Project - Saturation Throughput

EE597 Spring 2020
University of Southern California

Technical Report

Adithya Venkat Ramanan: 1932332235
Sirisha K S: 3251982100

Introduction

The ns-3 simulator is a discrete-event network simulator used for simulating real world networks on one computer by writing scripts in C++ or Python. Ns-3 helps to create various virtual nodes and with the help of various helper classes it allows us to install devices, internet stacks, application to our nodes. ns-3 provides models of how packet data networks work and perform and provides a simulation engine for users to conduct simulation experiments. PointToPoint, Wireless, CSMA connections can be created between nodes by using ns-3. PointToPoint connection is same as a LAN connected between two computers. Wireless connection is same as WiFi connection between various computers and routers. CSMA connection is same as bus topology between computers. After building connections we try to install NIC to every node to enable network connectivity. When network cards are enabled in the devices, we add different parameters in the channels (i.e., real world path used to send data) which are data-rate, packet size, etc. Now we use Application to generate traffic and send the packets using these applications.

802.11 protocol has been standardized by IEEE for wireless local area networks. This standard adopts a CSMA/CA medium access control protocol with exponential backoff, called distributed coordination function (DCF). DCF in the basic form consists of carrier sensing functionality and a random backoff protocol. The carrier sensing functionality helps a station with a packet to transfer, monitor the channel activity to check if the channel is free or busy. If the channel is busy, it helps to postpone transmission attempts until the medium is free. It checks if the channel is idle for a period equal to distributed interframe space (DIFS). The time immediately following an idle DIFS is slotted, and a station is allowed to transmit only at the beginning of each slot time, defined as the time needed at any station to detect the transmission of a packet from any other station. After sensing an idle DIFS, the station generates a random backoff interval before transmitting. The backoff time counter is decremented as long as the channel is sensed idle, stopped when a transmission is detected on the channel, and reactivated when the channel is sensed idle again for more than a DIFS. The station transmits when the backoff time reaches zero. At each transmission, the backoff time is uniformly chosen in the range (0, w-1). At the first transmission attempt, w=W, namely the minimum backoff window. After each unsuccessful transmission, w is doubled, up to a maximum value $2^{m}W$. All this is well cited in the work published by Bianchi **[1]**.

The aim of this project is to validate the analytical model presented by Bianchi [1] for computing the saturation throughput performance of the IEEE 802.11 CSMA/CA protocol. We have used two tools to simulate the performance of 802.11, namely NS-3 and MATLAB. We have generated plots similar to the saturation throughput performance plots from Bianchi's paper [1].

Mathematical Model

According to Bianchi's work, the basic DCF model can be mathematically modelled using a Markov chain. Let b(t) be the stochastic process representing the size of the backoff window for a given station at slot time t. Let the number of stations contending be n. Let $W_i=2^iW$, where i ε (0, m) be called the backoff stage and s(t) be the stochastic process representing this backoff stage (0,1,…,m) of the station at time t and p be the probability that a transmitted packet collides. The Markov chain is represented as a bidimensional process {s(t), b(t)}. This is represented in figure 1.

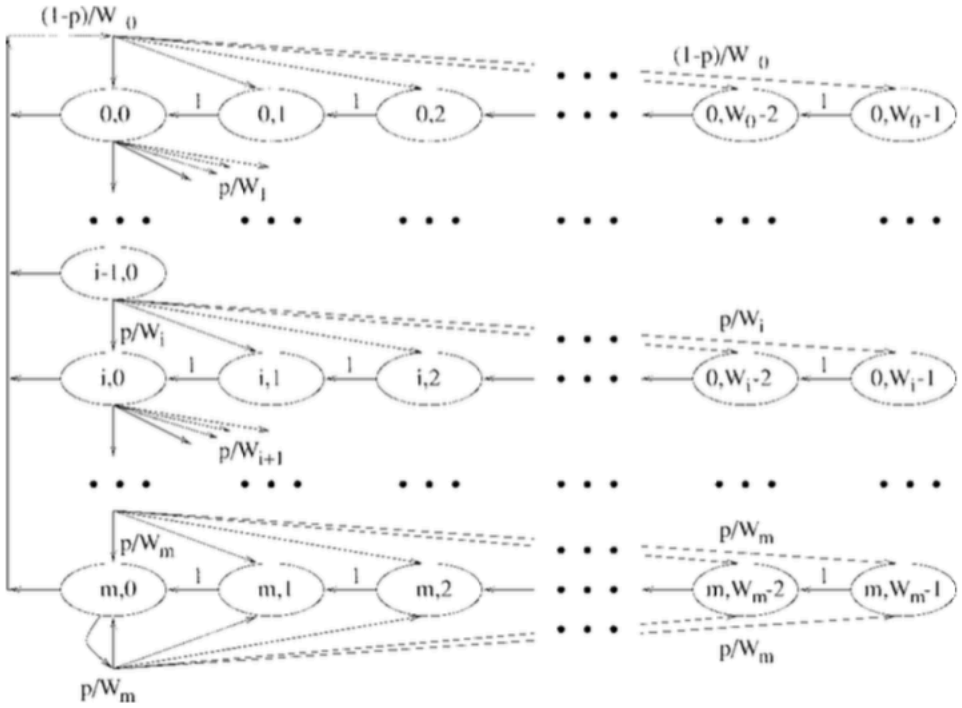

Figure 1: Markov chain model for the backoff window size

Let τ be the probability that a station transmits in a generic slot time. This can be computed using the formula below.

$$\tau = \frac{2\,(1-2p)}{(1-2p)(W+1)+pW(1-(2p)^m)}$$

The probability p that a transmitted packet collides is given as

$$p = 1 - (1 - \tau)^{n-1}$$

The probability P_{tr} that in a slot time there is at least one transmission, given n active stations and the probability P_s that a transmission is successful can be calculated using

$$P_{tr} = 1 - (1 - \tau)^n$$

$$P_s = \frac{n\tau(1 - \tau)^{n-1}}{P_{tr}}$$

Let Ψ be the r.v. representing the number of consecutive idle slots between two consecutive transmissions on the channel such that

$$E[\Psi] = \frac{1}{P_{tr}} - 1$$

The normalized system throughput S is given as

$$S = \frac{E[time\ used\ for\ successful\ transmission\ in\ \ interval]}{E[length\ of\ renewal\ interval]}$$

$$S = \frac{P_s\, E[P]}{E[\Psi] + P_s T_s + (1 - P_s) T_c}$$

where E[P] =average packet length
T_s = average time the channel is sensed busy because of a successful transmission
T_c= average time the channel is sensed busy by the stations during a collision

(Note: E[P], T_s , T_c must be measured in slot times)

Let the packet header $H = PHY_{hdr} + MAC_{hdr}$
δ = propagation delay
E[Ψ*] = average length of the longest packet payload involved in a collision

$$T_s = H + E[P] + SIFS + \delta + \text{ACK} + \text{DIFS} + \delta$$

$$T_C = H + E[P^*] + \text{DIFS} + \delta$$

Table 1 presents parameters employed for the matlab and ns-3 simulation.

Channel bit rate	1 Mbps
Propagation Delay	1 µs
SIFS	28 µs
DIFS	128 µs
Slot time	50 µs
MAC header	272 bits
PHY header	128 bits
ACK length	112 bits +PHY header
Packet payload	8184 bits

Table 1

Slot Time, Backoff Stage (m), Minimum and Maximum Contention Window values for the IEEE 802.11a/b/g.

PHY standard	Slot Time	CW_{min}	CW_{max}	Backoff Stages
802.11a	9 µs	15	1023	7
802.11b	20 µs	31	1023	6
802.11g	9 or 20 µs	15 (31)	1023	7 (6)

Table 2

MATLAB Simulation

The input parameters are set according to Bianchi's paper [1]. These parameters are listed in Table 1.

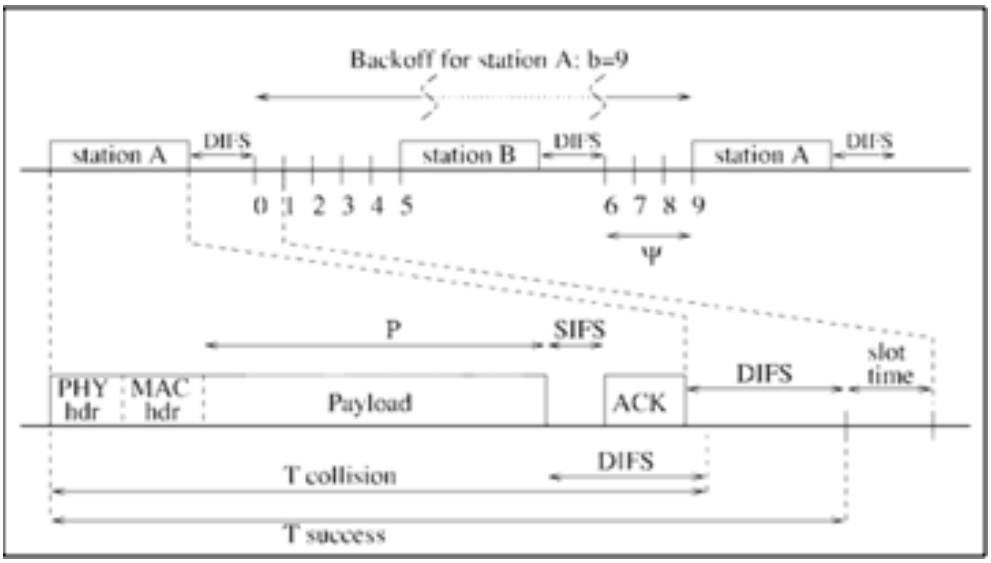

Figure 2: Basic Access Mechanism

From figure 2, we calculate the Tc and Ts in slot time units in MATLAB. The calculation for throughput is exactly like how Bianchi has explained in his paper. Using the formulae mentioned above with its underlying assumptions, we theoretically compute the throughput using MATLAB.

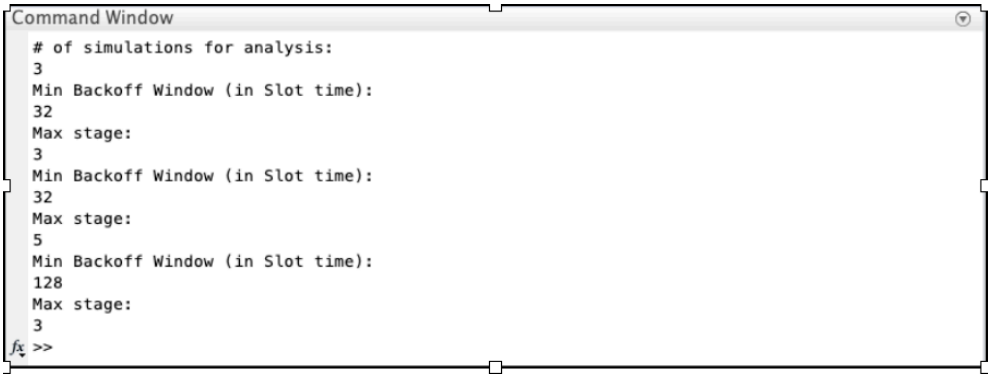

Figure 3: MATLAB Command Line Input

With the simulation parameters as in figure 3, the desired graph is generated, in figure 4, similar to Bianchi's.

MATLAB Simulation Output

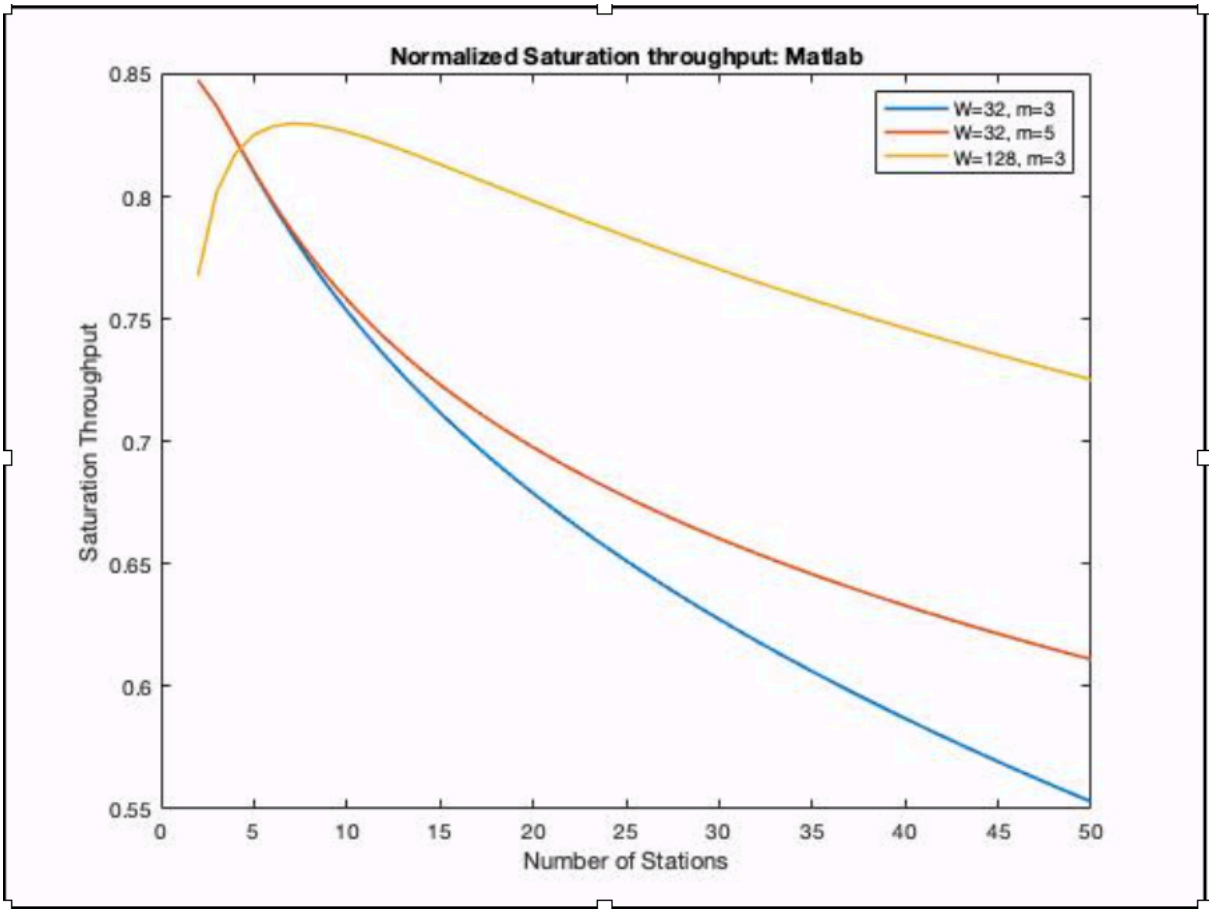

Figure 4: Plot of saturation throughput versus number of stations on MATLAB

Observation

By comparing the graph shown in Figure 4 with the graph obtained in Bianchi's paper [1], it can be seen that the matlab simulation that we implemented resulted in a saturation throughput graph which is very similar to the ideal graph in Bianchi's paper [1]. The throughput values obtained using the matlab simulation match the throughput values in the paper [1]. It can be clearly seen that as we vary the values of contention window, backoff counter and number of stages, the throughput values also vary. We have run the simulation for 3 different cases.

1. W= 32, m=3
2. W= 32, m=5
3. W= 128, m=3

By observing the above graph it can be concluded that as the number of nodes contending for the medium increase, the saturation throughput decreases. This is because as the number of nodes contending for the medium increase, the number of collisions also increase. This results in a lower throughput. This can be overcome by increasing the size of contending window and stages, which results in a slightly higher throughput even as the number of contending nodes increase.

NS3 Simulation

The IEEE 802.11 protocol with DCF backoff process which can be modelled as a bi-dimensional Markov chain can be simulated using ns3 simulator. This simulation environment is widely used in networking and has a flexible architecture with numerous existing models. Ns3 nodes can contain a collection of NetDevice objects, much like an actual computer contains separate interface cards for Ethernet, Wifi, Bluetooth, etc. By adding WifiNetDevice objects to ns3 nodes, one can create models of 802.11-based infrastructure. The set of 802.11 models provided in ns3 attempts to provide an accurate MAC-level implementation of the 802.11 specification and to provide a packet-level abstraction of the PHY-level for different PHYs, corresponding to 802.11a/b/e/g/n/ac/ax specifications. Ns3 implements the PHY layer of wireless networks at a packet abstracted level for the end-to-end transmissions, where PHY models are utilized to statistically evaluates the successful or failure of the reception of each frame. The physical layer models are responsible for modeling the reception of packets. The YansWifiPhy class has been the only physical layer model until recently. The widely used MAC high models presented by Ns3 are Access Point (AP) (ns3::ApWifiMac) and non-AP Station (STA) (ns3::StaWifiMac). The MAC low layer consists of ns3::ChannelAccessManager and ns3:: DcfState which implement the DCF and EDCAF functions. ns3::Txop and ns3::QosTxop which handle the packet queue, packet fragmentation, and packet retransmissions if they are needed. The ns3::Txop object is used by high MACs that are not QoS-enabled and for transmission of frames. The YansWifiChannel is the only concrete channel model class in the ns3 wifi module. The ns3::WifiPhy is an abstract base class representing the 802.11 physical layer functions. Packets passed to this object are sent over a channel object, and upon reception, the receiving PHY object decides (based on signal power and interference) whether the packet was successful or not. This class also provides a number of callbacks for notifications of physical layer events, exposes a notion of a state machine that can be monitored for MAC-level processes such as carrier sense, and handles sleep/wake models and energy consumption. The ns3::WifiPhy hooks to the ns3::MacLow object in the WifiNetDevice. The WifiPhyStateHelper helps in maintaining the PHY state machine and InterferenceHelper helps in tracking all packets seen in the channel. Class ns3::YansWifiPhy is responsible for taking packets passed to it from the MAC and sending them onto the ns3::YansWifiChannel to which it is attached. It is also responsible to receive packets from that channel, and, if reception is deemed to have been successful, to pass them up to the MAC. We have used the WIFI_PHY_STANDARD_80211b model for the WifiHelper to set the MAC layer since default parameters shown in Table 2 for slot time, CWmin and CWmax are close to that used in Bianchi's paper. We have used PacketSinkHelper and OnOffHelper for the MAC layer of the AP and STA nodes. The 802.11 Distributed Coordination Function is used to calculate when to grant access to the transmission medium. ConstantRateWifiManager is one of the many rate control algorithms available in ns3. The constant rate control algorithm always uses the same transmission mode for every packet. All the above mentioned specifications has been taken from documentation listed in **[2], [3], [4]**.

Topology Setup

We have used ns3 tools to create a test environment that closely matches that of Bianchi's [1]. We used the objects from the allocator models in ns3 to allocate positions for one access point (AP) and 'n' STA nodes in our network. We have chosen to vary the 'n' STA nodes from 2 to 50 by incrementing it in steps of 1 in order to plot the saturation throughput graph similar to the graph in Bianchi's paper [1]. MobilityHelper is used to allocate positions for all the nodes in the wifi network. Ns3 has various mobility models that can be used to set up a desired network topology.

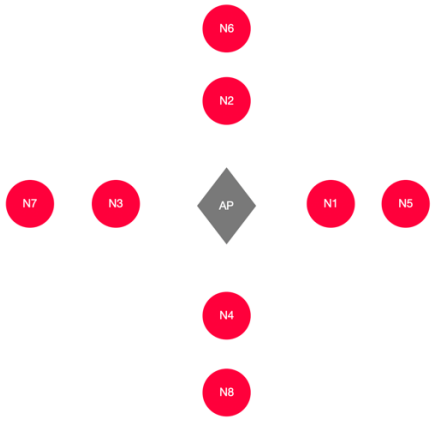

Figure 5: Network with '8' STA nodes and 1 AP

Saturation Throughput

After setting up the network and its parameters using ns3, the saturation throughput is calculated by simulating this on ns3 and using the formula shown below.

$$S = \frac{N_{frame} L_{MSDU}}{T_{sim} R_{datarate}}$$

where T_{sim} = Simulation duration (seconds)
N_{frame}= Number of packets received
$R_{datarate}$= Data rate during transmission (bps)
L_{msdu}= Length msdu packet (bits)

Ns3 Simulation Output

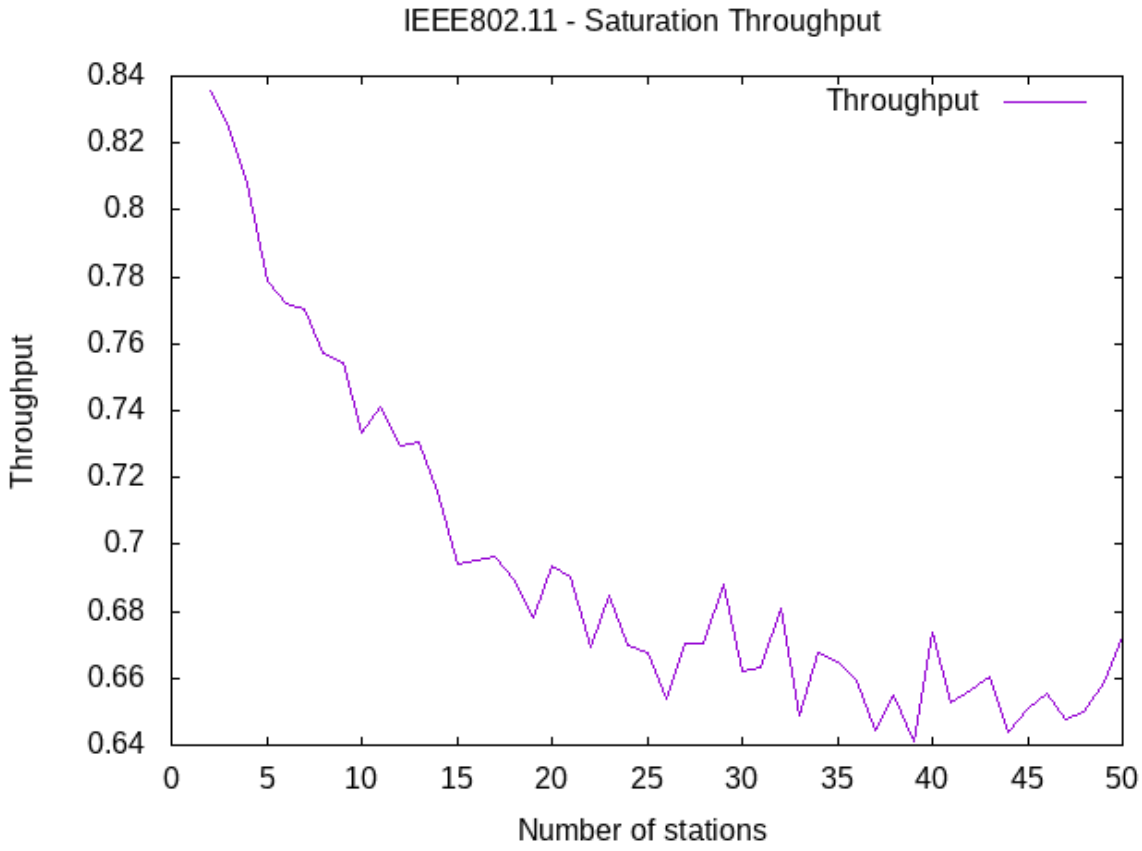

Figure 6: Plot of Saturation throughput versus number of stations on Ns3

Observation

By comparing the graph shown in Figure 6 with the graph obtained in Bianchi's paper [1], it can be seen that the ns3 simulation that we implemented resulted in a saturation throughput graph which is similar to the ideal graph in Bianchi's paper [1]. The throughput values obtained using the ns3 simulation are close to the throughput values in the paper [1]. We have run the simulation for the case when W=32 and m=5.

By observing the above graph it can be concluded that as the number of nodes contending for the medium increase, the saturation throughput decreases. This is because as the number of nodes contending for the medium increase, the number of collisions also increase. This results in a lower throughput. This can be overcome by increasing the size of contending window and stages,

which results in a slightly higher throughput even as the number of contending nodes increase. It can be concluded that the practical values of saturation throughput obtained using ns3 simulation and close to the theoretical values.

References

[1] G. Bianchi. IEEE 802.11-saturation throughput analysis. IEEE Communications Letters, 2(12):318–320, Dec 1998.
URL: https://pdfs.semanticscholar.org/4a5c/ f874e9469815113c7ea93ff97317bdb52a90.pdf, doi:10.1109/4234.736171

[2] https://repository.lib.ncsu.edu/bitstream/handle/1840.20/37030/etd.pdf?sequence=1

[3] https://www.nsnam.org/docs/release/3.13/models/html/wifi.html

[4] The ns-3 Wi-Fi Module Documentation
www.nsnam.org › bugzilla › attachment