DiversiFi: Robust Multi-Link Interactive Streaming

Rajat Kateja†, Nimantha Baranasuriya‡, Vishnu Navda†, Venkata N. Padmanabhan†
†Microsoft Research India ‡National University of Singapore

ABSTRACT

Real-time, interactive streaming for applications such as audio-video conferencing (e.g., Skype) and cloud-based gaming depends critically on the network providing low latency, jitter, and packet loss, much more so than on-demand streaming (e.g., YouTube) does. However, WiFi networks pose a challenge; our analysis of data from a large VoIP provider and from our own measurements shows that the WiFi access link is a significant cause of poor streaming experience.

To improve streaming quality over WiFi, we present DiversiFi, which takes advantage of the diversity of WiFi links available in the vicinity, even when the individual links are poor. Leveraging such cross-link spatial and channel diversity outperforms both traditional link selection and the temporal diversity arising from retransmissions on the same link. It also provides significant gains over and above the PHY-layer spatial diversity provided by MIMO. Our experimental evaluation shows that, for a client with two NICs, enabling replication across two WiFi links helps cut down the poor call rate (PCR) for VoIP by 2.24x.

Finally, we present the design and implementation of DiversiFi, which enables it to operate with single-NIC clients, and with either minimally modified APs or unmodified APs augmented with a middlebox. Over 61 runs, where the baseline average PCR is 4.9%, DiversiFi running with a single NIC, switching between two links, helps cut the PCR down to 0%, while duplicating wastefully only 0.62% of the packets and impacting competing TCP throughput by only 2.5%. Thus, DiversiFi provides the benefit of multi-link diversity for real-time interactive streaming in a manner that is deployable and imposes little overhead, thereby ensuring coexistence with other applications.

CoNEXT '15, December 01-04, 2015, Heidelberg, Germany
 ISBN 978-1-4503-3412-9/15/12…$15.00
DOI: http://dx.doi.org/10.1145/2716281.2836120

CCS Concepts

•**Networks** → **Network layer protocols; Wireless local area networks;**

Keywords

Real-time Streaming, Wi-Fi, Multi-path, VoIP

1. INTRODUCTION

WiFi networks have been patterned after Ethernet that preceded them. Just as a host plugs into an Ethernet port, a WiFi client "plugs into" a single access point in its vicinity through the process of association, and remains associated with it until the link is broken, say due to mobility.

We argue that this approach, inspired by the wired world, is suboptimal and needs to be revisited. At any point, the client typically has a choice of multiple WiFi links — not merely WiFi access points within range but in fact ones that the client has the credentials to connect to. This is especially so in settings such as enterprises, schools, airports, hotels, malls, etc. where there is often a single entity that has deployed a large number of access points.

The quality of a wireless link varies frequently, causing sporadic bursts of packet loss, which can severely impact real-time streaming due to its sensitivity to packet loss, delay, and jitter. Therefore, rather than performing *selection*, i.e., picking one "best" link to connect over, we argue that the client would be better off *hedging*, i.e., connecting over multiple links simultaneously. If the client's traffic were then replicated across these links, the client would then enjoy the performance offered by the best link, even as which link is the best changes frequently.

However, replicating traffic over multiple links would mean a substantial overhead. To address this challenge, we present DiversiFi, which leverages *network-side buffering*, whether at existing WiFi APs or at a separate middlebox, to provide much of the benefit of diversity through replication but with little of its overhead. *This combination of benefit without the overhead is a key novel contribution.*

Specifically, to make the case for DiversiFi, we focus on *real-time, interactive streaming* (e.g., voice-over-IP (VoIP)) [1],

[1]As noted in Section 5, we focus here on the downlink.

which places stringent demands on the network in terms of end-to-end latency, jitter, and packet loss rate, much more so than on-demand streaming (e.g., YouTube), which has the luxury of a large playout delay. For instance, the round-trip latency must be under 300 ms for VoIP [6] and under 100 ms for interactive games [25], for otherwise the user would notice a lag. Also, the packet loss rate for VoIP should ideally be under 1% [34], although burst losses can cause noticeable artifacts even at lower loss rates.

We start by presenting experimental data to show that the WiFi last hop is a significant factor in poor streaming quality. Our analysis draws on both a year's data from a large VoIP provider that serves hundreds of millions of users, and also our own experiments with the participation of 274 users across 22 countries and involving 9224 simulated VoIP calls over a 2-month period. From our experiments, we find that the overall poor call rate (PCR) is over 10%. Further, for both the VoIP provider data set and ours, the *relative* difference in PCR between WiFi and wired clients is 40-50%.

Thereafter, we focus specifically on the WiFi last hop and make the case for leveraging link diversity to improve real-time streaming performance. We argue that clients often have more than one AP within range that they could connect to. Furthermore, the performance of the links to these multiple APs is only weakly correlated, say in terms of the pattern of packet loss. Thus, replicating real-time traffic over multiple links — we focus on doing so over two links, a *primary* and a *secondary* — yields significant benefits. We term this approach *cross-link replication.* We show that the diversity benefit of such cross-link replication dominates both fine-grained link selection and also temporal replication (i.e., retransmission with an offset) over a single link. It also provides significant benefits over and above PHY-layer spatial diversity in MIMO systems such as 802.11ac. Our experiments with a client equipped with two WiFi NICs show that cross-link diversity helps cut down the poor call rate (PCR) for VoIP by 2.24x relative to single-link transmission.

Having established the benefits of cross-link replication, we turn to how this could be realized in practice, *without* requiring two NICs, incurring the overhead of duplicated traffic or, importantly, *impacting other, non-realtime applications*. Our general approach involves confining cross-link replication to real-time flows, with non-real-time traffic continuing to flow over a single, default link, as it would have in the absence of DiversiFi. Further, the replicated packets are held in network-side buffers, close to the client but yet not being transmitted over the air unless the need arises (i.e., a packet is lost on the primary link). We show how a simple change to make the AP's queuing policy *head-drop* instead of the usual tail-drop, coupled with a shortening of the queue length, enables an efficient realization of DiversiFi. We also present an alternate approach, which leaves the APs unmodified and moves the buffering into a network middlebox. The introduction of the middlebox also helps avoid the overhead of duplicating the stream over the WAN.

Our experimental results confirm that DiversiFi provides the benefit of replication for real-time flows (e.g., a reduction in PCR from 4.9% down to 0% over 61 runs), with minimal overhead (only 0.62% of the packets being duplicated wastefully compared to nearly 100% with naive duplication). Importantly, a competing TCP flow, which is oblivious to DiversiFi being used for the real-time flows, only suffers a throughput degradation of 2.5%.

Thus, DiversiFi imposes a minimal duplication overhead and ensures coexistence with competing, non-real-time applications, which remain oblivious to the use of DiversiFi for improving the performance of real-time flows.

In summary, our contributions in this paper include:

- A study based on data from a large VoIP provider and our own experiments, to show that WiFi has a significant bearing on real-time streaming performance.
- An empirical analysis to show the significant benefits of cross-link replication.
- An architecture and implementation to gain the benefits of cross-link diversity without its overhead.

2. RELATED WORK

Network support for real-time flows has received a lot of attention over the years, and has even led to the creation of standards such as DiffServ [17] and 802.11e [3], which provide prioritized service to packets based on the type of service (ToS) bits carried in the header. The ToS bits, however, are typically not preserved across administrative boundaries on an end-to-end path. More importantly, in the context of our focus here, *such prioritization is aimed at congestion and is of little use in the face of wireless packet loss.*

Having clients connect to and maintain multiple links simultaneously has been explored in several pieces of prior work. Some of this work has considered links spanning heterogeneous technologies such as WiFi and cellular [16, 20] while other work has focused on links over a homogeneous technology such as WiFi (e.g., [18] enables multiple WiFi associations by virtualizing a single WiFi NIC in software). One motivation for combining links is to enhance performance through bandwidth aggregation, especially in the backhaul [14, 24]. A different motivation is to enable seamless handoffs through a make-before-break strategy for managing connections to multiple APs [19], with multi-path TCP [30] providing the glue. *In contrast, our goal is to leverage the diversity of multiple WiFi links to improve reliability, hence our focus on real-time streaming rather than TCP flows.*

Improving reliability through link diversity has also been considered in prior work. Multi-radio diversity (MRD) [27] uses multiple radios to simultaneously receive the same transmission and then combines the individual copies, which may each be errored, to try and reconstruct the transmitted packet. Leveraging such *receiver diversity* would require the radios

to expose the raw decoded bits to the upper layers, which is a departure from the currently-deployed WiFi hardware and firmware. Complementary work such as Source-Sync [29] on *sender diversity* involves synchronized transmissions from multiple APs to a receiver to improve reliability and throughput. *Again, this would be hard to accomplish with the existing WiFi hardware.*

[36] also exploits the diversity of multiple links, either using multiple WiFi NICs or using a single one but switching it between links, but without requiring any changes to the exiting WiFi hardware. This work focuses on the uplink, and employs a static switching strategy that cycles through the available links in a round-robin fashion and furthermore employs coding to recover from (non-bursty) packet loss. Other work [35] has assumed knowledge of the statistical models of the packet loss for each link and has accordingly derived the optimal switching strategy.

In contrast to the above, DiversiFi is a software-only solution that does not depend on any new hardware capability, does not perform proactive switching, and also addresses the downlink direction. DiversiFi actually observes packet loss on a link and then switches to a different link *reactively*.

There has also been work on multipath streaming over WAN paths [21, 15, 37], with a coded or duplicated stream of packets being spread across multiple paths. *Coding and duplication imposes an overhead over the source stream and furthermore it is assumed that the receiver would be able to receive the streams sent on the multiple paths concurrently.* Our focus on the WiFi last hop is complementary to this body of work. Furthermore, *DiversiFi takes advantage of the diversity provided by multiple paths, without incurring the overhead of switching between links or duplicated transmission, unless there is actually a packet loss.*

Finally, fine-grained link *selection*, wherein a client switches rapidly from one link/channel to another, has also been studied [28, 22]. Our work derives from this work in that we also advocate switching between links/channels rapidly. However, *our focus is on diversity rather than selection*, in that the client derives the benefit of multiple links. Diversity means that, for example, if the client does not receive a packet over one link, it can still receive the missed packet over another link (by taking advantage of network-side buffering), thus achieving significantly better loss recovery (as described in Section 4.1). In contrast, switching links based on a link selection strategy could only possibly help in the delivery of future packets.

3. WIFI STREAMING PROBLEMS

We often hear anecdotes of users facing poor VoIP call quality when connected over a WiFi link. In this section, we seek to quantify the problem and a potential solution, by answering the following questions: (a) in real-world deployments, do WiFi-connected clients encounter worse performance than wired clients?, and (b) are there opportunities to exploit diversity by connecting to other APs in the vicinity?

Table 1: Change in PCR relative to the baseline. '+' denotes a better (lower) PCR while '-' denotes a worse (higher) PCR.

	Subset	EE	EW	WW
1	All	+27.7%	+1.6%	-18.4%
2	/24s with #E≥#W	+31.9%	+6.3%	-11.9%
3	PC	+34.2%	+12.9%	-5.4%
4	/24s with #E≥#W	+36.6%	+15.1%	-3.1%

3.1 Analysis of a Large VoIP Service

We begin by analyzing a year's worth of data from a large VoIP service, which serves hundreds of millions of users and carries over a billion talktime minutes each day. The service uses a proprietary peer-to-peer protocol for calls, and supports a suite of audio codecs, including the SILK codec with FEC support. Our analysis focuses solely on the question of whether the WiFi link is a significant contributor to poor call quality. We do not consider any other aspect of the service.

At the end of each call in the VoIP service, the user is invited at random to rate their call experience on a 5-point scale. If the user actually chooses to respond, the rating provided is recorded. We define the two lowest ratings on the 5-point scale as "poor" and accordingly calculate the poor call rate (PCR) as the fraction of calls that are rated as poor. For the reasons noted above and also because there may be a bias in when users choose to provide a rating (e.g., they may be more likely to do so when they have a poor call experience), we do not focus on the PCR itself but instead examine how factors such as WiFi connectivity impact the PCR.

As the baseline, we first compute PCR_{all} based on all user-rated calls during 2014. We then compute the relative difference between PCR_{all} and PCR_X, for a subset, X, of the calls, as $PCR^{\Delta}_{all,X} = \frac{PCR_{all}-PCR_X}{PCR_{all}} * 100\%$. For example, if $PCR_{all} = 10\%$, and $PCR_X = 8\%$ and $PCR_Y = 15\%$ for subsets X and Y, respectively, then we would compute $PCR^{\Delta}_{all,X} = \frac{10-8}{10} * 100 = +20\%$ and $PCR^{\Delta}_{all,Y} = \frac{10-15}{10} * 100 = -50\%$. In other words, the PCR for subset X is 20% better than the baseline whereas that for Y is 50% worse.

Given our interest in analyzing the impact of the WiFi link on the PCR, we separate out the calls based on the type of last-hop connectivity of the communicating peers, and focus on the two dominant types: WiFi and Ethernet. We then compute the PCR when both peers are on Ethernet (labeled "EE"), both are on WiFi ("WW"), and one is on Ethernet and the other on WiFi ("EW"). As shown in row #1 of Table 1, relative to the baseline, $PCR^{\Delta}_{all,EE} = +27.7\%$ and $PCR^{\Delta}_{all,WW} = -18.4\%$, while $PCR^{\Delta}_{all,EW} = +1.6\%$ lies in the middle. In other words, having Ethernet clients at both ends yields the best (lowest) PCR, WiFi clients at both ends the worst (highest) PCR, and Ethernet on one end and WiFi on the other end an intermediate PCR.

Although the significantly higher PCR for WiFi-connected peers compared to Ethernet-connected ones points to WiFi

being a contributing factor, there are a couple of concerns with drawing this conclusion.

The first concern is that since clients on WiFi tend to be mobile, they would likely connect from a much more diverse set of locations than those on Ethernet. For instance, while the latter might be largely confined to well-connected locations such as enterprises and homes, the former would include more challenging enviornments such as airports and malls, where the backhaul connection itself might be constrained. To mitigate any consequent bias, we only consider pairs of /24 subnets (corresponding to the two communicating peers) for which there are at least as many data points (i.e., user-rated calls) for EE as there are for WW. Thus, subnets that primarily or overwhelmingly host only WiFi clients are excluded, thereby avoiding the concern noted above. As shown in row #2 of Table 1, when only such (presumably better-connected) subnets are considered, the PCR improves across the board (i.e., for both Ethernet- and WiFi-connected clients) relative to the full set reported in row #1. However, there is still a significant difference between the PCR for Ethernet-connected clients and WiFi-connected ones.

A second concern is that many WiFi clients would be inexpensive, low-end smartphones and tablets that might suffer from deficiencies in hardware (e.g., under-powered CPUs, low-quality microphones and speakers) that could negatively impact user-perceived call quality. Such poor calls would then be mistakenly attributed to WiFi as the underlying cause. To mitigate this concern, we consider the subset of PC-class devices, which includes a mix of Ethernet and WiFi connected desktops and laptops. As shown in rows #3 and #4 of Table 1, the PCR for the PC-class devices is better (lower) across the board than that for the overall population, more so when we consider just the (presumably better-connected) subnets noted above. However, there is still a significant difference in PCR, of about 40% relative to the baseline, between Ethernet- and WiFi-connected clients.

We conclude that the WiFi link is a significant contributor to poor call quality, so it is worthwhile exploring how WiFi can be made more reliable, which is our focus in this paper.

3.2 Distributed Testbed Measurements

Our analysis of data from large scale VoIP provider in the previous section did not allow us to control the calling pattern or report the *absolute* PCR. To overcome these limitations, we conducted a distributed measurement study, wherein we recruited 274 users[2] across 22 countries to install on their WiFi-connected Windows PC/laptop a simple measurement tool called NetTest that we developed. In addition, we install NetTest on 10 well-connected machines distributed across Microsoft Azure's datacenters worldwide. NetTest ran VoIP-like streams (64 Kbps, 20 ms interpacket spacing, 2 min duration) between various pairs of the participating clients. We orchestrated the pattern of these calls, for instance, to have a particular WiFi-connected client connect, in turn, to another WiFi connected client or to a well-connected node on Azure, with these connections happening either directly or through a relay in the cloud. This pattern of connections is designed to mimic typical connectivity patterns in a VoIP service.

[2]We obtained approval from our institution for this study.

Table 2: Poor Call Rates for different call categories.

Call Type	Total Calls	Poor Call Rate (%)
EW	6953	5.22
WW	1240	7.98
EW-Relayed	798	42.11
WW-Relayed	233	62.66
Total	9224	10.23

Based on our dataset of 9224 simulated calls, we analyze the voice quality of these calls by running the packet traces through a G711 codec, and using the degree of interpolation and extrapolation of voice samples to estimate Poor Call Rate (PCR), in accordance with well established models [10, 11]. We found the overall PCR, in *absolute* terms, to be 10.23%. Table 2 shows a breakdown of PCR across different call categories. Consistent with our above analysis on data from a large VoIP service, we find that calls from a (WiFi-connected) NetTest node to another WiFi-connected NetTest node have a higher PCR than those to a well-connected Azure node (7.98% vs. 5.22%, which corresponds to a 50% *relative* difference). In terms of spatial distribution, we find that while 57.9% of the users experience at least one poor call, 16.3% suffered a poor call rate of 20% or higher, implying that the problem of poor streaming quality is widespread and, in fact, quite severe for a sizable fraction of users.

Note that the much higher PCR for the relayed calls was an artifact of the overloading of the relay nodes. Nevertheless, the negative impact of WiFi connectivity vis-a-vis Ethernet connectivity manifests itself.

3.3 Availability of Multiple WiFi Links

Finally, we look at availability of multiple WiFi APs that a user could potentially connect to. We went to various enterprise and public locations, such as offices, campuses, serviced apartments, hotels, malls, etc., across 3 cities — Bengaluru, Seattle, and Singapore — in different countries and determined how many BSSIDs were within range on the networks that our client could connect to. Figure 1 presents the results. We see that the number of BSSIDs available was 6 at the median, with at least 2 across all locations and up to 13 in some locations. Even on an in-flight WiFi network the client had a choice of 6 BSSIDs! Even if we only count the number of distinct channels corresponding to the BSSIDs seen (marked by the bold dashes in Figure 1), to discount the possibility of virtual APs, the median count is 4, with the range spanning 2 to 9. We conclude, therefore, that there is amble scope for DiversiFi to operate in such locations.

The situation, however, is less rosy in residential loca-

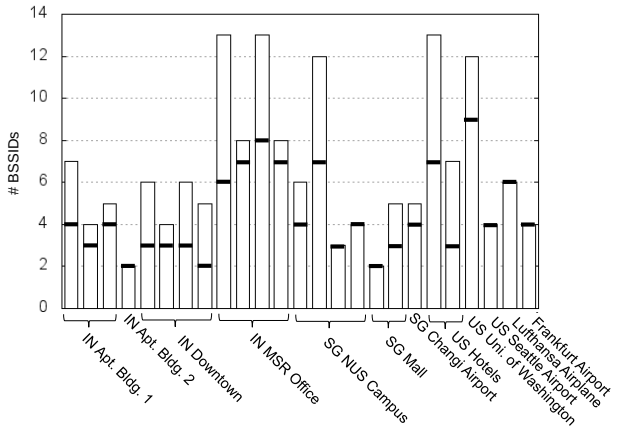


Figure 1: The number of BSSIDs (bars) and distinct channels (dashes) the client could connect to at various locations.

tions. Indeed, in our NetTest dataset discussed above, which is skewed towards residential locations, we found that in only 30% of the cases did the client device have more than one BSSID that it could connect to. Nevertheless, we note that even in such locations, the incipient trend towards multi-band APs would make multiple WiFi links available to clients.

Although the availability of multiple WiFi links does not by itself indicate the performance of these links, the results presented in Section 4.1 show that even a weak secondary link could be very beneficial.

4. LEVERAGING LINK DIVERSITY

To make the case for leveraging link diversity through replication in DiversiFi, we consider several questions:

1. How does *replication* for diversity compare with fine-grained link *selection* proposed previously (e.g., [28])?
2. How does the diversity arising from *cross-link replication* (in particular, the duplication of a stream concurrently over two links) compare with that from *temporal replication* over a single link?
3. Does *cross-link replication* provide benefits over PHY-layer spatial diversity in *MIMO*?

Our analysis in this section focuses solely on the benefits of diversity through replication. *Later, in Section 5, we explain how this benefit can be had without the overhead of duplicating traffic or requiring clients to have two NICs.*

Unless stated otherwise, the data presented in the subsections that follow is based on experiments conducted with G.711-like UDP data streams, with a data rate of 64 Kbps, 160-byte packets, and a 20ms inter-packet spacing. We used a Lenovo W520 ThinkPad laptop as the client, with an Intel Centrino Ultimate-N 6300 AGN internal WiFi NIC and a TP-Link TL-WDN3200 dual-band ABGN USB adapter. Each experiment comprised a 2-minute simulated call during which a copy of the G.711-like stream was sent to each NIC of the client. With this setup, we gathered data for 458 such simulated calls, at a variety of locations, including offices, serviced apartments, downtown areas, and a conference setting in Bengaluru and Singapore. The data set includes traces corresponding to various challenging situation such as a weak link, client mobility, external interference from a microwave oven, and network congestion.

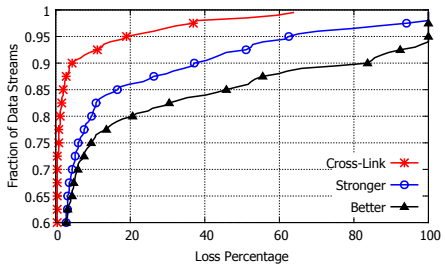
In terms of the metrics, we consider both network-level ones such as the packet loss rate and the one-way delay jitter, and also a voice call metric, the poor call rate (PCR). For the network-level metrics, we divide the simulated call into 5-second periods and focus on the worst such period, for there is evidence that worst degradation in a (short) call is a significant determinant of the overall user-perceived call quality [38]. For voice call metrics, we run the network trace through the G.711 codec, and use the degree of interpolation and extrapolation of voice samples to estimate PCR, in accordance with well-established models [10, 11]. Note that the trace captures all network-related impairments that a stream may suffer, including packet delay, jitter, and loss.

4.1 Cross-Link Replication vs. Link Selection

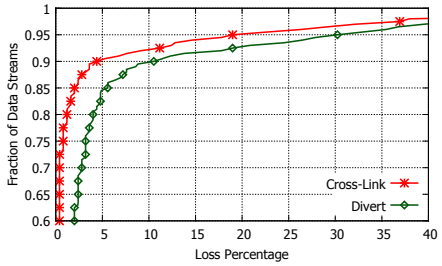
In the presence of multiple WiFi APs in the vicinity, *selection*, i.e., picking the one "best" link out of many choices, is what OSes typically do today. The question is how *replication* compares with this simple strategy of selection. To evaluate this question, we have the two client NICs connect to the two strongest APs, and then implement two selection strategies: `stronger`, which picks the stronger of the two links based on the RSSI (as is typically done in OSes), and `better`, which samples both links for the first 5 seconds and settles on the one that performs better during this trial period for the rest of the call. We compare these strategies to `cross-link`, which replicates the stream on both links.

Figure 2a shows the CDF of the packet loss rate during the worst 5-second period for the three strategies noted above. The figure shows that `cross-link` dominates both selection strategies, especially in the tail. For example, while the 90^{th}%tile packet loss rate is 37% and 84%, respectively, for the `stronger` and `better` selection strategies, it is only 4.4% for the `cross-link` replication strategies.

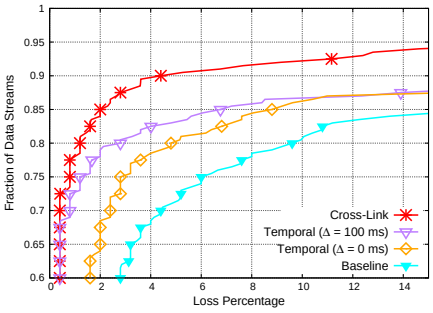
There has also been work on *fine-grained* link selection, wherein a fresh selection is made and the link switched, frequently. Divert [28] is an example of such a strategy. We evaluate this strategy, `Divert`, where a link switch is triggered if the number of frames lost exceeds a threshold T within a window H frames. We use a setting of $H = 1$ and $T = 1$, as in [28]. Figure 2b shows the CDF of packet loss rate for the worst 5-second period. We see that `Divert` performs worse than `cross-link`, with the 90^{th}%tile packet loss rate being 10.5% for `Divert` compared to 4.4% for `cross-link`. The key observation is that while the switching of links in `Divert` is triggered by one or more packet losses, the switching is only in the hope of improved likeli-



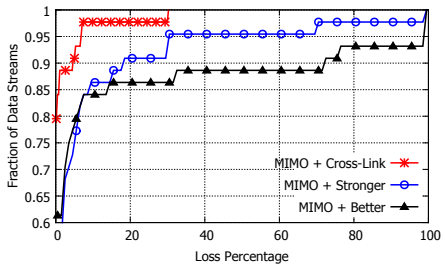
(a)



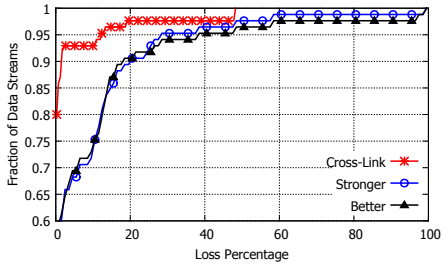
(b)



(c)



(d)



(e)

Figure 2: CDF of loss rate over worst 5-second period, comparing `cross-link` replication to (a) link selection based on `stronger` and `better`, (b) fine-grained link selection (`Divert`), (c) `temporal` replication, (d) 802.11ac with `MIMO`, and (e) for high-rate 5 Mbps streams.

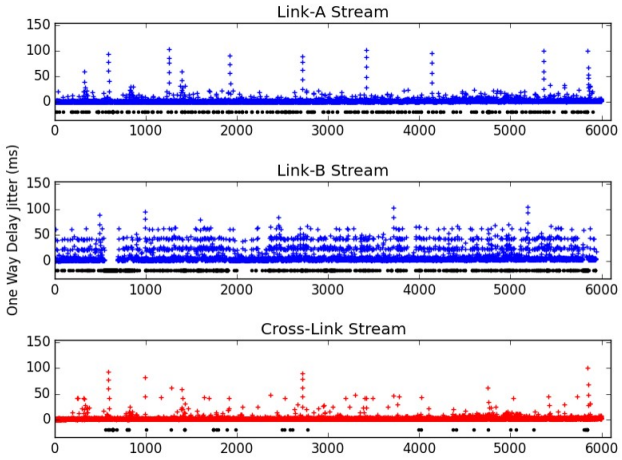


Figure 3: An example trace showing how cross-link replication over a weak link A (loss rate: 4.3%) and an even weaker link B (loss rate: 15.4%) yields a much lower loss rate of 0.88% (losses are shown as black dots along the bottom of each plot) and also a lower delay jitter (plotted on the y axis).

hood of reception for *future* packets. The packets that were lost before the switch are themselves not recovered. On the other hand, in `cross-link`, the receiver has the benefit of reception on *both* links, so packets that are lost on one link will likely be received on the other.

As a final illustration of how cross-link replication is qualitatively different from link selection, we show, in Figure 3, the case of two weak links, link A with an overall packet loss rate (i.e., for the entire 2-minute simulated call, not just the worst 5-second period) of 4.3% and link B with an overall packet loss rate of 15.4%. With traditional link selection, link A would clearly dominate B, so the client would be better off just sticking to A. However, with cross-link replication across A and B, the overall packet loss rate drops to 0.88%. In other words, *the better of the two links (link A) benefits from replication over a significantly worse link (link B), showing the benefit of diversity.*

4.2 Cross-Link vs. Temporal Replication

If replication is the key to improving performance, a natural question is why do *cross-link* replication. Would temporal replication, i.e., sending multiple copies of each packet spaced over time, on the same link offer the same benefits? Note that any such replication would be over and above link-layer retransmissions that already happen in the 802.11 MAC layer. However, unlike the latter, which would tend to happen on a very fine timescale (tens of μs), the temporal replication (`temporal`) would happen over a much longer timescale. The benefit of a larger temporal spacing is the greater diversity in the channel conditions across the multiple transmissions. However, the temporal spacing would be constrained by the deadline for the real-time stream.

We consider `temporal` with two copies of each packet

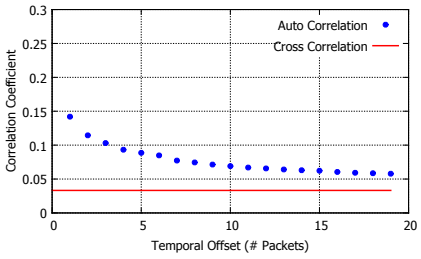


Figure 4: Auto-correlation vs. cross-correlation of the packet loss process within a link and across two links, respectively.

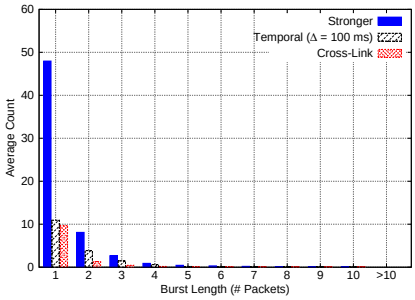


Figure 5: Distribution of packet loss burst lengths for `stronger` link selection, `temporal` replication, and `cross-link` replication.

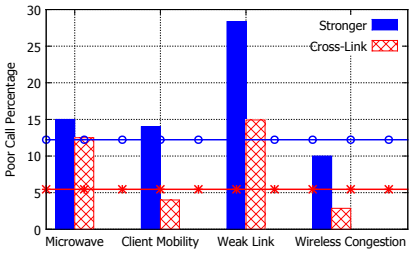


Figure 6: The poor call rate (PCR) with `stronger` link selection and `cross-link` replication, broken down by various impairments.

being transmitted, with a spacing Δ between them of up to 100ms. (Given the 300ms limit the end-to-end round-trip for VoIP [6], 100ms one-way on just the WiFi hop would seem to be a reasonable limit.) Figure 2c shows the CDF of the packet loss rate for the worst 5-second period for `temporal`, with Δ ranging from 0ms (i.e., two copies of each packet being transmitted back-to-back) to 100ms (i.e., the copies being spaced apart by 100ms). We find that temporal replication does improve the packet loss rate compared to the baseline (i.e., no replication), with the improvement being greater the larger the spacing Δ. For instance, at the 90^{th}%tile, `temporal` with $\Delta = 100$ms has a packet loss rate of 23.7% compared to 37.2% for the baseline. However, `temporal` still underperforms `cross-link`, which has a 90^{th}%tile packet loss rate of 4.4%.

Thus, it is clear that `cross-link`, with its spatial and channel diversity (i.e., links to APs at different locations and on different channels, possibly even different bands), dominates `temporal`. The latter suffers from the bursty nature of impairments on a link. We illustrate this in two ways. First, we compute the auto-correlation of the packet loss time series on a link and compare that with the cross-correlation of the corresponding series across two links. Figure 4 plots the results. We see that even with a lag of 20 packets (or 400ms, given the 20ms inter-packet spacing), the auto-correlation is greater than the cross-correlation. Second, since packet loss bursts are particularly problematic for real-time streams such as VoIP, we compare `cross-link` and `temporal` in this regard. Figure 5 plots the distribution of packet loss burst lengths for the baseline (which is `stronger`, without any replication), `temporal`, and `cross-link`. We see that not only does `cross-link` have a lower packet loss rate than `temporal`, its losses also tend to be less bursty. For instance, out of the 6000 packets in a 2-minute simulated call, 25.6 on average were lost in `cross-link`, out of which only 15.9 were in bursts of 2 or more consecutive packets. In comparison, an average of 61.9 packets were lost in `temporal`, with a much larger proportion, 51.0, occuring in bursts. [3]

4.3 Benefits over and above 802.11ac with MIMO

With the advent of MIMO, WiFi provides spatial diversity at the PHY layer. For example, 802.11n supports up to 4 spatial streams whereas 802.11ac supports up to 8. The question is whether cross-link replication (`cross-link`) provides any *additional* benefit when there already is PHY-layer diversity through MIMO, the gains due to which have been studied previously [33]. Since we have no control over the APs in our experiments in the wild, we conducted experiments in the lab with 6-antenna D-Link DIR-850L dual-band 802.11ac APs and 2-antenna Linksys WUSB6300 dual-band 802.11ac NICs on the client. We gathered data for 44 simulated calls and, as before, plot the CDF of the packet loss rate for the worst 5-second period in Figure 2d. We see that `cross-link` has a lower loss rate than `MIMO` alone.

Thus, `cross-link` provides benefits over and above `MIMO`. Why is this? The PHY-layer diversity in `MIMO` arises because the separation, on the order of the carrier wavelength, of multiple transmit (or receive) antennas, results in the multipath fading of the spatial streams being largely independent [26]. However, there are other link impairments that such PHY-layer diversity does not address. For example, with shadowing and external interference, all co-channel spatial streams could be affected simultaneously.

4.4 Improvement in VoIP Quality

So far we have focused on packet-level metrics such as the loss rate. However, the question remains as to how cross-link replication would impact VoIP quality. Figure 6 plots the estimated poor call rate (PCR) for the set of simulated calls. We see that `cross-link` helps cut down the PCR by 2.24x relative to `stronger`, from 12.23% to 5.45%. While the relative improvement is even greater (3.5x) in the cases of client mobility and wireless congestion, it is much

[3]These loss statistics are for the entire 2-min simulated calls while much of the preceding analysis was for the worst 5-sec period.

less (1.2x) in the case of microwave interference. This is an artifact of our experimental setting wherein a majority of the links available in the vicinity of the microwave were impacted by the interference (there were no 5 GHz links available). In general, greater diversity could be had from cross-technology replication (e.g., across WiFi and 3G/4G), but keeping the duplication overhead manageable would be more challenging and we defer investigation of this alternative to future work.

4.5 Benefit for High Bandwidth Streams

We also experimented with real-time streams of a much higher data rate than VoIP, as might be typical of video or gaming. We performed 80 runs, each lasting 2 minutes, with a 5 Mbps stream, comprising 1000-byte packets, with a 1.6 ms inter-packet spacing. Figure 2e plots the CDF of the packet loss rate for the worst 5-second period. Again, we see a significant benefit from `cross-link` replication, with the 90^{th}%tile packet loss rate improving to 1.7% compared to 20.5% for link selection based on `stronger`. This finding, coupled with the design we present in Section 5 to avoid the overhead of unnecessary duplication of packets on the air, means that `cross-link` replication can be applied to high-rate streams as well as to the low-rate ones.

4.6 Moving from Analysis to Design

The analysis presented above has made the case for cross-link replication. But how do we realize this in practice? In our experiments thus far, we have used a client with two WiFi NICs to receive a copy of the stream on each. This is a challenge for deployment since few clients would have two WiFi NICs. Also, cross-link replication done naively would mean the duplication of all packets, including the many that are received successfully on both links. This is undesirable not only because of the overhead on the recipient but also because of the negative impact it could have on other clients that share the medium, especially if the duplicated stream is a high-bandwidth one (e.g., a video or game stream).

In the next section (Section 5), we present our design and implementation of DiversiFi to achieve virtually all of the benefits of full cross-link replication but with little of its overhead.

5. DESIGN AND IMPLEMENTATION

We now present our design of DiversiFi, which centers on cross-link stream replication for diversity, but with network-side buffering and implicit or explicit packet selection, to limit the actual duplication of packets in the air to just (or close to) what is necessary. While we focus here only on streaming in the *downlink direction*, which is arguably the more challenging direction because of the lack of control over the APs, we believe our design would apply equally in the uplink direction and would likely be easier to implement because the client would have direct control over what packets are sent over which link and when.

5.1 Design Requirements

For reasons of *deployability*, in view of the large installed based of WiFi, we seek a solution that does not require any changes to WiFi at the hardware layer.

For reasons of *generality*, we would like our solution to be transparent to the application, be it audio/video conferencing [2, 7], cloud-based gaming [5, 8], or app streaming [4]. We would not like to depend on any application support other than characterizing the real-time stream in terms of the rate, deadlines, etc. (which the application would do anyway when using protocols such as RTP [32, 31]).

For reasons of *coexistence*, any steps taken to improve the quality of real-time streaming for a client should not be to the detriment of other traffic flows, including non-real-time ones, at that client or other clients.

5.2 Design Elements

5.2.1 Initialization

When an application initiates a real-time stream, we need to know the stream rate, packet size, and the packet deadlines, so that the network stack, whether at the local host or the remote peer, can take the appropriate actions in respect of replication, buffering, and selection, as noted below. Since real-time streaming applications are typically based on the Real-Time Protocol (RTP) [32], we can use the *payload type* field to look up the corresponding *profile* [31], without the need for modifying applications.

5.2.2 Multi-Link Association

To enable a client with a single WiFi NIC to associate with multiple APs, we use past work on multi-link association (e.g., [18]). In a nutshell, the client creates multiple virtual adapters, each with a different MAC address, and a separate AP association. It then switches between the links, changing channels, if necessary, and using the 802.11 power save mode (PSM) to keep its association on a link alive even when it has switched away to a different link.

In DiversiFi, the client creates separate virtual adapters and links for the real-time stream. For instance, in the setup shown in Figure 7, the client has two virtual adapters, labeled as *primary* and *secondary*, for the real-time stream, each associated with a different AP. The primary associates with AP chosen based on the standard OS policy (e.g., strongest AP) and the secondary with next-best AP. In addition, it has a default virtual adapter, DEF, used for all of the other (non-real-time) traffic (not shown in the figure).

At any point in time, the primary real-time virtual adapter would be associated with the same AP and on the same channel as DEF. So switching between the primary adapter used for a real-time flow and DEF used for a non-real-time flow would *not* incur any overhead.

However, switching between links on different channels, such as the primary and secondary links for a real-time flow, can take a non-negligable amount of time, e.g., 2.3ms in

our measurements reported in Section 6. So minimizing the frequency of switching between the primary and secondary links would be desirable, to minimize a negative impact on other flows on DEF, in line with our goal of coexistence.

5.2.3 Stream Replication

Cross-link stream replication lies at the heart of DiversiFi and the benefits reported in Section 4. The question is where and how the packet stream is replicated.

One option is to replicate at the source, i.e., the remote peer. However, this would not only require modification of the source, it would also entail the overhead of duplicating traffic over the entire end-to-end WAN path.

An alternative would be to replicate the stream close to the receiver, for instance, at an SDN-capable switch on an enterprise LAN. While this would avoid the overhead of duplication on the WAN, it would require the presence of an SDN-capable switch and also the ability for the receiver to configure the switch by installing suitable match-action rules [12], say using APIs such as [23].

5.2.4 Network-Side Buffering

While cross-link stream replication helps improve the quality of a real-time stream, it does so at the cost of wastefully duplicating on the secondary link the overwhelming majority of packets that are delivered successfully on the primary link itself. Packet buffering, along with packet selection discussed next in Section 5.2.5, is key to mitigating this overhead. The replicated packets are held in a buffer in the network, close to the receiving client (to help minimize the additional latency, should these need to be delivered), yet not delivered over the air (on the secondary link) unless needed.

Network-side buffering could be performed at the APs themselves, taking advantage of the power save mode (PSM), whereby the AP would start buffering packets destined to the client when the client sends a sleep message (Null frame with the Power Management bit set) and would deliver the buffered packets when the client sends a wakeup message (Null frame with the Power Management bit cleared). However, as discussed in Section 5.3.1, this would require slight modifications to the AP to ensure efficiency.

An alternative, which would leave the AP unmodified, is to perform buffering at a separate middlebox, as discussed in Section 5.3.2.

5.2.5 Packet Selection

While buffering close to the client allows the possibility of packets being delivered with a low (additional) latency, we need a way for the client to identify the packets of interest and have these delivered selectively. The method used for selection would depend on where buffering happens. If it is in the AP as part of the standard PSM processing, the selection is done implicitly, as discussed in Section 5.3.1 below. If it is in the middlebox, selection could be done explicitly, using the RTP sequence number and timestamp for identification. However, as noted in Section 5.3.2 below, our current implementation uses a simpler alternative.

5.3 Design of DiversiFi

An "End-to-End" approach to realizing DiversiFi would be to work with an unmodified network infrastruture, in particular, unmodified APs and no middleboxes. This is illustrated in Figure 7(a). As noted in Sections 5.2.4, the PSM mechanism could be used to perform network-side buffering on the secondary link. However, this can be quite inefficient.

The per-client buffer at the AP is typically managed as a tail-drop queue and moreover can grow to a large size (e.g., the default size is 64 packets in OpenWRT, and 50-500 packets in Aruba [13]). When the client, upon missing a packet on the primary link, switches to the secondary link and sends a wakeup message, the AP would only deliver the missing packet after it has delivered the possibly many packets ahead of it in the queue. This would result in a significant overhead due to unnecessary duplication of packets over the air. Further, if the tail-drop queue at the secondary AP fills up, new packets will be dropped, making these unavailable for recovery should these be lost on the primary link.

5.3.1 AP with Minimal Modification

The key source of inefficiency above is that the secondary AP's buffer is maintained as a tail-drop queue that can grow to a large size. What we need instead is a buffer that holds a *small number* of the *most recent* packets. Accordingly, we introduce two changes in how the AP's buffer is managed: *head-drop* behaviour instead of tail-drop, and a *settable maximum queue size*, as illustrated in Figure 7(b). *Note that these changes only apply to the real-time links; the default link, DEF, would remain unaffected.*

The maximum queue size should be set based on the characteristics of the stream. For example, for a VoIP stream, if we have a budget of 100ms for the WiFi hop (as noted in Section 4.2) and the inter-packet spacing is 20ms, the maximum queue size should be set to 5 packets. In general, the client could signal the desired maximum queue size to the AP on a per virtual interface basis, using an unused information element in the 802.11 association request frame.

With this "Customized AP" approach (the client logic for which is shown in Algorithm 1), head-drop queuing with a small queue size would ensure that the queue is purged of all but a small number of the most recent packets. Therefore, when the client switches to the secondary link to recover a packet it has missed on the primary link, it would at most receive the small number of packets that are in the secondary's queue. Thus, much of the inefficiency of the "End-to-End" approach is avoided.

As a further optimization, the client could perform implicit packet selection (Section 5.2.5), by switching to the secondary link just a little before the desired (i.e., missing) packet reaches the head of the queue. Then, in principle, the client should be able to receive the desired packet, and im-

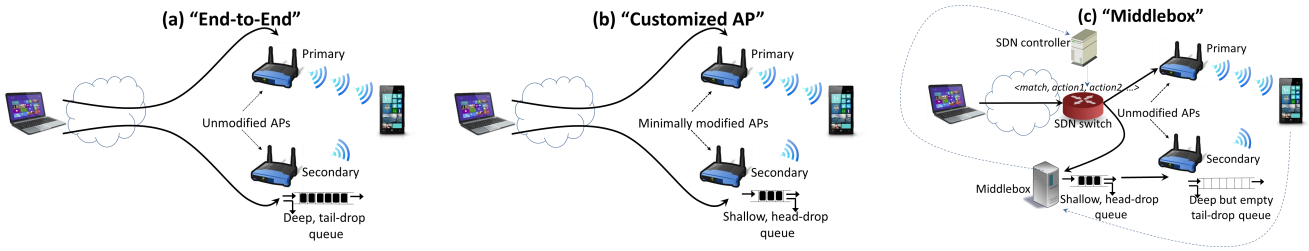


Figure 7: Architectural alternatives for DiversiFi. Solid/dashed lines show the data/control flow.

mediately switch back to the primary link, thereby avoiding any duplication. However, in practice we find that the AP could also transmit additional queued packets, when all of these are handed down to the hardware queue in one go. We quantify the consequent overhead in our experiments presented in Section 6.

Note that although there would be additional delay incurred in retrieving a missing packet over the secondary link, this would not impact the user-perceived quality so long as it is less than the playout delay. The *MaxTolerableDelay* parameter in Algorithm 1 is set accordingly, and any packet not retrieved within this period is deemed as lost. Hence, in our evaluation we focus on the residual packet loss even after the secondary link has been tapped rather than on the additional delay.

While the "Customized AP" approach calls for changes to the typical AP behaviour (albeit no hardware changes), we note that head-drop queuing is gaining broader support for other reasons (e.g., the CoDel proposal [1] to address the bufferbloat problem) and is already supported in certain AP implementations (e.g., OpenWRT). Ubuntu, which is used as an embedded OS, supports both head-drop and a settable queue size [9]. Further, we believe it would be easy for AP vendors to implement these.

5.3.2 *Unmodified APs with Middlebox*

An alternative that would leave the AP unmodified is to move the functionality of network-side buffering into a separate middlebox. An SDN-capable switch would replicate the stream, sending one copy directly to the client via its primary link and the second copy to the middlebox (which would otherwise not be on the data path). The middlebox would perform network-side buffering in lieu of buffering at the secondary AP. Figure 7(c) illustrates this architecture.

When the client misses a packet on the primary link and wishes to retrieve it over the secondary link (to benefit from link diversity, per Section 4), the client switches to the secondary link and sends a request to the middlebox for the specific packet(s) that it desires. After receiving the packets, it switches back to the primary link. Depending on the packet deadlines, this process can be defered a little to allow the client to retrieve more than one missing packet in one go. In any case, the secondary AP merely acts as a conduit for the packets forwarded to it by the middlebox and does not itself perform any buffer management.

Algorithm 1 Client logic with "Customized AP". Upon missing a packet, the client switches to secondary just in time for the missing packet to reach the head of the queue. The client also switches to the secondary periodically, just to keep the association alive.

procedure LINKSWITCH(*IPS, MTD, LSL*) // *InterPktSpacing = 20 ms, MaxTolerableDelay = 100 ms, LinkSwitchingLatency = 2.8 ms*
  SRT ← *40 ms // SecondaryResidencyTime*
  PLT ← *2 * IPS (= 40 ms) // PacketLossTimeout*
  AKT ← *30 s // AssociationKeepaliveTimeout*
  APQL ← $\frac{MTD}{IPS}$*(= 100/5 = 5) // APQueueLen*
  ETTRH ← *IPS * APQL - LSL // ExpectedTimeToReachHead*
  while true **do**
    ReceivePackets
    if *PacketNotReceived* **then**
      SetPacketAsLostOnPrimary
      ScheduleSwitchToSecondaryAfter (ETTRH)
      ScheduleSwitchBackToPrimaryAfter (ETTRH+PLT)
    if *LostPacketReceivedOnSecondary* **then**
      SwitchBackToPrimaryImmediately
    if *SecondaryInactiveFor(AKT)* **then**
      SwitchToSecondary
      ScheduleSwitchBackToPrimaryAfter(SRT)

Since the middlebox is under our control and the client performs explicit packet selection, this approach could, in principle, avoid duplicating any packets. However, our current implementation is based on a simple start-stop protocol, wherein the middlebox starts delivering packets upon receipt of start message and stops upon receiving a stop message. So there could be some duplication of packets.

Further, replication at an SDN-capable switch on the local network would mean that the remote peer remains uninvolved. Such a deployment, with a middlebox and an SDN-capable switch that the client has the credentials to install rules on, would likely be feasible in settings such as enterprises, campuses, etc., where the network is under the control of a single entity. We evaluate middlebox performance in Section 6.4.

5.4 Client-side Implementation

We have implemented the client-side piece of DiversiFi as an user-level library that delivers real-time streams to applications. Multi-link association is implemented using the stock ath9k driver on the Linux 2.6.33 kernel, which already supports multiple instances of virtual station mode needed to maintain multiple associations. The stock implementation had minor bugs, which we fixed. For instance, one corner case related to the the power-save message not being successfully received by the AP. To address this, we added 5 driver-level retry attempts to increase reliability, and also ensured that the channel switch operation is only invoked after the power-save message has been successfully delivered.

With regard to deployability on other platforms, both Windows and Android WiFi drivers also include support for multiple virtual NICs, for supporting WiFi-Direct mode in conjunction with station mode. We believe that multi-station mode support can be enabled through software-only changes, in the WiFi driver, similar to the existing implementation in the ath9k Linux driver.

6. EVALUATION

The goal of our evaluation is to see if our implementation of DiversiFi over a single NIC and carefully managed network-side buffering, yields a diversity gain along the lines of two NIC experiments showed. We are also interested in characterizing the overhead and coexistence issues, if any.

6.1 Experimental Setup

Our experiment setup consisted of a Linux laptop equipped with a single NIC — an Ath9k based DLink 802.11bgn PCMCIA card — which serves as the client. We implemented minimal AP customization from Section 5.3.1 on Netgear WNDR3800 802.11n AP with 2x2 MIMO running OpenWrt. We setup the APs on two different 2.4GHz channels (1 and 11) at the diagonal ends of a rectangular office space (30mX15m area) with cubicles and walls. At most locations within the office, the client could maintain concurrent associations with both APs. The stronger of the two links is set as the Primary, and the other one as the Secondary.

At each location we simulated VoIP calls from a wired client on the LAN to the WiFi-client. The WiFi-client then requests the sender-side DiversiFi user library to replicate the stream to the IP address of the secondary link in addition to the primary.

Separately, we also ran experiments with a middlebox, the details of which appear in Section 6.4 below.

6.2 Loss Recovery

We conducted 61 sets of runs that interleaved single link experiments and DiversiFi at different locations in the office building. We calculated the packet loss rate in worst 5-second window during each call. Figure 8 plots the CDF of these loss rates, both for the single link case (where the client connects over the stronger, primary link), and for DiversiFi (where the client connects over both primary and secondary links). We find that DiversiFi significantly outperforms the single-link case. For instance, while the 90^{th}%tile loss rate (computed by considering the worst 5-second period for each of the 61 calls) for the primary link was 11.6% and for the secondary link was 52%, that for DiversiFi was only 1.2%. The large gain with DiversiFi arises because it is quite unlikely that the worst 5-second period experienced on the secondary would coincide with that on the primary.

DiversiFi also helps cut down the incidence of burst losses, as shown in Figure 9. When just the primary link is used, each 2-minute call suffers a loss of 44.3 packets on average, 35.9 of those in bursts of two or more packets. In comparison, with DiversiFi only 0.9 out of the 2.7 packets lost on average per call was in bursts (which implies that only a fraction of the calls suffered any burst losses at all).

This performance improvement also translates into an improvement in the poor call rate (PCR), which drops from 4.9% and 26.2%, respectively, when the primary link or the secondary link alone is used, to 0% with DiversiFi.

6.3 Duplication Overhead and Fairness

While DiversiFi yields significant gains, the question is how much overhead is imposed by replication and what impact it has on fairness to other, non-real-time traffic. We focus here on the duplication overhead over WiFi; the overhead over the WAN would either be 100% if the replication is done at the source or 0% if it is done at a local middlebox.

For the set of 61 calls, the average packet loss rate on the primary link alone is 1.97%. (This is computed over the entire length of the calls, not just the worst 5-second period considered above.) With DiversiFi, the average packet loss rate drops to 0.05% because an overwhelming majority of the primary link losses are recovered on the secondary link. In addition to these *useful* transmissions over the secondary link, 0.62% of the packets were transmitted *unnecessarily* over the secondary link.

This (wasteful) duplication overhead of about 1 packet (0.62%) for every 3 lost on the primary (1.97%), arises because of two reasons: first, the AP inserts multiple packets into its hardware queue and transmits all of them even when the packet that the client is looking for is at the head of the queue, and second, the client has to receive packets from the secondary AP at some minimal periodicity, irrespective of whether there is packet loss, just to keep its (secondary) association alive.

To evaluate the impact of such wasteful duplication on other traffic, we performed iperf-based TCP measurements concurrently with a VoIP flow, with DiversiFi turned on or turned off alternately. Note that even when DiversiFi is turned on, the TCP traffic is confined to the DEF link (Section 5.2.2). However, when DiversiFi is turned on, the NIC would be switched between channels, so we need to examine the possibility of a performance impact on traffic flowing on DEF.

The results from 26 sets of runs is shown in Figure 10,

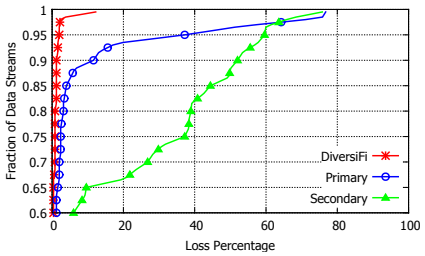


Figure 8: CDF of loss rates over the worst 5-second periods.

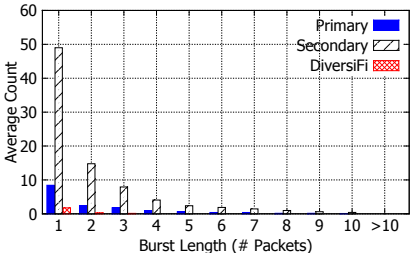


Figure 9: Distribution of packet loss burst length.

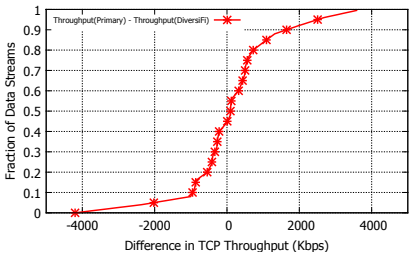


Figure 10: Difference in throughput of TCP flow, with and without DiversiFi.

which plots the CDF of the difference in TCP throughput between staying continuously on the primary link (DiversiFi is turned off) and switching back-and-forth between the primary and secondary links (DiversiFi is turned on). We find the points distributed almost evenly about the zero difference line, indicating that DiversiFi has little impact on the throughput of the TCP flow. Indeed, over the 26 runs, the average TCP throughput with DiversiFi turned on is 3.9 Mbps compared to 4.0 with it turned off, a difference of only 2.5%.

Thus, we conclude that DiversiFi provides significant gains for real-time VoIP flows while imposing little overhead in terms of channel switching or unnecessary duplication, thereby ensuring coexistence with competing, non-real-time traffic.

6.4 Middlebox Performance

We also evaluate the performance of network-side buffering at a middlebox compared to buffering at the AP. We implemented the middlebox using MIT Click router software V2.1 running on a quad-core i7 16GB RAM machine, and also setup an SDN switch using Open vSwitch V2.0.2, and Floodlight controller V1.0 software. The DiversiFi client library requests the middlebox to start replicating a specific real-time flow destined to the same client, which triggers the middlebox to have a match-action rule for replication installed in the SDN switch. The replicated packets are buffered in a shallow head-drop queue and retrieved by the client whenever packet loss is detected.

Table 3: Delay in milliseconds to collect a buffered packet on the Secondary link for two schemes

	Total	Switching	Network	Queuing
Middlebox	5.2	2.3	2	0.9
AP	2.8	2.3	0.5	-

One question is how much additional latency is entailed in the client contacting the middlebox for missing packets as compared to retrieving these from the secondary AP. We measured the delay between the reception of last packet on the primary link and the first packet received on the secondary link across 100 runs of primary-to-secondary switches Table 3 reports the total delay as well as the break up. Channel switching operations, which includes sending power-save and wakeup messages, takes up an average latency of 2.3 plus 0.5 = 2.8 ms. An additional 5.2 minus 2.8 = 2.4 ms delay is introduced by the middlebox, which is likely to be acceptable for real-time streaming applications.

To evaluate the scalability of the middlebox, we initiated concurrent real-time streams numbering between 0 to 1000, where each stream gets replicated and sent to the middlebox. We then measured the delay experienced by a WiFi-client to retrieve a lost packet from the middlebox. We find that the delay increases very gradually with an increasing load of flows. At 1000 streams, the total delay, including switching overhead, increased by only 1.1 ms compared to zero load, suggesting that a single middlebox can easily serve a large WiFi deployment.

7. CONCLUSION

WiFi links are often a significant cause of poor performance for real-time interactive streaming such as VoIP. DiversiFi helps improve performance for such streaming significantly using a simple idea: cross-link replication of the stream over the multiple WiFi links that are often available to a client. Such replication helps cut down the poor call rate for VoIP by 2.24x. We show how network-side buffering allows DiversiFi to provide the benefits of replication to real-time flows, with little duplication overhead and while ensuring coexistence with non-real-time applications.

Acknowledgments

We thank the anonymous ACM CoNEXT 2015 reviewers for their constructive feedback. Author Baranasuriya was an intern at Microsoft Research India during part of this work. He was supported in part by A*STAR, Singapore, under SERC Grant 1224104049.

8. REFERENCES

[1] Controlled Delay (CoDel). http://www.bufferbloat.net/projects/codel.

[2] Google Hangouts. http://www.google.com/+/learnmore/hangouts/.

[3] IEEE 802.11e Standard, 2005. http://standards.ieee.org/findstds/standard/802.11e-2005.html.

[4] Microsoft Azure RemoteApp. https://www.remoteapp.windowsazure.com/.

[5] OnLive. https://www.onlive.com/.

[6] Quality of Service for Voice over IP. http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.pdf.

[7] Skype. http://www.skype.com/.

[8] Sony PlayStation Now. http://www.playstation.com/en-us/explore/psnow/.

[9] Ubuntu Stochastic Fairness Queueing. http://manpages.ubuntu.com/manpages/trusty/man8/tc-sfq.8.html.

[10] P.862 : Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs, February 2001. ITU-T Recommendation.

[11] P.862.1 : Mapping function for transforming P.862 raw result scores to MOS-LQO, November 2003. ITU-T Recommendation.

[12] UDP packet replication using Open vSwitch, November 2013. http://blog.sflow.com/2013/11/udp-packet-replication-using-open.html.

[13] Buffer size of AP for clients in power save mode, June 2014. http://community.arubanetworks.com/t5/Controller-Based-WLANs/Buffer-size-of-AP-for-clients-in-power-save-mode/ta-p/176888.

[14] G. Ananthanarayanan et al. COMBINE: Leveraging the Power of Wireless Peers through Collaborative Downloading. In *MobiSys*, 2008.

[15] J. Apostolopoulos and M. Trott. Path Diversity for Enhanced Media Streaming. *IEEE Communication Magazine*, 42:80–87, 2004.

[16] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting mobile 3G using WiFi. In *MobiSys*, 2010.

[17] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services, December 1998. RFC 2475.

[18] R. Chandra, V. Bahl, and P. Bahl. MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card. In *INFOCOM*, 2004.

[19] A. Croitoru, D. Niculescu, and C. Raiciu. Towards Wifi Mobility without Fast Handover. In *NSDI*, May 2015.

[20] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan. WiFi, LTE, or Both? Measuring Multi-Homed Wireless Internet Performance. In *IMC*, 2014.

[21] L. Golubchik, J. Lui, T. Tung, A. Chow, A. Lee, G. Franceschinis, and C. Anglano. Multi-path Continuous Media Streaming: What are the Benefits? *Performance Evaluation*, 49:429–449, 2002.

[22] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan. Understanding and Mitigating the Impact of RF Interference on 802.11 Networks. In *SIGCOMM*, 2007.

[23] Jamie Stark. Lync and Software-Defined Networking, December 2013. http://blogs.technet.com/b/lync/archive/2013/12/17/lync-and-software-defined-networking.aspx.

[24] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput. In *NSDI*, 2008.

[25] K. Lee, D. Chu, E. Cuervo, Y. Degtyarev, S. Grizan, J. Kopf, A. Wolman, and J. Flinn. Outatime: Using Speculation to Enable Low-Latency Continuous Interaction for Cloud Gaming. In *MobiSys*, 2015.

[26] A. Lozano and N. Jindal. Transmit Diversity vs. Spatial Multiplexing in Modern MIMO Systems. *IEEE Transactions on Wireless Communications*, 9:186–197, 2010.

[27] A. K. Miu, H. Balakrishnan, and C. E. Koksal. Improving Loss Resilience with Multi-Radio Diversity in Wireless Networks. In *Mobicom*, 2005.

[28] A. K. Miu, G. Tan, H. Balakrishnan, and J. Apostolopoulos. Divert: Fine-grained Path Selection for Wireless LANs. In *MobiSys*, 2004.

[29] H. Rahul, H. Hassanieh, and D. Katabi. SourceSync: A Distributed Wireless Architecture for Exploiting Sender Diversity. In *SIGCOMM*, 2010.

[30] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *NSDI*, 2012.

[31] H. Schulzrinne and S. Casner. RTP Profile for Audio and Video Conferences with Minimal Control, July 2003. RFC 3551.

[32] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, July 2003. RFC 3550.

[33] V. Shrivastava, S. Rayanchu, J. Yoon, and S. Banerjee. 802.11n Under the Microscope. In *IMC*, 2008.

[34] T. Szigeti and C. Hattingh. *End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs*. Cisco Press, November 2004. http://www.ciscopress.com/store/end-to-end-qos-network-design-quality-of-service-in-9781587051760.

[35] E. Vergetis, R. Guerin, and S. Sarkar. Packet-Level Diversity - From Theory to Practice: An 802.11-based Experimental Investigation. In *ITC*, 2005.

[36] E. Vergetis, E. Pierce, M. Blanco, and R. Guerin. Packet-Level Diversity - From Theory to Practice: An 802.11-based Experimental Investigation. In *Mobicom*, 2006.

[37] A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker. Low Latency via Redundancy. In *CoNEXT*, 2013.

[38] B. Weiss, S. Moller, B. Belmudez, and B. Lewcio. Analysis of Call-Quality Prediction Performance for Speech-only and Audio-Visual Telephony. In *IEEE QoMEX*, September 2014.