Distributed Dynamic Priority Queuing Medium Access Control Protocol

by

Himal P. Karmacharya

B. S., Electrical Engineering and Computer Science

MIT

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Masters of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology
February 6, 2001

Copyright 2001 Himal P. Karmacharya. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author.....
Department of Electrical Engineering and Computer Science
February 6, 2001

Certified by..... .....
Whay Chiou Lee
VI-A Company Thesis Supervisor

Certified by.....
Kai-Yeung Siu
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by.....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

BARKER

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUL 1 1 2001
LIBRARIES

Distributed Dynamic Priority Queuing
Medium Access Control Protocol

by

Himal Karmacharya

Submitted to the
Department of Electrical Engineering and Computer Science

February 6, 2001

In partial fulfillment of the requirements for the Degree of
Masters in Engineering in Electrical Engineering and Computer Science

ABSTRACT

Deployment of local area networks in home environment requires implementing a quality of service (QoS) mechanism that provides a good delay and jitter performance for voice and video applications, both of which are common in home networks. A medium access control (MAC) protocol is a mechanism for coordinating access to a shared communication channel among a population of stations. QoS provision is an inherent problem in all Ethernet-like MAC protocols, which do not have a central scheduler with global information. A number of MAC protocols have been proposed to address this problem but they have not been able to completely resolve the issue.

We propose a new protocol, referred to as the Distributed Dynamic Priority Queuing (DDPQ) MAC protocol, which is similar to Ethernet and Distributed Fair Priority Queuing MAC protocols, the latter of which is a MAC protocol for an emerging HPNA (Home Phoneline Networking Association) technology. The proposed protocol provides a better delay and jitter performance for real-time traffic in home networking environment. This is shown by simulations, which compare the performance of the new protocol against the above two existing protocols. A comprehensive literature survey is also carried out to analyze some of the popular approaches taken to provide QoS in Ethernet-like conditions.

Faculty Thesis Supervisor: Kai-Yeung Siu

Title: Associate Professor of Mechanical Engineering

VI-A Company Thesis Supervisor: Whay Chiou Lee

Title: Member of Technical Staff, Motorola, Inc.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor at Motorola, Whay C. Lee, who persevered with me and gave me direction during the entire phase of my thesis work. I learnt greatly under him and much of what has been accomplished in this thesis is a lot due to him. I would like to thank Patrick Maurer who gave me guidance and help whenever I needed it. I would also like to thank my thesis advisor, Prof. Siu, who took his time to review my thesis. Lastly I would like to thank all my friends, my apartment mates, and my family members who were inspirational throughout the period and who egged me on just when I was starting to have doubts about finishing the thesis on time.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Acronyms

Acronym	Complete Expression
BEB	Binary Exponential Backoff
BL	Backoff Level
BLAM	Binary Logarithm Arbitration Method
CRL	Collision Resolution Length
CRP	Collision Resolution Period
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CTS	Clear to Send
DCF	Distributed Coordination Function
DDPQ	Distributed Dynamic Priority Queuing
DFPQ	Distributed Fair Priority Queuing
ETSI	European Telecommunications Standards Institute
FCFS	First-Come-First-Serve
FIFO	First In First Out
HOLP	Head-of-Line-Priority
HPNA	Home Phoneline Networking Alliance
ICT	Instantaneous Contention Throughput
IFG	Inter-Frame Gap
LAN	Local Area Networking
MAC	Medium Access Control
MBL	Maximum Backoff Level
OSI	Open Systems Interconnection
PC	Personal Computer
PCF	Point Coordination Function

QoS	Quality of Service
RTS	Request to Send
SLA	Service Level Agreement
TBEB	Truncated Binary Exponential Backoff
TDMA	Time Division Multiplexing Access
WBPTE	Wait-Before-Transmit Priority Ethernet

1. Introduction

In this thesis, we propose a novel Distributed Dynamic Priority Queuing (DDPQ) MAC protocol, which is an enhancement of an existing Distributed Fair Priority Queuing (DFPQ) MAC protocol. Our protocol provides good QoS capability to multimedia traffic by dynamically adapting to changing levels and mix of offered loads.

1.1. Home Networking

Home networking is the term describing a local area network (LAN) setup in a home environment that enables the connection and integration of multiple computing and communication devices. Until recently, home network has been largely ignored. However, rapid proliferation of personal computers (PCs) and the Internet, advancements in telecommunications technology, and progress in the development of smart devices have increasingly emphasized the market potential of home networking. Furthermore, as these growth and advancement trends continue, the need for simple, flexible, and reliable home networks will greatly increase. It is estimated that by 2005 all consumer devices that are not network ready will be obsolete. [FrH00]

Consumer devices include both traditional data appliances like PC, handheld PDAs, etc and multimedia appliances like TV, stereo, telephone, etc. Data traffic has different set of requirements than multimedia traffic and their convergence puts a lot of demand on the system. For example, multimedia applications have a very strict delay requirement and are not so much concerned with dropping some frames--formatted packets. For such applications, frames that are significantly delayed cannot be used for reassembling the content and have to be discarded. On the other hand data transmissions are non-real time and do not have a strict delay requirement, but they are very sensitive to frame losses. The problem facing the system then is the fair resolution of medium contention among these diverse traffics so as to reconcile some seemingly conflicting set of requirements.

Protocols that are employed to coordinate access to a shared channel are often referred to as MAC, or multi-access protocols. Traditional LANs consist of a multi-access channel that is allocated among a set of stations competing for its use. A data frame that arrives at a station is inserted in a first-in-first-out (FIFO) transmit queue where the frame waits until it gets access to the channel so that it can be transmitted to its destination. Traditional LANs were originally designed for best-effort data transmission and are not suitable for the strict delay requirement of real-time multimedia traffic.

The Home Phoneline Networking Alliance (HPNA) is an association of industry-leading companies working together to ensure adoption of a single, unified phoneline networking standard and rapidly bring to market a range of interoperable home networking solutions [HPNA]. HPNA technology uses the same pair of wires as the existing analog telephone services to support a network in the home. The first version of the HPNA technology, which was introduced in 1998, offers a 1 Mbps data rate and uses the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) as its MAC protocol. The second version, known as HPNA 2.0, offers a 10 Mbps data rate and uses the DFPQ MAC protocol to support integrated traffic classified into 8 priority levels.

1.2. Quality of Service

QoS is defined as the desired performance specification of the network (the network service, data rate, expected performance, and other contractual information) as specified in service level agreements (SLAs) between service providers and their customers [CFF00]. QoS is an important consideration for home networks because communication resources are shared among a variety of service classes (e.g. internet telephony, video, file transfer, Internet access, etc), such that there is a trade-off between the efficiency of resource allocation and how well QoS is satisfied for all users. These different service classes have different bandwidth and latency requirements that have to be met by the network.

The parameters for service provision in today's integrated networks include delay, jitter, and packet loss ratio, in addition to a service contract that specifies transmission rate requirements. Delay, the time from when the frame is received in the transmit queue to its receipt at the destination, is an important metric for QoS because information contained within the frame often has a time limit after which it becomes obsolete. If a frame is delayed beyond its time limit, it has to be discarded by the receiving station. This is especially true of real time applications. Jitter is the measure of the variation in the delay metric. It is the difference between the highest and the lowest frame delay readings. A high jitter could result, for instance, in the flickering of screen in video transmission. Frame loss ratio is important because it determines how often the packet has to be retransmitted. Generally a high frame loss ratio would mean a high delay factor and, consequently, poor QoS.

Non-real-time traffic, such as best-effort data, can tolerate higher delay and jitter, whereas real-time traffic is generally characterized by its stringent delay and jitter requirements. Supporting QoS requirement is difficult in a distributed LAN system because there is no central agent to schedule stations' channel access so as to meet their QoS needs. Traditionally, different types of traffic with diverse QoS requirements are assigned different priorities, so that these priorities may be used to implement different packet handling procedures to provide differentiated services. It is to be noted, however, that QoS differentiation is strictly speaking not equivalent to prioritization.

In a common approach for supporting integrated traffic, real-time traffic is assigned a higher priority than non-real-time traffic, wherein higher priority traffic is given preference in accessing channel resources so that it experiences smaller delay and jitter than lower priority traffic. With proliferation of emerging multimedia applications, there are increasingly more classes of real-time and non-real-time traffic with diverse QoS requirements. Thus, growing number of priorities are used in today's communication systems to provide finer grain QoS.

1.3. Medium Access Control

The MAC layer is one of two sub-layers that make up the Data Link Layer of the Open Systems Interconnection (OSI) networking framework [BeG92]. It is responsible for moving data packets to and from one station to another across a shared channel. Some of the features of a MAC layer is the presence of a shared medium, a population of users and a centralized or distributed scheduling algorithm that allocates transmission opportunities among the users.

There are two basic kinds of MAC protocols: contention-free and contention-based. While contention-free protocols feature scheduled transmissions either statically, or adaptively among end users such that collisions are entirely avoided, contention-based MAC protocols feature channel contention among end users wherein collisions are possible. In this thesis, we focus on contention-based MAC protocols because of their wide deployment (e.g., Ethernet), efficiency in handling bursty traffic, and scalability. In most contention-free MAC protocols, idle users consume a portion of the channel resources which become major in bursty traffic or when the number of potential users is large. Contention-based MAC protocols, on the other hand, are more scalable to bursty traffic and the number of users in the system. Idle users do not transmit and hence do not consume any portion of the channel resources.

Contention-based MAC protocols have been studied extensively in the past and have been deployed in a variety of wireless and wire-line networking environments [RoS90]. They employ collision avoidance and collision resolution mechanisms. One widely used way to minimize collision is carrier sensing. Stations sense the channel before transmitting and if they find the channel busy, they defer their transmission until the channel is idle for a pre-specified duration. This ensures that stations do not collide with ongoing transmission. The duration of idle period could be the same for all stations or could be a function of some state of the station (e.g. priority, arrival time, etc). Collisions can still occur if two or more stations transmit at the same time and are resolved using a variety of collision resolution algorithms. Typically, collision resolution algorithm makes use of the feedback information on the contention outcome that is

available to the users via the common channel. The users can either detect the state of the common channel, or a centralized network entity can communicate that state to them. Depending on the type of channel states (or channel feedback) that is available, different collision resolution algorithms may be employed to resolve collisions. For the purpose of this thesis, a contention cycle is defined as a packet-scheduling pattern that occurs whenever the channel is idle and stations are competing for channel access. A contention cycle is terminated by a successful transmission or a collision.

A hybrid approach is also common which attempts to combine the two general MAC protocols. It is a contention-based reservation in that a contention-based approach is used for transmission of reservation requests, and contention-free approach is used for actual data transmission. Collisions involving actual data frames carry a high overhead especially when the frames are large. Instead small slots are used to reserve the time slot for actual data transmission. These schemes derive their efficiency from the fact that reservation periods are shorter than transmission periods by several orders of magnitude.

1.4. Outline of Thesis

This thesis is divided into 8 chapters, the contents of which are described below.

Chapter 2 summarizes some of the relevant MAC protocols for QoS provision in a de-centralized multi-access channel and discusses their relative merits and demerits in providing QoS. All the protocols discussed are extensions of the popular CSMA/CD MAC protocol.

Chapter 3 compares the throughput performance of two separate collision resolution methods used by Ethernet and the protocol developed in this thesis.

Chapter 4 highlights the problems addressed in this thesis and the approaches taken to do so. It also outlines the motivation behind the study.

Chapter 5 presents the new DDPQ MAC protocol.

Chapter 6 discusses the simulation models and parameters used for the simulation purpose. It summarizes and discusses the results obtained from the simulations and

compares the result for the new protocol against the benchmark protocols, namely DFPQ and CSMA/CD.

Chapter 7 provides a conclusion to the thesis work.

Chapter 8 lists the references cited in the thesis work.

Finally, we provide an appendix to describe the traffic source models used for performance evaluation, and Matlab[1] code for some algorithms used in the thesis.

[1] Matlab is a registered trademark of The Mathworks, Inc.

2. Survey of Contention-Based MAC Protocols

Since the DDPQ MAC protocol is a contention-based MAC protocol, we provide a survey of some existing contention-based MAC protocols in this chapter.

2.1. ALOHA Protocols

A well known contention-based MAC protocol is the family of ALOHA protocols. Many LANs today implement some sophisticated variants of ALOHA protocols. Because of their great popularity, analyses have been carried out for a large number of variations. The protocols within the family differ in their transmission and retransmission strategies and also in their adaptation to different circumstances and channel features. The following section covers some of these variations.

2.1.1. Pure and Slotted ALOHA

The original version operates with continuous or unslotted time and is referred to as Pure, or Unslotted, ALOHA [Abr70]. In this protocol, stations transmit their packets as soon as they are generated. If there is a collision in the system, all colliding stations schedule their retransmission at a random time in the future. The maximum throughput for Pure ALOHA has been determined to be 1/(2e), based on an analysis that assumes an infinite population and a Poisson packet arrival process [BeG92]. Specifically, consider a packet (old or new) scheduled for transmission in time t. Packet transmission is successful when there is no packet scheduled for transmission in the time interval (t-T, t+T), where T is the time it takes for a packet to be transmitted in the channel. The length of this interval, 2T, is called the vulnerable period. Since packets are transmitted soon after their arrival, the probability of success P_{suc} is given by the probability that no other packet arrives in the vulnerable period (length 2T). Assuming a Poisson process,

$$P_{suc} = e^{(-2gT)}$$

where g is the mean arrival rate.

Packets are scheduled at the rate of g of which only a fraction P_{suc} is successful. Since throughput S is defined as the fraction of time useful information is carried on the channel, we get

$$S = g \cdot T \cdot e^{(-2gT)}$$
$$G = g \cdot T$$
$$\therefore S = G \cdot e^{-2G}$$

where G is the normalized offered load. At G = 1/2, S takes on its maximum value of 1/(2e).

A later version of the ALOHA protocol, which operates with discrete or slotted time, and packets of a fixed size, is referred to as Slotted ALOHA [Rob75]. As with Pure ALOHA, new arrivals are immediately scheduled for transmission in the next slot. However, each transmission may start only at the beginning of a slot. Slot length, denoted T, is set to the duration of packet transmission. This means that a transmission is successful if there is no other packet arrival in the time interval (t-T, t), i.e. the vulnerable region is of length T. Thus, the throughput of Slotted ALOHA is

$$S = G \cdot e^{-G}$$

The maximum throughput for a Slotted ALOHA system occurs at G = 1 and has the value of 1/e.

Both Slotted and Pure ALOHA protocols have been shown to be unstable at high load. Fayolle et. al. [FLB74] prove that under high load condition the number of backlogged packets in the slotted ALOHA system will grow to infinity. As a result no packet will be transmitted and the expected delay for a packet will be infinite.

2.1.2. Pseudo-Bayesian Algorithm

Rivest [Riv87] proposes a Pseudo-Bayesian algorithm as a way to stabilize ALOHA. Instead of immediately offering newly arrived packets for contention, as is the case in ALOHA, they are transmitted with a probability p, where p is varied depending upon the estimate of the backlogged packet number. An estimate n' of the backlog number n is maintained. The probability of transmission p is then given by p = min (1, 1/n'). The algorithm essentially tries to achieve an attempt rate (G) of 1 for any slot. When the backlog number is large, p is small and hence it is unlikely that two packets will commence transmission in the same slot, thereby reducing the chances of collision.

The following rule is observed in updating the estimated backlog

$$\begin{aligned} n'(k+1) &= \max(\lambda, n'(k) + \lambda - 1) \text{ if success or idle} \\ &= n'(k) + \lambda + (e-2)^{-1} \text{ otherwise} \end{aligned}$$

λ is an unknown too and is either estimated as the time average rate of successful transmissions or is taken as a fixed value. Assuming that λ is known and is a fixed value, the expected delay for the pseudo-Bayesian algorithm is given by

$$W = \frac{e - 1/2}{1 - \lambda \cdot e}$$

As shown by Tsitsiklis [Tsi87] this system is stable for all values of $\lambda < 1/e$. However as λ approaches 1/e, the delay becomes unbounded.

2.2. Tree Splitting Algorithm

The basic idea behind splitting algorithms is to inhibit the transmission of newly arrived frames until all collided frames get transmitted. Following a collision, collided packets choose one of the multiple slots allocated to resolve the collision. There are various ways by which the stations involved in a collision could choose which allocated

slot to transmit in. They could choose one of the slots randomly or they could use the arrival time of their collided frames to determine the slot or they could use their addresses, etc.

The most basic splitting algorithm is the tree algorithm [Cap77], [TsM78], [Hay78]. The algorithm has a rooted binary tree structure; when a collision occurs in the *k*th slot, the collided stations randomly split into two subsets. The first subset transmits in the next slot (k + 1) and if the result is a success or an idle, the second subset transmits in the slot afterwards (k + 2). However, if a collision occurs in k + 1, the first subset splits again and the second subset waits for that collision to be resolved. In case of a ternary-tree splitting, the collided stations randomly split into three subsets rather than two. DFPQ employs ternary tree splitting with an important distinction. It uses mini slots, compared to the data slots used in normal splitting algorithms, to establish a partial ordering among the collided frames. Stations transmit in the order they have been organized.

Another popular splitting algorithm is the First-Come-First-Serve (FCFS) algorithm [BeG92]. In this method, the collided stations choose a subset based on the arrival time of the collided frames. Each subset will consist of all packets that arrived in some given interval. If a collision occurs again, this interval is split into two smaller intervals. By always transmitting the earlier arriving interval first, the algorithm will normally transmit successful packets in the order of their arrival time.

2.3. Ethernet Protocol and Extensions

Ethernet [MeB76] is a variation of the original Slotted ALOHA protocol. It augments the Slotted Aloha by utilizing carrier sensing i.e. instead of transmitting in the next slot after a frame arrives, stations wait until the channel is idle for a predetermined period. This avoids collision with an ongoing transmission. The Ethernet MAC protocol was originally designed for best effort service and hence does not provide any QoS support for real-time traffic. Simple as it is, the protocol does not have a scheme that distinguishes between real-time traffic and non-real-time traffic. As explained later, it

also suffers from an undesirable capture effect. A number of approaches that have been proposed to overcome the capture effect phenomenon are also surveyed in this section.

2.3.1. CSMA/CD Ethernet

The Ethernet MAC protocol is based on CSMA/CD, which is specified in the IEEE 802.3 standards document [802.3]. Incidentally, HPNA 1.0 is also based on this MAC protocol.

In the CSMA/CD MAC protocol (see Figure 2.1), a station that is ready to transmit senses the channel and waits until the channel is idle. It defers contention while the channel is busy. As soon as the station senses an idle channel for one inter-frame gap (IFG), it transmits its frame. An IFG is defined as the predetermined delay between two frame transmissions and is chosen such that all stations in the system perceive an idle channel i.e. no station will transmit when any station in the channel is still hearing a transmission. In the event of a collision, the protocol utilizes a Truncated Binary Exponential Backoff (TBEB) scheme for collision resolution [MeB76]. In this collision resolution scheme, a station involved in collision waits a random number of slots which is uniformly distributed between 0 and $2^{\min(k,\ 10)} - 1$ where k is the number of collisions suffered by the frame, before it tries to retransmit the frame. After a maximum of 16 retransmissions, the frame is discarded. The flow-charts in Figure 2.1 and Figure 2.2 describe the operation of the CSMA/CD MAC protocol.

A station is said to be one-persistent if upon sensing the channel to be busy, it does not defer its access attempt, but continues listening until the channel becomes idle whereupon it attempts transmission [ToK75]. A station is p-persistent if it defers its access attempt for some period of time with a probability of 1-p. The persistence feature has been used to enhance the QoS capability in some CSMA/CD-like MAC protocols [ChR85].

It has been shown that under moderate to high loads, TBEB results in a channel capture effect where a single host sends a lot of frames in succession while other nodes

are forced to wait a long time to access the channel [YaR94]. Capture effect occurs because only the "winning" station of a contention attempt gets to reset its collision counter. Recall that under TBEB, each station updates its collision counter independently, and only in response to a successful transmission attempt. During a contention interval, when several stations are contending for the channel, it is expected that each of them would possess a non-zero collision counter value. Eventually, when one of the stations gains access to the channel and transmits successfully, it will reset its collision counter. Note that at the next end-of-carrier event, all other stations will still have non-zero collision counter values. Thus, the "winning" station is free to transmit its next frame immediately, while other active stations must continue to backoff, resulting in a probabilistic last-in-first-out scheduling discipline that would impose considerable variations in contention access delay. In the event of a collision, the newly arrived frame chooses a much smaller backoff period compared to the frames that have been colliding repeatedly. As a result, the same station is likely to win the second time and the third time and so on, such that it transmits a stream of frames while other stations are forced to wait a long period of time before they can transmit. This phenomenon is known as a capture effect. Note that the capture effect enables a "lucky" host to monopolize the channel. The capture effect is especially detrimental for real-time applications. It has been shown that in the presence of a bursty traffic (with a burst size as small as 10KB) the capture effect severely degrades audio/video performance even at low network usage of 10-15%. [ToD96]

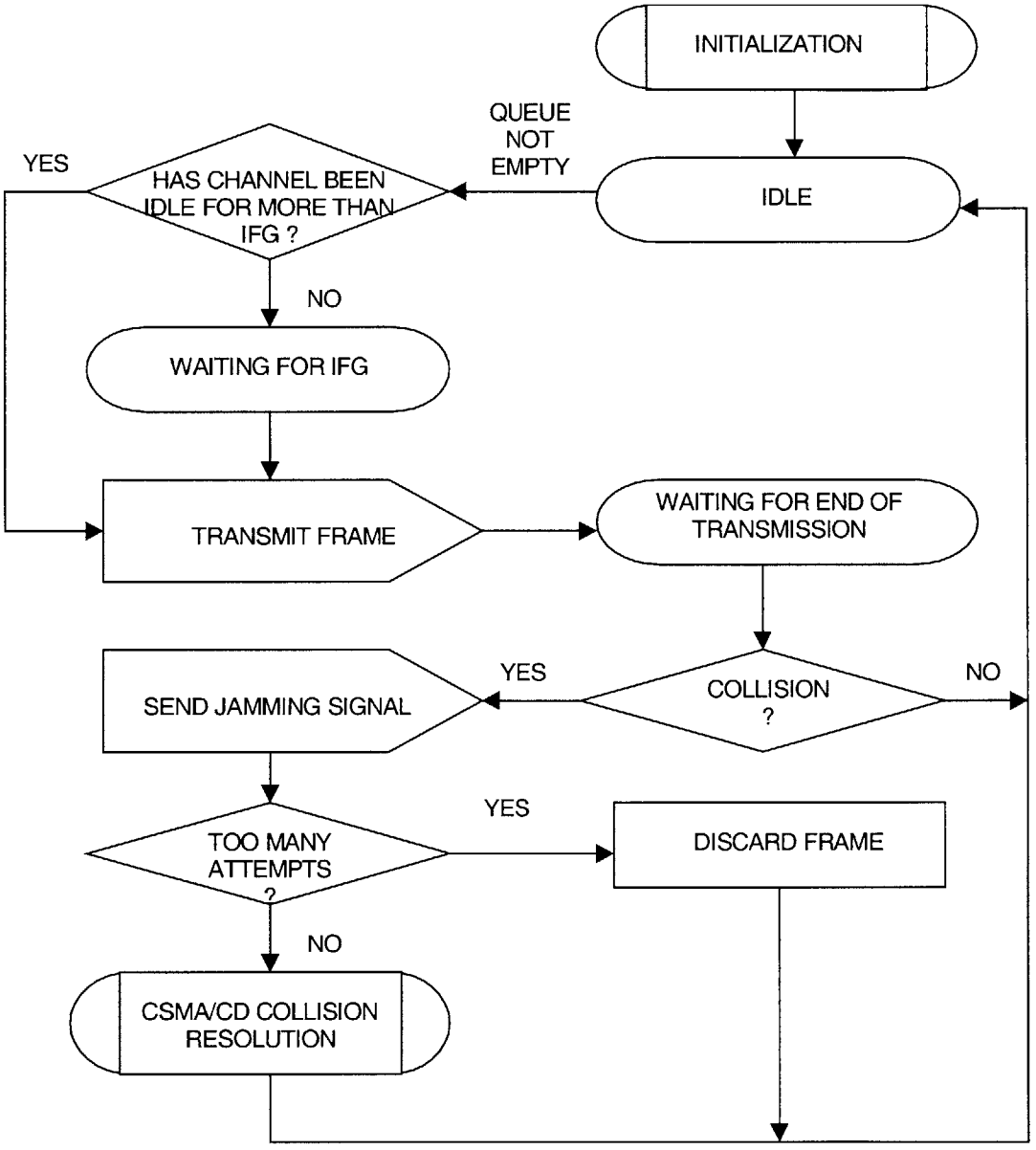


Figure 2.1: CSMA/CD MAC Protocol

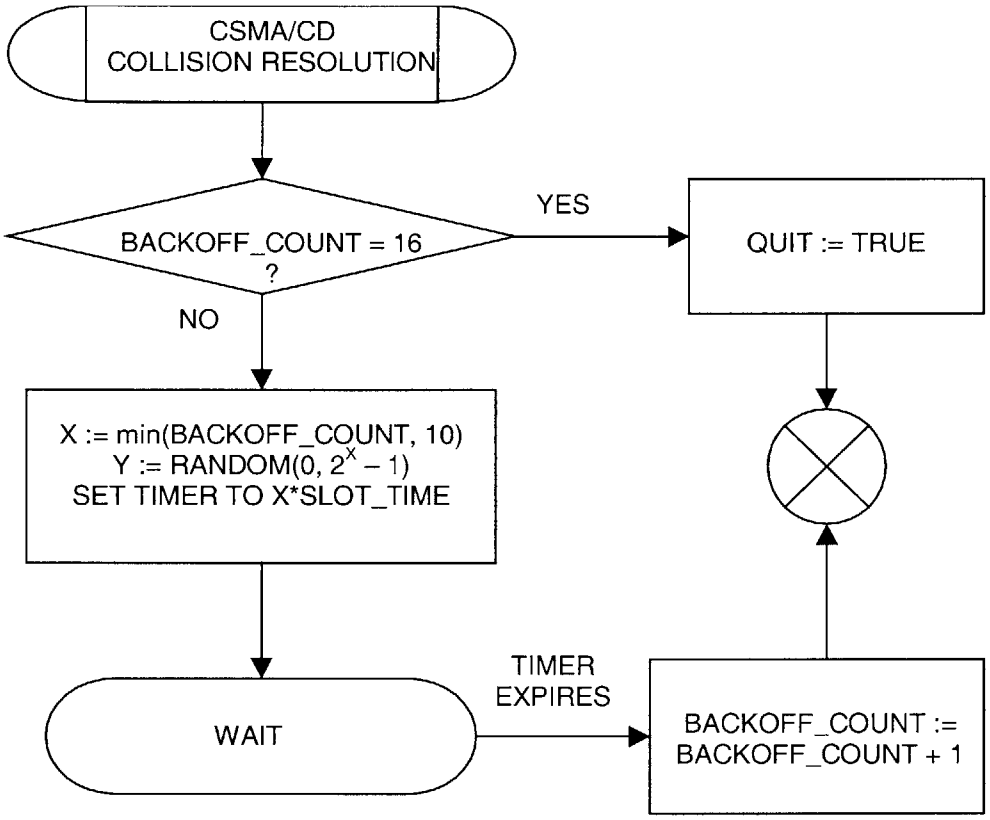


Figure 2.2: CSMA/CD Collision Resolution

2.3.2. Binary Logarithm Arbitration Method (BLAM)

In [Mol97], Molle proposes a MAC protocol that is interoperable with the standard IEEE 802.3 MAC protocol. This protocol was motivated by the need to address the capture effect in CSMA/CD. In this MAC protocol, a binary logarithmic arbitration method (BLAM) is introduced for collision resolution. Each station provides a collision counter that is incremented when any active station is involved in a collision, and reset when any station successfully transmits a burst of frames or after the station ceases transmitting for a predetermined period. Thus, the collision counters of transmitting and non-transmitting stations are updated symmetrically. Each station that is involved in a collision generates a random backoff delay from its collision counter using truncated exponential distribution. The station transmits its data after the backoff delay, unless another station is transmitting on the shared channel, in which case the station waits for the channel to become idle. Meanwhile if there is a collision in the channel, a new collision resolution cycle is triggered. It is worth contrasting the BLAM approach to the original CSMA/CD where stations only update their counter following their own activity. Because the collision counters of transmitting and non-transmitting stations are updated symmetrically in this method, we do not see the capture effect phenomenon.

2.3.3. Methods based on IFG Modification

Kalkunte, et al., have devised a number of methods that modify the IFG to address the capture effect in CSMA/CD. In one method [KaM98], if a station has just transmitted a frame and has another in its queue, then it must wait the IFG plus an additional slot time. The additional slot time pause allows other stations with frames waiting to acquire the channel.

In a more sophisticated method [KKM98], the delay times in Ethernet network devices are modified by adding an integer multiple of a delay interval to the minimum IFG, and decrementing the integer in each network station in response to a carrier sense. Each station has a unique integer value from the range of zero to the number of stations (n) minus one. The unique integer value ensures that each station has a different delay

interval in accessing the media after sensing an idle channel. The station having a zero integer value will get first access to the channel and it resets its integer counter to (n-1) after transmitting successfully. If it does not have any frame to transmit, it enters a deferral state where it waits out the remaining period in the contention cycle. All stations having nonzero integer values decrement their respective integer counters upon sensing a transmission. Thus the method of Kalkunte, et al. offers fair access by rotating indexes. Each station also includes a deferral timer that counts the maximum delay interval of (n-1) delay intervals plus the minimum IFG value, and thus establishes a bounded access latency for a half-duplex shared network.

This scheme scales poorly because the access latency bound increases linearly with the number of stations. In addition, the system incurs an overhead for idle stations. This overhead increases linearly with the number of stations. Besides, all stations in the system need to monitor the arrival or departure of stations in the network since n should be known accurately for the algorithm to work correctly. This might require significant overhead since stations can join in or leave the network at any time, thus requiring some signaling mechanism to inform the rest of the network when a station leaves the network or just establishes connection.

2.3.4. Longest-Waiting-Time-First Collision Resolution

In [SoK96], Sobrinho and Krishnakumar proposed an overlay mechanism to Ethernet's MAC protocol that provides QoS guarantees to real-time traffic. The approach relies on the ability of the network interface cards to sense channel activity and, in addition, requires only that those cards be capable of sending jam signals of pre-specified duration. Real-time host follows conventional Ethernet rules at the beginning of a session, possibly using a more expedite retransmission algorithm, to transmit its first frame. Subsequent frames are transmitted according to an overlay mechanism.

When a transmitting host detects a collision, it withholds its frame transmission but continues to jam the channel up to a pre-specified maximum allowed interval of time. This maximum duration is a direct function of the host's contention delay, measured from

the time that the access attempt has been scheduled until the host perceives the medium to be idle for an IFG interval. The jamming signal transmitted following a collision is called a black burst and is transmitted in discrete blocks called the black slot. The winner among contending hosts is thus the one that has been waiting the longest for access to the channel.

The collision resolution process distinguishes between a real time station and a non-real-time station. While real time stations utilize the jamming mechanism described above, non-real-time stations use the traditional TBEB mechanism to resolve collisions. And since they jam the channel immediately following a collision and are not randomly re-scheduled, real time sources have priorities over the data sources. Data source is blocked out until all real-time sources transmit their frames. Note that this first-come-first-serve collision resolution mechanism eliminates capture effect.

This protocol implicitly assumes that all real-time frames have the same delay requirement. It therefore gives channel access to the real-time station that has waited the longest. However, real-time traffics often have varying delay requirements in which case this method performs poorly. For instance, a frame that has been waiting for a long time would get access to the channel over a frame that has waited less even though the first frame could tolerate considerably higher delay than the latter frame.

2.4. Prioritized Contention-Based MAC Protocols

While capture effect is a huge problem for supporting real-time traffic on Ethernet MAC protocol, it is not the only problem. CSMA/CD MAC protocol is a best effort service i.e. it does not differentiate between real-time traffic and data traffic. In essence, both real-time traffic and data traffic are treated as equal even though the former has a strict delay bound whereas the latter can tolerate long delays. The approaches in the previous section for resolving capture effect fail to address this shortcoming of CSMA/CD.

A popular approach for enhancing the QoS capability of CSMA/CD implements priorities among different service classes. The priority values are then used to vary the

frame delay in a differentiated manner. A contending frame can experience two kinds of delays. It could incur a collision avoidance delay before being transmitted so as to minimize the probability of collision (for example, deferring transmission when a station finds a busy channel). It could also incur a delay when it collides and then undergoes collision resolution. This is called the collision resolution delay. One approach of prioritization varies the collision avoidance delay, so that higher priority frames enjoy preference over lower priority frames in gaining access to the channel as lower priority frames must defer initial contention in favor of higher priority frames. Another method varies the collision resolution delay, so that higher priority frames have smaller collision resolution delays than lower priority frames. Other methods exist that utilize a hybrid of the two methods described above.

In a prioritized contention-based MAC protocol, two types of collisions could occur: intra-priority collisions and inter-priority collisions. An intra-priority collision occurs when two or more stations of the same priority transmit at the same time. An inter-priority collision occurs when two or more stations of different priorities transmit at the same time. A prioritized contention-based MAC protocol could be designed to permit only intra-priority collisions. In this case, transmission resources are divided exclusively among service classes of different priorities. It could also be designed to permit both types of collisions. For example, all active stations, regardless of their priorities, may be permitted to contend for access to the channel initially, and only when they collide is a priority scheme utilized to impose different collision resolution delays to resolve the collision.

In this section, we present some of the popular protocols that use priority levels for QoS support.

2.4.1. Wait-Before-Transmit Priority Ethernet (WBTPE)

In [ObD93], Obaidat and Donahue propose a scheme for introducing priorities into the CSMA/CD protocol. In their scheme, referred to as WBTPE, each frame incorporates a random delay before transmission to allow higher priority frames first opportunity to transmit, and a collision delay which allows higher priority frames first opportunity to transmit following collisions. Specifically, the waiting is randomly selected from a range bounded between two values that depend on the slot time and the priority index of the frame. After a collision, an affected frame must wait and sense the channel for a period of time that is proportional to the slot time and the priority index of the frame.

The contention delay t_d experienced by a frame is computed as a function of its priority and given by

$$1.5 \cdot p \cdot t_{slot} \leq t_d \leq 1.5 \cdot (p + .5) \cdot t_{slot}$$

where p is the priority and t_{slot} is the length of priority contention slot. The smaller the value of p, the greater the priority of the frame.

A value of t_d is chosen randomly between the bounds so as to reduce the number of collisions between frames of equal priority. Higher priority frames get preference over lower priority frames because their contention delay period is shorter.

In the collision resolution process, a station automatically waits and senses the channel for t_d (= 2 * p * t_{slot}). If, during this wait period, there is an attempted transmission, the station waits for the completion of the transmission and repeats contention algorithm, else it implements the TBEB collision resolution algorithm of the CSMA/CD MAC protocol.

This approach suffers from excessive intra-priority collisions when the offered load is high in any priority. The randomization scheme is not very effective because the width of the period in which randomization occurs is less than a slot (worst-case propagation delay). Note that IFG length is equal to the slot length and that the IFG

length is determined such that stations can hear the transmission from any other station in the channel. Since the contention delays chosen by the stations are separated by less than IFG, the transmissions are more likely to collide.

2.4.2. Method based on Sharing of Contention Mini-Slots among Priority Classes

Ruszczyk, et. al. propose a contention-based reservation system for supporting multiple priority classes [RLC99a]. This prior system is based on a MAC protocol framework with contention-based reservation, also proposed by Ruszczyk, et al. [RLC99b]. In this framework, mini-slots are used for transmission of reservation requests in a contention-based approach whereas actual data are transmitted contention-free in data slots. Mini-slots are considerably shorter than transmission periods and hence entail smaller collision overhead; the ratio of mini-slot size to the transmission period determines the efficiency of the approach.

The prior system resolves collisions of reservation requests using a combination of probabilistic tree splitting and FCFS splitting techniques. The collision resolution procedure utilizes the FCFS splitting technique to select a collision resolution interval, and provides two contention mini-slots to improve the likelihood of successful reservations. Each contending user transmits a reservation request in a randomly selected contention mini-slot. Two contention mini-slots are provided for a predetermined maximum number of collision resolution iterations, after which only one contention mini-slot is provided.

A centralized controller allocates mini-slots for transmission of reservation requests among the stations. At the beginning of each contention cycle, the centralized controller transmits an entry poll message that includes feedback information for each contention mini-slot in the preceding contention cycle and an assignment of contention mini-slots for the current contention cycle. The system considers three kinds of feedback from a contention mini-slot. There could either be a SUCCESS or a COLLISION or an IDLE. The aggregate feedback state for the priority class is a function of the feedback states for all of the contention mini-slots assigned to that priority class. Specifically, the

aggregate feedback state is IDLE if the result is IDLE for all contention mini-slots; COLLISION if the result is COLLISION for at least one contention mini-slot; and SUCCESS otherwise. If the priority class is allocated zero contention mini-slots in a contention cycle, then the aggregate feedback state for the priority class takes on the value NONE, and the priority class is said to have sat out or be on vacation for the contention cycle.

In the prioritized system, the centralized controller allocates a variable number of contention mini-slots to each priority class, such that inter-priority collisions are avoided. The centralized controller keeps track of the feedback for each contention mini-slot, and for each priority class, determines an aggregate feedback state. Based on the aggregate feedback state the scheduler determines a preferred allocation of contention mini-slots, which it then uses to determine an actual allocation of contention mini-slots.

If the available number of contention mini-slots for a given contention cycle is less than the sum of the preferred numbers of contention mini-slots, the centralized controller allocates mini-slots in such a way that some of the priority classes get less than their preferred number of contention mini-slots. Essentially, the controller attempts to meet the requirements of higher priority classes, and allocates contention mini-slots to the lower priority classes only if contention mini-slots are available. On the other hand, if there is any mini-slot in excess of the sum of the preferred numbers of contention mini-slots, the remaining mini-slots are assigned one at a time to priority classes having zero contention mini-slots. If there are still more mini-slots remaining then they are assigned again one at a time to priority classes having one contention mini-slot. This process is repeated until either all contention mini-slots have been assigned or all priority classes have been assigned their maximum number of contention mini-slots.

2.5. Emerging MAC Standards for Home Networks

Many MAC protocols have emerged for home networks. They can be separated into wire-line (e.g., HPNA) and wireless (e.g., IEEE 802.11, HiperLAN) technologies. Because collision detection mechanism is not available in wireless medium, these

technologies are fundamentally different. In this section, we survey HPNA 2.0, IEEE 802.11, and HiperLAN MAC protocols.

2.5.1. HPNA 2.0

HPNA 2.0 MAC protocol supports eight priorities and is essentially a distributed Head-of-Line-Priority (HOLP) queuing discipline [FrH00], [G.pnt00]. In HOLP, a frame of a given priority is not served unless there is no higher priority frames pending transmission [Dai87]. The protocol does not support preemption, so a new arrival of a given priority must wait for the current frame transmission to end regardless of the priority of the frame.

In the first contention cycle upon start-up all active stations contend for channel access with the same lowest priority. Subsequently, inter-priority collisions are avoided by using separate contention slots for different priorities. All contention cycles consist of eight contention slots, each assigned to a separate priority level, ordered from high to low priorities. A user of a given priority class may not transmit in any contention slot of a higher priority. On the other hand, the user is permitted to transmit in any contention slot of its own or a lower priority, the latter case being an exception rather than a rule. Intra-priority collisions are resolved via a ternary tree splitting algorithm with partial ordering.

Each frame is prioritized a priori based on its need for QoS support. These priorities are used to determine the contention delay of a frame. Higher priority frames commence transmission in earlier contention slots and acquire channel before the lower priority frames. Note that this is a strict priority scheme, i.e. lower priority frames always defer to higher priority frames.

Although not a feature for QoS support, the collision resolution process of HPNA 2.0 MAC protocol employs a separate collision resolution mechanism, Distributed Fair Priority Queuing (DFPQ), that provides better performance than the TBEB mechanism of CSMA/CD [FrH00], [G.pnt00]. A period after each collision (following an IFG) is divided into three signal slots as shown in Figure 2.3. A collided station randomly chooses one of the slots and transmits a Backoff Signal in the slot. Note that more than

one station can transmit a Backoff Signal in the same signal slot, in which case they collide again. In essence, collision is resolved by means of a stack operation utilizing a collision resolution tree, which consists of three signal slots at each node. Because collision is resolved immediately and collided frames are transmitted before new arrivals, DFPQ does not exhibit the capture effect seen in TBEB. Access delay in DFPQ stays bounded even in the presence of excess load [FrH00]. This may be contrasted to TBEB where delay can be very high when the offered load is excessive.

Specifying priority levels for differentiated contention and resolving collisions with tree-splitting algorithm are common approaches for extending CSMA/CD. What is new to DFPQ though is the partial ordering through signal slots following a collision. The avoidance of any idle contention slots, in contrast to normal splitting algorithms, more than compensates for the small overhead of 3 signal slots required to establish the ordering. Each station keeps a Backoff Level (BL) and Maximum Backoff Level (MBL) counter. A BL counter indicates how many successful transmission of its priority a station has to wait before it can start contending again. An MBL counter specifies the BL value for a newly arrived packet. All stations monitor collision events and use the Backoff Signals to compute/update their BL and MBL values. The station(s) corresponding to the signaling slot 0 gets first access to the channel, followed by the station(s) corresponding to slot 1 and so on (a partial ordering is established among the frames that collided).

Note that stations with higher priority waiting frames may pre-empt the collision resolution process by transmitting in a contention slot earlier than the one in which the previous collision occurred. This means that higher priority frames do not wait for the collision to be resolved; only frames of the same priority or lower do.

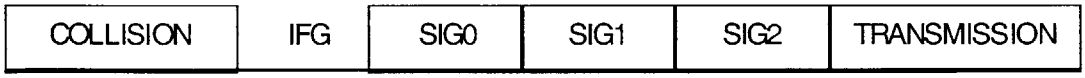


Figure 2.3: HPNA 2.0 Collision Resolution Mechanism

HPNA 2.0 MAC protocol has a few undesirable deficiencies. Intra-priority collisions are excessive when the offered load at any priority is high. Recall that the protocol has a rigid priority scheme, i.e. lower priorities always defer to higher priorities. Hence it does not effectively provide QoS guarantee for low priority users. Specifically, it is difficult to allocate bandwidth fairly, such that there can be some rate and/or delay guarantees for each priority class. Inter-priority collision avoidance may cause indefinite blocking to low priority frames. When only a few low priorities are present, channel resources are wasted because the contention slots allocated to all higher priorities must be waited out. Also since the scheme does not allow preemption, even the delay suffered by higher priority frames may be large in the presence of large data frames

2.5.2. IEEE 802.11

Another competing technology for home environment is the IEEE 802.11 MAC protocol, which has been adopted as a wireless standard by the IEEE body. This MAC protocol defines two access methods: Distributed Coordination Function (DCF) also known as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and Point Coordination Function (PCF) [ChG97].

The CSMA/CA MAC protocol [ChG97] is an adaptation of 802.3 CSMA/CD for wireless environment and features collision avoidance scheme since collision detection is not available in wireless environment. Like CSMA/CD, it is a best effort model but is expected to have a worse QoS performance due to greater interference from neighboring stations, absence of collision detection capability, and presence of hidden stations. The CSMA/CA MAC protocol defines two forms of medium contention: basic access method and channel access using Request to Send (RTS) / Clear to Send (CTS) frames.

In the basic access method, a station senses the channel before initiating transmission. If the channel is idle for an interval greater than IFG, it starts transmitting. If the channel is busy, it defers until after the channel is detected to be idle for IFG and then generates a random backoff period for an additional deferral time before transmitting. This further deferral minimizes collisions if multiple stations are

contending and is needed because there is no collision detection mechanism in wireless medium. Immediate positive acknowledgements are used to determine successful reception of data packets. In case an acknowledgement is not received, as is the case when there is a collision, the frame is presumed lost and a retransmission is scheduled.

If the data frame is large, it is better to use a different access scheme specified in the CSMA/CA protocol. The sender sends a RTS frame and the receiver responds with a CTS frame if it is ready to receive. A successful exchange of the RTS and CTS frames reserves the channel for the time duration needed to transmit the actual data frame. Since the RTS and CTS frames are small compared to the data frame, this scheme allows a fast collision detection and transmission path check. The rules for the transmission of the RTS frame are the same as those for data frames under basic access method. CTS frame is transmitted as an acknowledgement to the RTS. The overhead of RTS and CTS frames is justified only when the data frame is significantly larger than RTS and CTS frames.

The PCF [ViZ95] protocol specified as an alternative contention resolution method in the MAC layer protocol of IEE802.11 is designed to accommodate those services that have a strict delay requirement. It is a connection-oriented time division multiplexing access (TDMA) scheme, and provides contention free access to the polled stations. The access point is the point coordinator for a coverage area and it maps a prioritizing scheme to determine which station has the right to transmit. The station with the highest priority is polled first followed by the second highest and so on.

While wireless transmission would be a very attractive feature for home networking, 802.11 needs a better QoS support to be suitable for home networking. Of the two protocols outlined in the specifications for 802.11, CSMA/CA is a best effort service and does not support QoS requirements. And even PCF, which was claimed to be suitable for real-time traffic, has been shown to perform poorly for integrated data and voice transmission and is not scalable [ViZ95].

2.5.3. HiperLAN

HiperLAN is a standard for wireless LANs defined by the European Telecommunications Standards Institute (ETSI). The HiperLAN MAC protocol [ALM98] features a QoS support mechanism in the form of a dynamic priority scheme. The priority of a frame is calculated dynamically according to a predetermined mapping that takes into account both the initial user priority and the frame lifetime remaining. Frames that cannot be delivered within the allocated lifetime are discarded. The protocol also features a collision avoidance scheme similar to that of IEEE 802.11 MAC.

A station is allowed to transmit immediately if it senses the channel to be idle for IFG. Each data frame transmission must be explicitly acknowledged by the receiving station. If, however, the station senses a busy channel, it undergoes a prioritization mechanism at the end of ongoing transmission. It waits for the channel to be idle for p slots where p is the priority assigned to the frame. If the channel is sensed idle for p slots, the station transmits a burst in p + 1 slot, otherwise it stops contending and waits for the next contention cycle. All stations that transmit a burst in this phase again transmit a burst of B slots where B is a random variable with an exponential distribution. Following its burst if a station observes a busy channel, it forgoes contention and waits for the next contention cycle; otherwise it waits for a random number of slots before transmitting its frame. Once again if it senses a transmission while waiting, the station defers contention and waits for the next contention cycle.

This scheme is an improvement over the IEEE 802.11 MAC protocol. It features both a collision avoidance scheme and a QoS support mechanism for real-time frames. Collision avoidance incurs a huge overhead, which is justified in a wireless medium where collisions can only be identified after the receiving station performs an error check on the received frame but not in a wire-line medium where collision detection mechanism is available. There is also a large overhead involved with keeping the time-to-live value stamp for each frame (this has to be updated before each contention cycle).

3. Throughput Analysis for Ternary-Tree Splitting and BEB

In this section, we compare the throughput performances of Binary Exponential Backoff (BEB) and ternary-tree splitting collision resolution methods in slotted CSMA/CD. In TBEB, a station involved in collision waits a random number of slots which is uniformly distributed between 0 and $2^{\min(k,\ 10)} - 1$ where k is the number of collisions suffered by the frame, before it tries to retransmit the frame. In BEB, the backoff period is uniformly distributed between 0 and $2^k - 1$. Since the truncated nature of TBEB only occurs rarely and it makes the analysis complicated, we only analyze BEB for the sake of simplicity. Ternary-tree splitting is the collision resolution method for DFPQ as well as the protocol developed in this thesis. An analytical comparison would serve to highlight the differences in the two fundamentally different collision resolution approaches and to shed light on later simulation results. The analysis for BEB is based on the work of Tobagi and Hunt [ToH80] and the same approach is used for ternary-tree splitting. It is assumed that there is an infinite population of users. The frame arrival is a Poisson process with a given mean arrival rate.

3.1. BEB analysis

The channel alternates between busy periods (containing successful transmission and collision periods) and idle periods. The success or failure of a transmission period in the busy period depends only upon the length of the preceding transmission period. Given the transmission period is of length x, the length of the remainder of the busy period is a function of x and its average is denoted by B(x). Similarly, the average time that the channel is carrying successful transmissions in the remainder of the busy period is denoted by U(x). The average length of an idle period I is $\tau/(1-e^{-g\tau})$ [RoS90], where τ is the propagation delay. Let $a_n(x)$ give the probability of n arrivals in a period of length x. Then,

$$B(x)=\frac{a_1(x)}{1-a_0(x)}\left[T+\tau+[1-a_0(T+\tau)]\cdot B(T+\tau)\right]+\left(1-\frac{a_1(x)}{1-a_0(x)}\right)\left[\gamma+\tau+[1-a_0(\gamma+\tau)]\cdot B(\gamma+\tau)\right]$$

$$U(x) = \frac{a_1(x)}{1-a_0(x)}\left[T + [1-a_0(T+\tau)]\cdot U(T+\tau)\right] + \left(1 - \frac{a_1(x)}{1-a_0(x)}\right)\left[[1-a_0(\gamma+\tau)]\cdot B(\gamma+\tau)\right]$$

The first term in the expression for B(x) corresponds to a single packet arriving during x in which case its transmission takes T + τ time. If there is at least one arrival in T + τ (given by probability 1- a_0(T + τ)), the remainder of the busy period is denoted by B(T + τ). The second term corresponds to more than one packet arriving during x, in which case the busy period is of length γ + τ. If there is at least one packet arrival during this time, the remainder of the busy period is given by B(γ + τ).

The expression for U(x) is also similar except that in a transmission time of T + τ, the total utilized time is T, and in case of a collision, there is no utilized time. Note that just like in the case of B(x), the additional utilized time is given by U(T + τ) and U(γ + τ).

The above set of equations can be solved by substituting for x = T + τ and x = γ + τ and finding an expression for B(T + τ), B(γ + τ), U(T + τ), and U(γ + τ). The system throughput, S is given by

$$S = \frac{U(\tau)}{B(\tau) + I}$$

3.2. Ternary-tree Splitting analysis

Ternary-tree splitting method can be analyzed in a similar manner, with one notable difference. Whenever there is a collision, all the frames involved in a collision are resolved before any newly arrived frames get transmitted. Therefore, the second terms in the expressions for both B(x) and U(x) are different. The collision resolution length (CRL), instead of being just γ + τ as in the equations above, is a function of frame multiplicity (number of frames involved in the collision) and the expected number of cycles for that multiplicity before the collision is fully resolved. It is given by,

$$CRL(x) = \sum_{n=2}^{\infty} a_n(x)\cdot E(n)$$

where E(n) is the expected length of the collision resolution period in ternary-tree splitting for n frames. E(n) is derived from the work by Xu and Campbell [XuC93] and is given by,

$$
\begin{aligned}
E(n) &= 1 && \text{if } n=0 \text{ or } 1, \\
&= 1.5 && \text{if } n=2 \\
&= \frac{1+\sum_{k=2}^{n-1}\binom{n}{k}\cdot 2^{(n-k)}\cdot E(k)}{3^{(n-1)}} && \text{otherwise} \\
&\quad 1-3^{(1-n)}
\end{aligned}
$$

Since the collision resolution period (CRP) in ternary-tree splitting also involves the successful transmission of the collided frames, its CRL is substantially longer than in BEB and its utilization period, instead of being 0 as in the BEB equations for B(x) and U(x), is equal to the successful transmission times for all collided frames.

The figure below plots throughput against offered load for BEB and ternary-tree splitting. In case of ternary-tree splitting, we assume that any collision involving more than 6 frames is negligible. Please see Appendix A.2 for the derivation of the throughput plots. As can be seen from the plot, the throughput for BEB is significantly higher than that for ternary-tree splitting method especially at higher offered loads. This observation is also supported by simulation results (see Figure 6.2). The difference is due to higher number of collisions in ternary-tree splitting than in BEB. Recall that in the ternary-tree splitting method, all collided frames are transmitted first. Hence, it is more likely that there will be a number of backlogged stations (that have frame arrivals during CRP) immediately following the CRP; this would, in turn, lead to more collisions. In the case of BEB, it schedules a collided frame to be transmitted in some future time, determined randomly. Although unfair (capture effect), this process is very efficient in increasing the overall system capacity and reducing collisions.

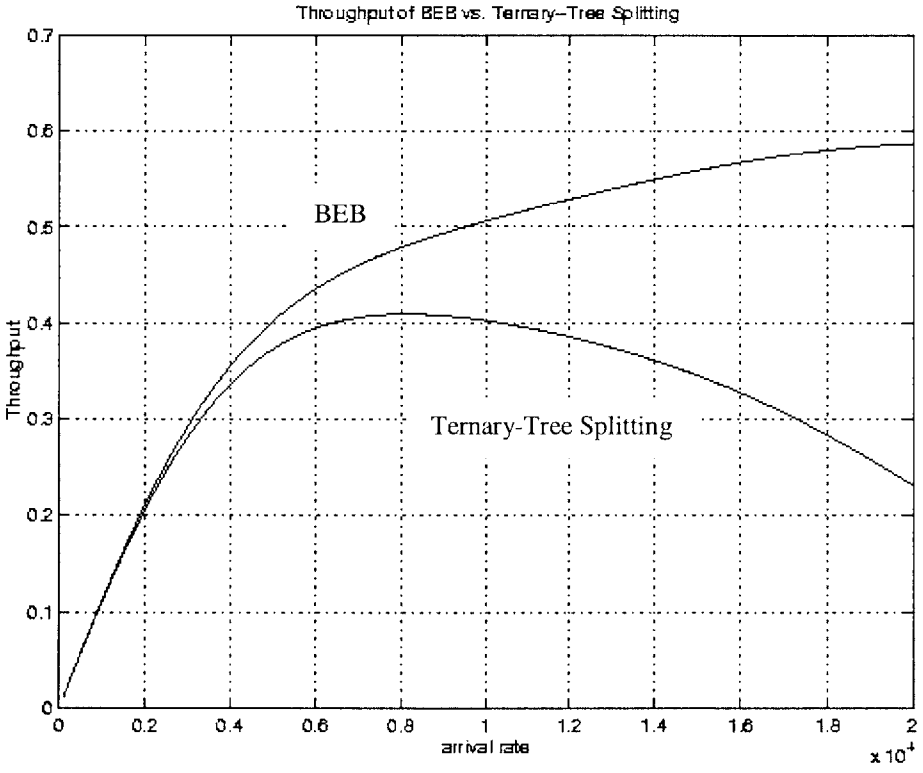


Figure 3.1: Comparison of BEB Throughput against Ternary-Tree Splitting Throughput

4. Objectives of Research

We have learned that many existing MAC protocols do not effectively address the problem of fair medium contention resolution in home networks while also providing QoS support to integrated traffic. The end objective of this research is to propose a MAC protocol that would effectively address this problem.

We select the DFPQ MAC protocol, of HPNA 2.0, as a starting point because of its simplicity, as well as its ability to support priorities and deal with capture effect. The protocol has, however, some undesirable deficiencies that could impair its performance in home networks. For example, intra-priority collisions are excessive when the offered load at any priority is high. Recall that the protocol has a rigid priority scheme, i.e. lower priorities always defer to higher priorities. Hence it does not effectively provide QoS guarantee for low priority users. Specifically, inter-priority collision avoidance may cause indefinite blocking to lower priority frames. And when only a few low priorities are present, channel resources are wasted because the contention slots allocated to all higher priorities must be waited out. Also since the scheme does not allow preemption, even the delay suffered by higher priority frames may be large in the presence of large data frames.

In this thesis, we aim to address the problem of excessive intra-priority collisions in DFPQ system while preserving the distributed nature of the protocol. While we also motivate some ways of addressing the indefinite blocking of lower priorities, we believe our approach will resolve the problem to a large extent. Our protocol is distributed, i.e. it does not require a central controller. This is very important from a design perspective because it eliminates the problem of a single node failure. In centralized networks, if the scheduler fails, the whole network goes down. A distributed network, on the other hand, is able to bypass the problem because no single node is critical for the functioning of the network. Besides, distributed networks are relatively easy to install.

The multiaccess communication model studied in this thesis is a distributed system, which uses an Ethernet-like contention resolution scheme. It is assumed that stations can sense the channel and hear their own transmission. In this scheme, there is

no separate control channel, and no central scheduler with global knowledge coordinating channel access for the users. A major goal is to provide good delay and jitter performance to real time traffic while also not unnecessarily penalizing data traffic.

5. Distributed Dynamic Priority Queuing

As discussed in chapter 2, existing approaches do not adequately address the issue of QoS provision in a distributed channel. We propose a new protocol, called the Distributed Dynamic Priority Queuing (DDPQ) in this section, which is an extension of the DFPQ MAC protocol. The new protocol estimates the number of backlogged stations at each priority and splits 8 priority slots among the priority levels such that chances of collisions, both intra-priority and inter-priority, are minimized. Note that this is in contrast to the present DFPQ protocol which has static slot allocation scheme and which resolves collisions only after they occur. Collision resolution incurs a significant overhead -- CSMA/CD collision detection length + 3 signal slots required for the partial ordering of collided stations.

Our approach consists of estimating the number of backlogged stations associated with each priority, dynamic slot allocation among the priorities based on the estimated backlog number, and a distributed mechanism for the stations to synchronize their slot allocation scheme. -The goal is to allocate contention slots among different priorities to minimize intra-priority collisions while dynamically adapting to a changing mix of offered loads associated with the priorities. Each station encodes its slot allocation scheme for the next contention cycle in a slot allocation profile. This profile is 2 bytes long and is piggybacked to the transmitted frame by a successful station. All listening stations check their profile against the transmitted one and, in case there is a difference, they start a synchronizing process.

The flow charts in Figure 5.4 to Figure 5.21 describe the operation of DDPQ MAC protocol. Just like in the DFPQ protocol, a new contention cycle starts after the end of a successful transmission or after a collision. Each station has an estimate for the number of backlogged stations in each priority. The estimation scheme is described in section 5.4. Based on the backlog number estimate, slots are allocated among the priorities such that the instantaneous throughput for that contention cycle is maximized. This slot allocation process is described in detail in section 5.5.

5.1. Profile Synchronization

Stations coordinate among themselves in a distributed manner to synchronize their slot allocation. It is necessary to have a synchronizing mechanism in order to correct erroneous stations and provide an initial slot allocation profile for stations that are recently active. A synchronized slot allocation scheme also ensures that there are no inter-priority collisions; inter-priority collisions are undesirable because they could lead to unfair priority reversal i.e. lower priority frames could get transmitted before higher priority frames. Note that the DFPQ *Collision Resolution* process (see Figure 5.20) employed by our protocol for resolving collisions assumes that all colliding frames have the same priority. Hence, in the event of an inter-priority collision, lower priority frames could get access to the channel before higher priority frames.

It is useful to clearly distinguish among the following profiles:

Null Profile: The profile that allocates no slot for any priority.

Initial Profile: The profile that allocates one slot for each priority.

Local Profile: The profile that is generated locally by a station and believed to be the correct profile.

Advertised Profile: The profile that is piggybacked to the transmitted frame i.e. the *Local Profile* of the transmitting station.

Contested Profile: An *Advertised Profile* that is different from the *Local Profile* of one or more stations.

Global Profile: The profile that is most recently advertised and uncontested, i.e. synchronized.

Figure 5.1 shows the state diagram for the profile synchronizing finite state machine (FSM) of a station. A new station enters the *updating* state if it sees that there is an ongoing transmission and that the system is synchronized. In this state, it inherits the *Advertised Profile* as the *Global Profile* and uses it for contention resolution in the next contention cycle. If the station acquires the channel, it passes a *Null Profile*. A *Null Profile* piggybacked with the frame indicates all stations to use previous *Global Profile*

for the next contention cycle. The new station remains in the *updating* state until its *Local Profile* matches the *Advertised Profile*. If, however, no carrier sense is detected until a timeout period, the station presumes that it is the first station to be transmitting in the system and enters the *new_sync* state. In this state, the station builds its arrival rate estimate for each priority over a window period before transmitting its *Local Profile*. Until then it transmits the *Initial Profile*. While in the *new_sync* state, if the station hears a successful transmission and the *Advertised Profile* is different from the *Initial Profile*, it determines that the system is synchronized and enters the *updating* state. In both *updating* and *new_sync* states the station refrains from jamming any profile.

An active station is in one of two states: *normal* and *synchronizing*. A station is in the *normal* state if its profile is synchronized to the system. Whenever a station senses a profile-jamming signal, it enters the *synchronizing* state. In *synchronizing* state, the station resets its synchronizing counters and its arrival rate estimate for the priorities, and it uses the *Initial Profile* to allocate slots for a period of window length. The synchronization counters of a station keep track of the number of times the station has been jammed and the number of distinct stations it has jammed. These numbers are useful in identifying a situation where a station is continuously erroneous.

Each station checks its *Local Profile* against the *Advertised Profile*. If the *Advertised Profile* matches the *Local Profile* of every station, the next contention cycle occurs as specified by the *Advertised Profile*. If any station disagrees and it is in not a new or updating station, it jams the channel as soon as the channel becomes idle and triggers the *synchronization* state. The jamming signal has a pre-specified duration so that it can be distinguished by all stations.

Note that whenever a profile jamming occurs, all stations reset. Resetting only the station that initiates the jamming or the station that is jammed could lead to a situation where all stations except an erroneous station reset. This could happen if all stations jam when an erroneous station transmits or when a "correct" station is jammed by an erroneous station. In such a situation, the "correct" station may never be able to

synchronize with the *Advertised Profile* of the erroneous station and will always remain in the *updating* state.

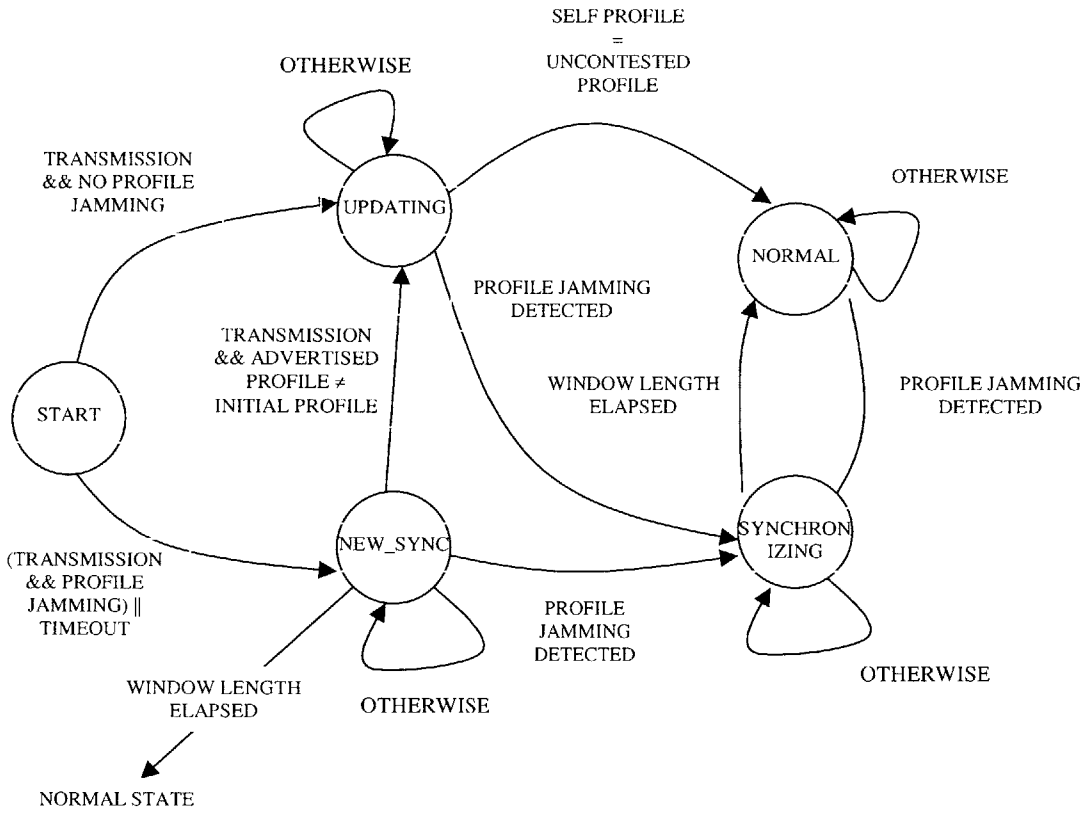


Figure 5.1: Profile Synchronization FSM

Each station keeps two counters to count the number of times its profile has been contested successively and the number of unique stations it has jammed. The purpose of these counters is to identify an erroneous station. If a station has jammed certain number of distinct stations, it is presumed to be erroneous. In that event, the station raises an error flag and halts activity. When a station has been successively jammed for a certain number of times, it ceases contention for one cycle so that some other station wins in the next contention cycle. This condition preempts a cyclical situation wherein, after the system has undergone the synchronizing process, the same station transmits and is jammed by the same erroneous station. Note that the counter for stations jammed only increases when a distinct station is jammed. In the situation outlined above, the system will continually reset without actually moving the counter for distinct stations i.e. the erroneous station is not detected. If no station contends in that cycle, the station initializes its counter and proceeds like a normal station. With this method the erroneous station will eventually be detected when enough distinct stations transmit.

The synchronizing mechanism is presumed to occur only rarely and, in any case, it will still give the same performance as DFPQ since the slot allocation scheme while the system is synchronizing is the same as DFPQ.

5.2. Profile Encoding

An eight-priority slot allocation can be encoded with two bytes. The first byte represents whether or not the priority corresponding to that position (decreasing from 7 to 0) has any slot allocation. A bit value of "1" means that the priority is allocated at least one slot, while a value of "0" means no slot allocation for the priority. The second byte represents where a priority change occurs, i.e. the slot has been allocated to a different priority. Each time a change occurs, the next bit is flipped. Together the two bytes include all the information contained in a slot allocation profile and can be decoded to obtain the information. Figure 5.2 gives an example for encoding a slot allocation using this algorithm.

Slot Allocation Scheme

(11000101)(01110011)

Encoding the Allocation Profile

Figure 5.2: Example of Profile Encoding

We now proceed to prove the correctness of our encoding algorithm. We prove that there exists a unique slot allocation scheme for any profile and vice versa.

i. There is only one slot allocation scheme for a profile.

A slot allocation scheme would differ from another slot allocation scheme if

a. it has different mix of priorities that are allocated slots or

b. at least one priority that is allocated one or more slots has different number of slots allocated to it.

If (a) were true, the first byte would be different since each bit represents whether or not the corresponding priority is allocated any slot. If (b) were true, the second byte would be different because the length of consecutive ones or zeros would be different for that priority. Hence, no two different slot allocations can have the same profile or that for any profile there is only one unique slot allocation.

ii. There is only one profile for a slot allocation.

Let us assume that there are two different profiles, A and B, for a slot allocation. Either

a. A's first byte is different from that of B or

b. A's second byte is different from that of B

If (a) were true, there would be different mix of priorities in the slot allocation corresponding to A from that of B. This is a contradiction. If (b) were true, the priorities would have different number of slots allocated to them in A and B. This is also a contradiction. Hence there can only be one profile for a slot allocation.

Since there is a bi-directional one-to-one mapping, the algorithm is correct. Figure 5.16 shows the flow chart diagram for encoding the profile and Figure 5.19 shows the flow chart decoding the profile.

5.3. Flow chart description

Figure 5.3 shows the flow chart for DDPQ MAC protocol. Just like in CSMA/CD MAC protocol, it consists of a module wherein the MAC station initializes its system parameters - DDPQ *initialization*. After initialization, the station enters the DDPQ *carrier sense* module where it waits for frame arrival. When the station has a frame to transmit, it enters DDPQ *collision avoidance* process after sensing the channel to be idle for IFG period. In the DDPQ *collision avoidance* process it randomly chooses one of the slots assigned to its priority by the *Global Profile*. If the station is not blocked it proceeds to the DDPQ *transmission processing* module; otherwise, it enters DDPQ *blocked processing* module.

In DDPQ *transmission processing* module, the station piggybacks its *Local Profile* for the next contention cycle with the frame and transmits it. In case of a collision, it undergoes DFPQ *collision resolution* process. Based on the decision in the DFPQ *collision resolution* process, it either discards the frame or retains it. It then reverts back to DDPQ *carrier sense* process.

In DDPQ *blocked processing* module, the station checks its *Local Profile* against the *Advertised Profile* and depending on the result decides whether or not to force *profile synchronizing* process. It reverts back to DDPQ *carrier sense* process after exiting the DDPQ *blocked processing* module.

We now describe each process individually.

DDPQ *initialization*:

Figure 5.4 shows the *DDPQ initialization* process. All the system parameters of the MAC station are initialized in this process. The parameters and their initial values are provided in Table 5.1. The station waits for a carrier sense. If it senses a successful transmission and there is no profile-jamming signal, it enters the *updating* state and inherits the *Advertised Profile* as its *Global Profile*. If it senses a successful transmission followed by the profile-jamming signal, it enters the *synchronizing* state and uses the *Initial Profile* as its *Global Profile*. If it does not sense a successful transmission by the end of a timeout period, it enters the *new_sync* state and uses the *Initial Profile* as its *Global Profile*.

DDPQ *carrier sense*:

Figure 5.5 shows the DDPQ *carrier sense* process. In this process, the station first executes the DDPQ *check synchronization* process. It then checks to see if its transmit queue is empty i.e. there is no packet waiting to be sent.

If the queue is empty, the station waits until there is a packet arrival. While waiting, if it senses channel activity, it executes the DDPQ *update transmission* process. Upon arrival, a new frame enters the DFPQ *update backoff level* process.

If the queue is not empty or there has been a new packet arrival, the station checks to see if the channel has been idle for more than IFG period in which case it exits out of the process. If the channel has not been idle for IFG period, the station waits until that is

the case. Once again while waiting, if the station senses channel activity, it executes the DDPQ *update transmission* process.

DDPQ *check synchronization:*

Figure 5.6 shows the DDPQ *check synchronization* process. In this process, the station checks the value of NOT_IN_SYNC. A value of 1 would indicate that its *Local Profile* is not consistent with the *Advertised Profile* and it jams the channel for t_sync period to force a DDPQ *profile synchronizing* process. If the station detects a jamming signal of duration t_sync, it enters the DDPQ *profile synchronizing* process. If no profile-jamming signal is sensed, a listening station sets its *Global Profile* equal to the *Advertised Profile*. In case of transmitting station, it only does so if it is in the *normal* state.

DDPQ *update transmission:*

Figure 5.7 shows the DDPQ *update transmission* process. In this process, the station first executes the DFPQ *update backoff level* process.

If the channel activity is due to a collision, it exits out of the process.

If the channel activity is due to profile jamming, it executes the DDPQ *check synchronization* process.

If the channel activity is due to a successful transmission, it gets its *Local Profile* using the *get profile* process. It then executes the DDPQ *check profile* process to compare the profile with the *Advertised Profile*.

DDPQ *check profile*:

Figure 5.8 shows the DDPQ *check profile* process. In this process, the station increments the TRANSMITTED_FRAMES[] counter for transmitted priority.

If it is in the *synchronizing* state, it exits out of the process.

If the station is in the *new_sync* state, it checks to see if the *Advertised Profile* is the same as the *Initial Profile*. If the profile is different, the station enters the *updating* state and exits out of the process.

If the station is in neither of the two states above, it checks to see if its *Local Profile* is the same as the *Advertised Profile*. If the profile is the same as the *Advertised Profile* and the station is in the *updating* state, it changes its state to *normal* and initializes all of its synchronizing counters. If the station is not in the *updating* state, it checks to see if the transmitting station has been listed as one of the stations that it had jammed. If so it decrements its STATIONS_JAMMED counter. If the *Local Profile* is different than the *Advertised Profile*, it checks to see if the *Advertised Profile* is a *Null Profile*. If so the station sets the *Advertised Profile* to the *Global Profile*. Otherwise, the station checks to see if it is in the *updating* state. If it is not in the *updating* state, it sets its NOT_IN_SYNC to 1 before exiting out of the process.

DDPQ *profile synchronizing*:

Figure 5.9 shows the DDPQ *profile synchronizing* process. In this process, the station uses the *Initial Profile* for contention and switches its state to *synchronizing*. If its profile was contested, it increments the TOTAL_NUMBER_JAMMED. If it is the station that jammed the channel and the station it jammed is a new one (i.e. not in its list of stations jammed), it increments its STATIONS_JAMMED counter. If STATIONS_JAMMED counter reaches its threshold value, the station raises an error flag and halts activity. Likewise if TOTAL_NUMBER_JAMMED reaches its threshold value, the station forgoes transmission for one contention cycle.

DFPQ *update backoff level*:

Figure 5.10 shows the DFPQ *update backoff level* process [FrH00], [G.pnt00]. In this process, the station checks to see if the frame waiting is newly arrived. If so, the BACKOFF_LEVEL value for the frame is determined by its MAX_BACKOFF_LEVEL

value. Else if a collision is sensed and the station is not involved, the station waits out three signal slots and the IFG period. This ensures that all stations defer while the system is establishing partial order among the collided stations. If the collision is in the same priority as the station or it is one of the collided stations, its BACKOFF_LEVEL and MAX_BACKOFF_LEVEL values are established based on the previous values and the Backoff Signals seen before its signal slot (which is all if the frame did not itself collide). If instead there is a valid transmission in the priority, its BACKOFF_LEVEL and MAX_BACKOFF_LEVEL values are decremented.

DDPQ *collision avoidance*:

Figure 5.11 shows the DDPQ *collision avoidance* process. In this process, the station checks to see if its BACKOFF_LEVEL has a nonzero value. If so, the station is blocked and it exits the process. Otherwise it decodes the *Global Profile*. If the slot allocated to its priority is 0, the station sets it temporary priority value to 8. Else it sets its temporary priority value to be equal to the sum of the slots assigned to the higher priorities and the random number chosen between 0 and the number of slots assigned to it by the *Global Profile*. Having set the temporary priority value, the station waits out all the higher priority slots. If any channel activity is detected, the station is blocked and it exits the process. Otherwise, it enters the DDPQ *transmission processing*.

DDPQ *transmission processing*:

Figure 5.12 shows the DDPQ *transmission processing* process. In this process, the station first gets its *Local Profile* for the next contention cycle using the DDPQ *get profile* process. This profile is piggybacked with the frame and transmitted. If the station is in the *updating* state, it transmits a *Null Profile*. If the station is in the *synchronizing* or *new_sync* state, it transmits the *Initial Profile*. If a collision is detected, the station jams the channel and enters the DFPQ *Collision Resolution* process. Otherwise it enters the

DFPQ *update backoff level* process to update its backoff level counters before exiting the process.

DDPQ *blocked processing*:

Figure 5.13 shows the DDPQ *blocked processing* process. In this process, the station first sets its BLOCKED variable to FALSE, thus ensuring that it is blocked for only one contention cycle. It then gets its *Local Profile* using the DDPQ *get profile* process.

While waiting for the channel to go idle, if the station senses a collision or a successful transmission, it executes the DFPQ *update backoff level* process. If the carrier sense was due to a successful transmission, it executes the DDPQ *check profile* process.

DDPQ *get profile*:

Figure 5.14 shows the DDPQ *get profile* process. In this process, the station obtains estimate for the number of backlogged stations at each priority level using the previous backlog number estimate, the number of packets successfully transmitted after the last contention cycle (which is multiple if the priority level enters the collision resolution interval), total waiting time since last contention cycle, and the arrival rate estimate for the priority. It then executes the DDPQ *assign slots* and DDPQ *encode profile* processes to optimally allocate slots among the priority levels and encode the allocation.

DDPQ *assign slots*:

Figure 5.15 shows the DDPQ *assign slots* process. In this process, the station allocates slots optimally among the 8 priority levels given the backlog number for each priority. In the first step, each priority is assigned slots equal to the number of backlogged stations estimated for the priority going from the highest priority to the

lowest. If the total number of slots allocated is equal to 8, slot allocation is completed. If there are still extra slots remaining to be assigned, once again the station allocates slots to each priority proceeding from the highest to the lowest such that the number equals the number at which maximum ICT occurs. This number is given by $x\ (= b + \lfloor b/6 \rfloor + 1)$ where b is the number of backlogged stations for the priority. Slot allocation is completed when the total number of slots allocated is equal to 8. If there are still extra slots remaining to be assigned, the remaining slots are assigned one slot per priority to the idle priorities going from highest priority to the lowest until there is no more unallocated slot.

DDPQ *encode profile*:

Figure 5.16 shows the DDPQ *encode profile* process. The slot allocation profile, as specified by the DDPQ *assign slots* process, is encoded into two bytes. The first byte is encoded by the DDPQ *first byte allocation* process and the second byte is encoded by the DDPQ *second byte allocation* process.

DDPQ *first byte allocation*:

Figure 5.17 shows the DDPQ *first byte allocation* process. Each of the bits in the first byte corresponds to the priority with the first one corresponding to the highest and the last one corresponding to the lowest. A bit value of 1 denotes that at least one slot has been allocated to the priority whereas a bit value of 0 denotes that there is no slot allocated to the priority.

DDPQ *second byte allocation*:

Figure 5.18 shows the DDPQ *second byte allocation* process. The second byte notifies how many slots have been allocated to each priority that has a one in its position in the first byte. A list of consecutive ones or zeros indicates that the slots in those

positions have been allocated to the same priority. The bit is flipped every time there is a priority change. Proceeding from the first slot to the last (note there are eight slots) and starting with a zero, the next bit is assigned the same value as the current if the next slot is allocated to the same priority and reversed if the next slot is allocated to a different priority.

DDPQ *decode profile*:

Figure 5.19 shows the DDPQ *decode profile* process. Two counters are specified which track the first byte and the second byte. The bit value of the first byte denotes whether or not the priority corresponding to the bit position is allocated any slot. The second counter counts the number of consecutive ones or zeros which gives the number of slots assigned to the corresponding priority (given by the first counter).

DFPQ *collision resolution*:

Figure 5.20 shows the DFPQ *collision resolution* process [FrH00], [G.pnt00]. In this process, the station chooses one of the three signal slots randomly. It then waits for its signal slot in which it transmits a backoff signal. At the end of the three signal slots it executes the DFPQ *update backoff level* process.

DDPQ *Update Arrival Rates*:

Figure 5.21 shows the DDPQ *Update Arrival Rates* process. This process runs parallel to the main DDPQ process. Once every window length, the process updates the estimate for the arrival rates of each priority using the exponential smoothing model. The new estimate for a priority is a function of the old estimate and the transmission rate for the priority as seen in the last window. The process also checks if the station has just transitioned into the synchronizing state. If so, it resets the timer. If the station is in *synchronizing* or *new_sync* state, it transitions into normal state after a window period elapses.

System Variable	Initialized Value	Comment
QUIT	0	Variable which indicates whether the frame should be discarded
TOTAL_NUMBER_JAMMED	0	Number of times the host's *Advertised Profile* has been jammed consecutively
STATIONS_JAMMED	0	Number of distinct stations whose *Advertised Profile* has been jammed by the host
BACKOFF_LEVEL	0	Backoff Level
MAX_BACKOFF_LEVEL	0	Maximum Backoff Level
ARRIVAL_RATE[8]	0	Estimate for the arrival rates for each priority
BACKLOGGED_NUM[8]	0	Estimate for the backlogged stations at each priority
NOT_IN_SYNC	0	Variable which indicates if the host's *Local Profile* is not the same as the *Advertised Profile*
FRAMES_TRANSMITTED[8]	0	Number of frames transmitted at each priority in the last window
BLOCKED	FALSE	Variable which indicates if the host should forgo transmission in the next contention cycle
TEMP_PRIORITY	PRIORITY_VAL	Priority dynamically assigned to the station before each contention cycle
TIME_WAITING[8]	0	Time at each priority since its last non-CRP transmission
LAST_TRANSMITTED[8]	0	Number of frames transmitted at each priority since its last non-CRP transmission
LAST_CONT_TIME[8]	Current Time	Time at each priority when its last non-CRP transmission occurred
LAST_TRANSMISSION_CONTESTED	0	Variable which indicates if the host's last *Advertised Profile* was jammed
LOCAL PROFILE[16]	0	*Local Profile* Value
ADVERTISED PROFILE[16]	0	*Advertised Profile* Value
GLOBAL PROFILE[16]	0	*Global Profile* Value
PROFILE[16]	0	Profile Value used to determine contention
WINDOW	10 ms	Length of the window period
SLOT_ALLOCATED[8]	1	Number of contention slots allocated to each priority

Table 5.1: Initialization Parameters for DDPQ

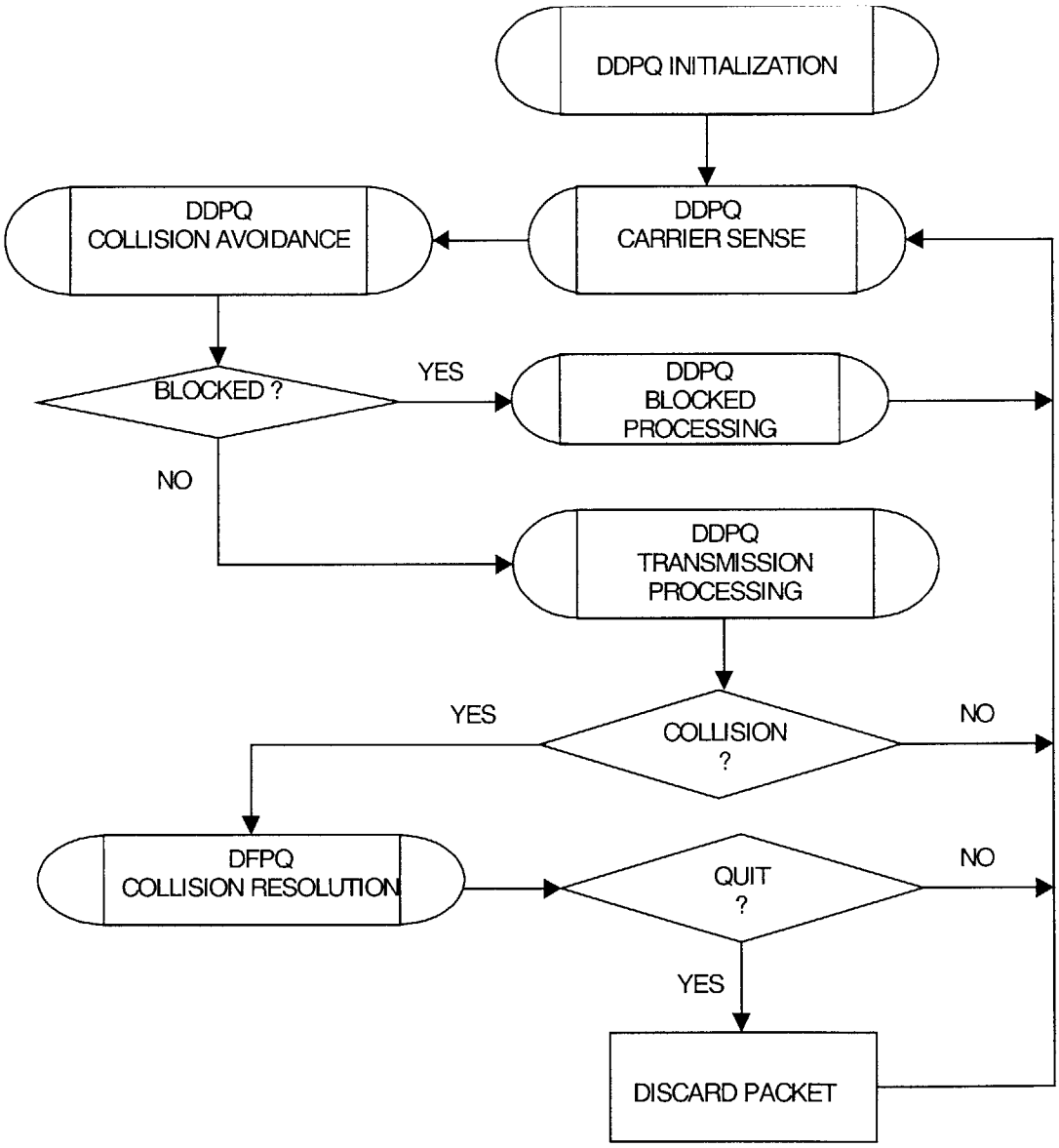


Figure 5.3: DDPQ MAC Protocol

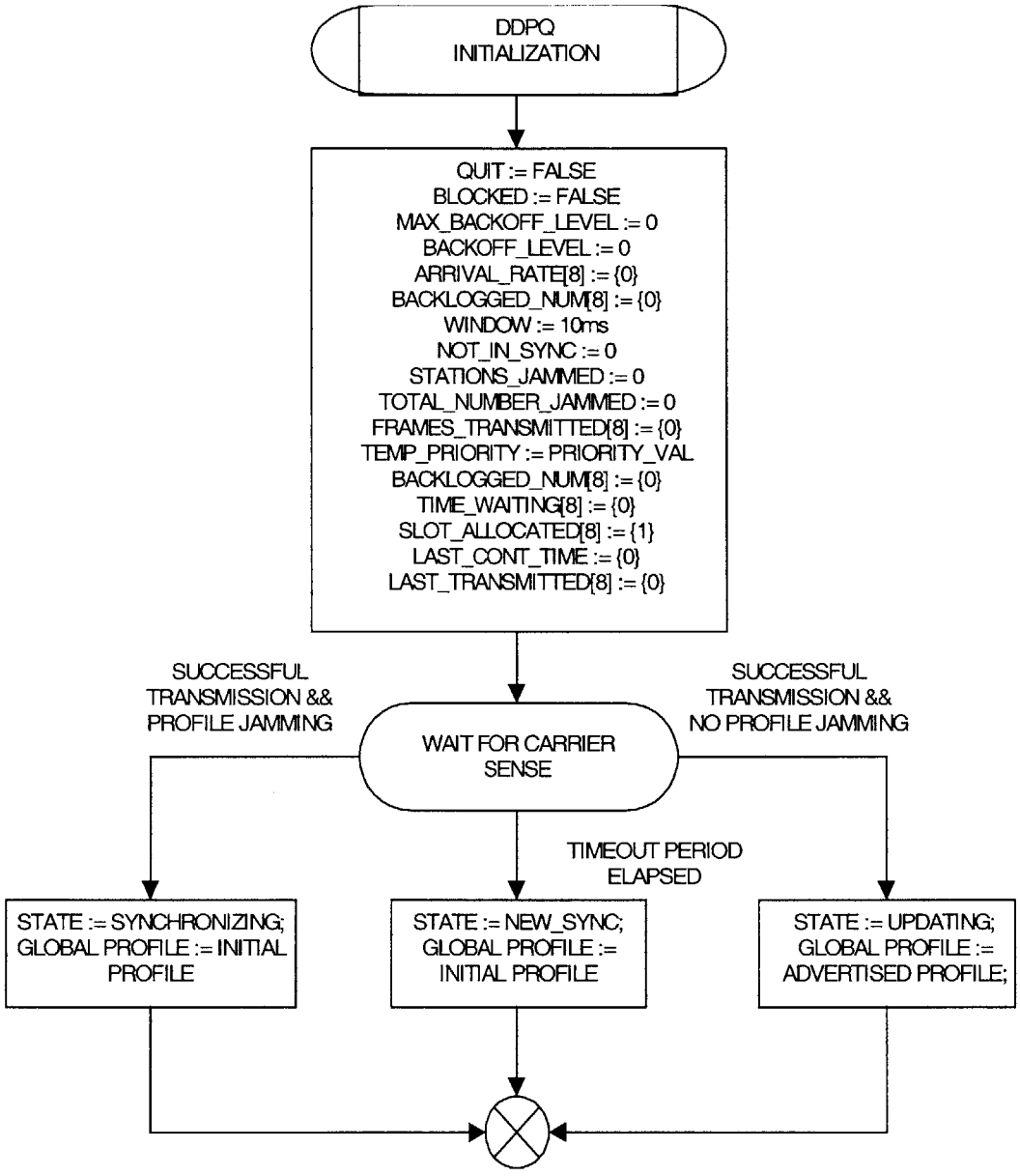


Figure 5.4: DDPQ initialization process

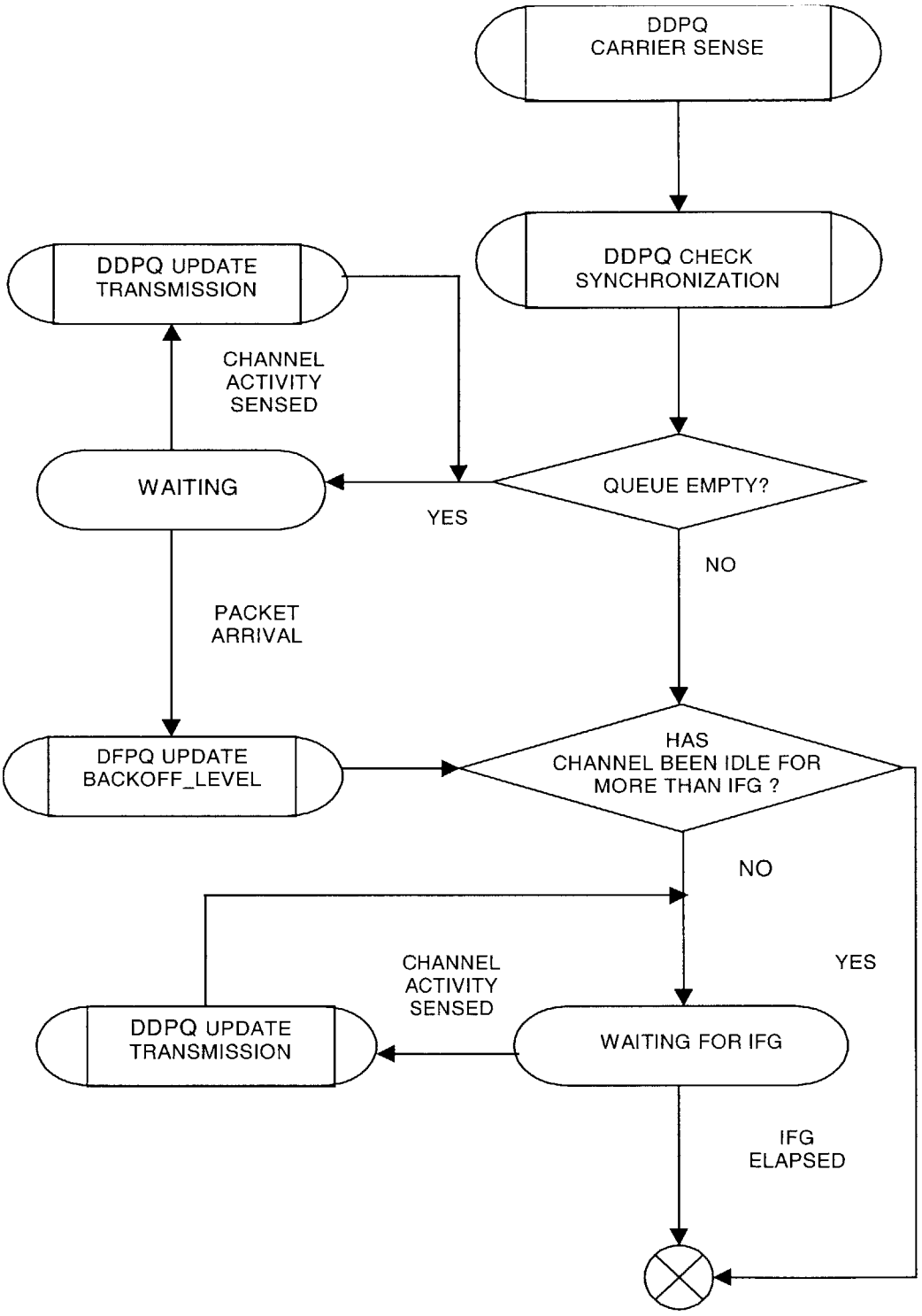


Figure 5.5: DDPQ carrier sense process

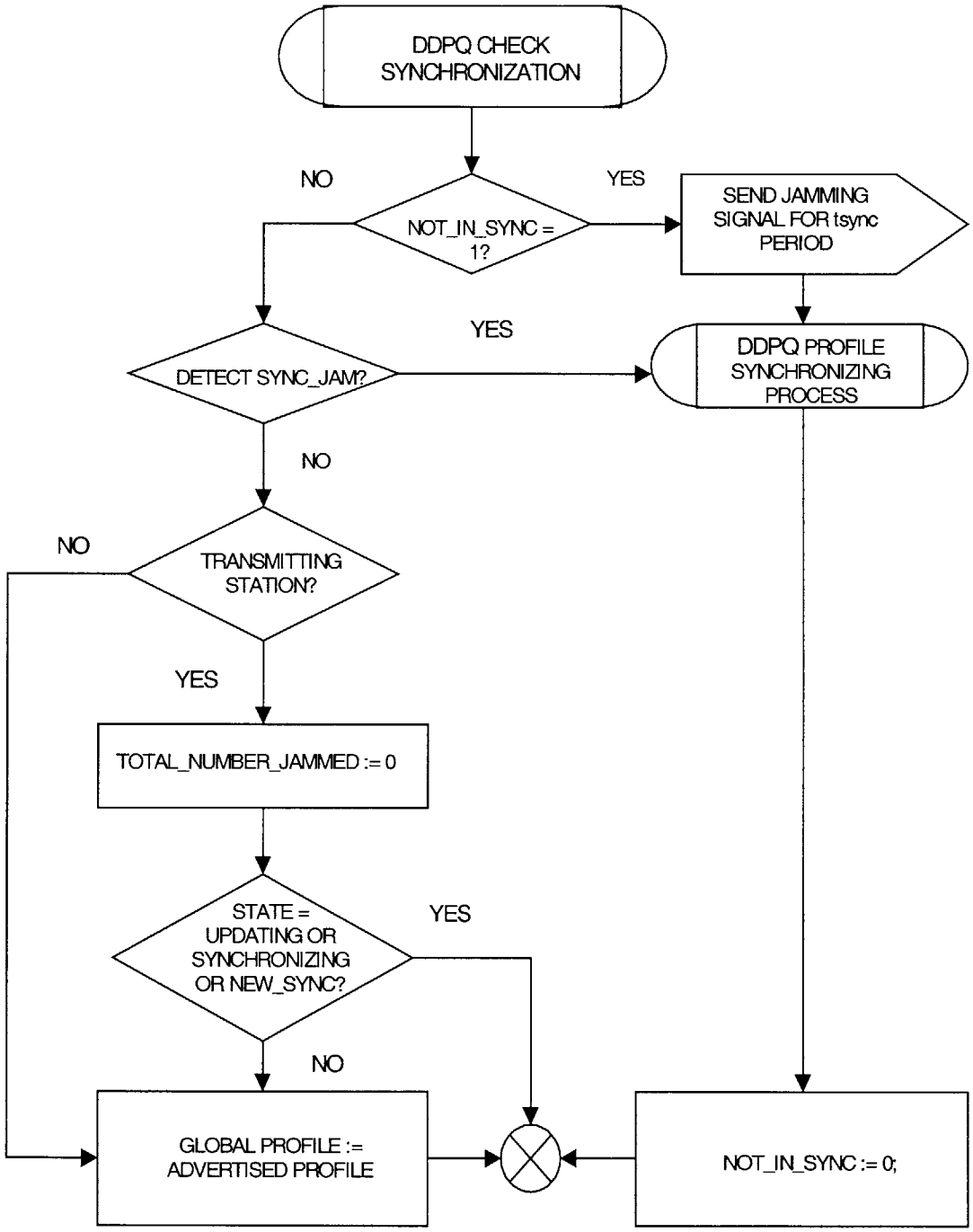


Figure 5.6: DDPQ check synchronization process

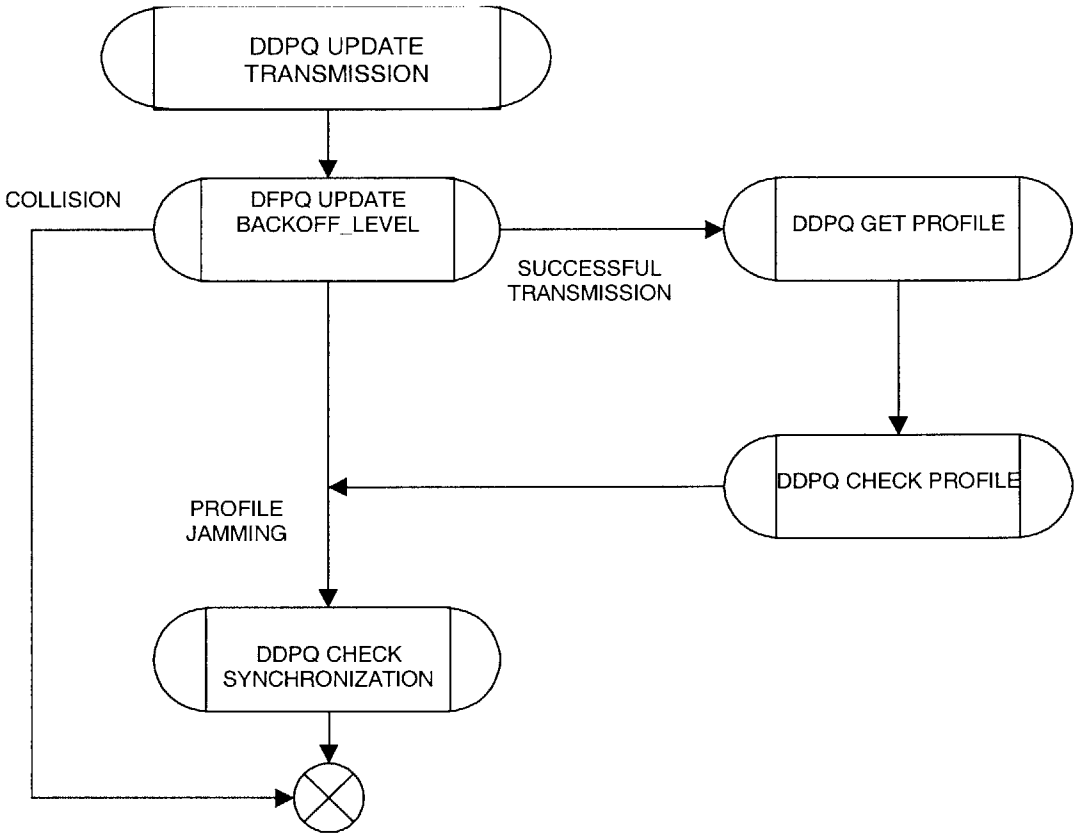


Figure 5.7: DDPQ update transmission process

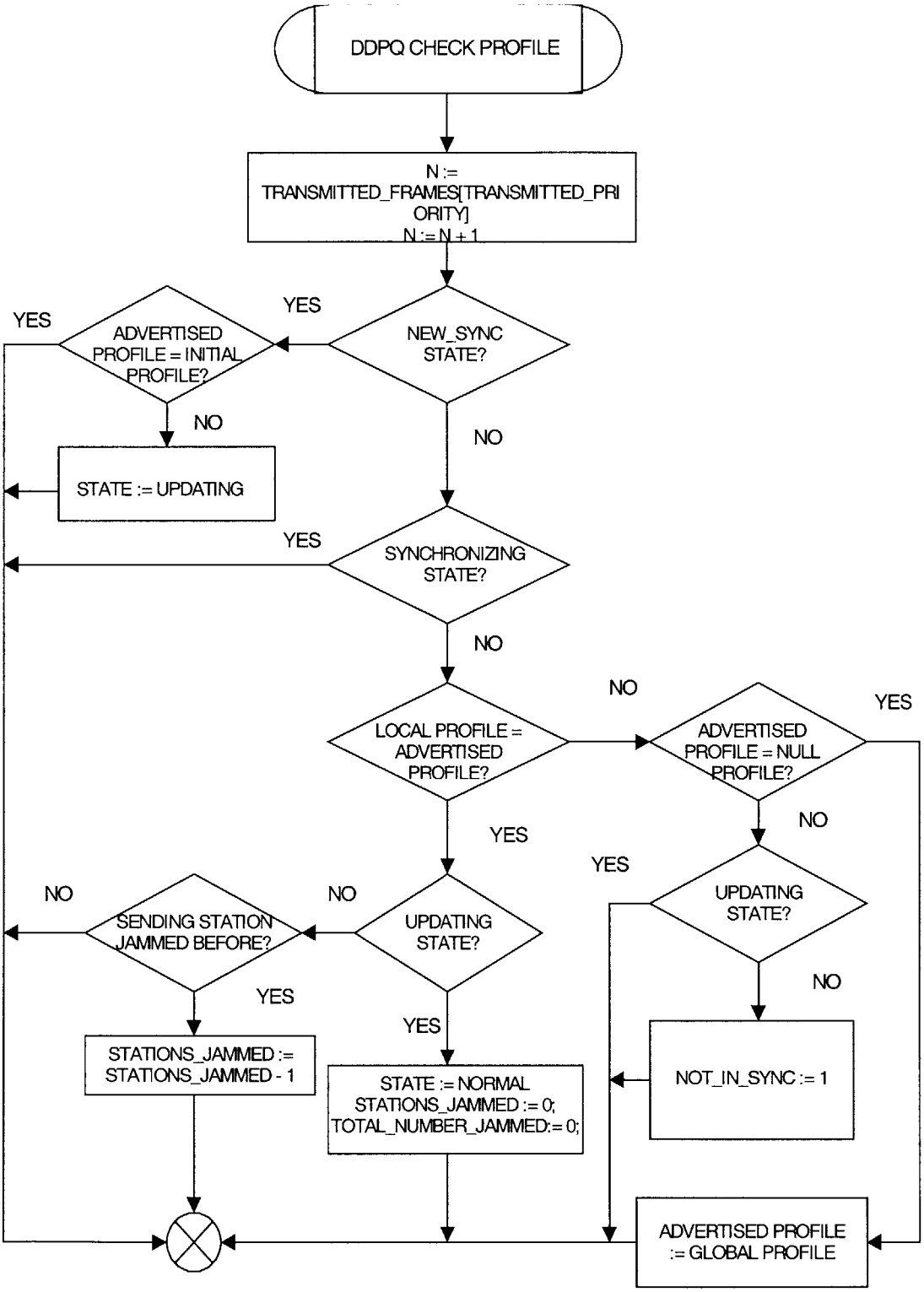


Figure 5.8: DDPQ check profile process

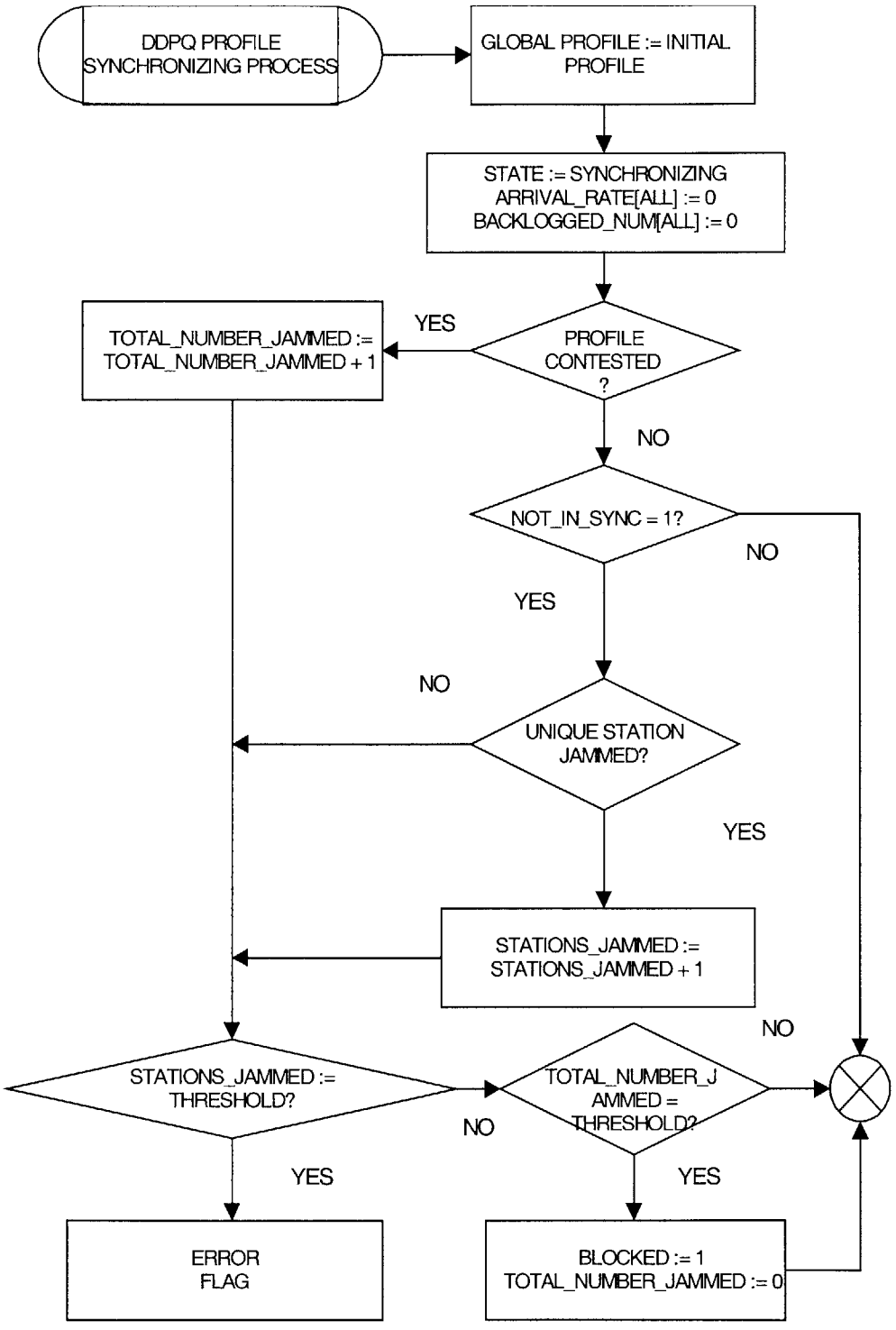


Figure 5.9: DDPQ profile synchronizing process

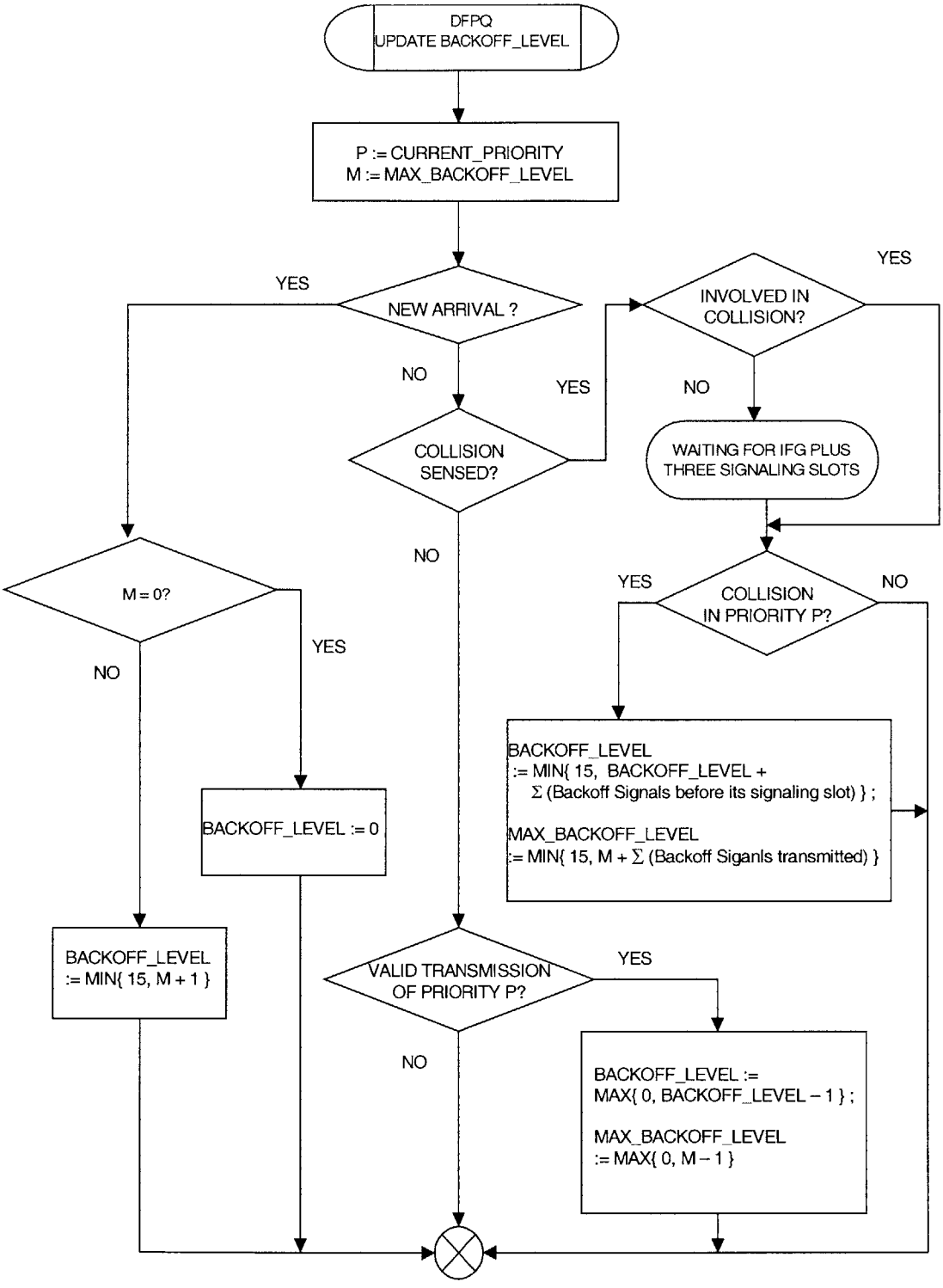


Figure 5.10: DFPQ update backoff level process

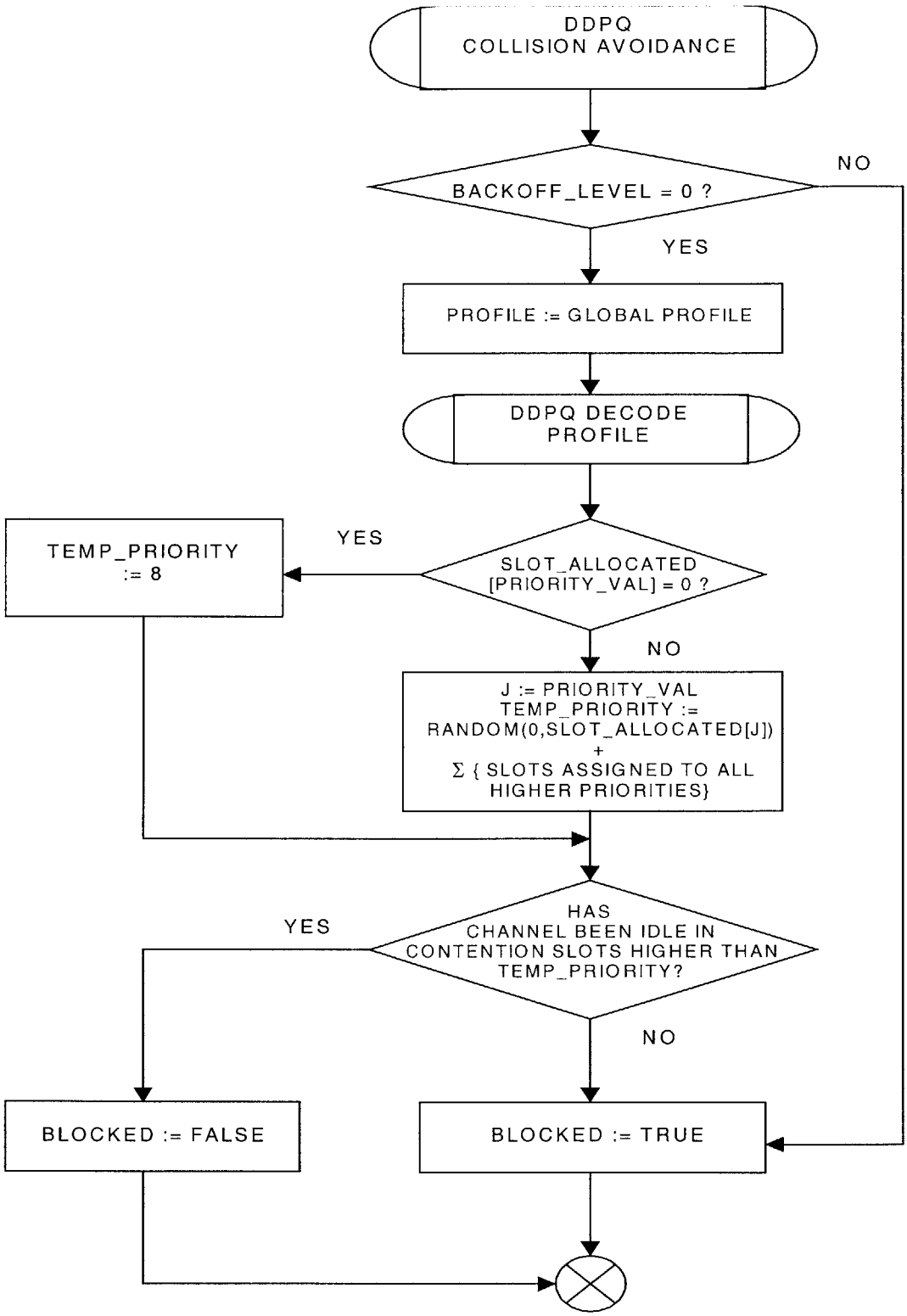


Figure 5.11: DDPQ collision avoidance process

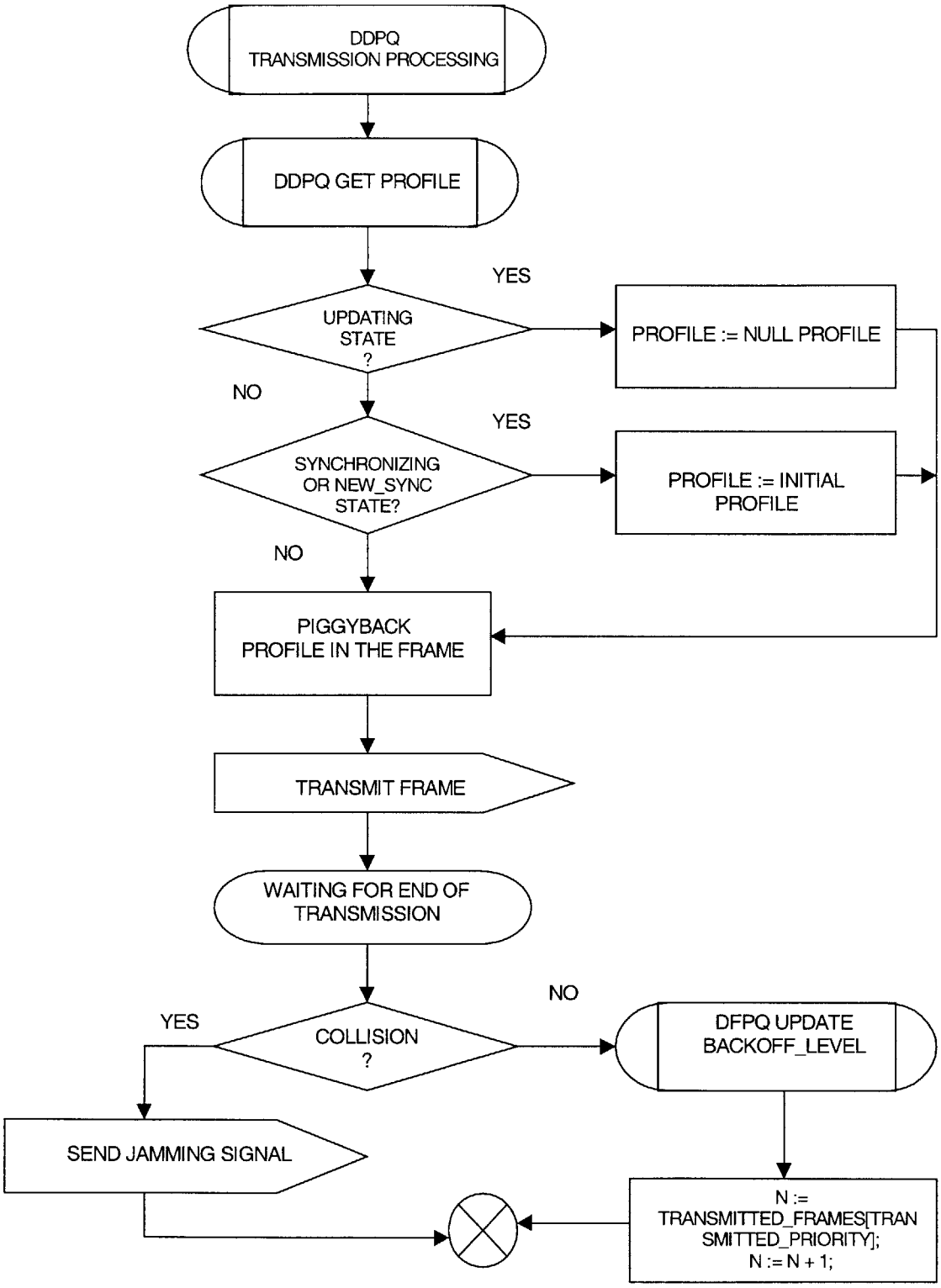


Figure 5.12: DDPQ transmission processing process

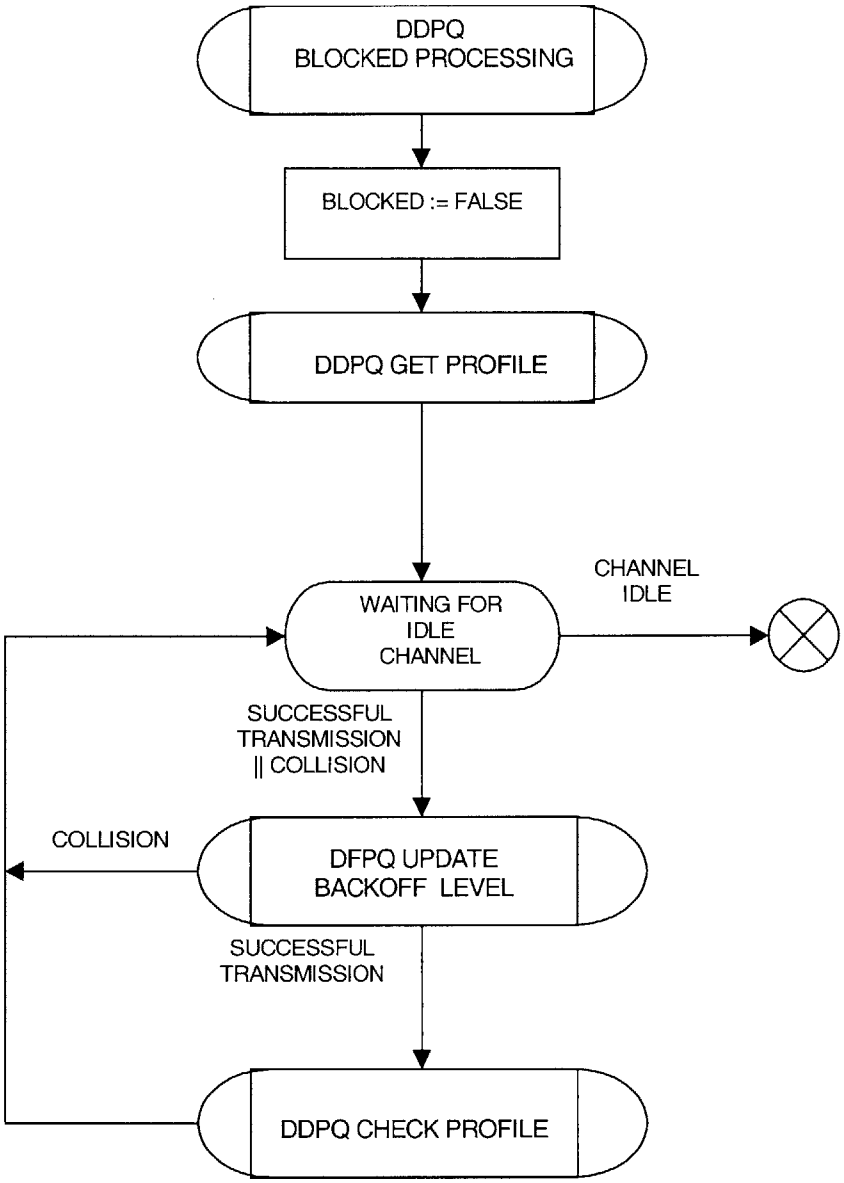


Figure 5.13: DDPQ blocked processing process

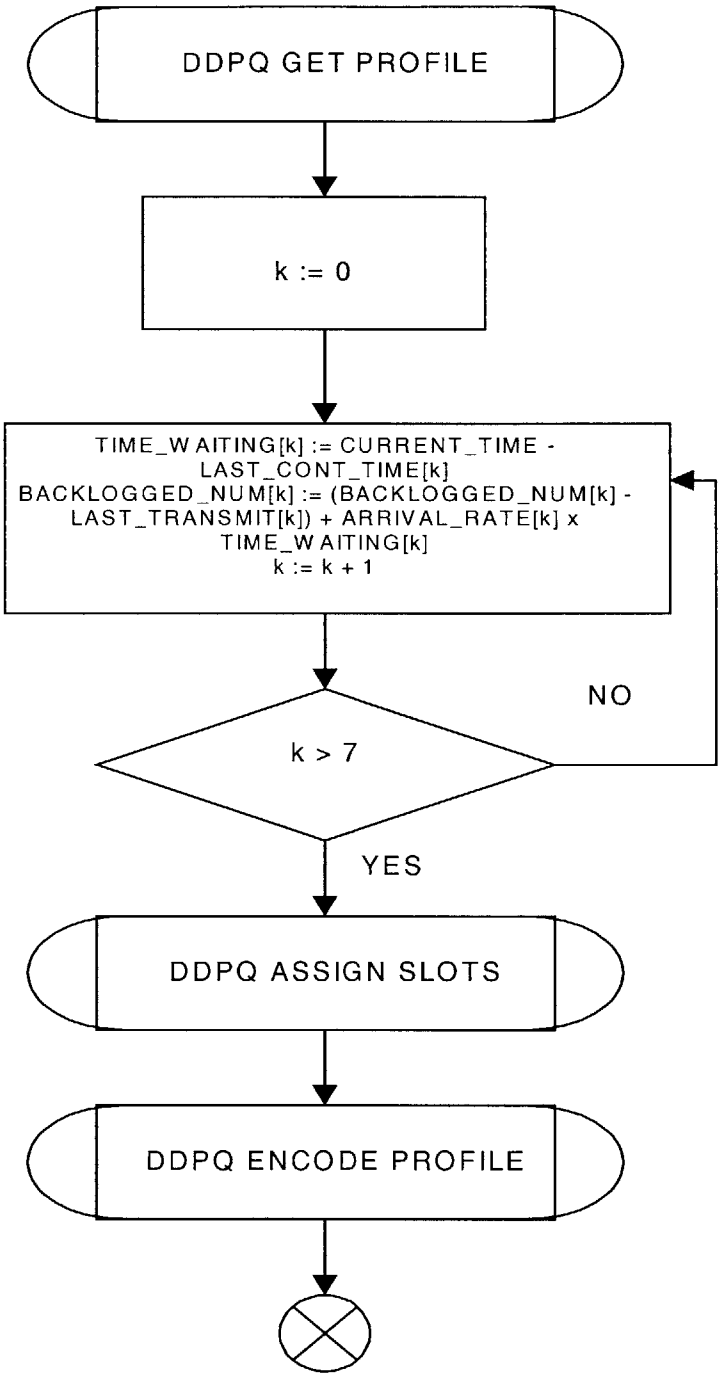


Figure 5.14: DDPQ get profile process

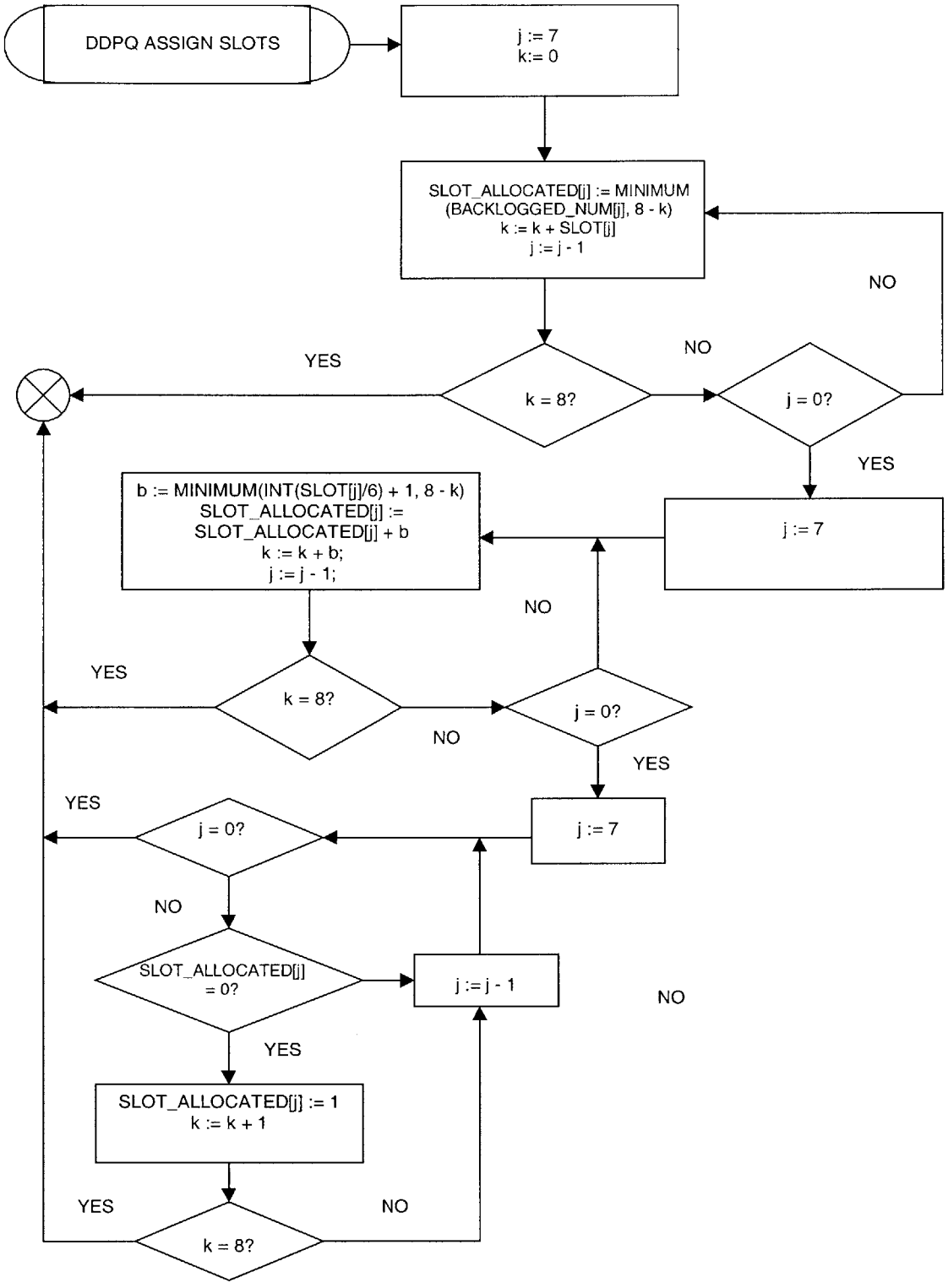


Figure 5.15: DDPQ assign slots process

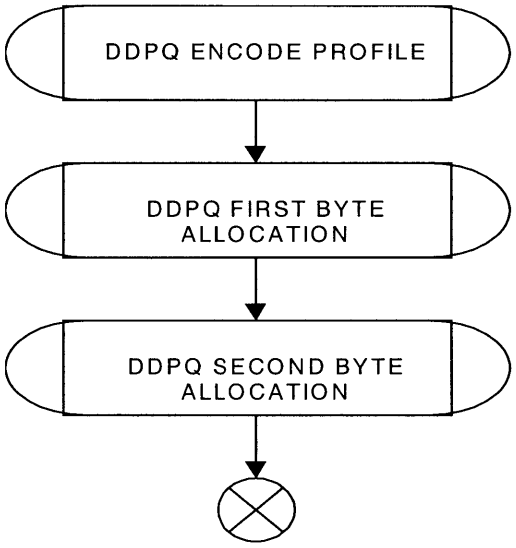


Figure 5.16: DDPQ encode profile process

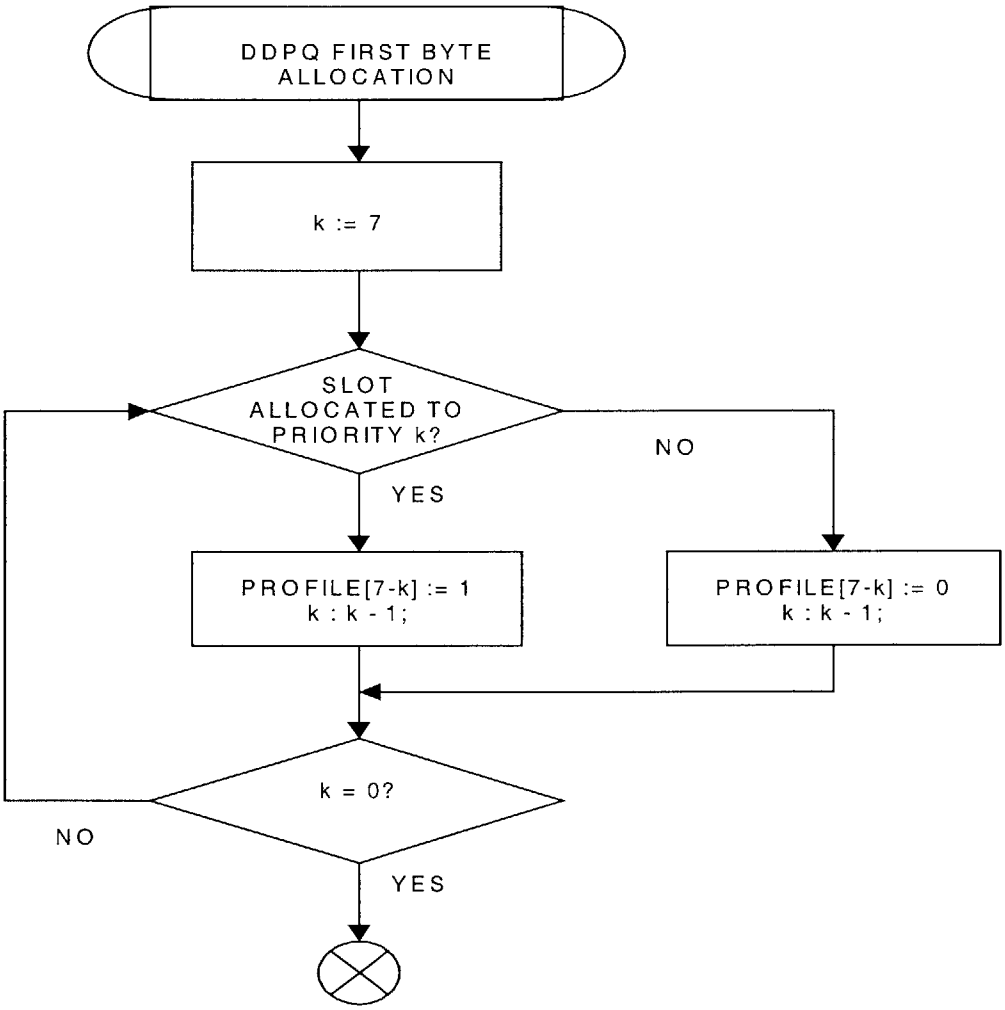


Figure 5.17: DDPQ first byte allocation process

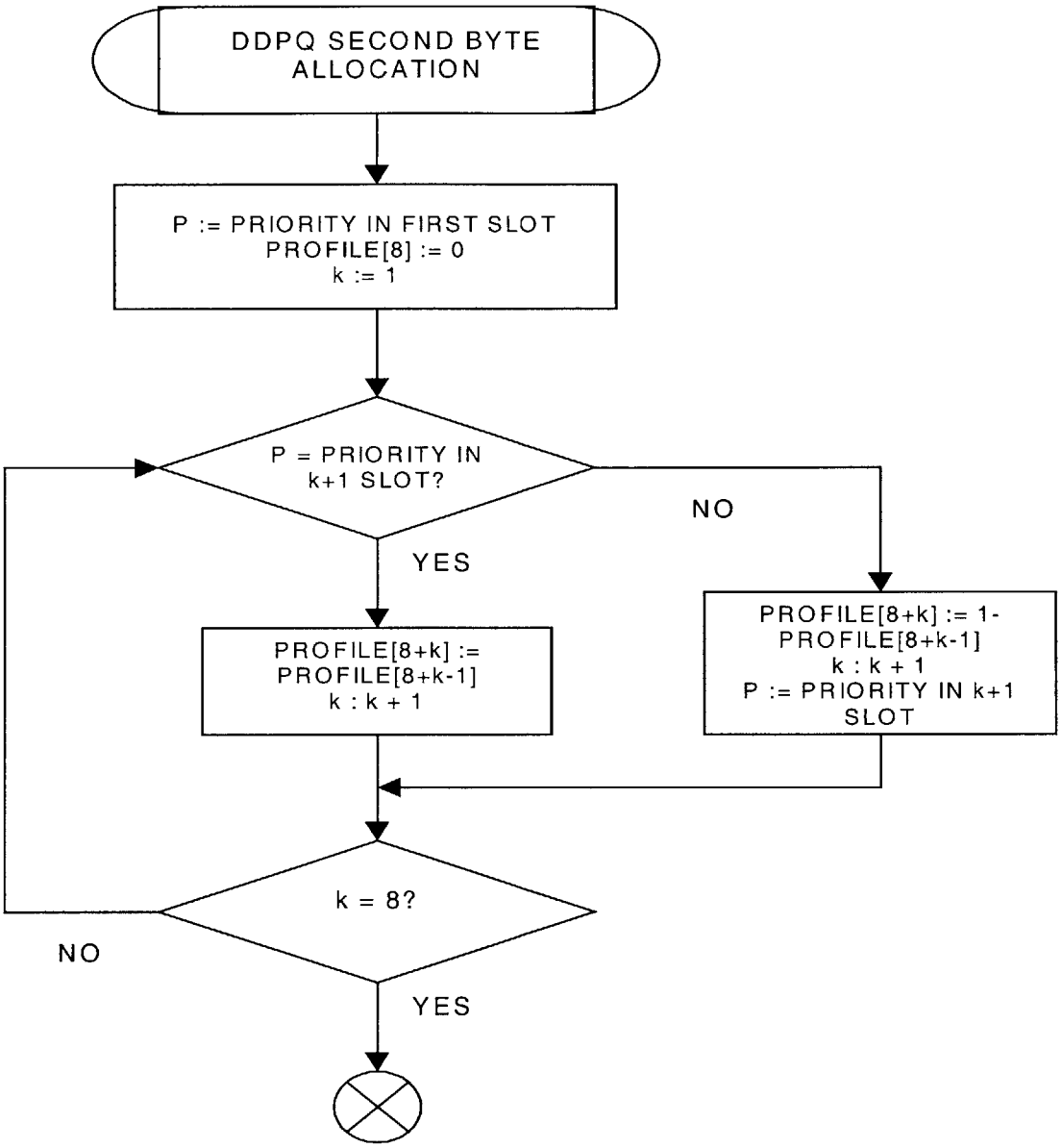


Figure 5.18: DDPQ second byte allocation process

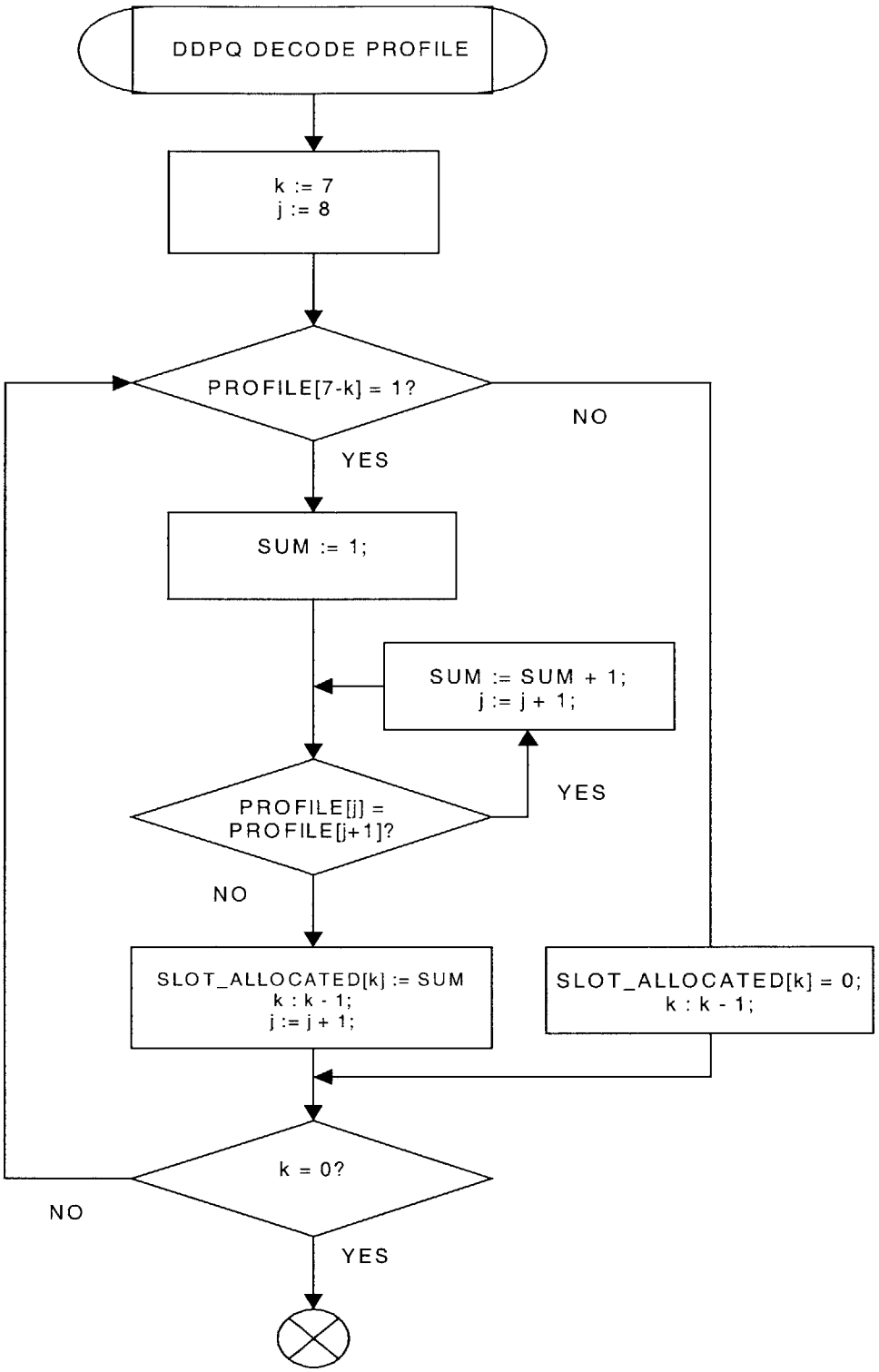


Figure 5.19: DDPQ decode profile process

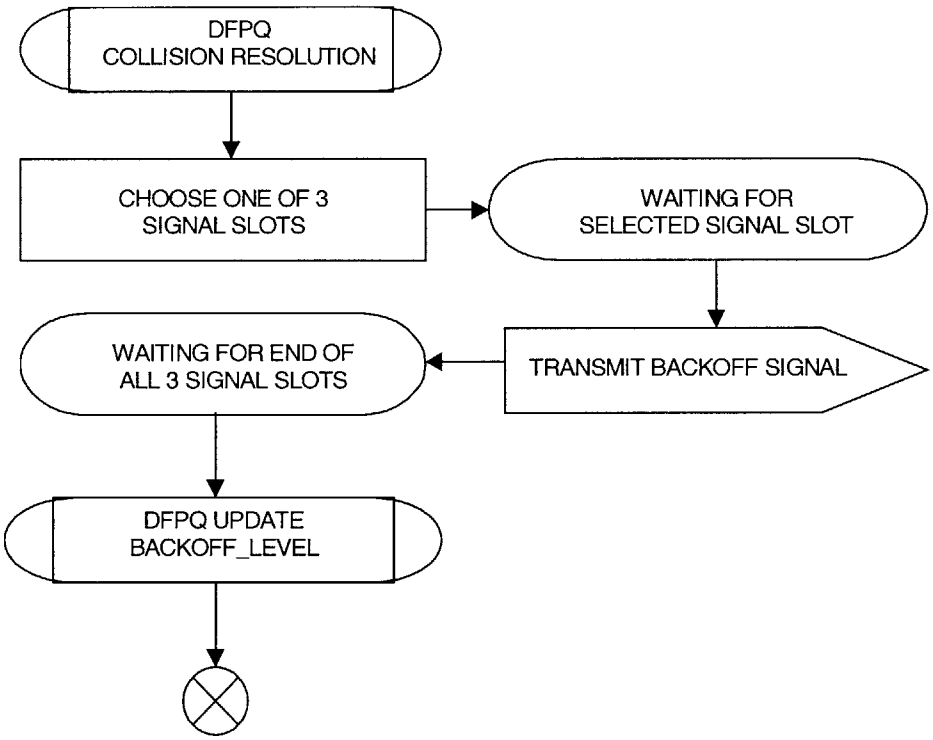


Figure 5.20: DFPQ collision resolution process

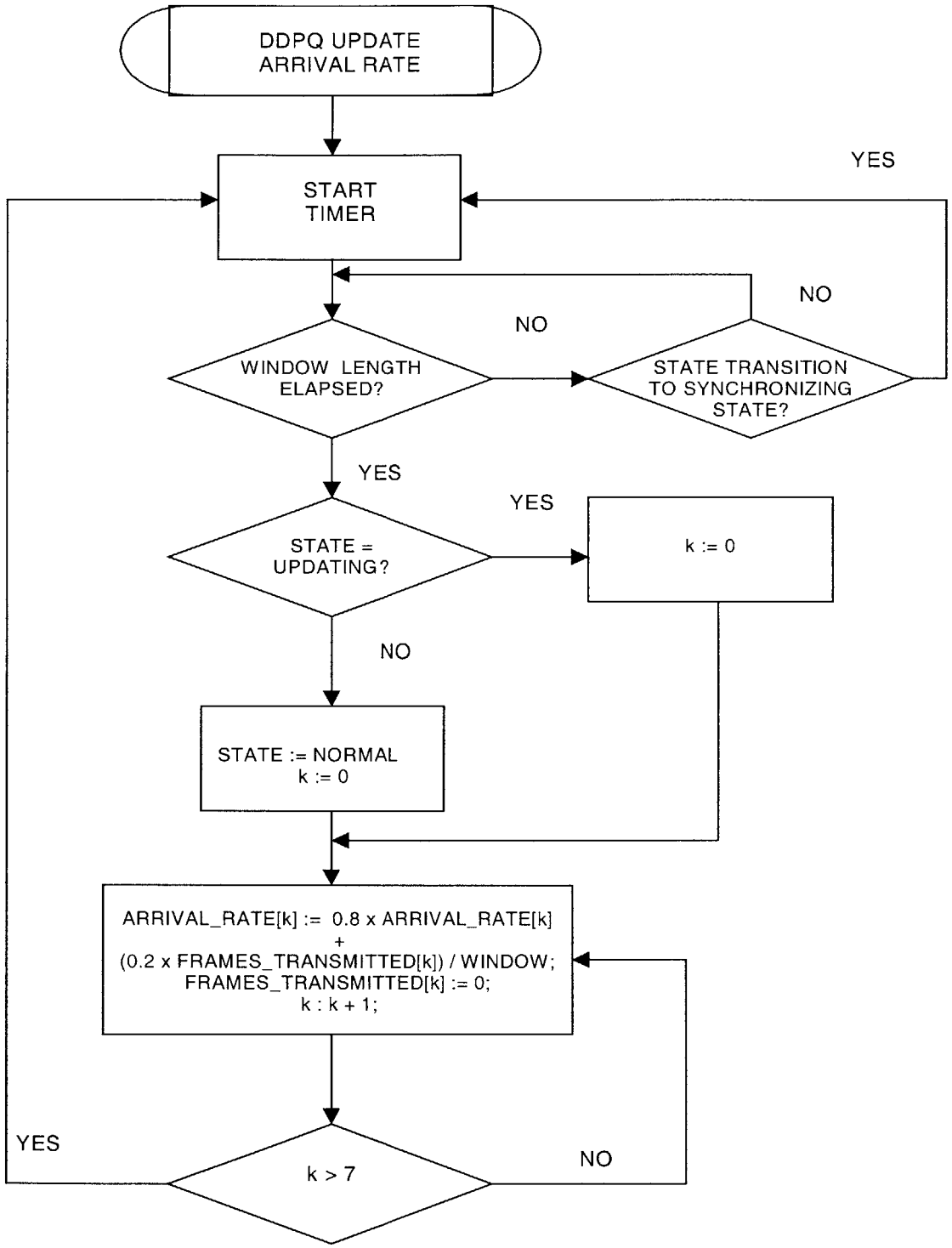


Figure 5.21: DDPQ update arrival rate process

5.4. Method of Load Estimation

We use a simple exponential smoothing model to estimate the arrival rate at each priority. Exponential smoothing model is a popular method for estimating a time series in which the past data is discounted in a gradual fashion. It is assumed that frame arrival for each priority is a Poisson process with a given rate. This rate is "locally stationary" i.e. it changes very slowly. Given this assumption, the estimate of the arrival rate for a priority is taken as a weighted sum of the previous estimate and the load in the trailing window.

On start, all stations proceed in the same way as DFPQ MAC. The number of successful transmissions of each priority class is monitored for a fixed window size and this information is used to estimate the arrival rate of the priority for the window period. Each station keeps an estimate for the arrival rate for every priority class.

Specifically,

$$g_p(t) = w \cdot g_p(t-1) + (1-w) \cdot r_p(t)$$

$$r_p(t) = \frac{n}{L}$$

where,

$g_p(t)$ is the estimated arrival rate for priority p,

w is the weight assigned to the previous estimate,

$r_p(t)$ is the arrival rate in the last window,

n is the number of successful transmissions in the last window, and

L is the window length

Then,

$$\text{Offered\_lad}(p) = (N_p - k_p) + g_p(t) \cdot T_p$$

where,

N_p is the number of estimated backlogged stations for priority p in the last contention cycle,

k_p is the number of successful transmission for the priority in the last contention cycle, and

T_p is the time elapsed from the last contention cycle to the present

When no transmission is perceived in the allocated slots for a priority, its estimated backlog number for the previous cycle (N_p) is initialized to 0. If a collision resolution process (CRP) is ongoing, the estimated backlog number for the priority is set at 1. This is because only those stations that have a Backoff Level value of 0 (as assigned by the tree-splitting algorithm) can transmit during the CRP. Frames that arrive during the CRP contend only after the process is over. Hence, in the contention cycle following the CRP we set T_p to be the length of the collision resolution cycle (contention cycle at which collision occurred to the present). Similarly, we obtain N_p from the contention cycle before the collision and k_p is set to the number of successful transmissions of the priority in the CRP.

A new station or a resetting station waits at least one window period before it uses its arrival rate estimates to obtain the estimate for the number of backlogged stations. The value of w, the weight assigned to the previous estimate, in the equation above is obtained through simulations. We started with w = 0.5 and perturbed it both ways in steps of 0.1 to see the change in performance. It was found that w = 0.8 gives the best performance in terms of delay and the number of intra-priority collisions. L, the length of the window, is chosen such that it allows the resetting stations to synchronize quickly to the system. Once again we started from a value of 15ms and again varied it both ways in steps of 1ms. It was found that L = 10ms gave the best performance, again in terms of delay and the number of intra-priority collisions. Given the load estimation scheme, the next step is to allocate slots among the priorities such that intra-priority collisions are minimized.

5.5. Optimal Slot Allocation

We define an instantaneous contention throughput (ICT), which is the ratio of the probability of a success in an initial contention period to the expected length of the initial contention period. ICT gives a measure of the slot expense for a likely success. Maximizing this ratio is equivalent to minimizing the probability of intra-priority

$$ICT = \frac{1 \cdot P_s(n, b) + 0 \cdot P_c(n, b)}{P_s(n, b) \cdot s^* + P_c(n, b) \cdot c^*} = \frac{P_s(n, b)}{P_s(n, b) \cdot s^* + P_c(n, b) \cdot c^*}$$

collision.

Where,

n is the number of slots allocated,

b is the number of backlogged stations,

$P_s(n,b)$ is the probability of success given n slot allocation for a backlog number b,

$P_c(n,b)$ is the probability of collision given n slot allocation for a backlog number b,

s* is the expected number of slots for a successful transmission to occur given that there is a success, and

c* is the expected number of slots for a collision to occur given that there is a collision in the contention cycle.

The numerator in the expression above gives the probability of success in the contention cycle whereas the denominator gives the total number of slots that are likely be spent in the cycle. The expression, therefore, is the throughput of the system for the contention cycle.

We now define individual terms in the equations below. $P_c(n,b)$ is the complement of $P_s(n,b)$; hence the first equation. $P_s(n,b)$ is equal to 0, if the number of slots allocated,n, is 0 or 1. In the expressions below, we only consider the cases where b is greater than 1 as the cases where b is equal to 0 or 1 are trivial. If n is greater than 1, then $P_s(n,b)$ is equal to the sum of the probabilities of first success happening in the y^{th} slot where y varies from 1 to n-1. For a first success to occur in the y^{th} slot all preceding

slots must be idle. Hence, the success cannot occur in last n^{th} slot because, since all preceding slots are idle, the slot must hold all b packets i.e. there will be a collision. The probabilities of a success occurring in the slots are mutually exclusive; therefore, the aggregate probability of success is a sum of the individual probabilities. The probability of first success occurring in the y^{th} slot, P(y), is given by the probability for the case where one frame chooses the y^{th} slot and all other frames choose the slots after the y^{th} slot. Note that in this manifestation, all slots preceding the y^{th} slot will be idle.

s* is the expected length of the slots where a success occurs, given that there is a success in the system. We obtain an expression for s* by summing the product of lengths y, where y varies from 1 to n-1, with their respective probability of success P(y). As noted above, for a first success to occur in the y^{th} slot all slots preceding y must be idle. c* is expected length of the slots before a collision occurs, given that a system experiences a collision. We obtain this expression by summing the product of lengths y, where y varies from 1 to n, with their respective probability of collision C(y). C(y) gives the probability that all slots preceding y are idle and at least two of the backlogged stations will choose the y^{th} slot. Note that collision could occur in any slot as indicated by the range of y (1 to n).

$$P_s(n, b) = 1 - P_c(n, b)$$

$$P_s(n, b) = 0 \quad \text{if } n = 0 \text{ or } 1$$

$$= \sum_{y=1}^{n-1} P(y) \quad \text{otherwise}$$

$$s^* = \sum_{y=1}^{n-1} y.P(y)$$

$$c^* = \sum_{y=1}^{n} y \cdot C(y)$$

$$P(y) = \binom{b}{1} \cdot \frac{1}{n} \cdot \left(\frac{n-y}{n}\right)^{b-1}$$

$$C(y) = \binom{b}{2} \cdot \left(\frac{1}{n}\right)^2 \cdot \left(\frac{n-(y-1)}{n}\right)^{b-2}$$

We have determined, via numerical computation, that for a given number of backlogged stations associated with a given priority, this ratio achieves a maximum value with a unique allocation (see Figure 5.21), and that this allocation is related to the given number of backlogged stations almost linearly for a practical range of this number (see Figure 5.22). This enables an optimization scheme for slot allocations among the priorities.

It is unlikely that in a home environment, even with increasingly networked appliances, there will be any more than 15 stations transmitting in the same priority level. With that assumption, as can be seen from Figure 5.22, the relationship between the backlog number and the slot allocation at which ICT is maximized is almost linear. An approximation for this relationship is given by the following equation

$$m = b + \lfloor b/6 \rfloor + 1 \qquad \text{(a)}$$

where m is the number of slots at which maximum ICT occurs, b is the estimated number of backlogged stations. This relationship holds for all cases where b is less than 17.

The slot allocation at which ICT is maximized is only slightly higher than the backlog number. This is similar to the slotted ALOHA where the throughput is maximized when the slot allocation is equal to the number of backlogged stations. Recall that every contention cycle is of length eight priority slots divided in some way among the priorities and is terminated by a success or a collision. Our calculation is for finite station assumption where the number of stations contending at any contention slot is given by the backlog number b. This is in contrast to the slotted ALOHA system, which assumes an infinite population of users, and where the throughput is a result of observation on many slots [Rob75]. The system throughput when a finite number of users are transmitting is higher than the case where an infinite population of users is assumed [TaK85]. This is evident from Figure 5.21 where the maximum ICT achieved stabilizes around 0.6 which is higher than that of the slotted ALOHA where the maximum value is upper bounded by 1/e.

We have formulated an optimization problem, which maximizes the aggregate ICT while favoring higher priority packets over lower priority ones. The objective is to allocate slots in such a way that we maximize the ICT for each priority at every contention period. One method of optimization would be to use the Greedy method [CLR90].

The Greedy algorithm is a local optimization method. It assumes that the path to the best global optimum is a series of locally optimal steps. The principal advantage of Greedy algorithms is that they are usually straightforward, easy to understand and computationally very efficient. Because we know the ICT value of assigning one extra slot for any priority, Greedy method will find the most optimal slot allocation.

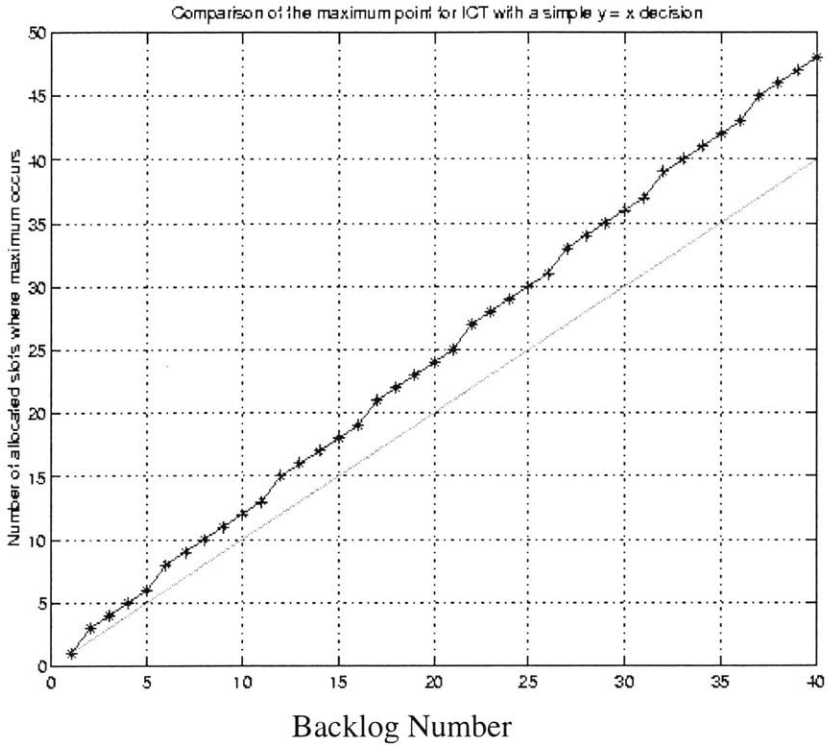


Figure 5.22: Maximum ICT point vs. Backlog Number

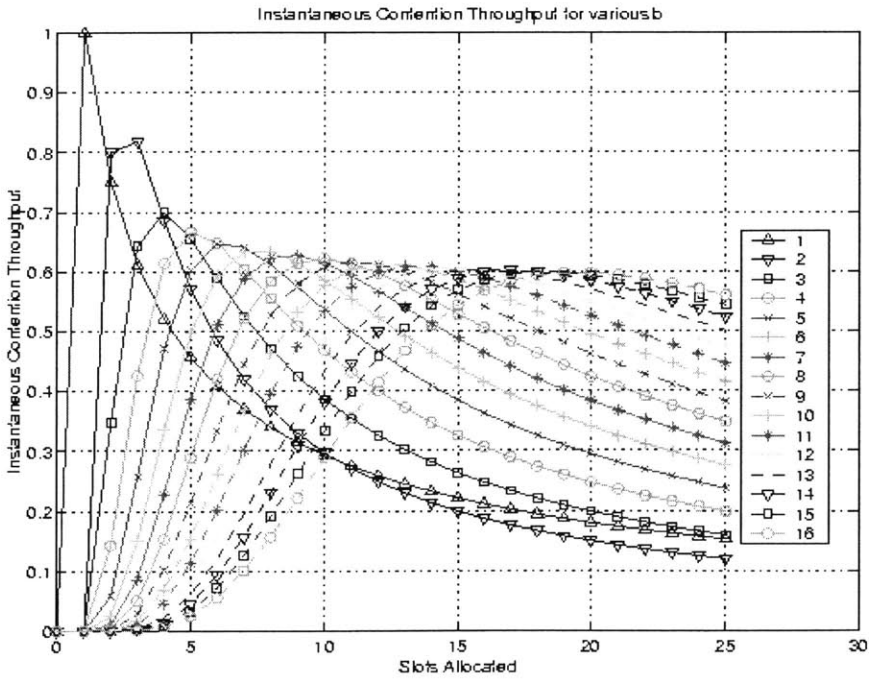


Figure 5.23: ICT vs. Number of Slots Allocated

In a Greedy approach, we would start by allocating each priority m slots, where m is equal to the number where its ICT is maximized. This number depends on the number of backlogged stations and is given by equation (a). If the sum of all the slots so allocated is greater than 8, we de-allocate slots one at a time such that the aggregate objective function (sum of all ICTs) is least decreased. While de-allocating the slots, we want to favor higher priorities i.e. take slots away from lower priorities. This is because higher priorities get access to the channel before lower priorities. And because contention cycle is terminated by a success or a collision, the probability of a success in the cycle is greatly influenced by higher priorities; only in the event of an idle in all higher priorities (due to incorrect estimates) does a lower priority frame get access to the channel. Higher priorities could be favored by assigning them greater weights relative to the lower priorities. In such a case, if a higher priority has the same backlog number as a lower priority, the aggregate objective function would be least decreased when the next slot is de-allocated from the lower priority. Contrast this to the case where the next slot is de-allocated from the higher priority. Since both priorities have the same backlog number and they are initially assigned the same number of slots (where their ICT is maximized), the probability of a success after de-allocation will be lower in the higher priority. And since higher priority gets first access to the channel, the overall probability of success in the contention cycle is less than would be the case if the slots were de-allocated from the lower priority.

Such weight assignment mechanism is also consistent with the fact that higher priority frames have a greater QoS need than lower priority frames. This optimization technique, however, has a huge computational overhead because obtaining ICT for each priority is computationally intensive. A simple method is presented which approximates the greedy approach and which also incorporates a bias for the higher priorities.

From Figure 5.21 it is apparent that equating the number of slots allocated to the backlog number is not the most optimal method since the greatest ICT for a priority occurs slightly above the $x = b$ point, where x is the number of slots allocated to the priority and b is the backlog number. However, it can also be seen that the value of ICT

at this point is not very far displaced from the maximum. Using this knowledge, we develop a simple optimization method as described below (see Figure 5.15):

- Proceed from the highest priority to the lowest. Assign slots to each priority equal to their backlog number estimate. If the total number of slots allocated equals 8, slot allocation is completed.
- If there are more slots remaining to be assigned, once again proceed from highest priority to the lowest. Assign slots to each priority such that the allocation equals the number at which maximum ICT occurs. This number, denoted x, is given by $x = m = b + \lfloor b/6 \rfloor + 1$, where b is the backlog number estimate for the priority. If b is equal to 1 or 0, the number at which maximum ICT occurs is the same as b. Slot allocation is stopped when total number of slots allocated equals 8 or when the last priority is reached whichever occurs first.
- If there are still extra slots remaining to be assigned, the remaining slots are allocated to idle priorities, one slot per priority, going from highest priority to the lowest. Slot allocation is stopped as soon as the total number of slots allocated is equal to 8.

The motivation for allocating slots from highest priority to the lowest lies in the fact that higher priority frames get first access to channel by virtue of their higher priority. Note that any successful transmission or collision ends the contention cycle. Hence in order to ensure higher priorities greater probability of successful transmission, we allocate slots in a descending order of priority. Slots are first allocated equal to the backlog number estimate. The reason being even though not maximum, the value of ICT at x = b, where x is the number of slots allocated and b is the backlog number estimate, is very close to the maximum and hence presents an easy starting point for optimization.

6. Simulation

An event-based simulation was written in OPNET[2] to model CSMA/CD, DFPQ and DDPQ MAC protocols. The simulation parameters are defined in section 6.1. In our simulation runs, we measure the system throughput, mean access delay and access jitter for a given offered load. Access delay is the delay from when a frame is ready to transmit till the time it is transmitted i.e. it gives a measure of the service time for the frame in the system. Note that this delay is different from queue delay, which is measured from the time the frame is introduced into the queue till the time it is transmitted. Because a frame has to wait in the queue until all the frames before it is transmitted, its queue delay can grow unbounded when the offered load exceeds the queue service rate. This wait time is not incorporated in the access delay.

We have chosen three scenarios for our simulation studies. In the first two scenarios we limit contention to a single priority level. We choose a priority level of 7 so that stations will not have to wait out any higher priority slots in DFPQ and DDPQ protocols. This allows for a fair comparison with CSMA/CD, which has a first-come-first-serve scheduling mechanism. Limiting contention to a single priority level allows us to observe the effectiveness of DDPQ protocol in minimizing intra-priority collisions and how the performance compares against other two protocols while the system offered load or the number of stations varies. In one scenario, we keep the number of stations constant, but vary the offered load. In another scenario, we keep the offered load of each station constant but vary the number of stations. The offered load is the same for all stations. In the third scenario, we look at the performance of the protocols when different priorities are present. Specifically, three priorities are used to simulate three prevalent traffics in home networks--voice, video, and data. Voice frames have the highest priority of 7. Video frames have the next highest priority 6. And data frames have the next priority 6. This case is more complicated than the first two scenarios discussed because the delay associated with a frame at any priority is dependent not only on the number of collisions in its priority but also on the delay characteristics of all higher priorities. For

this case, we only look at the situation where the number of stations remains constant and the offered load is varied.

Each of the scenarios was run for the DDPQ, DFPQ and CSMA/CD MAC protocols. The test was run three times, each time under different initial seed value for the random number generator. The random number generator is used by the simulation program to generate packet arrival times, to allow collided stations to choose one of the three signal slots and in the case of DDPQ, allow stations to choose one of the allocated slots. The random number generator draws from a single random number sequence initialized with the value of the seed environment attribute. Choosing different seed values essentially changes the random number sequence i.e. the new simulation run is different from previous runs. By averaging over three simulation runs, we attempt to make our simulation results more accurate. It was stopped when either the mean delay achieved a stable value or the system showed that the delay was unbounded. This time frame usually varied between 30 sec to 50 sec.

Figure 6.1 shows a snapshot of the simulated network derived from the OPNET simulator.

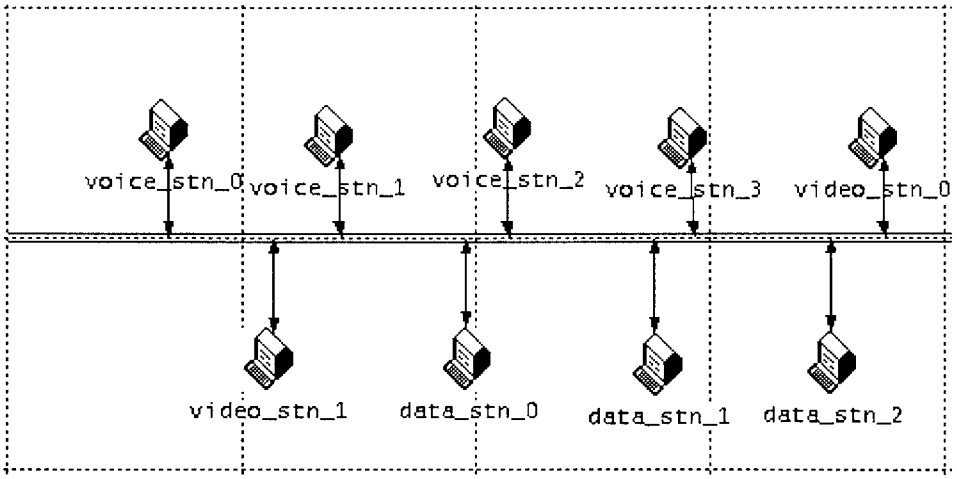


Figure 6.1: Simulated Network

[2] OPNET is a registered trademark of MIL3, Inc.

6.1. Simulation Parameters

Parameters for the traffic sources:

Traffic	Arrival Process	Arrival Rate
Data	Poisson Process	Variable
Voice	Constant	64 Kbps
Video	Constant	Constant

Table 6.1: Parameters for Traffic Sources

General channel parameters:

Parameter	Symbol	Nominal Value
Slot time	t_{slot}	512 bits
Interframe Gap	t_{ifg}	96 bits
Jam Signal	t_{jam}	32 bits
Overhead bits per frame	H	208 bits
Channel bit rate	r_c	10 Mbps
Path delay	τ	2.3 μs

Table 6.2: General Channel

Parameters unique to DFPQ and DDPQ:

Parameter	Symbol	Nominal Value
Priority Slot	t_p	19 μs
Signal Slot	t_{sig}	26 μs

Table 6.3: Parameters unique to DFPQ and DDPQ

Parameters unique to DDPQ:

Parameter	Symbol	Nominal Value
Profile	-	16 bits
Window	t_{win}	5 ms
Profile Synchronizing Jam	t_{sync}	25 bits

Table 6.4: Parameters unique to DDPQ

6.2. Simulation Results and Discussions

In the scenarios simulated, we see that DDPQ performs better than both DFPQ and CSMA/CD protocols. In general, compared to DFPQ the access delay and jitter are better in DDPQ and it also obtains a higher system throughput. Hence, DDPQ is more suited to real-time applications than DFPQ. Comparison with CSMA/CD is very interesting because although the protocol does not have QoS support, it obtains a better system throughput than the other two protocols. However, the performance of CSMA/CD is very poor for integrated traffic conditions and it also suffers from the capture effect at high offered loads, which makes it particularly unsuitable for real-time traffics.

An important observation from scenarios 1 and 2 is that while the delay for CSMA/CD is much better than that for DDPQ and DFPQ protocols, its jitter is significantly worse. The explanation, which is taken up in detail during the discussions of individual simulation scenario results, lies in the collision resolution characteristics of the protocols. Binary Exponential Backoff is partial to recently arrived frames while resolving collisions. This means that the mean delay for a frame in CSMA/CD would be less when compared to the ternary-tree splitting method but the worst case delay would be considerably inferior since some frames will have to wait unnecessarily long before getting access to the channel. And the delay statistics does not even incorporate the delays of the frames that suffer excessive (more than 16) collisions and are dropped. In the case of DDPQ and DFPQ, all frames contribute to the statistics as no frames are dropped because of excessive collisions. The worst case delay is what makes the CSMA/CD MAC protocol unsuitable for integrated traffic.

All of the plots presented in this section show performance measures as a function of the offered load or the number of stations in the system.

6.2.1. Scenario #1

We first show the performance of the algorithm when contention is limited to the same priority. The number of contending stations is kept constant. The mean arrival rate is uniformly varied to adjust the offered load.

In Figure 6.2, the mean throughput of the network is shown vs. offered load. Note that CSMA/CD has a much higher throughput compared to DFPQ and DDPQ. In fact, the throughput for CSMA/CD approximates the offered load and maximizes at about 7.1 Mbps. In contrast, the throughputs for DDPQ and DFPQ protocols maximize at 4 Mbps and 3.1 Mbps respectively.

Stations are more likely to collide in DDPQ and DFPQ protocol than in CSMA/CD. Since all stations are transmitting in the highest priority level (in this regard they observe a first-come-first serve scheduling mechanism like that of CSMA/CD), the greater number of collisions is not due to the stations transmitting in the same priority level. The reason, instead, lies in their collision resolution processes. Note that the Binary Exponential Backoff (BEB) algorithm schedules a collided frame to be transmitted in some future time, determined randomly. Although unfair (capture effect), this process is very efficient in increasing the overall system capacity and reducing collisions. Newly arrived stations do not wait for the collided stations to be transmitted before attempting contention. This means there are no newly arrived frames queued due to the ongoing collision resolution period (CRP).

In the case of ternary-tree splitting method, all collided frames are transmitted first. Newly arrived frames wait until all collided frames get transmitted. Hence, it is more likely that there will be a number of backlogged stations immediately following a CRP; this would, in turn, lead to more collisions. In fact, the dynamic slot allocation scheme is partially motivated by this shortcoming of ternary-tree splitting. Note also that the range over which the collided station chooses a backoff period--between 0 and 2^n-1 where n is the number of collisions for the frame--increases with n for CSMA/CD. This is in contrast to DDPQ or DFPQ, where stations always choose one of the three signal slots regardless of the number of times the frame has collided. Hence, the probability that

two colliding stations will choose the same backoff period is lesser in CSMA/CD than in DDPQ or DFPQ, especially when the frames have collided more than once.

Since DDPQ estimates the number of backlogged stations and allocates slots based on the estimate, it observes much lesser intra-priority collisions than DFPQ MAC protocol. The reduced overall collision overhead in DDPQ gives it a better throughout performance than DFPQ.

At higher offered loads, the capture effect of BEB process reduces the number of contending stations for periods of time (capture effect), thus resulting in lower collision resolution overhead for the system.

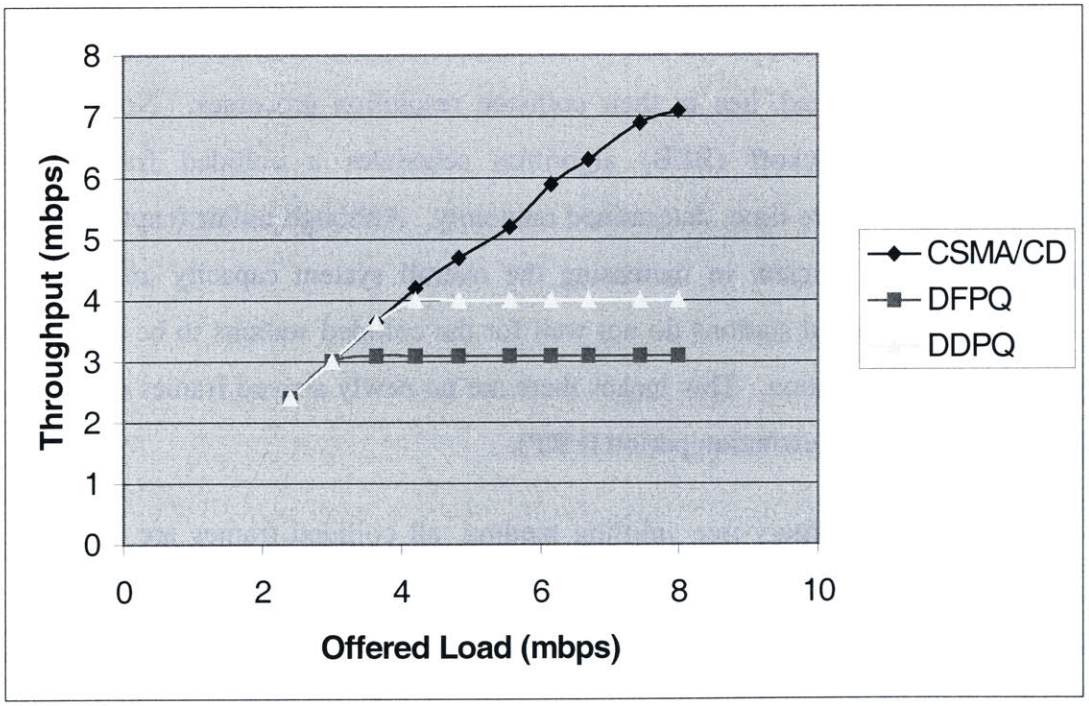


Figure 6.2: Scenario #1 Throughput vs. Offered Load

Figure 6.3 shows mean access delay vs. offered load. The mean access delay for CSMA/CD falls after a point, again because the captured stations don't have a chance to contribute as often to the delay statistics. Also note that any frame that collides more than 16 times is discarded. The lower mean access delay for CSMA/CD is also due to the reduced number of collisions in CSMA/CD compared to DFPQ and DDPQ. Collided frames have to undergo the signal slots to establish partial ordering and wait for their turn before being able to access the channel. Hence, they have longer delays. The mean delays for DDPQ and DFPQ saturate at about 1.8 ms and 2.1 ms respectively. The saturation is mainly due to the fact that all collided frames are transmitted before the newly arrived frames. The number of collisions per packet is lower in DDPQ than in DFPQ due to dynamic slot allocation, and hence frames are transmitted faster.

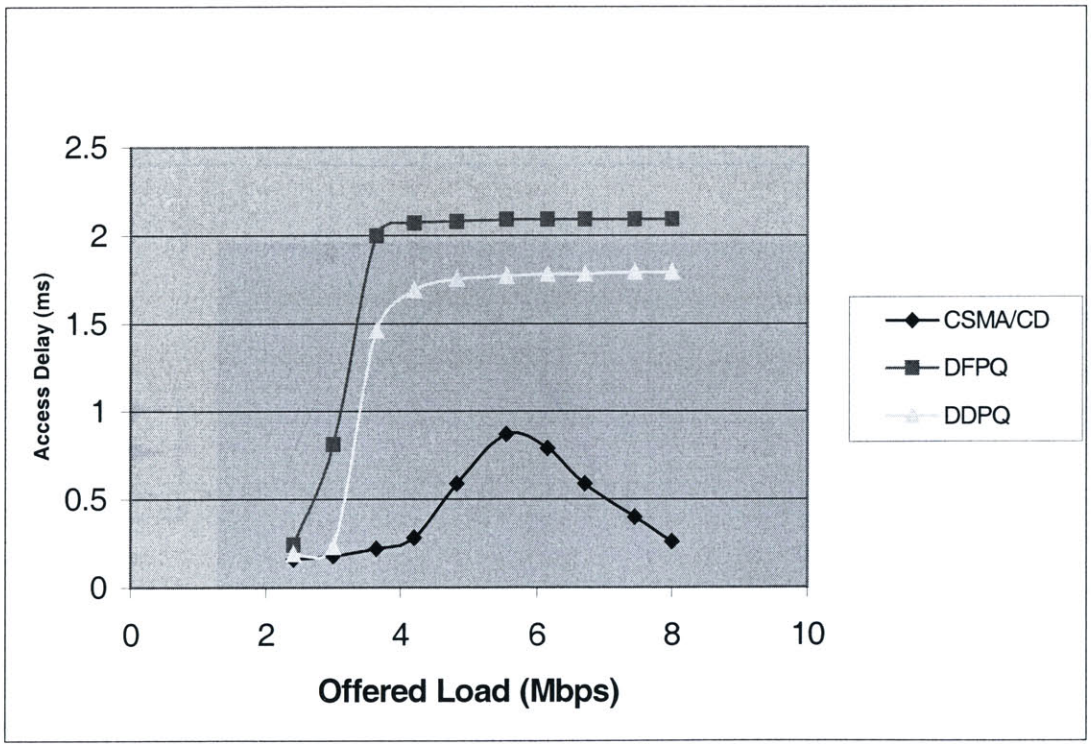


Figure 6.3: Scenario #1 Mean Access Delay vs. Offered Load

Figure 6.4 shows the access jitter vs. the offered load. Recall that jitter represents the difference between the highest and the lowest delay. It, therefore, gives us a sense of the worst case delay. The jitter of CSMA/CD increases exponentially to the point where it becomes unstable. At higher loads, it is about 6 times greater than that for DFPQ and about 15 times greater than DDPQ. Because of the unfairness in channel access due to capture effect, some frames suffer very long delays while others only suffer marginal delays before transmission. So the average delay may be small compared to DDPQ and DFPQ but the jitter and, therefore, the worst case delay is much larger.

In contrast, both DFPQ and DDPQ have frame delays that saturate. The jitter for DFPQ saturates at about 44 ms and the jitter for DDPQ saturates at about 18 ms. The bounded delay is mainly due to the fact that all collided frames are transmitted before the newly arrived frames. Bounded jitter is better for real-time traffic. The jitter for DFPQ is more than twice that of DDPQ, again due to greater collision per frame.

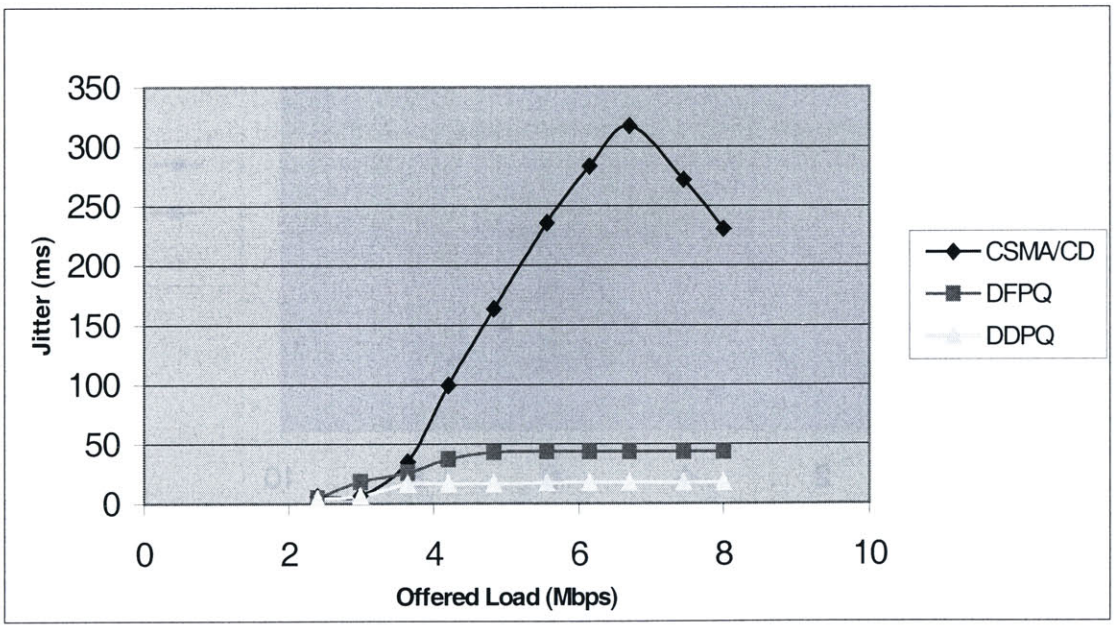


Figure 6.4: Scenario #1 Jitter vs. Offered Load

6.2.2. Scenario #2

We keep the offered load constant at about 5.84 Mbps. We vary the number of stations transmitting in a priority from 2 to 10. Note that all stations are transmitting in the same priority.

Figure 6.5 shows the system throughput against the number of stations. As can be seen across all three protocols, the throughput decreases as the number of stations increases. This is mainly due to the fact that inter-priority collisions increase with the number of stations. Collision resolution requires system resources and, therefore, reduces the overall system throughput. At large number of stations, the system throughput for CSMA/CD is greater than that for DFPQ by about 2 Mbps and than that for DDPQ by about 1.5 Mbps. As explained in the first scenario, this is because the BEB method, even though unfair, is very efficient in resolving collisions compared to the ternary tree splitting method. Since DDPQ estimates the number of backlogged frames at each contention cycle, it sees a lower number of collisions than DFPQ and, hence, achieves a greater system throughput.

Figure 6.6 and Figure 6.7 show the plot of the access delay and jitter respectively. The delay and jitter for all three protocols worsen as the number of stations increases. In case of DDPQ and DFPQ, this delay increase is almost linear. The number of collisions increases with the population of stations since it is more likely that more than one station will transmit at the same time. Likewise, the number of frames involved in a collision also increases. For a small range of numbers involved in a collision (from 2 to 10 frames) the expected length of collision resolution interval (CRI) increases linearly with the number [XuC93]. In the case of ternary-tree splitting, CRI denotes a collision resolution attempt through partial ordering. For example if 4 frames are involved in a collision, the CRI will be greater than 1. This is because at least 2 frames will choose the same slot and, therefore, collide again. Recall that each of the frames has to choose one of the 3 signal slots. DDPQ, by virtue of its dynamic slot allocation and, therefore, reduced intra-priority collisions, exhibits a better delay and jitter performance than DFPQ.

CSMA/CD gives a better delay performance than DDPQ and DFPQ. This is because when offered load is kept constant and the number of stations is varied, the ternary-tree splitting method's performance suffers considerably compared to BEB. The number of frames involved in a collision increases with the number of stations and so does the length of CRP. More frames will accumulate in the mean time, which will lead to more collisions. This is in contrast to BEB, where such frame buildup is not present and hence collisions are fewer and smaller in the number of frames involved. With BEB when the number of stations is small, some collided frames wait longer both due to the capture effect. Hence, the corresponding jitter is worse than that for DDPQ and DFPQ. At higher number of stations, capture effect is not present because each station has a smaller offered load and hence jitter performance is comparable to the other protocols.

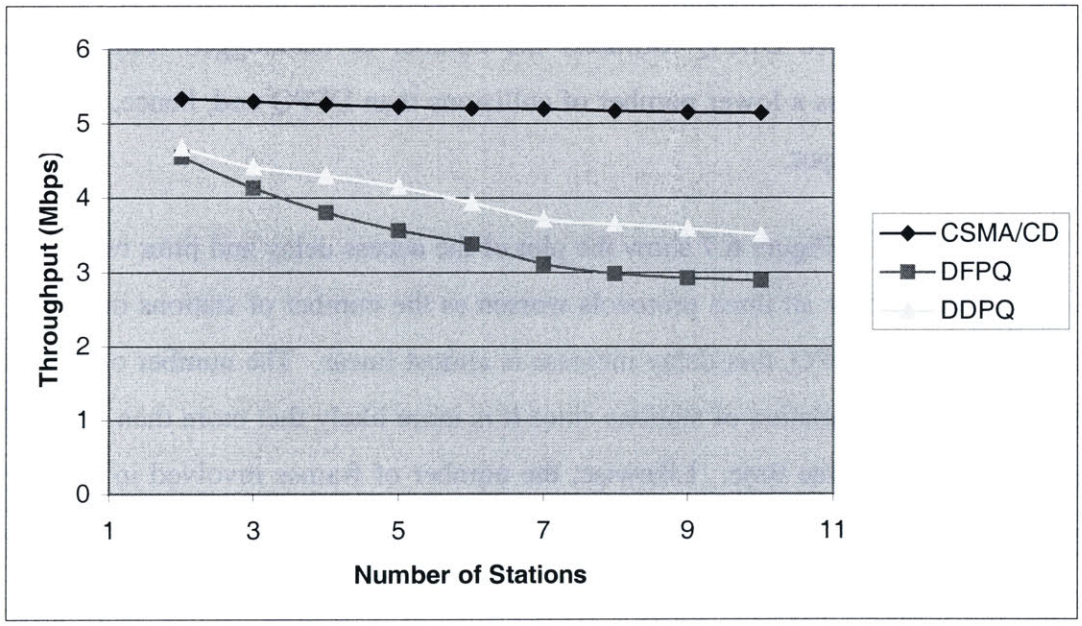


Figure 6.5: Scenario #2 Throughput vs. Number of Stations

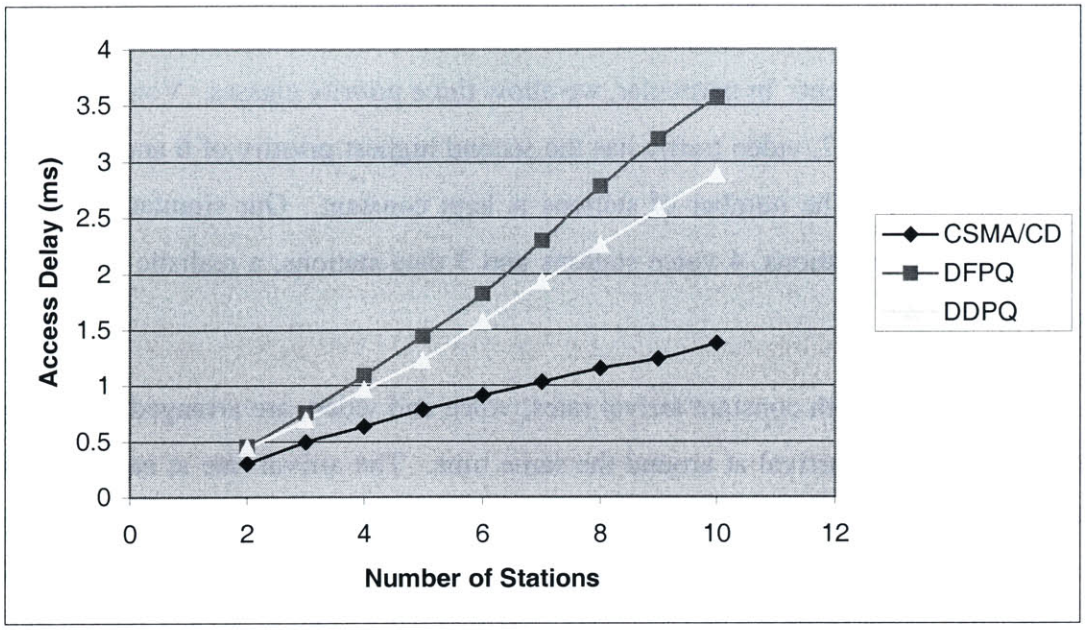


Figure 6.6: Scenario #2 Mean Access Delay vs. Number of Stations

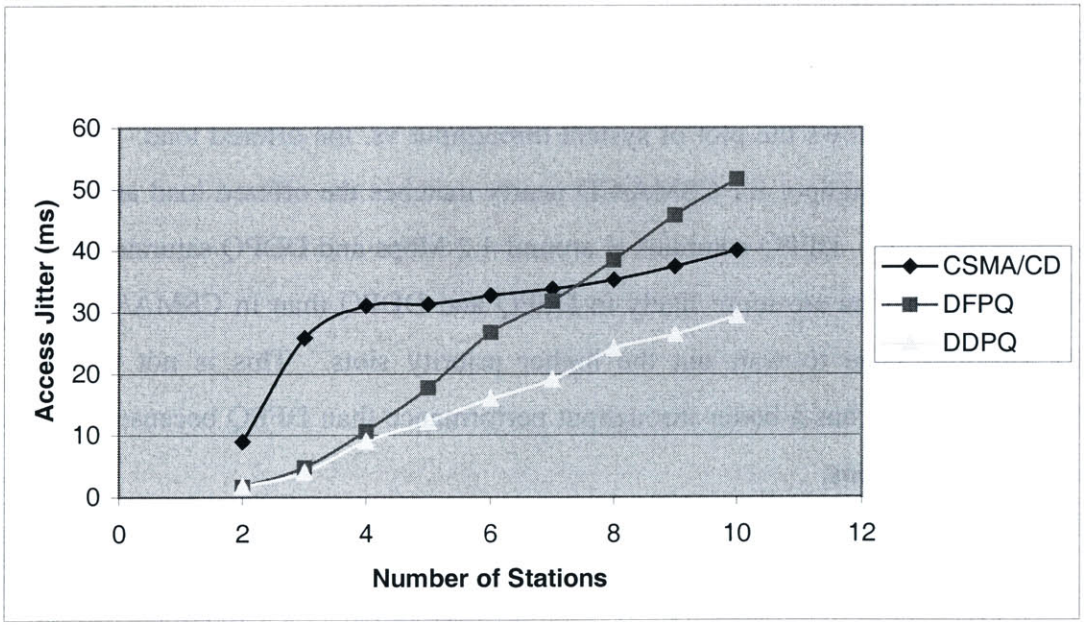


Figure 6.7: Scenario #2 Jitter vs. Number of Stations

6.2.3. Scenario #3

We now evaluate the performance of the algorithm for situations where various priority levels are present. In particular, we allow three priority classes. Voice traffic has the highest priority of 7, video traffic has the second highest priority of 6 and data traffic has a priority of 5. The number of stations is kept constant. Our simulation scenario consists of 2 Video stations, 4 voice stations and 3 data stations, a realistic scenario for home networks.

The sources with constant arrival rates, voice and video, are arranged so that they schedule their packet arrival at around the same time. The arrival rate at each station is different and approximates its application. For example, in the case of the video stations the arrival rate varies from 500 to 1800 packets per second. That is a variation from about 584 Kbps to 1.52 Mbps. In case of data stations, the arrival rate varies from 200 to 700 packets per second (234 Kbps to 818 Kbps). In case of voice station, the variation is from 50 to 300 packets per second (58 Kbps to 350 Kbps). We vary the mean arrival rate at each station to adjust the offered load. The proportion of each traffic in the overall offered load is kept roughly constant.

Figure 6.8 shows the plot of system throughput vs. the offered load. Once again we see that the throughput for CSMA/CD nearly matches the offered load and is higher than other protocols. DFPQ saturates at around 4.2 Mbps and DDPQ saturates at around 4.6 Mbps. Collisions are more likely in DFPQ and DDPQ than in CSMA/CD and the lower priorities have to wait out the higher priority slots. This is not the case in CSMA/CD. DDPQ has a better throughput performance than DFPQ because of reduced intra-priority collisions.

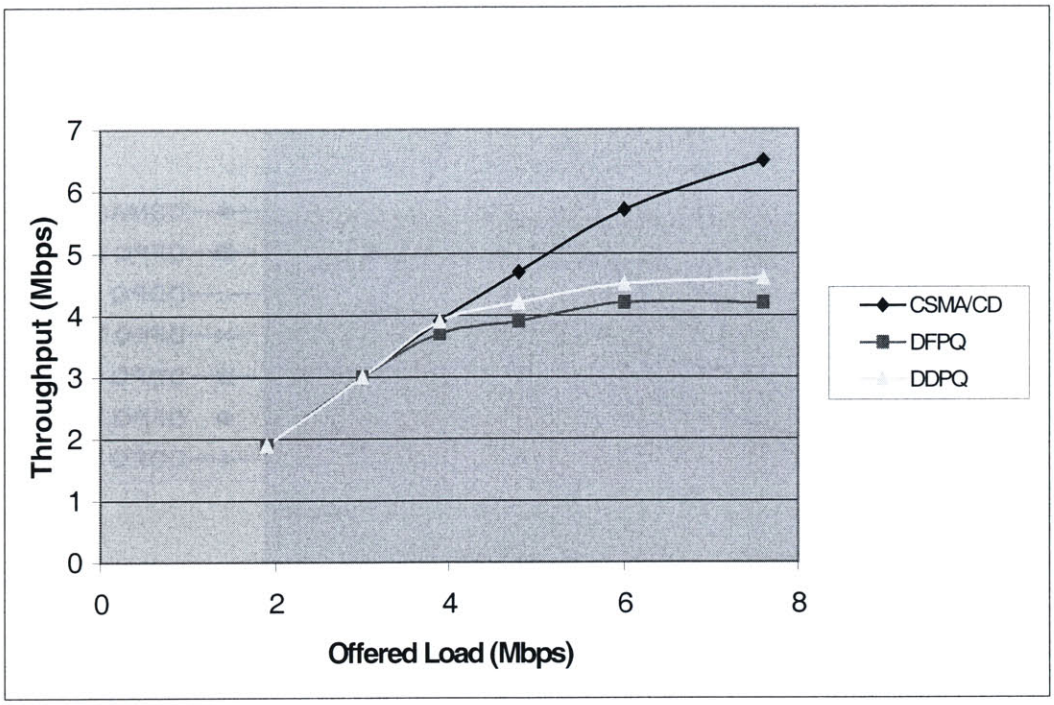


Figure 6.8: Scenario #3 Throughput vs. Offered Load

Figure 6.9 and Figure 6.10 depict the delay and jitter performance respectively for all three priority classes. While the results for individual priorities are presented in subsequent figures, the plots serve to show the comparative delay and jitter performances for the three priorities. As expected, voice and video frames which are assigned higher priorities have much better delay and jitter than data frames. Since data frames access channel only after video frames which occupy the bulk of the system throughput, its performance is significantly worse. In case of CSMA/CD there is no priority distinction; hence, all frames undergo the same delay and jitter performance.

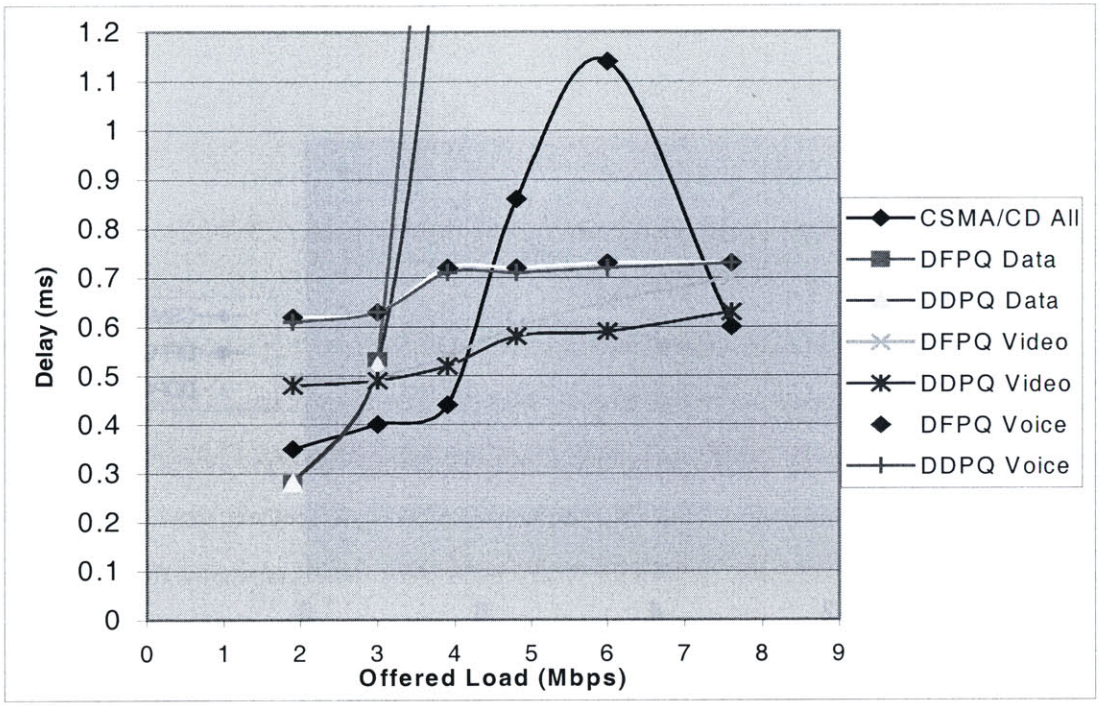


Figure 6.9: Scenario #3 Access Delay vs. Offered Load

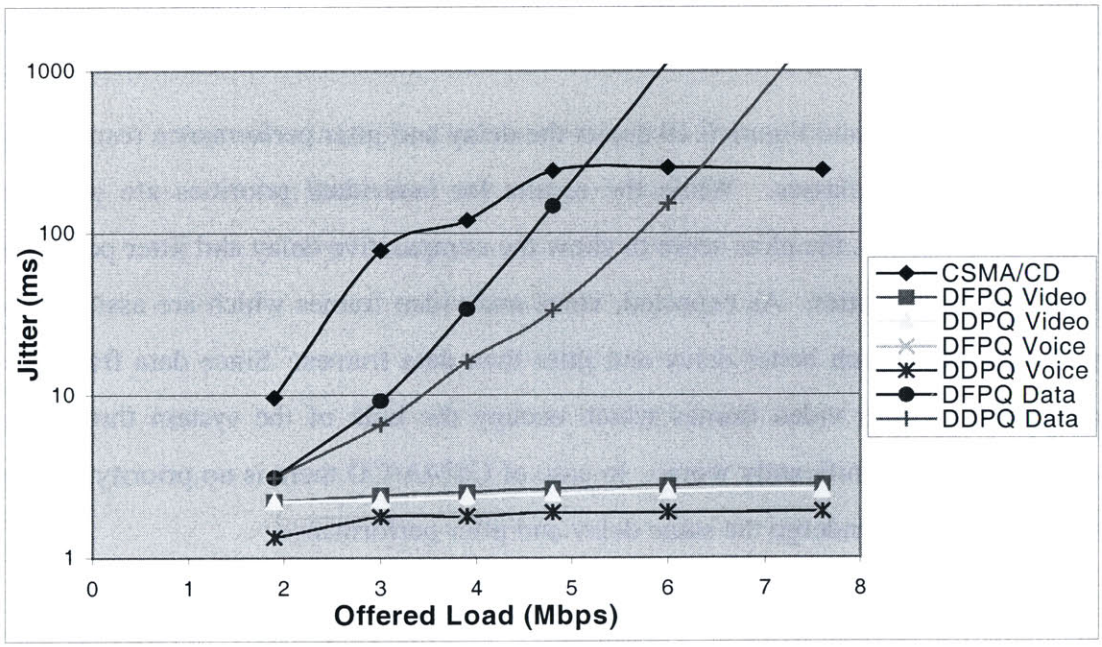


Figure 6.10: Scenario #3 Jitter vs. Offered Load

Figure 6.11 and Figure 6.12 show the mean access delay and jitter for voice frames vs. the offered load. Since CSMA/CD does not have a priority framework, its shows a uniform delay and jitter performance for all three traffics. The jitter performance of CSMA/CD is unsuitable for real-time traffic. In fact, even for an offered load of 4 Mbps the worst case delay is more than 100 ms, which is substantially greater than what is tolerated by voice and video frames (5 ms). The bad jitter performance is because the delay of voice frames in CSMA/CD depends on the overall offered load as all frames are treated equally. This is unlike DFPQ and DDPQ where voice frames get a preferential treatment (highest priority). Hence, voice frames experience good delay and jitter performances in those protocols. Because the offered load for voice frames are relatively low, both DFPQ and DDPQ exhibit similar delay and jitter characteristics. Note that the low offered load means that the backlog number for voice priority at any contention cycle will be low, most likely either 0 or 1. In such a case, DDPQ allocates either 0 or 1 slots, most likely 1 because extra slots are distributed among idle priorities going from the highest to the lowest. Hence, we see identical performances in the two protocols for voice frames.

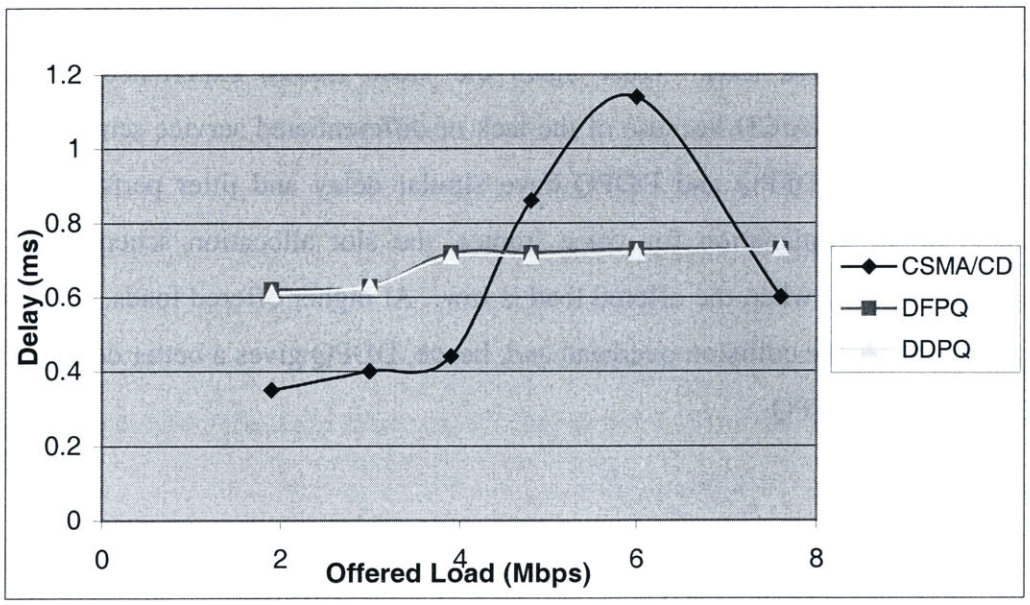


Figure 6.11: Scenario #3 Voice Access Delay vs. Offered Load

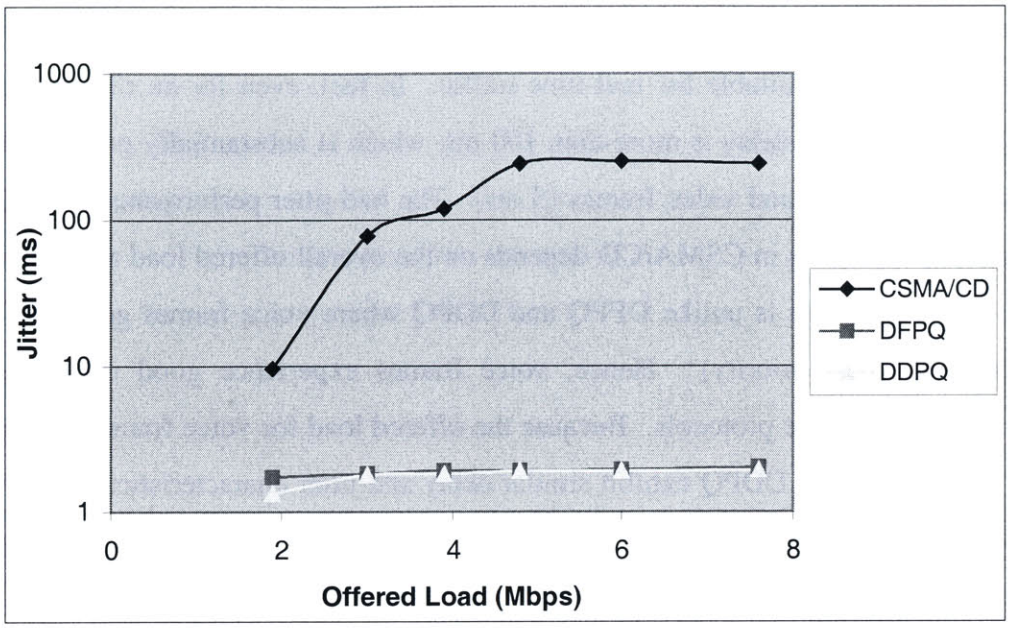


Figure 6.12: Scenario #3 Voice Jitter vs. Offered Load

Figure 6.13 and Figure 6.14 show the mean access delay and jitter for video frames vs. the offered load. Once again the video frames experience a bad jitter performance in CSMA/CD because of the lack of differentiated service scheme. At low offered loads, both DFPQ and DDPQ have similar delay and jitter performance. As explained in the explanation for voice frames, the slot allocation scheme of DDPQ approximates DFPQ when the offered load is low. At higher offered loads, dynamic slot allocation reduces the collision overhead and, hence, DDPQ gives a better delay and jitter performance than DFPQ.

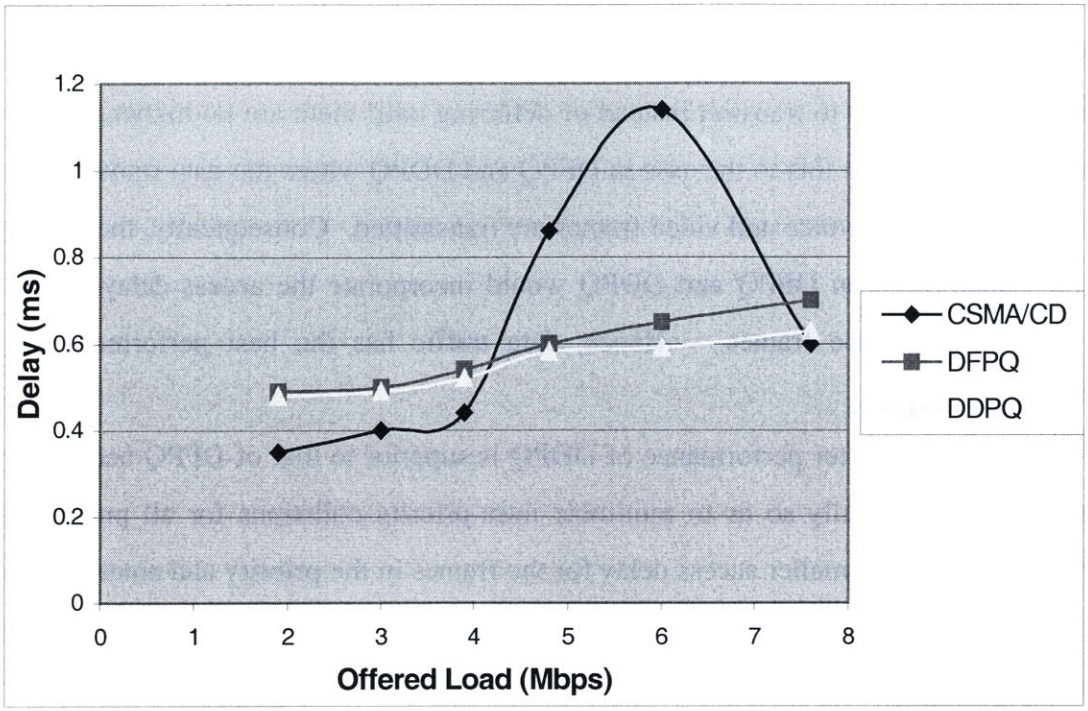


Figure 6.13: Scenario #3 Video Access Delay vs. Offered Load

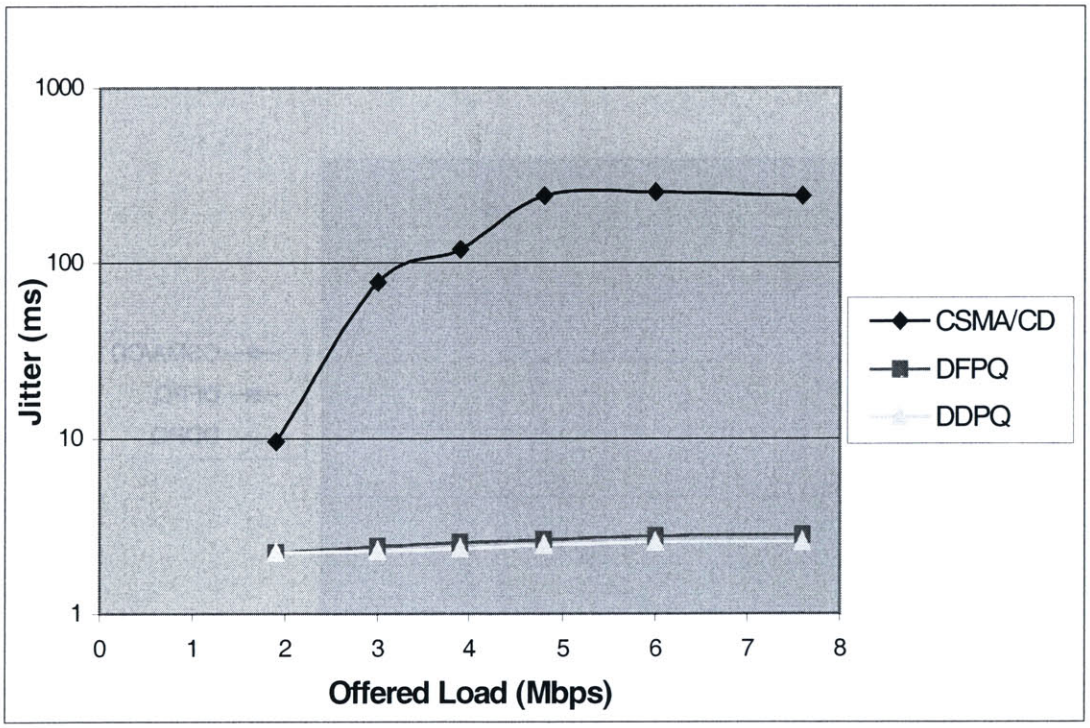


Figure 6.14: Scenario #3 Video Jitter vs. Offered Load

Figure 6.15 and Figure 6.16 show the mean access delay and jitter for data frames vs. the offered load. In CSMA/CD data frames contend with video and voice frames whenever they are ready to transmit instead of deferring until there are no higher priority frames present. Contrast this to the case in DFPQ and DDPQ where the data frames have to wait until all waiting voice and video frames are transmitted. Consequently, the access delay for a data frame in DFPQ and DDPQ would incorporate the access delays of all waiting video and voice frames. Hence, data traffic has the best performance in CSMA/CD.

The delay and jitter performance of DDPQ is superior to that of DFPQ because it allocates slots dynamically so as to minimize intra-priority collisions for all priorities. Lower collisions mean smaller access delay for the frames in the priority and since access delay also incorporates the access delays for higher priorities, lower priority frames perform better in DDPQ than in DFPQ.

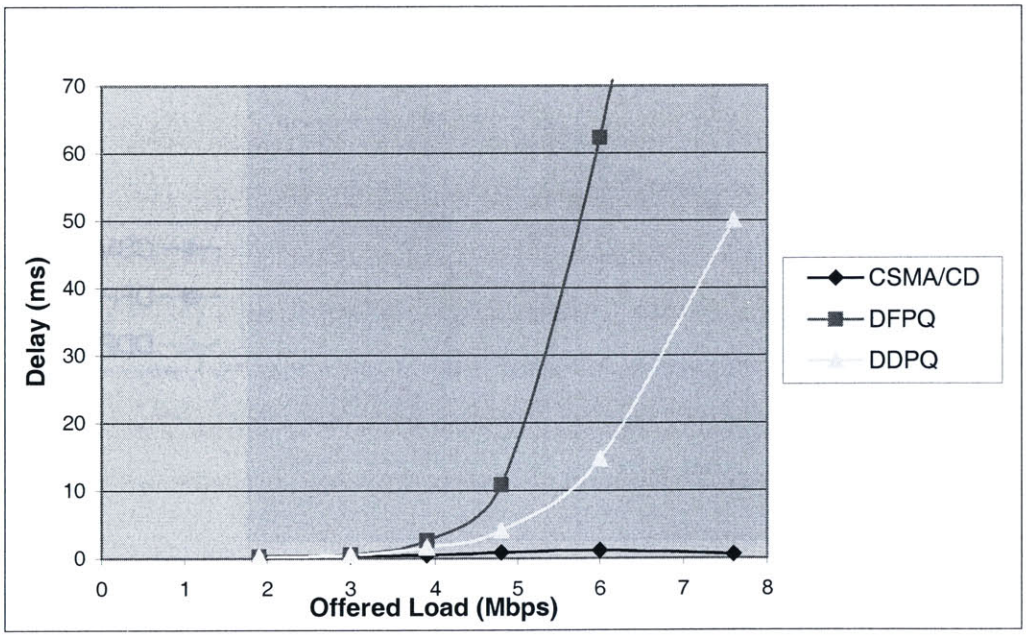


Figure 6.15: Scenario #3 Data Access Delay vs. Offered Load

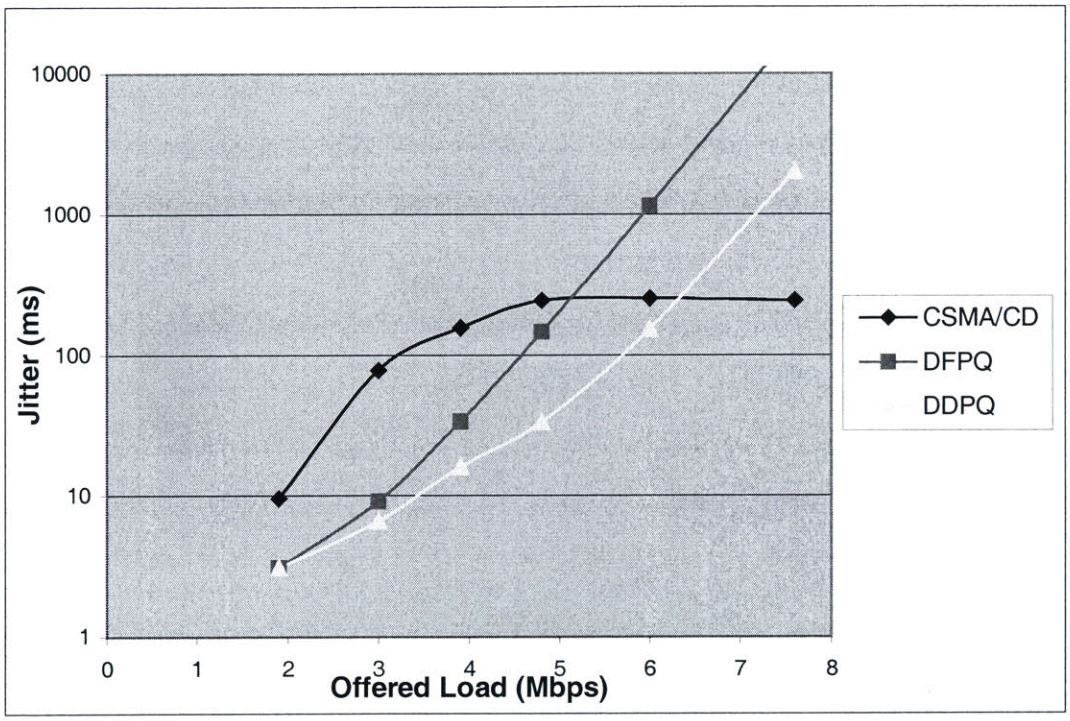


Figure 6.16: Scenario #3 Data Jitter vs. Offered Load

7. Conclusion

The problem facing a distributed LAN system is the fair resolution of medium contention among different traffic flows with diverse QoS requirements. Frames in real-time flows have a very strict delay and jitter requirement; frames that are significantly delayed cannot be used for reassembling the content and have to be discarded. Data frames, on the other hand, are non-real time and do not have a strict delay or jitter requirement. Traditional LANs were originally designed for best-effort data transmission and are not suitable for the QoS need of real-time multimedia traffic. In this thesis, we have presented a new MAC protocol, DDPQ, which addresses this problem of providing QoS to different traffic flows.

DDPQ represents an enhancement of an existing approach, the DFPQ, to provide a better QoS capability to real-time traffic. It similarly utilizes 8 priorities for channel contention and ternary-tree splitting mechanism for collision resolution. Collisions only occur among frames of the same priority. Where the protocols differ is how they allocate slots to the priorities. While DFPQ allocates contention slots in a static manner — one contention slot per priority in each contention cycle — DDPQ allocates variable number of contention slots based on the estimate of backlogged stations on each priority. The goal is to dynamically allocate contention slots such that intra-priority collisions are minimized. The protocol keeps an estimate of the offered load at each priority, wherein the offered load is updated every window length.

DDPQ proposes a dynamic contention slot allocation, similar to the method proposed by Ruszczyk, et. al. [RLC99a], [RLC99b] However, DDPQ is a distributed protocol and utilizes ICT to find the preferred slot allocation. In Ruszczyk et. al., a centralized scheduler determines the aggregate feedback for a priority class and accordingly determines the preferred mini-slot allocation in the contention-based reservation system.

Our simulation results show that DDPQ MAC protocol performs better than DFPQ and CSMA/CD MAC protocols. CSMA/CD is a best effort service and as can be seen from the simulation results, it does not provide QoS support for real-time traffic.

Besides it also suffers from the capture effect phenomenon at higher offered loads. At low backlog number, DDPQ performs similar to DFPQ; this is expected, as the contention slot allocated by DDPQ approximates the contention slot allocation scheme of DFPQ. At higher backlog numbers, DDPQ performs noticeably better than DFPQ. This is because it allocates contention slots dynamically in each cycle, such that there is a greater probability of success as compared to DFPQ, which always allocates one contention slot per priority in each cycle. The cumulative effect is more apparent when a mix of priorities are present. Since the delay and jitter performance at lower priorities are related to that for higher priorities, one expects to see a markedly better performance for lower priorities in DDPQ compared to DFPQ. Simulation results show that this is indeed the case.

DFPQ adds a priority structure over CSMA/CD to give preference to real-time frames over data frames. DDPQ provides an enhancement to DFPQ such that it dynamically adapts to changing offered load and hence has a more robust QoS capability. Hence, by virtue of this enhancement DDPQ is more suited to home networks where integrated traffic is prevalent.

Additional work on admission control is useful to further enhance the performance of the DDPQ MAC protocol. Recall that the lower priority frames always defer to higher priority frames. Hence, in order to ensure adequate performance for lower priority frames, it is essential to limit the offered load at higher priorities.

8. References

[802.3] Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements. Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, IEEE Standard 802.3, 1998

[Abr70] N. Abramson, "The Aloha System - Another Alternative for Computer Communications," Proc. of the Fall Joint Computer Conference, pp. 281-285, 1970.

[ALM98] G. Anastasi, L. Lenzini and E. Mingozzi, "Stability and Performance Analysis of HIPERLAN," IEEE Infocom, pp. 134, Mar./Apr. 1998.

[BeG92] D. Bertsekas and R. Gallagher, *Data Networks*, Prentice Hall, Inc., New-Jersey, 1992.

[Cap77] J. I. Capetanakis, "The Multiple Access Broadcast Channel: Protocol and Capacity Considerations," in *Ph.D. Dissertation*, Department of Electrical Engineering, MIT, Aug. 1977.

[CFF00] Campbell, Ferrari, Fischer, and Wolf, "Quality of Service in Networks and Distributed Systems," Dagstuhl, Wadern, Germany, no. 00191-274, May 2000.

[ChG97] H. S. Chhaya and S. Gupta, "Performance modeling of asynchronous data transfer methods of IEEE 802.11 MAC protocol," Wireless Networks, vol. 3, 1997.

[ChR85] G. L. Choudhary and S. S. Rappaport, "Priority schemes using CSMA-CD," IEEE Transactions on Communications, vol. COM-33, no. 7, Jul. 1985.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge, 1990.

[Dai87] J. N. Daigle, "Message Delays with Prioritized HOLP and Round-Robin Packet Servicing," IEEE Transactions on Communications, vol. COM-35, no. 6, pp. 609-619, 1987.

[FLB74] G. Fayolle, J. Labetoulle, D. Bastin, and E. Gelenbe, "The Stability Problem of Broadcast Packet Switching Networks Computer Networks," Acta Informatica, vol. 4(1), 1974.

[FrH00] E. H. Frank and J. Holloway, "HPNA 2.0 10Mbps Home Phoneline Networking," 2000 NCTA Technical Papers, May 2000, page 172-187.

[G.pnt00] "G.pnt: Media Access Protocol," B00-03-05-collision-QoS-00028, ITU Study Group 15, April 3-14, 2000.

[Hay78] J. F. Hayes, "An Adaptive Technique for Local Distribution," IEEE Transactions on Communications, vol. COM-26, Aug. 1978.

[HPNA] Home Phoneline Networking Association, http://www.homepna.org/.

[KaM98] Kalkunte and Mangin, "Method and Apparatus Avoiding Capture Effect by Adding a Slot Time to An Interpacket Gap Interval in a Station Accessing an Ethernet Network," U.S. Patent Number 5,854,900, Dec. 29, 1998.

[KKM98] Kalkunte, Kadambi, Mangin, and Krishna, "Rotating Priority Arrangement in an Ethernet Network," U.S. Patent Number 5,784,375, Jul. 21, 1998.

[MeB76] R. M. Metcalf and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," Communications of the ACM, vol. 19, no. 7, pp. 395-404, Jul. 1976.

[Mol97] Molle, "Binary Logarithmic Arbitration Method for Carrier Sense Multiple Access with Collision Detection Network Medium Access Control Protocols," U.S. Patent Number 5,600,651, Feb. 4, 1997.

[ObD93] M. S. Obaidat and D. L. Donahue, "WBTPE: A Priority Ethernet LAN Protocol," Proceedings of the Conference on 1993 ACM Computer Science, Indianapolis, Feb. 16 - 18, 1993.

[Riv87] R. L. Rivest, "Network Control by Bayesian Broadcast," IEEE Trans. on Information Theory, vol. IT-33(3), pp. 323-328, May 1987.

[RLC99a] Ruszczyk, et al., "System, device, and method for sharing contention mini-slots among multiple priority classes," United States Patent 5,886,993, Mar. 23, 1999.

[RLC99b] Ruszczyk, et al., " System, device, and method for contention based reservation in a shared medium network," United States Patent 5,960,000, Mar. 23, 1999.

[Rob75] L.G. Roberts, "ALOHA Packet System With and Without Slots and Capture," Computer Communications Review, Vol. 5, No. 2, Apr. 1975.

[RoS90] R. Rom and M. Sidi, *Multiple Access Protocols: Performance and Analysis*, Springer-Verlag NewYork Inc., New York, pp. 2-5, 1990.

[SoK98] J. L. Sobrinho and A. S. Krishnakumar, "EquB: Ethernet quality of service using black bursts," 20th IEEE Conference on Local Computer Networks, 1998.

[TaK85] H. Takagi and L. Kleinrock, "Throughput Analysis for Persistent CSMA Systems," IEEE Trans. Commun., vol. COM-33, pp. 627-683, Jul. 1985.

[Tob80] F. A. Tobagi, "Multiaccess Protocols in Packet Communication Systems," IEEE Trans. Commun., vol. COM-28, pp. 468-488, Apr. 1980.

[ToD96] F. A. Tobagi and I. Dalgic, "Performance Evaluation of 10Base-T and 100Base-T Ethernets Carrying Multimedia Traffic," IEEE Journal on Selected Areas in Communications, vol. 14, no. 7, pp. 1436-1454, 1996.

[ToH80] F. A. Tobagi and V. B. Hunt, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection," Computer Networks, vol. 4, pp. 245-259, 1980.

[ToK75] F. A. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part II - The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy

Tone Solution," IEEE Trans. on Communications, vol. COM-23, pp. 1417-1433, Dec. 1975.

[Tsi87] J. N. Tsitsiklis, "Analysis of a Multiaccess Control Scheme," IEEE Trans. on Automatic Control, vol. AC-32, pp. 1017-1020, Nov. 1987.

[TsM78] B. S. Tsybakov and V. A. Mikhailov, "Free Synchronous Packet Access in a Broadcast Channel with Feedback," Probl. Information Transmission, vol. 14, pp. 259-280, Oct.-Dec. 1978.

[ViZ95] M. A. Visser and M. El Zarki, "Voice and Data Transmission over an 802.11 wireless network," Proceedings of PIMRC'95, Toronto, Canada, Sep. 1995, pp. 648-652.

[XuC93] W. Xu and G. Campbell, "A Distributed Queueing Random Access Protocol for a Broadcast Channel," Sigcomm 93, San Francisco, Sept. 93.

[YaR94] H. Yang and K. K. Ramakrishnan, "The Ethernet capture effect: Analysis and solution," Proceedings of the 19th IEEE Local Computer Networks Conference, Oct. 1994.

A. Appendix

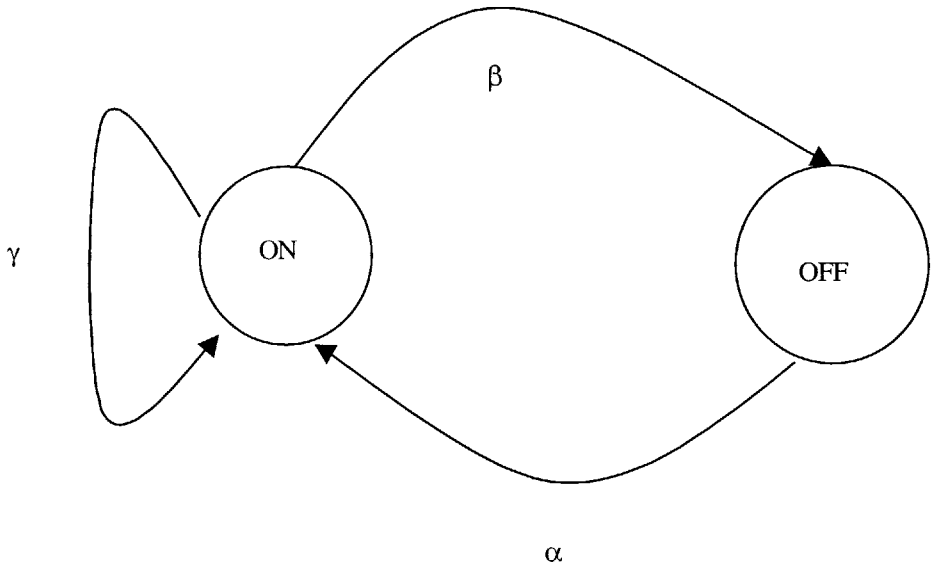
In this Appendix, we provide additional description of the traffic sources used in the performance evaluation simulations and Matlab codes for the algorithms used in the thesis.

A.1 Traffic Sources

Three kinds of traffic sources are considered.

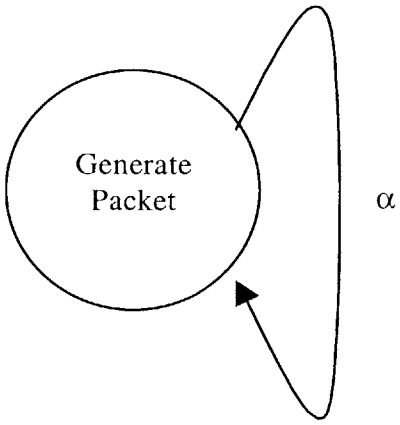
Voice Source Model

Voice source is modeled with two states: ON and OFF. The source is in the ON state when the speaker is actually talking. While in this state, the source generates fixed length packets (1 Kb) at regular intervals (1.56e-2). The source is in the OFF state when the speaker is silent. While in this state the source does not generate any packet. The time spent in ON and OFF states is exponentially distributed with mean α-1 (1.2s) and β-1 (1.8s) respectively.



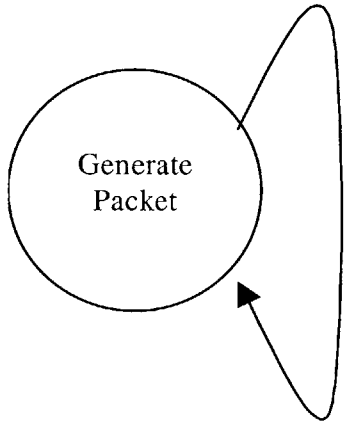
Data Source Model

Data source is modeled with one state. The source generates a packet every time it enters the Generate state. Packet size (constant, uniform distribution, exponential distribution) is dependent on the type of application. The arrival rate is exponentially distributed with mean α-1.



Video Source Model

Video source is also modeled with one state. But unlike data source that has an exponential arrival rate, video source is modeled to have a constant arrival rate. The packet sizes are also constant.



A.2 Throughput for BEB and Ternary-Tree Splitting

Probability of packet arrival

```
function[l] = get_a(num,time,arrival_rate)

delta = arrival_rate * time;

l = (delta^num)*exp(-delta)/factorial(num);
```

Busy Period

```
function[l] = get_B(time,arrival_rate)

T_suc = 1168/10000000;

IFG = 0.000023;

T_col = (32/10000000) + 3 * IFG;

m = T_suc + IFG;

CRM_2 = 1.5 * (IFG + 3 * 0.000026) + (0.5 * (T_col + IFG)) +
(2 * m);
CRM_3 = 2.25 * (IFG + 3 * 0.000026) + (1.25 * (T_col + IFG))
+ (3 * m);
CRM_4 = 3.115 * (IFG + 3 * 0.000026) + (2.115 * (T_col +
IFG)) + (4 * m);
CRM_5 = 4.026 * (IFG + 3 * 0.000026) + (3.026 * (T_col +
IFG)) + (5 * m);
CRM_6 = 4.951 * (IFG + 3 * 0.000026) + (3.951 * (T_col +
IFG)) + (6 * m);

%Collision Resolution Length for TERNARY TREE SPLITTING
%CRM = get_a(2,time,arrival_rate) * CRM_2 +
get_a(3,time,arrival_rate) * CRM_3 +
get_a(4,time,arrival_rate) * CRM_4 +
get_a(5,time,arrival_rate) * CRM_5 +
get_a(6,time,arrival_rate) * CRM_6;

%Collision Resolution Length for BEB
CRM = 0;

n = T_col + IFG + CRM;

qn = get_a(1,n,arrival_rate)/(1-get_a(0,n,arrival_rate));

qm = get_a(1,m,arrival_rate)/(1-get_a(0,m,arrival_rate));
```

```
B_Num = ((qn*(m-n) + n) * (1-qm) * (1-
get_a(0,n,arrival_rate))/((1-
get_a(1,m,arrival_rate))*(get_a(0,n,arrival_rate) +
get_a(1,n,arrival_rate)))) + ((qm*(m-n) + n)/(1-
get_a(1,m,arrival_rate)));

B_Denom = 1 - ((1-get_a(0,m,arrival_rate)) *
get_a(1,n,arrival_rate) * (1-qm)/(1-get_a(1,m,arrival_rate))
* (get_a(0,n,arrival_rate) + get_a(1,n,arrival_rate)));

Bm = B_Num/B_Denom;

Bn = ((Bm * qn * (1 - get_a(0,n,arrival_rate))) + (qn * (m -
n)) + n)/(get_a(0,n,arrival_rate) +
get_a(1,n,arrival_rate));

qt = get_a(1,time,arrival_rate)/(1-
get_a(0,time,arrival_rate));

l = (qt * (m + ((1-get_a(0,m,arrival_rate)) * Bm))) + ((1-
qt) * (n + ((1 - get_a(0,n,arrival_rate)) * Bn)));
```

Utilized Period

```
function[1] = get_U(time,arrival_rate)

IFG = 0.000023;

T_suc = 1168/10000000;

T_col = (32/10000000) + 3*IFG;


m = T_suc + IFG;


CRM_2 = 1.5 * (IFG + 3 * 0.000026) + (0.5 * (T_col + IFG)) +
(2 * m);
CRM_3 = 2.25 * (IFG + 3 * 0.000026) + (1.25 * (T_col + IFG))
+ (3 * m);
CRM_4 = 3.115 * (IFG + 3 * 0.000026) + (2.115 * (T_col +
IFG)) + (4 * m);
CRM_5 = 4.026 * (IFG + 3 * 0.000026) + (3.026 * (T_col +
IFG)) + (5 * m);
CRM_6 = 4.951 * (IFG + 3 * 0.000026) + (3.951 * (T_col +
IFG)) + (6 * m);
```

```
%Collision Resolution Length for TERNARY TREE SPLITTING
%CRM = get_a(2,time,arrival_rate) * CRM_2 +
get_a(3,time,arrival_rate) * CRM_3 +
get_a(4,time,arrival_rate) * CRM_4 +
get_a(5,time,arrival_rate) * CRM_5 +
get_a(6,time,arrival_rate) * CRM_6;

%Collision Resolution Length for BEB
CRM = 0;

n = T_col + IFG + CRM;

qn = get_a(1,n,arrival_rate)/(1-get_a(0,n,arrival_rate));

qm = get_a(1,m,arrival_rate)/(1-get_a(0,m,arrival_rate));

T_u_col = 0;

%T_u_col = get_a(2,time,arrival_rate) * (2 * T_suc) +
get_a(3,time,arrival_rate) * (3 * T_suc) +
get_a(4,time,arrival_rate) * (4 * T_suc) +
get_a(5,time,arrival_rate) * (5 * T_suc) +
get_a(6,time,arrival_rate) * (6 * T_suc);

U_n1 = qm * (T_suc - T_u_col);

U_n2 = qn * (T_suc - T_u_col);

U_n3 = get_a(0,n,arrival_rate)+ get_a(1,n,arrival_rate);

U_n4 = (1-qm) * (1-get_a(0,n,arrival_rate))/(1-
get_a(1,m,arrival_rate));

U_Num = ((U_n1 + T_u_col)/(1-get_a(1,m,arrival_rate))) +
(U_n4 * (U_n2 - T_u_col)/U_n3);

U_Denom = 1 - (U_n4*qn*(1-get_a(0,m,arrival_rate))/U_n3);

Um = U_Num/U_Denom;

Un = (U_n2 - T_u_col + (qn*(1-
get_a(0,m,arrival_rate))*Um))/U_n3;

qt = get_a(1,time,arrival_rate)/(1-
get_a(0,time,arrival_rate));

l = (qt * (T_suc + ((1-get_a(0,m,arrival_rate)) * Um))) +
((1-qt) * (T_u_col + ((1 - get_a(0,n,arrival_rate)) * Un)));
```

Throughput

```
function[l] = get_throughput(arrival_rate)

IFG = 0.000023;
I = IFG/(1-exp(-arrival_rate*IFG));
B = get_B(IFG,arrival_rate);
U = get_U(IFG,arrival_rate);

l = U/(B + I);
```

Plot

```
i = 100:100:20000;

for j = 1:200
rh(j) = get_throughput(i(j));
end

plot(i,rh)
```

A.3 Instantaneous Contention Throughput

```
function[l] = ICT(x,b)

if b == 0
  l = 0;
else
if x == 0
  l = 0;
else

       if b == 1
          if x == 1
             l = 1;
          else
            l = 0;
            for i = 1:x,
               l = ((1/x)*(1/i)) + l;
            end
          end

       else
          if x == 1
             l = 0;
          else
            num_s = 0;
```

```
            denom_suc = 0;
            denom_coll = 0;

            for y = 1:x-1,
              num_s = num_s + ((x-y)^(b-1));
              denom_suc = denom_suc + (y*((x-y)^(b-1)));
              denom_coll = denom_coll + (y*((x-y+1)^(b-2)));
            end

            num = num_s * b/(x^b);

            denom_s = num * (denom_suc);

            denom_c = (1-num) * (denom_coll + x) * (b-1)/2;

            l = num_s/(denom_s + denom_c);

          end
        end
end
end
```