Poster: *AMuSe*: An Agile Multipath TCP Scheduler for Dual-Band 802.11ad/ac Wireless LANs

Swetank Kumar Saha[1], Shivang Aggarwal[1], Dimitrios Koutsonikolas[1], Joerg Widmer[2]
[1]University at Buffalo, The State University of New York, [2]IMDEA Neworks Institute, Madrid, Spain
{swetankk,shivanga,dimitrio}@buffalo.edu,joerg.widmer@imdea.org

ABSTRACT

802.11ad links provide data rates up to 6.7 Gbps but remain highly susceptible to blockage and mobility. On the other hand, legacy 802.11ac/n links yield much lower rates but are robust even under dynamic scenarios. In this work, we explore using Multipath TCP (MPTCP) to engage both 802.11ad and 802.11ac interfaces simultaneously for performance speed-up and improved reliability. We show that vanilla MPTCP achieves these goals under static conditions but often results in performance worse than using the faster interface alone under dynamic scenarios. We then design and implement *AMuSe*, a new MPTCP scheduler that allows MPTCP to perform near-optimally under all scenarios.

ACM Reference Format:
Swetank Kumar Saha[1], Shivang Aggarwal[1], Dimitrios Koutsonikolas[1], Joerg Widmer[2]. 2018. Poster: *AMuSe*: An Agile Multipath TCP Scheduler for Dual-Band 802.11ad/ac Wireless LANs. In *The 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18), October 29-November 2, 2018, New Delhi, India.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3241539.3267755

1 INTRODUCTION

Millimeter-wave (mmWave) wireless is fast emerging as the prime candidate technology for providing multi-Gbps data rates in future wireless networks. The IEEE 802.11ad standard provides data rates of up to 6.7 Gbps. It achieves this multi-fold increase over legacy WiFi through 2 GHz-wide channels. Nonetheless, communication at mmWave frequencies faces fundamental challenges due to the high propagation and penetration loss. The use of directional transmissions makes links susceptible to disruption by human blockage and client mobility. Even if future PHY/MAC improvements may result in faster beam steering, any realistic indoor scenario is expected to contain enough dynamism to cause a large number of re-connection events.

In this work, we tackle the challenge of supporting the multi-Gbps throughput provided by the 60 GHz technology while still providing the reliability of legacy WiFi, which is the key for wide-spread adoption of 60 GHz WLANs. Specifically, we explore Multipath TCP (MPTCP), a transport layer protocol that uses the 802.11ad and 802.11ac interfaces simultaneously to achieve higher throughput when both networks are available and seamlessly falls back to 802.11ac in an application-transparent manner when the 802.11ad network becomes unavailable. MPTCP's design as a transport layer solution decouples it from other layers.

In spite of these attractive features, using MPTCP in multi-band WLANs is far from straightforward. A large number of recent studies investigated the performance of MPTCP in scenarios combining WiFi and cellular (3G/LTE) networks (e.g., [2, 3]) and showed that the protocol performs poorly over heterogeneous paths. The authors in [1, 4] even argue that the two radios should never be used simultaneously.

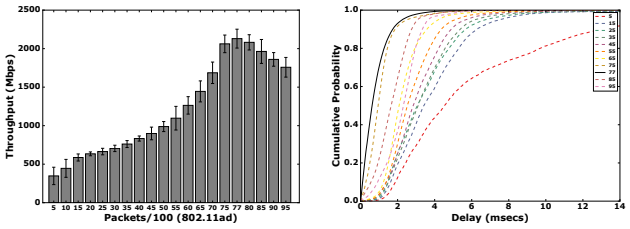
In contrast, to the best of our knowledge, our work is the first to show that the use of MPTCP is not just a viable but even a very promising solution towards dual-band 5/60 GHz WLANs. Our experimental study, using COTS APs and laptops, reveals that MPTCP can provide optimal throughputs under baseline, static scenarios. However, realistic dynamic environments, e.g., 802.11ac contention or blockage of the 802.11ad link, can result in severe performance degradation. We then design and implement *AMuSe*, a new MPTCP scheduler that addresses the root-cause of the performance degradation. Our evaluation in real indoor WLAN scenarios shows that *AMuSe* can achieve up to **2.5x** throughput improvement and can reduce application-level delay by several 10s of seconds compared to the default MPTCP.

2 MPTCP PERFORMANCE

We first experimented with four congestion control algorithms available in the Linux MPTCP implementation: *Cubic* (decoupled), *Lia*, *Olia*, and *Balia* under backlogged traffic

MobiCom '18, October 29-November 2, 2018, New Delhi, India

ACM ISBN 978-1-4503-5903-0/18/10.
https://doi.org/10.1145/3241539.3267755



(a) Throughput vs. packet ratio. (b) Delay (*ofo-queue*): $Pkts_{ad}$.

Figure 1: Impact of packet scheduling decisions.

and found that for each of the four algorithms, *MPTCP can indeed achieve performance very close to the expected sum* (at least greater than 96% and up to 99%). We next turn our attention to another key MPTCP component: the **packet-scheduler**, responsible for the distribution of application traffic among the subflows. To understand how the traffic distribution between the subflows impacts MPTCP performance, we design an MPTCP scheduler *FixedRatio* that performs packet assignment based on a user-defined ratio. Fig. 1(a) plots the MPTCP throughput against the number of packets assigned to the 802.11ad subflow ($Pkts_{ad}$) out of every 100 packets. Maximum throughput of ~2.1 Gbps is achieved with $Pkts_{ad} = 77$ and performance worsens as we move away from this value with the worst throughput being as low as 400 Mbps ($Pkts_{ad} = 5$).

We found that the stark difference in performance with different assignment ratios is a result of the degree to which packets arrive *out-of-order* in the end-to-end MPTCP flow. A higher number of out-of-order packets can cause packets to be buffered in the receiver's *ofo-queue* and in extreme cases can even result in throttling of the sender because of limited/no space in the receiver's buffer. In fact, in Fig. 1(b), which plots the CDF of the delay experienced by bytes in the *ofo-queue*, we observe that the $Pkts_{ad} = 77$ value (that results in highest throughput) indeed yields the lowest delay.

Throughput-optimal ratio. $Pkts_{ad} = 77$ results in optimal throughput as the the underlying packet-distribution ratio imposed by this assignment $Pkts_{ratio} = Pkts_{ac}/Pkts_{ad} = 23/77 = 0.29$ is nearly identical to the ratio of the actual individual throughputs of the two interfaces $Tput_{ratio} = Tput_{ac}/Tput_{ad} = 500/1600 = 0.31$. Assigning packets in this very specific ratio minimizes the chance of packets arriving out-of-order at the meta-level MPTCP buffers/queues.

3 PERFORMANCE ISSUES

We now look at more challenging scenarios where we show that the default MPTCP architecture is unable to adapt, resulting in sub-optimal performance which is often worse than that of single path TCP (SPTCP) over the faster subflow.

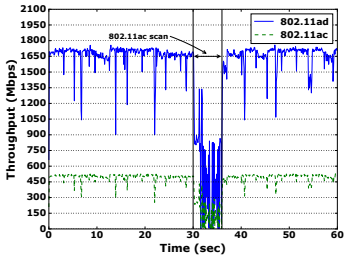
Network Scans. Fig. 2(a) shows the throughput of 802.11ad and 802.11ac subflows over 60 s and the scan initiated at the 30 s mark. The 802.11ac throughput is cut down severely during the scan period (marked as "802.11ac scan") that lasts for around 6 s, as expected. However, we observe that the 802.11ad flow is also impacted negatively during this period, even though the scan takes place in the 5 GHz band. On investigation, we observed a 6x increase in the amount of data held in the *ofo-queue* at the receiver end. During the scan period, the packet scheduler, unaware of the scan, assigns packets in the same ratio as before the scan. This is problematic as the receiver's packet stream now has *gaps*, preventing the receiver from delivering packets to the application.

WiFi (802.11ac) Contention. Fig. 2(b) shows a timeline of the per-flow throughput of a 180 s MPTCP session. We start with a static link where 802.11ad and 802.11ac are at their maximum throughputs of ~550 Mbps and ~1650 Mbps, respectively, and we introduce contention with 300 Mbps TCP cross-traffic at the 30th second for 30 s. The throughput of the 802.11ac subflow drops by 300 Mbps to around 250 Mbps, as expected. Surprisingly, the 802.11ad subflow is also affected negatively during the contention period with its throughput dropping below 1200 Mbps. In fact, the MPTCP throughput during the contention period averages to ~1450(=1200+250) Mbps (less than 802.11ad alone). On further investigation, we found that during the contention period the receiver advertised buffer space (*recv_win*) reduces significantly. Note that the *recv_win* is maintained at the meta-level and, although advertised on both subflows, is actually shared among them. Under such a scenario, the global sequence numbers cannot advance, even though *cwnd* allows for it, resulting in reduction of throughput on both interfaces.

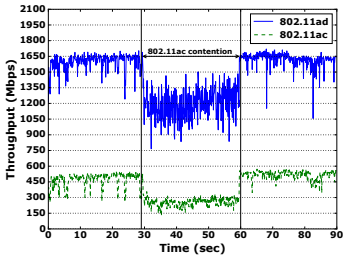
60 GHz (802.11ad) Blockage. Fig. 2(c) shows a timeline of subflow throughputs along with link status. Blockage is introduced at the 20^{th} second causing the link status to switch to fail after a further 2 s. Once the blockage is removed, connection at the MAC layer is restored at the 31^{st} second. We observe the following two issues:

(i) Although the 802.11ad link is restored at the 31^{st} second, MPTCP does not resume traffic on the 802.11ad subflow for another ~12 seconds until the 43^{rd} second. We found that in the case of a timeout-based loss event, TCP congestion-control sets the `pf` flag on the socket, indicating the flow to be *potentially failed*. The MPTCP scheduler treats subflows with the `pf` flag set as being unavailable and does not schedule any packets on them. TCP congestion-control, on the other hand, is waiting for an ACK to unset the `pf` flag and enter the `TCP_CA_RECOVERY` state that can restore the *cwnd* to the value before the loss event, but no packets are being directed to the 802.11ad subflow.

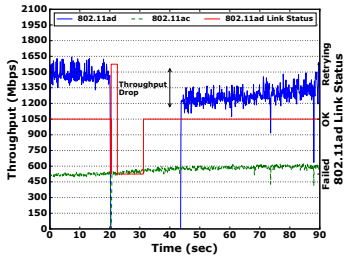
(ii) On resumption, 802.11ad flow starts with a *cwnd* and *ssthresh* that are half of their pre-loss values. Fig. 2(c) shows a sample timeline where the 802.11ad flow resumes to 1350 Mbps instead of 1650 Mbps.



(a) Network scan: Throughput timeline.

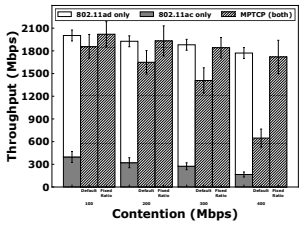


(b) 802.11ac contention: Timeline showing throughput drop during contention.

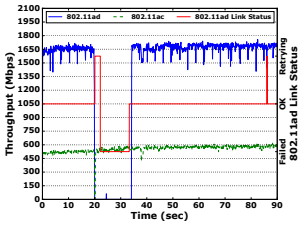


(c) 802.11ad blockage: 802.11ad throughput drop after re-connection.

Figure 2: Performance issues.



(a) *minRTT* vs. *FixedRatio*.



(b) 802.11ad Blockage: Improved recovery time.

Figure 3: *AMuSe* evaluation.

4 *AMUSE*: DESIGN & IMPLEMENTATION

***AMuSe*-SCAN** arbitrates the network scan requests generated from the user space and disables the scheduling of packets to the subflow where the request has been made for the duration of the network scan. However, disabling future scheduling alone may not be enough to prevent packets from being held-up in the TCP queues or at any of the buffers in the lower layers of the network stack. We thus adopt a two-step approach, where *AMuSe*: (1) stops the assignment of packets to subflow about to undertake scanning and (2) waits for the subflow-level *send-queue* to be emptied out. We observed that with *AMuSe*'s network scan management solution applied the 802.11ad throughput remains unaffected during the scan interval. We repeated the measurements several times and observed **2.2x** performance gain on average.

***AMuSe*-CONTENTION** leverages our findings regarding the existence of a unique MPTCP throughput-optimal ratio, for given subflow throughputs. The reaction to contention is to set the packet-assignment ratio to match the ratio of the throughput of 802.11ad and 802.11ac flows, accounting for the drop in 802.11ac throughput due to contention. We test our solution under different amounts of contention (from 100 Mbps to 400 Mbps). Fig. 3(a) shows the expected sum, after accounting for 802.11ac throughput reduction in the presence of contention, and MPTCP performance under the default and *FixedRatio* scheduler, which uses the optimal ratio for a given amount of contention. In all cases, *FixedRatio*'s throughput is close to the expected sum while the default scheduler's throughput can be much lower.

***AMuSe*-BLOCKAGE** reduces the delay in resuming traffic over the 802.11ad subflow by resetting the `pf` flag to allow for traffic to be scheduled on the 802.11ad subflow. However, we found that this alone was not enough to resume the traffic flow on the 802.11ad interface. When the 802.11ad link is blocked, the subflow-level *cwnd* is cut to 1, with packets in flight also equal to one. As a result, the scheduler is unable to schedule any new packets on 802.11ad subflow since the *cwnd* is reported as being full. In order to overcome this, *AMuSe* uses the TCP's window recovery mechanism to restore the *cwnd* to the value just before loss the event. In contrast to Fig. 2(c), where MPTCP resumed traffic on the 802.11ad subflow after a 12 s delay, with *AMuSe* engaged (Fig. 3(b)) MPTCP starts using the 802.11ad interface in less than 1s after link re-establishment. This is a vast reduction in delay for resuming traffic on the subflow. In a dynamic environment, where such blockage events will occur quite frequently, *AMuSe*'s gains would translate into a significant improvement in user-experience.

5 ACKNOWLEDGEMENTS

This work has been supported in part by NSF grant CNS-1553447, the ERC project SEARCHLIGHT grant no. 617721, the Ramon y Cajal grant RYC-2012-10788, and the Madrid Regional Government through the TIGRE5-CM program (S2013/ICE-2919).

REFERENCES

[1] Kien Nguyen, Mirza Golam Kibria, Kentaro Ishizu, and Fumihide Kojima. 2017. Feasibility Study of Providing Backward Compatibility with MPTCP to WiGig/IEEE 802.11ad. In *Proc. of IEEE Vehicular Technology Conference Fall (VTC-Fall)*.

[2] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. 2012. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *Proc. of 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

[3] Swetank Kumar Saha, Abhishek Kannan, Geunhyung Lee, Nishant Ravichandran, Parag Kamalakar Medhe, Naved Merchant, and Dimitrios Koutsonikolas. 2017. Multipath TCP in Smartphones: Impact on Performance, Energy, and CPU Utilization. In *Proc. of ACM MobiWac.*

[4] Sanjib Sur, Ioannis Pefkianakis, Xinyu Zhang, and Kyu-Han Kim. 2017. WiFi-Assisted 60 GHz Wireless Networks. In *Proc. of ACM MobiCom.*