ABSTRACT

For rural ISPs and organizations, purchasing high-bandwidth, high-quality Internet connections is expensive, if such connections are even available. We propose MultiWAN, a system for load-balancing at packet granularity across a number of low-bandwidth, low-quality wide area network links. Doing so allows us to create an aggregated connection with performance properties similar to that of a single high-bandwidth connection, in particular that a single flow is able to use available capacity on any connection, rather than being tied to a single link. MultiWAN requires no modification to end hosts and operates as a transparent middlebox. We demonstrate that it provides performance at least as good and in some cases better than traditional flow-based load balancers, approximating that of a single high-bandwidth connection, and show that implementing MultiWAN can be cost-effective for rural ISPs and organizations.

1. INTRODUCTION

As organizations grow, they face a limited number of options for increasing the capacity of their Internet connection from their ISP. One option is simply upgrading to a higher class of service which provides higher bandwidth or usage limits, or better quality of service. This is ideal from both a management and performance perspective: the organization does not have to deal with the complexities of multihoming, and the single “fat pipe” allows users within that organization to burst to the full capacity of the link. However, this option quickly becomes more expensive as the class of service increase. Moreover, higher classes of service are not available in all areas from all ISPs; this is particularly the case for organizations operating in rural or developing regions with poor infrastructure.

The main alternative available to such organizations is purchasing additional circuits from one or more ISPs and bonding them together. Even if such links were present, purchasing multiple small links and aggregating them has the potential to be more cost-effective than purchasing one single large link. For example, India’s largest telecommunications provider, BSNL, charges roughly three times as much for a 155Mbps link than it does for five 34Mbps links [8], which provide similar aggregate bandwidth. If aggregating the smaller links proves more economical than purchasing a single high-bandwidth link, rural ISPs and organizations would benefit from a substantial reduction in costs.

When upgrading a single connection to handle more traffic is not an option, a common solution is to use a load balancer to distribute flows at the transport level across multiple connections. Choosing the flow as the unit of load balancing is a logical decision due to the predominance of TCP traffic on the Internet: splitting a single TCP flow across multiple network paths causes packet re-ordering, greatly reducing the flow’s performance [1, 7]. While this approach is both simple to implement and to manage, it limits the burst rate of individual flows to that of the connection on which they are placed. This is particularly unfortunate for typical web-heavy traffic mixes, where the bulk of flows are short and can complete more quickly if able to burst to high bandwidths.

In this paper, we propose MultiWAN, a system for load-balancing traffic at the packet level, allowing full utilization of multiple WAN connections without significantly impacting TCP performance. We achieve this by creating a logical tunnel for each WAN connection from the organization using MultiWAN to another location under their control with cheap, plentiful bandwidth, such as a cloud provider’s data center or educational institution. Because MultiWAN runs at both ends of these tunnels, we are able to apply more sophisticated load balancing techniques than previous approaches. We use in-band available bandwidth measurements to dynamically redistribute load across connections and to control the degree of reordering caused by splitting a single flow across multiple network paths. By doing so, we enable a single flow to achieve a throughput near the aggregate throughput of the available WAN connections, significantly improving utilization of network resources. MultiWAN also decreases the average flow completion time, improving a key metric for user-perceived performance of web traffic.

In addition to improving the web-usage experience for users in the developing world, MultiWAN is useful for improving performance for mobile Internet access, as one might find on a bus or train. These systems typically load-balance users connected via 802.11 Wifi across multiple Internet connections to the cell phone network. As a result, bandwidth is both very limited and highly erratic.

The remainder of this paper is structured as follows. We first describe related work on load balancing across multiple networks and the active measurement techniques we use. In section 3, we explain our design goals and section 4 describes the design of MultiWAN. We then evaluate the performance of MultiWAN and compare it against both a flow-based load balancer as well as a single “fat pipe”. We also provide a comparison of costs for each before concluding.

2. RELATED WORK

Most of the individual components of MultiWAN are not novel – we proudly borrow solutions to particular sub-problems we face from other work, in particular our flowlet-level traffic splitting scheme to avoid re-ordering [5] and our latency-based congestion detection [7]. However, we are the first to combine these pieces into a real-time, dynamically adaptive system and to optimize our system for the unique needs of organizations and ISPs in the developing world.
Devices that perform flow-based load balancing are available commercially, and this feature is supported in modern operating systems such as Linux. These systems operate by binding transport layer flows to particular upstream connections, but do not allow any single flow to take advantage of excess capacity on other connections. Link aggregation is another commonly used load balancing technique, which like MultiWAN uses packets (frames) as the unit of load balancing. However, link aggregation operates at the link layer and neither works across a WAN nor provides any accommodation for preventing re-ordering.

There have been a variety of proposals for taking advantage of the aggregate bandwidth multiple independent paths to improve application performance, such as MPTCP [13] and SCTP [11]. However, these protocols are meant to be run on end systems as replacements for standard single-path transport protocols. MPTCP in particular maintains a congestion window for each path that a sub-flow uses, and determines allocation across paths based on the relative growth rate of each path’s congestion window. MultiWAN operates as a middlebox, requires no modification or special configuration of end hosts, and because it operates at the network layer supports arbitrary transport protocols.

Several recent papers have considered the problem of dynamic load balancing, such as COPE [12], TeXCP [4], and MATE [2]. These have primarily considered the case of load balancing traffic within an ISP’s backbone network. In particular, these assume some degree of cooperation from routers across all paths utilized, which is infeasible for the deployment scenarios MultiWAN is designed for. While MATE is specifically designed for MultiProtocol Label Switching (MPLS) networks, it uses method similar to our own (described in 4.2) for detecting congestion based on relative latency differences between probe arrivals.

We adopt the reordering-reducing traffic splitting mechanism used by FLARE [5]. While the original FLARE design assumed the desired traffic distribution was provided by an oracle and attempted to match a periodically-updated “desired” distribution of traffic, our approach uses in-band measurements to dynamically adapt to changing network conditions in real-time. The novelty of our approach is this integrated system, as well as its deployment model.

3. DESIGN GOALS

Small Internet Service Providers (ISP) operating in rural communities, especially in developing countries, must rely on scarce, expensive, and poor quality upstream Internet bandwidth. As providing reliable service is a priority for a service provider, these operators prefer to buy bandwidth from multiple upstream ISPs to increase their availability. High bandwidth Internet connections are often not available in rural areas; in the rare locations where they are, customers must often cope with extremely long lead times to establish connectivity and generally unaffordable prices. As a result, purchasing multiple residential-targeted connections and aggregating them together via some form of load balancing becomes an attractive alternative, promising higher reliability and bandwidth at reasonable costs.

A secondary effect of the poor connectivity situation in rural developing regions is that network operators must struggle to achieve profitability by using an arsenal of techniques aimed at increasing their oversubscription ratio while attempting to minimize the negative impact such measures may have on user satisfaction. Common techniques used to make more efficient use of limited upstream bandwidth include aggressive caching, traffic prioritization, and filtering or rate-limiting of resource-intensive applications such as high-definition video and peer-to-peer file transfer.

As bandwidth demand increases, operators may experiment with more drastic measures such as selective compression of incoming traffic. Such a proxy, ideally located in a well-connected data center, compresses incoming traffic and routes the scaled-down content back to the rural community. Our core insight is that once an operator begins using a tunneling proxy to improve their network performance, the operator can apply much more sophisticated load balancing techniques to aggregate their multiple WAN connections together, in addition to other desirable traffic engineering mechanisms.

We aim to decrease costs for a network operator in the developing world by increasing the degree of oversubscription of their upstream Internet connections without affecting user-perceived performance. When we consider how to build a system to achieve this, several key design goals come to mind.

**Emulate a single high-bandwidth link as closely as possible.** The ideal Internet connection is both high-capacity and reliable. In particular, a single large connection allows traffic to burst to the full capacity of the link, leading to efficient utilization of available resources. This allows user flows to complete more quickly, increasing user-perceived performance and delaying the onset of congestion in the link. Unfortunately, such connections are not available in much of the developing world, and even when they are they are not cost effective.

**Perform no worse than a flow-based load balancer.** The current state-of-the-art for aggregating multiple Internet connections is flow-based load balancing, in which traffic is load balanced at the transport layer such that a single flow is bound to one particular connection. Network appliances that implement this scheme are commercially available and widely deployed, with well-understood and accepted performance characteristics. While a flow-based load balancer is not the most efficient approach to aggregating multiple Internet connections, deploying one is a low-risk proposition for a network operator. As a result, any alternative load balancing technique, particularly one which violates conventional wisdom by splitting a single flow across multiple paths, must perform no worse than a flow-based load balancer. Otherwise, few network providers would be willing to deploy it.

**Improve the performance of web traffic.** Numerous analyses have shown that web traffic comprises the bulk of the traffic on the Internet [7]. Users are highly sensitive to web performance.

While other applications such as peer-to-peer file sharing and VOIP are popular and important, a load balancing system should not optimize for their performance. In the former case, the volume of bandwidth generated by peer-to-peer applications presents significant challenges to network operators; indeed a system that reduced the performance of these applications would be considered beneficial by many network operators.

This goal also imposes important and challenging constraints. Web traffic uses TCP as the transport layer, which provides both end-to-end congestion control and reliability for each flow. In order to provide good performance, any load balancing technique must not create conditions which cause TCP to suffer poor performance. An ideal load balancing system would allow TCP flows to reach their steady-state fair traffic allocation as quickly as possible, while not unnecessarily causing a flow to reduce its sending rate. In particular, this means that the load balancer should keep a flow in the aggressive-growth “slow-start” phase as long as possible and should keep packet re-ordering to a minimum.

We specifically target a usage profile that consists primarily of asymmetric web traffic, with the network connected to the local
MultiWAN tunnels technology. The MultiWAN instances establish a number of logical instances at the IP level; in particular, we do not require that the physical network connections may (and in fact should) be independent at the IP level; in particular, we do not require that the number of physical connections available at each endpoint be equal. While we used a simple IP-in-IP tunnel.

Once a flow has left the slow-start phase and is no longer aggressively growing its window size, the primary determinant of throughput is the available bandwidth of the connection it is using. Thus, we should minimize the congestion on all WAN connections in the tunnel. This will both keep flows in slow-start for as long as possible and optimize the utilization of available bandwidth in the tunnel.

Intelligently route traffic across available paths. Since most of our traffic is TCP and TCP connections often complete while still in slow-start we need to intelligently choose what paths these flows take. Keeping flows in slow-start on connections with low latency will increase their window size much faster than those on high latency lines, just by the way TCP uses the RTT to clock itself and grow its window.

In addition not all traffic is best suited to travel over certain available paths. The path’s end-point may be no where near the intended destination. This could be “domestic traffic” intended for local destinations. A possible solution to this is using geo-located IP’s to classify traffic as domestic or international and send them down particular paths accordingly.

4. THE MULTIWAN SYSTEM
4.1 Overview

The MultiWAN system consists of two tunnel endpoints, each of which runs the congestion-proportional load balancer described in section 4.2. The local endpoint is physically located within the organization which is bonding together multiple independent WAN connections. The remote endpoint is physically located in a facility with a high-bandwidth Internet connection, such as an educational institution or data center. Ideally, the remote endpoint would be located at a site with low latency to the providers of the content most requested by users at the organization running MultiWAN. For this reason, we envision remote endpoints being run on public cloud infrastructure.

Each endpoint runs a symmetrical instance of MultiWAN, and each has one or more physical network connections as well. The physical network connections may (and in fact should) be independent at the IP level; in particular, we do not require that the connections share the same routing view, ISP, or even link layer technology. The MultiWAN instances establish a number of logical MultiWAN tunnels between each endpoint equal to the number of independent physical network connections available at the local endpoint. We do not require any particular tunneling strategy, nor do we require that the number of physical connections available at each endpoint be equal. While we used a simple IP-in-IP tunneling scheme in our evaluations, MultiWAN could run equally well over tunnels created with standard tools such as OpenVPN; doing so would allow MultiWAN users to enjoy encryption and compression of their data.

While designing our system we put ourselves under the restriction of non-synchronized clocks. Since one endpoint is likely to be in a developing region where even power is unreliable, the idea of keeping the clocks on either side of the tunnel synchronized seemed beyond our reach. However we do assume they progress at the same rate.

To fully fill our goal of emulating a single high-bandwidth link as closely as possible and improve the performance of web traffic we aim to maximize our WAN connection utilization and maximize TCP performance. We maximized our connection utilization through the traffic allocation across the different connections, explained in section 4.2. To perform no worse than flow-based load balancing we implemented FLARE [5] along with in-band latency measurements in section 4.3. Finally the goal to intelligently choosing route traffic across available paths led us to the idea of grouping our connections by latency, explained in 4.4.

4.2 Traffic Allocation

Our goal of maximizing our WAN connection utilization can be defined as minimizing the congestion on each connection. Notice we are not trying to eliminate the congestion. We do not have control of the packet rate nor do we want to throttle it ourselves. We wanted to do the least amount of disturbance to the traffic. So decided to let the network itself do the throttling, while we simply try to keep the congestion on any given connection as low as possible.

It is also important to note we do not have omniscience when it comes to what is going on in the tunnel. What happens to the packets between our two endpoints is unknown, the only information we have are the measurements we can manage at the endpoints. This means we also cannot predict the rate of the packets arriving to travel across our tunnel. However since the connections are relatively small in comparison to what most web servers today expect a single user to have, we assume that the packet rate will usually be enough to fill our tunnel and engineer for this common case.

4.2.1 Congestion Measurement

A common practice to measure available bandwidth is to send packet trains with a specific elapsed time between each packet and then measure the difference between the packets arrivals [7]. If the gap between the packets has grown the path is considered congested because the latter packet spent more time waiting in a router’s queue than the former packet due to more packets in the network.

Instead of creating packet trains we use the actual traffic we are sending down each path. Since we cannot have synchronized clocks we insert our own header, referred to in the rest of the paper as the MultiWAN header, with a timestamp between the encapsulating IP header and the packet. We use the timestamp in the MultiWAN header to calculate the inter-send time for the packets in each WAN connection in the tunnel. The inter-send time gap is compared to the inter-arrival time gap. If the inter-arrival time gap is greater than the inter-send time gap we consider the packets to have detected congestion, which we will call a congestion event.

However this is a single measurement and could be just a temporary phenomena. On top of that we are detecting congestion towards that tunnel endpoint, however the tunnel endpoint that can utilize the congestion information is the other end. To solve both of these issues we include another field in the MultiWAN header, a bitmap about the last 16 packet inter-send/arrival gaps. If the bit is 1
the inter-arrival gap is greater than the inter-send gap, 0 otherwise. This bitmap is placed on all outgoing packets, so the endpoint receiving the packets has the most up-to-date information on the state of the WAN connections leading from it to the other endpoint. A sequence number is included in the MultiWAN header to know the freshness of the bitmap. It is incremented for each new measurement, so the endpoints know how many of the measurements are new and have not yet been reacted to.

### 4.2.2 Distribution Algorithm

To reiterate our goal is to minimize the congestion across the connections. A general idea of the algorithm is to use the bitmap information for each WAN connection to move traffic from more congested connections to less congested ones.

First we convert the bitmap to what we termed a congestion score. A congestion score is the number of 1’s in the congestion bitmap, which gives us the number congestion events since our last update or the last 16 packet intervals. The average of the congestion scores is calculated and all scores above the average will have a percentage of their traffic moved to a connection below the average.

There are two possible ways to choose which below average WAN connection receives the moved traffic. The first possibility is there is a connection(s) with a congestion score of zero. If such a connection(s) exists it receives a percentage of traffic from each of the above average congestion WAN connections. The second, and more likely, possibility is all the connections have a congestion score greater than zero. In this case, for each of the connections with an above average congestion score a below average congested connection is uniformly chosen based on a distribution of the inverse of their scores. Or in other words the lower the congestion score the more likely that connection is chosen to receive the moved percentage of traffic.

The update interval, percentage of traffic moved per interval, and minimum traffic allocation per line are important variables in our implementation. The update interval and traffic percentage control how quickly we react to congestion. We chose 100ms for the update interval due to the bitmap size and RTT of the tunnel. Since the bitmap is only 16 bits (small for the sake of keeping the header as small as possible) the interval should not be so big that we lose information, nor so small that the bitmap has barely any new congestion measurements in it. The highest bit rate we tested was 1Mbps, so for 1500 byte packets we have 16 packets every 400ms. However the tunnel’s RTT in our experiments is 300ms. An interval of 400ms means we could potentially be reacting to congestion that the endpoints have already noticed and reacted to. After experimenting in our test bed we found that 100ms, gave us the trade-offs we wanted.

Since we do not want the traffic distribution to not change very quickly nor vary drastically over time, a small update interval means we should have a small percentage change, while a large interval should accordingly have a larger percentage. With the update interval set at 100ms we set the percent of traffic we move per interval to 1%.

Finally both our traffic measurements are in-band and our needed information exchange is sent in-band, we needed all the connections to have a minimum of traffic. After testing we set this number to 5%.

After testing we found the distribution would periodically become completely off balance with some connections set to the minimum allocation. Deciding this was caused by a lack of exploring for potentially available bandwidth, we added a mechanism that would move the distribution towards uniform. Whenever the congestion scores for all the connections is equal, we move 1% of traffic from the connection with the biggest allocation to the smallest allocation. This was found to help stabilize the distributions.

### 4.3 Reorder Handling

TCP is very sensitive to reordering and reacts to it as if there is a loss and therefore congestion on the network. However since we are sending packets down connections that have potentially different latencies, we are introducing the potential of reordering that has nothing to do with loss. To prevent possible reordering we implemented FLARE [5], however it required periodic measurements to find the maximum RTT delta to use as its minimum time before switching-ability (MTBS). We already send the timestamp in our own MultiWAN header to detect congestion, we realized that we can also use this information to determine the max one-way-latency delta, which is a more accurate value to use for deciding what line a packet can travel.

To calculate the max one-way-latency delta we use the timestamp in the MultiWAN header and the timestamp of when packets arrive. Using these timestamps we can get the latency between two lines, line A and B by doing the following. Let line A’s send time of a packet be \( t_A \) and arrival time be \( t_A' \), and line B’s send time of the same packet be \( t_B \) and arrival time be \( t_B' \). The max one-way latency is calculated as:

\[
\text{Max One-Way Latency} = \max((t_A' - t_A), (t_B' - t_B))
\]
B’s send and arrival times $t_B$ and $t'_B$ respectively. Even though the clocks on each tunnel endpoint are not synchronized we can assume they progress at the same rate, therefore the difference between the send and arrival times is simply the latency plus the difference in the endpoint’s clocks. So the latency of line A and B would be $l_A = t'_A - t_A + c$ and $l_B = t'_B - t_B + c$ respectively, where $c$ is the difference in the two clocks. To get the latency delta between the two lines one just needs to subtract them which is $D_{AB} = l_A - l_B = (t'_A - t_A) - (t'_B - t_B)$. To find the max latency delta we simply calculate the difference for all the lines and find the longest time difference.

### 4.4 Latency Control

Developing regions utilize a diverse set of Internet connections, including DSL, satellite, and broadband. Therefore there is a high possibility the tunnel will include both a low latency connection and a high latency connection. However these two kinds of connections have very different latencies, which can cause a high variance in the RTT if a flow’s packets are sent down both connections. A high variance can cause problems in detecting lost packets and more importantly window size growth.

To handle this we divide the WAN connections into classes based on latency. All flows start traveling on the class with the lowest latency. When that latency class becomes too congested the oldest flows are moved to the next higher latency class. We move the oldest flows because they are the most likely to have finished their slow-start phase. Therefore their aggressive window growth has already passed and increasing their RTT will not be as strong a performance hit as those that are still in slow-start. Also most TCP flows are relatively short, the older the flow is the more likely it is a long lived flow and therefore getting it finished quickly is not as important as the short flows that are still in slow-start.

We defined a class’s congestion score to be the exponentially weighted moving average of the average of its connections’ congestion scores. When the class’s congestion score is above a threshold of 3 we move 1 flow from the current class to next higher class. The weight of the move average is set to 0.5, this way we react to congestion that is currently happening, but with an equal emphasis on historical congestion as well so we are not reacting to something that is only momentary. However we noticed that with the weight moving average after reacting to congestion we continue to react to it because of the history. To remedy this whenever we move flows to another class we also reset the moving average to zero.

### 5. Evaluation

#### 5.1 Methodology

We evaluated the performance of three scenarios that a rural network operator could face: using MultiWAN, a flow-based load balancer, and a single “fat pipe” connection. We implemented both MultiWAN and a simple flow-based load balancer in Click [6], and used Dummynet [10] to simulate network latency and bandwidth constraints across simulated WAN connections between the routers. We simulate multiple WAN connections between the routers by differentiating packets’ source and destination IP addresses using NAT in Click, or in the case of MultiWAN through the outer header’s source and destination IP address from our IP-in-IP encapsulation scheme.

Our experimental testbed is arranged in a dumbbell topology, and is shown in Figure 2. Both routers are identically-configured servers with two 2.6GHz Opteron CPUs and a dual-port Broadcom BCM5704 1Gbps NIC. Each router is connected via a switch to its respective traffic generator and is directly connected to the monitor, a quad-core 2.4GHz Opteron server with its two 1Gbps NICs operating bridged together. All traffic between the routers is captured by a monitor server, allowing us to capture and analyze our experimental traffic as it appears on the wire. We run two traffic generator tools on our end systems. The first of these is iperf, which we use to analyze how our MultiWAN affects the performance of a small number of flows. The second tool is tmix [3]. The tmix traffic generator replays the application level behavior of a trace, which allows us to generate highly realistic, trace-derived traffic for our experiments. Thus, we are able to evaluate MultiWAN per-

---

1Our flow-based load balancer randomly assigns new flows to a particular WAN connection based on a hash of the flow’s source and destination IP address and port.
performance for a typical web-heavy traffic mix we would expect to see in a rural network operator’s network.

Our input to the traffic generators was derived from a network trace we obtained from a rural Internet service provider in India taken at the border gateway between their network and their upstream provider. This trace was taken from a point after their caching proxy, so it only includes requests that were not handled by the cache and that would incur a WAN access under normal circumstances. The input trace was four hours long, and was captured in February 2011. We consider only the first 30 minutes of the trace for our experiments. For the input to our traffic generators, we further consider only connections that complete within the first 30 minutes, allowing us to focus on only relatively short connections.

In 4, we see the throughput over time of a replay using our sample of the original trace. The difference in offered load between our replay trace and the original input trace is accounted for by limiting ourselves to only short, completed connections.

5.2 Throughput
We first compare the maximum achieved throughput of a single TCP flow under our three experimental conditions. We simulated two WAN connections at the local endpoint, each constrained to 2Mbps, with an RTT on both of 300 milliseconds, a typical latency for intercontinental Internet traffic. We modeled this scenario with a single flow generated by iperf from the local-side traffic generator to the remote-side traffic generator. Each line was given a bandwidth constraint of 2Mbps, and we emulated an RTT of 300ms per line. The results of our experiment are shown in ???. As expected, traffic is effectively split between the two lines, and our aggregate performance exceeds the per-line bandwidth constraint of the experiment.

5.3 Improving Flow Completion Times
While MultiWAN can enhance the performance of a small number of long lived flows, the real metric that matters to users is how quickly their web pages load. Thus, we next consider how MultiWAN affects the flow completion time (FCT) of typical web traffic. We consider two scenarios, and compare the performance of MultiWAN to the flow-based load balancer and the single “fat pipe”.

Constrained. We first constrain our two links to 256Kbps and 256Kbps, well below the offered load of our replay trace. Our results are in figure 6.

Unconstrained. Next, we set the bandwidth constraint of each line to 1Mbps. This ensures the replay does not suffer any constraint.

6. DISCUSSION

6.1 Limitations
While the results of our evaluation indicate that MultiWAN achieves its design goals, we do not consider MultiWAN to be a philosopher’s stone for Internet connections, transmuting a number of poor-quality connections into a high-quality one. Indeed, the aggregate connection that MultiWAN provides is only as good as its constituent underlying physical links. For instance, aggregating multiple VSAT connections with MultiWAN may increase performance by allowing flows to use bandwidth on all connections rather than just one, but latency can see no improvement. MultiWAN specifically only addresses network-layer issues such as congestion and routing outages, and in general cannot compensate for poorly-performing underlying infrastructure\footnote{It can and does, however, compensate for unreliable underlying infrastructure by dynamically re-allocating traffic when a link becomes unusable.}. We do not see MultiWAN as a replacement for high-quality network connections, but rather as a system for extracting the most performance out of a limited set of existing resources.
6.2 Cost-Effectiveness

Our key goal for MultiWAN is to allow rural network operators in developing regions to operate more efficiently and at lower per-user cost. Thus, any performance advantages MultiWAN offers are useless if it cannot be deployed cost-effectively. Indeed, one key disadvantage of MultiWAN is that it requires high-quality bandwidth at the remote endpoint: thus, a user of MultiWAN must pay not only for their expensive local Internet connections, but also the remote endpoint’s Internet connection as well. Costs of deployment are highly dependent on the business model used by the network operator and the entity that deploys the MultiWAN system, which may not be the same. We believe that MultiWAN is cost-effective under certain reasonable business models.

Exploiting the fact that MultiWAN has very low CPU and memory demands, sharing a single remote endpoint among several local endpoints is feasible. The operator of the remote endpoint – either a separate business entity or a consortium of network operators – would deploy the remote MultiWAN endpoint in a colocation facility, taking advantage of the relatively low-cost bandwidth available there. Such a colocation facility uses wholesale bandwidth rather than retail, which is found to be orders of magnitude more expensive than wholesale.

7. CONCLUSIONS

Our proposed system, MultiWAN, achieves the key design objectives of an ideal load balancer for network operators in rural developing regions. Not only does MultiWAN outperform a traditional flow-based load balancer, it roughly approximates the performance characteristics of a single high-bandwidth Internet connection. At the same time, it still preserves the availability benefits of having multiple independent upstream Internet providers. We have shown with trace-derived simulations that MultiWAN can improve flow completion times for a typical web-heavy traffic mix of a rural ISP, thereby improving user-perceived performance and increasing user satisfaction. While MultiWAN is not a panacea for the challenging connectivity situation faced by rural network operators, it is a practical and cost-effective system in many scenarios. The system is a drop-in middlebox that operates transparently (and without any modifications) to both end hosts and other middleboxes the network operator may be using to achieve their traffic engineering goals.

8. REFERENCES