All in the Family: Cost-Aware Networking for First-Party Cloud Applications

Sagar Bharadwaj Parth Thakkar Harsha Sharma Sreangsu Acharyya
Yogesh Bansal Ranjita Bhagwan Vijay Kumar Venkata N. Padmanabhan Kathleen Voelbel
Microsoft
{t-sabhar,t-pathak,t-hsharm,srach,yobansal,bhagwan,vijara,padmanab,kathleev}@microsoft.com

ABSTRACT

The growth of cloud-scale services has fueled huge growth in inter-data center (DC) Wide-Area Network (WAN) traffic. As a result, cloud providers provision large amounts of WAN bandwidth at very high costs. We observe that inter-DC traffic is largely dominated by *first-party applications*, i.e. applications that are owned and operated by the same entity as the cloud provider. This creates a unique opportunity for the applications and the network to co-operate closely, going beyond past work on traffic prioritization and scheduling, and leading to more efficient use of the WAN and consequently driving down costs. In this paper, we show, through three example scenarios, how co-operating applications and networks can achieve significant cost savings, reducing WAN capacity costs by as much as 45% in certain regions.

CCS CONCEPTS

• **Networks → Network architectures**;

1 INTRODUCTION

The growth of cloud-scale mega services [1] has fueled a huge increase in network traffic, not only within data centers (DCs) and on the Internet but, importantly, also on the inter-DC wide-area network (WAN). We observe, however, that the inter-DC WAN traffic for large public cloud providers is dominated by first-party applications and services e.g., the e-commerce service in the case of Amazon, Google search and Gmail in the case of Google, and Bing search and Office 365 in the case of Microsoft. This presents both a challenge and an opportunity.

The challenge is that, as application usage grows, so does the inter-DC WAN traffic. This leads to increased provisioning of inter-DC WAN capacity, often with a multiplicative factor to provide redundancy. The increased availability of bandwidth, and the fact that it is sunk cost (i.e., already paid for, say by lighting up new fiber, and so might as well be utilized), fuels the appetite of existing and new applications. They consume more bandwidth, causing the network to forecast higher usage, which in turn leads to increased network provisioning and so forth. Such a vicious cycle is quite expensive since inter-DC WAN bandwidth tends to be much more expensive per unit of usage than either egress bandwidth or intra-DC bandwidth.

This is quite different from the third-party application setting, wherein market forces of cost and price operating between the consumer of bandwidth (third-party application) and the provider of bandwidth (cloud provider) would tend to keep the process in check. If the provider has excess capacity, they could find new customers to sell it to rather than encouraging the existing customers to be profligate in their use of bandwidth.

While we could simply replicate in the first-party setting the market mechanisms operating in the third-party setting, we argue that this would be sub-optimal and also rather challenging, given the twin goals of fully utilizing the provisioned bandwidth (which is sunk cost), while at the same time avoiding or delaying the need for fresh provisioning. In fact, there is an opportunity to do much better in the first-party case, by leveraging *close cooperation across the network and the applications*. We present some specific examples of such informed cost optimization:

Intelligent Redundancy: using application-level information, e.g. knowing which slice of traffic is in the user's critical path and is hence latency-sensitve, to inform the provisioning of network capacity. Doing so allows the network to support applications without user impact while reducing cost significantly. We show that, for a large enterprise collaborative platform, such cooperation can reduce costs by up to 45%.

Leveraging Spatial Flexibility: using knowledge of widely varying link costs from the network to perform *traffic matrix re-engineering*, i.e., changing source-destination flows, e.g., by doing appropriate replica placement, which would provide significantly greater flexibility than the traditional approach of traffic engineering.

Cross-application Cooperation: enabling multiple first-party applications to cooperate in freeing up bandwidth and consuming excess bandwidth, without impacting the network as a whole and thereby avoiding an impact on cost. This becomes quite relevant when there is a sudden or transient shift in bandwidth (e.g., during the current pandemic) and some first-party applications (e.g., email) are more flexible in their bandwidth needs than others (e.g., audio-video conferencing).

While there has been much prior work on cooperative prioritization and scheduling of traffic flows in first-party settings [7, 8], our work is distinguished by the time scale and scope of the proposed optimizations. For example, we explore optimizing long-term capacity provisioning and re-engineering the traffic matrix itself, which are qualitatively different and go well beyond prioritizing and scheduling traffic flows.

We conclude with a discussion on some open problems and directions for future research.

2 INTELLIGENT REDUNDANCY

To tolerate failures, every DC is connected via multiple redundant paths. Provisioning for such redundancy is expensive, especially where links require submarine fiber, for example. Therefore it is

[1] We use "services" and "applications" interchangeably.

desirable to be frugal, while ensuring that the services riding on the network are not impacted. We outline how the first-party setting enables such *intelligent redundancy* provisioning.

Traditionally, network operators provision capacity based on the aggregate volume of traffic from each application. However, a first-party setting allows for a more nuanced approach. Depending on the latency tolerance of the individual application components, some of them can simply be paused when the network signals a link failure to the application and then resumed only when the link has been restored. Consequently, the network need not provision any redundancy on account of such traffic.

For this to be effective, an application needs to accurately classify different slices of traffic based on latency tolerance. Traffic from *user-facing* components are highly latency-sensitive and need to be scheduled as soon as possible. Traffic from *background* components such as indexing and replication can be delayed up to a certain *deadline* without affecting the application's performance [11, 14]. Some background components such as content-based analytics and load-balancing can be paused for an extended period of time thereby leading to deadlines of days or even weeks since, though important, these components are not essential to the core functionality of the service. If a component's deadline is significantly higher than the mean time to repair (MTTR) [2] of the links connecting these data centers, and if the application can pause the background component for the period of link failure, then the network need not provision redundant capacity for this background component since it can be paused until the failed link recovers.

Consider the following illustrative example. Data centers DC_1 and DC_2 are connected by links with an MTTR of 3 days. Let the peak bandwidth utilization of the application, user-facing and background components combined, be 1 Gbps. The network builds in a redundancy factor of 3 for a total capacity of 3 Gbps between DC_1 and DC_2.

Say user-facing components use only 0.5 Gbps, background components use 0.5 Gbps, and all background components have a deadline of 1 week, which is much longer than the link's MTTR of 3 days. In such a scenario, the network need provision only 2 Gbps: 3×0.5 Gbps sustains user-facing components even when there is failure, and an additional 0.5 Gbps sustains background components which do not need redundancy, since they can remain suspended until the link is repaired. Capacity provisioned for the data center pair DC_1-DC_2 hence decreases from 3 Gbps to 2 Gbps.

Additionally, user-facing traffic exhibits seasonality, going up during the day and falling at nights and weekends. Hence background traffic can be delayed until the specified deadline of the component and sent out to fill in the "troughs" formed by the user-facing traffic. Such smoothing of traffic further reduces required capacity since provisioning is done based on peak usage.

Translating such reduced capacity to cost savings depends upon many factors, such as whether links are owned or leased, and the specifics of the business relationship between the buyer and seller of bandwidth. However it is safe to say that a significantly lower capacity forecast will lower network costs significantly.

[2]While we use MTTR for simplicity here, our framework would carry over if we were to use a different metric such as a high percentile of the time to recovery (TTR).

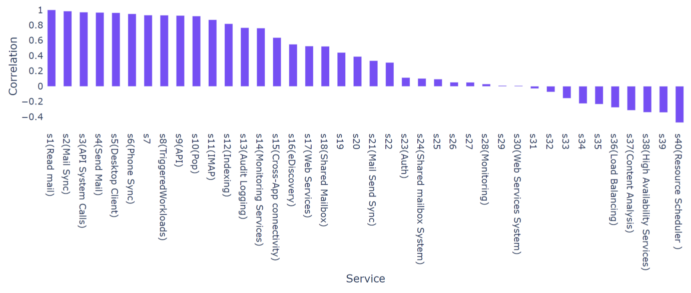

Figure 1: This figure shows the Pearson correlation coefficients that we obtain by correlating components' traffic patterns with that of a reference user-facing component. In this case, a REST API endpoint that services email clients is the reference.

2.1 Example Application

We demonstrate how this approach leads to lower capacity provisioned for Exchange Online, a large-scale enterprise email application. Exchange Online runs many hundreds of components with varying functionality. Some components are user-facing such as API end-points that support calls from email clients while others run background tasks such as replication, load-balancing, indexing, and content analytics. We formulate the capacity provisioning problem between any two DCs as a linear program with the objective of minimizing network capacity under several constraints. The following steps outline the process.

Step 1: Categorize components. Every component is tagged as "user-facing" or "background", every background component has a deadline. We believe there is the need and an interesting opportunity to automate component classification, especially when an application comprises hundreds or thousands of components, in which case manual classification may be error-prone or subjective. For instance consider the component that synchronizes an email client with server state. This component runs in the "background" and yet, insofar as it affects user-experience, it is "user-facing".

We therefore devise a novel automated technique to classify components. We identify a component that is known to be user-facing and use its traffic pattern over time as a *reference*. We correlate this reference with the traffic patterns of other components over the same time period. The higher the correlation the more "user-facing" the component is. In this instance, we use a REST API endpoint as the reference since we know that it is a user-facing component.

Figure 1 shows the Pearson correlation coefficient for a subset of Exchange Online components calculated using traffic patterns over 8 weeks. Components such as `IMAP`, `POP` and `sendmail` show high correlation while components such as `content-analytics` and `load-balancing` are anti-correlated with the reference as expected (since these are typically relegated to times when the user is inactive). For the sake of our analysis, we choose a conservative cut-off value of 0: all components with correlation above this are considered user-facing. For every background component (correlation < 0), we interview the component owner to determine the deadline. We leave automatically determining deadlines to future work.

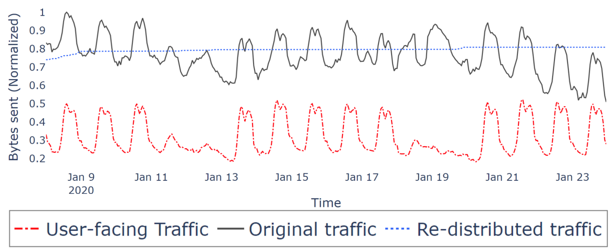

Figure 2: This plot shows how the network usage of Exchange Online varies across time. The black curve shows total network traffic, red curve shows user-facing traffic, and the blue curve shows the projected network usage had our optimization been used to plan capacity.

Step 2: Determine redundancy requirements. For each DC pair, link MTTRs determine which components need redundancy provisioning. While all user-facing components definitely need redundancy to be built in the network, a background component needs redundancy only if its deadline is less than link MTTR. For example, a failure in an undersea cable takes much longer to repair than a failure in a terrestrial link. Hence DC pairs that are connected by undersea cables will need to provision more capacity than those that use purely land-bound links.
Step 3: Formulate optimization. We now formulate a linear program that minimizes peak traffic given user-facing components and background component deadlines for a given DC pair. For each background component i and for a time interval t, we introduce a variable b_i^t that captures the network traffic that the component sends in that time interval. The total traffic S^t sent out in a time interval t is therefore the sum of these variables and the user-facing traffic sent within that time interval, u^t, i.e. $S_t = \sum_i b_i^t + u^t$. User-facing traffic often displays diurnal trends. Hence we extend this formulation to delay background traffic up to its deadline to fill in the "troughs" in user-facing traffic, which effectively smooths the traffic curve.

Capacity is usually planned as a function of the peak traffic (such as the 95th percentile [11]). Hence, the objective is to *minimize the maximum traffic sent across all time intervals.* The minimized value of peak traffic gives us the capacity that the network has to provision between these data centers. The total capacity to be provisioned is therefore the sum of this steady-state capacity and a redundancy factor required only for user-facing components.

2.2 Feasibility Analysis

In this section, we analyze how this proposed technique can reduce capacity provisioned for Exchange Online. We use 12 weeks of network and application data collected in early 2020 for this experiment. This data, aggregated at hourly intervals, provides a component-level breakup of network usage. Figure 1 shows the set of components and their correlations used in this analysis. All components with correlation <= 0 are considered background. We obtained deadlines for them by interviewing the component owners.

DC Pair	Link type	MTTR	Smoothing	Intelligent Redundancy
1	Land	3 days	18.56 %	45.26 %
2	Land	3 days	11.64 %	34.33 %
3	Sea	1 Month	5.91 %	24.89 %

Table 1: Capacity reduction achieved for three DC pairs with varying characteristics. MTTR values are illustrative only. Smoothing is achieved by the optimization setup. Intelligent Redundancy includes both smoothing and reduced redundancy for background traffic.

Using this data, we now categorize network traffic as user-facing and background. Figure 2 shows a time-series of network traffic usage for Exchange Online over a 4 week period for a DC-pair connected by a link with an MTTR of 3 days. The y-axis is normalized based on the maximum network capacity used in this period. The black solid curve shows total network usage and the red dashed curve shows user-facing traffic. The difference between these gives us the traffic generated by background components.

Next, we perform a retrospective analysis on this data which gives us how much we could have potentially saved on capacity provisioning had we known the network usage of each component in advance. This gives us the best-case savings that we can achieve using our approach since it assumes a perfect forecasting model. There are two sources to savings: a) the reduction in provisioning due to traffic smoothing and b) reduction in provisioning due to reduced redundancy for background traffic. The blue dotted curve in Figure 2 shows how the constraint-based optimization smooths the required capacity. This gives us a 15% reduction in the capacity to be provisioned, using traffic smoothing alone.

We now evaluate potential reduction in provisioning across *three* different DC pairs: two of these span a continent each and are therefore largely land-bound, while the third spans an ocean and therefore uses a submarine cable. Table 1 shows percentage capacity reduction for these three pairs. The *MTTR* column shows the representative values of mean time to repair for land-bound and under-sea links that we use in this analysis. The table shows reduction due to only smoothing, and due to smoothing combined with reduced redundancy (intelligent redundancy).

While smoothing itself gives us 5-19% gains, trimming redundancy yields huge savings of between 24% and 45% in the capacity to be provisioned. Two factors affect this magnitude. First, paths that have lower MTTRs (DC pairs 1 and 2) show higher savings because they have to provision redundancy for fewer background components. Second, the nature of the workload in a particular region affects the capacity savings. DC pair 1 experiences a burstier workload than DC pair 2. A bursty workload leads to higher peak usage, and as provisioning is done based on peak or 95th percentile usage, such DC pairs see higher provisioned capacity. As a result, we see higher gains (45%) for DC pair 1 than for DC pair 2 (34%), despite the similar link characteristics in terms of MTTR.

3 SPATIAL FLEXIBLITY

The first-party setting also provides flexibility in the spatial dimension. Unlike traditional traffic engineering, where the traffic matrix (source-destination flows) is taken as input and the goal is to route these optimally, there is the opportunity to reconfigure the traffic matrix itself, i.e., perform *traffic matrix re-engineering* by taking advantage of information cutting across the application (or service) and the network layers.

Large services typically have a significant replication component to preserve availability. For instance, every byte written might be replicated n-ways (where n is typically 2 or 3), to help ensure both durability and availability. This means that the write traffic at the backend could be multiple times that on the frontend. Even reads could trigger replicated traffic flows, for instance, to update meta data (e.g., time of last read). Besides replication traffic, there is also traffic that is triggered by load balancing, which results in a user's data (e.g., their primary mailbox replica in Exchange Online) being moved to a new location, sometimes across data centers, i.e., over the inter-DC WAN. In Exchange Online, such a move is effected by reading the relevant table(s) in the current database where the user's data resides and writing it out in a database at the new location. Such a move of the primary replica could, in turn, trigger database-level replication flows to the secondary sites. For all of these reasons, replication-related background traffic constitutes a whopping 50-60% of the traffic in Exchange Online.

While the network typically exposes a simplified pricing model to applications (e.g., uniform pricing within geographic region [3, 9]), the cost of the underlying links, and therefore that of the paths within a region, could could vary considerably depending on the local market conditions, the nature of the links (e.g., submarine or not), etc. The question we ask is whether we can exploit spatial flexibility in the application to align the volume network traffic flowing on various links and paths with the underlying network cost.

Even if the placement of replicas at each DC is constrained by factors other than network bandwidth (e.g., resources such as CPU, storage, or IOPS), we can achieve significant network cost savings by altering the *geometry* of replication. We consider two specific opportunities for network-cost-aware optimization: *replica grouping* (Section 3.1), which entails the grouping of the primary and secondary replica sites for a database, and *replication topology* (Section 3.2), which centers on the application-level paths chosen for propagating replication updates. Note that there is flexibility in picking both the replica grouping and the replication topology to minimize network cost, while leaving the usage of other, non-network resources largely unaltered.

To analyze the potential benefit of these approaches, we consider the 9 DC locations of Exchange Online in the Asia-Pacific zone, where network link costs tend to be particularly high and also diverse.

3.1 Replica Grouping

Consider 4 DCs, A, B, C, and D. For simplicity, assume 1-way replication, so that for each database, there is a primary replica and one secondary replica. Say we group the replicas as $\{A, B\}$ and $\{C, D\}$, i.e., A as the primary and B as the secondary for one database and C as the primary and D as the secondary for another. Such a replica grouping would mean that replication traffic would flow on the $A \rightarrow B$ and $C \rightarrow D$ paths. What if either or both of these paths is expensive? We could change the replica grouping to $\{A, D\}$ for one database and $\{C, B\}$ for the other, ensuring that the replication traffic then flows on the $A \rightarrow D$ and $C \rightarrow B$ paths, avoiding the expensive $A \rightarrow B$ and $C \rightarrow D$ paths. Note that the number of replicas hosted at each DC would remain unchanged (at one) despite the change in the grouping, ensuring that the usage of other, non-network resources remains (largely) undisturbed. We now generalize this.

Formally, we have M databases $\mathrm{DB}_1, \mathrm{DB}_2, \cdots, \mathrm{DB}_M$, each of which is replicated n ways. For each database, we keep the active DC fixed so as to keep the proximity, and hence the latency, to the user unchanged. If we have K DC locations in all, we can choose $n-1$ replica locations out of the remaining $K-1$ (excluding the active DC). So, for each database we must select one out of $\binom{K-1}{n-1}$ possible grouping. Further, for each database DB_i we know the peak traffic T_i flowing from the active DC to the passive DCs (i.e., the replica locations).

Once we choose a placement for a database (set of active and passive DCs), we can estimate the replication component of the network cost due to that DB as:

$$\begin{aligned}\mathrm{Cost}_{\mathrm{repl}}(\mathrm{DB}_i) &= \sum_j \mathrm{PathCost}(\mathrm{DC}^{\mathrm{Active}}, \mathrm{DC}_j^{\mathrm{Passive}}) \cdot T_i \\ &= \mathrm{GroupingCost} \times T_i\end{aligned}$$

Here, DC_j^{Passive} refers to the jth passive DC for DB_i, and $\mathrm{PathCost}(A, B)$ is a heuristic for the per-Mbps cost [3] of the paths connecting sites A and B. In our estimation, we took the PathCost as the cost of the cheapest path connecting the two sites, as a result of which our savings are an underestimate. The cost of a single path is the sum of the per-Mbps costs of inter-DC WAN links along the path.

The total network cost is the sum of individual database costs, which is expressed as:

$$\begin{aligned}\mathrm{TotalNetworkCost} &= \mathrm{TotalCost}_{\mathrm{other}} + \mathrm{TotalCost}_{\mathrm{repl}} \\ &= \mathrm{TotalCost}_{\mathrm{other}} + \sum_i \mathrm{Cost}_{\mathrm{repl}}(\mathrm{DB}_i)\end{aligned}$$

Where $\mathrm{TotalCost}_{\mathrm{other}}$ is cost due to non-replication traffic, which doesn't change in our optimization.

We encoded the $\mathrm{TotalCost}_{\mathrm{repl}}$ as a matrix product of the demands T_i (constant), the costs of every replication placement (constant) and a choice matrix (variable) which is used to determine which database gets which placement. Using an off-the-shelf optimizer such as Z3, we can minimize this matrix product, subject to the constraint that in the new allocation, for each DC, the number of databases should be within $\alpha\%$ of that of the original allocation. With $\alpha = 10\%$, we observe savings of up to 9%, whereas for $\alpha = 20\%$, we observe savings of up to 13%.

3.2 Replication Topology

Given a replica grouping, there is further opportunity for optimizing the network cost by choosing a suitable replication topology. For instance, instead of the default star topology, in which the active (i.e., primary) site directly sends replication traffic to each replica

[3]Typically, bandwidth capacity is acquired in discrete quanta, but we assume the per-Mbps simplification here.

site (which might not be cost-optimal if the direct path from the primary to a certain replica is expensive), we find a minimum cost topology in which a replica could forward the replication traffic to other replicas, i.e., perform *replication chaining*. Unlike prior work [1, 23], our use of chaining is to reduce network cost, with the benefit of explicit network cost information afforded by the first-party setting.

For example, consider a DB that is replicated across 4 sites A, B, C and D. In the network graph formed by these 4 nodes with PathCost values described in section 3.1 as edge weights, the minimum cost spanning tree will give the minimum cost topology in which replication traffic can be propagated. The replication data starts at the active DC and gets propagated along this MST to all the replica DCs. For a given DB, the total cost of replication along this topology is:

$$\text{Cost}_{\text{repl}}(\text{DB}_i) = \sum_{a \to b} \text{PathCost}(\text{DC}_a, \text{DC}_b) \times T_i$$

Where $a \to b$ denote the edges in the MST and T_i is traffic sent by the active copy. We compute the total cost by adding up cost due to replication of all DBs and cost due to other traffic. Note that the total cost depends upon the peak traffic, which is what we use for our calculations.

As per this analysis, we found that we could save 28% of overall WAN costs in the APAC region by replicating along the min-cost topology. We defer to future work a more careful analysis of the cost savings that also takes into account details such as the discrete quanta (e.g., multiples of 40 or 100 Gbps) in which network bandwidth is acquired, the lock-in period for fiber leases, etc.

4 CROSS-APPLICATION COOPERATION

In large organizations, first-party applications are often developed in silos, without higher order cooperation, and operate well independently. Yet, in constrained network situations, such first-party applications can cooperate among themselves to make efficient use of the network. In this section, we illustrate this through a cooperative effort by many first-party applications in a large organization (Microsoft) as they prepared for the drastic change in network usage due to the COVID-19 pandemic.

During the Spring of 2020, COVID-19 changed the nature of work, learning and recreation across the world as businesses and educational institutes transitioned from a predominantly in-person presence to ubiquitous virtual interaction. Consequently, certain first-party applications such as video-based collaboration platforms saw an explosive growth in usage. Networks were not planned for the unexpected surge in load. For instance in March 2020, Vodafone reported a 50% increase in Internet traffic in Europe [2].

In response to these growing demands, applications started prioritizing WAN traffic at a fine-grained level so that low-priority traffic of one application could be throttled to accommodate unexpected increases in high-priority traffic of another application. This helped avoid the need for emergency bandwidth augments, which would have been very expensive. To prioritize traffic, operators considered three factors: latency sensitivity, contribution to application functionality (the impact on the application if the traffic were squelched), and application importance. Video conferencing traffic was universally considered high priority since not only is it latency sensitive, it has become an essential service replacing important in-person interactions such as classrooms and medical consultations. On the other hand, replication traffic is much less latency sensitive and also not user-facing, and is hence low priority. Another example of low-priority traffic are APIs that provide visual elements to enrich user-experience since, while such elements are nice to have, their absence does not significantly impact application functionality.

A variety of methods implemented these priorities such as marking larger volumes of traffic with coarser DSCP markings [6], application initiated throttling and front-end load balancing. An instance where such prioritization helped was in India and South Africa after nation-wide stay-at-home orders were announced. The WAN in these countries was severely strained at this point. The fine-grained priorities helped to proactively allow higher utilization since low-priority traffic such as replication in several applications would pause if there were any failures impacting capacity. This ensured that the video-conferencing application would receive a major share of WAN bandwidth.

5 DISCUSSION

We briefly discuss some challenges and avenues for future work arising from the optimizations presented here.

5.1 Pricing Model

The myriad first-party services might each be a sizeable service in its own right, often developed or acquired independently of the others. As such, it is likely to be unwieldy and impractical for centralized coordination across the services to realize some of the optimizations outlined in this paper.

We believe that a suitable pricing model could enable loose coordination across the services. However, simply extending the peak pricing policy (e.g., P95-based), which is typically employed by ISPs, to the first-party setting would be sub-optimal. The reason is that a first-party service, which is being charged based on its peak bandwidth usage, might be tempted to fill up the valleys in the utilization curve "for free", even if it means sending unimportant traffic, since it would not add to the cost incurred by the service. However, the peak of one first-party service could coincide with the valley of another, increasing the aggregate peak across the services and driving up the overall cost.

To enable each service to manage its own bandwidth individually, while at the same time, avoiding a poor choice at the aggregate level for the entire organization, we could employ a hybrid model that combines peak- and volume-based pricing. This would ensure that each individual service is still motivated to reduce its peak and utilize the valleys, while still not going overboard in trying to fill up the latter. Furthermore, the pricing could be made dynamic, depending on the supply-demand equation, so that each service can decide on its own whether to press ahead or to back off depending on the urgency of its communication (e.g., in situations such as that discussed in Section 4).

5.2 Iterative Exploration

While we seek to insulate each first-party service from the others and to allow for each to optimize on its own, some degree of coupling might be unavoidable because of non-linearity in the cost equation. Network capacity is typically available in quanta (e.g., 100 Gbps quanta) and moreover there are fixed costs involved in acquiring or operating a fiber path regardless of how much it is utilized. So, if even one service chooses to retain an expensive link, a substantial cost would likely to incurred by the organization as a whole, even if all other services are ready to migrate away from this link.

Therefore, rather than a one-shot procedure, wherein each service specifies its requirements (e.g., traffic volume and latency tolerance) and then the network plans for capacity, we believe capacity planning should ideally be an iterative process. Based on the requirements specified by all the services and the non-linearity noted above, the network could identify opportunities for optimization and provide feedback to the services. For example, a service could be informed that if it were to relax its latency requirement a little, a substantial reduction in cost might result. Devising such an iterative procedure to achieve efficiency in both cost and convergence is an interesting challenge.

6 RELATED WORK

Network-aware application design has been explored in many domains, for instance, real-time video streaming and mobile multimedia delivery [13, 17, 20]. Such work focuses on achieving better performance despite adverse network conditions, and not on cutting across layers for overall cost savings. Moreover, previous work has looked mostly at real-time feedback from the network to adapt the application whereas our purview includes longer-term issues such as capacity provisioning and replica placement.

State-of-the-art traffic engineering techniques such as B4 [8, 10, 15] and SWAN [7] perform bandwidth allocation to different flows based on priorities and thereby make efficient use of bandwidth. Tempus [14] and Amoeba [25] target the problem of traffic engineering by allowing applications to specify different deadlines for different slices of traffic. Tempus uses a constraint solver while Amoeba uses a custom algorithm to optimally schedule flows in time, given deadlines. Pretium [11] addresses the problem of fair bandwidth sharing using dynamic pricing.

These systems address the problem of bandwidth provisioning in the near-term whereas our work targets the problem of long-term capacity provisioning through application-aware redundancy management. Moreover, our approach to intelligent redundancy depends fundamentally on the application providing explicit information of differentiated components using the deadline; previous work does not use any such inputs from the application.

Cost-aware placement, routing and scheduling is a widely researched topic [4, 5, 12, 12, 16, 18, 19, 21, 22, 24]. Prior work, however, considers simpler cost models such as uniform link-cost or link costs as exposed by the cloud to third-parties. Our proposed strategy not only benefits from the novel use of chain replication, we also use a more nuanced and fine-grained cost model for the network than previously proposed.

7 CONCLUSION

We have described how cooperation, cutting across applications and the network, can significantly lower the cost of inter-DC WANs in a first-party setting. Through three example scenarios, we show how we can provision WAN capacity more carefully, engineer the traffic-matrix for more cost-effective network usage, and maintain application performance in unexpected situations.

REFERENCES

[1] Managed chain replication in mongodb. https://docs.mongodb.com/manual/tutorial/manage-chained-replication/, 2020. [Online; accessed 8-Sep-2020].

[2] Netflix to cut streaming quality in europe for 30 days. https://www.bbc.com/news/technology-51968302, 2020. [Online; accessed 23-June-2020].

[3] M. Azure. Azure bandwidth pricing. https://azure.microsoft.com/en-us/pricing/details/bandwidth/. Accessed June 11, 2020.

[4] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive. A flexible model for resource management in virtual private networks. ACM SIGCOMM 1999, pages 95–108, New York, NY, USA.

[5] D. K. Goldenberg et al. Optimizing cost and performance for multihoming. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM 2004, pages 79–92, New York, NY, USA, 2004. Association for Computing Machinery.

[6] D. Hardt. Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers. RFC 2474, December 1998.

[7] C.-Y. Hong et al. Achieving high utilization with software-driven wan. In *SIGCOMM 2013*, pages 15–26, 2013.

[8] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev, et al. B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan. In *SIGCOMM 2018*, pages 74–87, 2018.

[9] C. M. Insider. Aws bandwidth pricing. https://www.cloudmanagementinsider.com/data-transfer-costs-everything-you-need-to-know/. Accessed June 11, 2020.

[10] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a globally-deployed software defined wan. In *SIGCOMM 2013.*

[11] V. Jalaparti, I. Bliznets, S. Kandula, B. Lucier, and I. Menache. Dynamic pricing and traffic engineering for timely inter-datacenter transfers. In *SIGCOMM 2016.*

[12] Y. Jin et al. Toward cost-efficient content placement in media cloud: Modeling and analysis. In *IEEE Transactions on Multimedia*, pages 807 – 819, 2016.

[13] Jinwei Cao et al. An overview of network-aware applications for mobile multimedia delivery. In *37th Annual Hawaii International Conference on System Sciences, 2004.*

[14] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula. Calendaring for wide area networks. In *SIGCOMM 2014.*

[15] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, et al. Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing. In *SIGCOMM 2015.*

[16] Lili Qiu et al. On the placement of web server replicas. In *IEEE INFOCOM 2001.*

[17] N. Miller et al. Collecting network status information for network-aware applications. In *IEEE INFOCOM 2000.*

[18] J. Sahoo et al. A survey on replica server placement algorithms for content delivery networks. In *IEEE Communications Surveys and Tutorials.*

[19] J. Sahoo et al. A survey on content placement algorithms for cloud-based content delivery networks. In *IEEE Access*, pages 91–114, 2017.

[20] S. Sengupta et al. Hotdash: Hotspot aware adaptive video streaming using deep reinforcement learning. In *2018 IEEE 26th International Conference on Network Protocols (ICNP).*

[21] U. Sharma et al. Kingfisher: Cost-aware elasticity in the cloud. In *2011 Proceedings IEEE INFOCOM*, pages 206–210, 2011.

[22] M. Uluyol, A. Huang, A. Goel, M. Chowdhury, and H. V. Madhyastha. Near-optimal latency versus cost tradeoffs in geo-distributed storage. In *NSDI 20*, pages 157–180, Santa Clara, CA, Feb. 2020.

[23] R. van Renesse and F. B. Schneider. Chain replication for supporting high throughput and availability. In *OSDI '04*, page 7, USA, 2004. USENIX Association.

[24] A. Verma et al. pmapper: Power and migration cost aware application placement in virtualized systems. In *Middleware 2008*, pages 243–264, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[25] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang. Guaranteeing deadlines for inter-data center transfers. *IEEE/ACM transactions on networking*, 25(1):579–595, 2016.