

Channel Capture Effect in 802.11: A Study

Joe Di Natale, Sarah J. Andrabi

University of North Carolina, Chapel Hill

ABSTRACT

When multiple devices try to send data through the medium, they do so taking turns—that is the expected behavior. Sometimes, however, some of these devices tend to send more data than they should be as per the specified turns, or even worse they keep sending their data till they are done and don't let the other devices send till then. This behavior is known as the channel capture effect, observed first for wired medium. In this paper we conduct a set of experiments to observe the channel capture effect in wireless 802.11. We study the effect of the number of clients, their various configurations and distances from an access point, on the observed behavior. We also observe the short-term unfairness due to the channel capture effect.

I. INTRODUCTION

Channel Capture Effect is the phenomenon where one user of a shared medium "captures" the medium for a significant time [2]. Other users that try to access the medium during that duration are denied access. Capture Effect was first observed in wired networks using CSMA/CD in Ethernet [2]. In CSMA/CD nodes sense whether the medium is free or not to be able to sense the data. Thus when multiple nodes try to send data (capture the medium) at the same time, a collision occurs. Now if one of the nodes captures the medium, and any other node tries to send data, a collision occurs and the other nodes back off, they wait a random amount of time. Now once the 'active' node has finished sending data, it again senses the medium, and since the other nodes had backed off, this node again 'captures' the medium. Now when the other nodes' back-off time is over, they again sense the medium—they find the medium busy, and again back-off a random amount of time, which is greater than what it was before, and again the whole cycle repeats till the 'active' node is done transmitting. Thus one node continues to win the link, thus introducing short term unfairness. It is still long-term fair as every station has an opportunity to "capture" the medium once one station is done transmitting [2].

A significant problem for all adhoc wireless networks is the poor performance displayed by the transport protocol over a number of different MAC protocols [3], [4]. Critical to the transport protocol is the performance of the MAC protocol in terms of fairness and delay. One characteristic of poor MAC performance is this 'channel capture'. In this paper we see if the capture effect can be observed in wireless 802.11b networks. We try different configurations for a two client system, a three client system and then try increasing the number of clients to up to four clients for one access point, a server which serves as the receiver for all the clients and a sniffer. Our measure of interest is the successive number of frames sent by a client.

The rest of this paper is organized as follows. Section II describes the testbed for the experiment. Section III presents the challenges encountered during the project. Section IV investigates the traces obtained from the sniffer and the server and the clients and presents the analysis. We present future work in section V and finally the conclusion in section VI.

II. EXPERIMENTAL TESTBED SETUP

For the purpose of our study, we use a server, clients, a sniffer and a Linksys WRT350N access point supporting 802.11 n/g/b and running DD-WRT firmware. The server and clients all run Windows 7 and 8 OS, with wireless hardware support for 802.11b, for which we collect our results. The server and the clients both run jperf, which is the GUI front-end for iperf. The main advantage of using iperf is that it can be used with any type of network, both wired and wireless, thus allowing flexibility of measurements taken. Iperf was originally developed by NLANR/DAST as a tool for measuring maximum TCP and UDP bandwidth performance. Iperf allows the tuning of various parameters and UDP characteristics. Iperf reports bandwidth, delay jitter, datagram loss [5]. Iperf has a client and server functionality, and can measure the throughput between the two ends, either unidirectionally or bi-directionally [6].

In our test beds, the server has a wired connection to the access point; this is the only wired connection in the entire test bed; all the clients and the sniffer use wireless 802.11b. The Sniffer runs Wireshark and Aircrack-ng, as will be explained in the next section. Aircrack-ng is a network software suite consisting of a detector, packet sniffer, WEP and WPA/WPA-PSK cracker and analysis tool for 802.11 wireless LANs [7]. Both Wireshark and Aircrack-ng require wireless network interfaces that support monitor mode. Since WinPcap does not support monitor mode, the sniffer runs on a machine running Ubuntu 12.04, as Linux supports monitor mode. From the Aircrack-ng suite the tools of interest to us are airmon-ng—to enable monitor mode on wireless interfaces [8]. Pcap consists of an API for capturing network traffic. For our analysis of traces we use the C# wrapper for the WinPcap library called SharpPcap to extract the measures of interest from the traces collected by Wireshark.

For the collection of traces, we start with a two client scenario, with three different configurations, in one of which the clients are equidistant from the access point and in the second one, Client 1 is closer to the Access point than Client 2, and in the third one Client 2 is further away as shown in Figure 1. We then take the measurements again with three clients, with two different configurations, in the first one the clients are equidistant from the access point, in a T-configuration and the other is a line configuration, as can be seen in Figure 2. Finally we take another set of measurements for a 4 client scenario, as shown in Figure 3, in one of the configurations the clients are equidistant from the access point, and the other one is a line configuration. In all of these configurations, the server and the sniffer are placed right next to the access point. All these traces were collected in the Networks Lab, in the Computer Science Department at UNC, Chapel Hill. We also collected a set of traces for two clients in an apartment complex where a lot of other Wireless Access points were present, and hence a lot of interference. While in the networks lab there were only 4 other wireless networks visible.

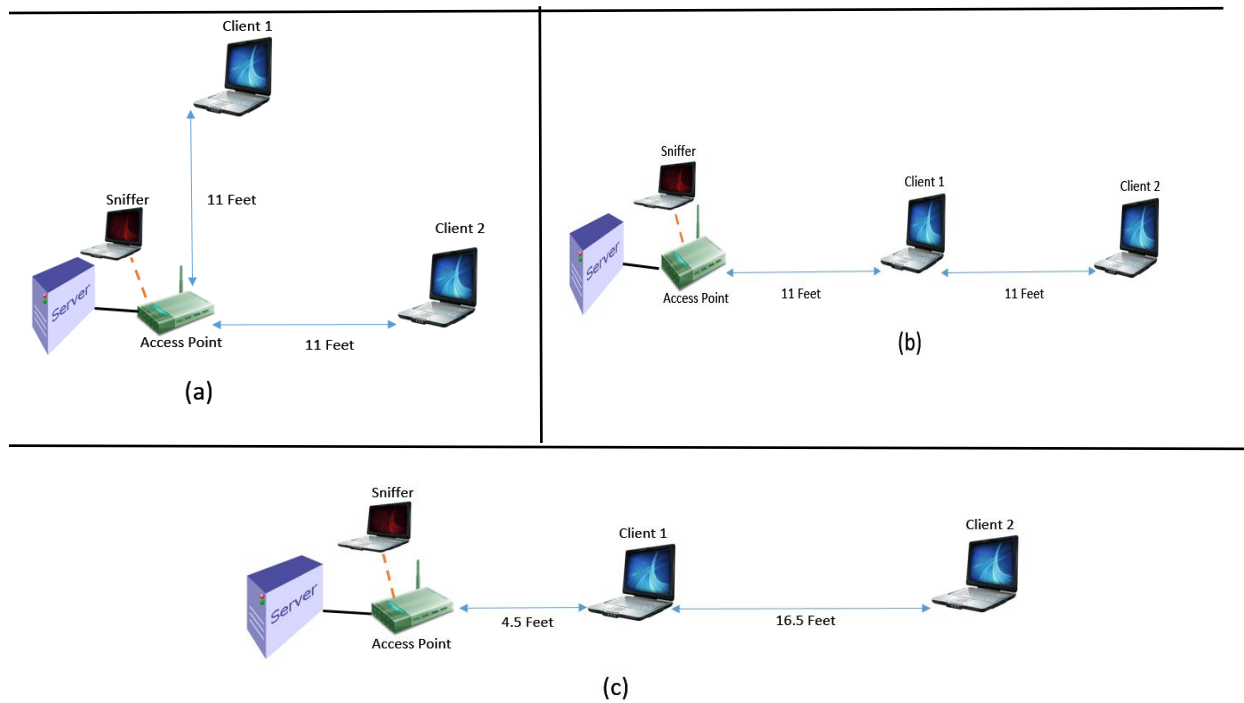


Figure 1. Two Client setups (a) Two Clients, equidistant from the Access Point, (b) Two Clients, in a line configuration and (c) Two Clients, in a line configuration, with one client further away.

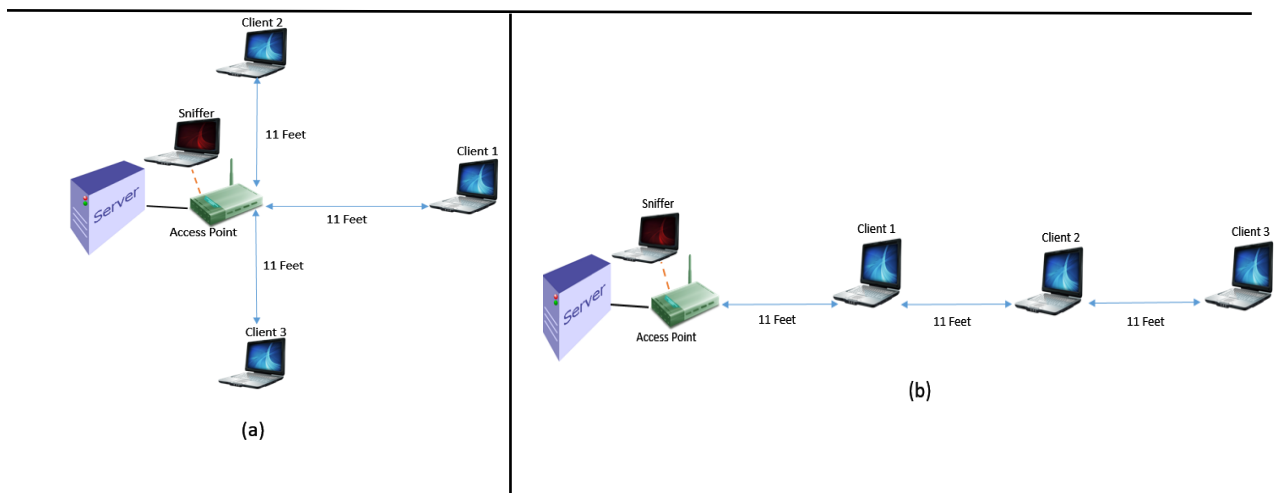


Figure 2. Three Client setups (a) Three Clients, equidistant from the Access Point in a T-configuration, (b) Three Clients, in a line configuration

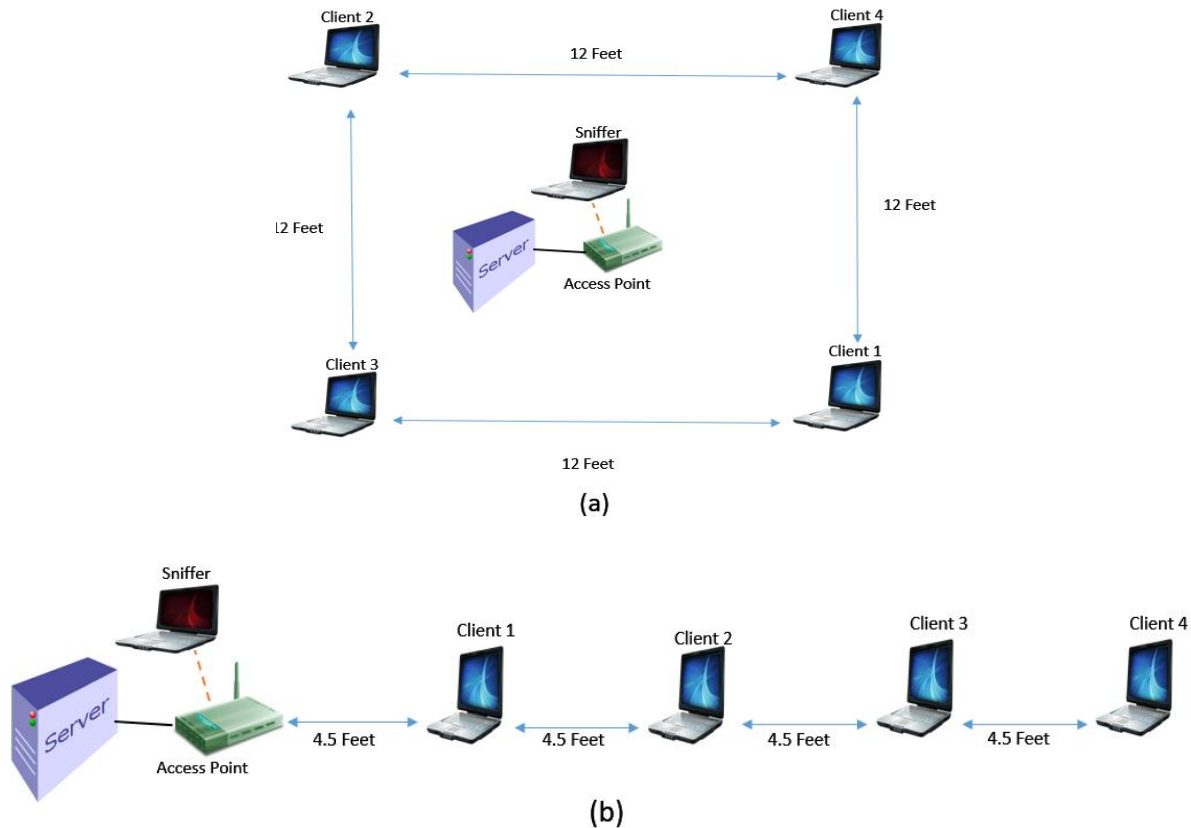


Figure 3. Four Client setups (a) Four Clients, equidistant from the Access Point, in a square configuration with the AP at the center of the square , (b) Four Clients, in a line configuration

III. CHALLENGES FACED

The initial challenge faced was obtaining hardware to perform our experiments since there is no wireless test bed in the Networks lab. At the beginning of the semester, we had approximately 10 volunteers. Once we were ready to perform our experiments near mid-semester, it became difficult to coordinate volunteers (many of our volunteers could not part ways with their laptop due to assignment/work deadlines). We were only ever able to test four clients on the same network at once.

The second challenge was properly setting up a wireless sniffer. WinPcap (Windows Pcap) does not support monitor mode which is required to do full 802.11 packet sniffing, but is available in some other libpcap libraries available on other operating systems such as Linux [10]. Once Linux was running on a laptop which had hardware that supported monitor mode (an old Dell Inspiron 1420), new wireless card firmware and drivers had to be installed [11]. Even with the correct drivers and firmware, there were still issues including only being able to see beacon and probe requests. The solution to these issues was installing a suite of tools called Aircrack-ng which provided tools including its own tracing tools and a tool to enable a monitoring wireless interface [9]. Leveraging some of the functionality in Aircrack-ng, we were able to verify that the sniffer was working as intended.

Finding a good experimental space was also a challenge. We had to get approval to use a wireless hotspot in the Networks Lab to carry out our experiments. We also opted to perform our experiments on campus as to not inconvenience our volunteers as well. In the end, all experiments were carried out in the Networks Lab even though the space was limited.

Finally, once we got our test bed running and ran our various configurations, it became apparent that different hardware had different transmit power thus affecting the signal seen by the access point. The reason for the differences is because we had heterogeneous hardware which was provided by volunteers and we could not configure the systems how we wished. We opted not to mitigate the effect of different hardware and left that for future work.

IV. EVALUATION AND ANALYSIS

In this section we first present the traces obtained from the testbed, and then present an analysis of the results obtained.

For every setup as shown in Figures 1, 2 and 3, the clients and the server run jperf, and the sniffer uses Wireshark along with Aircrack-ng to collect the traces. We use UDP as the transport layer protocol, with a bandwidth of 1 MBps and a packet size of 1400 Bytes. Each client transmits data to the server for 60 seconds and all the clients send at the same time. The Report interval for jperf is set to 1 second i.e. it reports what has been transmitted, at the client side, and what has been received at the server side. We use the 2.442 GHz channel for our wireless 802.11b network.

All traces were obtained using Wireshark. We filtered the traces so only the UDP traffic we generated was visible. We then used a custom written C# application backed by the SharpPCap library (a wrapper around the Pcap and WinPcap libraries) which generated statistics such as average, minimum, and maximum successive frames sent per client as well as average, minimum, and maximum signal strength per client. Management, beacon, and probe frames were excluded because these are considered overhead and the experiment was focused strictly on data packets. We did not distinguish between retries and regular transmits because the client had to recapture the medium to attempt a retransmit.

Figure 4(a) shows Client 1, transmitting data for the configuration shown in Figure 2(b). Similar results were obtained on the other two clients. Figure 4 (b) shows the total data and the jitter on server receiving data from all the three clients.

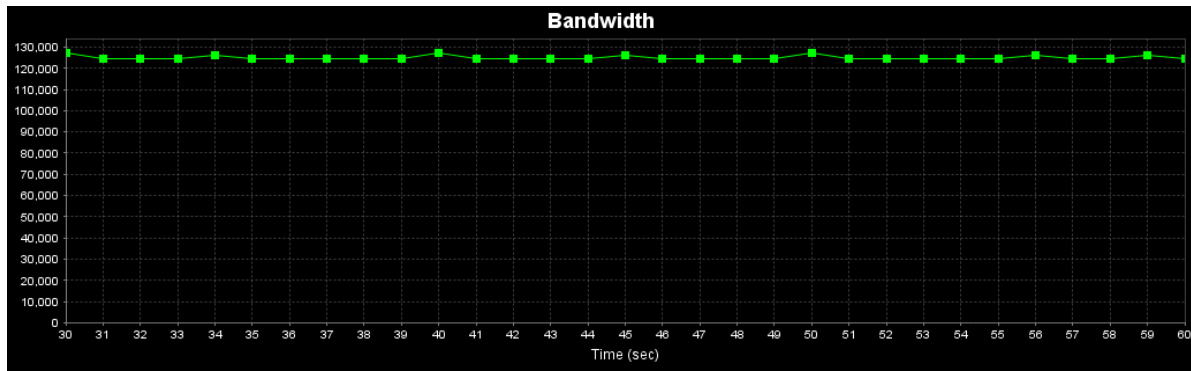


Figure 4(a)—Client 1 sending data to the server, in the presence of two other clients, in a line configuration setup. This shows the number of bytes sent on the y-axis and the time (in seconds) on the x-axis, obtained from jperf.

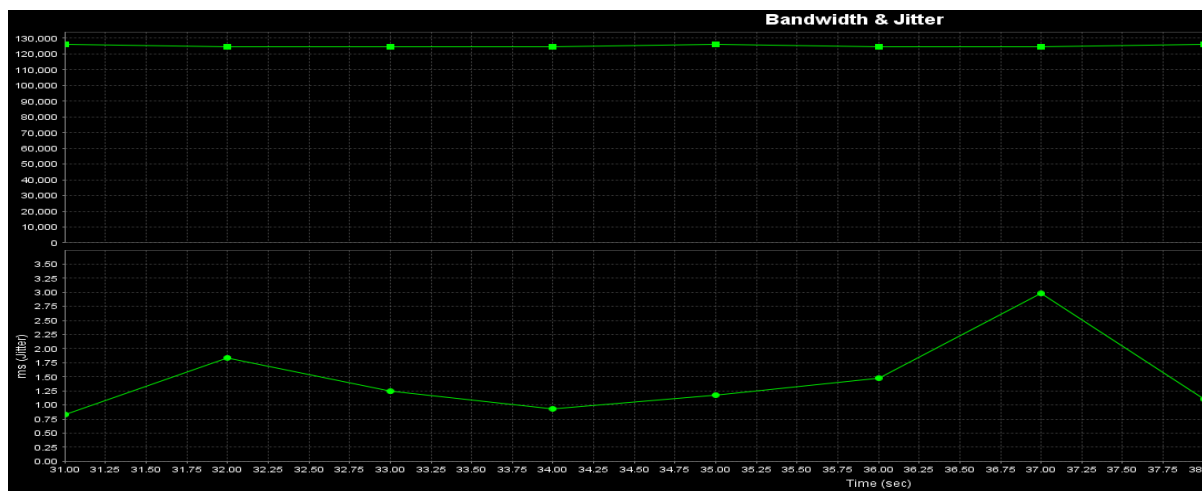


Figure 4(b)—Server receiving data from three clients, in a line configuration setup. This shows the number of bytes received and the jitter on the y-axis and the time (in seconds) on the x-axis, obtained from jperf.

We obtained the number of successive UDP Frames sent by each client during the duration of the entire transmissions from all the clients. Figure 5(a) shows the plot obtained for the configuration obtained in Figure 1(b) and Figure 5(b) shows the plot obtained for the configuration in Figure 1(c).

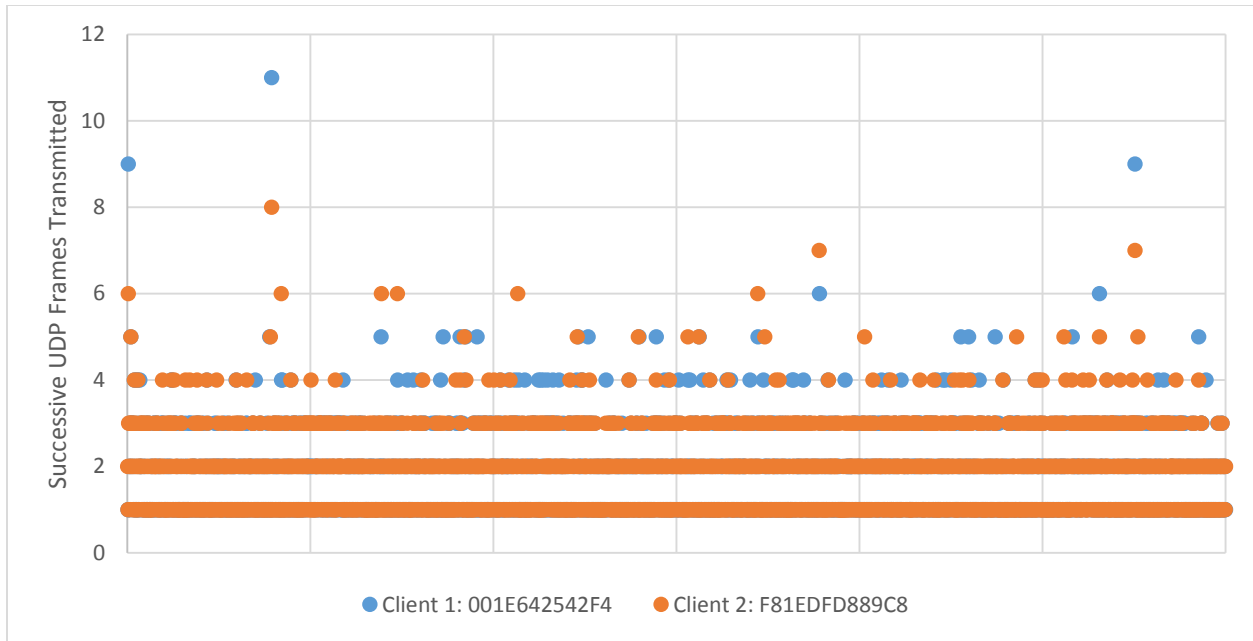


Figure 5 (a)—Successive UDP Frames sent by two clients equidistant from the access point configuration.

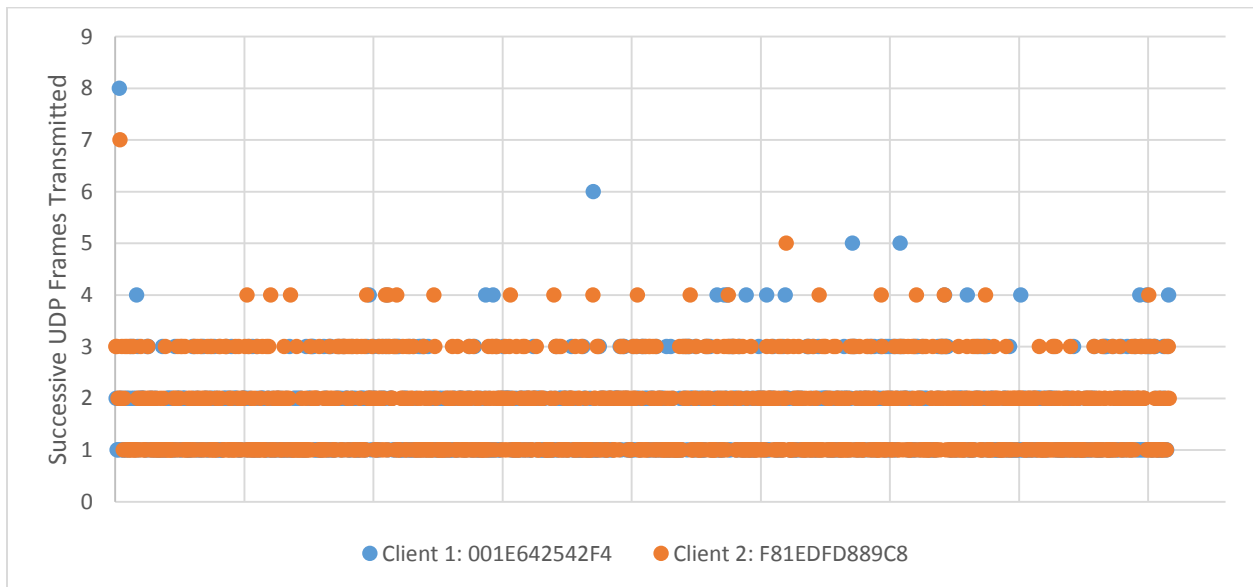


Figure 5 (b)—Successive UDP Frames sent by two clients at unequal distances from the access point in a line-configuration.

In both these configurations, there are two clients, but their distances from the access point are different in the configuration for Figure 1(c). As can be seen from the plots, the client which is further away from the access point sends a fewer number of successive frames. This is despite the fact that Client 2 has better hardware and consequently more transmission power than Client 1. In Figure 5(a), it can be seen that both the clients are capturing the medium, thus avoiding any long term unfairness.

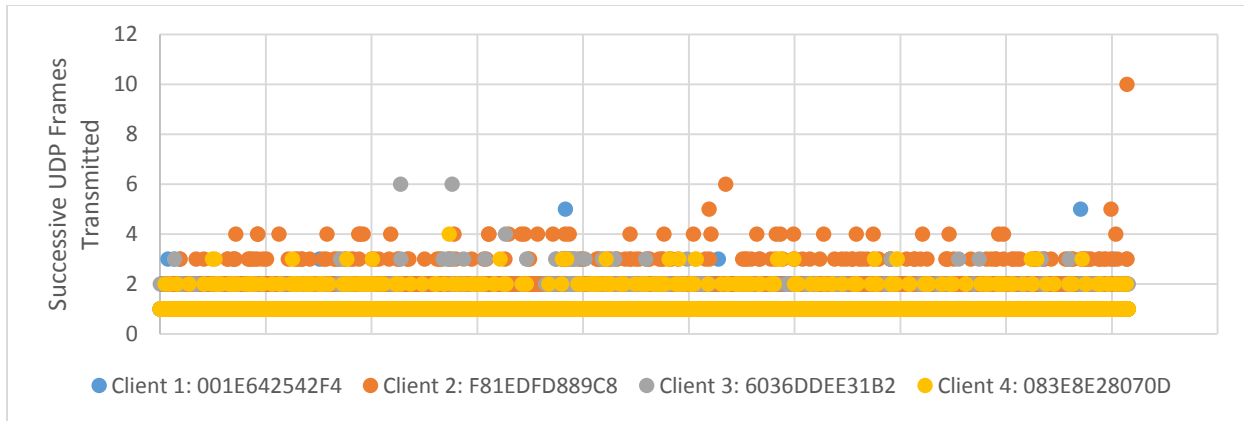


Figure 6(a)—Successive UDP Frames transmitted by four clients equidistant from the access point, for a part of the transmission.

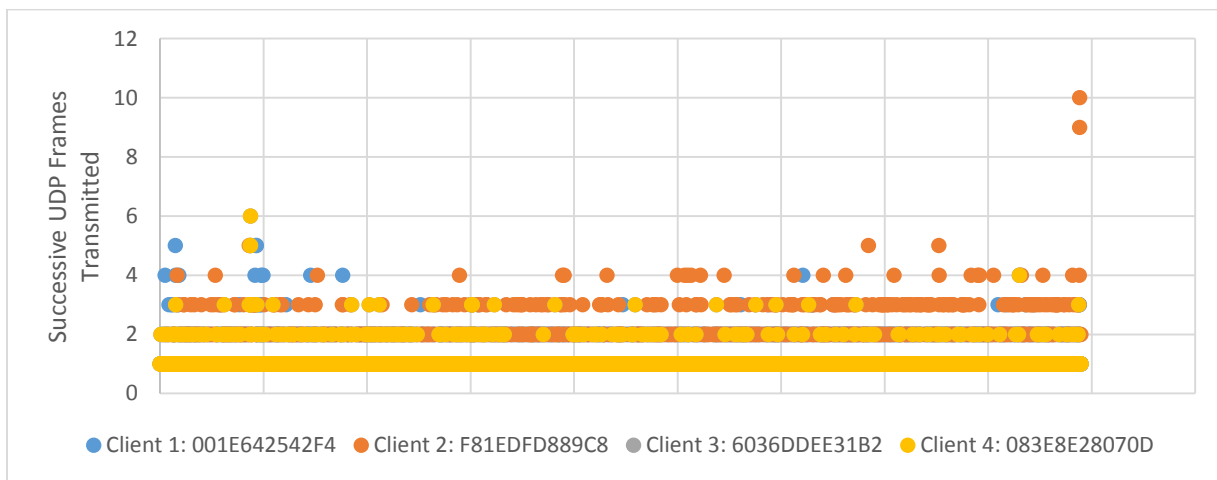


Figure 6(b)--Successive UDP Frames transmitted by four clients at different distances from the access point, in a line configuration, for a part of the transmission.

Figure 6(a) and (b) show the plots for four equidistant clients, in different configurations pertaining to Figure 3(a) and 3(b). As can be seen, that apart from the number of clients affecting the behavior of the successive number of frames sent by the clients, the distance also affects it. As can be seen, the number of average successive frames sent is lesser for the clients further away from the access point as compared to the clients which are closer. For Figure 6(b), Client 1 is the closest to the access point and the average successive frames sent by it is 1.27, while as Client 2 which is after Client 1 sends 1.358 average successive frames, client 3 sends 1.0489 and client 4 which is the farthest send an average of 1.07 successive frames. As this data indicates—the first two clients which are closer, send more successive number of frames and the ones after that send lesser, than the ones that are closer. The fact that Client 2 sends more than Client 1 is due to the discrepancies in hardware, as Client 2 has better transmission than Client 1 due to better hardware.

Also another thing to note is that as the number of clients increases the number of successive frames sent also decreases, as can be seen from Figure 5 and 6. As the number of clients increases, more clients are able to capture the medium in a fair way. Also as the distance between clients and the access point increases, the clients further away capture the medium for a lesser successive time, as compared to the ones closer to the client.

Also it is generally the case that when one client sends more successive frames, then the other clients and also the first client, sends fewer frames. There is no abrupt change like if client 1 sent 7 successive frames then in the absolute next moment Client 2 will also send around 7 frames. This kind of behavior is not present, which again emphasizes the fact that even though the channel capture effect is observed, it is not long term unfair.

V. FUTURE WORK

For future work, we would like to redo this experiment with homogenous hardware that is not provided by volunteers. We would preferably like to have hardware that we can tweak and change as needed. Ideally, we would run machines with a Linux distribution and MadWifi drivers so we can control various factors such as transmit power and ensure all hardware runs consistently.

Another factor we would like to vary in future iterations of this experiment would be the version of 802.11 that is used. Due to hardware limitation, we only were able to test 802.11b for this experiment. In the future, we would like to see if the channel capture effect is visible and attempt to analyze it on 802.11g and n networks.

If we are able to redo the experiment with homogenous hardware, we would also like to find a bigger, open area to conduct our experiments. Since all our experiments were done in the Networks Lab, we could only take one trial per configuration because there simply was not enough room to spread out in the lab. If we are able to conduct the trials in an open area, we can vary the distances in the various configurations and get a better idea of how distance impacts the Capture Effect.

We would also like to come up with some way to incorporate management frames into analysis since they were considered overhead in terms of this experiment. We would like to see if the management frames are causing the bursty behavior. Another possibility is that clients are sending many more frames than we are aware of since we strictly considered data frames.

Another side effect of the channel capture effect is channel idle time [2]. Due to time constraints, we did not analyze potential idle times as a result of the effect. In future work, we would like to define what idle time is and attempt to measure how much idle time the effect creates, if any.

VI. CONCLUSION

Our experiments can help us conclude a few things. In 802.11b networks, there is clear short term unfairness between clients. This unfairness can be attributed to the bursty behavior exhibited as clients capture the medium. Even though this short term fairness is present, it does not create long term unfairness. Overall 802.11 seems to be long term fair.

There is also clear indication that the number of clients connected to a single access point has an impact on the number of successive frames sent from a single client. Our experiments show that as the number of clients increase, there is a lesser chance for high number of successive frames to be sent. We believe this has to do with how random back off works in 802.11 networks. The more clients added to the network, the more likely it is for a client to select a smaller back off value. If a client does select a smaller back off, then there is actual competition for the link which can dramatically reduce the channel capture effect.

Our experiments also show that there does appear to be a trend involving distance and signal strength. From our trials, it appears that if a client has higher signal strength than the rest of the clients on the network, the client can win the medium repeatedly which allows for it to send more successive frames. Similarly, the less distance between a client and the access point, the stronger their signal, therefore the higher chance of that client winning the medium. The results of our trials cannot confirm that this is always the case, but they appear to suggest that signal strength (and by proxy distance) plays a major role in the capture effect.

REFERENCES

- [1] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz. *Unfairness and capture behaviour in 802.11 ad hoc networks*. IEEE International Conference on Communications, 1:159 –163, 2000
- [2] Channel Capture Effect, Wikipedia
- [3] M Gerla, K Tang, and R Bagrodia. *TCP performance in wireless multihop networks*. In 2nd IEEE Workshop on Mobile Computing Systems and Applications, volume 1, pages 41 -50. IEEE, 1999.
- [4] G Holland and N Vaidya. *Analysis of TCP performance over mobile ad-hoc networks*. In Mobicom 99, Seattle, 1999.
- [5] iperf, Google Code, <https://code.google.com/p/iperf/>
- [6] Iperf, Wikipedia, <https://en.wikipedia.org/wiki/Iperf>
- [7] Aircrack-ng, <http://en.wikipedia.org/wiki/Aircrack-ng>
- [8] Documentation Aircrack-ng, <http://www.aircrack-ng.org/documentation.html>
- [9] Aircrack-ng, Wikipedia, <http://www.aircrack-ng.org/doku.php>
- [10] WLAN (IEEE 802.11) capture Setup, <http://wiki.wireshark.org/CaptureSetup/WLAN>
- [11] b43 and b43legacy, <http://wireless.kernel.org/en/users/Drivers/b43>