Reorder Density (RD): A Formal, Comprehensive Metric for Packet Reordering[1]

Nischal M. Piratla, Anura P. Jayasumana, and Abhijit A. Bare

Department of Electrical and Computer Engineering, Colorado State University,
Fort Collins, Colorado 80523, USA.
{Nischal.Piratla, Anura.Jayasuamna, Abhijit.Bare}@colostate.edu

The increase in link speeds, increased parallelism within routers and switches, QoS support and load balancing among links, all point to future networks with increased packet reordering. Unchecked, packet reordering will have a significant detrimental effect on the end-to-end performance, while resources required for dealing with packet reordering at routers and end-nodes will grow considerably. A formal analysis of packet reordering is carried out and Reorder Density (RD) metric is defined for measurement and characterization of packet reordering. RD captures the amount and degree of reordering, and can be used to define the reorder response of networks under stationary conditions. Properties of RD are derived, and it is shown that the reorder response of the network formed by cascading two subnets is equal to the convolution of the reorder responses of individual subnets. Packet reordering over the Internet is measured and used to validate the derivations.

1 Introduction

The reasons for out of order arrival of packets include but are not limited to: (i) packet striping at layer 2 and 3 links, i.e., when an earlier packet is placed in a longer queue and later packet in a shorter queue, the packets may arrive out of order [4,7], (ii) retransmissions on wireless links [3] and due to TCP, (iii) diffServ scheduling where the flow that exceeds the constraints, e.g., the non-conformant packets are dropped or given a lower priority leading to the packet placement in different queues resulting in an out-of-order delivery [6], and (iv) route fluttering where for example a route may oscillate due to dynamic load splitting among the links. In such cases, different packets of the same stream take different routes leading to different delays [12].

Packet reordering, irrespective of its cause, impacts applications based on both TCP and UDP significantly. In the case of TCP, when packets in forward path go out

[1] This research was supported in part by NSF ITR Grant No. 0121546, Ixia University Partners Program, and Agilent Technologies.

of order, the receiver may perceive packets as lost, resulting in a reduced congestion window, and increased number of retransmissions [4,5] that further degrade the performance. Reverse-path reordering, i.e., reordering of acknowledgements, results in the loss of TCP's self-clocking property, leading to bursty transmissions and possibly to increased congestion [4]. Approaches for mitigating the impact of out-of-order packet delivery on TCP performance include adjusting 'dupthresh' parameter, i.e., the number of duplicate ACKs to be allowed before classifying a following non-acknowledged packet as lost [19]. In delay sensitive applications based on UDP, e.g., IP telephony, an out-of-order packet that arrives after the elapse of playback time is treated as lost thereby decreasing the perceived quality of voice. To recover from reordering, the out-of-sequence packets are buffered until they can be played back in sequence to the application. Thus, an increase in out-of-order delivery by the network consumes more resources at the end-hosts, and also affects the end-to-end performance of the applications.

There is an effort to address this issue at intermediate nodes, at IP level. Many contemporary routers attempt to eliminate the reordering caused by the scheduling schemes within these nodes by either a) input reordering, i.e., identifying the individual streams and forwarding the packets of same stream to the same queue thus preventing reordering, or b) output reordering, i.e., buffering packets at the output of the router to ensure that the packets belonging to the same stream are released in order of their entry into the node [9]. For example, the network processors from vendors such as IBM, Motorola, Vitesse, TI and Intel, have built-in hardware to track flows. While these approaches reduce the reordering that occurs inside a router, they cannot eliminate reordering due to multiple paths. Furthermore, the complexity of these approaches will increase significantly as the number of parallel flows in a pipe increases (due to the need to keep information on a large number of parallel flows), and as the ratio of packet time to routing latency decreases.

The delay in sequencing the packets back in order (as in output buffering) is proportional to log (C_s/U) where C_s is the capacity of the link and U the packet size [18]. The buffer size requirements to put the packets back in order are also shown to increase dramatically with link speed. With the increase in number of flows, keeping track of them becomes difficult too. Moreover, the number of table entries in routers are growing exponentially [17], which means that in spite of high-speed links in future, packets will spend more time within the network, i.e., a higher end-to-end delay to packet time ratio, leading to more load splits and higher packet reordering. Increase in latency required at the intermediate switches and routers to reorder the packets add to the end-to-end latency and the RTT, thereby affecting the performance as well [2].

Next generations of networks have to deal with the problem of out-of-sequence packets proactively. However, scant attention has been paid so far towards understanding the nature of reordering that occurs in networks. Characteristics of reordering that occurs in a given end-to-end path may be used to enhance the ways that the protocols deal with this problem. Measurement and characterization of reordering in packet sequences and models that provide insight into this problem can lead to the development of scalable techniques to deal with reordering and tools that diagnose network problems.

This paper investigates reordering of packets and proposes a metric, Reorder Density (RD), for measuring reordering in a packet sequence. This metric captures the magnitude and statistical properties of reordering occurring in a network. Thus, we define the reorder response of a network, as the RD of the sequence leaving the network corresponding to an in-order input packet sequence. Further, it is shown that the reorder response of a cascaded pair of networks corresponds to the convolution of the reorder responses of the individual networks. This allows us to systematically measure, model and evaluate reordering in complex networks. Next section identifies the requirements of metrics for packet reordering that will make the metric a comprehensive and a useful tool to capture, model, and study packet reordering. In Section 3, the reorder problem is formulated and the proposed metric RD is presented. Section 4 presents measurements of reordering over the Internet for a range of network spans, and verifies the model for computing reordering on cascaded networks. Conclusions are presented in section 5.

2 Metrics for Reordering

The percentage of out-of-order packets has been used as a metric for characterizing packet reordering [3,6]. This method is vague, incomplete and does not provide information about the nature of reordering. Consider two packet sequences (1,3,4,2,5) and (1,4,3,2,5). It is possible to interpret the out-of-orderliness of packets differently in these cases, for example, packets 2, 3 and 4 are out-of-order in both the cases or only packet 2 is out-of-order in the first sequence, while packets 2 and 3 are out-of-order in second case or packets 3 and 4 are out-of-order in both the cases. Even with a convention such as only late packets are considered reordered, the measure is insufficient [15]. Percentage does not capture the amount by which a packet is out of order. The amount of displacement of a packet directly influences the complexity of hardware/software to recover from the reordering. In the case of a sequence such as (1,3,4,2,5), if buffers are available to store packets 3 and 4 while waiting for packet 2, it is possible to recover from the reordering. However, there may be cases where the application requirement is such that arrival of packet 2 after this delay renders it useless. For such cases, we could place a threshold that treats any packet that is more than 2 places late as lost. While one can argue that a good packet reordering measure should capture such effects, a counter argument can also be made that packet reordering should be measured strictly with respect to the order of delivery and should be application independent. A framework for metrics presented in [13] states that "The metrics must be useful to users and providers in understanding the performance they experience or provide."

A metric for capturing out-of-order nature of a packet sequence ideally should have the following properties:

1. Simplicity: The measure should be simple, yet contain enough information to be useful.
2. Orthogonality: Metric should, to the extent possible, be independent or orthogonal to other phenomena that affect the packet streams, e.g., packet loss and duplication.

3. Differentiability: Metric should provide insight into the nature of reordering, and perhaps even into possible causes. It should capture both the amount and extent of reordering.
4. Usefulness: Rather than being a mere representation of the amount of reordering in a packet stream, reorder metric must be useful to the application and/or resource management schemes. For example, it may allow one to determine the size of buffer that is required to recover from reordering.
5. Evaluation complexity: The metric should be computable in real-time. In evaluating reordering in an arbitrarily long sequence, one should be able to keep a running measurement, without having to wait till all the packets have arrived. The memory requirement, i.e., the amount of state information, should not grow with the length of the sequence (N) and the computation time should be O (N).
6. Robustness: Reorder measurement should be robust against different network phenomena and measurement peculiarities such as a very late arrival of a duplicate packet or a burst of losses.
7. Broader Applicability: A good metric would have applicability beyond just characterizing the nature of reordering in a given sequence of packets. For example, a good metric may allow one to combine the characteristics of individual networks to predict the reorder behavior of the cascade of these networks. Regeneration of a sequence that follows the measure is also a very useful application.

Recently, Internet Engineering Task Force (IETF) has presented a few metrics for reordering [8,11]. The metrics in [11], a work in progress, are classified into sample metrics and receiver assessment metrics. Sample metrics are defined on a sample of sequence numbers and percentage reordering can be an example of such a metric. However, these metrics fail to meet many of the criteria mentioned above, especially those related to differentiability, usefulness and robustness. The receiver assessment metric mainly relies on n-reordering, which looks only at few late packets. For sequence (1,4,5,2,3,2); the n-reordering method treats 1, 4 and 5 as 0-reordered. Since 2 is late by 2 places, it is 2-reordered but 3 is 0-reordered whereas duplicate 2 is 1-reordered. Besides the ambiguity in the definition, this metric is not orthogonal to duplication. In addition, the metric uses a buffer to store current arrivals to compute the n-reordering of the future arrivals leading to larger buffer requirements for higher amount of reordering. The reorder metric proposed in [1] measures the occupancy density of the reorder buffer, and as such will be referred to as Reorder Buffer-occupancy Density (RBD) [8].

3 Analysis of Reordered Sequences

In this section, we develop a general formulation for characterizing packet reordering, and define Reorder Density (RD). We then evaluate the reorder response of a cascade of networks in terms of the reorder responses of the individual subnets.

3.1 Representation of Reordering

Without loss of generality, consider a sequence of packets (1,2…N) transmitted over a network. A receive_index (1,2…) is assigned to each packet as it arrives at the destination. Lost and duplicate packets are not assigned a receive_index. First consider the case in which no losses or duplication of packets occur in the network. If the receive_index assigned to packet m is (m + d_m), with $d_m \neq 0$, we say that a reorder event has occurred, and this event is denoted by r(m, d_m). In the absence of reordering the sequence number of the packet and the receive_index are same, i.e., $d_m = 0$ for each packet. A packet is late if $d_m > 0$, and early if $d_m < 0$. Thus, packet reordering of a sequence of packets is completely represented by the union of reorder events, R, referred to as the reorder set: $R = \bigcup_m \{r(m, d_m) \mid d_m \neq 0\}$

Table 1. (a), (b) and (c) Examples of reordered sequences with corresponding R.

Arrived Sequence	1	2	3	5	4	6	8	7
Receive_index	1	2	3	4	5	6	7	8
Displacement	0	0	0	-1	1	0	-1	1

(a) No losses or duplicates R = {(4,1), (5,-1), (7,1), (8,-1)}

Arrived Sequence	1	2	5	3	6	7	8	9
Receive_index	1	2	3	5	6	7	8	9
Displacement	0	0	-2	2	0	0	0	0

(b) Packet 4 is lost R = {(3, 2), (5, -2)}

Arrived Sequence	1	2	6	4	3	5	3	3
Receive_index	1	2	3	4	5	6	-	-
Displacement	0	0	-3	0	2	1	-	-

(c) Packet 3 is duplicated R = {(3, 2), (5, 1), (6, -3)}

If there is no reordering in a packet sequence then R = {ϕ}. Conventionally, we represent R with non-decreasing order of m. Table 1(a)-(c) show the examples of the arrived sequence (sequence number), assigned receive_index and displacement, as well as the corresponding reorder sets for three cases: a) without losses or duplication, b) with loss, and c) with duplication of packets.

Lemma 1: In the absence of losses and duplicates, the sum of displacements of all packets in a sequence is equal to zero, i.e. $\sum d_i = 0$.

Proof: Consider a packet sequence of size N: (1,2 …N). If d_i *denotes the displacement of* i^{th} *packet due to reordering, then the new positions of the packets 1,.. N are:*

$$(1+d_1),\ (2+d_2),\ ..,\ (i+d_i)..,\ (N+d_N)$$

Since the number of positions at the sender and receiver are equal, the sum of receive_indices is the sum of integers from 1 to N. Therefore we have,

$$\sum_{i=1}^{N} i = \sum_{i=1}^{N} (i + d_i) \Rightarrow \sum_{i=1}^{N} d_i \quad \text{(Q.E.D.)}$$

Next, consider the case where packets may be lost or duplicated in transit. Assume that the loss of a packet can be detected at the receiver. We skip the receive_index corresponding to the sequence number of the lost packet, i.e., if packet 'e' is lost, then the receive_index = e is not assigned. In case of duplicates, we consider only the first copy of the packet at the receiver end, and discard the duplicate, i.e., the duplicate is not assigned a receive_index. These two cases are illustrated in Tables 1 (b) and (c) where the packet with sequence number 4 is lost and the packet with sequence number 3 is duplicated respectively. How to detect the loss of packets on the fly to skip the receive_index and deal with duplicate packets in a measurement environment, is addressed in subsection 3.4 below and [8] in detail.

Lemma 2: When lost and duplicate packets are detected and receive_index is assigned as specified, Lemma 1 holds true, i.e., $\sum d_i = 0$.

Proof: Consider the case where packet 'e' is lost and 'j' is duplicated. Since we ignore the duplicate copy of j, no index is assigned to the duplicate of that packet. As the index 'e' is reserved for the lost packet, it is not used in the (N-1) receive indices. Thus, the sum of (N-1) receive_index values assigned at the receiver is

$$\sum_{i=1, i\neq e}^{N} (i + d_i) = \sum_{i=1}^{N} (i) - e \Rightarrow \sum d_i = 0 \text{ (Q.E.D).}$$

3.2 Reorder Density (RD)

RD is defined as the discrete density of the frequency of packets with respect to their displacements, i.e., the lateness and earliness from the original position. Let $S[k]$ denote the set of reorder events in R with displacement equal to k, i.e.,

$$S[k] = \{\mathrm{r(m}, d_m) \mid d_m = k\} \ .$$

Let $|S[k]|$ be the cardinality of set $S[k]$. Thus, RD[k] is defined as $|S[k]|$ normalized with respect to the total number of received packets ($\mathrm{N'}$). Note that $\mathrm{N'}$ does not include duplicates or lost packets. RD[0] corresponds to the packets for which receive index is the same as the sequence number. Thus

$$\mathrm{RD}[k] = |S[k]| / \mathrm{N'} \text{ for } k \neq 0 \text{ and RD[0]} = 1 - \sum_{k\neq 0} |S[k]| / \mathrm{N'}$$

A reorder histogram function, where $|S[k]|$ is used directly without normalization, may prove useful for certain applications, where the actual number of packets that is reordered is given. The results presented in this paper can easily be modified to cover such a case.

Lemma 3: Reorder Late/Early density function (RD) satisfies $\sum_k (k * \mathrm{RD}[k]) = 0$

Proof: Since RD [k] implies that there are $N' * RD[k]$ *packets with displacement k, the sum of displacements of packets is given by*

$$\sum_k (k*\mathrm{N}'*\mathrm{RD}[k]) = \mathrm{N}'*\sum_k (k*\mathrm{RD}[k]) = \sum_m d_m = 0$$ *(from Lemma 2) (Q.E.D)*

3.3 Reorder Response of Networks

The characterization of reordering in a sequence of packets in the form of RD, in addition to capturing the out-of-orderliness of a packet sequence accurately, also allows us to model networks with respect to reordering introduced by sub-networks forming the network.

First note that RD corresponding to a sequence of in-order packets is a unit impulse (at $k = 0$). Consider sending this sequence of packets through a network. RD of the sequence observed at the output corresponds to the reordering introduced by the network, and hence we call it the reorder response ($\boldsymbol{J}[k]$) of the network. Note that ($\boldsymbol{J}[k]$) is defined with respect to an input and an output of a network. Thus each input/output port pair of the network has a corresponding reorder response. $\boldsymbol{J}[k]$ of a network in which there is no reordering corresponds to a unit impulse at k=0. $\boldsymbol{J}[k]$ can also be interpreted as the probability that a certain packet gets displaced by k.

The reorder response of a network captures the effect of the network on a sequence of packets flowing from the source port to the destination port of interest. Since a network is a time-varying and highly dynamic environment, it may not be possible to characterize the network completely using a single reorder response function. In fact, the reorder response will be a function of many factors including the background or cross traffic in the network, and statistical characteristics of the inter-packet gap. We hypothesize however that a reorder response exists for a network operating under stationary conditions and a given inter-packet gap distribution, provided the reordering introduced by the network is not dependent on the sequence number of the packet. This excludes, for example, a node that looks at the packet sequence number and puts the packets in a certain order. In fact, a set of measurements that we have carried out supports this hypothesis [15]. Thus the following discussion applies only to networks operating under stationary conditions, i.e., statistical characteristics of the network and packet sequence do not change with time. This excludes, for example, a node that looks at the packet sequence number and puts the packets in a certain order. We expect that a theory developed for reordering in networks under such stationary conditions will lead the way towards a more general solution in future.

Theorem: The reorder response $\boldsymbol{J}[k]$ *of a network formed by cascading two subnets, with reorder responses* $\boldsymbol{J_1}[k]$ *and* $\boldsymbol{J_2}[k]$ *respectively, is given by the convolution of* $\boldsymbol{J_1}[k]$ *and* $\boldsymbol{J_2}[k]$ *, i.e.,* $\boldsymbol{J}[k] = \boldsymbol{J_1}[k] * \boldsymbol{J_2}[k]$.

Proof: Consider an arbitrary packet with sequence number 'm' that enters the cascade of two networks. It undergoes a displacement $d_m^{(1)}$ *in Net-1, corresponding to the event* $r(m, d_m^{(1)})$. *Thus, packet m occupies the position of m' as it enters Net-2, with* $m' = m + d_m^{(1)}$. *When the packet with sequence number m' enters Net-2, it undergoes a displacement* $d_m^{(2)}$ *in Net-2. The overall displacement of the packet* $d_m = d_m^{(1)} + d_m^{(2)}$.

The corresponding reorder event is $r(m, d_m) = r(m, d_m^{(1)} + d_m^{(2)})$.

The probability distribution of $d_m^{(1)}$ *and* $d_m^{(2)}$ *are given by reorder responses of Net-1* $\boldsymbol{J_1}[k]$, *and Net-2* $\boldsymbol{J_2}[k]$, *respectively. Since* $d_m = d_m^{(1)} + d_m^{(2)}$, *and the packet reordering is considered to be independent of the sequence number, the probability density distribution of* d_m *is given by the convolution of the distributions of* $d_m^{(1)}$ *and* $d_m^{(2)}$, *i.e.,* $\boldsymbol{J_1}[k] * \boldsymbol{J_2}[k]$ *(Q.E.D)*

3.4 Evaluation of RD at the receiver

Here we describe the evaluation of RD at the receiver. Lost packets and duplicates are taken into account by skipping the receive_index corresponding to lost packets, and not assigning a duplicate packet a receive_index. However, this process needs detection of losses and duplicates, both of which provide implementation challenges.

When can a packet be considered lost? One possibility is to consider it lost if it does not arrive when it is expected, but if it arrives later, go back and make appropriate corrections. At the other extreme, one can wait till the end of the received sequence to declare a packet as lost. However this requires keeping track of all received packets, as well as applying corrections to computations performed so far. Both these approaches are memory consuming, and also preclude real-time evaluation of the metrics. Maintaining a threshold D_T and an early arrival buffer addresses this problem. If a packet is not received within D_T packets from where it is expected, it is considered lost. A packet is classified as a duplicate on its arrival, if it already exists in early arrival buffer or current D_T window or the packet number is less than current receive_index. With the use of threshold D_T, since it is not known whether a packet is lost until D_T packets are received, real-time RD evaluation may be done in one of two ways:

1. Go-back D_T: In this method, the rules are applied at each arrival. If a packet that was supposed to arrive D_T places ago does not arrive, then this sequence number is removed from the receive_index, and RD is recomputed for the previous D_T steps. Consider a received sequence (1,3,4,5,6,7,2) and $D_T = 3$. As soon as 5 arrives, 2 is classified as lost and we go back and correct the previous D_T receive_indices and displacements. When 2 actually arrives later, we do not assign receive_index to this arrival, i.e., consider it as lost and discard the packet. This method requires recording the previous D_T packet numbers, and additional processing as we recomputed offsets when a packet is lost. However, if the amount of reordering is low, the overall computation is quicker than the next method.

2. Stay-back D_T: Here the computation lags by D_T packets, i.e., the packet with receive_index *i* is not used in the evaluation until D_T more packets have arrived after that. Thus, we do not correct or adjust any displacements. This method also requires buffering of the next D_T arrivals.

A small D_T value is used in above illustrations of the concept for convenience. It may be set much higher for practical measurements. A larger threshold value results in higher memory requirements for these implementations. However, the computation complexity in both cases is of the order N, where N is the size of the received sequence. Use of the threshold D_T also improves the robustness of RD. For example, if a packet was late by a large number say 1000, then the next 1000 packets would be shown as early in the absence of a threshold. By using a threshold, we can eliminate such large impacts on RD due to a single reordering event. Furthermore, it allows us to recover from conditions such as a large number of missing or duplicate packets. We have not completely described this aspect in the present paper. Perl scripts and algorithms are available in [14].

4 Measurements and Results

This section presents measurement-based results to illustrate and characterize the reordering in networks using the RD concept. To measure reordering, using RD, multiple files were downloaded to a host on subnet 129.82.x.x (in Colorado, USA) from different sites around the world. To make sure that the connection is not short-lived, sites with downloadable files of sizes more than 2MB were chosen. Using IP2Location software and ping results, we obtained the geographical location, the average one-way delay value (D) and approximate standard deviation of the delay (SD). We are presenting only 3 of 10 sets of measurements [15] here, corresponding to a range of end-to-end delays. TCP based applications namely; HTTP and FTP were used for servers in USA (209.211.x.x) and Italy (62.94.x.x) respectively. Data collected from them had D = 45.5 ms and 81.9 ms, and SD = 0.81 ms and 0.19 ms respectively. From host on subnet 130.105.x.x, a media file was played and packets had D = 95.4 ms and SD = 0.17 ms. Using 'tcpdump' tool to collect the information of received data, the TCP and RTP sequence numbers were mapped to (1,2 …). Fig. 1 shows the observed RDs, which is also the reorder response of the network, from these sites. (Vertical axis is broken to increase the clarity of diagrams).

RD provides a comprehensive measure of reordering and we can draw a number of inferences from these measures: (i) Net-1 has the smallest average delay, but due to larger deviations, the amount of reordering is comparatively high. Conversely, looking at the shape of RD, we can comment on the delay deviations in the network. The wider the RD spread, the higher the variance in delay. (ii) For Net-1 it is evident that the network deviates from the normal expectation, as large number of packets arrives reordered. Knowing the corresponding RD, we can tune this network by allocating a larger buffer size to recover from reordering in UDP application or increasing the number of duplicate ACKs to wait before fast retransmit with TCP [10, 12]. (iii) For Net-2, the application can recover from reordering by having a buffer size equal to 2 packets. Although Net-2 has approximately 2% more reordering than

Net-3, applications using Net-2 will perform better due to lower displacements of reordered packets. (iv) In the case of Net-3, the RD has a discontinuity. One percent of the packets can be as late as 4 positions. It is possible that due to phenomenon like packet stripping in a network, these packets take an alternate path. Here, instead of using triple-ACK for *fast retransmit*; we could use 2-ACK, given the effect of 1% reordering on performance is acceptable or use 4-ACK to account of all reordered packets.

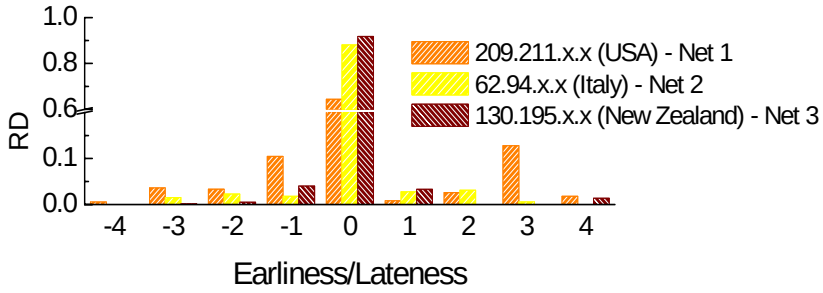

Fig 1. RD based on measurements for Net-1, 2 and 3. ($D_T = 5$)

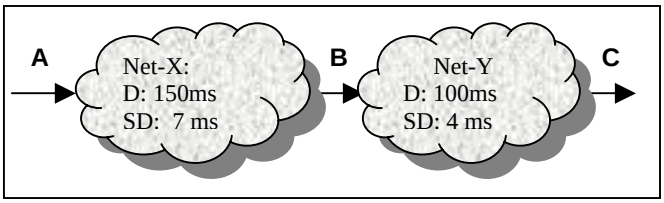

Fig 2. Emulated network topology to verify convolution theorem for RD

To verify the convolution theorem for reordering, we emulated two networks using NISTNet on Linux boxes. The connectivity used is shown in Fig. 2. RDs were used as reorder responses for the networks. Net-X was emulated using mean (D) and standard deviation (SD) of delays as 150 ms and 7 ms respectively and Net-Y is emulated using D = 100 ms and SD = 4 ms. Packet streams of sizes 10,000 packets were sent from A to B and B to C separately. Another packet stream of 10,000 packets was sent from A to C on this cascade. Constant inter-packet gap (of 500μs) was maintained in each of the input streams. These emulators running on the workstations were then connected using a cross-cable. The RD computed in this case was compared to the result based on convolution of the RDs for A to B and B to C cases. This comparison is shown in Fig. 3. The mean square error (MSE) between the two was equal to 8.77e-06, strongly validating the theorem.

It is interesting to note that for the packet stream from A to C, even though the inter-packet gap at A was 500μs, by the time packets transit through B, the inter-packet gap is no longer a constant. On the other hand, the reorder response for Net-Y used for convolution, shown in Fig. 3(a) corresponds to a constant inter-packet gap.

This indicates that the reorder response of a network, while dependent on the inter-packet gap distribution in general, may not be highly sensitive to it. This is an aspect of reorder response that we continue to investigate.

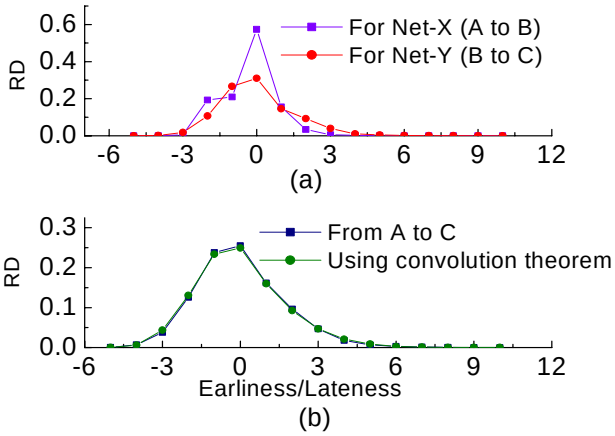

Fig. 3 (a) RD from A to B and B to C with separate streams (b) Verification of the theorem of convolution using network emulators for RDs as reorder responses

5 Conclusions

The existing metrics to measure reordering are vague and insufficient to characterize packet reordering. We have presented a formal method for representing out-of-order sequences of packets, and defined the reorder density (RD) metric. RD characterizes and measures packet reordering comprehensively, is simple, and orthogonal to losses and duplicates. It captures both the amounts of packets affected and the magnitude of reordering, and can be used as the probability density function corresponding to the displacement of an arbitrary packet. The metric can be evaluated in real-time as the packets arrive at a node. A threshold D_T limits the complexity of implementation, by considering a packet that is late by D_T to be lost. The computation complexity of the algorithm is O(N), where N is the number of packets in the sequence. The memory requirement for the implementation is proportional to D_T. The use of D_T also makes it robust, allowing it to recover from cases such as very early or very late packets, and sequences of duplicates, or bursty losses. Further, the metric can be used to characterize the reordering introduced by a network, and under a fairly broad set of conditions, the reorder measurement of different subnets can be combined to predict the end-to-end reorder characteristics of a network. Currently, sequence regeneration algorithm is available for RD measures [2].

Reorder response of a network depends on factors such as the network load, background traffic, and the distribution of the inter-packet gap. At high sending rates, inter-packet gaps have negligible correlations, also validating convolution results [16]. Our present work includes measurements to understand packet reordering over

the Internet in more detail. By keeping track of RD for an on-going connection, one can dynamically tune transport protocols to obtain superior performance.

References

1. Banka, T., Bare, A. and Jayasumana, A., "Metrics for Degree of Reordering in Packet Sequences," Proc. of the IEEE 27th LCN, Nov. 2001, pp. 333-342.
2. Bare, A. A., "Measurement and Analysis of Packet Reordering," Masters Thesis, Department of Computer Science, Colorado State University, 2004.
3. Bellardo, J. and Savage, S., "Measuring Packet Reordering," Proc. of Internet Measurements Workshop (IMW'02), Nov. 2002, pp.97-105.
4. Bennett, J. C. R., Partridge, C. and Shectman, N., "Packet Reordering is Not Pathological Network Behavior," Trans. on Networking IEEE/ACM, Dec. 1999, pp.789-798.
5. Blanton, E. and Allman, M., "On Making TCP More Robust to Packet Reordering", ACM Computer Comm. Review, 32(1), Jan. 2002, pp.20-30.
6. Bohacek, S., Hespanha, J., Lee, J., Lim, C. and Obraczka, K., "TCP-PR: TCP for Persistent Packet Reordering," Proc. of the IEEE 23rd ICDCS, May 2003, pp.222-231.
7. Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J. and Towsley, D., "Measurement and Classification of Out-of-sequence Packets in Tier-1 IP Backbone," Proc. of IEEE INFOCOM, Mar. 2003, pp. 1199- 1209.
8. Jayasumana, A., Piratla, N. M., Bare, A. A., Banka, T., Whitner R., and McCollom, J., "Reorder Density Function - A Metric for Packet Reordering Measurement," IETF draft (work in progress).
9. Liu, H., "A Trace Driven Study of Packet Level Parallelism," Proc. of International Conference on Communications (ICC'02), New York, NY, 2002, pp. 2191-2195.
10. Loguinov, D. and Radha, H., "End-to-End Internet Video Traffic Dynamics: Statistical Study and Analysis," Proc. of IEEE INFOCOM, Jun. 2002, pp.723-732.
11. Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and Perser, J., "Packet Reordering Metric for IPPM," IETF draft (work in progress).
12. Paxson, V., "Measurements and Analysis of End-to-End Internet Dynamics," Ph.D. Dissertation, Computer Science Department, University of California, Berkeley, 1997.
13. Paxson, V., Almes, G., Mahdavi, J. and Mathis, M., "Framework for IP performance metrics," RFC 2330.
14. Perl scripts for RD, http://www.cnrl.colostate.edu/Reorder_perl_scripts.html. Last modified on Nov. 3, 2004.
15. Piratla, N., "Metrics, Measurements and Techniques for the End-to-end Characterization of Networks (tentative)," Ph. D. Dissertation, Department of Electrical and Computer Engineering, Colorado State University, (work in progress).
16. Piratla, N. M., Jayasumana A. P., and Smith H., "Overcoming the Effects of Correlation in Delay Measurements using Inter-Packet Gaps," Proc. of IEEE International Conference on Networks (ICON), Singapore, Nov 2004, pp. 233-238.
17. Ruiz-Sanchez, M., Biersack, E. W. and Dabbous, W., "Survey and Taxonomy of IP Address Lookup Algorithms," IEEE Network, Mar./Apr. 2001, pp. 8-23.
18. Xia, Y. and Tse, D., "Analysis on Packet Resequencing for Reliable Network Protocols," Proc. of IEEE INFOCOM, San Francisco, CA, Mar. 2003, pp. 990-1000.
19. Zhang, M., Karp, B., Floyd, S., and Peterson, L., "RR-TCP: A Reordering-Robust TCP with DSACK," Proc. The Eleventh IEEE International Conference on Networking Protocols (ICNP 2003), Atlanta, GA, Nov. 2003, pp. 95-106.