra, A. K. Gupta and V. Kumar, "Comparative study of various traffic generator tools." in *Proc. of the Recent Advances Engineering and Computational Sciences*, pp. 1-6, March 6-8, 2014. Article (CrossRefLink).

[44] M. Allman, V. Paxson and E. Blanton, "TCP congestion control," *RFC 5681, IETF*, September, 2009. Article (CrossRefLink).

[45] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP selective acknowledgment options," *RFC2018, IETF*, October, 1996. Article (CrossRefLink).

[46] S. Shin, D. Han, H. Cho, J. M. Chung, I. Hwang and D. Ok, "TCP and MPTCP retransmission timeout control for networks supporting WLANs," *IEEE Communications Letters*, vol. 20, no. 5, pp. 994-997, May, 2016. Article (CrossRefLink).

[47] S. Floyd, J. Mahdavi, M. Mathis and M. Podolsky, "An extension to the selective acknowledgment (SACK) option for TCP," *RFC 2883, IETF*, July, 2000. Article (CrossRefLink).

[48] M. Allman, W. M. Eddy and S. Ostermann, "Estimating loss rates with TCP," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 3, pp. 12-24, December, 2003. Article (CrossRefLink).

[49] H. Wu and J. Gong, "Packet loss estimation of TCP flows based on the delayed ACK mechanism," in *Proc. of the Asia-Pacific Network Operations and Management Symposium*, pp. 540-543, September 23-25, 2009. Article (CrossRefLink).

[50] NS-2 – The network simulator version 2.34, 2012. Article (CrossRefLink).

Haoliang Lan is a Ph.D candidate in School of Cyber Science and Engineering of Southeast University. His major research interests include network measurement, network management and network security.

Wei Ding received B.S degree in the computer soft from Nanjing University in 1982. She received Ph.D degree from Southeast University in 1995. Nowadays she is a professor of Southeast University. Her major research interests include high speed communications, network management and network security.

YuMei Zhang is a M.S candidate in school of big data and information engineering of Guizhou University. Her major research interests include network measurement and network management.