

Review 1

There are several issues with the analysis in Section V.

Eq. 1: Why are $N-1$ connections used for UDP: N is the number of TCP connections. Is it assumed that it is also the number of UDP clients as shown in Fig. 2? Even if that is the case, shouldn't it be $2N$?

Total N clients, one of them is the game server. The other $N-1$ clients connect to the game server by UDP. Therefore, only $N-1$ UDP traffic.

Equation 3 is a restatement of 1.

Yes, we can remove equation 3

Eq. 4 seems to be an incorrect application of Little's Law: the denominator should be bit arrival rate, not service rate. This error may not have made significant impact on the results because, in steady state, the arrival rate was probably close to the service rate.

Also, if C is the number of segments, then shouldn't that be used in computing TCP bits?

We need to add "The arrival rate close to the service rate in steady state".

C is not the number of segments. It's the number of bits.

Equation 5: does not take into account retransmissions and possible send rate reductions; even if the TCP congestion control is turned off on the wireless part of the split connection, a loss will still result in back off and retransmission owing to the congestion control on the wired part of the connection.

Let's remove equation 5. Equation 5 only works when no buffer loss in AP. If packet lost in AP, the throughput downlink and uplink is unfair. Also, the proxy in this paper doesn't maintain the end-to-end semantic, equation 5 doesn't consider the loss on wire link.

2. The ramifications of split-TCP connections are not addressed.

The connection set up delays are increased.

It exposes switch-to-controller communication link to denial of service attacks.

This is not a security problem only for this paper. It's a problem for all SDN controller which reacts to the new flows. This problem can be solved by placing business firewall between the controllers to the switch.

If the wireless portion of the connection is set up first and if the controller refuses the connection, there needs to be a way to clean up the connections already set up.

Connection termination also takes more work and makes switch ports unavailable during this time.

Need to analyze the additional work the switch needs to do to change TCP headers in split-TCP connections. This can have a significant on performance when the traffic is high.

"the wireless portion of the connection is set up first". It won't be happened in the scenario described in this paper.

"This can have a significant on performance when the traffic is high" This paper runs the experiment with standard OpenFlow protocol. The controller inserts flow tables to change TCP headers. It's already been the most efficient approach for SDN. If it has performance problem, it will be the SDN platform problem.

On the positive side, the paper presents an implementation and experimental results. Fairness index data is helpful. However, the uneven TCP throughputs for traditional case (in TCP only scenario) should be explained. It may have been caused by the WiFi portion of the connection. This should have been investigated more carefully. Are the experiments done multiple times and results averaged? What are the confidence intervals?

Traditional TCP provides fairness when the RTT is same. Also it takes time to adjust to the fairness state which can be broken by packet loss of any flow. Therefore, it doesn't work well on long RTT and different RTT.

We ran repeated testing for TCP flows in ns3. But only one test on real switch for gaming traffic. All the gaming traffic ran for long time and we calculate the average value.

Review 2

No detailed Comments, Main Weaknesses

The system was tested on a 802.11b wifi which is quite outdated. It would be interesting to see wether such improvements still hold with 802.11n or ac type of networks. Also there is a claim that the wireless link is the bottleneck, although with 802.11n and ac this might not be the case anymore. It is not clear how this different scenario would change the results. This somehow limits the credibility of the results.

This paper assumes the wireless link is the bottleneck. If beak this assumption, it won't have large delay for both TCP and UDP traffic. The proposed proxy is useless.

We can try to test 802.11n on Ns3. but the described scenario has to change from video stream traffic to be bulk file transfer. Otherwise, the 802.11n won't be the bottleneck.

Review 3 (Reviewer D)

Not sure what this means "The background TCP video and UDP gaming traffic share the same queue. " They are technically using different queues at AP. But they share the same wireless channel, and do have contentions with each other.

"Understanding TCP fairness over Wireless LAN" publish in infocom2003 assumes only one queue in the wifi station for all clients.

2. Based on the analysis of formula (5), can you present the theoretical results along with the exp results?

Equation 5 will be removed

3. There's a large amount of works on bandwidth sharing between TCP and UDP. Basically make UDP flow mimic TCP behavior, which was called TCP friendly. Can you also comment that on related works and compare the pros and cons?

UDP and TCP are separated into different queue described in the paper " On class-based isolation of UDP, short-lived and long-lived TCP flows". While, this paper don't need to create a separated queue for UDP. All traffic go through the same queue.