Introduction:

Welcome to my lab report. This report details my observations for the Ns3 lab assignment under different rubrics. For this simulation, I have used **NS-3.36.1** as the variant, as the previous versions had some problems with installation in my wsl. Importantly, to run a code, this compiler only accepts ./ns3 instead of ./waf, but I believe the code is backward compatible and should be fine to run in a waf environment too. I also attached pcap files instead of CSV files, as for me pcap files told a lot more story about the system and also did not take as much time to run a code as a gnu plot takes, which i tried but took tremendous amounts of time and hence I resorted to pcap files.

Also, the format of the files within different folders are as follows:

CaseA_E1 and CaseB_E1 folders: node0_pcap_**number-nodes**_0-0.pcap, where 'number-nodes' indicates the no of nodes in the system eg: node0_pcap_9_0-0.pcap file indicates a 9 node system.

CaseA_E2 and CaseB_E2 folders: node0_pcap_**datarate**_0-0.pcap, where datarate indicates the link rate of the system for each file.

Explanation of my Scenario:

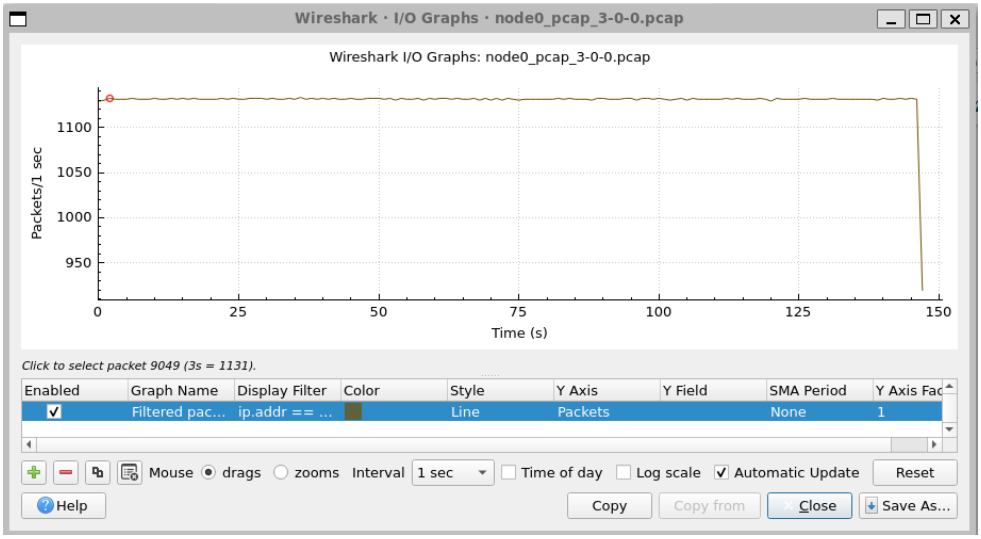
The code essentially implements an **ad-hoc 802.11** scenario employing a simple Friz's path loss model, where a no of nodes (N) transmit at a constant rate (CBR traffic) to a single node (node 0) using UDP (a grid based no mobility configuration). Node 0 essentially acts as an UDP packet sink to collect all the packets for a **simulation time of 147 seconds**, and also logs all the data as a pcap file for us to deeply analyze. I used ad-hoc mode primarily because it is a simpler variant of an AP based system and thus allowed me to deeply analyze the effects of 802.11 CSMA parameters only and not worry about routing layer and its associated effects into the system at all. Let us proceed to the rubrics.

Rubrics:

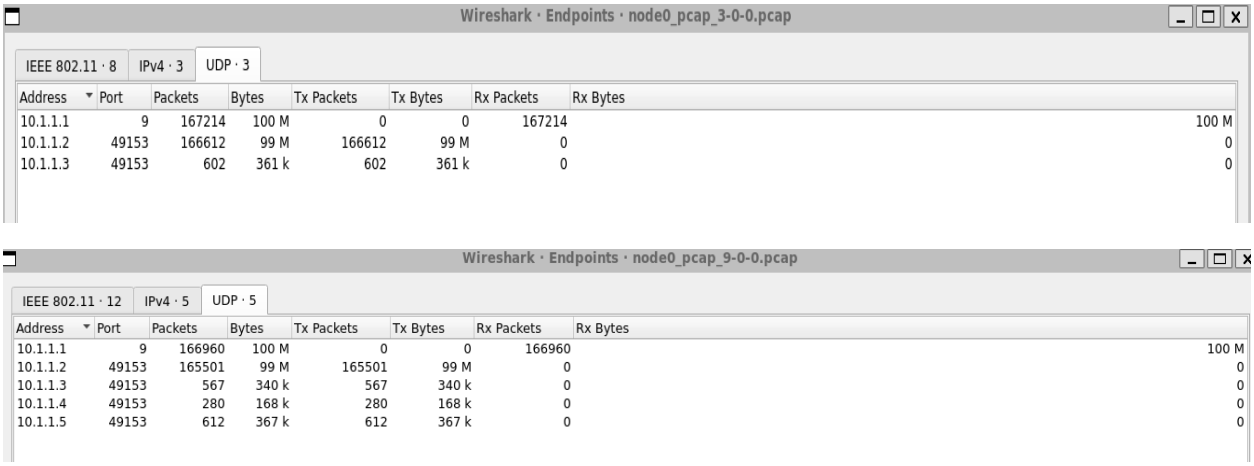
Case A E1:

For this case, I have used a CWmin of 1 and CWmax of 1023 and was able to modify the window size with the help of txop library. I have set a link rate of 11 Mbps. By varying the no of nodes as 3, 5, 7, 9 and 10, pcap files had been yielded for every number (located in CaseA_E1 folder). Here I have attached a sample throughput snapshot for a 3 node system over time (can be accessed from a pcap file, by **applying a display filter and then**

proceeding to Statistics -> I/O graphs). We can view the throughput in y axis as being displayed in packets/sec vs time. Wireshark also allows us to display multiple nodes's throughput by using the display filter.



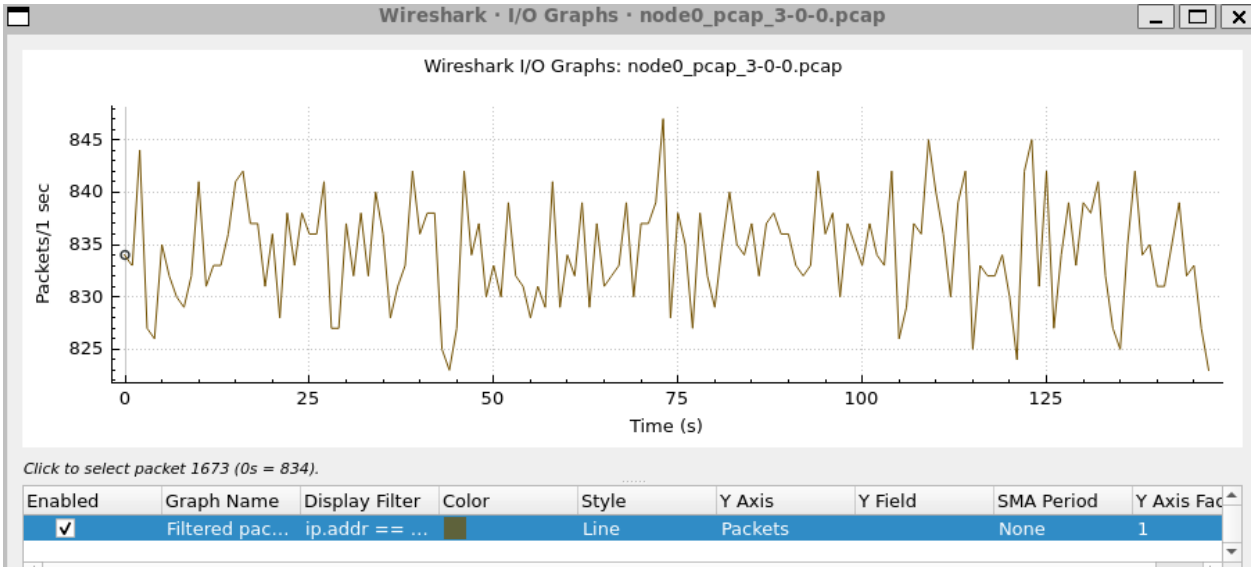
The first observation has been that as the number of nodes increases, the **average throughput decreases**. Another important observation is that the **per-node throughput is not the same and varies tremendously for every node** depending on the distance it is from the receiver node (node 0) and is very lopsided, I believe the main reason for this behavior is that the very low contention windows topple the behavior of the system in such a way that the lower distance nodes benefit greatly and the higher distance nodes have a terrible service, outweighing the higher contention flexibility, which is till 1023. PFA a snapshot capturing this scenario for a 3 node and a 9 node system (**Statistics -> Endpoints**).



You can clearly see the difference in the no of packets sent to the receiver (which is 10.1.1.1) by different nodes, and **this worsens as we increase the no of nodes,** as we can see for a 9 node system, there are some nodes that did not send any packet at all for the whole duration of 147s!, and the pattern is that the more distant nodes collide much more and thus tremendously fail. Coming to collision, I will cover it as the last rubric subsequently in this report.

Case B E1:

This case also holds an interesting spot in this report. Nearly everything is same in the code as in previous case with the exception of CW size. Here, the CWmin is 63 and CWmax is 127. I have attached a snapshot of the traffic across a 3 node network below.



The blatant thing I noticed compared to the previous case is the traffic here is extremely mercurial in nature, a lot of ups and downs, this should tell us about how every node transmits their packet and about the **Fairness** of this system.

Coming to the main point, **I noticed that the average throughput of the system decreased slightly as n increases, but not the net throughput of node 0, it infact had an increase in the number of packets it received for the fixed duration of 147s! For increasing n.** The reason was laid in my second observation when I navigated to the end points in the pcap file, PFA a snapshot for a 3 node and a 9 node system .

Wireshark · Endpoints · node0_pcap_3-0-0.pcap

IEEE 802.11 · 8 | IPv4 · 3 | UDP · 3

Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
10.1.1.1	9	123466	74 M	0	0	123466	74 M
10.1.1.2	49153	63848	38 M	63848	38 M	0	0
10.1.1.3	49153	59618	35 M	59618	35 M	0	0

Wireshark · Endpoints · node0_pcap_9-0-0.pcap

IEEE 802.11 · 20 | IPv4 · 9 | UDP · 9

Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
10.1.1.1	9	148250	88 M	0	0	148250	88 M
10.1.1.2	49153	23539	14 M	23539	14 M	0	0
10.1.1.3	49153	21004	12 M	21004	12 M	0	0
10.1.1.4	49153	19556	11 M	19556	11 M	0	0
10.1.1.5	49153	18084	10 M	18084	10 M	0	0
10.1.1.6	49153	16958	10 M	16958	10 M	0	0
10.1.1.7	49153	16277	9766 k	16277	9766 k	0	0
10.1.1.8	49153	16340	9804 k	16340	9804 k	0	0
10.1.1.9	49153	16492	9895 k	16492	9895 k	0	0

We can clearly see that the throughput distribution among different nodes is relatively fair for this window size. I believe this fairness had caused a net increase in throughput for the receiver, but still had an" average" decrease. This means this window size really is optimal for the system as a whole compared to Case A's window size.

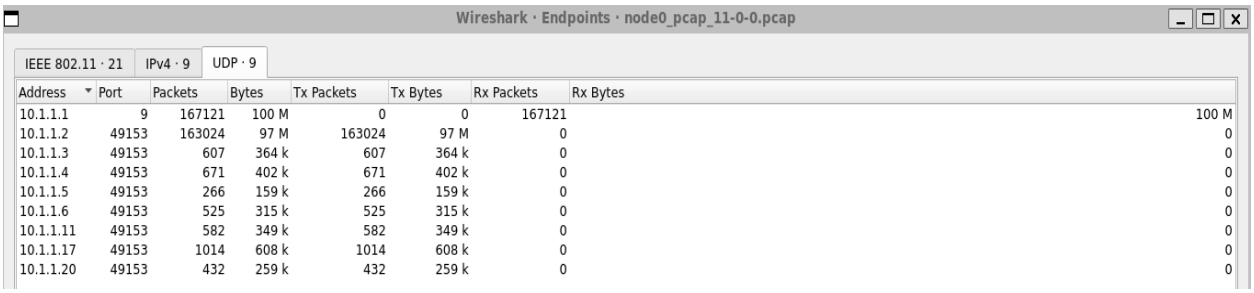
Case A E2:

In this case, I have fixed the number of nodes to 20, and varied the maximum data rate that a node can in this manner {1,3,5,7,11} Mbps. The trend is, the average throughput seemed to slightly increase with link rate when we compare 1 and 3 mbps, but stayed constant for 5, 7 and 11 mbps values. I presume that this is due to the fact that for the same success time, having a higher throughput allows you to send more packets than a lower throughput case, but the overall collision saturates the system to an extent that the utility is not exactly linear but more stooped. I have attached the endpoint snapshots for a 1mbps and a 11 mbps link here.

Wireshark · Endpoints · node0_pcap_1-0-0.pcap

IEEE 802.11 · 29 | IPv4 · 12 | UDP · 12

Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
10.1.1.1	9	158838	95 M	0	0	158838	95 M
10.1.1.2	49153	35877	21 M	35877	21 M	0	0
10.1.1.3	49153	35828	21 M	35828	21 M	0	0
10.1.1.4	49153	32892	19 M	32892	19 M	0	0
10.1.1.5	49153	3890	2334 k	3890	2334 k	0	0
10.1.1.6	49153	17971	10 M	17971	10 M	0	0
10.1.1.7	49153	2457	1474 k	2457	1474 k	0	0
10.1.1.8	49153	1865	1119 k	1865	1119 k	0	0
10.1.1.10	49153	9323	5593 k	9323	5593 k	0	0
10.1.1.11	49153	9820	5892 k	9820	5892 k	0	0
10.1.1.13	49153	1484	890 k	1484	890 k	0	0
10.1.1.18	49153	7431	4458 k	7431	4458 k	0	0

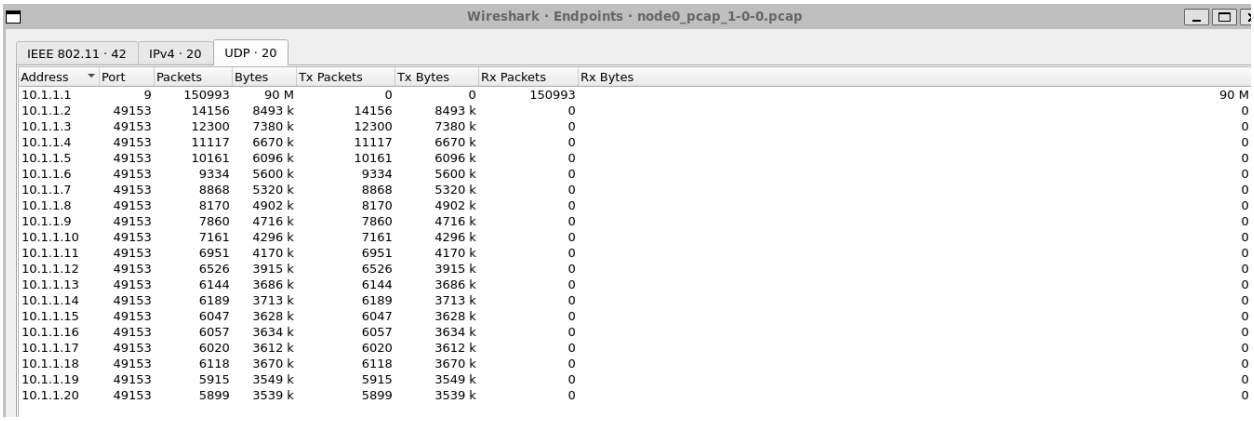


Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
10.1.1.1	9	167121	100 M	0	0	167121	100 M
10.1.1.2	49153	163024	97 M	163024	97 M	0	0
10.1.1.3	49153	607	364 k	607	364 k	0	0
10.1.1.4	49153	671	402 k	671	402 k	0	0
10.1.1.5	49153	266	159 k	266	159 k	0	0
10.1.1.6	49153	525	315 k	525	315 k	0	0
10.1.1.11	49153	582	349 k	582	349 k	0	0
10.1.1.17	49153	1014	608 k	1014	608 k	0	0
10.1.1.20	49153	432	259 k	432	259 k	0	0

We can also see that infact the 11 mbps system had lesser no of nodes that successfully transmitted atleast a packet compared to the 1 mbps system, thus I believe increasing the throughput makes the success time smaller, thus causing more collisions than a lower data rate system! But we can fundamentally attribute this causality to once again the window size that caused havoc in case A E1, I believe.

Case B E2:

For this case, as before, just the window size is set between 63 and 127. The result I witnessed is surprising. The overall net throughput v link rate did not seem to vary at all! At least for my figures from 1 to 11 Mbps. PFA the snapshots for a 1 mbps and a 11 mbps system.



Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
10.1.1.1	9	150993	90 M	0	0	150993	90 M
10.1.1.2	49153	14156	8493 k	14156	8493 k	0	0
10.1.1.3	49153	12300	7380 k	12300	7380 k	0	0
10.1.1.4	49153	11117	6670 k	11117	6670 k	0	0
10.1.1.5	49153	10161	6096 k	10161	6096 k	0	0
10.1.1.6	49153	9334	5600 k	9334	5600 k	0	0
10.1.1.7	49153	8868	5320 k	8868	5320 k	0	0
10.1.1.8	49153	8170	4902 k	8170	4902 k	0	0
10.1.1.9	49153	7860	4716 k	7860	4716 k	0	0
10.1.1.10	49153	7161	4296 k	7161	4296 k	0	0
10.1.1.11	49153	6951	4170 k	6951	4170 k	0	0
10.1.1.12	49153	6526	3915 k	6526	3915 k	0	0
10.1.1.13	49153	6144	3686 k	6144	3686 k	0	0
10.1.1.14	49153	6189	3713 k	6189	3713 k	0	0
10.1.1.15	49153	6047	3628 k	6047	3628 k	0	0
10.1.1.16	49153	6057	3634 k	6057	3634 k	0	0
10.1.1.17	49153	6020	3612 k	6020	3612 k	0	0
10.1.1.18	49153	6118	3670 k	6118	3670 k	0	0
10.1.1.19	49153	5915	3549 k	5915	3549 k	0	0
10.1.1.20	49153	5899	3539 k	5899	3539 k	0	0

Wireshark · Endpoints · node0_pcap_11-0-0.pcap

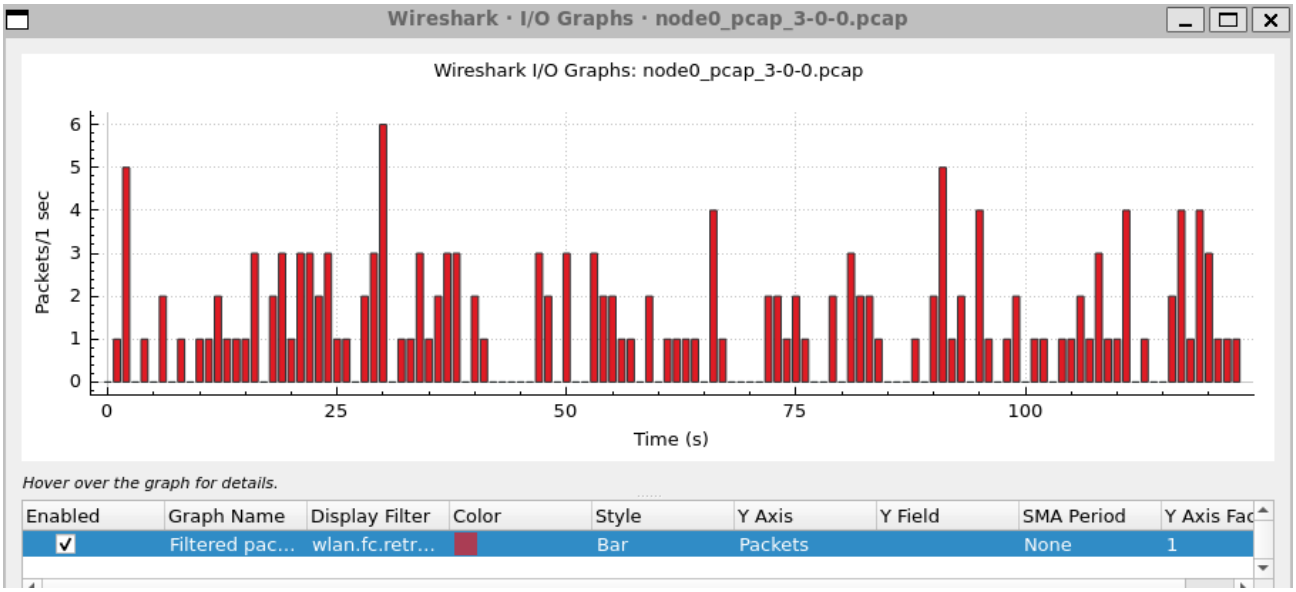
IEEE 802.11 · 42 | IPv4 · 20 | UDP · 20

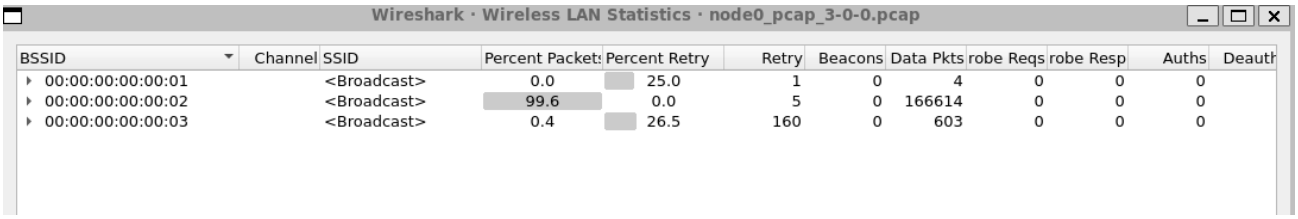
Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
10.1.1.1	9	151203	90 M	0	0	151203	90 M
10.1.1.2	49153	14528	8716 k	14528	8716 k	0	0
10.1.1.3	49153	12553	7531 k	12553	7531 k	0	0
10.1.1.4	49153	11523	6913 k	11523	6913 k	0	0
10.1.1.5	49153	10168	6100 k	10168	6100 k	0	0
10.1.1.6	49153	9550	5730 k	9550	5730 k	0	0
10.1.1.7	49153	9062	5437 k	9062	5437 k	0	0
10.1.1.8	49153	8700	5220 k	8700	5220 k	0	0
10.1.1.9	49153	7858	4714 k	7858	4714 k	0	0
10.1.1.10	49153	7362	4417 k	7362	4417 k	0	0
10.1.1.11	49153	6959	4175 k	6959	4175 k	0	0
10.1.1.12	49153	6648	3988 k	6648	3988 k	0	0
10.1.1.13	49153	6677	4006 k	6677	4006 k	0	0
10.1.1.14	49153	6456	3873 k	6456	3873 k	0	0
10.1.1.15	49153	6471	3882 k	6471	3882 k	0	0
10.1.1.16	49153	6297	3778 k	6297	3778 k	0	0
10.1.1.17	49153	6351	3810 k	6351	3810 k	0	0
10.1.1.18	49153	6312	3787 k	6312	3787 k	0	0
10.1.1.19	49153	1675	1005 k	1675	1005 k	0	0
10.1.1.20	49153	6053	3631 k	6053	3631 k	0	0

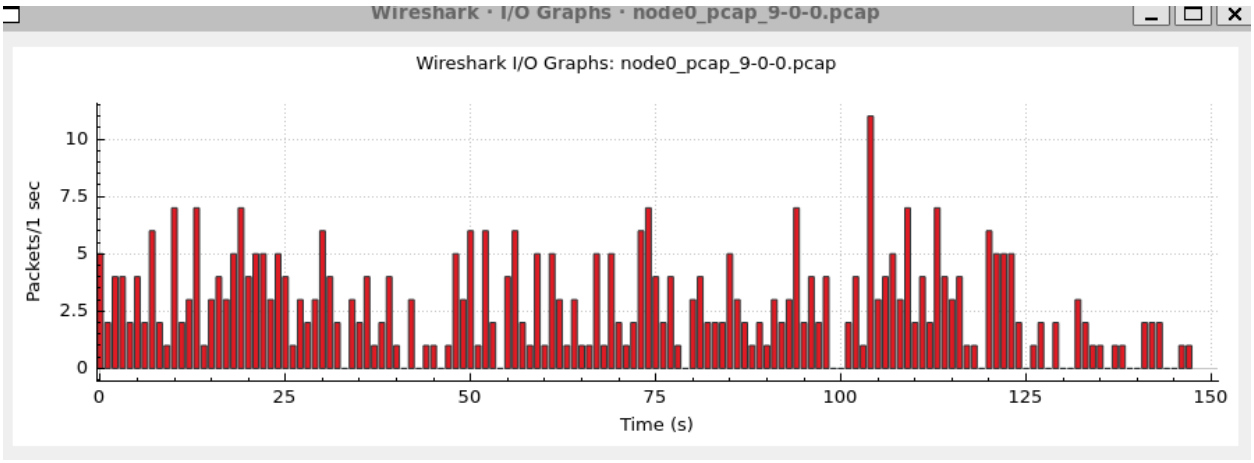
Again, I postulate the reasoning I had in my previous case here too! increasing the link rate, causes more collisions due to a reduced successful packet transmission time offsetting the increase in no of packets by increased collisions.

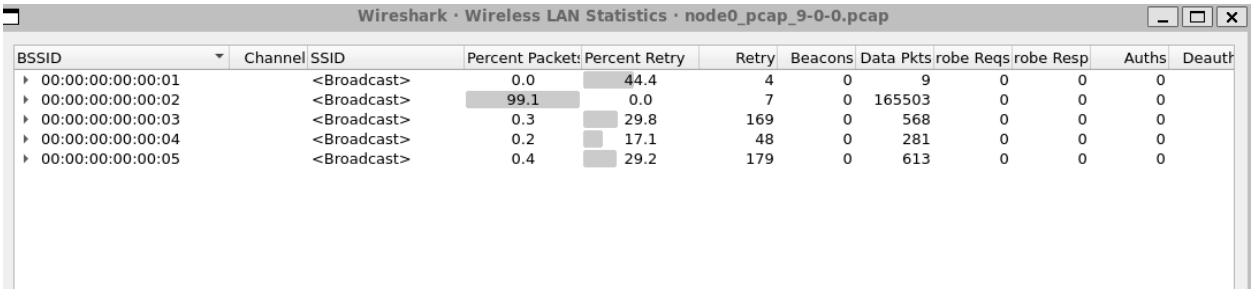
Collision Rate:

The average collision rate of a system for a particular number of nodes can be figured out from the pcap file, in Wireshark: **Wireless -> WLAN Traffic** leads us to a window **where we can view the retransmission data for that system.** We can also view them by applying the filter wlan.fc.retry==1 in the filter pane. Here I have attached two snapshots for a 3 node and a 9 node system for case A window size. (all E1)



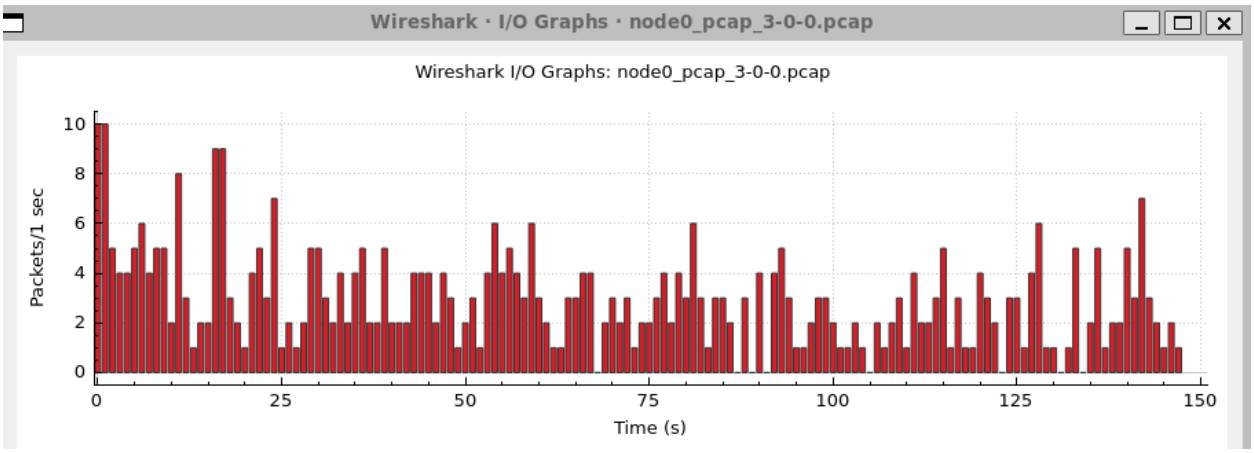


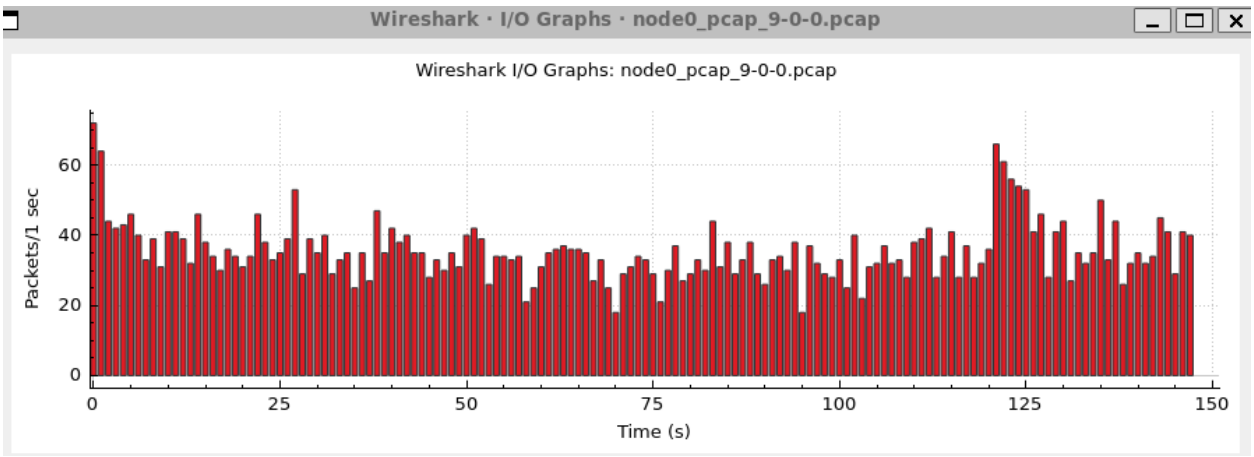




We can observe that the 9 node system experiences much more collisions than a 3 node system **in average and my observation was that collision rate increased with increasing nodes, i.e. the re-transmissions graph gets denser.**

Now for Case B (3 and 9 node systems):

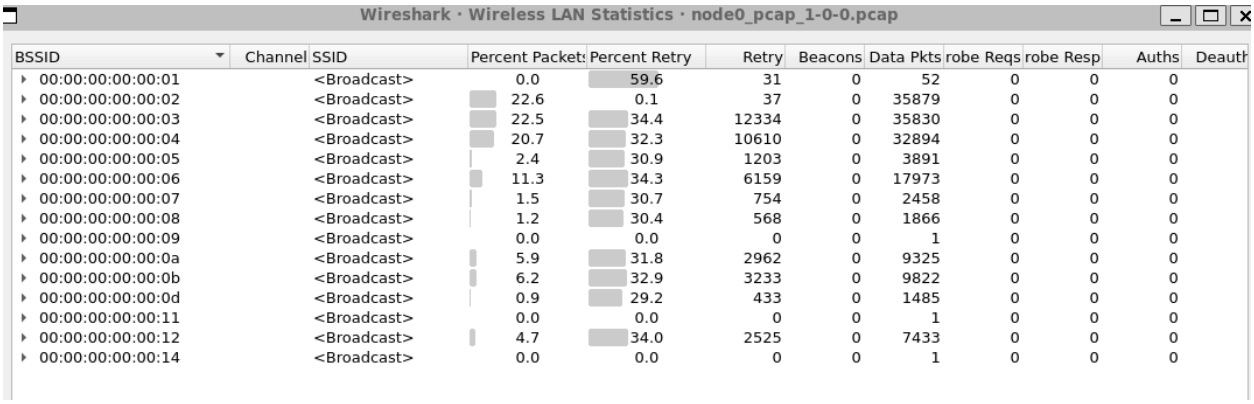




In this case, we can clearly see that the collision rates increase as n increase, **but compared to case A, case B has relatively lot more retransmissions! Thus, more attempts to transmit a packet by every node giving rise to more fairness!**

Now, coming to E2 case (for 1 mbps and 11 mbps systems respectively):

Case A:



BSSID	Channel	SSID	Percent Packets	Percent Retry	Retry	Beacons	Data Pkts	robe Reqs	robe Resp	Auths	Deauth
00:00:00:00:00:01		<Broadcast>	0.0	59.6	31	0	52	0	0	0	
00:00:00:00:00:02		<Broadcast>	22.6	0.1	37	0	35879	0	0	0	
00:00:00:00:00:03		<Broadcast>	22.5	34.4	12334	0	35830	0	0	0	
00:00:00:00:00:04		<Broadcast>	20.7	32.3	10610	0	32894	0	0	0	
00:00:00:00:00:05		<Broadcast>	2.4	30.9	1203	0	3891	0	0	0	
00:00:00:00:00:06		<Broadcast>	11.3	34.3	6159	0	17973	0	0	0	
00:00:00:00:00:07		<Broadcast>	1.5	30.7	754	0	2458	0	0	0	
00:00:00:00:00:08		<Broadcast>	1.2	30.4	568	0	1866	0	0	0	
00:00:00:00:00:09		<Broadcast>	0.0	0.0	0	0	1	0	0	0	
00:00:00:00:00:0a		<Broadcast>	5.9	31.8	2962	0	9325	0	0	0	
00:00:00:00:00:0b		<Broadcast>	6.2	32.9	3233	0	9822	0	0	0	
00:00:00:00:00:0d		<Broadcast>	0.9	29.2	433	0	1485	0	0	0	
00:00:00:00:00:11		<Broadcast>	0.0	0.0	0	0	1	0	0	0	
00:00:00:00:00:12		<Broadcast>	4.7	34.0	2525	0	7433	0	0	0	
00:00:00:00:00:14		<Broadcast>	0.0	0.0	0	0	1	0	0	0	

Wireshark · Wireless LAN Statistics · node0_pcap_11-0-0.pcap

BSSID	Channel	SSID	Percent Packets	Percent Retry	Retry	Beacons	Data Pkts	robe Reqs	robe Resp	Auths	Deauth
00:00:00:00:00:01		<Broadcast>	0.0	56.7	17	0	30	0	0	0	
00:00:00:00:00:02		<Broadcast>	97.5	0.0	39	0	163026	0	0	0	
00:00:00:00:00:03		<Broadcast>	0.4	26.2	159	0	608	0	0	0	
00:00:00:00:00:04		<Broadcast>	0.4	28.1	189	0	673	0	0	0	
00:00:00:00:00:05		<Broadcast>	0.2	20.6	55	0	267	0	0	0	
00:00:00:00:00:06		<Broadcast>	0.3	25.1	132	0	526	0	0	0	
00:00:00:00:00:0b		<Broadcast>	0.3	28.1	164	0	583	0	0	0	
00:00:00:00:00:0f		<Broadcast>	0.0	0.0	0	0	1	0	0	0	
00:00:00:00:00:11		<Broadcast>	0.6	27.6	280	0	1016	0	0	0	
00:00:00:00:00:14		<Broadcast>	0.3	26.5	115	0	434	0	0	0	

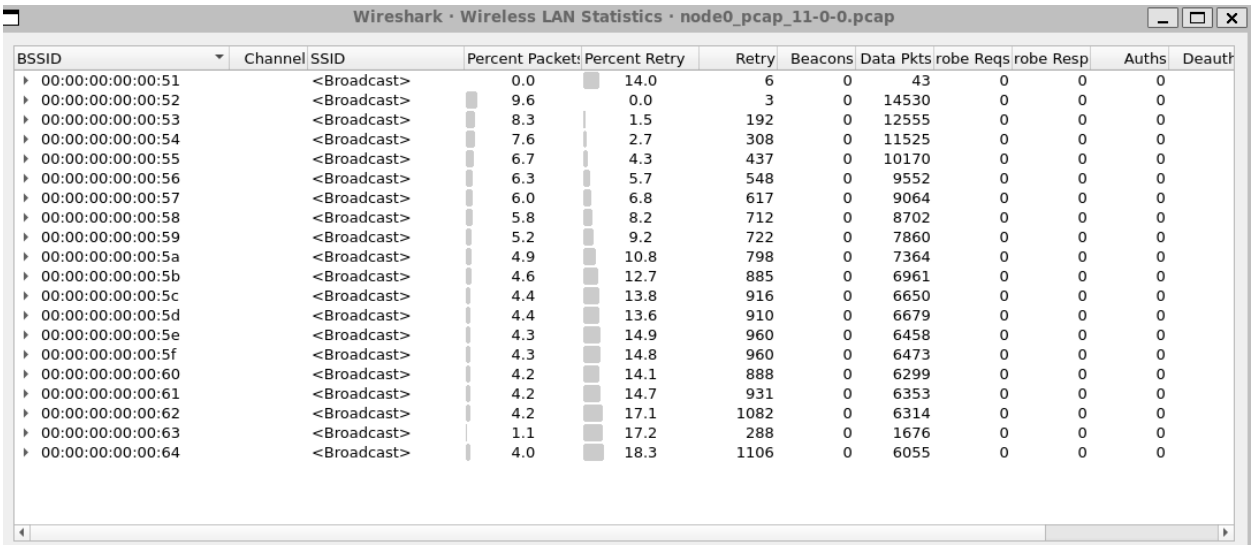
For case A, as we predicted, the 11 mbps variant had relatively few collisions (notwithstanding certain nodes which did not transmit at all!) and thus had an effectively more number of packets being transmitted.

Case B:

For case B, as I surmised, the higher throughput variant experienced more collisions than the lower throughput variant!

Wireshark · Wireless LAN Statistics · node0_pcap_1-0-0.pcap

BSSID	Channel	SSID	Percent Packets	Percent Retry	Retry	Beacons	Data Pkts	robe Reqs	robe Resp	Auths	Deauth
00:00:00:00:00:01		<Broadcast>	0.0	15.6	7	0	45	0	0	0	
00:00:00:00:00:02		<Broadcast>	9.4	0.0	1	0	14158	0	0	0	
00:00:00:00:00:03		<Broadcast>	8.1	0.4	44	0	12302	0	0	0	
00:00:00:00:00:04		<Broadcast>	7.4	0.5	58	0	11119	0	0	0	
00:00:00:00:00:05		<Broadcast>	6.7	0.9	90	0	10163	0	0	0	
00:00:00:00:00:06		<Broadcast>	6.2	1.0	95	0	9336	0	0	0	
00:00:00:00:00:07		<Broadcast>	5.9	0.8	70	0	8870	0	0	0	
00:00:00:00:00:08		<Broadcast>	5.4	1.3	104	0	8172	0	0	0	
00:00:00:00:00:09		<Broadcast>	5.2	1.2	95	0	7862	0	0	0	
00:00:00:00:00:0a		<Broadcast>	4.7	1.5	107	0	7163	0	0	0	
00:00:00:00:00:0b		<Broadcast>	4.6	1.7	115	0	6953	0	0	0	
00:00:00:00:00:0c		<Broadcast>	4.3	1.8	118	0	6528	0	0	0	
00:00:00:00:00:0d		<Broadcast>	4.1	1.8	110	0	6146	0	0	0	
00:00:00:00:00:0e		<Broadcast>	4.1	1.7	105	0	6191	0	0	0	
00:00:00:00:00:0f		<Broadcast>	4.0	1.9	115	0	6049	0	0	0	
00:00:00:00:00:10		<Broadcast>	4.0	1.9	116	0	6059	0	0	0	
00:00:00:00:00:11		<Broadcast>	4.0	1.9	113	0	6022	0	0	0	
00:00:00:00:00:12		<Broadcast>	4.1	1.9	119	0	6120	0	0	0	
00:00:00:00:00:13		<Broadcast>	3.9	2.5	145	0	5917	0	0	0	
00:00:00:00:00:14		<Broadcast>	3.9	2.3	134	0	5901	0	0	0	



BSSID	Channel	SSID	Percent Packets	Percent Retry	Retry	Beacons	Data Pkts	robe Reqs	robe Resp	Auths	Deauth
00:00:00:00:00:51		<Broadcast>	0.0	14.0	6	0	43	0	0	0	
00:00:00:00:00:52		<Broadcast>	9.6	0.0	3	0	14530	0	0	0	
00:00:00:00:00:53		<Broadcast>	8.3	1.5	192	0	12555	0	0	0	
00:00:00:00:00:54		<Broadcast>	7.6	2.7	308	0	11525	0	0	0	
00:00:00:00:00:55		<Broadcast>	6.7	4.3	437	0	10170	0	0	0	
00:00:00:00:00:56		<Broadcast>	6.3	5.7	548	0	9552	0	0	0	
00:00:00:00:00:57		<Broadcast>	6.0	6.8	617	0	9064	0	0	0	
00:00:00:00:00:58		<Broadcast>	5.8	8.2	712	0	8702	0	0	0	
00:00:00:00:00:59		<Broadcast>	5.2	9.2	722	0	7860	0	0	0	
00:00:00:00:00:5a		<Broadcast>	4.9	10.8	798	0	7364	0	0	0	
00:00:00:00:00:5b		<Broadcast>	4.6	12.7	885	0	6961	0	0	0	
00:00:00:00:00:5c		<Broadcast>	4.4	13.8	916	0	6650	0	0	0	
00:00:00:00:00:5d		<Broadcast>	4.4	13.6	910	0	6679	0	0	0	
00:00:00:00:00:5e		<Broadcast>	4.3	14.9	960	0	6458	0	0	0	
00:00:00:00:00:5f		<Broadcast>	4.3	14.8	960	0	6473	0	0	0	
00:00:00:00:00:60		<Broadcast>	4.2	14.1	888	0	6299	0	0	0	
00:00:00:00:00:61		<Broadcast>	4.2	14.7	931	0	6353	0	0	0	
00:00:00:00:00:62		<Broadcast>	4.2	17.1	1082	0	6314	0	0	0	
00:00:00:00:00:63		<Broadcast>	1.1	17.2	288	0	1676	0	0	0	
00:00:00:00:00:64		<Broadcast>	4.0	18.3	1106	0	6055	0	0	0	

Backoff Time:

The mean and variance of the backoff time slots per transmission is directly proportional to the time interval between successive packet transmissions/ frames (essentially the bin-width, if we plot the throughput vs time as a Bar plot), i.e. the average delay of the system, I believe, and from the looks of it, as the number of nodes increase, the initial trend is that the time between successive packet receptions was high (which I presume is due to collision between the nodes) and then starts to decrease. But theoretically, the mean and variance can be easily calculated using mincw+maxcw/2 and variance of (maxcw-mincw)/12 for every stage! Not the whole mincw, maxcw range! But I am right now unable to directly pinpoint from where we can extract this from Wireshark.

END OF REPORT