Hashing Based Distributed Backoff (HBDB) Mechanism for IEEE 802.11 Wireless Networks

M. Devipriya, B. Nithya, and C. Mala*
Department of Computer Science and Engineering
National Institute of Technology
Tiruchirappalli - 620015, Tamil Nadu, INDIA
{306111051, nithya, mala}@nitt.edu

Abstract

Binary Exponential Backoff (BEB) is the De-facto mechanism for contention control in IEEE 802.11 Wireless LAN. The exponential growth of Contention Window (CW) in all BackOff (BO) stages and randomness in BO selection causes unnecessary waiting time, high collision rate and unfairness in accessing the channel among the nodes. To overcome these anomalies, this paper proposes Hashing Based Distributed Backoff (HBDB) algorithm. It allows a node to dynamically adopt different CW values based on the collision probability. To choose unique collision probability value, linear probing is used. The simulation results show the effectiveness of the proposed algorithm in linear and random topologies under low, medium and high traffic scenarios.

Keywords:WLAN, BackOff Algorithm and Contention Window.

1 Introduction

Wireless Local Area Network (WLAN) is used in Educational Institutions and Work places. IEEE 802.11 WLAN supports two configurations such as Infrastructure mode and Adhoc mode. Client server WLANs are centralized networks that need an Access Point (AP) which controls and coordinates the nodes that are accessing the channel. Adhoc networks are decentralized networks that do not rely on AP. Adhoc networks do not have any fixed infrastructure, nodes themselves are responsible for the channel access. The channel for transmission is broadcast in nature. Each node randomly transmits data whenever the medium becomes idle. The receiver nodes identified by a particular transmitting node have its own neighboring nodes surrounding it, which may not be detected by the transmitting node [1]. Hence a mechanism is needed for the nodes to avoid collision at the receiver side. The nodes which are in the same communication range can send their data simultaneously provided that there is no interference. Otherwise simultaneous transmissions cause collision which leads to loss of data, thereby degrading overall network performance.

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is the medium access mechanism used in wireless shared medium communication system. IEEE 802.11 Distributed Coordination Function (DCF) is used for Adhoc networks and is based on CSMA/CA. It uses physical carrier sensing and optional virtual carrier sensing to avoid collisions at transmitter and receiver respectively. Physical carrier sensing approach uses Clear Channel Assessment (CCA) signal to detect whether the channel is idle or not. RTS/CTS handshake incorporated in virtual carrier sensing reduces packet collisions.

In DCF [5], a node that wishes to transmit should find whether the channel is idle for at least Distributed Inter Frame Space (DIFS) interval. If the channel is idle, the node can send data at the beginning of a slot and this is possible only when traffic is very low. Under high load, to resolve collision, BO

Journal of Internet Services and Information Security (JISIS), volume: 5, number: 3 (August 2015), pp. 1-18
*Corresponding author: Tel: +91-431- 2503208

algorithm is used [11]. If channel is sensed as busy, the node should defer its transmission and wait for the channel to become idle for at least DIFS. At that time, it chooses a BO value from the CW. The CW consists of set of slots starting from 0 to the value specified by the BO Algorithm. Within the CW, the node should choose a random number of slots. BO timer counts down on each idle slot and freezes its counter when the medium is busy. The node which has got the lowest BO value among the competing nodes will access the channel first. Other nodes should defer their transmission. If the channel becomes idle for more than DIFS then the nodes resume their residual BO timer and new competing nodes choose a new BO value. The node whose counter becomes 0 can access the channel.

DCF uses Binary Exponential BackOff (BEB) [7] algorithm to increase the CW depending on the channel condition. It uniformly selects a BO time in the interval [0, CW-1]. BEB starts with minimum CW (CW_{min}) and then it is doubled for every failed transmission bounded by maximum CW (CW_{max}), i.e., $CW_{min} \leq \text{CW} \leq CW_{max}$. Whenever the transmission succeeds, BEB reschedules its CW to CW_{min}. The problem exists in BEB as it increases CW exponentially even if there is a collision under light load. This leads the channel to be idle for most of the time under low traffic scenario. Resetting CW to its minimum value after the successful transmissiom leads to unfairness problem and increases collision probability under high traffic scenario. The Algorithm proposed is Hashing Based Distributed BackOff (HBDB). It changes the CW according to the changes in the network traffic. The scope of the paper is limited to compare HBDB with BEB.

The rest of the sections are organized as follows. Section 2 describes the related work, Section 3 is the proposed algorithm and Section 4 analyses the performance of the proposed algorithm with BEB. Finally, section 5 concludes the paper.

2 Related Work

In this section, various BO algorithms are discussed. Formulae used to compute CW in various BO algorithms along with the simulation parameters are listed in Table 1.

In [8], the Balanced BackOff Algorithm (BBA) is proposed. This mechanism increases the CW linearly for N transmission failures. For N+1 to M transmission failures, CW increases exponentially. After M^{th} transmission failure, it again increases CW linearly. Upon successful transmission, CW is reduced by half. But linear increase in CW, may not give the optimum waiting time under high traffic scenarios. Therefore, it increases collision rate and number of retransmissions.

In [13], Smart Adaptive BackOff Algorithm (SABA) is introduced. It uses 2Mb/s channel, CBR (Constant Bit Rate) traffic, and random way-point model for mobility. In this method, CW is increased exponentially for the successful transmissions and stores CW for last 5 successes in the history array. If the transmission fails, the node enters the BO strategy. In this, if the history array has 5 elements and it is used for the first time, then the average of those elements is taken as new CW. If the second transmission failure occurs, it should either take logarithmic increase or linear increase depending on threshold value N. For a set of continuous successes, CW increases exponentially hence there will be a chance for choosing large BO leading to high end-to-end delay.

In [3], Waiting Time based BackOff (WTB) is developed for calculating the CW using Waiting time of the nodes. The algorithm assumes ideal channel condition, no hidden terminal effect, equal average packet arrival rate for all stations and finite number of stations. In this technique, when the channel is busy, the node chooses the BO value from default CW. If the node encounters any transmission failure, it calculates the Waiting time (Wt). Wt is calculated using three parameters such as channel busy time during successful transmission, collision of packets of all the stations and waiting time while decrementing the back off timer. Wt is then compared with Maximum Waiting time (Wt_{max}) in order to determine retry limit (a) of the node. CW is calculated using retry limit (a) and CW_{min} value. If waiting time of a

node is less than one third of Wt_{max} then CW increases 4 times, it exhibits huge delay in the low traffic scenario.

In [10], Traffic Adaptive BackOff Algorithm (TABA) is introduced as a new CW Scheme. In this algorithm, error free channel, negligible propagation delay, no interference from nearby Basic Service Set (BSS) are assumed. A Channel of 16Mbps and slot time of 50 μs is used. Considering centralized WLAN, it will have a monitoring period (T) of 8 to 1023 slots to monitor the number of collisions. Depending upon the number of collisions, T is increased. U is the sum of slots used to carry data and collided slots in the monitoring period. Using U & T, TABA calculates CW value (CW_{TABA}). T is doubled if T $< CW_{TABA}$, T remains as such if T $\geq CW_{TABA}$. Here CW_{TABA} extends from 7 slots to 7796 slots. Increasing the CW to 7796 leads to channel being idle for longer time. In low traffic condition, even 7 Slots for CW_{min} may lead to collision. In the formula, the term log (T-U) may become indefinite when T and U become equal.

In [9], Binomial BackOff Algorithm (BiBA), is introduced. It assumes nodes under saturated condition, error free channel and no hidden terminal. In BiBA, Binomial distribution gives different probability for different slots. If CW is 31, there is 100% probability to choose same channel i.e, (0mod31,31mod31 =0). If CW is 63, there is 50% probability to choose same channel (0mod31) and 50% chance to choose next channel (63mod31=1). If CW is 127, there is 50% chance to choose the same channel and 50% chance to choose next 3 channels(127mod31), same procedure is followed until the CW value becomes 1023. The efficiency of this protocol rely on the ability of the ability of radios to switch between the channels quickly to increase network throughput without a central coordinator.

In [4], Contention Window Control (CWC) Scheme is proposed considering decentralized network of 2Mb/s channel, 50 nodes, and 600 s of simulation time, CBR traffic. In this algorithm, BO range is divided into sub ranges and assigned to particular collision resolution level. Both lower and upper bounds of BO range are increased during collision. CS history array stores the network condition which is taken into account for CW optimization. If channel is idle, the node stores 0 in the CS array. If the channel is busy (or collision) then the node stores 1 in the CS array. The algorithm uses different transitions (CW assignment) for both successful transmission and collisions. Since the lower bound (lb) of CW is also increased, this scheme reduces the number of slots. This may be feasible if the interval between Lower bound (lb) and Upper bound (ub) value is large, otherwise the possibility of the collision is high if different nodes have same backoff subrange. It increases contention window twice or 1.7 times whenever channel is busy. If successfully transmitted, $CW_{ub} = CW_{ub} * 0.57$ and CW_{lb} = CW_{ub}-32. If the interval between lb and ub is less, it can cause collision in the high traffic scenario.
To overcome the above mentioned problems, this paper proposes Hash Based Distributed BackOff (HBDB) algorithm to enhance the throughput performance under random & linear topologies. Based on the collision probabilities stored in the hash table, Hash Based Distributed BackOff (HBDB) algorithm effectively chooses CW value which is neither large nor small to guarantee successful transmission thereby increasing network throughput.

3 Proposed BO Algorithm

In this section, the proposed Hash Based Distributed BackOff (HBDB) algorithm is discussed.

3.1 Hash Based Distributed Backoff (HBDB)

In HBDB algorithm, each node calculates the transmission probability in a selected slot. Using this transmission probability, the success probability for a particular node among n nodes and the collision probability are determined. Collision probability is used to predict the current channel condition. This

Table 1: Formulae used to Calculate Contention Window in various BO algorithms

BO Algorithm	**CW Formula upon success**	**CW Formula upon collision**	**Simulation Parameters**
BBA [8]	CW = CW/2 upon success	if(collision count $<$ N) CW = CW+l; if(collision count $>$ N & $<$ M) CW = CW * 2; if (collision count $>$ M) CW = CW + l;	2Mbps channel, CW_{min} = 32, CW_{max}=1024
SABA [13]	if(success) Exponential Increment; save it in history array;	if(failure) then if(history array contains all 5 values for succesful transmission) if(array used 1^{st} time) then CW = average (array elements); else if(CW $>$ N) Linear Increment; else Logarithmic Increment;	2Mbps channel, CBR traffic, random waypoint model for mobility with maximum speed of 4 m/s, 900 s of simulation time, 10 to 100 nodes, CW_{min} = 32
WTB [3]	CW = CW_{min} * pow(2,a-1); b = (float)rand() / RANDMAX; BO value = (int)(CW * b);	CW = CW_{min} * pow(2,a-1); b = (float)rand() / RANDMAX; BO value = (int)(CW * b);	11Mbps channel, CW_{min} = 32, CW_{max} = 1024, 20 nodes, RTS/CTS-DATA-ACK, slot time 20μs.
TABA [10]	CW_{TABA} = [log(T-U)-log(2T)] / [log(T-1)-log T];	CW_{TABA} = [log(T-U)-log(2T)] / [log(T-1)-log T];	Centralized WLAN with 16Mbps channel, Slot time 50μs, CW_{min} = 7 slots, CW_{max} = 7746 slots
BiBA [9]	Upon success it remains in the default CW_{min} as BEB. In order to chose a slot value, it uses Binomial distribution.	Upon failure it increments the slot by (+1 or +3 or +5) depending on the CW mod CW_{min} value. CW increases exponentially during failure. slot value is chosen using Binomial distribution.	11Mbps channel, CW_{min} = 32, CW_{max} = 1024 , CBR traffic, users RTS/CTS-DATA-ACK and DATA-ACK access method.
CWC [4]	CW_{ub}(i) = CW_{ub}(i-1)*Z; CW_{lb}(i) = CW_{ub}(i)-size; i be contention level, Z be a number, size be CW range for every level.	CW_{ub}(i) = CW_{ub}(i-1)*Z; CW_{lb}(i) = CW_{ub}(i)-size; i be contention level, Z be a number, size be CW range for every level.	2Mbps channel, 50 nodes, random waypoint model for mobility, simulation time of 600s.

will give a clue for the nodes to determine whether the transmission will succeed or not. Each node maintains a hash table to store the collision probabilities. Whenever a collision occurs, the node selects a value from the hash table. This value is compared with Minimum Threshold (MinTh) and Maximum Threshold (MaxTh) values. Based on the compared results, CW is either linearly or exponentially or polynomially increased to avoid repeated immediate retransmission and unnecessary delay. After choosing the CW value, a random value from the interval [0,CW] is chosen by the node as its BO value.

3.2 Calculating Collision Probability

The value T, probability of a node that transmits in a particular slot when BO reaches to zero is given in Eq.1 [12],

$$T = \frac{2(1-2p)}{(1-2p)(CW_{min}+1)+pCW_{min}(1-(2p)^i)} \tag{1}$$

where p is the probability that BO timer is reduced by 1 at the beginning of each idle slot [6]. CW_{min} is the minimum contention window value; i is the maximum BO stage. Probability of successful transmission (P_s) of a node is given in Eq.2 [2],

$$P_s = \frac{nT(1-T)^{n-1}}{1-(1-T)^n} \tag{2}$$

where n is number of nodes in the network. Probability of collision using P_c is given in the Eq.3.

$$P_c = 1 - P_s \tag{3}$$

After calculating P_c values, those values are stored in Hash table as mentioned in the following subsection 3.3.

3.3 Hash Table

Hash Table h[r] stores the set of P_c values. It is used to avoid assigning the same collision probability value again. The uniqueness of collision probabilities stored in the hash table is important. A hash value is retrieved using the proposed Eq.4,

$$r = rand() \,\%\, HashTableSize; \tag{4}$$

r is random value which is used to retrieve a P_c from the Hash table. Linear probing technique in hashing is used to resolve collision in storing the values in the hash table. It is employed to choose a new value of hash every time. It increments the hash block by 1 to choose a new probability value if h[r] value is chosen before.

3.4 Determining Contention Window using P_c

After retrieving a P_c value from the hash table, it is compared with MinTh and MaxTh to determine the level of network contention. The low contention level is detected, if the P_c is lesser than MinTh value. To minimize the channel access delay, the linear increment on CW is employed as given in the proposed Eq.5. If the collision probability (P_c) is greater than MaxTh, then CW is exponentially incremented as in Eq.6 to minimize collisions in highly congested network. Otherwise the polynomial increment on CW as given in the proposed Eq.7 is utilized to guarantee the successful transmission.

$$CW = ((\beta * i)+1) * CW_{min} \tag{5}$$

$$CW = \beta^i * CW_{min} \tag{6}$$

$$CW = (i+1)^\beta * CW_{min} \tag{7}$$

where, β is a constant value. In all these cases, different β values are taken to obtain different HBDB's CW sequences as in Table 2. These sequences are tested and simulated using NS2 simulator. The simulation results are presented and analysed in section 4.

Table 2: Comparison between BEB and HBDB using CW values

BO stage	**BEB**	**HBDB**				
		h[r]>MaxTh	h[r]>MinTh && h[r]≤MaxTh		h[r]<MinTh	
		CW = $\beta^i * CW_{min}$	CW=((i+1)$^{\beta}$) $* CW_{min}$		CW=$(\beta * i)+1) * CW_{min}$	
		β=2	β=1	β=1.5	β=5	β=7
0	31	15	15	15	15	15
1	63	30	30	42	90	120
2	127	60	45	77	165	225
3	255	120	60	120	240	330
4	511	240	75	167	315	435
5	1023	480	90	220	390	540
≥6	1023	960	105	277	465	645

3.5 The Proposed Hash Based Distributed Backoff (HBDB) Algorithm

The proposed HBDB algorithm given in Algorithm 1 is used to change CW value whenever there is a collision and choose an appropriate BO value from that CW. On every BO stage, it is found that the CW Sequence obtained by HBDB is less than BEB. This is to provide frequent accessibility of channel to the nodes. This helps to prevent the network from being idle for long time. Hence it minimizes the end-to-end delay.

Algorithm 1: The Proposed HBDB Algorithm

Input: P_c : collision probability
h[r] : hash value
r : random value
MaxTh : Maximum Threshold
MinTh : Minimum Threshold
i : BackOff Stage
Step1: *P_c is calculated using Eq.(3) and stored in the hash table h[r].*
Step2: *Choose a value from h[r],*
if *h[r] > MaxTh* `// high traffic scenario`
then
CW = $(\beta^i) * CW_{min}$ `// exponential increment`
if *h[r]* $\leq$ *MinTh* `// low traffic scenario`
then
CW =$(\beta * i)+1) * CW_{min}$ `// linear increment`
if *(h[r]>MinTh)&&(h[r]* $\leq$*MaxTh)* `// medium traffic scenario`
then
CW=((i+1)$^{\beta}$) $* CW_{min}$ `// polynomial increment`
Step3: *BO = Rand(0, CW-1)*Slot time;*

4 Simulation Parameters

This section presents the details of simulation parameters. All simulation parameters used in HBDB are tabulated in Table 3. The standard 802.11 for wireless networks is available in NS2 which is sufficient for our work. FTP application traffic over TCP is used. There is no configuration parameters for this FTP object. It may send any number of fixed length packets. Nodes in the random topology are randomly placed. All the nodes share the same technical characteristics. The routing algorithm used in HBDB is AODV. Two ray ground reflection model adopted in the proposed algorithm considers both direct path and ground reflection path. It gives more accurate prediction of received power at long distance than free space model. Each simulation was run about 10 times to calculate the confidence interval values.

Table 3: Simulation Parameters

Parameters	**Values**
Propagation Model	Two Ray Ground
Link Bandwidth	2 Mbps
Transmission Range	250m
Routing Protocol	AODV
Transport Layer Protocol	TCP Protocol
TCP Packet Size	512 bytes
TCP Window Size	32
Traffic Pattern	FTP
Access Method	RTS/CTS-DATA-ACK
CW_{min}	15
CW_{max}	960
Minimum Threshold (MinTh)	0.4
Maximum Threshold (MaxTh)	0.6
Simulation Time	100s

5 Simulation and Performance Analysis

This section presents the simulation results of the proposed algorithm. NS2 simulator is used to analyze HBDB's performance. Four way hand shake mechanism is used to avoid the collision of data packets in the receiver side. The control packet RTS is much smaller than data packet. Hence collision of RTS packet does not cause much overhead to the network. The analysis consists of set of nodes forming Adhoc network. Number of node connections for low traffic is 40%, medium traffic is 60% and high traffic is 90% with a linear topology of 10 nodes. Random topology of 30 nodes is considered with low traffic (23% connections), medium traffic (50% connections) and high traffic (90% connections).

5.1 Random Topology

To analyze the performance of HBDB in random topology, different β values are considered to control the CW incrementing rate. As shown in Figures 1-9, HBDB is simulated with $\beta = 7$ (in linear CW increment) and $\beta = 1.5$ (in polynomial CW increment). In another case, β is assigned as 7 for linear CW increment and 1 for polynomial increment.

5.1.1 Throughput

Throughput is the number of packets received in a unit time. It is measured in Kilobits/second. Figures 1, 2 and 3 that depict the throughput performance in high, medium, low traffic scenarios under random topology. By reducing the Contention Window, the waiting time is reduced in HBDB when compared to BEB which leads to more packet transmission avoiding large unnecessary delays. Hence it increases the throughput of HBDB when compared to BEB.

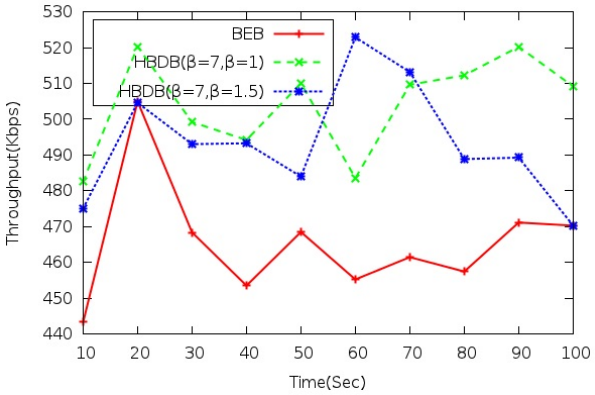

Figure 1: Random Topology : Throughput On High Traffic

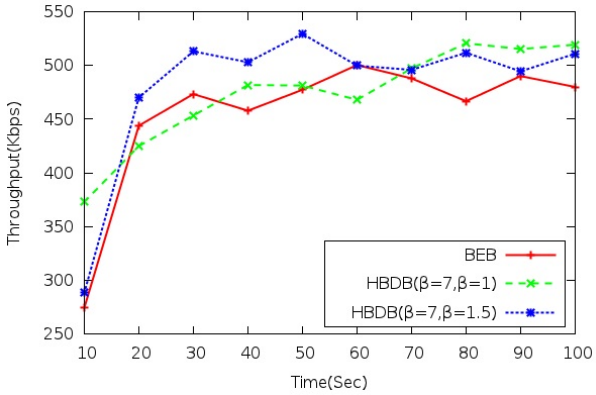

Figure 2: Random Topology : Throughput On Medium Traffic

5.1.2 Packetloss

Packet loss is the number of packets dropped in a unit time during transmission. Even though Contention Window is reduced, choosing waiting time using the collision probability reduces the collision which gives better result for HBDB than BEB. This can be observed from Figures 4, 5 and 6 that depict the packetloss performance in high, medium, low traffic scenarios under random topology.

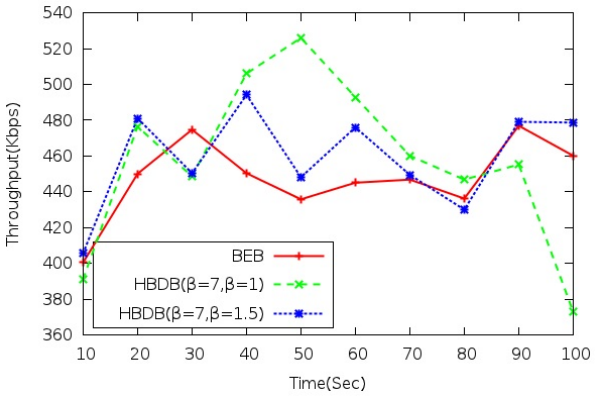

Figure 3: Random Topology : Throughput On Low Traffic

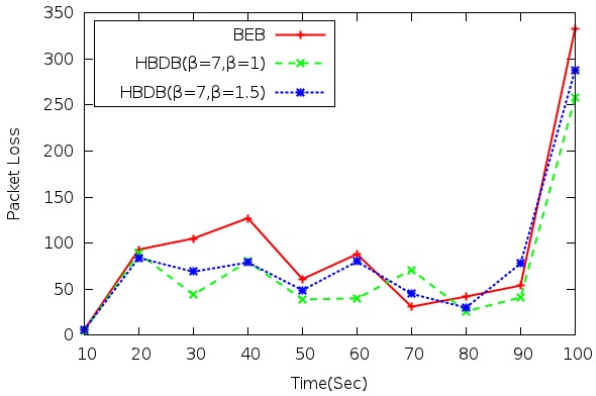

Figure 4: Random Topology : Packetloss On High Traffic

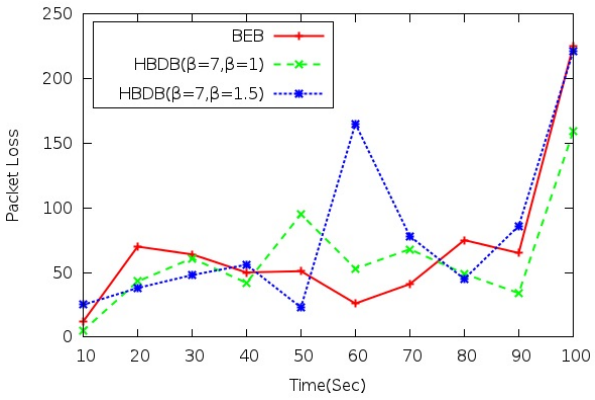

Figure 5: Random Topology : Packetloss On Medium Traffic

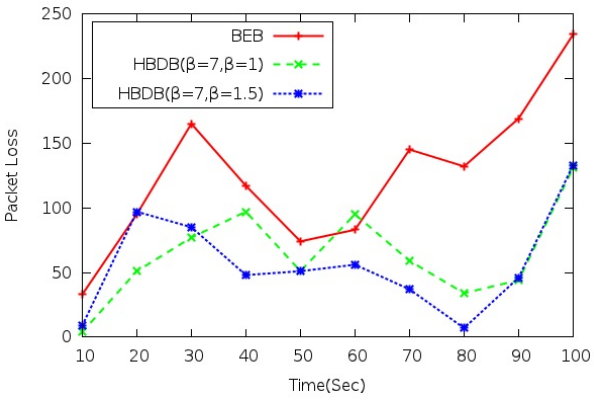

Figure 6: Random Topology : Packetloss On Low Traffic

5.1.3 Latency

Latency is the time taken by the packet to reach its destination. It is measured in millisecond.The latency of HBDB is low compared to BEB in random topology. The results can be observed from the Figures 7, 8 and 9 that depict the latency in high, medium and low traffic scenarios.

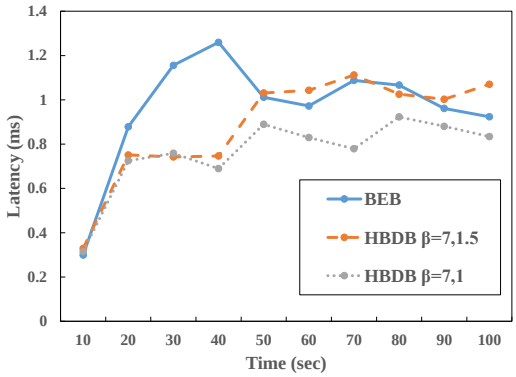

Figure 7: Random Topology : Latency On High Traffic

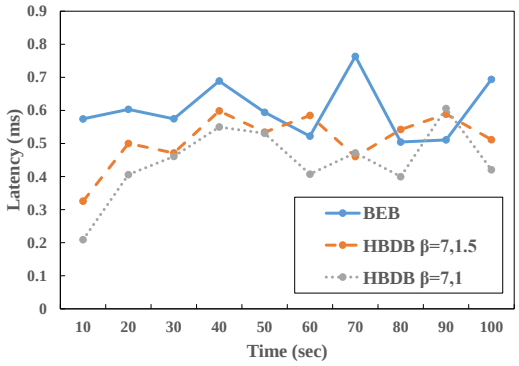

Figure 8: Random Topology : Latency On Medium Traffic

5.2 Linear Topology

The metrics used for comparing linear topology performance are Throughput, packet loss and latency. As shown in Figures 10-18, HBDB is simulated with β = 7(in linear CW increment) and β = 1.5 (in polynomial CW increment). In another case, β is assigned as 5 for linear CW increment and 1.5 for polynomial increment.

5.2.1 Throughput

Throughput of the HBDB is consistent and higher than BEB due to the reduced number of BO slots. These minimum slots give less waiting time than the BEB thereby increases the packet transmission rate.

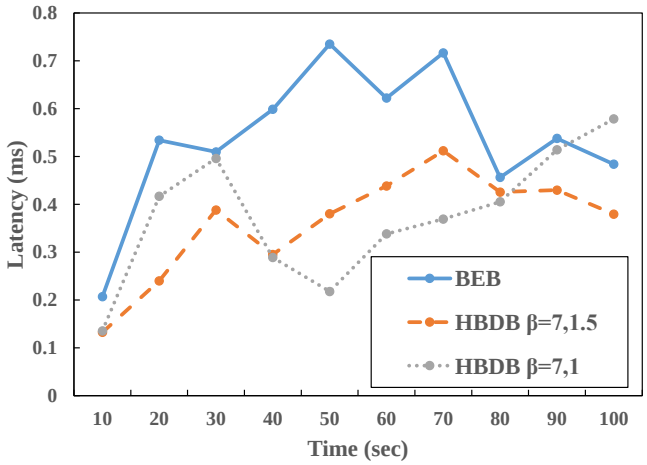

Figure 9: Random Topology : Latency On Low Traffic

Figures 10, 11 and 12 that depict the throughput performance in high, medium, low traffic scenarios under linear topology.

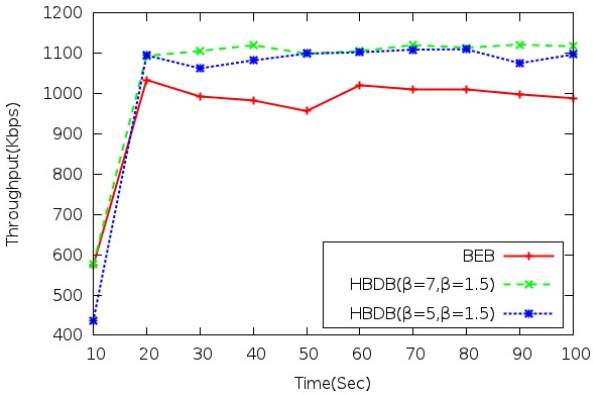

Figure 10: Linear Topology : Throughput On High Traffic

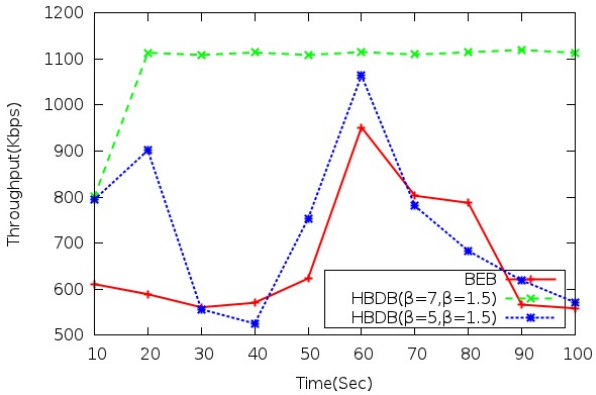

Figure 11: Linear Topology : Throughput On Medium Traffic

5.2.2 Packetloss

Though Packet loss is increased in few instances of time, the possibility of two stations sending at the same time is reduced using probability value chosen from the hash table in the HBDB algorithm. Hence the overall performance of HBDB is high compared with BEB. This is shown in Figures 13, 14 and 15 that depict the packetloss performance in high, medium, low traffic scenarios under linear topology.

5.2.3 Latency

Latency is high only in the low traffic scenario others such as high and medium traffic gives good value for HBDB. This is shown in Figures 16, 17 and 18.

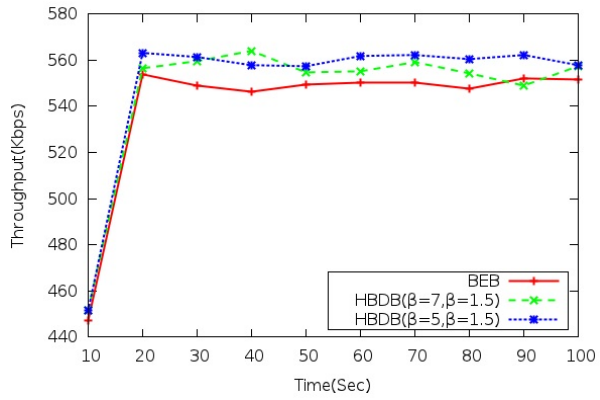

Figure 12: Linear Topology : Throughput On Low Traffic

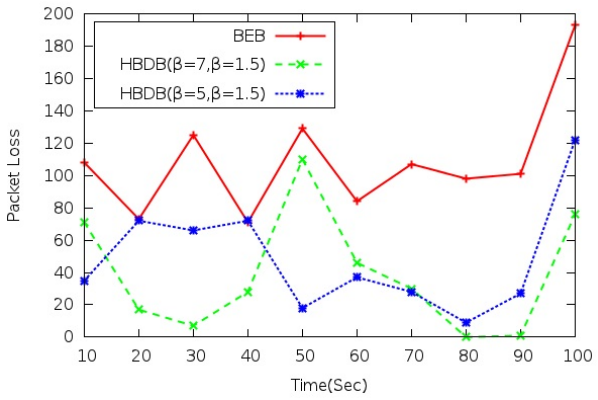

Figure 13: Linear Topology : Packetloss On High Traffic

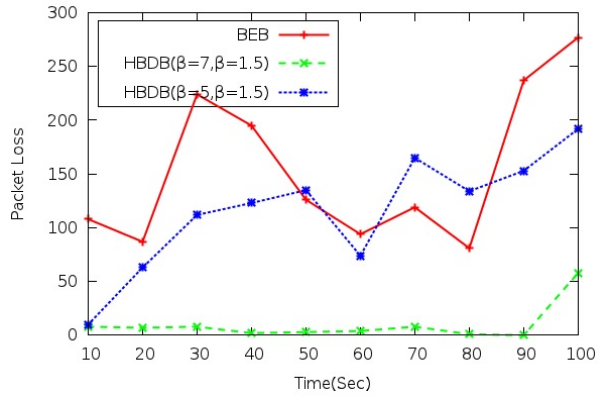

Figure 14: Linear Topology : Packetloss On Medium Traffic

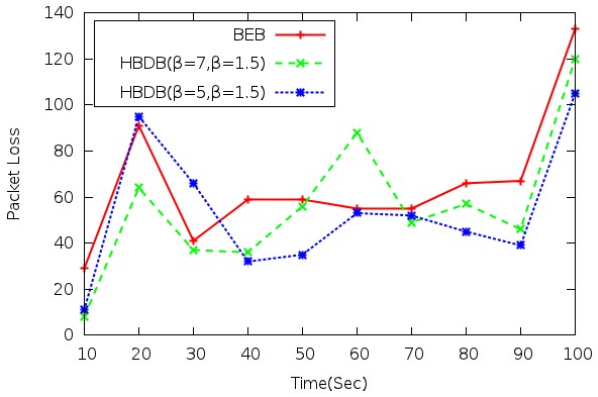

Figure 15: Linear Topology : Packetloss On Low Traffic

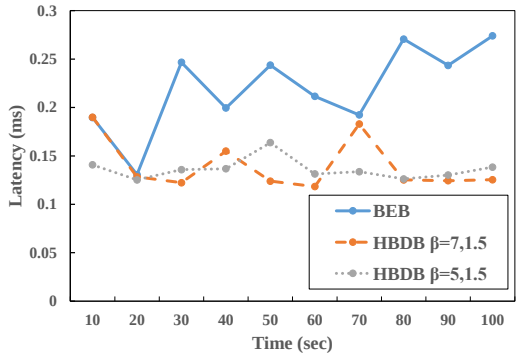

Figure 16: Linear Topology : Latency On High Traffic

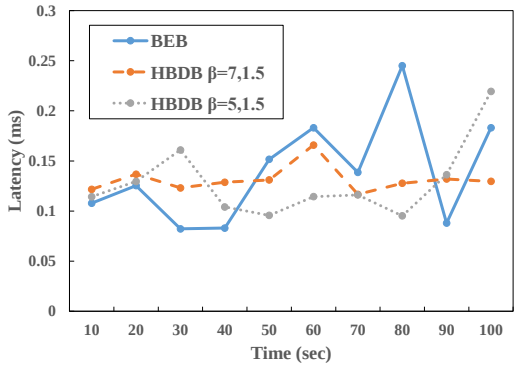

Figure 17: Linear Topology : Latency On Medium Traffic

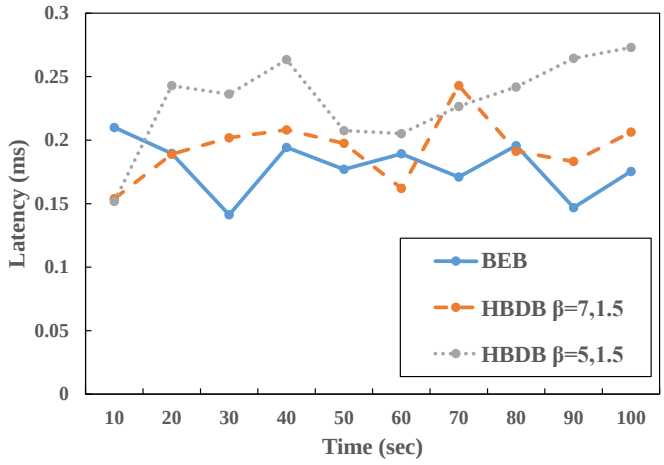

Figure 18: Linear Topology : Latency On Low Traffic

5.3 Individual Node Throughput, Packetloss and Latency Comparison

Some of the nodes are taken to calculate the throughput, packetloss and latency for HBDB and BEB. Even though the throughput of individual nodes is high for some nodes and low for some nodes, overall throughput of HBDB is high than BEB. Same is the case for packetloss and latency. The results can be seen from Figures 19 to 30.

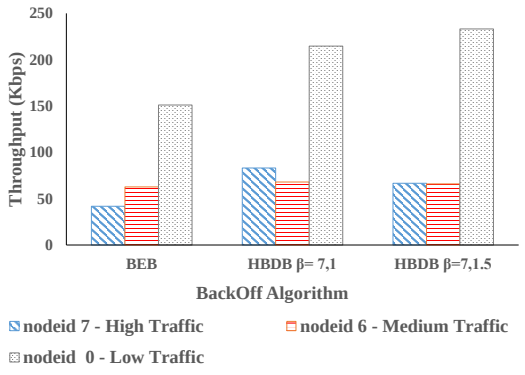

Figure 19: Random Topology - High Throughput for HBDB

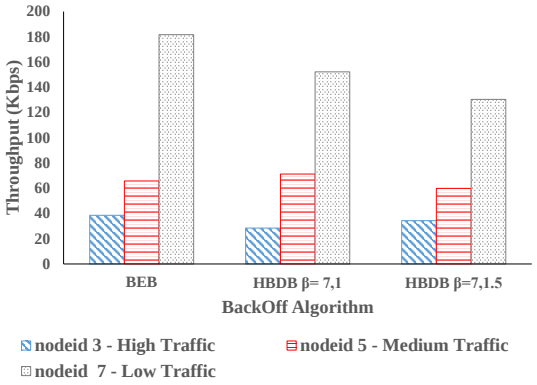

Figure 20: Random Topology-Low Throughput for HBDB

5.4 Comparison of Mean Throughput, Packetloss and Latency

95% confidence level is chosen to calculate the confidence interval for throughput, packetloss and latency. Mean and confidence interval of throughput, packetloss and latency of both the algorithms are tabulated in Table 4 and 5. Figures 31 to 36 depict the comparison of mean values between BEB and HBDB.

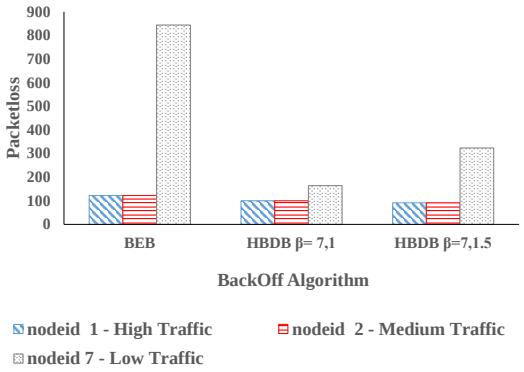

Figure 21: Random Topology-Low Packetloss for HBDB

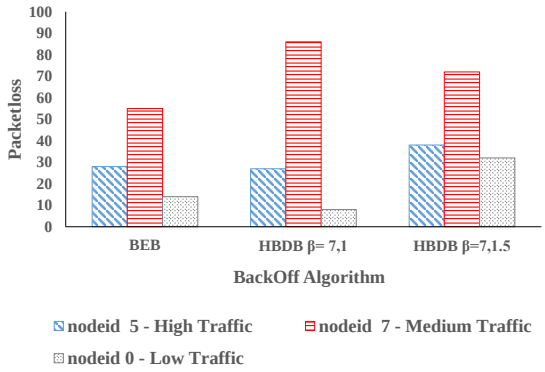

Figure 22: Random Topology-High Packetloss for HBDB

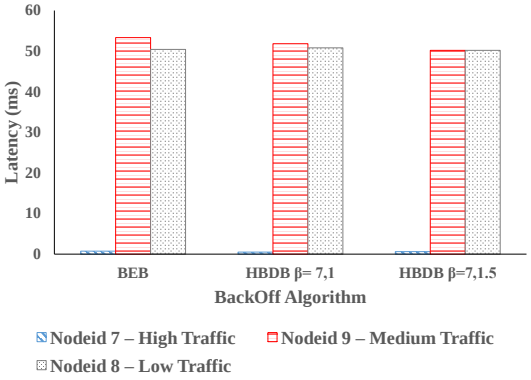

Figure 23: Random Topology-Low Latency for HBDB

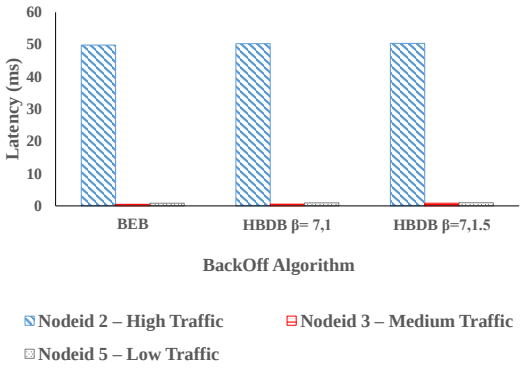

Figure 24: Random Topology-High Latency for HBDB

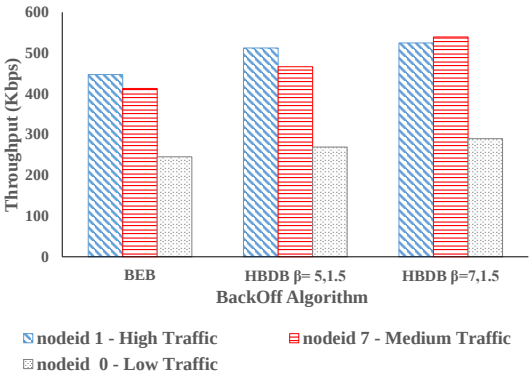

Figure 25: Linear Topology - High Throughput for HBDB

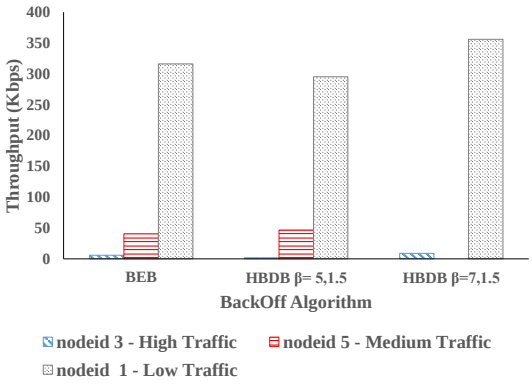

Figure 26: Linear Topology-Low Throughput for HBDB

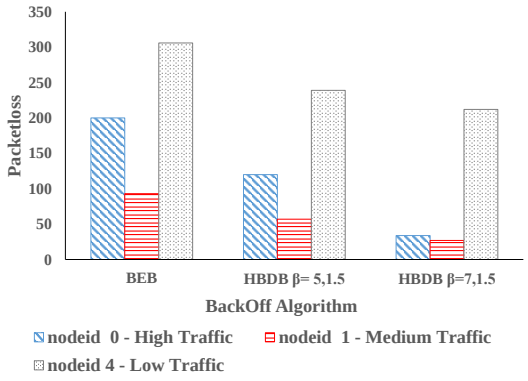

Figure 27: Linear Topology-Low Packetloss for HBDB

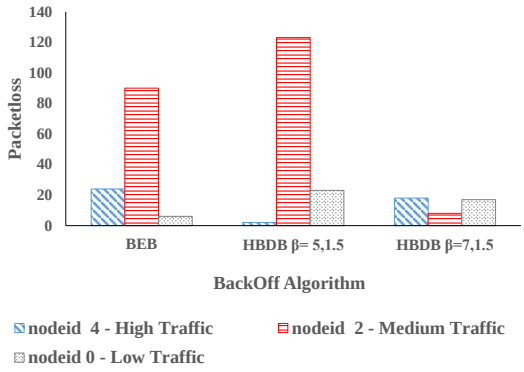

Figure 28: Linear Topology-High Packetloss for HBDB

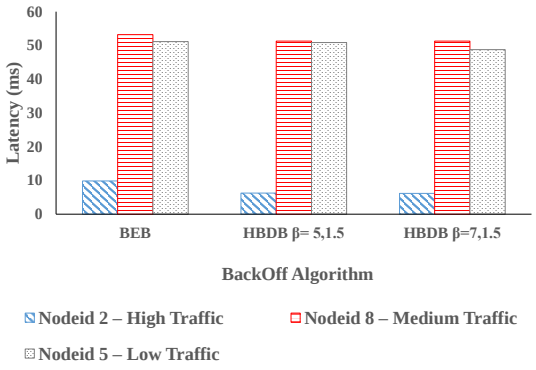

Figure 29: Linear Topology-Low Latency for HBDB

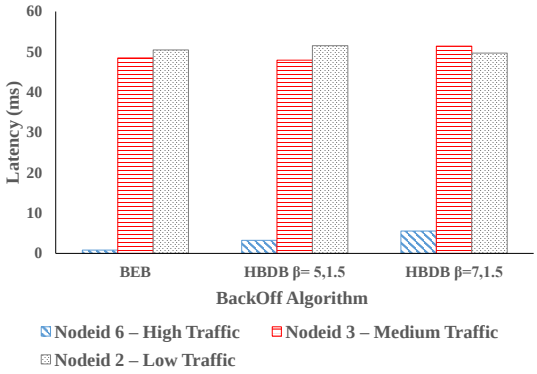

Figure 30: Linear Topology-High Latency for HBDB

Table 4: Confidence Interval for Throughput, Packetloss and Latency in Linear Topology

BEB								
Throughput			Packetloss			Latency		
Traffic	Mean	Confidence Interval	Traffic	Mean	Confidence Interval	Traffic	Mean	Confidence Interval
High	899	±78.65	High	703.9	±66.07	High	0.211	±0.01
Medium	656	±27.89	Medium	731	±96.67	Medium	0.133	±0.01
Low	539.7	±0.3	Low	133.9	±8.99	Low	0.169	±0.01
HBDB								
Throughput			Packetloss			Latency		
Traffic	Mean	Confidence Interval	Traffic	Mean	Confidence Interval	Traffic	Mean	Confidence Interval
High	958.5	±27.89	High	654.6	±153.09	High	0.199	±0.02
Medium	728.8	±85.53	Medium	637.8	±145.4	Medium	0.141	±0.01
Low	543.1	±0.68	Low	161	±13.02	Low	0.21	±0.01

Table 5: Confidence Interval for Throughput, Packetloss and Latency in Random Topology

BEB								
Throughput			Packetloss			Latency		
Traffic	Mean	Confidence Interval	Traffic	Mean	Confidence Interval	Traffic	Mean	Confidence Interval
High	512	±10.51	High	277.5	±53.61	High	0.488	±0.1
Medium	424.5	±20.76	Medium	206	±33.02	Medium	0.279	±0.07
Low	471	±30.97	Low	150.5	±27.19	Low	0.218	±0.1
HBDB								
Throughput			Packetloss			Latency		
Traffic	Mean	Confidence Interval	Traffic	Mean	Confidence Interval	Traffic	Mean	Confidence Interval
High	515.7	±6.73	High	293.6	±24.3	High	0.46	±0.09
Medium	442.6	±19.77	Medium	228	±25.29	Medium	0.25	±0.06
Low	442	±16.11	Low	195	±101.65	Low	0.27	±0.04

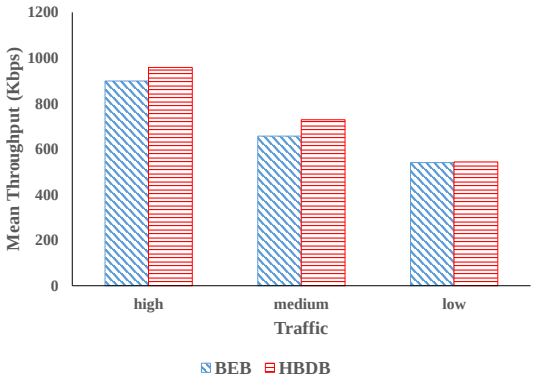

Figure 31: Linear Topology-Comparison of Mean Throughput

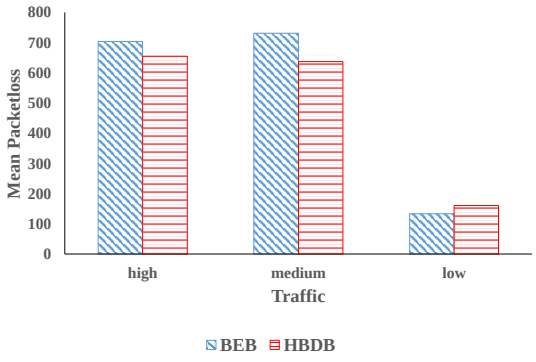

Figure 32: Linear Topology-Comparison of Mean Packetloss

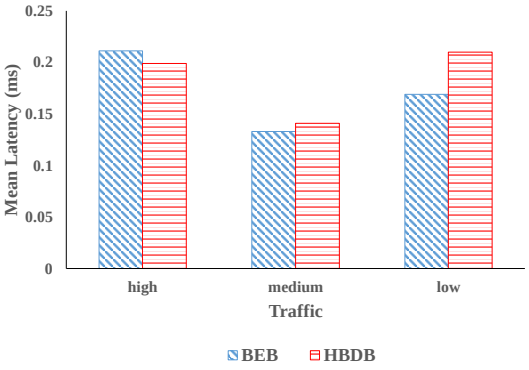

Figure 33: Linear Topology-Comparison of Mean Latency

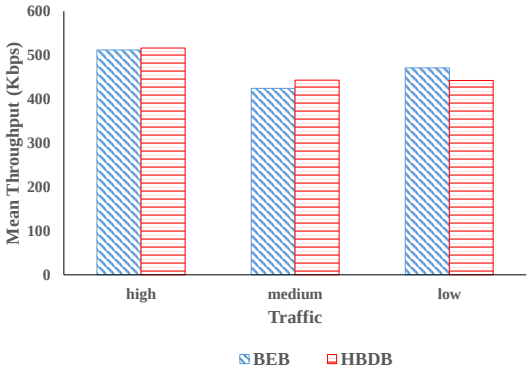

Figure 34: Random Topology-Comparison of Mean Throughput

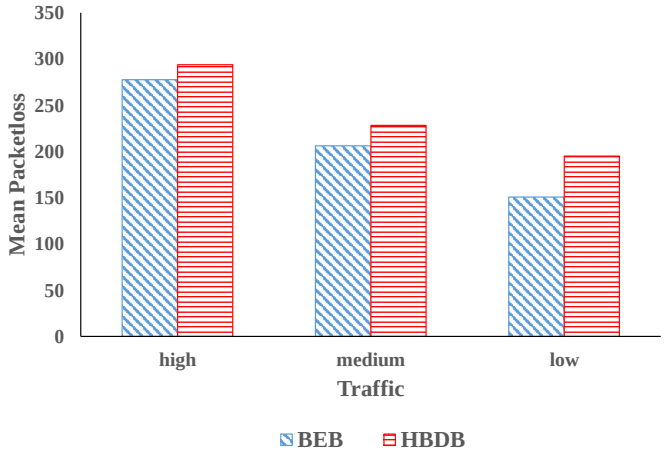

Figure 35: Random Topology-Comparison of Mean Packetloss

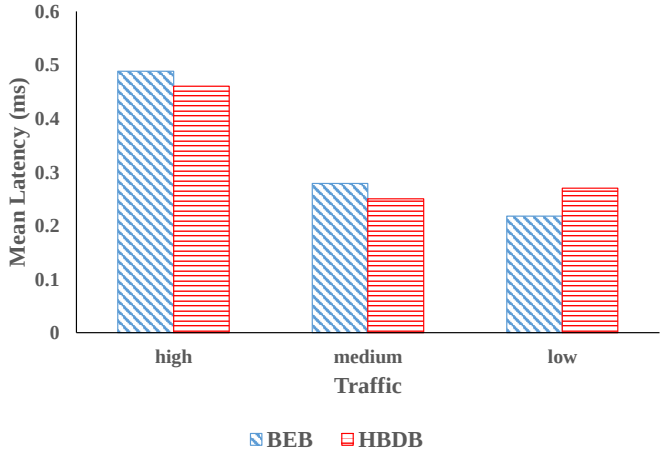

Figure 36: Random Topology-Comparison of Mean Latency

6 Conclusion

The proposed HBDB algorithm increases the throughput by effectively applying linear or polynomial or Exponential CW increment mechanisms. This selection is made based on the collision probabilities which depict the current status of the network. To make sure that the same contention window increment is not repeatedly used, linear probing is used to choose unique collision probability value. The performance of the algorithm has been validated by NS2 simulation with linear and random topologies under low, medium and high traffic scenarios. The simulation results show that better throughput is achieved by HBDB algorithm than BEB.

References

[1] D. P. Agrawal and Q.-A. Zeng. *Introduction to Wireless and Mobile Systems*. Cengage Learning.

[2] R. Ahmad, M. Hasna, and A. Abu-Dayya. Two Way Opportunistic Mac Protocol for Adhoc Networks. In *Proc. of the 2011 GLOBECOM Workshops, Texas, USA*, pages 497 – 502. IEEE, December 2011.

[3] T. Alekya, B. Mounika, E. Jyothi, and B. Bhandari. A Waiting-time based BackOff Algorithm in the IEEE 802.11 based Wireless Networks. In *Proc. of the 2012 National Conference on Communications (NCC'12), Kharagpur, India*, pages 1–5. IEEE, February 2012.

[4] A. Balador, A. Movaghar, S. Jabbehdari, and D. Kanellopoulos. A Novel Contention Window Control Scheme for IEEE 802.11 WLANs. *IETE Technical review*, 29(3):202–212, May 2012.

[5] M. Barbeau and E. Kranakis. *Principles of Ad Hoc Networking*. Wiley.

[6] H. Ferdous and M. Murshed. Analytical Modelling of Enhanced IEEE 802.11 with Multiuser Dynamic OFDMA under Saturation Load. In *Proc. of the 2010 Wireless Telecommunications Symposium (WTS'10), Florida, USA*, pages 1–6. IEEE, April 2010.

[7] Jochen.H.Schiller. *MobileCommunication*. Pearson Education.

[8] D. J. Kadhim, S. H. Abdulhussain, B. M. Ridha, and A. M. Abbas. A Balanced Backoff Algorithm for IEEE 802.11 wireless network. *Iraqi Journal of Applied Physics*, 8(1):27–33, January 2012.

[9] C.-Y. Kuo, Y.-H. Huang, and K.-C. Lin. Performance Enhancement of IEEE 802.11 DCF using Novel BackOff Algorithm. *EURASIP Journal on Wireless Communications and Networking*, 2012(1):274–285, December 2012.

[10] Y. Lee, J. Yun, S. Hwang, G. Seong, K. Lee, B. Kim, and K. Han. A Traffic Adaptive Backoff Approach for Wireless Networks. In *Proc. of the 5th International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services (CENTRIC'12), Lisbon, Portugal*, pages 34–37. Think Mind, November 2012.

[11] P.Sumathi and N.C.Sumathi. Backoff Schemes for Mobile Adhoc Networks-A Survey. *Journal of Wireless Networking and Communications*, 3(1):1–5, January 2013.

[12] Y. Xu, M. Huang, M. Lin, and Y. Zheng. A Self-adaptive Minimum Contention Window Adjusting Backoff Algorithm in IEEE 802.11 DCF. In *Proc. of 2nd International Conference on the Consumer Electronics, Communications and Networks (CECNet'12), Hubei, China*, pages 1577 – 1582. IEEE, April 2012.

[13] M. O. B. Yassein, S. S. Manaseer, and A. A. Momani. Adaptive Backoff Algorithm for Wireless Internet. *Journal of Emerging Technologies in Web Intelligence*, 4(2):155–163, May 2012.

Author Biography

M. Devipriya is currently pursuing MS degree in Wireless Networks at the Department of Computer Science and Engineering, National Institute of Technology, Trichy, India.

B. Nithya received Ph.D degree from National Institute of Technology, Trichy in 2015. She is working as an Assistant Professor in National Institute of Technology (NIT) Trichy since 2007. Her research interests include Wireless Networks, Mobile Computing and Optimization Techniques.

C. Mala is currently serving as an Associate Professor in the Department of Computer Science and Engineering, National Institute of Technology, Trichy, Tamilnadu, India. She received Ph.D from National Institute of Technology, Trichy in 2008. Her research interests include Wireless Networks, Parallel Algorithms, Network Security, Soft Computing and Image processing.