Revisiting Congestion Control for WiFi Networks

Xinle Du
Huawei Technologies
duxinle1@huawei.com

Jie Li
Huawei Technologies
leo.lijie@huawei.com

Yiyang Shao
Huawei Technologies
shaoyiyang@huawei.com

Wei Wang
Huawei Technologies
wangwei375@huawei.com

Shuihai Hu
Huawei Technologies
hushuihai1@huawei.com

Jingbin Zhou
Huawei Technologies
zhoujingbin@huawei.com

Kun Tan
Huawei Technologies
kun.tan@huawei.com

ABSTRACT

WiFi networks, widely utilized by wireless devices, have become increasingly complex and congested environments, leading to noticeable delays, jitter, and throughput degradation for end-to-end network flows in today's Internet. Through detailed experimental observations, we identified the TCP victim problem in WiFi, where TCP erroneously detects congestion and interacts with WiFi routers, resulting in a significant throughput decrease for certain hosts. In this paper, we introduce Cupid, a novel congestion control algorithm that relies on receiver-side WiFi physical layer measurements. Cupid accurately assesses congestion by measuring parameters such as airtime utilization, concurrency, and rates, allowing for precise rate adjustments. Our results demonstrate that whether employed alone or alongside other congestion controls, Cupid effectively mitigates the TCP victim problem. Furthermore, when used independently, Cupid can also reduce latency.

CCS CONCEPTS

• **Networks** → **Transport protocols**.

KEYWORDS

WiFi Networks, Congestion Control

ACM Reference Format:
Xinle Du, Jie Li, Yiyang Shao, Wei Wang, Shuihai Hu, Jingbin Zhou, and Kun Tan. 2024. Revisiting Congestion Control for WiFi Networks. In *The 8th Asia-Pacific Workshop on Networking (APNet 2024), August 03–04, 2024, Sydney, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3663408.3663421

1 INTRODUCTION

WiFi network channels serve as the primary connection for the termination of end-to-end data flows to a mobile endpoint, leading to increased delays, latency fluctuations, packet loss, and bandwidth constraints [10, 18]. TCP and its variants (e.g., NewReno [8], Cubic [11], BBR [6]) are known to perform poorly over WiFi networks due to high capacity variability, stochastic packet losses that are not linked to congestion, and large bandwidth-delay products (BDP) [10, 24, 25]. On one hand, the rapid changes in the WiFi physical layer channels make it challenging for TCP to adjust quickly [10, 25]. The bandwidth of WiFi standards such as WiFi 5 [1] (433 - 6933 Mbps), WiFi 6 [2] (574 - 9608 Mbps), and even the latest WiFi 7 [3] (1376 - 46,120 Mbps) continues to increase, making them more susceptible to signal interference. On the other hand, the competition among multiple users and user mobility further contributes to channel fluctuations.

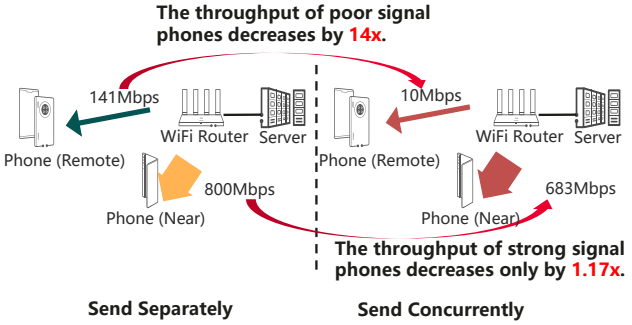


Figure 1: An example of TCP victim problem in WiFi.

In addition to the widely known issues mentioned above, a new problem, which prevents TCP hosts in WiFi from accessing available bandwidth, has been discovered by us. We refer to this phenomenon as the TCP victim problem in WiFi scenarios: when hosts concurrently download, there is always a portion of hosts who cannot access sufficient bandwidth, especially those who are farther away from the wireless router. Figure 1 illustrates an example. In Figure 1, two mobile phones are connected via WiFi in the same wireless router. One phone is in close proximity to the WiFi router, while the other is at a greater distance. They are both downloading data from the server. When the phones download separately, the phone (Remote) with poor signal quality achieves a throughput of approximately 141Mbps, while the phone (Near) with good signal quality reaches up to 800Mbps. However, when they download concurrently, the throughput of the poor signal phone (Remote) drops by over 14 times, while the phone (Near) with strong signal

APNet 2024, August 03–04, 2024, Sydney, Australia

ACM ISBN 979-8-4007-1758-1/24/08
https://doi.org/10.1145/3663408.3663421

quality experiences only a 1.17x decrease in throughput. In an ideal scenario, the throughput of both phones should decrease to half of the individual sending throughput when they are concurrently downloading.

The TCP victim issue in WiFi occurs due to inaccurate congestion detection, resulting in the bandwidth converging towards an incorrect value. It is not due to the physical layer's inability to allocate more bandwidth to weak signal hosts. For congestion control based on packet drop (e.g., Cubic [11]), the weak hosts slow down due to congestion-unrelated stochastic packet losses and insufficient batching. For CC based on probing rate (e.g., BBR [6]), the weak hosts have a less aggressive rate probing, resulting in a low bandwidth.

We must investigate novel congestion signals to better distribute wireless channel usage in WiFi, aiming to overcome the constraints of conventional WiFi congestion detection methods and inadequate rate adjustment laws. In this paper, we propose Cupid[1], a novel congestion control based on WiFi physical-layer bandwidth measurements, taken at the mobile endpoint. Cupid consists of two modules: a) A subtle congestion detection and identification mechanism that achieves highly accurate measurements of WiFi links such as airtime utilization, concurrency, and rates, enabling more precise control of the sender's rate when attempting to match its transmission rate with the available wireless capacity (if the bottleneck capacity is within the wireless link itself). b) A receiver-driven rate adjustment rule that accelerates the convergence speed of rate adjustments, exhibits robustness to sudden traffic spikes, and adapts to changes in link capacity. In this paper, Cupid only addresses congestion in WiFi networks. The Internet bottleneck congestion will be considered as future work.

We have implemented an Cupid prototype on top of UDP, and built a small testbed for preliminarily validating the effectiveness of Cupid. We find that: (1) Cupid can accurately measure network bandwidth and adjust sending rate to achieve high throughput, low latency, and fairness; (2) Compared to Cubic and BBR, Cupid can address the victim problem, resulting in a 50X increase in throughput and a 5X reduction in latency for weaker users; (3) If Cupid coexists with other algorithms, when Cupid is the victim, it actively competes for bandwidth; conversely, when Cupid is not the victim, it voluntarily yields bandwidth, maximizing overall network efficiency.

2 WIFI DATAPATH

Let's start by introducing the data path of WiFi to facilitate our understanding of the queuing models in WiFi routers and how they differ from those in wired routers. Figure 2 shows the life of a packet at a WiFi router.

(1) When each packet enters the wireless router, it goes through an input arbiter to identify the specific mobile device to which it is destined.
(2) After passing through the arbiter, each packet will be queued in its per-user queue.

[1]Cupid is the god of love, depicted as flying in the air with two wings and precisely aiming at two lovers with bows and arrows. We use 'Cupid' to denote the precision with which the sending rate aligns with the available WiFi air channel bandwidth.

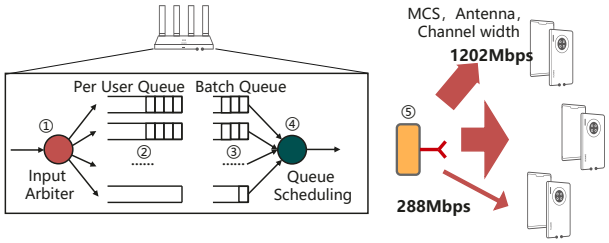


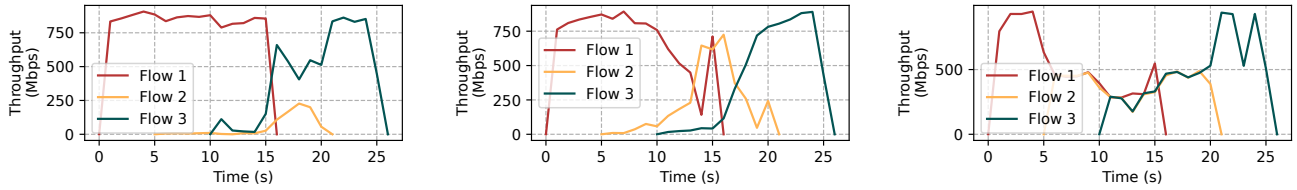
Figure 2: The datapath in WiFi Routers.

(3) After passing through the per-user queue, each packet enters the batch queue. In order to increase the WiFi air channel efficiency, WiFi routers aggregate packets to be sent in batches (e.g., A-MSDU and A-MPDU [9]). Therefore, in the batch queue, packets not only wait to be transmitted individually but also wait to be aggregated into a batch.
(4) The specific process of queue scheduling is quite complex, involving low-level physical scheduling information, frame aggregation, and other factors (e.g., OFDMA scheduling, MU-MIMO scheduling and SU scheduling [17]). For ease of understanding, we simplify the model here and approximate it as a round-robin (RR) algorithm. However, in queue scheduling, priority is given to scheduling queues that are full in the batch queue.
(5) As the wireless air medium is a shared resource, only one set of data can be transmitted in space at any given moment. In general, a wireless router typically transmits data for only one user at a time. Each user is matched with an underlying physical rate based on factors such as distance, interference, Modulation and Coding Scheme (MCS), and the number of antennas [21]. This rate is subject to change as conditions vary.
At the lower level, CSMA/CA and RTS/CTS control the medium, allowing it to be accessed by only one user at a time. Due to advancements in physical layer technologies and frequency domain signal multiplexing (such as OFDMA, MU-MIMO [17]), it is possible to have multiple users receiving simultaneously. However, the conditions for the utilization of these technologies are stringent, and in most cases, only one user is receiving at a time.

In summary, compared to traditional wired routers, wireless WiFi routers have the characteristic that **(1) each user has their own queue, (2) these queues need to be batch processed, (3) full batch queues are of high priority, and (4) each user has a different physical bandwidth**.

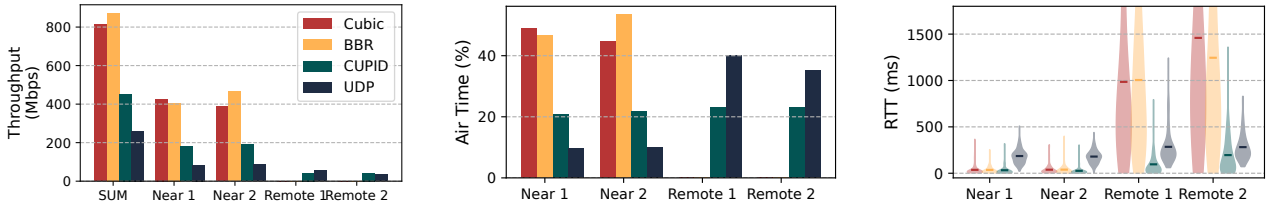
3 OBSERVATION AND INSIGHTS

TCP and its variants face challenges on WiFi networks due to capacity variability, random packet losses, and increasing bandwidth. The dynamic WiFi channels, coupled with user competition and mobility, contribute to these issues. In addition to the well-known issue mentioned above, we have identified a TCP victimization problem in WiFi. When hosts download simultaneously, there's consistently a segment of hosts unable to access ample bandwidth, particularly those situated farther from the wireless router. Next,



(a) Cubic (b) BBR. (c) Cupid.

Figure 3: Fairness results among the three smartphones for the TCP victim problem.



(a) Throughput. (b) Airtime utilization. (c) RTT.

Figure 4: Near-Remote results among the four smartphones for the TCP victim problem.

we will showcase the TCP victim problem in WiFi through experiments, analyze the root causes of the issue, and delve into insights provided by Cupid.

3.1 Observations

We now characterize the TCP victim problem in WiFi using the following testbed: smartphones equipped with WiFi 6 support, each featuring two antennas, and a WiFi 6 wireless router with six antennas connected to an Ubuntu server. The air channel bandwidth for handshaking between the smartphones and the WiFi router reaches a maximum of 1201Mbps, while the actual TCP throughput for a single user is around 800Mbps. We use Perf3 [19] for traffic generation. We varied the congestion control algorithms used, including Cubic, BBR, and UDP (representing no congestion control). We also used ping to measure the Round-Trip Time (RTT).

First, we test for fairness. We place three smartphones in close proximity to the router, spaced 5 seconds apart in their sending starts, with each smartphone sending for 15 seconds. Figure 3 illustrates the results. In Figures 3(a) and 3(b), when flows 2 and 3 start, Cubic and BBR cannot obtain sufficient bandwidth, making the later-starting flows victims. In Figure 3(c), Cupid ensures fairness across all flows. This unfairness is probabilistic, but when the signal quality varies, the unfairness becomes inevitable.

Next, we test fairness under different proximity scenarios. We placed two smartphones at a greater distance from the wireless router and two closer to it. we simultaneously sent data from the server to all four smartphones. Figure 4(a) illustrates the throughput results, revealing significantly low throughput for the remote user in TCP. (1) For Cubic, the throughput for the weak hosts is 0.03-0.51 Mbps, while the strong host achieves a throughput of 388-425 Mbps. (2) For BBR, the weak host is only 0.02-0.28 Mbps, while strong the host 406-465 Mbps. (3) For Cupid, the weak host is 35-54 Mbps, while strong the host 180-188 Mbps. (4) For UDP (without congestion control), the weak host is 35-54 Mbps, while the strong host achieves a throughput of 83-86 Mbps.

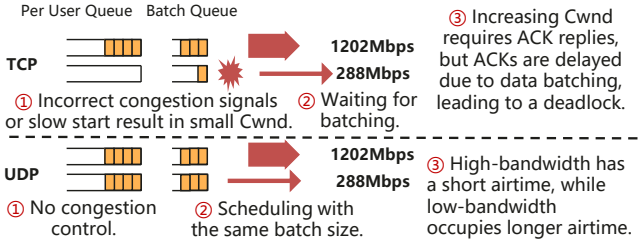


Figure 5: The reason of TCP victim problem in WiFi.

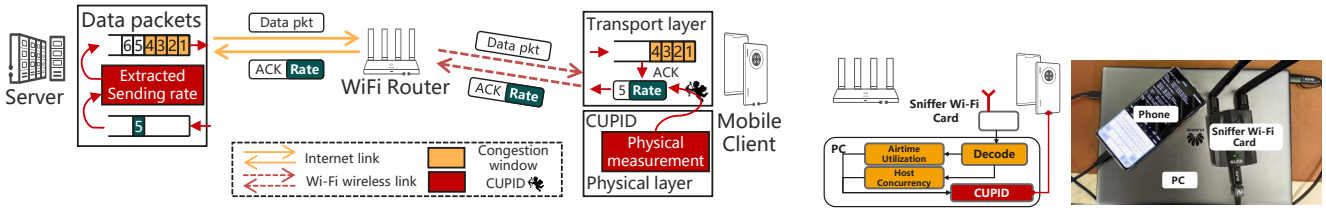
UDP can avoid the TCP victim problem by not restricting the rate of the weak host. This allows the weak host to achieve a high throughput. However, in the case of purely unordered transmission, overall airtime bandwidth utilization will decrease (UDP throughput combined will lose 68%), leading to another form of unfairness in terms of airtime utilization. Figure 4(b) illustrates a comparison of airtime utilization. It can be observed that the two weak hosts with UDP almost monopolize 80% of the airtime. BBR, Cubic, and UDP fail to fairly utilize airtime, whereas Cupid can.

Additionally, in the WiFi scenario, a more peculiar phenomenon occurs: the latency of weak hosts in Cubic and BBR remains remarkably high, as shown in Figure 4(c). Compared to TCP, UDP consistently exhibits high latency due to buffer filling, whereas Cupid manages to maintain a relatively low latency consistently.

Through the above analysis, it is evident that CUBIC and BBR adversely affect the throughput of weak hosts, while UDP adversely impacts the throughput of strong hosts. In the next section, we will carefully analyze the root causes behind why such phenomena occur.

3.2 Interaction Issues

TCP victim problem in WiFi leads to extreme unfairness in throughput, resulting in a significant disparity in user experience. Figure 5 illustrates the causes of the victim problem. For weak TCP users,



(a) Framework. (b) Testbed.

Figure 6: An overview of Cupid congestion control. The mobile clients decode the WiFi channel, which computes detailed available wireless capacity. Cupid senders control their send rate based on the estimated bottleneck capacity that the mobile user explicitly sends back, or based on the presence of ACKs from the receiver.

who already have lower bandwidth, (1) inaccurate congestion signals or slow start in TCP can result in some erroneous slowdowns (reduction in Cwnd). (2) The queue scheduling algorithm of the WiFi router prioritizes users with a full Batch queue. Therefore, if the Cwnd of a weak host is smaller than the queue length of the Batch queue, the weak host must wait for the strong host to finish sending before it can transmit. (3) The weak host faces a challenge in increasing its sending rate. It needs ACKs to increase its Cwnd, but ACKs require data packets to reach the receiver. Data packets cannot be sent until the Batch queue is full, and filling the Batch queue requires increasing the Cwnd of the weak host. This creates a deadlock, making it difficult for the weak host to improve its sending rate until the strong host finishes sending.

UDP results in significantly lower utilization for the strong host compared to the fair share. (1) UDP has no congestion control, leading it to indiscriminately fill the queue with data packets without restrictions. (2) The batch queue ensures a consistent amount of data is sent for each host with each transmission. (3) Hence, users with higher throughput send faster but occupy less airtime, while users with lower throughput send slowly, occupying more airtime.

In conclusion, existing congestion control mechanisms lack accurate congestion signals and struggle to achieve convergence targets in WiFi scenarios. We need novel congestion control approaches that utilize new congestion signals to converge towards a defined fair target state.

3.3 New Signal and New Target

Cupid needs new congestion signals and congestion control targets. Traditional transport layer congestion signals such as RTT, packet loss, and rate detection are not accurate enough in the WiFi. Only the actual bandwidth occupancy at the physical layer can accurately reflect the real WiFi congestion.

Traditional congestion control lacks accurate congestion information, hence it can only reach the target rate through probing. Let's revisit the fundamental goal of all congestion control, which is to make the sending rate match the (1) fair and (2) available residual bandwidth. From a god's-eye view, if each sender can accurately know the capacity C of the bottleneck link and the number N of concurrent senders, then each sender setting its sending rate to C/N would accurately match the rate, thereby achieving the mission of congestion control. However, due to the inability to accurately determine the available bandwidth and the number of concurrent connections, existing congestion control mechanisms need to explore and detect through probing.

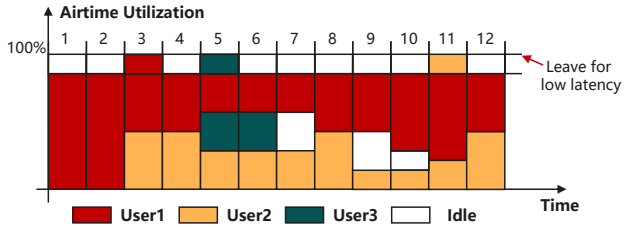


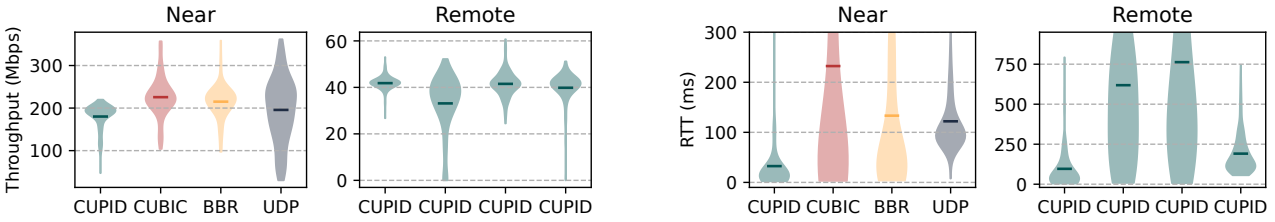
Figure 7: An example of Cupid.

Cupid utilizes the airtime occupancy to measure congestion and controls fairness by statistically measuring the number of concurrent senders. In WiFi, each user has a different airtime rate, and bandwidth cannot be considered as a shared resource. Based on the characteristics of WiFi, only one set of data is transmitted on the airtime at any given moment. Although the rates of each data set may differ, they share the resource of time. If there is constant transmission on the airtime, it indicates congestion. If each user occupies the same airtime, they are fair on a time basis. Therefore, Cupid employs **(1) airtime occupancy as the congestion signal** and aims for **(2) airtime fairness as the objective of congestion control**.

4 CUPID DESIGN

Cupid is an end-to-end congestion control algorithm designed specifically for WiFi, applicable to flows terminated on mobile devices through WiFi networks. The Cupid mobile client measures the WiFi physical channel, providing insights into the available airtime wireless capacity and the number of concurrent users. With its fine-grained capacity estimation, Cupid can promptly increase its transmission rate to capture new available capacity without causing any congestion when the bottleneck is a wireless hop, and correspondingly decrease its transmission rate if competition with other mobile users or wireless channel contention reduces wireless capacity. Figure 6 illustrates an overview of Cupid.

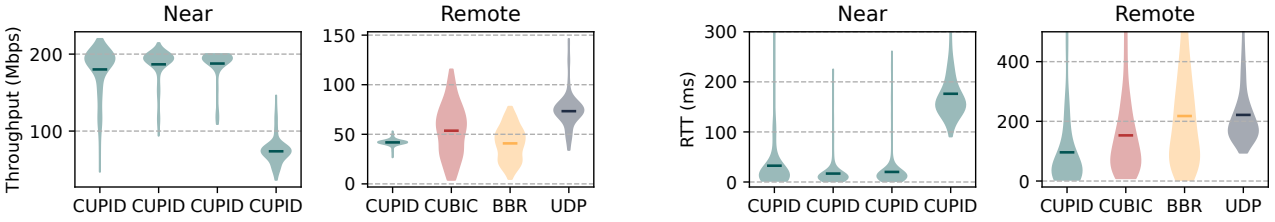
Due to the highly dynamic nature of traffic patterns, the end-to-end connection faces two possible network states, depending on the relative capacities of the bottleneck links in the Internet and the WiFi links. Most of the time, the connection is in what we refer to as the wireless bottleneck state, where the wireless WiFi link is the bottleneck for the entire end-to-end connection. In this state, Cupid mobile users can estimate and track the bottleneck capacity of the entire connection by measuring the WiFi physical control channel. The Cupid sender matches its transmission rate with the



(a) Throughput comparison among the four smartphones.

(b) RTT comparison among the four smartphones.

Figure 8: Cupid results while all remote host use Cupid.



(a) Throughput comparison among the four smartphones.

(b) RTT comparison among the four smartphones.

Figure 9: Cupid results while all near host use Cupid.

bottleneck capacity explicitly fed back by the mobile user, making almost full utilization of the capacity while causing minimal packet buffering in the network. On the other hand, if the capacity of the Internet bottleneck is smaller than the capacity of the wireless cellular link, the connection is in an Internet bottleneck state. This part is not covered in this paper and is considered as future work, with detailed discussions in § 6.

4.1 Connection Start: Line Rate Increase

On connection start, a Cupid sender i executes a line rate increase in order to approach a fair-share of the bottleneck capacity. By measuring the WiFi channel, each Cupid user knows the number of other users sharing the WiFi bandwidth, as shown in Figure 6. Cupid therefore calculates expected fair-share $R_i^{init}(t)$ using the total bandwidth available in the time slot and the number of active users (including the mobile itself):

$$R_i^{init}(t) = \eta \frac{\delta(C_i(t))}{N} \tag{1}$$

Where η is a constant close to 1 (e.g., 0.8), indicating the expected utilization rate of the entire air channel bandwidth. The function $\delta()$ represents the mapping from the underlying physical bandwidth to the throughput at the transport layer, as the actual bandwidth at the WiFi layer cannot be fully achieved due to the overhead of packet headers and control messages. $C_i(t)$ represents the current physical channel bandwidth at time t.

The condition for a user to exit the startup phase is either reaching the fair bandwidth point $\eta \frac{\delta(C_i(t))}{N}$ or experiencing no increase in rate within 3 RTTs.

4.2 Steady State: Congestion Avoidance

When the user i enters a steady state, the following rate adjustment should be performed:

$$R_i(t) = \begin{cases} \eta \frac{\delta(C_i(t))}{N} & \text{if } U(t) \geq \eta \ (MD) \\ R_i^{receiver}(t-1) + R_{AI}(t) & \text{if } U(t) < \eta \ (AI) \end{cases} \tag{2}$$

Here, $U(t)$ represents the airtime utilization at time t. If $U(t) \geq \eta$, it indicates that there is congestion and multiplicative decrease (MD) is needed. Therefore, sender i sets its rate to fairshare $\eta \frac{\delta(C_i(t))}{N}$. Conversely, if the utilization is insufficient, sender i linearly increases (AI) its sending rate. The increment is the previous instant's receiving rate $R_i^{receiver}(t-1)$ plus a linear increase $R_{AI}(t)$.

$$R_{AI}(t) = (\eta - U(t)) \frac{\delta(C_i(t))}{N} \tag{3}$$

$R_{AI}(t)$ represents the 1/N of the remaining bandwidth to ensure the overall airtime utilization.

Figure 7 illustrates an example of the Cupid algorithm. User 1 can set the rate to $\eta\delta(C_1(t))$ at the beginning, thereby occupying the entire airtime. Once User 2 appears, User 2 directly sets the speed to $0.5\eta\delta(C_2(t))$, and User 1 notices that the utilization $U(t)$ exceeds η, so it adjusts its rate to $0.5\eta\delta(C_1(t))$ as well. When User 3 finishes sending, Users 1 and 2 immediately detect the absence of a user and quickly adjust to each occupy half of the available bandwidth. At time 9, User 2 does not have enough data to send, so it releases idle bandwidth. At this point, User 1 linearly increases (AI) its sending rate to obtain the remaining bandwidth. Through the above steps, Cupid can rapidly, accurately, and fairly perform congestion control.

5 PRELIMINARY RESULT

We have developed a prototype of Cupid on top of the UDP and constructed a small testbed for preliminarily evaluating Cupid.

The client-side Cupid module receives measurements as input and communicates with the sender side through a commercial mobile phone connected to the host PC, as shown in Figure 6(b). Some key experimental results have been presented in Figures 3 and 4 in § 3.

In Figure 3(c), Cupid ensures fairness across all flows. In Figure 4(a), it can be observed that compared to Cubic and BBR, Cupid enables the throughput of the remote (victim) to not degrade, increasing from 0.5 Mbps to 41 Mbps, which show that Cupid effectively addresses the victim problem, improving victim throughput by up to 50 times. In comparison to UDP, Cupid does not compromise throughput at the near host, improving from 83 Mbps to 180 Mbps. In Figure 4(c), Cupid's RTT is also significantly lower compared to Cubic and BBR. Due to its ability to contend for bandwidth, Cupid's average RTT at the remote end can decrease from 2,000 ms to 375 ms, resulting in a 5X reduction in latency for weaker users. In comparison to UDP, which lacks accurate bandwidth estimation, Cupid achieves a similar RTT reduction from 500 ms to 375 ms.

We also conducted tests on the coexistence of Cupid with other algorithms. Figure 8 illustrates a scenario where Cupid is used at the remote end while various algorithms are employed at the near host. The results indicate that if the victim adopts the Cupid algorithm, it remains unaffected regardless of the algorithm used by others. However, the RTT may experience slight elevation compared to the scenario where all users utilize Cupid. Figure 9 illustrates a scenario where Cupid is employed at the near end while other algorithms are used at the remote end. Cupid reserves bandwidth for remote BBR and Cubic users, ensuring fair sharing among all users. However, if UDP is utilized at the remote end, the performance of Cupid at the near end also suffers.

In summary, if everyone uses Cupid, it ensures both high throughput and low latency. If various algorithms coexist, Cupid actively acquires bandwidth when it's being victimized, while it reserves bandwidth for the victims when it's not being victimized.

6 DISCUSSION & FUTURE WORK

Measurement accuracy: When mobile devices are far away from the WiFi router, measurements may become inaccurate due to signal attenuation in the wireless medium, causing devices to miss some signals. In such cases, calibration of the measurement results is necessary. Cupid plans to adjust the convergence speed and parameters to correct it.

The Internet bottleneck congestion and WiFi congestion: Cupid is designed to address congestion in WiFi networks, assuming congestion occurs in the air interface transmission. However, in real-world scenarios, congestion can also occur in the Internet bottleneck network. Therefore, Cupid also needs to determine the location of congestion and adjust congestion control accordingly.

More test scenarios: Other more detailed tests, such as comparing more congestion control methods, the Internet bottleneck congestion, mobility, parameter sensitivity analysis, etc., are left for future work.

7 RELATED WORK

Congestion Control: The work most similar to Cupid is PBE-CC [25], which focuses on congestion control in cellular networks. PBE-CC [25] is a congestion control protocol designed for 5G networks. It measures the control frames in 5G networks to achieve accurate congestion control.

Wireless routers can also provide congestion signals. ABC [10] observes the behavior of queues on wireless routers and utilizes ECN for congestion control. Zhuge [18] separates the wireless and wired components, allowing wireless routers to provide rapid congestion feedback to maintain low latency in queues. Cupid can also be deployed on routers, accurately placing rates on ACKs and sending them back to the sender.

Loss-based algorithms [8, 11, 14, 23] and delay(/rate)-based algorithms [4–6, 15] achieve poor capacity utilization when competing with concurrent loss-based algorithms [4, 23]. Sprout [24] utilizes packet arrival times to predict link capacity variations in the network path. Similarly, ExLL [20] adjusts the send rate based on packet arrival patterns and cellular bandwidth usage. Unlike these approaches, Verus [26] aims to learn a delay profile correlating target send window size with end-to-end delay. However, these algorithms solely relying on end-to-end statistics may suffer from rate estimation inaccuracies and sensitivity to network dynamics, as we have shown.

Airtime Fairness: The 802.11 Airtime Fairness Anomaly [12] is a well-known property of WiFi networks: if devices on the network operate at different rates, the MAC protocol will ensure throughput fairness between them, meaning that all stations will effectively transmit at the lowest rate. Despite various mitigation strategies proposed in the literature (e.g., [7, 13, 16, 22]), widespread real-world deployment of these solutions remains elusive, as evidenced by observations in § 3.1 showing lower throughput for near UDP hosts. The assumption above is that there is enough data to be batched. This paper also finds that TCP causes there to be not enough data to be batched, which leads to another kind of unfairness.

8 CONCLUSION

This paper first demonstrates the TCP victim problem in WiFi networks, where traditional TCP cannot acquire bandwidth. Then, the paper introduces Cupid, a new congestion control specifically designed for WiFi networks. Cupid accurately assesses congestion in wireless WiFi networks by measuring parameters such as airtime utilization, concurrency, and rates, enabling precise rate adjustments. Our results show that Cupid effectively addresses the victim problem, improving victim throughput by up to 50 times.

REFERENCES

[1] 2013. IEEE Standard for Information technology– Telecommunications and information exchange between systemsLocal and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications–Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz. *IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, and IEEE Std 802.11ad-2012)* (2013), 1–425. https://doi.org/10.1109/IEEESTD.2013.6687187

[2] 2021. IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN. *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)* (2021), 1–767. https://doi.org/10.1109/IEEESTD.2021.9442429

[3] 2023. IEEE Draft Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and

Physical Layer (PHY) Specifications Amendment: Enhancements for Extremely High Throughput (EHT). *IEEE P802.11be/D3.0, January 2023* (2023), 1–999.

[4] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical {Delay-Based} congestion control for the internet. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 329–342.

[5] Lawrence S Brakmo, Sean W O'malley, and Larry L Peterson. 1994. TCP Vegas: New techniques for congestion detection and avoidance. In *Proceedings of the conference on Communications architectures, protocols and applications*. 24–35.

[6] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: Congestion-based congestion control. *Commun. ACM* 60, 2 (2017), 58–66.

[7] Cisco. [n. d.]. *Air Time Fairness (ATF) Phase1 and Phase 2 Deployment Guide*. https://content.cisco.com/chapter.sjs?uri=/searchable/chapter/c/en/us/td/docs/wireless/technology/mesh/8-2/b_Air_Time_Fairness_Phase1_and_Phase2_Deployment_Guide.html.xml 2019.

[8] Sally Floyd, Tom Henderson, and Andrei Gurtov. 2004. Rfc3782: The newreno modification to tcp's fast recovery algorithm.

[9] Boris Ginzburg and Alex Kesselman. 2007. Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11 n. In *2007 IEEE Sarnoff Symposium*. IEEE, 1–5.

[10] Prateesh Goyal, Anup Agarwal, Ravi Netravali, Mohammad Alizadeh, and Hari Balakrishnan. 2020. {ABC}: A simple explicit congestion controller for wireless networks. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 353–372.

[11] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review* 42, 5 (2008), 64–74.

[12] Martin Heusse, Franck Rousseau, Gilles Berger-Sabbatel, and Andrzej Duda. 2003. Performance anomaly of 802.11 b. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, Vol. 2. IEEE, 836–843.

[13] Toke Høiland-Jørgensen, Michał Kazior, Dave Täht, Per Hurtig, and Anna Brunstrom. 2017. Ending the Anomaly: Achieving Low Latency and Airtime Fairness in {WiFi}. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. 139–151.

[14] Van Jacobson. 1988. Congestion avoidance and control. *ACM SIGCOMM computer communication review* 18, 4 (1988), 314–329.

[15] Cheng Jin, David X Wei, and Steven H Low. 2004. FAST TCP: motivation, architecture, algorithms, performance. In *IEEE INFOCOM 2004*, Vol. 4. IEEE, 2490–2501.

[16] Tarun Joshi, Anindo Mukherjee, Younghwan Yoo, and Dharma P Agrawal. 2008. Airtime fairness for IEEE 802.11 multirate networks. *IEEE Transactions on Mobile Computing* 7, 4 (2008), 513–527.

[17] Evgeny Khorov, Anton Kiryanov, Andrey Lyakhov, and Giuseppe Bianchi. 2018. A tutorial on IEEE 802.11 ax high efficiency WLANs. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 197–216.

[18] Zili Meng, Yaning Guo, Chen Sun, Bo Wang, Justine Sherry, Hongqiang Harry Liu, and Mingwei Xu. 2022. Achieving consistent low latency for wireless real-time communications with the shortest control loop. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 193–206.

[19] Mathijs Mortimer. 2018. iperf3 Documentation.

[20] Shinik Park, Jinsung Lee, Junseon Kim, Jihoon Lee, Sangtae Ha, and Kyunghan Lee. 2018. ExLL: An extremely low-latency congestion control for mobile cellular networks. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. 307–319.

[21] semfionetworks. [n. d.]. *MCS TABLE (UPDATED WITH 802.11AX DATA RATES)*. https://semfionetworks.com/blog/mcs-table-updated-with-80211ax-data-rates/ 2019.

[22] Godfrey Tan and John V Guttag. 2004. Time-based Fairness Improves Performance in Multi-Rate WLANs.. In *USENIX annual technical conference, general track*. 269–282.

[23] Kun Tan, Jingmin Song, Qian Zhang, and Murad Sridharan. 2006. A compound TCP approach for high-speed and long distance networks. In *Proceedings-IEEE INFOCOM*.

[24] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. USENIX Association, Lombard, IL, 459–471. https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/winstein

[25] Yaxiong Xie, Fan Yi, and Kyle Jamieson. 2020. PBE-CC: Congestion control via endpoint-centric, physical-layer bandwidth measurements. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 451–464.

[26] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. 2015. Adaptive Congestion Control for Unpredictable Cellular Networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 509–522. https://doi.org/10.1145/2785956.2787498