HIPERSIM: A Sense Range Distinctive Simulation Environment for HiperLAN Networks

G. I. Papadimitriou **T. Lagkas** **A. S. Pomportsis**

Department of Informatics, Aristotle University, Box 888, 54124 Thessaloniki, Greece.

Abstract

This paper presents the simulator "HIPERSIM" which was developed and used to examine the behavior of HIPERLAN. HIPERSIM simulates the HIPERLAN network under various conditions, assuming that the communication range and the sense range (signal detection range) of a node are different. In a wireless LAN, like HIPERLAN, the medium access protocol, the hidden nodes, the packet forwarding and the power saving mechanism are important issues that affect significantly the overall performance. The intention is to provide a HIPERLAN specialized tool (HIPERSIM) which can simulate most of the features of this WLAN protocol, in order to examine the performance of HIPERLAN. The main focus is to simulate the channel access mechanism accurately, so as to examine the effectiveness of the EY-NPMA protocol, and underline its advantages and the elements that need improvement. Also, in HIPERSIM, there is emphasis on the "hidden nodes" issue. More specifically, we distinguish between the communication range and the signal detection range of a node, since this is a characteristic of the wireless nature and it affects the network operation. The simulation results show that EY-NPMA is effective and suitable for WLANs. Probably there could be some improvement in order to avert the collisions close to the receiver.

Keywords: HIPERSIM, HIPERLAN, simulator, simulation, WLAN, wireless, sense range

1. Introduction

Lately, there is a significant growth in the area of the wireless communication networks. There is great emphasis on the wireless local area networks [WLAN]. The need for mobility and "stay connected" has led to the development of WLAN standards in Europe (ETSI: HIPERLAN) and in the USA (IEEE 802.11). Their main intention is to expand the range of the fast backbone wired networks, and enhance the wide area wireless communication networks (such as the cell networks). A full replacement of wired networks by wireless networks is at the moment out of scope.

HIPERLAN (HIgh PErformance Radio Local Area Network) is a standard for wireless LANs that was defined by the European Telecommunications Standards Institute [ETSI]. HIPERLAN is a rather short range WLAN (~50m), it supports slow moving stations (1.4 m/s), it can be an infrastructured or an ad hoc network, its high transmission rate is 23.529Mbps, and it operates at 5GHz.

An important characteristic of HIPERLAN is that it can support a variety of services. It combines asynchronous communication, such as file transfer, with time bounded communication, such as voice and video transmission. This is due to the fact that the Medium Access Control protocol supports Quality of Service [QoS], aided by the Channel Access Mechanism [CAM] that assigns the packet priorities.

The need to evaluate the HIPERLAN standard before it eventually reaches the market has arisen. So, in order to analyze the performance and the behavior of HIPERLAN under various operation conditions, a full parameterized simulator, called HIPERSIM, was developed in C++. HIPERSIM has the ability to distinguish hidden nodes between those that are in sense range (the signal can be detected) and those that are out of sense range. Mainly because of the complexity of the medium access mechanism and in order to have as fewer simplifications as possible, the simulator operation is almost exhaustive. This means that at every time unit each node is examined separately. The duration of a time unit is 1 high rate bit-period, that is the time interval required to transmit a bit in maximum bit-rate. The end of the simulation is defined by the simulation time, which is set by the user in msecs. The user is able to parameterize the simulator in order to represent a specific network environment. HIPERSIM was developed in the W32 platform and it uses the respective Graphical User Interface in order to be easy to use. The base of the application structure is object oriented. The results of every simulation and the current values of the parameters are automatically saved in a database file, by using the ADO mechanism. Every record stores the values of the parameters and the results of the corresponding simulation.

2. The HIPERLAN Network

2.1. The Channel Access Mechanism Priority

The QoS support is based on two mechanisms: the user priority assigned to a packet and the lifetime of the latter. The user application sets the user priority of a packet at the value low or high, according to its importance and the need for fast transmission. For example, during a file transfer the packets can be set at “user priority = low”, while during a video transmission it is preferred to set user priority at “high”.

When a packet is generated, it is assigned a lifetime. If the packet does not reach its destination within the specific time interval, then it is assumed to be invalid and it is rejected. The default value of the packet lifetime is 500msecs. According to the kind of the transmission and the offered service, the lifetime may take small values (synchronous transmission of voice - video) or large values (file transfer). During the lifetime of a packet the residual lifetime is continuously updated [Figure 1]. When the value of the residual lifetime becomes 0, the packet is rejected. The user priority and the residual lifetime of the packet, give its CAM priority. By default there are five values – levels of CAM priorities (0 – 4), where the highest priority is the one with the smallest value. In Figure 2, the CAM priority values are shown, according to the user priority and the residual lifetime. [1,2,3]

2.2. Queuing Discipline : Earliest Deadline First

Every node has a buffer where the packets to be sent are stored. In HIPERLAN, the mechanism that selects a packet from the buffer is based on the Earliest Deadline First (EDF) queuing discipline. According to it, the necessary steps in order to select a packet to be sent are: The packet that carries the highest (arithmetically minimum) CAM priority is selected. If there are more than one packets of the same highest CAM priority, then the packet having the minimum RML is selected. In case there are more than one packets having the same minimum RML, then the packet carrying the highest (arithmetically minimum) User Priority is selected. If there are still more than one packets of the same highest User Priority, then one of them is selected randomly.

It becomes obvious that this packet selection mechanism eventually favors the packets of the highest priority, which must be sent earlier than the others [Figure 3]. This method contributes in supporting different kinds of communication (asynchronous and time bounded). The HIPERSIM results show the efficiency of the above mentioned packet selection mechanism in contrast to the poor performance of the classic method "FIFO".[1,2]

2.3. The Medium Access Protocol : EY-NPMA

The medium access protocol that is used in HIPERLAN is the Elimination Yield Non-pre-emptive Priority Multiple Access (EY-NPMA) protocol. A problem related to the wireless networks is that a node is not always able to detect the transmission from another node, because of the limited communication and sense range. EY-NPMA defers from the conventional MAC protocols used in the wired networks, because it is based on active signaling instead of passive signaling. Active signaling is a channel contesting method that reduces the collisions caused in the wireless environment. The medium access mechanism is based partially on the well known CSMA, since the node eventually "listens" to the medium to find out if it can transmit or not. However, the main mechanism differs from CSMA, since every node contests for the medium access in an active way by transmitting some special signals. It has been proved that active signaling has a better performance than passive signaling, especially when the number of nodes increases [4]. Below, the operation of the EY-NPMA protocol is presented, according to the "ETSI functional specification EN 300 652 v1.2.1" [1].

EY-NPMA operates in channel access cycles. Every node that has a packet to send initially senses the channel to find out whether it is idle. If the node does not "listen" any transmission for a time interval equal to 2000 high rate bit-periods, it enters the "channel free condition" and transmits the packet. One high rate bit-period is the time interval required to transmit one bit in high transmission rate, that is the maximum transmission rate 23.529Mbps. In case the channel is not idle and a data transmission is detected, all stations that want to transmit a packet synchronize at the end of the expected acknowledgement packet. At that time, these stations enter the "synchronized channel condition". The synchronized channel access cycle consists of three phases, which define the access pattern for the competitive nodes. These are described below. [1,4,5]

2.3.1 Prioritization Phase

In the first phase of the synchronized channel access cycle, known as the Prioritization Phase, every node allows a number of idle slots, where the default slot length is 168 high rate bit-periods. The number of the idle slots is equal to the arithmetic value of the CAM priority of the packet. Every contending node senses the channel, while it allows the idle slots. If it detects a signal transmission, it defers, that is it quits the effort to gain access to the channel and waits for the next channel access cycle to try to transmit. When a node detects no transmission during the Prioritization Phase, it transmits a pulse right after the idle slots, and proceeds to the next phase. This pulse is the one listened by every "defeated" node. The nodes that proceed to the next contention phase have a packet to send of the same highest CAM priority. In Figure 4, it is shown how node B "wins" during the Prioritization Phase and makes node A to defer. CAM priority of node A packet is 2, while CAM priority of node B packet is 1 (higher priority).

2.3.2 Elimination Phase

The nodes that "survive" the Prioritization Phase keep on trying to gain access to the channel. The objective of this medium access mechanism is to eliminate as more contending nodes as possible, but of course not all of them. During the Elimination Phase, a great percentage of the contending nodes is eliminated, but at least one of them survives. Every node that has not been defeated during the Prioritization Phase transmits an elimination pulse which is actually the lengthening of the priority pulse. Every node lengthen its pulse independently of the others according to a geometric distribution of probability p = ½. So, the probability the pulse is longer than 1 slot is 1 / 2, longer than 2 slots is 1 / 4 and so on. However, there is a limit for the length of the elimination pulse. Right after the end of this pulse, the nodes allow an idle slot, which is called survival verification slot, during which they sense the channel. If a node detects a transmission during this time interval, this means that the specific node is "defeated", so it defers. Thus, the nodes that survive the Elimination Phase carry the packets of the highest priority and they have transmitted the longest elimination pulse. In Figure 5, it is shown how node B "wins" during the Elimination Phase and makes node A to defer. Node A transmits an elimination pulse 1 slot long, while node B transmits an elimination pulse 2 slots long.

2.3.3. Yield Phase

The Yield Phase is the last phase before the transmission of a data packet and it is the last effort to reduce the number of the contending nodes. The nodes that have survived the Elimination Phase enter the Yield Phase allowing a number of idle slots. Every node that detects transmission during these slots quits the current effort to gain access to the channel and waits till the next channel access cycle. If a node detects no transmission, it eventually transmits its data packet. Thus, a node "loses" in Yield Phase, when it listens some other node transmitting a data packet. The number of the idle slots is random and uniformly distributed between 0 and 9. In Figure 6, it is shown how node B "wins" during the Yield Phase and makes node A to defer. Node A allows 3 idle slots, while node B allows 2 idle slots.

An overview of the EY-NPMA protocol is shown in Figure 7. Conclusively, EY-NPMA protocol is appropriate for use in a wireless LAN, although it is rather complicated. HIPERSIM tries to simulate it accurately, that is why the simulation mechanism is almost exhaustive.

2.4. Range Issues

2.4.1 The Communication and Sense Range Distinction

In a WLAN, not all of the nodes are expected to be able to communicate directly with each other. This is due to the fact that the antenna of a node has a limited range, and the obstacles that might intercept the communication. Specifically, the range of HIPERLAN is approximately 50 meters when the nodes are moving slower than 1.4m/s.

In the wired networks there can actually be only two cases regarding communication range: In the first case, two nodes are directly connected and they are able to communicate directly with each other. In the second case, the two nodes are not directly connected with a cable, so they are not able to communicate directly or even to detect each other. In this case, there might be a forwarder between these two nodes, but there is no way to physically "listen" directly to each other. However, in WLANs, there can be a third case regarding communication range. According to this case, two nodes are not able to communicate directly, but they can sense the signal transmitted by each other. This is a special characteristic of the wireless nature.

More specifically, in HIPERLAN two nodes are able to exchange data packets when they are able to detect the transmitted signal and identify the bits sent. This happens when the signal attenuation due to distance and the signal fading are not intense enough to make the communication between the two nodes impossible. In this case, the two nodes are assumed to be in communication range. When the quality of the received signal is not good enough to allow data transmission, but it can still be detected, then the two nodes are assumed to be in sense range (signal detection range). Obviously, the sense range is greater than the communication range, since the former includes the latter, so it is likely that in a WLAN two nodes can detect each other even when they are not able to communicate directly. HIPERSIM distinguishes between these two kinds of ranges, when it simulates the hidden nodes. [6]

2.4.2. The Hidden Nodes

The “hidden nodes” issue is important for the wireless LANs including HIPERLAN. The hidden nodes are pairs of nodes where the one cannot receive data from the other, because of the distance or the obstacles between them. They cause overall reduction of the system performance. The range issues mentioned before are related to the “hidden nodes”. We assume that two nodes are hidden when they are out of communication range. Two hidden nodes might be in sense range or not. The HIPERSIM results show that a great reduction of the HIPERLAN performance is caused by hidden nodes that are out of sense range. In Figure 8, we can see three nodes of a HIPERLAN network. Every Ci area represents the communication range (data reception range) for node I and every Si area represents the signal detection range (sense range) for node I. As we can see, node B is able to send and receive data from nodes A and C, while nodes A and C are hidden from each other, that is they are not in communication range. All three nodes are able to detect the signal of the others. In general, EY-NPMA takes advantage of the fact that two hidden nodes in sense range are able to detect the transmitted signal. [3,7]

2.4.3. Packet Forwarding

There are two types of nodes in HIPERLAN: the forwarders and the non-forwarders. The non-forwarders know only their direct neighbors, that is the nodes which are in communication range, while the forwarders are aware of the network topology. In case a non-forwarder wants to transmit a packet to a node that is not in communication range (hidden node), it sends it to a forwarder, by setting the latter as the intermediate node of the transmission. The packet forwarding in HIPERLAN is based on a table-driven routing protocol. This forwarding mechanism increases the system complexity, since it requires the continuous watch of the network topology which changes dynamically. In Figure 9, three nodes (A, B, C) of a HIPERLAN network are represented, where nodes A and B are in communication range, nodes B and C are in communication range, while nodes A and C cannot communicate directly with each other. Nodes A and C constitute a hidden pair, while node B is a forwarder, that is it forwards packets from node A to B and vice-versa. The packet forwarding mechanism is simulated in HIPERSIM. [1,7,8]

2.5. Power Saving

The nodes of a WLAN are wireless mobile devices that use batteries with limited energy. Thus, the objective is to minimize power consumption in the wireless networks, so that to ensure the highest energy autonomy of the mobile devices. The process of an antenna transmitting or receiving signal is high energy consuming, so a possible solution to the power problem is to turn the antenna off when there is no data transmission or reception. In HIPERLAN, some nodes are P_Savers, while some others are P_Supporters. P_Savers are set at status "OFF" for specific time intervals. During these time intervals, they are not able to receive data packets. P_Supporters have the responsibility to collect the data packets that have as destination a P_Saver which is "OFF". When the P_Saver returns to normal operation status, the P_Supporters forward the packets to it. Power saving is optional and it is not fully defined by the HIPERLAN protocol. The HIPERLAN products manufacturers are free to decide the method they will use to support power saving. The HIPERSIM simulates the power saving mechanism and shows that it has satisfactory results.[1,8]

3. The HIPERSIM Simulator

3.1. Environment Description

Figure 10 shows the environment of HIPERSIM.

In this image we can see the parameters that are set by the user, the simulation results, the "start" button that starts or ends the simulation, the "about" button that shows some information about the developer, and the simulation progress bar.

The most important of the HIPERLAN features that are simulated by HIPERSIM were presented in the previous section. The HIPERSIM simulation engine is based on the "ETSI functional specification EN 300 652 v1.2.1" for HIPERLAN. In the next two subsections, the parameters that are set by the user, and the simulation results are presented one by one.

3.1.1 HIPERSIM Parameters

- *Simulation Time (msecs):* This is the "virtual" time duration of the simulation in msecs.
- *Number Of Nodes:* This is the total number of nodes that constitute the LAN.
- *Number Of Forwarders:* This is the number of the nodes that work as forwarders. When the simulation starts, the forwarders are chosen randomly. We assume that every forwarder can communicate with all the nodes of the LAN, it is aware of the overall topology and it is not hidden from anyone. Any forwarder can be P_Supporter at the same time, but it is definitely not a P_Saver. Any non-forwarder node can use any forwarder at any time needed.
- *Number Of P_Supporters:* This is the number of the nodes that work as P_Supporters. When the simulation starts, the P_Supporters are chosen randomly. We assume that every P_Supporter can communicate with all the nodes of the LAN, it is aware of the overall topology and it is not hidden from anyone. Any P_Supporter can be forwarder at the same time, but it is definitely not a P_Saver. Any P_Saver can use any P_Supporter at any time needed.
- *Number Of P_Savers:* This is the number of the nodes that work as P_Savers. When the simulation starts, the P_Savers are chosen randomly, satisfying the limitations that were mentioned above relatively to the forwarders and the P_Supporters. There cannot be any P_Saver, if there is no P_Supporter.
- *Probability Of Hidden Pair:* This is the probability two nodes are hidden from each other, that is they cannot receive data directly from each other (out of communication range). When the simulation starts, the pairs of hidden nodes are chosen randomly, satisfying the limitations that were mentioned above relatively to the forwarders and the P_Supporters. The user can choose a value for this parameter between 0 and 1 (step 0.01). When there are forwarders in the LAN, the hidden nodes that want to communicate with each other communicate through a randomly chosen forwarder. If there are no forwarders, the hidden nodes never even try to communicate with each other during the simulation.
- *Queue Discipline:* It concerns the method used by every node for the management of the packet buffer. There are two options: Earliest Deadline First (EDF) and First In First Out (FIFO). EDF is the default option, as it is defined by the HIPERLAN standard.

- *Packet Arrival Method:* This is the way that the packet generation takes place in every node. HIPERSIM uses the value of this parameter to calculate the time of the next packet arrival. The packet arrival can be Poissonian or bursting.
- *Poisson Parameter (packets per msec):* This parameter is enabled only when the Poisson packet arrival method is selected. It defines the packet generation rate in every node, that is the number of packets generated per msec. When the value of this parameter increases, the load of the network increases as well.
- *Average Burst Length:* This parameter is enabled only when the Bursting packet arrival method is selected. It defines the average number of packets generated the one after the other (per 1 msec), creating by this way a burst. In general, we can say that the bigger the burst length is, the less the times a burst appears [9].
- *Packet Buffer Max Size (Kbits):* This is the size of the buffer that keeps all the packets to be sent in a node. If a packet arrives and it must be stored in the buffer but does not fit, it is rejected.
- *Bit Error Rate % (after check):* This is the rate a bit error occurs, after the appliance of all the error detection and error correction checks that are defined by the HIPERLAN standard. Obviously, after all these checks, the probability of a bit error decreases significantly, that is why the default value of this parameter is so low. The simulator uses the value of the packet size and the Bit Error Rate to decide whether a bit error will occur in the transmitted packet, and if it does, then the packet is assumed invalid.
- *Average Packet Size (bits):* This is the average size of the generated packets, which measures in bits. The packet size is calculated according to the exponential distribution. The value of this parameter is the average value of this distribution.
- *Max Packet Life Time (msecs):* When a packet is generated, its MPDU Lifetime (ML) is set. This is the time interval that the packet is valid. If this time interval expires before the packet reaches its destination, then the packet is rejected. The value of the ML is chosen randomly between 10 and the value of this parameter.
- *Packet User Priority:* This is the user priority that every packet carries. The user’s options are: a) high , b) low or c) random User Priority for all the packets.

- *Power On Period (msecs):* This parameter is enabled only when there are P_Savers in the simulated network. It defines the time interval during which a P_Saver works as a common node, which means that it is able to send and receive packets.
- *Power Off Period (msecs):* This parameter is enabled only when there are P_Savers in the simulated network. It defines the time interval during which a P_Saver is in power saving mode, which means that it is able only to send but not receive packets.
- *Add Control Bits:* The user chooses to add extra control bits, headers etc in every generated packet or not. Obviously, when the value of this parameter is "TRUE", the real average packet size, that is the size of the packets when they are transmitted, is rather greater than the value of the parameter "Average Packet Size (bits)".
- *Probability Of Sensing Hidden Signal:* It defines the probability two hidden nodes are in sense range. When the simulation starts, the pairs of hidden nodes, which are in sense range, are randomly selected, according to the value of this parameter.

3.1.2 HIPERSIM Results

- *Successfully Transmitted Packets:* This is number of the packets that were successfully transmitted, which means that they reached their destination, during the simulation.
- *Successfully Transmitted Bits:* This is the number of the bits that were successfully transmitted, which means that they reached their destination, during the simulation.
- *Rejected Packets:* This is the number of the packets that were rejected because their lifetime (ML) was over before their successful transmission or because there was not enough space for them in the packet buffer in the node.
- *Load (packets):* This is the number of the packets generated in the system.
- *Load (Kbits):* This is the number of the *Kbits* generated in the system.
- *Load (percentage of bitrate):* The bitrate is the maximum transmission rate at 23.529Mbps. The value is the result of the operation [Load / 23.529 * 10^6], assuming Load in bits.
- *Throughput (packets per msec):* This value is the result of the number of the successfully transmitted packets divided by the simulation time in msecs. It is a good indication of the overall system performance.

- *Throughput (bits per msec):* This value is the result of the number of the successfully transmitted bits divided by the simulation time in msecs. It is a good indication of the overall system performance.
- *Throughput (percentage of bitrate):* This value is the result of the operation [Throughput / 23.529 * 10^6], assuming Throughput in bits per msec. It is a good indication of the overall system performance.
- *Average Delay (msecs):* "Delay" is the time a packet remains in the system until its successful arrival at the destination node. The average delay is the sum of all the delays of the successfully transmitted packets divided by the number of these packets. It is an indication of the overall system performance.
- *Number Of Transmissions:* This is the total number of the transmissions that were attempted during the simulation.
- *Collisions:* This is the number of the collisions when trying to transmit data.
- *Collisions (Percentage of Transmissions):* This is the number of the collisions divided by the number of the transmissions.

3.2. Code Structure

As it was mentioned before, the basic structure of the code is object oriented. The implemented classes in the simulation mechanism are: "Packet", "Carrier" and "Node".

3.2.1. The Class "Packet"

The class "Packet" represents the common packet in the network. Whatever transmitted in HIPERLAN is a "Packet" object. Some of the properties of the "Packet" are the size, the generation time, the sender's ID, the receiver's ID, the lifetime, the priority et al. The basic methods are the "GetRML" that returns the residual packet lifetime (Residual ML), and the "GetCAM_Priority" which returns the CAM priority of the packet. In Figure 11, the structure of the class "Packet" is presented.

3.2.2 The Class "Carrier"

The class "Carrier" represents the communication medium of the network. In general, the class "Carrier", which produces a single object, "stores" the current transmitted packet, the status of the medium (idle or data transmitting or in collision et al), the nodes transmitting the current moment, the time that the current transmission will be completed et al. The basic methods are the “PutPacketOnCarrier” which “puts” a packet on the medium, and the “AddNodeInTransmittingInfo” which adds the node that starts transmitting in the list of the nodes that are currently transmitting. A view of the class “Carrier” is shown in Figure 12.

3.2.3 The Class “Node”

The class “Node” is definitely the most significant class and it implements the most of the operations of the simulator. Every node is represented by an object of the class “Node” and carries a unique ID. There is a great number of properties and methods in the class “Node”. The most important properties are the ID of the node, the type of the node (Simple or Forwarder or P_Supporter or P_Saver), the list of nodes that are hidden from it, the list of the hidden nodes that it can sense, the Packet Buffer, the packet to be sent, a great number of variables that are time indicators and which help to implement the medium access protocol of HIPERLAN et al. The methods of the class “Node” constitute the “heart” of the simulation mechanism. Below, these methods are presented one by one.

- *AddPacketInBuffer:* It adds a packet in the buffer.
- *ChangeP_Status:* It concerns only the P_Savers. It checks the conditions and changes the status of the node from On to Off and vice-versa.
- *ChannelAccess:* This is the basic function that implements the MAC protocol of HIPERLAN, the “EY-NPMA”. When a node has a packet to transmit, it uses this method to gain access to the channel.
- *CheckExpiredPackets:* It checks the packets in the buffer to discover the packets that their lifetime has expired. After that, it rejects them.
- *CheckNode:* This is the method that is called at every time unit for every node – object of the network. It checks the node state and decides which actions must be executed by calling other methods.
- *GetCurrentBufferSize:* It returns the current size of the packet buffer in bits. It is used to find out if there is enough space in the buffer to add a packet.
- *InternalPacketArrival:* It implements the generation of a new packet inside the node. It decides the size of the packet, the destination, the lifetime et al.

- *IsNodeHidden:* The Boolean returned result shows if the node, which is the argument of the method, is hidden from the node that calls the method. In case the node is hidden, it shows if its signal can be at least detected.
- *IsTransmissionListened:* If the node that is calling this method can detect the current transmission, then the method returns the time that this transmission is completed.
- *NextArrival_Burst:* It computes the time that the next packet generation takes place, when the selected packet arrival method is the "Bursting".
- *NextArrival_Poisson:* It computes the time that the next packet generation takes place, when the selected packet arrival method is the "Poisson".
- *ReceiveData:* This method receives the transmitted data packet which is destinated to the calling node. If the calling node is an intermediate (Forwarder or P_Supporter) for this packet, then the packet is added in the buffer so that it is sent later to its final destination.
- *RetransmitData:* After the transmission of a data packet and the non-reception of the corresponding acknowledgement, this method is called in order to reschedule the transmission of the packet that was not acknowledged.
- *SelectPacketToSend:* The packet to be sent is selected from the buffer. The selection is made using the "EDF" or the "FIFO" method, according to the user's choice.
- *SendAck:* It implements the acknowledgement sending, after the successful reception of a data packet.
- *SendData:* This method is responsible for the transmission of a data packet. If it is necessary, an intermediate node for this transmission is set.

In Figure 13, the whole structure of the class "Node" with its properties and methods is presented.

3.3. Operation Sequence

Here we describe briefly the sequence of the actions that take place during the simulation. First of all, we must make clear that every simulation is independent from any other, since the simulation mechanism is initiated every time.

According to the parameters set, packets are generated inside the nodes, and if there is enough space they are stored in the buffer. The destination of every packet is another node of the LAN. The source node tries to gain access to the channel, according to EY-NPMA, in order to send the packet selected from the buffer. In an exhaustive way, the nodes are checked one by one at every time unit. The simulation is terminated when the simulation time is over and the results are calculated and stored.

In the beginning of every simulation, all the nodes are initiated as elements of a dynamic list of “Node” objects. The carrier, which is represented by the single object of the class “Carrier” (objCarrier), is also initiated. Next, the “forwarders” and the “P_Supporters” are selected, provided that the user has made the corresponding choices. After that, the P_Savers, the pairs of hidden nodes and the hidden nodes that are in sense range are selected. As it was mention before, a special feature of HIPERSIM is the fact that it distinguishes hidden nodes between those that are in sense range and those that are out of sense range. In the next section, the HIPERSIM results show that the hidden nodes that are out of sense range affect significantly the HIPERLAN performance. The time the first packet generation takes place is calculated for every node. Afterwards, the application enters the main loop of the simulation, which checks every node at every time unit, by using the method “CheckNode”.

The function “CheckNode” decides whether some other methods of the class “Node” will be called. Initially, it checks if there is a packet arrival from another node, so as to call the function “ReceiveData”. Then, it checks if a data packet retransmission is needed, and if it does, it calls the function “RetransmitData”. After that, the function “ChannelAccess” is called. If there is no packet to be sent at the current moment, the “ChannelAccess” function performs no action. Then, the function “CheckNode” checks if an acknowledgement must be sent, so as to call the function “SendAck”. Afterwards, if it is time to generate a new data packet, the function “InternalPacketArrival” is called. Lastly, the function “CheckNode” checks if the node status must change from On to Off and vice-versa, by using the function “ChangeP_Status”. It is obvious that the latter function can be called only when the calling node is a P_Saver.

When the simulation time is over and the simulation is completed, the results are calculated and presented. A new record is created in the table of the results.mdb, and the values of all the parameters and the results are recorded there. In order to get these results, some special variables are inserted in different places in the code and their values are updated during the simulation. There are also some global, assistant functions, which are called when needed. For example, one of these functions is the msecs conversion to high rate bit-periods (1 msec = 23529 high rate bit-periods). An overview of the simulation mechanism is shown in Figure 14.

4. HIPERSIM Simulation Results

This section analyses the HIPERSIM results about the HIPERLAN operation under various conditions.

In every simulation the packet generation inside the nodes is Poissonian. The packet size is variable and exponential. Its default average value is 3000 bits, without counting in the added control bits. The packet lifetime is randomly selected between 10 and 800 msecs.

Initially, we examine the system throughput depending on the number of the nodes that constitute the WLAN. The throughput measures as a percentage of high bit-rate. In Figure 15, we notice that the throughput value remains almost stable when the number of nodes increases. This satisfactory performance is due to the fact that EY-NPMA deters the great increase of the collisions even when the number of nodes increases significantly.

Likewise, the average packet delay, that is the average time the packet remains in the system from its generation time till its successful transmission and reception, is almost stable while the number of nodes increases (Figure 16).

Figure 17 presents the average packet delay depending on the Poisson parameter which concerns the packet generation in every node every msec. Actually, the increase of the value of the Poisson parameter causes the increase of the system load, since the packet generation rate increases. In this graph, we examine the performance of the Earliest Deadline First buffer discipline in comparison to the traditional First In First Out. As it was expected, the average delay is bigger when FIFO is used, especially when the load increases. HIPERSIM can simulate both the EDF and the FIFO queue discipline.

We examine the way that the average packet size affects the throughput, assuming two different values for the Bit Error Rate (after check). The “bit error rate” is the rate a bit error occurs, after the appliance of all the predefined error detection and error correction checks (that is why we assume so low bit error rates and we add the “after check” characterization). According to the value of the bit error rate and the current packet length, HIPERSIM decides whether a bit error occurs during a packet transmission. In Figure 18, we see that the increase of the average packet size initially causes some increase of the throughput (because of the smaller overhead) which eventually stabilizes. But when we increase the bit error rate, the increase of the average packet size eventually causes a decrement of the throughput.

Another issue is the LAN performance when there are nodes that operate as P_Savers. When the number of the P_Savers increases, the system throughput decreases. HIPERSIM simulates the “ON” and “OFF” P_Saver periods. In Figure 19, the simulation results are presented for different numbers of P_Savers, when the total number of nodes is 30 and the number of P_Supporters is 2. Every P_Saver can randomly use any P_Supporter.

HIPERSIM simulates active signaling, a feature of EY-NPMA, according to which a node that can just sense the signal of another node (sense range) is able to participate successfully with the latter in the synchronized channel access cycle. In case there are two hidden nodes that cannot even detect the signal of one another (they are not in sense range), there is a big possibility collisions to happen, which would lead to significant performance reduction. The reason for this performance reduction is the fact that these two nodes would not be able to synchronize directly, since each one would have a different view of the channel status. In HIPERSIM, every node has its own view of the channel status, depending on the network topology.

In Figures 20 and 21 we use the term “Probability Of Sensing Hidden Signal”. This is the probability two hidden nodes are in sense range. The “Probability Of Hidden Pair” is the probability two nodes are hidden from each other. As we can see in Figures 20, when the Probability Of Hidden Pair increases, the network throughput decreases. Especially when the Probability Of Sensing Hidden Signal is low (less than 0.5), the system performance can be characterized as unacceptable. But when the value of the Probability Of Sensing Hidden Signal is high and close to one, then the throughput is sufficiently high. The same behavior can be seen in Figures 21, by studying collisions rate. As it was expected, the Percentage Of Collisions is high when the Probability Of Hidden Pair is high. When the Probability Of Sensing Hidden Signal increases, the collisions rate decreases.

5. Conclusion

HIPERSIM is a simulation environment for the HIPERLAN WLAN. It is a full parameterized simulator that can test the performance of HIPERLAN under various conditions. It simulates the special features of the wireless environment, like the “hidden nodes”. The hidden nodes are pairs of nodes that are out of communication range. HIPERSIM distinguishes between the communication range and the sense range (signal detection range), in order to simulate accurately the wireless topology. The simulation mechanism includes the priority mechanism of HIPERLAN, the EY-NPMA protocol, the packet forwarding and the power saving. HIPERSIM is based on the "ETSI functional specification EN 300 652 v1.2.1" for HIPERLAN.

The simulation engine of HIPERSIM is almost exhaustive, so as to simulate the complicated EY-NPMA protocol and the wireless environment accurately. Every node is examined separately at every time unit. When the simulation time is over, the results are presented and recorded. The code structure of HIPERSIM is object oriented and the base classes are: the "Packet", the "Carrier" and the "Node". HIPERSIM has a graphical user interface according to the W32 pattern.

The HIPERSIM results have shown that HIPERLAN is an efficient WLAN standard. Probably, some improvement of the protocol is necessary, so that the frequent collisions between nodes that are out of sense range are avoided. Basically, the "hidden nodes" problem concerns the collisions that take place close to the receiver and not in the sender's region. So it would be efficient if there were a mechanism that informed the neighbors of the receiver about the oncoming transmission. In that case, the receiver's neighbors would not collide the arrived data, so the overall system performance would improve.

6. References

[1] EN 300 652 V1.2.1 (1998-07), ETSI, Broadband Radio Access Network (BRAN); HIgh PErformance Radio Local Area Network (HIPERLAN) Type 1; Functional Specification

[2] Data Transfer for HIPERLAN, Philippe Jacquet, Pascale Minet, Paul Muhlethaler and Nicolas Rivierre, Inria Le Chesnay France, Wireless Personal Communications 4: 65-80, 1996

[3] Performance of the EY-NPMA Protocol, K. FU, Y.J. GUO and S.K. Barton, Department of Electronic and Electrical Engineering, University of Bradford, UK, Wireless Personal Communications 4: 41-50, 1996

[4] Priority and Collision Detection with Active Signaling – The Channel Access Mechanism of HIPERLAN, Philippe Jacquet, Pascale Minet, Paul Muhlethaler and Nicolas Rivierre, Inria Le Chesnay France, Wireless Personal Communications 4: 11-26, 1996

[5] Analysis and Optimization of the HIPERLAN Channel Access Contention Scheme, S.Chevrel, A.H. Aghvami, H.-Y. Lach and L. Taylor, Centre for Telecommunication Research, King's College London, University of London, The Strand, UK, Wireless Personal Communications 4: 27-39, 1996

[6] HIPERLAN, Tim Wilkinson, HP Labs Europe

[7] Performance Study of Access Control in Wireless LANs – IEEE 802.11 DFWMAC and ETSI RES 10 HIPERLAN, Jost Weinmiller, Morten Schlager, Andreas Festag, Adam Wolisz, Technical University Berlin, Germany

[8] Wireless LAN Standards: HIPERLAN and IEEE802.11, ECPE 6504 Wireless Networks and Mobile Computing (Individual Project Report), Jing Zeng 04-24-2000

[9] Learning-Automata-Based TDMA Protocols for Broadcast Communication Systems with Bursty Traffic, Georgios I. Papadimitriou, Andreas S. Pomportsis, IEEE Communication Letters, Vol. 4, No.3, March 2000

[10] Stability and Performance Analysis of HIPERLAN, G. Anastasi, L. Lenzini and E. Mingozzi, Dept. of Information Engineering, University of Piza, Italy

[11] WIRELESS NETWORKS, P. Nicopolitidis, M.S. Obaidat, G.I. Papadimitriou, A.S. Pomportsis, WILEY

[12] ETSI – Telecom Standards, http://www.etsi.org/

[13] ComNets Annual Report 1996 - HIPERLAN - High Performance Radio LAN, http://www.comnets.rwth-aachen.de/report96/node263.html

[14] HIPERLAN - the approaching standard for Wireless LAN's, http://www.netplan.dk/hip.htm

[15] Wireless LANs, http://www.netplan.dk/wireless.htm#HIPERLAN

Notation:

ML: MPDU Lifetime

RML: Residual MPDU Lifetime

UP: User Priority (low = 1 , high = 0)

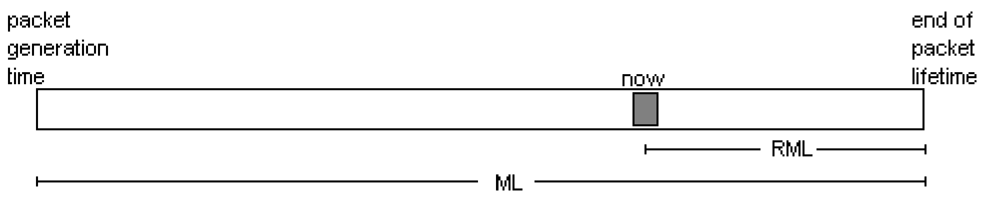


Figure 1. The Packet Lifetime

UP ↓	RML (msec) →	< 10	10 - 20	20 - 40	40 - 80	> 80
0		0	1	2	3	4
1		1	2	3	4	4

Figure 2. The CAM Priority Assigning

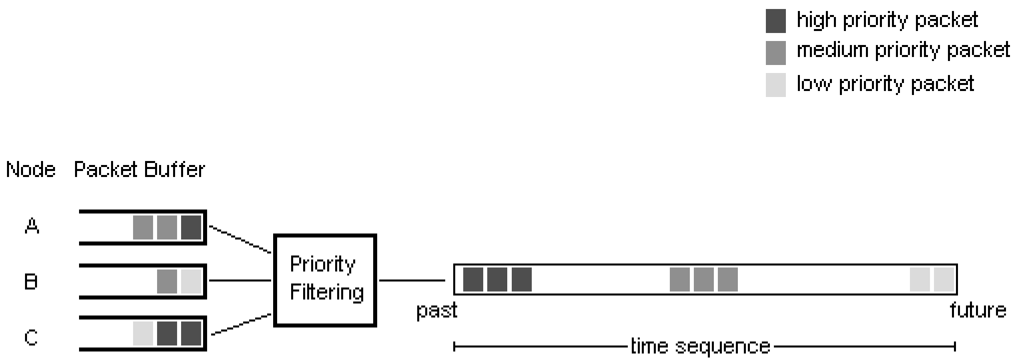


Figure 3. The Priority Scheme

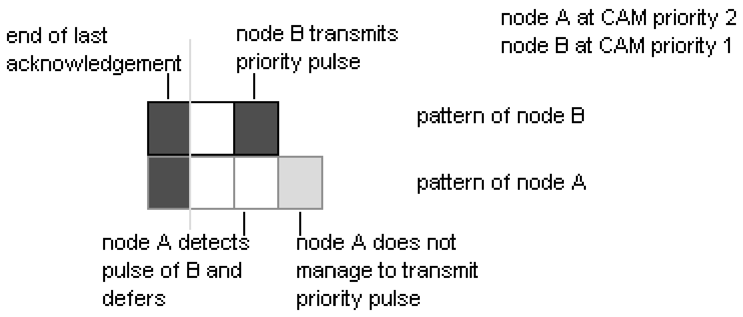

end of last acknowledgement
node B transmits priority pulse
node A at CAM priority 2
node B at CAM priority 1
pattern of node B
pattern of node A
node A detects pulse of B and defers
node A does not manage to transmit priority pulse

Figure 4. The Prioritization Phase

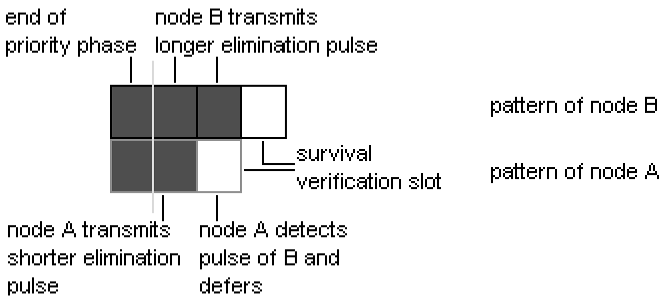

end of priority phase
node B transmits longer elimination pulse
pattern of node B
survival verification slot
pattern of node A
node A transmits shorter elimination pulse
node A detects pulse of B and defers

Figure 5. The Elimination Phase

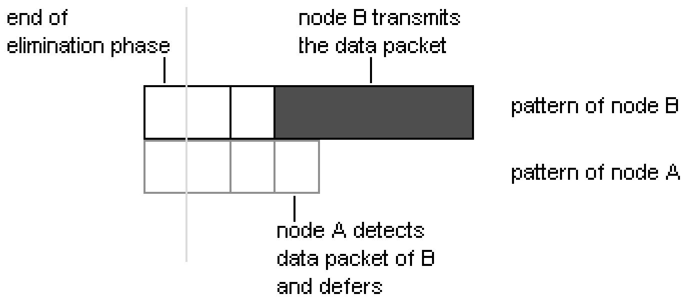

end of elimination phase
node B transmits the data packet
pattern of node B
pattern of node A
node A detects data packet of B and defers

Figure 6. The Yield Phase

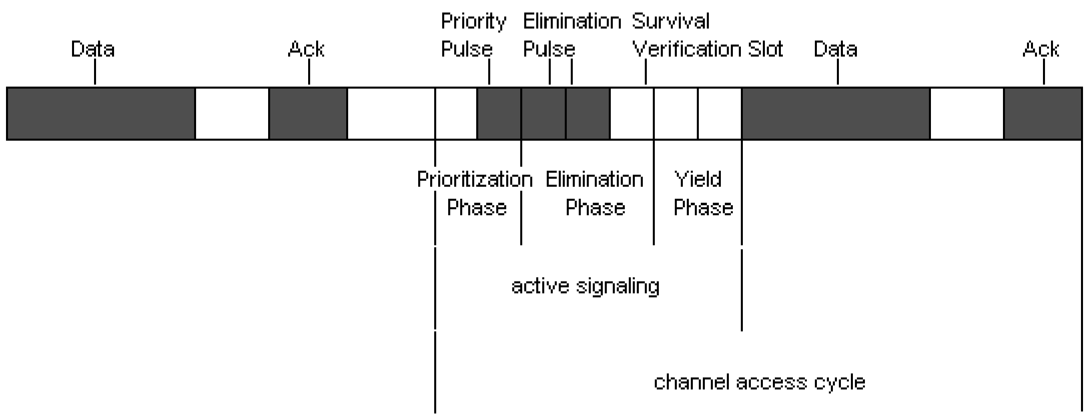


Figure 7. Overview of the EY-NPMA protocol

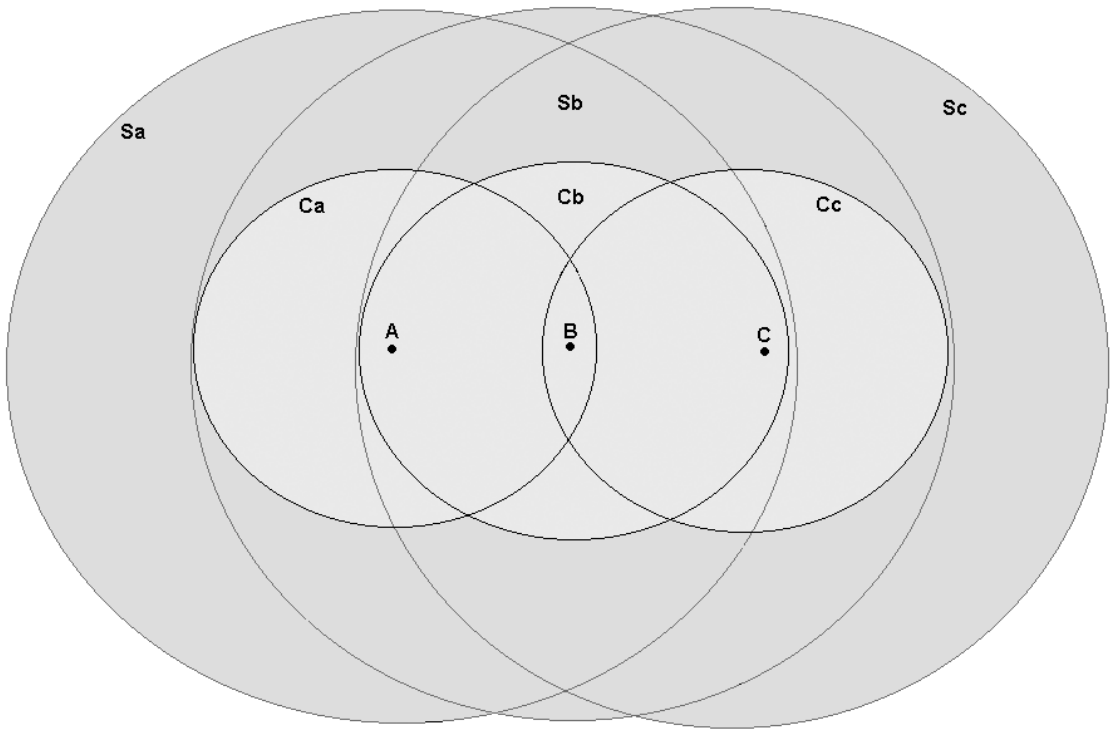


Figure 8. Representation of the Communication Range (Ci) and the Sense Range (Si)

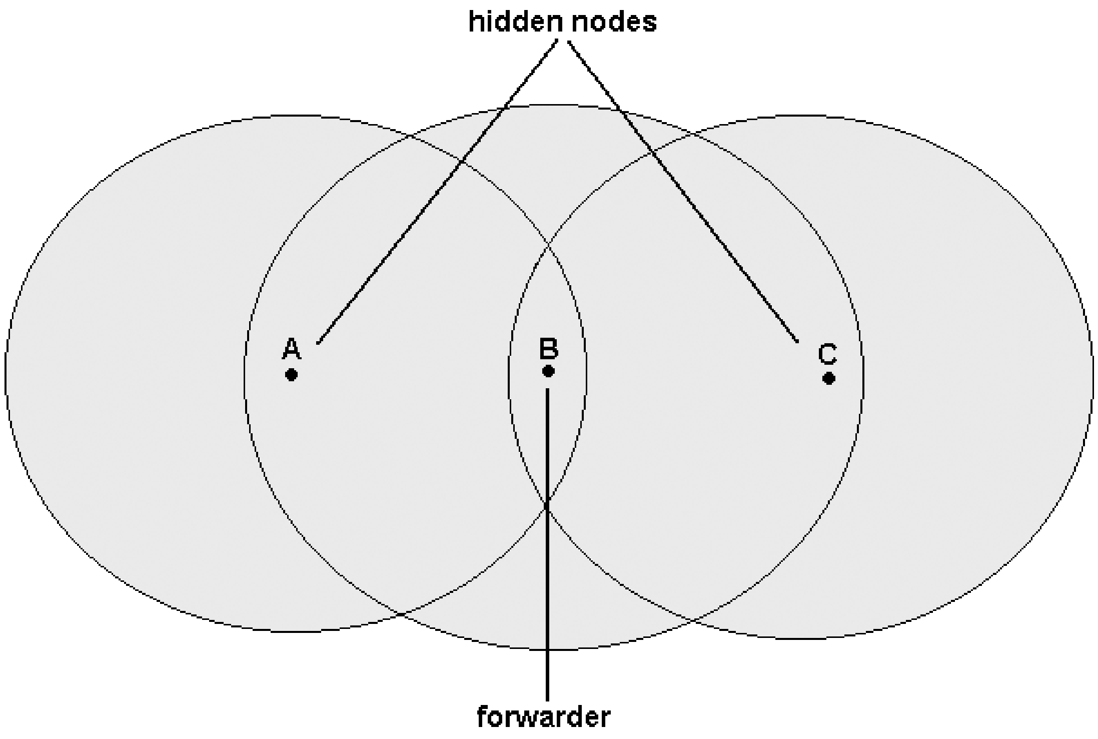
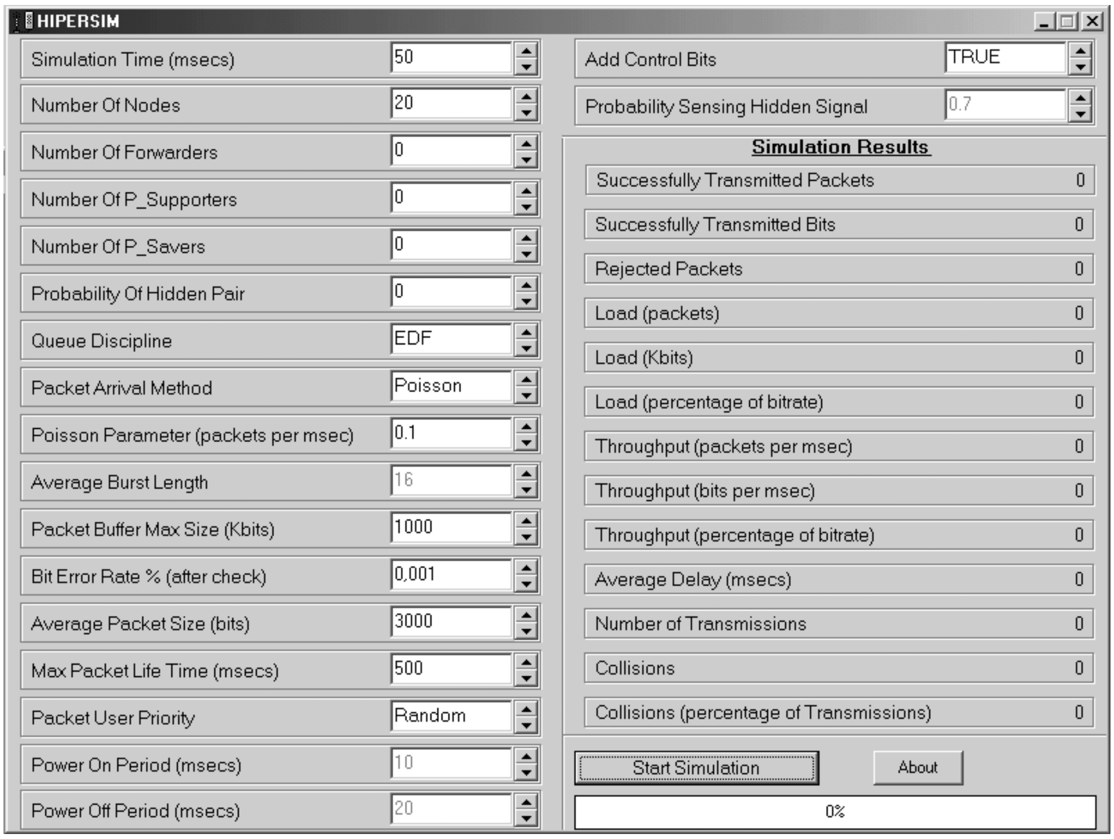


Figure 9. Representation of the Packet Forwarding

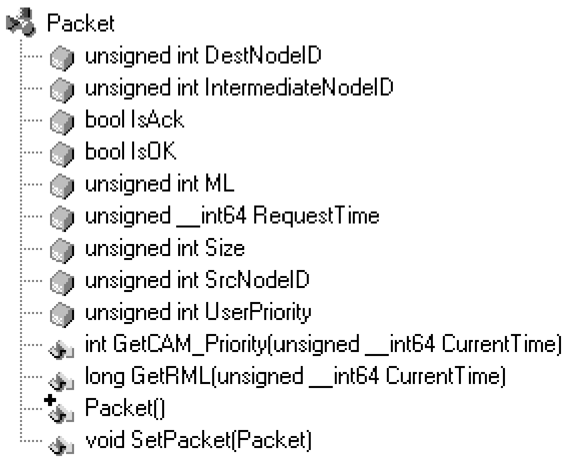


Simulation Time (msecs)	50
Number Of Nodes	20
Number Of Forwarders	0
Number Of P_Supporters	0
Number Of P_Savers	0
Probability Of Hidden Pair	0
Queue Discipline	EDF
Packet Arrival Method	Poisson
Poisson Parameter (packets per msec)	0.1
Average Burst Length	16
Packet Buffer Max Size (Kbits)	1000
Bit Error Rate % (after check)	0,001
Average Packet Size (bits)	3000
Max Packet Life Time (msecs)	500
Packet User Priority	Random
Power On Period (msecs)	10
Power Off Period (msecs)	20
Add Control Bits	TRUE
Probability Sensing Hidden Signal	0.7

Simulation Results

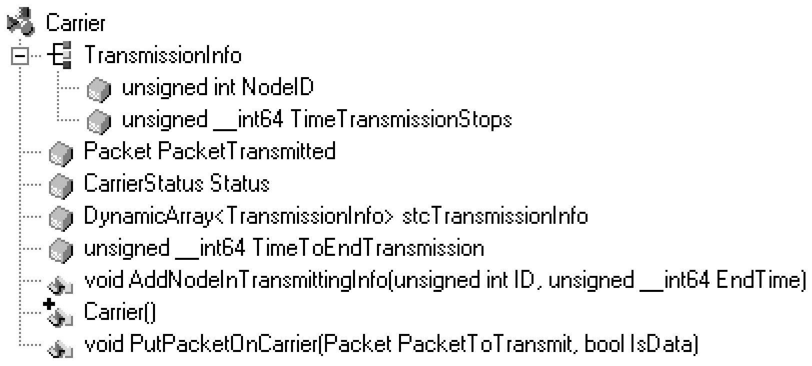
Successfully Transmitted Packets	0
Successfully Transmitted Bits	0
Rejected Packets	0
Load (packets)	0
Load (Kbits)	0
Load (percentage of bitrate)	0
Throughput (packets per msec)	0
Throughput (bits per msec)	0
Throughput (percentage of bitrate)	0
Average Delay (msecs)	0
Number of Transmissions	0
Collisions	0
Collisions (percentage of Transmissions)	0

Figure 10. The Graphical User Interface of HIPERSIM



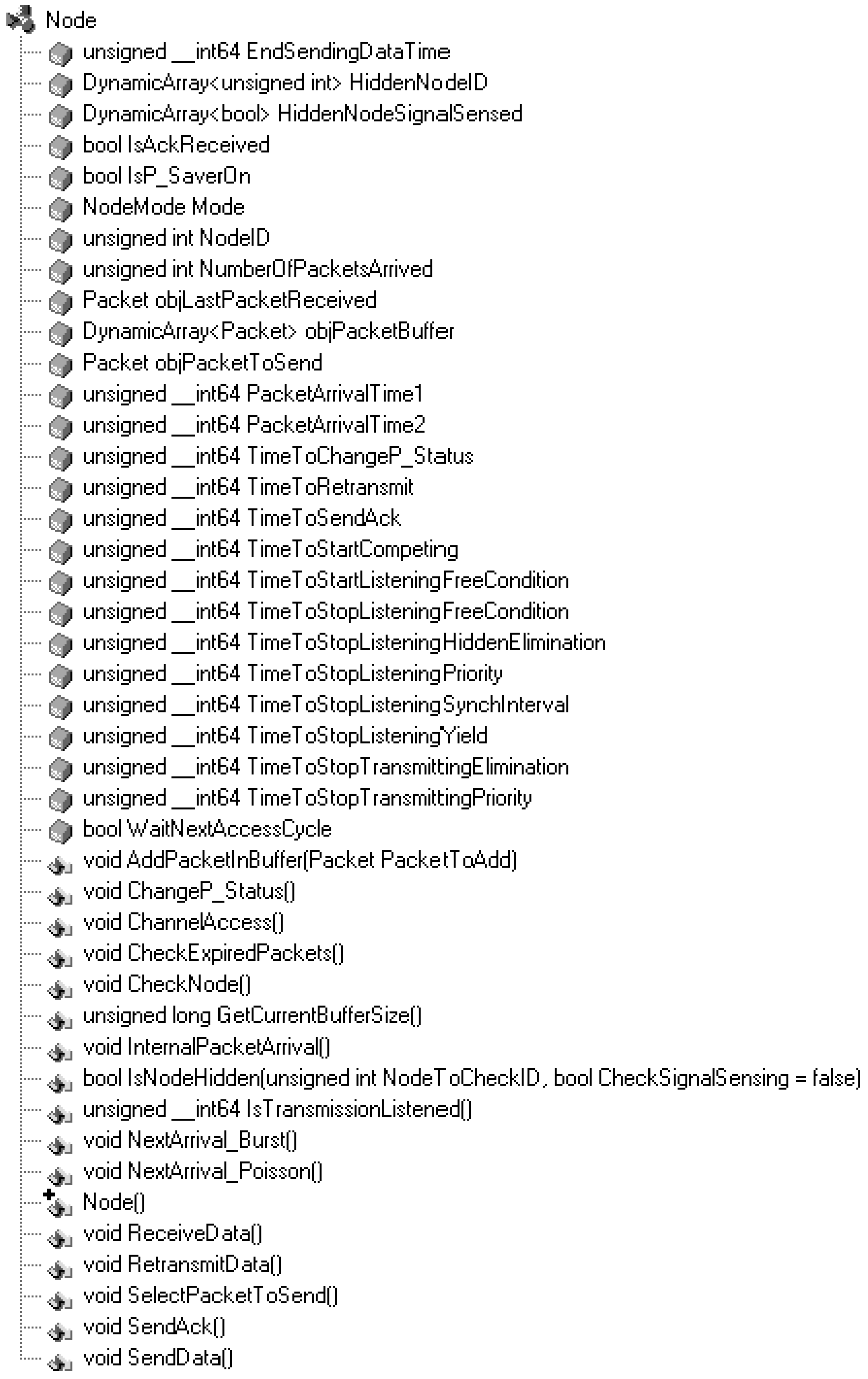
```
Packet
    unsigned int DestNodeID
    unsigned int IntermediateNodeID
    bool IsAck
    bool IsOK
    unsigned int ML
    unsigned __int64 RequestTime
    unsigned int Size
    unsigned int SrcNodeID
    unsigned int UserPriority
    int GetCAM_Priority(unsigned __int64 CurrentTime)
    long GetRML(unsigned __int64 CurrentTime)
    Packet()
    void SetPacket(Packet)
```

Figure 11. The class "Packet"



```
Carrier
    TransmissionInfo
        unsigned int NodeID
        unsigned __int64 TimeTransmissionStops
    Packet PacketTransmitted
    CarrierStatus Status
    DynamicArray<TransmissionInfo> stcTransmissionInfo
    unsigned __int64 TimeToEndTransmission
    void AddNodeInTransmittingInfo(unsigned int ID, unsigned __int64 EndTime)
    Carrier()
    void PutPacketOnCarrier(Packet PacketToTransmit, bool IsData)
```

Figure 12. The class "Carrier"



```
Node
    unsigned __int64 EndSendingDataTime
    DynamicArray<unsigned int> HiddenNodeID
    DynamicArray<bool> HiddenNodeSignalSensed
    bool IsAckReceived
    bool IsP_SaverOn
    NodeMode Mode
    unsigned int NodeID
    unsigned int NumberOfPacketsArrived
    Packet objLastPacketReceived
    DynamicArray<Packet> objPacketBuffer
    Packet objPacketToSend
    unsigned __int64 PacketArrivalTime1
    unsigned __int64 PacketArrivalTime2
    unsigned __int64 TimeToChangeP_Status
    unsigned __int64 TimeToRetransmit
    unsigned __int64 TimeToSendAck
    unsigned __int64 TimeToStartCompeting
    unsigned __int64 TimeToStartListeningFreeCondition
    unsigned __int64 TimeToStopListeningFreeCondition
    unsigned __int64 TimeToStopListeningHiddenElimination
    unsigned __int64 TimeToStopListeningPriority
    unsigned __int64 TimeToStopListeningSynchInterval
    unsigned __int64 TimeToStopListeningYield
    unsigned __int64 TimeToStopTransmittingElimination
    unsigned __int64 TimeToStopTransmittingPriority
    bool WaitNextAccessCycle
    void AddPacketInBuffer(Packet PacketToAdd)
    void ChangeP_Status()
    void ChannelAccess()
    void CheckExpiredPackets()
    void CheckNode()
    unsigned long GetCurrentBufferSize()
    void InternalPacketArrival()
    bool IsNodeHidden(unsigned int NodeToCheckID, bool CheckSignalSensing = false)
    unsigned __int64 IsTransmissionListened()
    void NextArrival_Burst()
    void NextArrival_Poisson()
    Node()
    void ReceiveData()
    void RetransmitData()
    void SelectPacketToSend()
    void SendAck()
    void SendData()
```

Figure 13. The class "Node"

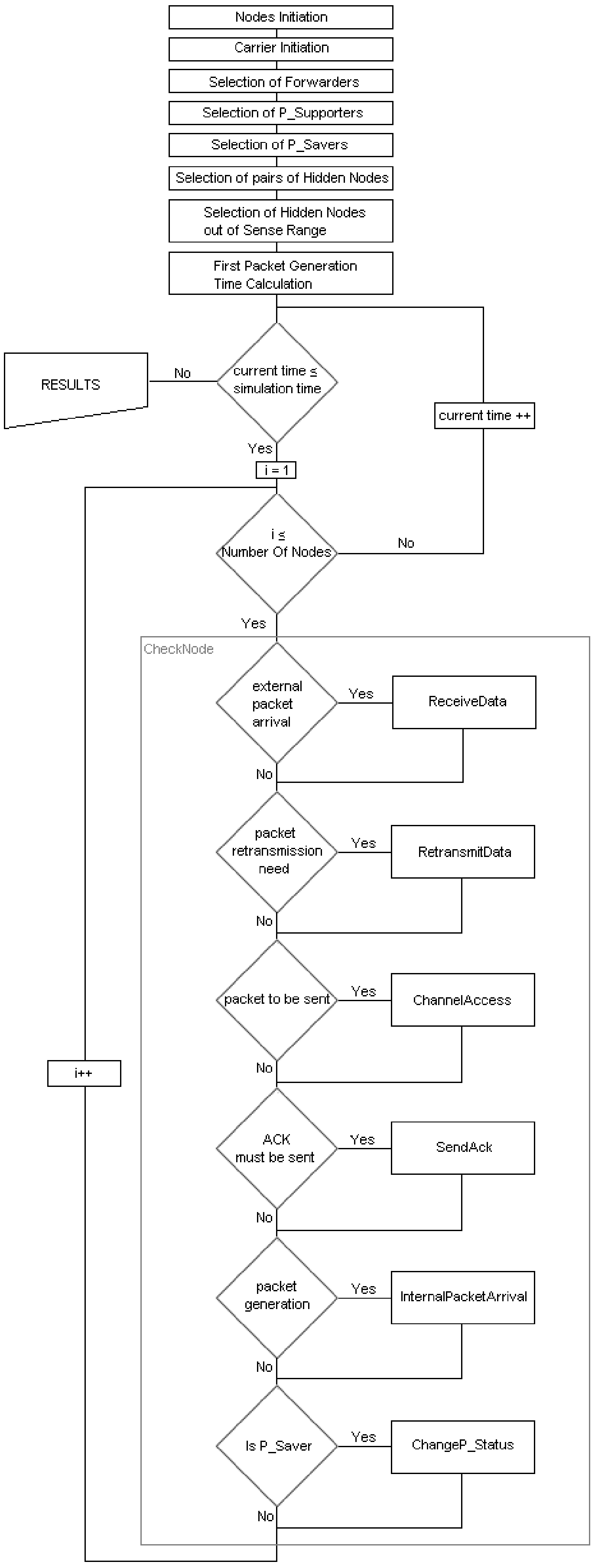


Figure 14. Overview of the HIPERSIM simulation mechanism

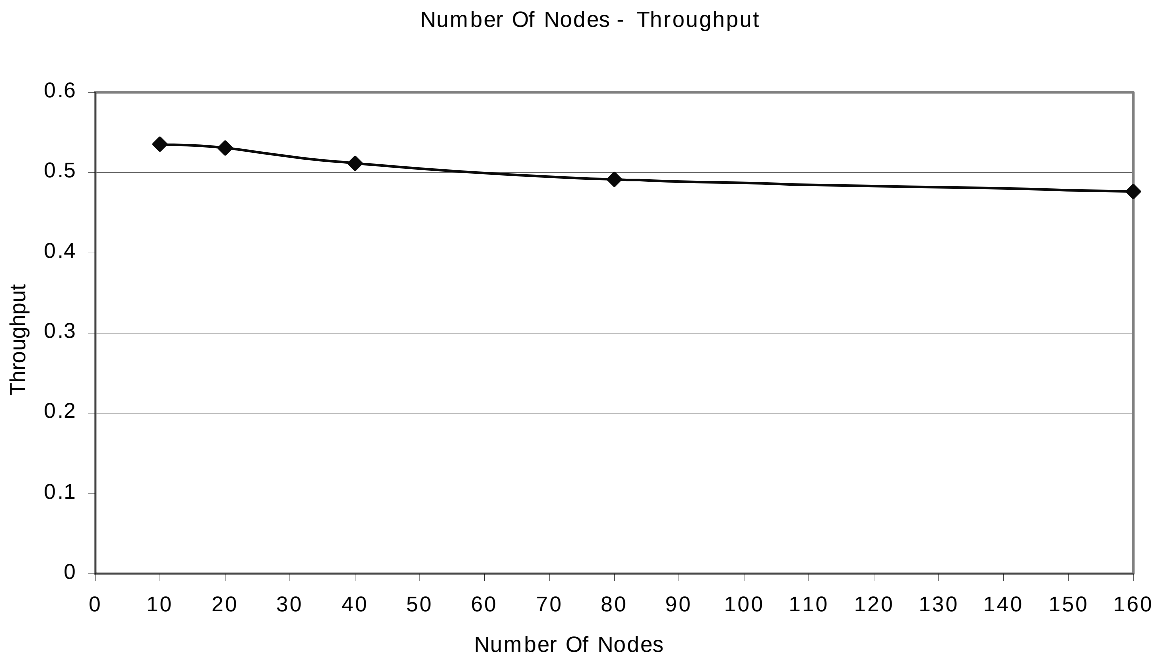


Figure 15. System throughput versus number of nodes

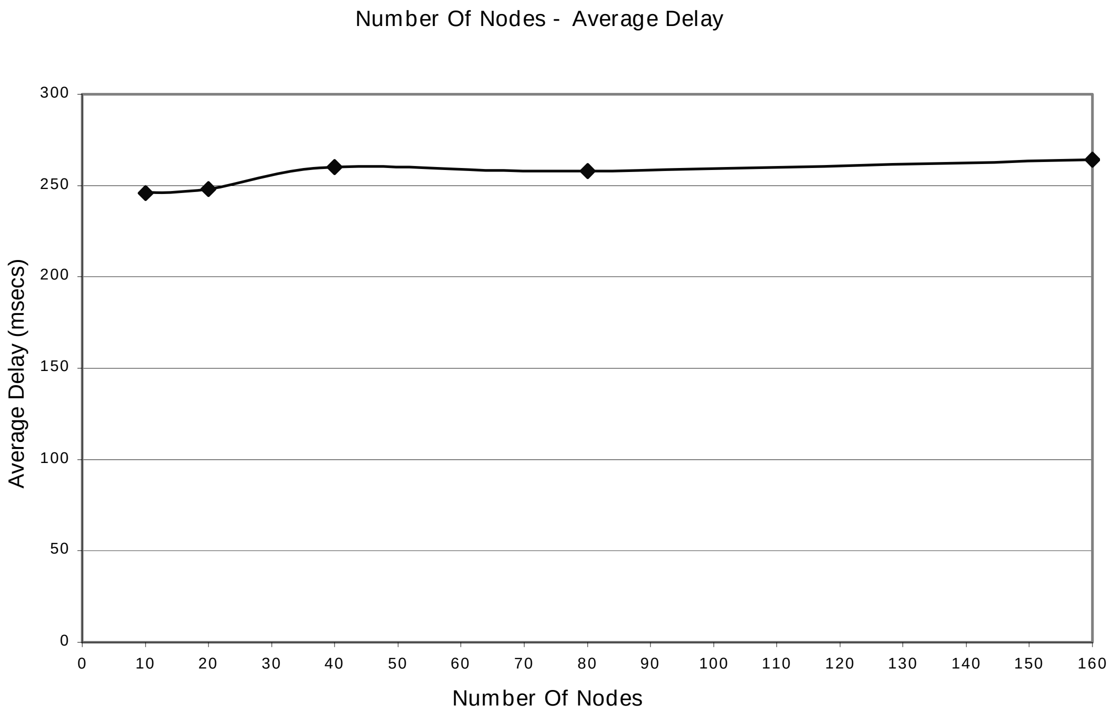


Figure 16. Average delay versus number of nodes

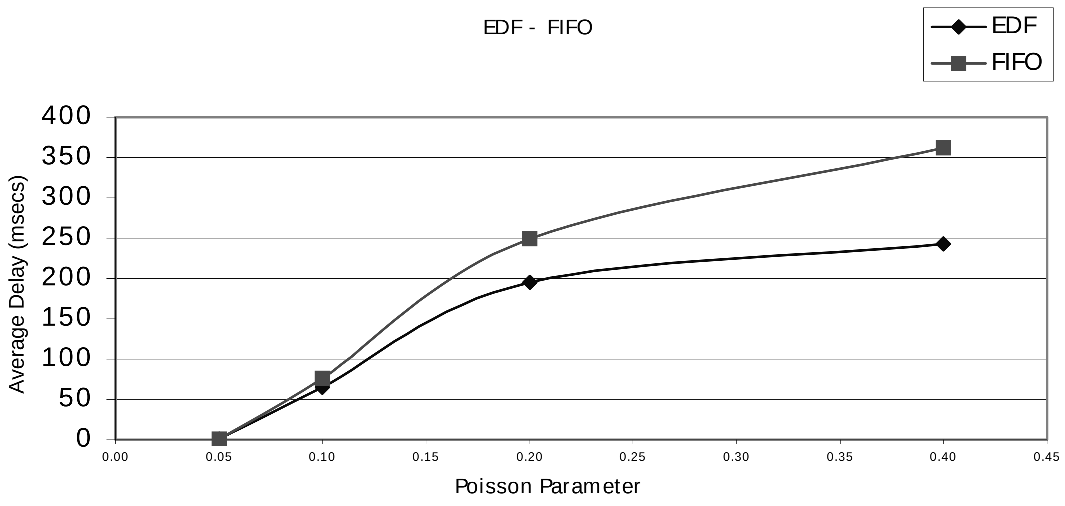


Figure 17. Comparison between the EDF and the FIFO queuing discipline

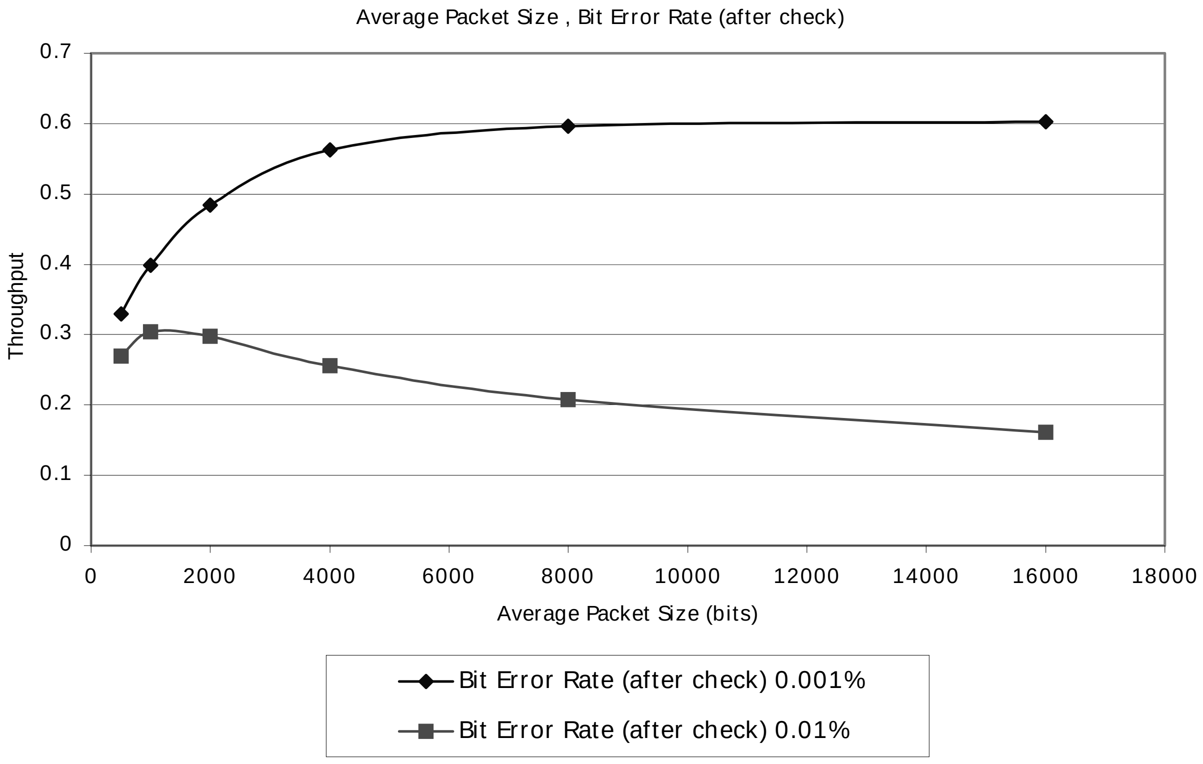


Figure 18. System throughput as a function of the average packet size and the bit error rate

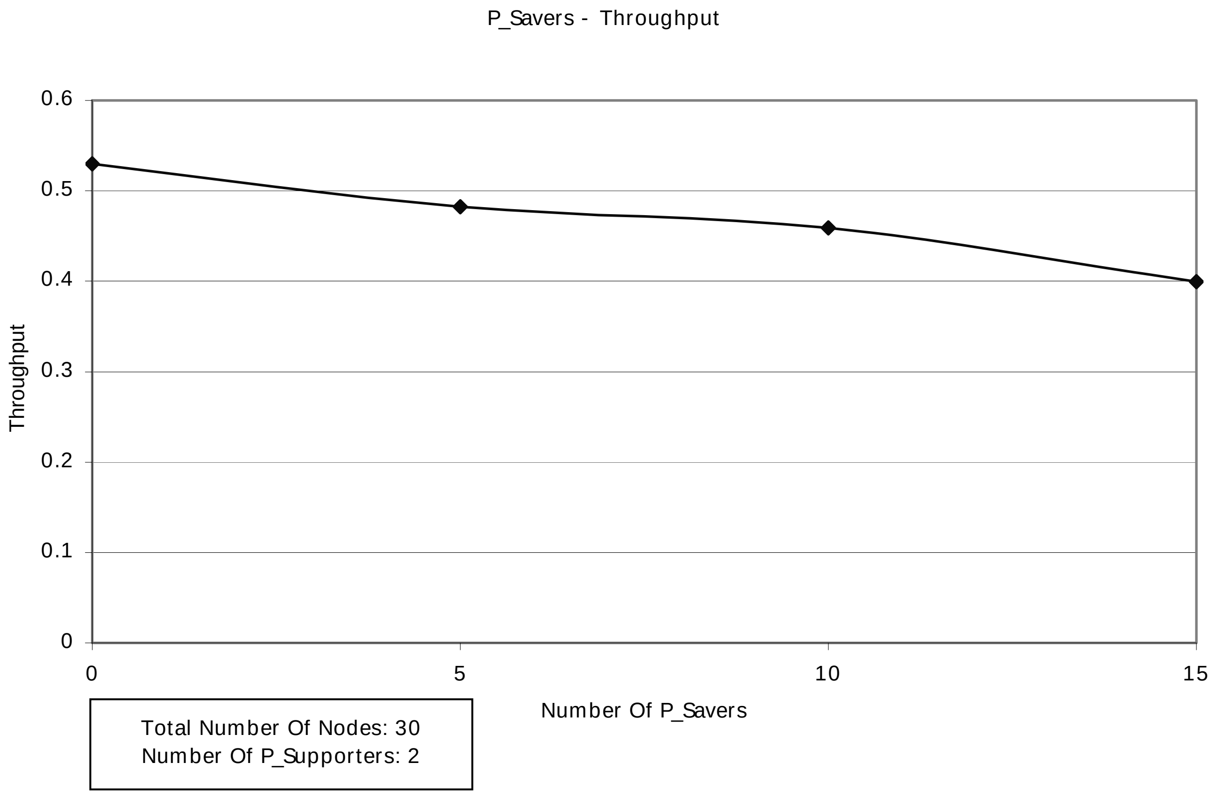


Figure 19. System throughput versus number of P_Savers

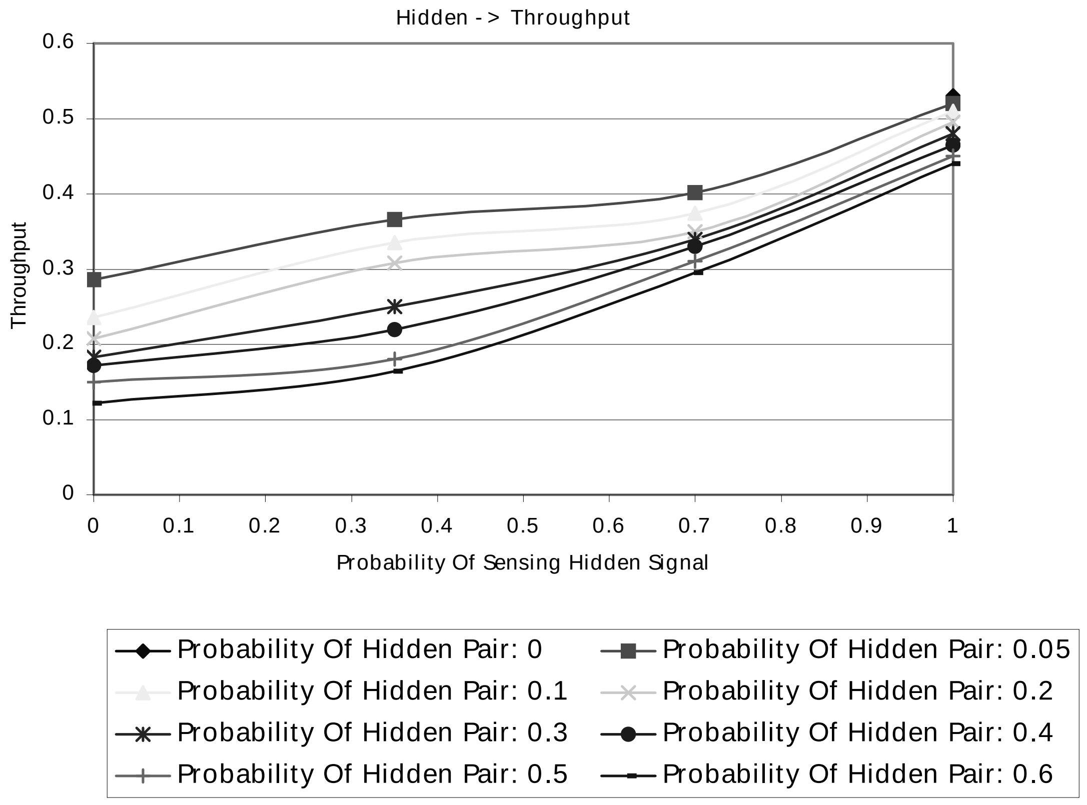


Figure 20. Throughput as a function of the "Probability Of Sensing Hidden Signal" and the "Probability Of Hidden Pair"

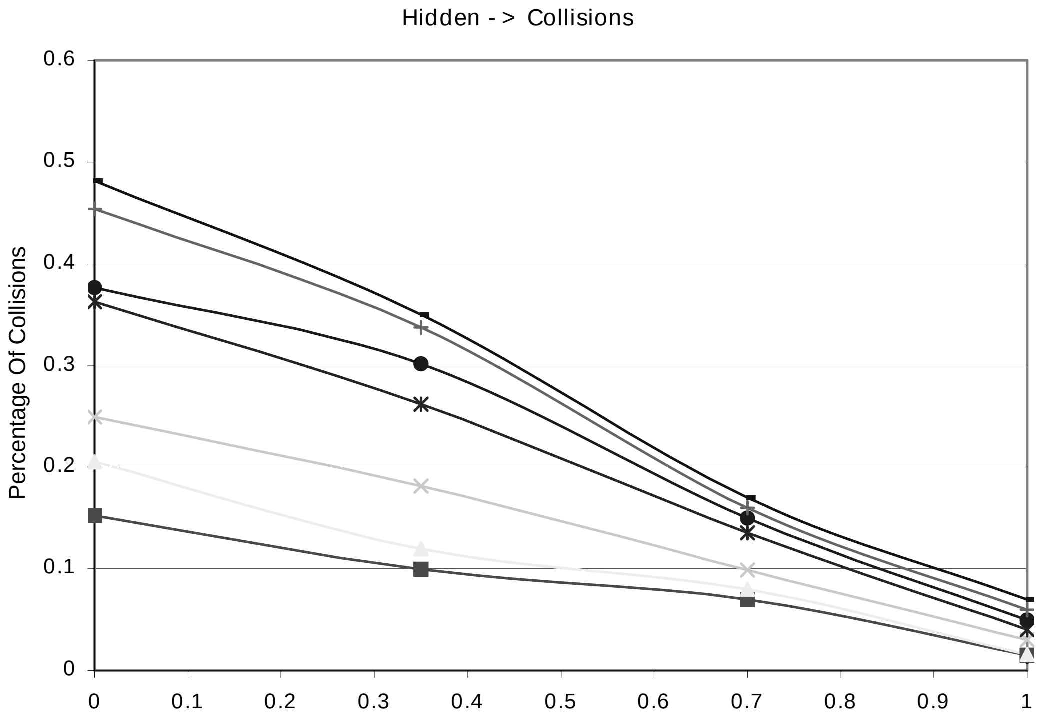


Figure 21. Collisions as a function of the "Probability Of Sensing Hidden Signal" and the "Probability Of Hidden Pair"