Revision 1: August 26, 2016

Hop-By-Hop Best Effort Link Layer Reliability in Named Data Networking

Satyanarayana Vusirikala*, Spyridon Mastorakis†, Alexander Afanasyev†, Lixia Zhang†
*MSR India, satya.vus@gmail.com
†University of California, Los Angeles, {mastorakis, aa, lixia}@cs.ucla.edu

***Abstract*—Named Data Networking (NDN) supports the best effort data retrieval using Interest-Data exchanges. When either an Interest or Data packet is lost and not recovered by the router's Interest forwarding strategy (e.g., by retransmitting the Interest along an alternative path), relying on end applications for loss detection may add significant delays in data retrieval that can impact application performance. In this paper, we propose a best effort hop-by-hop link layer reliability protocol (BELRP) for point-to-point communication links. BELRP makes a quick, non-persistent attempt to detect and recover packet losses over a single link with minimal delay. We evaluated BELRP through simulations and our results show that BELRP can significantly shorten data retrieval delays and hence improve application performance, especially over networks with non-negligible probability of packet losses.**

I. Introduction

The Named Data Networking (NDN) [1] architecture provides a best effort data retrieval service: applications send Interest packets which carry the names of requested data, and Data packets with matching names are returned. If either an Interest or Data packet is lost during the transmission due to bit errors, network congestion, link outages, or hardware failures, the application needs to re-express the Interest if it still wants the data. However, relying on applications to detect and recover from packet losses may significantly impact data retrieval delays. Non-trivial packet loss rate can also severely impact the quality of delay-sensitive applications which cannot afford the delays associated with end-to-end loss recoveries.

In this paper, we design a best effort hop-by-hop link layer reliability protocol (BELRP) that makes the best effort to recover from packet losses across point-to-point links. Intuitively, hop-by-hop loss detection and recovery should cost much less delay and overhead than the end-to-end approach. However, given loss detection and data retransmission necessarily introduce a certain amount of delay, one design challenge is how to minimize the delay due to BELRP to work seamlessly for delay-sensitive applications, and to minimize the interference with end-to-end reliability mechanisms. Another design challenge is how to make BELRP as simple as possible to minimize the processing and storage overhead on forwarders.

To address the above challenges, the BELRP design, as described in Section III, uses sequence numbers instead of retransmission timers for loss detection and piggybacks acknowledgments on packets going the reverse direction. We evaluate BELRP performance through extensive simulations (Section IV), which show that BELRP can substantially improve application performance in high-loss network environments with very low cost in both additional delay and node processing and storage. We discuss design trade-offs and possible extensions in Section V. Although BELRP is designed to meet the performance needs of the Named Data Networking (NDN) architecture, as a link layer protocol BELRP is independent of the network architecture, thus should be generally applicable to other packet-switched networks including today's TCP/IP-based Internet.

The contribution of this report is two-fold. First, it presents the BELRP design and elaborates on the design decision and trade-off considerations. Second, it reports the performance evaluation results from simulation studies showing that BELRP achieves its design goals.

II. Related Work

The existing works on reliable data delivery protocols can be roughly sorted into two categories. The first one, represented by TCP and its variants, performs loss detection and recovery at the end-to-end level [2, 3, 4, 5]. As suggested by Saltzer et al. [6], this end-to-end principle removes the dependency on the underlying protocols for reliable data delivery to applications.[1]

The second category of reliability protocols, such as X.25 [7], 802.11 [8] and other protocols for wireless networks [9, 10, 11], performs hop-by-hop loss recovery. In this case, intermediate nodes detect packet losses and perform retransmissions. Some of them, including 802.11, only provide best effort reliability, retransmitting each lost packet up to a given number of times only, without guarantee of eventual recovery of all packet losses.

A common ingredient in all the schemes for reliable delivery is retransmission timers. For example, TCP uses a retransmission timer to detect losses and trigger retransmissions. However, because timers use past network conditions to make an estimation of the future round trip delays to decide when to retransmit a packet, choosing a proper retransmission timer value faces the dilemma between superfluous retransmissions

*This work was done while the author worked at UCLA as an intern during summer 2015.

[1]Note that the end-to-end principle does not preclude lower layers from assisting the reliable delivery, it simply stresses that the higher levels should be ultimately responsible for delivery guarantee.

when timing out too early and delayed loss recovery when timing out too late [12, 13, 14, 15, 16, 17].

To avoid superfluous retransmissions, TCP retransmission timer setting leans towards the conservative end. To avoid waiting for too long before retransmission, TCP introduces a fast retransmission mechanism, counting the number of duplicate ACKs, to detect packet losses and initiate retransmission *before* the timer goes off [18]. Another approach to detect losses without relying on timers is by observing gaps in message sequence numbers [19]. When such a gap is detected, it is assumed to be a lost packet which is then retransmitted without waiting for the timeout. However as pointed out by [12], a retransmission timer is still needed as a last resort for loss detection when all other means fail. For example, either ACK or NACK packets can get lost, and one cannot detect the loss of the last packet in a sequence by observing gaps in sequence numbers.

BELRP detects losses and performs retransmissions on a hop-by-hop basis. Instead of using a retransmission timer, it achieves fast loss detection by observing gaps in sequence numbers carried in the packet between two adjacent neighbor nodes. It bounds the loss recovery delay by limiting the recovery attempts to a small number[2] and gives up after that, hence it is termed a *best effort* protocol.

III. Protocol Design

In this section, we present the design of BELRP. We first describe our design goals and then we elaborate on the basic ideas of the design, including best effort reliability, loss detection mechanism, and acknowledgment redundancy. Finally, we analyze the overhead of our scheme in terms of the required network and host resources, application performance and host processing.

A. Design Goals

NDN applications that require reliable data delivery will deploy end-to-end reliability mechanisms. BELRP aims to minimize end-to-end retransmissions by minimizing packet losses as observed at the application level, at the same time avoiding falsely triggering end retransmissions. The basic design goals include

- Keep the additional delays due to hop-by-hop retransmission minimal.
- Keep the protocol simple to avoid adding too much complexity to node processing.
- Minimize protocol cost and overhead.

B. Best Effort Link Layer Reliability

There is a trade-off between data delivery delay and delivery reliability. When a particular data frame gets lost repeatedly, if the link layer keeps retransmitting until it succeeds, chances are that the delay introduced by the retransmissions will interfere with end applications' reliability mechanisms. Specifically, while a link layer protocol is trying to restore the lost frame, the message encapsulated in this frame might be retransmitted by a higher layer reliability mechanism or be no longer necessary.

To minimize this potential interference, BELRP limits its number of retransmission attempts, i.e., is a best effort service. If a frame is lost more than a pre-configured number of times across a link, BELRP stops trying and leaves to the higher layer mechanisms to take appropriate actions.

To fully understand the trade-off of how hard BELRP should try to restore a lost frame, we performed simulation experiments with varying numbers of maximum retransmissions as discussed in section IV.

C. BELRP Loss Detection

BELRP uses sequence numbers to detect data frame losses. More specifically, when a packet is received from the network layer, BELRP adds two sequence numbers to it: (1) *a frame number*, and (2) *a packet number*. A new packet number is assigned to every packet received from the network layer, and a new frame number is assigned to every outgoing frame transmitted by the link layer. When a packet P needs to be retransmitted, P keeps the same packet number, but a new frame number is assigned to the frame carrying this retransmitted P. The sender uses a buffer to keep every frame that has not been acknowledged, which contains both its packet and frame numbers.

Upon receiving a frame, the link layer receiver responds with an *acknowledgment (ACK)* containing the frame number of the received frame. When the ACK of a frame is received, the sender removes the corresponding frame from the buffer. When a gap is observed in the incoming ACK stream, the sender assigns the lost frame F the next unused frame number and retransmits it immediately, then increase the retransmission count by 1. Using separate packet and frame numbers helps the sender keep track of which packets have not been ACKed based on the packet sequence number, and use gaps in the acknowledged frame number sequence to detect losses, a solution similar to the one used in QUIC [20].

Algorithm 1 describes the protocol operation when the link layer receives a packet from the network layer (Send function) and when the receiver receives a frame from the sender (Receive function). Received ACKs are processed using the ReceiveACK function.

D. Minimizing Unnecessary Retransmissions

According to the above description, a retransmission can be triggered by the following 3 cases:

1) The frame carrying network layer packet P is lost;
2) The frame or its corresponding ACK is delivered out-of-order; and
3) The corresponding ACK is lost.

Retransmissions due to the first case are necessary, retransmissions due to the last 2 cases should be minimized, if they cannot be eliminated. In particular, we note that out-of-order frame arrivals are likely to happen when NDN runs over UDP tunnels. To minimize false retransmissions due to out-of-order

[2]This number can be adjusted based on the link delay and loss probability.

Algorithm 1 Send and Receive functions used by the link layer to send and receive a frame respectively. Received ACKs are processed using the ReceiveACK function

```
1: procedure SEND(PACKET)
2:     m ← NewFrameNumber()
3:     n ← NewPacketNumber()
4:     ACKs ← ACKsToBePiggybacked()
5:     Frame ← AddHeader(Packet, m, n, ACKs)
6:     StoreInBuffer(Packet, m, n)
7:     Transmit(Frame)
8: end procedure
9: procedure RECEIVE(FRAME)
10:    m ← GetFrameNumber(Frame)
11:    ACKs ← GetPiggybackedACKs(Frame)
12:    Packet ← RemoveHeader(Frame)
13:    ReceiveACK(ACKs)
14:    SendToNetworkLayer(Packet)
15:    SendACK(m)
16: end procedure
17: procedure RECEIVEACK(ACKS)
18:    for m ← GetFrameNumber(ACKs) do
19:        RemoveFromBuffer(m)
20:    end for
21:    for FrameNum ← LostFrames() do
22:        Packet ← FetchFromBuffer(FrameNum)
23:        if Retransmissions(Packet) < MAX_RT
   then
24:            m ← NewFrameNumber()
25:            n ← GetPacketNumber(FrameNum)
26:            ACKs ← ACKsToBePiggybacked()
27:            Frame ← AddHeader(Packet,m,n,ACKs)
28:            Transmit(Frame)
29:        else
30:            RemoveFromBuffer(FrameNum)
31:        end if
32:    end for
33: end procedure
```

arrivals, a sender interprets that a frame F(m, n) (m: frame number, n: packet number) is lost only after receiving 3 distinct ACKs, each with a sequence number greater than m. After that, the sender retransmits the packet n.

To minimize unnecessary retransmissions due to the third case, BELRP adds redundancy in frame acknowledgment. More specifically, BELRP lets the receiver send each ACK 3 times for every received frame. This reduces false retransmission probability with a slight increase in the protocol overhead. We believe that the level of ACK redundancy should be adjustable to achieve the best trade-off between protocol overhead and minimizing spurious retransmissions, and that the overhead can be reduced through smart ACK encoding; this is left for future work.

To further reduce the protocol overhead, BELRP piggybacks ACKs on the frames going the opposite direction. Upon receiving a frame, the receiver decides whether ACK piggybacking is possible on the frames that are already in its outgoing queue. If so, it piggybacks $\min(\text{queue size}, 3)$ ACKs with the frames at the front of the queue and the remaining $3-\min(\text{queue size}, 3)$ (if the result > 0) ACKs with future traffic. When the outgoing queue is empty, but an ACK needs to be sent back to the sender to report a gap in received frame sequence numbers (an ACK reports a gap to the sender if the receiver has not received one of the previous 3 frames), the receiver will transmit this ACK immediately. ACKs that do not report gaps are piggybacked on future traffic. This design aims to minimize both the loss detection time and the cost of transmitting standalone ACKs.

E. Protocol Cost

BELRP introduces the following cost:

- *Buffer space at the sender* : Each sender uses a buffer space to store each frame for which the ACK has not been received, so that lost frames can be retransmitted. Because only the outstanding frames are stored in the buffer, the outgoing link's round-trip pipe size is the upper bound of the required buffer size. The exact buffer size at the sender is determined by the packet sending rate, the propagation delay, and the link capacity.
- *2 sequence number fields per frame*: These are the frame and packet numbers.
- *Acknowledgments*: An ACK field, which may carry multiple sequence numbers, is added to every link layer frame; if a link has no traffic in one direction to piggyback ACKs, stand-alone ACKs will be sent.
- *Occasional unnecessary retransmissions*: The sender will retransmit a successfully received frame if all its redundant ACKs are lost.

IV. Simulation Studies

We used simulations to evaluate how well BELRP can enhance delivery reliability and examine its cost, in the context of a Named Data Networking (NDN) network. The NDN communication model assumes that a request for Data (an Interest packet expressed by a consumer application) is forwarded across the network and eventually brings back the requested Data, either from a router cache or otherwise from the data producer application. End data consumer applications will retransmit Interest packets if they do not receive the requested Data within the lifetime of the expressed Interest packet.

We implemented BELRP in ndnSIM [21], an NS-3 based NDN simulator, running directly the code of the NDN Forwarding Daemon (NFD) [22]. The simulation scenarios that we will mention in this section can be found here: https://github.com/spirosmastorakis/NDN-Link-Layer-Reliability-Scenario. Specifically, they include simulations on the following topologies:

1) A linear topology of fixed size (6 nodes).
2) A linear topology of variable hops.
3) The Abilene topology [23] with 12 nodes and 15 links.

For each of these topologies, we evaluate the consumer observed performance (consumer data retrieval delay and the number of retransmissions) and the network observed performance (spurious link layer retransmissions, loss detection

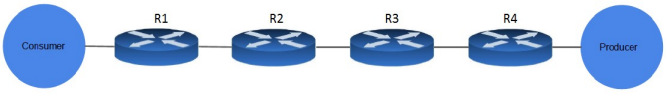


Fig. 1: A linear topology of hop length 5

time, and percent of recovered losses). In the case of the Abilene topology, we also evaluate the effect of in-network caching to the hop-by-hop link layer reliability in terms of the data retrieval time and the cost of BELRP in terms of buffer utilization and bandwidth overhead.

A. Simulation Scenario and Setup

We study the performance of the protocol in terms of the following parameters:

1) Consumer retransmissions,
2) Data retrieval time,
3) Link layer retransmissions,
4) Loss detection time,
5) Percent of losses recovered,
6) Buffer utilization, and
7) Total traffic generated.

We first use a simple linear topology to examine the data retrieval time, the consumer retransmission probabilities, and the percentage of packet losses recovered by BELRP. We have the consumer application installed on one end node, and the producer application installed on the other end node, with 4 routers in between (Figure 1). The parameters used in this simulation are shown in Table I. The line utilization of all the links is around 82% when using these parameters.

In our simulations, the size of an Interest packet, Data packet, and a sequence number are 28, 1061, and 4 bytes, respectively. Each piggybacked ACK is of size 4 bytes. Except for the simulation scenario in Section IV-D, in-network caching is enabled for all other simulations.

TABLE I: Simulation parameters for the linear topology

link capacity	10Mbps
propagation delay	5ms
loss distribution	uniform
Interest rate at consumer application	900 Interests/sec
number of Interests sent by consumer	20000
Interest packet size	28 bytes
content size of Data packet	1024 bytes
piggybacked ACK size	4 bytes

To verify the results obtained from the simple linear topology, we used the Abilene topology. We randomly selected 7 consumer-producer pairs with varying number of path lengths from 2 to 5 hops, each consumer sent $10,000$ Interests during the simulation run. For the simulation purposes, we scaled down the link capacities of all the links in the Abilene topology by a factor of $1,000$. In order to allow comparison with the results from the previous simulation, we adjusted the Interest rates of the consumer applications to keep 82% average utilization of the bottleneck links.

In all simulations, consumers generate Interests following a Poisson distribution model and ensure data fetching reliability using a retransmission timer similar to that of TCP [24]. Because our simulations do not have background traffic, all observed RTT values have rather small variations, i.e., the value of $RTTVAR$ is small, and consequently the retransmission timer value of $SRTT + 4 \times RTTVAR$ is very close to the value of $SRTT$. To prevent consumers from frequent false timeout, we modified the RTO setting to be $SRTT + 8 \times RTTVAR$ instead of $SRTT + 4 \times RTTVAR$.

Since BELRP introduces additional delays in data retrieval, it increases the chance of false Interest retransmissions by end consumers. As mentioned in Section III-B, this duality represent a trade-off between how hard the link layer should try for loss recovery and how much cost one is willing to pay. To study this trade-off, we obtained the results from different maximum number of retransmissions by the link layer sender.

B. Consumer Observed Performance

All the results shown in this section are collected from 20 simulation runs using the same set of parameters.

1) Consumer Retransmissions: The number of consumer retransmissions is the sum of the number of real losses experienced by the consumer applications and false retransmissions caused by the delay of BELRP. Figure 2a shows the counts of consumer retransmissions for the linear topology with hop count 5 and various packet loss probability over each link. To understand the impact of path lengths, we present the results obtained for the linear topology with different hop lengths in figure 2b when the packet loss probability over each link is set to 5%.

Intuitively one would expect that, as either the loss probability or the hop length increases, a growing number of packets would be lost across the network and the number of consumer retransmissions would increase accordingly. This is true in the absence of BELRP, i.e., loss recovery by end-to-end reliability only. The trend is the opposite in the presence of BELRP: Figure 2a shows that, except the case of BELRP retransmitting lost packets only once, the number of consumer retransmissions goes down as the packet loss probability goes up; Figure 2b shows that, for the fixed link loss probability, the number of consumer retransmissions goes down as the path length goes up.

Since the actual number of packet losses over the network should not go down with either higher loss rate per link or higher path lengths, one can only attribute the reduction of consumer retransmissions to the reduction of spurious consumer retransmissions: as more packets are lost and recovered by BELRP, the end-to-end delay variations increase, and the RTO value goes up by a factor of 4. Consequently, the number of spurious consumer retransmissions decreases. To provide further evidences of our reasoning, Figures 2c, 2d plot the number of unnecessary consumer retransmissions with respect to the link loss rate, and the hop length respectively. These results show that, at least within the scope of our simulation parameter range (5% or lower packet loss rate per link, 7 hops or less path length), most of the consumer retransmissions are spurious when the network runs BELRP with the number of retransmission set to 2 or higher, suggesting that consumer

applications should use a more conservative RTO setting when the network runs BELRP.

2) Data Retrieval Time: Data retrieval time is the interval between an Interest's first transmission and the receipt of the corresponding Data packet by the consumer. BELRP improves the data retrieval time by reducing the dependency on end-to-end loss recoveries.

Table II summarizes the results for the linear topology of hop length 5 with varying link loss probability, and Table III shows the results for the same topology but varying the number of hops. In order to have adequate comparisons of the data retrieval time between line topologies of different hop lengths, we present the results as a percentage of the ideal delay value (the sum of the link propagation delays, with zero queueing delay).

Surprisingly, we observed that the ratio between the actual and the ideal retrieval times decreases as the number of hops increases. This is the result of the decreasing number of consumer retransmissions (Section IV-B1). A false Interest retransmission by the consumer can result in the corresponding Data packet being sent again over some hops, increasing link load and hence the queuing delay. Reductions of spurious consumer Interest retransmissions lead to decrease of queue sizes at routers, hence the reduction of consumer data retrieval time.

The above results show that BELRP is effective in reducing data retrieval time, especially as the hop length increases. When the consumer solely relies on retransmission timers to detect losses, since the timer is set to a conservative value to minimize spurious retransmission, the detection takes a relatively long time. Moreover, the re-expressed Interest may have to travel a number of hops before meeting the requested data, all adding to loss recovery time. As the hop length increases, increasing number of packets may get lost across the network, which have to undergo costly loss recovery process, therefore the data retrieval time increases rapidly when one solely relies on end-to-end reliability protocol. On the other hand, BELRP detects losses within a link's round-trip time, and each loss recovery is over a single hop, taking much less time than end-to-end recovery.

It is obvious that the data retrieval time in the presence of packet losses is high if one relies on end-to-end reliability protocol for loss recovery. Interestingly, the data retrieval time of the Interests which are not lost is also high in the case of the end-to-end reliability protocol. This is because the consumer recovers losses by retransmitting the Interests corresponding to the lost packets. End host retransmissions can be queued at multiple routers, causing additional delays for the packets which are not lost across the network. On the other hand, link retransmissions are queued on a single router and hence cause smaller delays. This fact can be better illustrated in the cumulative distribution function (CDF) of the percentage of Interest-Data exchanges satisfied versus the data retrieval time shown in Figure 3. We define an Interest-Data exchange as the transmission of an Interest packet from the consumer to the producer and the transmission of the corresponding Data packet from the producer. We observe that about 53% of the Interest-Data exchanges are not lost across the network and are satisfied in roughly the same time (marked with the first arrow in the figure). 85% of the Interest-Data exchanges are satisfied after one link layer retransmission (marked with the second arrow in the figure) with an additional delay of 11–16 ms compared to the previous case. The overall RTT is about 56 ms. As the results suggest, the retrieval time for the Interest-Data exchanges which are not lost is also high when only the end-to-end reliability protocol is used.

C. Network Observed Performance

Below we examine the performance of BELRP in terms of 1) spurious link layer retransmissions, 2) loss detection time, and 3) percent of losses recovered.

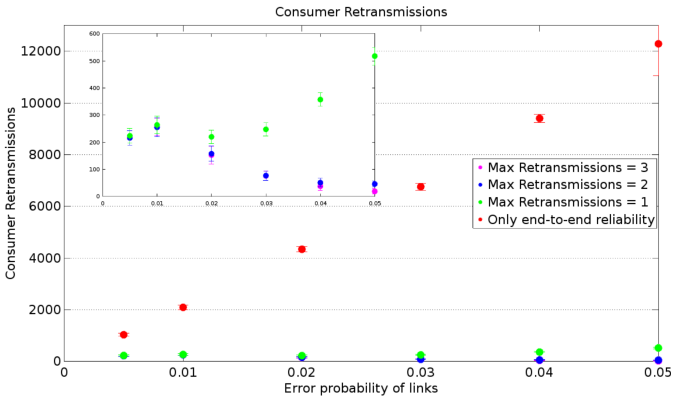
1) Spurious Link Layer Retransmissions: As discussed in section III-D, a sender can retransmit spuriously if either an ACK is lost or multiple packets arrive out-of-order within the link. Spurious retransmissions increase the link utilization and are counted as one source of BELRP's overhead.

We examine the number of link layer retransmissions in the Abilene topology. Figure 4a presents the results from the WASHng-ATLAng link with bidirectional traffic and about 82% link utilization. As the loss probability increases, an increasing number of frames is lost, and the number of link layer retransmissions increases, however, the percentage of retransmissions is in harmony with the packet loss probability of the link. Figure 4b shows the number of spurious link layer retransmissions for the same link. Because of the limitations of our current simulation setting (which does not produce out-of-order arrivals), we show only the number of spurious retransmissions due to ACK losses. The results show that the number of spurious link layer retransmissions are negligible when redundant ACK mechanism is used.

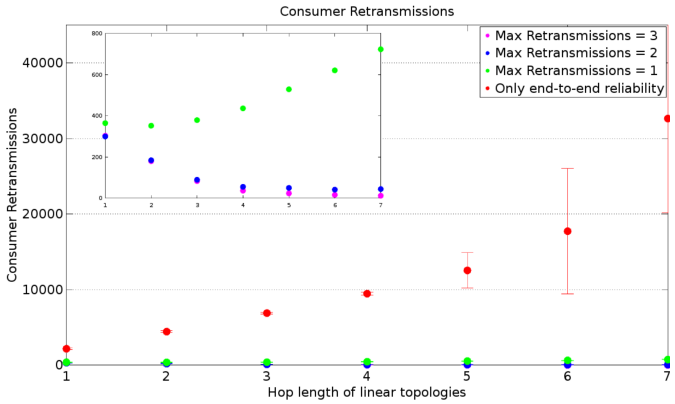
2) Loss Detection Time: This is defined as the time interval between a frame's transmission time and the detection of its loss by the sender. Again we examine the loss detection time of the WASHng-ATLAng link in the Abilene topology. Figures 5a and 5b summarize the results for ATLAng, WASHng nodes respectively. Note that RTT delay for the link is approximately 11.15 ms.

We observe that, as the loss probability increases, there is a slight increase in the loss detection time. This is due to the fact that the increasing number of link layer retransmissions leads to an increasing queuing delay, and subsequently increasing the time taken for up to 3 subsequent ACKs to reach the sender.

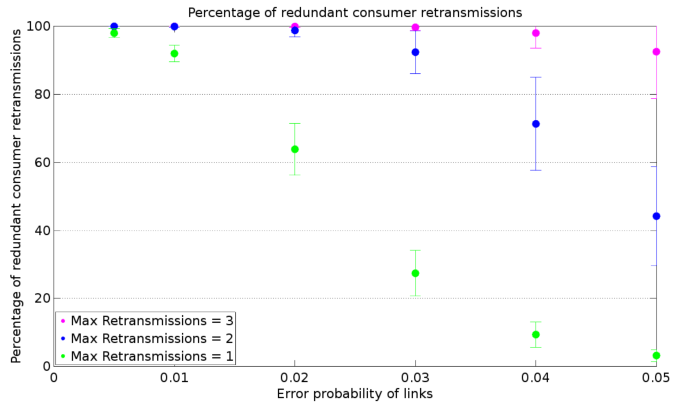
3) Percentage of Losses Recovered: An Interest-Data exchange is said to be lost if the Interest or its corresponding Data packet is lost at least once across the network. When an Interest-Data exchange is lost, the end host must recover the loss when the network provides no loss recovery. With BELRP, a lost Interest-Data exchange is said to be recovered by BELRP if it is recovered every time the Interest or its corresponding Data is lost when crossing the network. For example, assuming that an Interest is lost three times when crossing the network, the link layer protocol is said to recover the lost Interest-Data exchange if and only if BELRP is able to recover the loss all the 3 times.



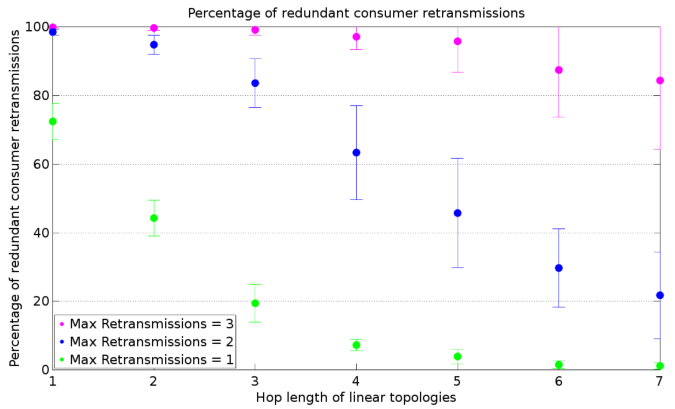
(a) The consumer retransmissions (for 20000 Interests) for the linear topology of hop length 5. The error bars in the plot indicate 90% confidence intervals.



(b) The consumer retransmissions (for 20000 Interests) for the linear topology of various hop lengths at 5% loss probability. The error bars in the plot indicate 90% confidence intervals.



(c) The number of spurious consumer retransmissions (as a percent of the total number of consumer retransmissions) for the linear topology of hop length 5. The error bars in the plot indicate 90% confidence intervals.



(d) The number of spurious consumer retransmissions (as a percent of the total number of consumer retransmissions) for linear topology of various hop lengths at 5% loss probability. The error bars in the plot indicate 90% confidence intervals.

Fig. 2

TABLE II: The data retrieval time (as percent of its ideal value) for the linear topology of hop-length 5. The results are summarized for 20 runs, where the consumer sends 20000 Interests during each run

Loss Probability	3 Maximum Retransmissions		2 Maximum Retransmissions		1 Maximum Retransmission		End-to-end loss recovery	
	Mean	90% Confidence Interval	Mean	90% Confidence Interval	Mean	90% Confidence Interval	Mean	90% Confidence Interval
1%	106.09	100.09-125.49	106.08	100.09-125.49	106.17	100.09-125.52	117.30	100.904-242.95
2%	109.07	100.10-130.92	109.08	100.10-130.77	109.57	100.11-131.14	134.93	100.904-283.36
3%	112.18	100.11-140.66	112.20	100.11-140.56	113.37	100.11-140.65	156.68	100.904-363.65
4%	115.35	100.11-148.69	115.44	100.11-148.52	118.06	100.11-150.04	185.67	100.904-468.24
5%	118.72	100.13-153.75	118.86	100.13-153.64	123.34	100.13-156.98	224.70	100.904-577.49

TABLE III: The data retrieval time (as percent of its ideal value) for the linear topology of various hop-lengths at 5% loss probability. The results are summarized for 20 runs, where the consumer sends 20000 Interests during each run

Hop Length	3 Maximum Retransmissions		2 Maximum Retransmissions		1 Maximum Retransmission		End-to-end loss recovery	
	Mean	90% Confidence Interval	Mean	90% Confidence Interval	Mean	90% Confidence Interval	Mean	90% Confidence Interval
1	132.90	100.08-236.35	132.79	100.08-236.76	133.40	100.08-236.76	159.49	100.895-508.01
2	124.07	100.08-182.88	124.07	100.08-182.79	125.69	100.08-182.62	168.07	100.895-438.79
3	121.16	100.10-172.94	121.32	100.10-173.04	123.94	100.10-172.76	180.79	100.904-477.90
4	119.58	100.12-161.94	119.68	100.12-161.58	123.27	100.12-163.34	195.65	100.904-500.52
5	118.89	100.13-154.12	119.08	100.13-154.13	123.58	100.13-157.45	225.13	100.904-577.49
6	118.03	100.26-149.39	118.23	100.25-149.33	123.59	100.30-155.80	401.91	101.18-651.11
7	117.87	100.58-148.17	118.15	100.57-148.09	124.30	100.59-154.91	302.66	102.67-810.84

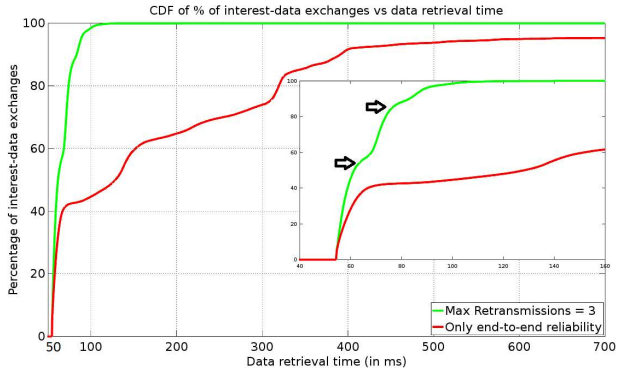


Fig. 3: Cumulative distribution function of percentage of Interest-Data exchanges satisfied vs Data retrieval time (linear topology of hop length 5 with a loss probability of 5% for all the links)

In this evaluation case, we examine the percentage of the lost Interest-Data exchanges recovered by BELRP. One should note that this metric is different from the percentage of lost frames recovered. Figure 6a presents the results for the linear topology of various hop lengths. Given that the link's packet loss probability is 5%, if a frame is lost across the link, the first retransmission also has a 5% chance of being lost. The probability that two retransmissions are lost across the link is 0.25%, and the probability that the 3 retransmissions are also lost across the link is 0.0125%. These statistics are reflected in Figure 6a. The results also indicate that the performance of the protocol in recovering losses is resilient against the hop length increase.

Figure 6b summarizes the results from the Abilene topology. We observe that, as the loss probability increases, the chance for an unrecovered Interest-Data exchange loss across the network also increases slightly, which can be negligibly small if the number of retransmissions is set to 2 or higher.

D. Hop-by-hop Reliability and In-network Caching

NDN features in-network caching, so that future requests for the same Data would be satisfied directly from caches instead of reaching the producer application. Caching also contributes to the improvement of the data retrieval time in case of Data packet losses, as the re-expressed Interests for the lost Data would be satisfied by the router just before the point where the Data got lost. However, there is no equivalent in-network caching for lost Interest packets. When an Interest gets lost across the network, the original consumer application needs to re-express the Interest if it still wants the Data.[3]

BELRP operates at the link layer, where both Interests and Data packets are treated the same way as link layer frames. It acts in a complementary way to the beneficial effect of in-network caching and further reduces data retrieval time.

We performed simulation studies on Abilene topology (discussed in IV-A) to compare data retrieval time for 3 cases:

- In-network caching with BELRP
- Only in-network caching
- Neither in-network caching nor BELRP

Note that end-to-end reliability is used in all the above cases; it recovers losses that BELRP, as a best effort protocol, fails to recover.

The results are summarized in Table IV. We observe that as the loss probability increases, the effect of the hop-by-hop reliability is beneficial in reducing data retrieval time. This comes as a result of the growing number of frames being lost, which, without BELRP, would need to be recovered by the end application retransmissions.

E. BELRP Cost Evaluation

Along with the performance improvement, BELRP incurs additional cost in terms of a) buffer space required to store the state information at every intermediate router and b) additional bytes to be transmitted to carry ACKs and sequence numbers. We use the simulation results from the Abilene topology to examine the cost. The results are explained in the following subsections.

1) Buffer Utilization: BELRP requires a buffer space at every router to store all the outstanding frames for which an ACK has not been received, so that a lost frame can be retransmitted.

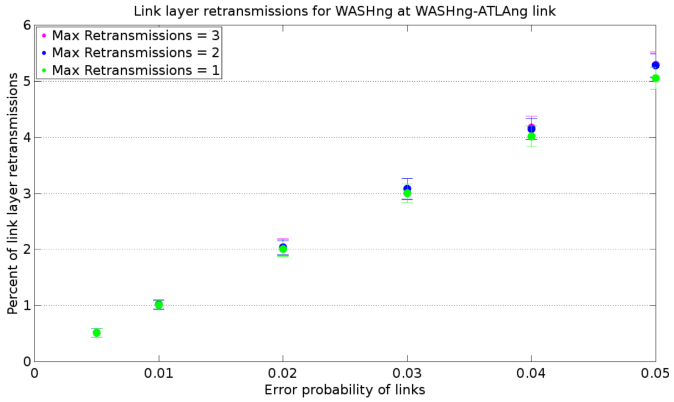
We measure the maximum buffer size used (in number of packets) during our simulations. Table V summarizes the results for the WASHng-ATLAng link. In order to calculate the ideal buffer utilization, we perform the same simulation with loss probability equal to 0 and with the consumer sending Interests with a constant rate. The buffer utilization in this scenario is 26 packets, representing the bandwidth $\times$ round-trip-time product of the link.

As shown in Table V, the buffer utilization at the sender increases as the link loss probability increases. This increase in the buffer utilization can be explained by the following two factors.

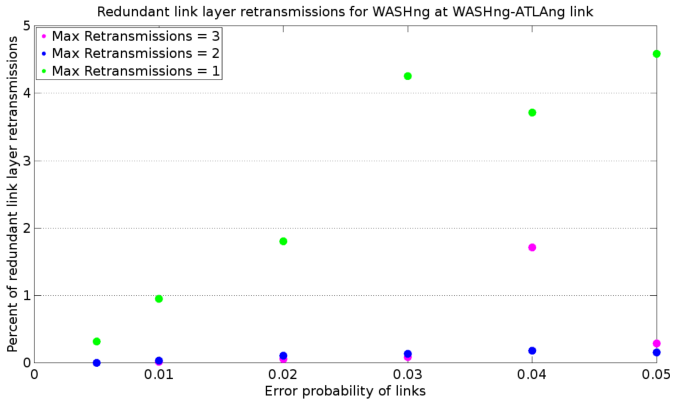
- A frame has to be stored in the buffer until it is acknowledged, and a lost frame has to be stored in the buffer until an ACK for the retransmitted frame is received, i.e. the lost frames stay in the buffer for a longer time. As the loss rate of the link increases, an increasing number of frames get lost across the link, thus the buffer utilization increases.
- When the loss rate increases, the number of link layer retransmissions increases. This results in higher link utilization and more queueing at the sender, effectively increasing the round trip pipe size and hence the buffer utilization.

2) Bandwidth Overhead: BELRP incurs bandwidth overhead due to its sequence numbers, acknowledgments, and link layer retransmissions, which increase the number of bytes transmitted by every node and, thus, the link utilization. This overhead may be considered critical in the case of high link utilization. However, examining the cost of these individual factors does not give an idea of the overall cost incurred by the protocol because BELRP reduces end-to-end retransmissions,

[3]Our simulations use a simple Interest forwarding strategy that does not resend failed Interests. An adaptive forwarding strategy with NACK [25] can speed up recovery by re-expressing Interests along different paths.

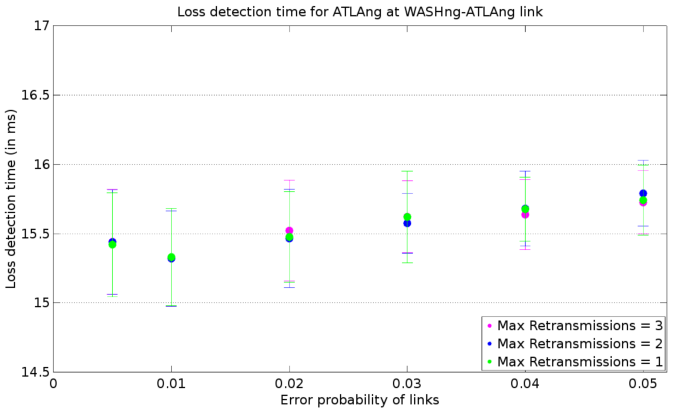


(a) Link layer retransmissions (as percentage of total number of frames to be transmitted ideally) for WASHng node at the WASHng-ATLAng link of the Abilene topology. Error bars indicate 90% confidence interval.

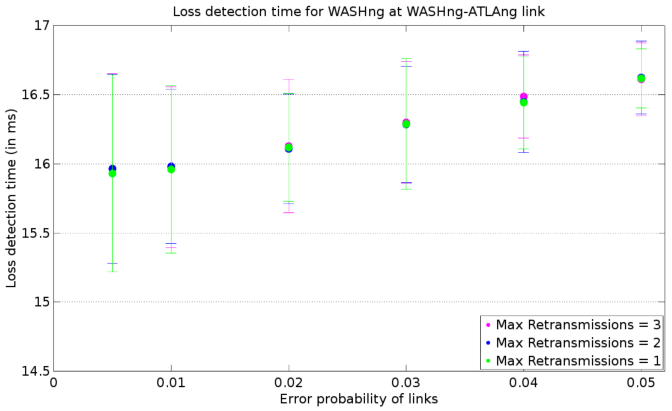


(b) Spurious link layer retransmissions (as percentage of total number of the retransmissions) for WASHng node at the WASHng-ATLAng link of the Abilene topology.

Fig. 4: Measurement of link layer retransmissions and spurious retransmissions.

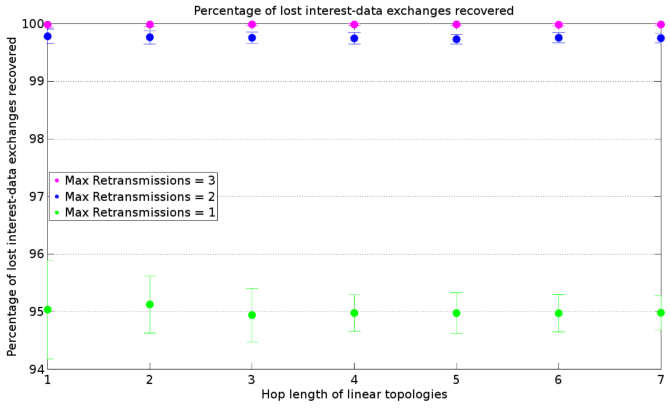


(a) Average time taken for ATLAng node to detect loss at link layer. Error bars represent 90% confidence intervals.

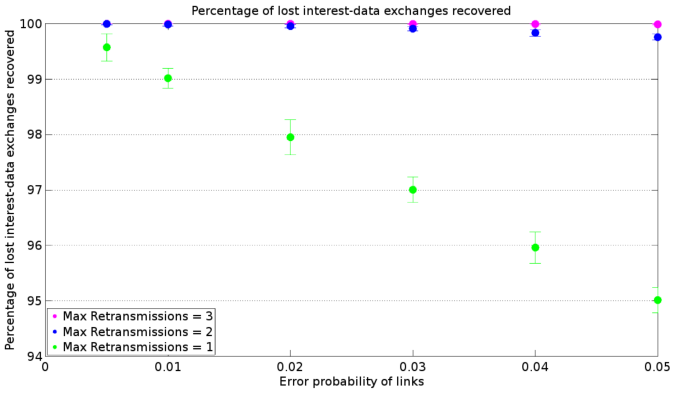


(b) Average time taken for WASHng node to detect loss at link layer. Error bars represent 90% confidence intervals

Fig. 5



(a) Percentage of the lost Interest-Data exchanges recovered by the protocol for the linear topology of various hop lengths at 5% loss probability for all the links. Error bars indicate 90% confidence intervals



(b) Percentage of the lost Interest-Data exchanges recovered by the protocol for the abilene topology. Error bars indicate 90% confidence intervals

Fig. 6

TABLE IV: Comparison of data retrieval time by the mean with 90% confidence interval

Loss probability	Caching, Link layer reliability	Only caching	No caching, No link layer reliability
0.5%	117 ± 28	120 ± 61	118 ± 64
1%	119 ± 31	127 ± 90	125 ± 100
2%	123 ± 37	145 ± 138	142 ± 152
3%	126 ± 43	166 ± 187	173 ± 210
4%	130 ± 49	194 ± 247	208 ± 281
5%	134 ± 55	234 ± 331	471 ± 3359

TABLE V: Maximum buffer size used (in packets) at the sender, with the mean & 90% confidence interval calculated from 20 simulation runs.

Loss probability	3 Maximum retransmissions	2 Maximum retransmissions	1 Maximum retransmission
0.5%	62 ± 12	62 ± 12	62 ± 13
1%	62 ± 13	63 ± 14	63 ± 13
2%	63 ± 13	64 ± 13	64 ± 15
3%	65 ± 17	65 ± 16	67 ± 16
4%	67 ± 16	67 ± 17	70 ± 21
5%	72 ± 17	71 ± 20	71 ± 20

hence the link utilization. Accordingly, we should study the total traffic generated across the network, which refers to the sum of the total number of bytes transmitted by every node during the entire simulation. This is a general metric which accounts for both the cost due to sequence numbers, ACKs, link retransmissions as well as the gain due to the decrease of consumer retransmissions. Figure 7 summarizes the results from the Abilene topology.

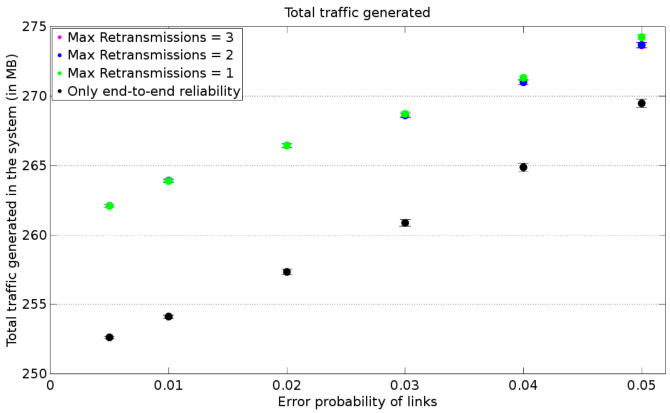


Fig. 7: The total traffic generated (in bytes) by all the nodes of the Abilene topology during the entire simulation. Error bars in the plot indicate 90% confidence interval.

We observe that the overhead due to the sequence numbers and the ACKs dominate the gain from reduced consumer retransmissions at low link loss rates, whereas the approach becomes more favorable at higher loss rates.

V. Discussion

A. Adaptive Link Layer Reliability

Our protocol can be extended to provide adaptivity to the loss rate, packet size, and out-of-order frame delivery. We have identified the following three possible adaptations:

- When the loss rate is low, keeping a high degree of ACK redundancy is not cost-effective, as the first ACK would have a good chance to successfully reach the sender. In such cases, we may consider removing ACK redundancy together with the associated bandwidth overhead.
- For small-size packets, spurious link retransmissions become cheaper in terms of the traffic volume. Transmitting an ACK multiple times might be more expensive than letting the sender retransmit the original packet. The receiver could maintain a threshold on the incoming packet size, below which it adopts a smaller ACK redundancy amount.

The simulations presented in Section IV use hardwired parameters and represent results on the conservative end, which can be further improved with an adaptive BELRP.

B. Link Layer Reliability Persistency

BELRP recovers most of the packet losses at the cost of increasing the data retrieval delay, which can trigger spurious retransmissions by higher layer end-to-end reliability mechanisms. Figures 2c and 2d show that when the link layer tries harder to recover losses, the percentage of spurious end-to-end retransmissions would go up. However, if one can relax the setting of end-to-end retransmission timers, one may be able to more fully exploit the benefit from link layer protocol capabilities.

C. Alternatives to Improve ACK Loss Rate

ACK losses lead to spurious link layer retransmissions. We have proposed to increase the chance of ACK delivery by transmitting them redundantly, which incurs the bandwidth overhead. However, there are also other means to increase ACK delivery reliability. One way could be through intelligent coding, e.g., using Forward Error Correction (FEC) to decrease the loss rate [26]. Interleaving the ACKs that refer to the same frame (instead of sending them in consecutive frames) could also reduce ACK loss rate due to burst errors. However, this may add further delay to ACK delivery. When the loss probability of a frame depends on its size, sending stand-alone ACKs also helps in improving the loss rate of ACKs.

D. NACK-based Protocol

BELRP uses ACKs to detect losses. One could also use NACKs (Negative Acknowledgements) to do so. In other words, the receiver could detect losses based on the gaps in the arriving frame numbers and inform the sender by sending back a NACK containing the sequence number of the missing frame. The sender, upon receiving the NACK, can then retransmit the corresponding frame.

This protocol uses additional traffic only for the lost frames, but at the cost of reduced probability of detecting the loss. For example, when a NACK is lost, the sender does not have any knowledge about it, and, therefore, does not retransmit the frame. The problem can be mitigated by transmitting each NACK multiple times.

Another concern about a NACK scheme is that it informs the sender about losses, but not about successful frame deliveries. Consequently, the sender may not have a good way to decide when to flush a frame out of the buffer.

E. BELRP for Multiaccess Links

Note that the existing BELRP design considers only point-to-point links. Over a multiaccess link, the same design should work if the frame exchanges are between two specific nodes on the link. However, if one multicasts a frame, e.g., when multiple consumers request the same Data packet, it remains an open question of how best to pass the ACKs around.

The most common multiaccess links are wireless links, and wireless access enables mobility which leads to another interesting question related to the BELRP design. BELRP assumes *static* point-to-point links, so that the nodes at the two ends can keep track the (packet, frame) sequence numbers. When nodes move dynamically, it brings up the question of how to synchronize the sequence numbers between two communicating nodes over a wireless channel.

VI. Conclusions

In this paper, we described the design of a hop-by-hop best effort link layer reliability protocol (BELRP), and evaluated its performance and cost through a simulation study. The protocol recovers from packet losses at the same hop of their occurrence, improves the delivery rate with a small increase in delay, as seen by the end application. Our evaluation results demonstrate that best effort hop-by-hop retransmissions can lead to notable performance gains for applications.

We observe that the protocol works well even when the path lengths increases, leading to a low number of retransmissions by higher layer end-to-end reliability schemes; it has a resilient loss recovery capability against path length and well as loss rate increases. It is worth mentioning that the results presented in our evaluation are on the conservative end, as we have identified several optimizations to make BELRP adaptive to the network conditions.

Finally, we would like to share with the research community several lessons that we learned during the design process:

- Recovering packet losses by the end hosts is a costly process, the performance of which can be enhanced by the use of a hop-by-hop reliability scheme.
- A best effort hop-by-hop reliability protocol acts as a performance enhancement but not a replacement for end-to-end reliability schemes.
- Relaxing the end-to-end retransmission timers can enable our link layer protocol to exploit its full capacity.

References

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM Computer Communication Reviews*, June 2014.

[2] R. Rejaie, M. Handley, and D. Estrin, "Rap: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 1999, pp. 1337–1345.

[3] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "Tcp selective acknowledgment options," 1996.

[4] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "Tcp westwood: end-to-end congestion control for wired/wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 467–479, 2002.

[5] L. S. Brakmo and L. L. Peterson, "Tcp vegas: End to end congestion avoidance on a global internet," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 8, pp. 1465–1480, 1995.

[6] J. H. Saltzer, D. P. Reed, and D. Clark, "End-to-end arguments in system design," in *Second International Conference on Distributed Computing Systems*, 1981, pp. 509–512.

[7] *Specification for X.25 LAPB-Compatible DTE Data Link Procedures*, ISO 4335/4, International Standards Organization, 1985.

[8] M. Gast, *802.11 wireless networks: the definitive guide*. " O'Reilly Media, Inc.", 2005.

[9] H. Lee, Y. Ko, and D. Lee, "A hop-by-hop reliability support scheme for wireless sensor networks," in *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*. IEEE, 2006, pp. 5–pp.

[10] G. Wagenknecht, M. Anwander, and T. Braun, "Hop-to-hop reliability in ip-based wireless sensor networks-a cross-layer approach," in *Wired/Wireless Internet Communications*. Springer, 2009, pp. 61–72.

[11] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *ACM SIGCOMM computer communication review*, vol. 36, no. 4. ACM, 2006, pp. 243–254.

[12] L. Zhang, "Why tcp timers don't work well," in *ACM SIGCOMM Computer Communication Review*, vol. 16, no. 3. ACM, 1986, pp. 397–405.

[13] I. Psaras and V. Tsaoussidis, "Why tcp timers (still) don't work well," *Computer Networks*, vol. 51, no. 8, pp. 2033–2048, 2007.

[14] R. Ludwig and R. H. Katz, "The eifel algorithm: making tcp robust against spurious retransmissions," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 1, pp. 30–36, 2000.

[15] P. Sarolahti, M. Kojo, and K. Raatikainen, "F-rto: an enhanced recovery algorithm for tcp retransmission timeouts," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 51–63, 2003.

[16] A. Gurtov and R. Ludwig, "Responding to spurious timeouts in tcp," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3. IEEE, 2003, pp. 2312–2322.

[17] H. Ekström and R. Ludwig, "The peak-hopper: A new end-to-end retransmission timer for reliable unicast transport," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4. IEEE, 2004, pp. 2502–2513.

[18] W. R. Stevens, "Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," 1997.

[19] R. Kanodia, "A lost message detection and recovery protocol. 1974. available from: Stanford research institute, augmentation research center, menlo park," CA.(RFC# 663, NIC# 31387), Tech. Rep., 1974.

[20] J. Roskind, "Quic: quick udp internet connections," https://docs.google.com/document/d/1gY9-YNDNAB1eip-RTPbqphgySwSNSDHLq9D5Bty4FSU/edit, December 2013.

[21] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," Tech. Rep. NDN-0028, 2015.

[22] A. Afanasyev, J. Shi *et al.*, "NFD Developer's Guide," Tech. Rep. NDN-0021, 2015.

[23] "Abilene network topology," http://web.archive.org/web/20080113120821/http://abilene.internet2.edu/.
[24] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing tcp's retransmission timer," Tech. Rep., 2011.
[25] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013, iSSN 0140-3664. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2013.01.005
[26] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1993.