Leveraging the Power of Cloud for Reliable Wide Area Communication

Osama Haq and Fahad R. Dogar
Tufts University

Abstract

We make a case for judicious use of cloud infrastructure – as an overlay to aid IP's best effort service. As an example, we propose ReWAN, a packet recovery service for real time, wide area communication. ReWAN uses cloud-based edge proxies that exchange "recovery" packets, which are used *only* in case of packet loss. ReWAN leverages the cloud provider's well-connected, global data center network, and its ability to handle many concurrent users. To minimize bandwidth cost, it uses coding to generate a small number of recovery packets that are sent across the inter data center network. The recovery packets use both FEC, which is applied within a user stream, and network coding, which is applied across user streams. If a small fraction of packets within the user stream are lost, ReWAN uses the FEC packets for recovery; for other losses, ReWAN recovers them with the help of other receivers and the network coded packets. Preliminary measurements show the promise of ReWAN in providing a fast and cost effective packet recovery service.

Categories and Subject Descriptors

C.2 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design

Keywords

Reliability, Internet; Packet Loss; Data Center

1. INTRODUCTION

The need to go beyond IP's best effort service model is well-known. The late 90's and early 2000's saw a slew of proposals that used *overlays* to provide rich network services, such as: resilient routing [6, 15], multicast [26], anonymity [13], mobility [29], QoS [30], and content based services [14]. The use of end-points as overlay nodes turned out to be cost effective and easy to deploy, but created challenges of performance and availability, which the above proposals had to contend.

Motivated by changes in technology, specifically the emergence of the "cloud", we revisit the role of overlays in providing richer network services. Instead of end-points, we consider the use of *data centers* (DC) as overlay nodes; these DCs form an overlay network to provide services to end-points. This allows services to be easily scaled up and down, and high performance and availability can be achieved using the cloud's global, well-provisioned infrastructure. However, using cloud as an overlay can be costly: cloud providers charge for the use of their resources (e.g., processing, network, etc), with WAN bandwidth being particularly expensive to use [18, 33].

In this paper, we make a case for using cloud based overlays in a *judicious* manner – to *supplement* the best-effort service provided by the Internet, thus exploiting the benefits of the cloud without incurring excessive cost. While this approach can potentially benefit different scenarios (e.g., QoS, multicast), the focus of this paper is on reliable, wide area communication for latency sensitive applications. Such applications experience packet loss and jitter in today's best effort Internet. Unfortunately, traditional retransmission based recovery (ARQ) is too slow in a wide-area setting while forward error correction (FEC) based mechanisms have limited efficacy due to the bursty nature of losses. A reliable and low latency wide area service will benefit many current applications (e.g., gaming, video conferencing, short web transfers, etc) as well as emerging real time applications (e.g., remote patient monitoring).

Towards this goal, we sketch the design of ReWAN, a cost-effective cloud-based *packet recovery* service for wide area communication. ReWAN uses a proxy based design: source(s) send a copy of their data to a nearby cloud proxy (P1) which sends a small number of *recovery* packets across its inter-data center link to a cloud proxy (P2) close to the receiver(s). The recovery packets use both forward error correction (FEC), which is applied within a user stream, and network coding, which is applied across user streams. If a small fraction of packets within the user stream are lost, the DC uses the FEC packets for recovery; for other losses, the DC undertakes a "cooperative recovery" – it gets packets from other receivers and combines them with the network coded packets to generate the missing packet(s).

ReWAN's design exploits the unique aspects of the cloud, including low latency edge access, concurrent users, and highly reliable but expensive inter-DC paths, by applying two well-known strategies: i) use of FEC and network coding to reduce the recovery overhead, and ii) use of proxies for fast, local retransmissions at the edges. We conduct a feasibility analysis of ReWAN with the help of experiments on PlanetLab and Google Cloud. Our measurements show that for typical inter-continental paths, ReWAN can recover packet losses within a fraction of the round trip time while only consuming a small fraction of cloud bandwidth.

ReWAN's design is preliminary and likely to evolve in future. However, it serves as a good example of the opportunities and challenges in using the cloud to support richer network services. We believe that ReWAN's initial design and feasibility analysis would serve as a starting point for a future research agenda on how to overcome the limitations of today's best effort Internet with the help of the cloud.

HotNets '15, November 16–17 2015, Philadelphia, PA USA

http://dx.doi.org/10.1145/2834050.2834109

2. HOW TO BEST USE THE CLOUD?

2.1 Advantages

We focus on two key advantages of using the cloud as an overlay[1] – performance and availability, factors considered a bane for end-host based overlays.

Availability. In traditional overlays, availability can be low because end-hosts can fail, or can arbitrarily leave or join the overlay network (i.e., churn). Availability can also suffer because of access link dynamics (e.g., link failures or congestion). In contrast, DC's have much higher availability. Cloud providers guarantee between three to four 9's of availability i.e., 99.9% to 99.99% uptime for VMs. DC networks are also more reliable: for connectivity with end-users, DCs typically leverage multi-homing while for inter-DC links, cloud operators have dedicated or well provisioned infrastructure [19], which is more reliable than traditional Internet paths and can provide up to five 9's of availability with the use of standard techniques (e.g., FEC) [17].

Performance. The limited capacity of end-hosts can cause queue build up for various resources (e.g., processing, storage) under high load, leading to poor performance. The limited capacity also results in long overlay paths, if a large number of users need to be supported. This path stretch can add high latency which is detrimental for latency sensitive applications. In contrast, DC's have good network connectivity amongst themselves as well as with end-hosts. Cloud operators put considerable effort in ensuring that their data centers have low latency access to end-users. DC's also do well under high load; they have elastic supply of resources, so they are well equipped to respond to spikes in loads, both for processing and other resources.

2.2 Cost

Cloud operators charge for the use of their resources and the pricing model can be fairly elaborate for some resources (e.g., spot pricing for Amazon EC2 compute [1]). Bandwidth pricing is relatively straightforward in comparison but still has some caveats that apply across major cloud operators. For example, providers do not charge for ingress bandwidth and only charge for egress bandwidth, and the cost varies depending on the region: US egress bandwidth is typically cheaper than the egress bandwidth usage in Asia.

We do a back-of-the-envelope calculation for the bandwidth cost incurred in supporting a video conferencing service, using cloud as an overlay. We consider a wide area setting with users in two different continents (US and Asia) communicating via their nearby DCs. This is similar to how the Google Hangout service uses the cloud infrastructure [32]. Given the typical pricing model, the above communication will incur the egress bandwidth charges at both the locations (US and Asia).

To support an average of 1000 concurrent video conferencing sessions, the monthly bandwidth cost of the service comes out to be between $120K - $150K for the three major cloud providers (Amazon [1], Google [3], and Microsoft [2])[2]. This is based on a unidirectional bandwidth requirement of 1.2Mbps for each session. To put these numbers in context, most multi-player games also have similar bandwidth requirement and can have on the order of 300K concurrent users. So the above calculations show that even with a conservative estimate, bandwidth charges are significant and likely to be the dominant cost for network based overlay services.

2.3 Judiciously Using the Cloud

We advocate that in many scenarios cloud can be a supplement to existing Internet (or end-host based overlays). We can rely on the performance and availability of the cloud, but only leverage its features when required. This is based on the observation that services typically need to meet some service level objective (SLO) and Internet's best effort nature often gets us closer to the SLO or may even meet the SLO at certain times. The cloud can come into the picture to enhance the best effort service of the Internet, helping to bridge the gap between application SLO and what the Internet can offer.

We envision two possible roles in which the cloud can help: using DC as a *backup* which is used in case of failures, and ii) using DC to cover up the performance limitations of the Internet. We provide three examples to illustrates these roles; the first two examples are discussed briefly, while the third example is used as a case study in the rest of the paper.

Multicast. Multicast services construct a tree which is used to deliver the content. Construction and maintenance of the tree is a major overhead and becomes a challenge under high churn rate [26, 10]. We can enhance traditional end-host based overlays by having a DC act as a backup node for all end-hosts within its vicinity. So each overlay node has a "virtual backup node" running in a nearby DC which takes over when the primary node fails. When the end-host is back up again, it can claim back its original role. This ensures that the multicast tree experiences minimal disruptions due to churn.

QoS. Many applications require an end-to-end bandwidth guarantee which is difficult to provide in today's best effort Internet. Even if access links support the required bandwidth, applications may observe significant fluctuations in the wide area setting. These applications can benefit from a cloud-based bandwidth guarantee service which uses its inter-DC links to bridge the gap between the available bandwidth on the Internet and the application's required bandwidth.

We have conveniently ignored many issues in the above scenarios. To illustrate the challenges and opportunities in such scenarios, we now present a detailed example, which shows how the cloud can be judiciously used for packet recovery.

3. PACKET RECOVERY AS A SERVICE

ReWAN is a wide area packet recovery service which targets latency sensitive applications (e.g., voice, gaming, short web flows, etc). Logically, it sits below the transport layer – it enhances the reliability of IP's best effort service, but transport layer's end-to-end reliability is still required, if applications desire so.[3]

3.1 Overview

Fig 1 shows the high level working of ReWAN through a simple example. There are two sender-receiver pairs (S1-R1 and S2-R2); the

[1]Providers like Microsoft and Google already use their data centers as overlays for applications like search [23] and video conferencing [32].

[2]Charge calculated based on the pricing on July 10, 2015

[3]Service-centric Internet architectures (e.g., XIA [16]) can natively support a service like ReWAN.

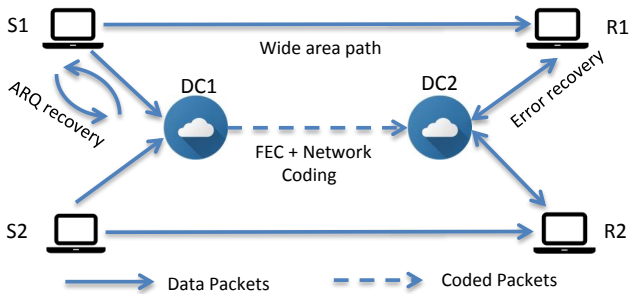


Figure 1: ReWAN design overview with two sender-receiver pairs. The two DCs only exchange coded packets over their inter-DC path. In case of a burst loss, DC2 undertakes cooperative recovery with the help of the other receiver. At the edges, we use local retransmissions (i.e., ARQ) with limited retries.

senders and receivers are in different continents. There are two DCs that are running the ReWAN service: DC1 is close to the senders while DC2 is close to the receivers. The sender and receiver communicate using IP's best effort service, just like how they do it today, but in addition, each sender also sends a copy of the data to its nearby ReWAN service (DC1).[4]

DC1 generates a small number of *recovery* packets, which are sent across to the recovery service at DC2, using the inter-DC path. The recovery packets include both FEC and network coded packets; they protect against common packet loss patterns (FEC for random loss and and network coding for bursty losses/outages).

Each receiver sends feedback (e.g., acknowledgements) to DC2; the feedback informs DC2 about the received packets, enabling it to undertake recovery, in case of packet loss. For some packet loss instances, FEC packets suffice, but in other instances, the DC may need to undertake a *cooperative* recovery, which entails contacting other receivers, getting their packets and combining them with the network coded packets to recover the lost packets. Finally, ReWAN uses local retransmissions (with a limited number of retries) at the edges (source-DC1 and DC2-receiver); benefits of such fast, localized retransmissions are well-known in the context of TCP splitting [23, 8, 11, 12, 5] .

Cost. ReWAN's design explicitly minimizes the bandwidth cost of using the cloud's egress bandwidth. This is reflected in two design decisions. First, ReWAN only exchanges a small amount of recovery packets between DCs. This is the major difference between ReWAN and a solution that *only* uses the cloud. Second, recovery between DC2 and the receiver is *on-demand*: this saves the egress bandwidth charges at DC2, albeit at the expense of (slightly) slower recovery.

3.2 Recovery Packets

We present a strawman design for generating recovery packets – it uses simple FEC and networking coding.[5] Fig. 2(a) shows an example of how recovery packets are generated. The example considers four user streams (A, B, C, D) with each stream having five data packets (indicated by their subscript). These packets are arranged in a square grid to help explain how the recovery packets are generated. For simplicity, we take XOR to generate the coded packet, but other systematic codes (e.g., Reed-Solomon [31]) can also be used.

The example shows one FEC packet for five data packets, i.e., $n = 5; k = 1$. The values of n and k depend on several factors, including the amount of delay that the application can tolerate, the prevailing path conditions, and nature of losses. Our experiments on PlanetLab indicate that a small amount of FEC (k = 1) is sufficient for well-connected end-hosts. For other types of paths (e.g., wireless), we may require greater amount of FEC.

The network coding decision needs to consider the number of packets as well as which packets should be coded together. The above example uses network coding across all user streams: it XORs the same subscript packets of all users to generate a coded packet. Practically, we may consider a number of factors for improved gains, such as: the number of concurrent streams, loss pattern, the capability of the receivers, and the acceptable delay for cooperative recovery.

Both the FEC and network coded packets constitute the recovery packets that are sent across to the other DC. Each recovery packet contains meta-data that identifies the data packets corresponding to the recovery packet; this information is used to undertake suitable recovery in case of a loss.

Fig. 2(b) and Fig. 2(c) show how different types of losses are recovered. A single packet loss per user stream can be recovered through FEC (Fig. 2(b)). Two or more losses require cooperative recovery. For example, if all of A's packets are lost, we will recover $(n-1)$ packets through network coding (by getting relevant packets from other receivers) and the last packet can be recovered through FEC (Fig. 2(c)).

3.3 Recovery Protocol

The repair packets are kept at the DC close to the receivers and only sent to the receiver in case of packet loss. This "on-demand" nature of recovery is similar to ARQ-style recovery (used in protocols like TCP and 802.11), but there are important differences: recovery is done by a node other than the original source, and that coded packets (instead of original data packets) are used for the recovery. This means that standard ARQ based techniques (i.e., acks, timeouts, retransmissions) have to be tweaked in ReWAN.

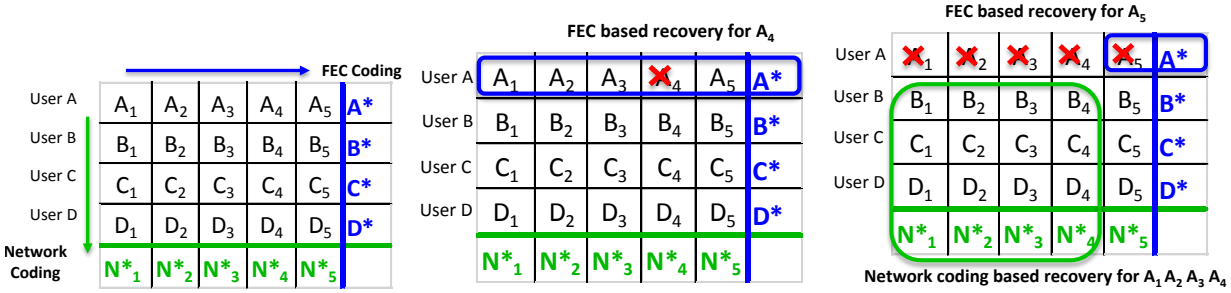
One such issue is deciding *when* to initiate the recovery. Recall that DC2 receives acks from the receivers; these acks can indicate missing data. However, acks can be lost too, so we also need a timeout mechanism. Setting a suitable timeout value is tricky in the absence of any regular data exchange between DC2 and receivers. One option is to have regular exchange of heartbeats from the receivers to the nearby DC. Another option is to set a fixed timeout value for a batch of packets, based on the maximum delay they can tolerate and subtracting from it the time required for the repair packet(s) to reach the receiver.

Finally, packet loss can occur during recovery or when the source sends the copy of the data to the nearby DC. ReWAN provides local recovery at the edges (source-DC1 and DC2-receiver) to improve the recovery efficiency. Because ReWAN operates below the transport, it uses a small number of retries before giving up and letting the end-points do an end-to-end recovery, if required.[6]

[4]Given the relatively low bit rate of real-time applications and the increasing access link capacity, we claim that sending a copy of the data to the cloud is feasible.

[5]We expect higher gains with more sophisticated coding solutions, but we leave that exploration for future work.

[6]This is in line with the end-to-end principle [25].



(a) Recovery packet generation (b) FEC based recovery for random packet loss (c) Network coding based recovery for bursty packet loss

Figure 2: Example illustrating how FEC and network coding can be used for recovery. A single FEC packet per user stream is used (see (a)) and it helps in recovering a single packet loss (see (b)). For a burst loss, cooperative recovery is used along with the network coded packets (see (c)).

3.4 Conditions under which ReWAN is useful

ReWAN can provide fast, reliable recovery, if the following conditions hold:

- **Latency inflation is low.** For timely recovery, the repair packets must arrive at DC2 before they are required (in case of a packet loss) – their delay should be lower or comparable to the propagation delay between the source and the receiver.
- **Cooperative Recovery is feasible.** For timely cooperative recovery, the propagation and transmission delays for getting the packets from other receivers should be low. We argue that receivers typically have low latency access to the DC, so propagation delay is low. Also, transmission delay is low because each receiver uploads a small fraction of the recovery traffic while the main burden is on the incoming DC link, which has high bandwidth.
- **Losses of receivers are independent.** Network coding across users is infeasible if the users experience burst losses at the same time. While losses at the edges (which are most common) are likely to be independent, a careful selection of candidate nodes for network coding (e.g., nodes in different subnets), can further decrease the likelihood of correlated losses.
- **Inter-DC losses and Internet WAN losses are independent.** If the recovery packets are lost, no recovery is possible. We argue that such cases (when both data and recovery packets get lost) are likely to be rare given the high reliability of inter-DC paths and the (likely) small overlap between Internet paths and inter-DC paths.

4. PRELIMINARY EVALUATION

We have conducted measurements on PlanetLab and Google Cloud to understand the feasibility of using ReWAN and to motivate the benefit of using FEC and network coding. We pick a random subset of PlanetLab nodes (between 4-10 nodes) in different continents as the source and destination, and use their nearby Google DC for the ReWAN recovery service. The well-connected nature of PlanetLab nodes means that our results are more representative of enterprise users rather than wireless or home users. However, some of our insights (e.g., loss correlation on wide area paths) have broader applicability.

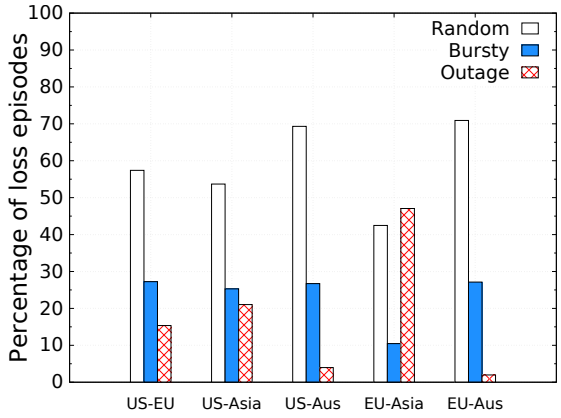


Figure 3: Breakup of loss episodes for PlanetLab paths. Three types of losses within a burst of 15 packets are shown: random (1 packet loss); burst (2-14 packet drops), and outage (all 15 packets get lost).

4.1 Loss Characterization

We first investigate the nature of losses on the wide area paths between PlanetLab nodes. Each sender sends three consecutive bursts of fifteen packets directly (i.e., using the Internet path) to the receiver. The bursts are sent after every 1 minute and this experiment runs for two weeks. Overall, we experience more than 400K loss events; each event is categorized based on the number of packets lost in a fifteen packet burst. Random refers to a single packet loss; Bursty refers to 2-14 packets being lost; and Outage refers to all 15 packets being lost.

Random losses are common. Fig. 3 shows the proportion of each of these three types in the total loss events that we observe. This accounts for the frequency of different loss events. We observe that a large fraction of loss events correspond to Random losses; these losses can be recovered through low overhead FEC (i.e., k = 1). We also observe other loss events, with outages being the least likely to

happen for most of the paths.

Outages contribute the most to loss rate. We also evaluate the contribution of each of these three types to the overall loss rate that we observe (result not shown). Even though outages happen rarely, they contribute the most to the loss rate – on some paths they contribute up to 80% of the total losses. Outages are hard to manage through FEC because of the high overhead and the likelihood that the FEC will also be lost. This motivates our decision to use network coding to recover such loss events.

Losses across PlanetLab paths are not correlated. We evaluate the correlation between losses experienced by paths across the same source and destination continents (e.g., all US-Asia paths). Recall that correlated losses limits the effectiveness of network coding (or complicates the decision of which packets to use for coding). We compute the pearson correlation coefficient and observe that loss events have no correlation – the maximum correlation coefficient value is 0.15.

WAN losses across the cloud and Internet are not correlated. Next, we evaluate whether there is any correlation between losses in the wide area for the Internet and inter-DC paths. Recall that ReWAN can recover edge losses through local retransmissions, but if the recovery packets are lost on the inter-DC path, such losses cannot benefit from retransmissions because of the high delay of wide area. To evaluate this correlation, we enhance our experiment in the following way: for every packet sent directly between the source and destination PlanetLab nodes, we also send a packet through the cloud, i.e., packet is sent to the nearby DC which forwards it across its inter-DC path to the other DC. Our results show no correlation in losses between these paths – the pearson correlation coefficient between wide area losses and inter-DC losses turns out to be low (maximum value is 0.04).[7]

4.2 How fast can ReWAN recover?

We now evaluate ReWAN's ability to recover lost packets in a timely fashion. First, we measure the latency of end-points to their nearest DC.

DCs are located close to users. For this experiment, we randomly pick PlanetLab nodes in US and Europe; for each node, we measure its round-trip time to the nearest DC belonging to any of the three major cloud providers (Amazon, Microsoft, and Google). Figure 4 shows that 80% of the nodes in Europe and 50% of the nodes in US can reach their nearest DC in less than 20ms. We believe that in future this latency (especially the tail) will further go down.

Latency inflation of cloud path is low. Second, we evaluate the latency inflation (if any) due to using the cloud path. We compare the latency of the direct path (src-dst) with the latency of sending a packet via the inter-dc path to the DC close to the receiver (i.e., DC2). This analyzes the likelihood that DC2 would be in a position to recover any lost packets. We compute the percentage difference between the latency on the direct path and the latency of using overlay path for each src-dst pair. We plot the CDF of the difference for

[7]We also analyzed the AS level paths of the wide area Internet and the inter-DC cloud paths and found little or no overlap.

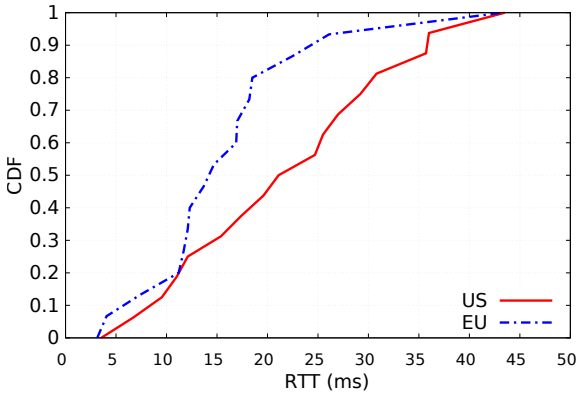


Figure 4: Latency between PlanetLab nodes and their closest DC. PlanetLab nodes are located in US and Europe. For the nearest DC, we consider three cloud providers (Amazon, Microsoft, and Google) and consider the one with the minimum RTT.

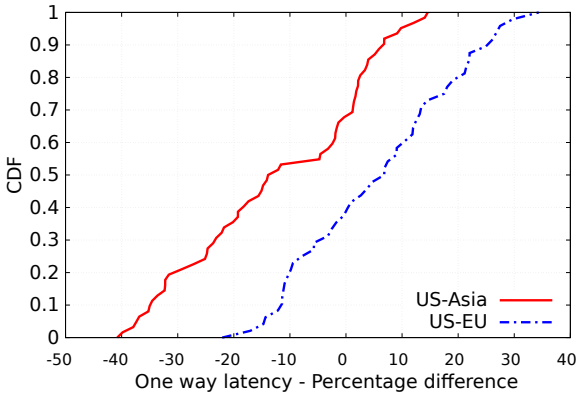


Figure 5: Percentage difference between one-way latency of direct PlanetLab path with the path from source to the DC close to the receiver (via the inter-DC path)

all pairs in the two sets of inter-continental paths – US-Asia and US-Europe. Fig. 5 shows that for many src-dst pairs, the overlay path has lower delay (indicated by a negative percentage). Even in the worst case, we observe a latency inflation of around 10% for US-Asia paths and around 30% for US-Europe paths. This substantiates prior results that show negligible overhead of using the cloud for middlebox processing [27].

Cooperative Recovery is Feasible. To highlight the feasibility of cooperative recovery, we zoom into the US-Europe paths and evaluate the time required to complete a cooperative recovery through network coding. On these paths, the one-way latency is around 60-80ms; any end-to-end recovery will take at least one extra RTT which will exceed our goal of an end-to-end delay of less than 150ms.

For our experiment, we consider a scenario where one client/receiver loses all its packets in a burst and these packets are reconstructed with the help of other receivers. We vary the number of receivers and the packets required from each receiver and evalu-

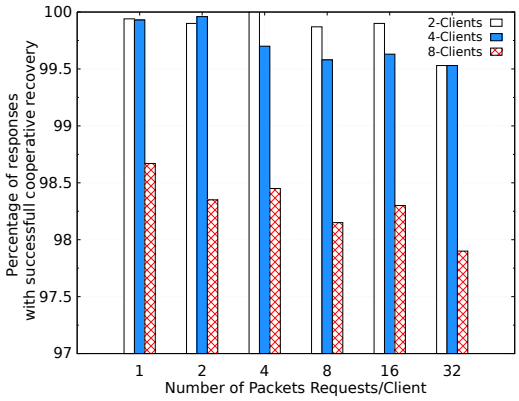


Figure 6: Feasibility of network coding based cooperative recovery. In most cases, the recovery completes within the allowed time budget of 150ms.

ate the recovery time. Note that a high fanout (i.e., need to contact many other clients) increases the likelihood of delay in completing the recovery. Similarly, greater number of packets from each client also increases the transmission delay, and hence the recovery time. We repeat our experiment 1500 times and plot the proportion of runs which were successful (i.e., completed within a budget of 150ms). Fig. 6 shows that even with a high fanout of 8 clients and 16 packets per client (the associated network coding overhead, in terms of inter-DC bandwidth, of this scenarios is only 11%), we are able to do the end-to-end recovery in more than than 95% of the cases while meeting our 150ms latency budget i.e., recovery is within 0.5RTT.

While the above result is promising, we believe that ReWAN can potentially do much better with a more intelligent network coding scheme. For our experiments, we considered a strawman network coding plan, which randomly picked and combined the packets of different receivers. Our preliminary analysis shows that taking into account the latency between receivers and the nearby DC, while deciding the network coding plan, can provide further reduction in recovery latency. We plan to explore this as part of future work.

5. RELATED WORK

Our work is inspired by *overlay* networks that improve availability by using detour points (e.g., RON [6], MONET [7], one-hop source routing [15], Spines [4], etc). Our use of cloud as an overlay creates unique opportunities and challenges. For example, we can do sub-RTT recovery, but to minimize cost, we have to send a small number of recovery packets.

Individual aspects of ReWAN's design also resonate with other overlay based solutions. For example, our use of DCs as overlay nodes has similarities with super peers in overlay networks [34]: DCs can be viewed as super peers with dedicated links between them, with cost associated with the use of these links (and other resources). Similarly, applying coding across users is similar to applying QoS across streams [30].

Our work complements the large body of recent work on *inter-data center* networking. This includes application of software defined networking (SDN) to such environments (e.g., SWAN [18], B4 [18]), as well as techniques that meet specific workload needs (e.g., application deadlines [33, 20]). Similarly, studies on inter-data center measurements [22, 33] have mainly focused on inter-data center *bandwidth*. ReWAN's use of inter-DC paths to send coded packets for packet recovery is novel and complementary to these prior efforts.

ReWAN uses several *building blocks*, including FEC [9], ARQ-based recovery (similar to TCP), and network coding [21]. Our work combines FEC and ARQ in a unique way: FEC is sent to a node (i.e., DC) close to the receiver which uses ARQ style protocol to recover packet losses. Our use of network coding across wide area user streams and sending these packets on the cloud path, along with the FEC packets, is also unique.

Finally, we share the goals of recent proposals that call for *low latency and highly reliability* for wide area communication [28, 24]. Arrow [24] is proposed as a reliable, wide area service; it uses reliable wide paths as tunnels to improve end-to-end reliability. While inter-DC paths are likely to have similar properties as Arrow's reliable paths, our overall approach of only using these paths for recovery is different and complementary to Arrow's goals.

6. DISCUSSION AND CONCLUSION

Using cloud as an overlay has its own set of advantages and cost. We call for using it in a judicious manner, such that we can leverage its benefits while incurring little cost. ReWAN is an example that illustrates how this can be potentially done. Its strawman design helps to appreciate the potential benefits and challenges associated with a packet recovery service. There are many interesting open issues in the design of ReWAN – we discuss three important questions that we plan to explore in future.

- One key issue is how to *detect* a loss. An approach like TCP's fast retransmit can be potentially used under some scenarios, but a timeout based approach is required for the general case. Setting a suitable timeout value is challenging in the absence of any regular data exchange between DC2 and receivers. One option is to have regular exchange of heartbeats from the receivers to the nearby DC. Another option is to set a fixed timeout value for a batch of packets, based on the maximum delay they can tolerate and subtracting from it the time required for the repair packet(s) to reach the receiver.
- A second important issue concerns packet loss *during* cooperative recovery. More broadly, if one of the receivers fails to respond or is a straggler, it will slow down the cooperative recovery process. To address this scenario, we could build some extra redundancy in the coding, so not everyone in the cooperative recovery phase needs to respond. Specifically, we can recover a packet if any k out of the n requested nodes respond in a timely fashion.
- A third issue is designing suitable coding techniques that can exploit the loss characteristics of different types of links (e.g., cellular) as well as adjust to different application delay requirements.

In addition to these challenges, we believe our work raises interesting questions for other network services and how they can potentially use the cloud in a judicious manner. While we discuss a few such examples (e.g., multicast and QoS) in the paper, we are far from understanding the full potential of this approach. We hope that the case we make in this paper (with the help of ReWAN) will serve as a starting point for a larger community discussion on cloud's impact on the Internet and the basic services it provides.

References

[1] Amazon AWS. http://aws.amazon.com.

[2] Microsoft Azure. http://azure.microsoft.com/.

[3] Google Cloud. https://cloud.google.com/.

[4] Y. Amir and C. Danilov. Reliable communication in overlay networks. In *Proc. IEEE DSN*, 2003.

[5] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis. 1-800-overlays: using overlay networks to improve voip quality. In *Proc. ACM NOSSDAV*, 2005.

[6] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. SOSP*, 2001.

[7] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. N. Rao. Improving web availability for clients with MONET. In *Proc. Usenix NSDI*, 2005.

[8] A. V. Bakre and B. Badrinath. Implementation and Performance Evaluation of Indirect TCP. *IEEE Transactions on Computers*, 46(3):260–278, 1997.

[9] M. Balakrishnan, T. Marian, K. Birman, H. Weatherspoon, and E. Vollset. Maelstrom: Transparent Error Correction for Lambda Networks. In *Proc. Usenix NSDI*, 2008.

[10] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth Multicast in Cooperative Environments. In *Proc. ACM SOSP*, 2003.

[11] F. R. Dogar and P. Steenkiste. Architecting for Edge Diversity: Supporting Rich Services Over an Unbundled Transport. In *Proc. ACM CoNext*, 2012.

[12] F. R. Dogar, P. Steenkiste, and K. Papagiannaki. Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices. In *Proc. ACM MobiSys*, 2010.

[13] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proc. ACM CCS*, 2002.

[14] M. Gritter and D. R. Cheriton. An Architecture for Content Routing Support in the Internet. In *Proc. USITS*, 2001.

[15] P. K. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, D. Wetherall, et al. Improving the Reliability of Internet Paths with One-hop Source Routing. In *Proc. Usenix OSDI*, 2004.

[16] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste. XIA: Efficient support for evolvable internetworking. In *Proc. 9th USENIX NSDI*, San Jose, CA, Apr. 2012.

[17] O. Haq and F. Dogar. A Measurement Study of Inter Data Center Paths. *Technical Report*, 2015. https://goo.gl/453MfE.

[18] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven WAN. In *Proc. SIGCOMM*, 2013.

[19] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, et al. B4: Experience with a globally-deployed software defined WAN. In *Proc. SIGCOMM*, 2013.

[20] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula. Calendaring for wide area networks. 2014.

[21] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the Air: Practical Wireless Network Coding. In *Proc. SIGCOMM*, 2006.

[22] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. In *Proc. ACM IMC*, 2010.

[23] A. Pathak, Y. A. Wang, C. Huang, A. Greenberg, Y. C. Hu, R. Kern, J. Li, and K. W. Ross. Measuring and evaluating TCP splitting for cloud services. In *Proc. of PAM*, 2010.

[24] S. Peter, U. Javed, Q. Zhang, D. Woos, T. Anderson, and A. Krishnamurthy. One tunnel is (often) enough. In *Proc. ACM SIGCOMM*, 2014.

[25] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 1984.

[26] Y.-H. C. Sanjay, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. In *Proc. ACM Sigmetrics*, 2002.

[27] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: network processing as a cloud service. In *Proc. SIGCOMM*, 2012.

[28] A. Singla, B. Chandrasekaran, P. Godfrey, and B. Maggs. The Internet at the Speed of Light. In *Proc. ACM HotNets*, 2014.

[29] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proc. SIGCOMM*, 2002.

[30] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. OverQoS: An Overlay Based Architecture for Enhancing Internet QoS. In *Proc. Usenix NSDI*, 2004.

[31] S. B. Wicker and V. K. Bhargava. *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[32] Y. Xu, C. Yu, J. Li, and Y. Liu. Video telephony for end-consumers: measurement study of Google+, iChat, and Skype. In *Proc. ACM IMC*, 2012.

[33] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang. Guaranteeing deadlines for inter-datacenter transfers. In *Proc. EuroSys*, 2015.

[34] B. Y. Zhao, Y. Duan, L. Huang, A. D. Joseph, and J. D. Kubiatowicz. Brocade: Landmark routing on overlay networks. In *Peer-to-Peer Systems*, pages 34–44. Springer, 2002.