SIMULATION OF COMMUNICATION PROTOCOLS

by

Abdulaziz H. Al-Zoman

TR94-090 November 1994

CS6431 Report - Reading Course
with
J.M. DeDourek and B.J. Kurz

Faculty of Computer Science
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada E3B 5A3

Phone: (506) 453-4566
Fax: (506) 453-3566

```
98



User_id         : aziz
Host            : vermis.cs.unb.ca
Student Name    : Aziz_AlZoman
Date            : 11/14/94
Time            : 11:51:29
Doc type        : Special
Pages Printed   : 98

Page Credits Before Job : 110
Current Page Credits    : 12


Comments : Job Completed no errors



Page Count for This Printer : 142156
```

98 05/450

CS 6431 Report
Simulation of Communication Protocols

Abdulaziz H. Al-Zoman
202607

Prof. J. DeDourek Dr. B. Kurz

Faculty of Computer Science
THE UNIVERSITY OF NEW BRUNSWICK
May–June 1992

Abstract

This report is divided into two parts. The first part is devoted to highlight some topics related to the lowest two layers of the OSI model, i.e. the physical layer and the data link control layer. This report will not attempt to cover all the subjects associated to those layers. Nevertheless, it will pursue to elaborate on some of those topics that are of much interest to the reading course on simulation of communication protocols, "CS6431". The second part gives a review of basic simulation techniques and their applicability to simulation of communications protocols.

Contents

List of Figures

Part I

Channels and Protocols

Chapter 1

Channels

A typical communication system (point-to-point) model is depicted in Figure 1.1. Information, in the form of messages or data, is transmitted from the information source (or transmitter) to the destination (or receiver) over a communication channel.

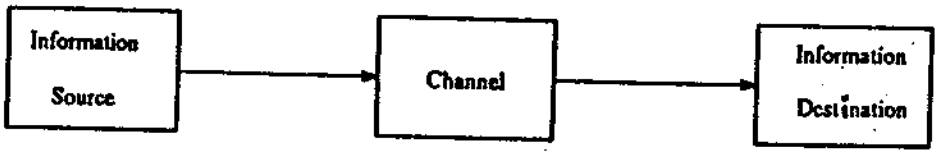

Figure 1.1: A Communication System Model

The source (e.g. a computer or terminal) forms a sequence of symbols (a message) selected from a finite alphabet $S = \{s_1, s_2, s_3, ..., s_M\}$, where M is the alphabet size. Then this message is changed into a signal (e.g. an electrical current wave or electromagnetic wave), and later transmitted to the destination over the communication channel (e.g. wire, air, or space). The receiver changes the transmitted signal back into a message [21].

1.1 Information Theoretical Treatment

This section highlights some of the important issues related to the information theory, such as information measures, bandwidth and spectrum of a signal, and channel capacity.

1.1.1 Information Measures

An essential feature of the information theory is that it is not worried about the meaning, but rather the amount of information. In 1920s, Hartley pointed out that the reasonable choice for a measure of information is a logarithmic unit [21].

In a usual communication system, the interest is placed on the transmission of information from an information source to information destination via a channel. The source is assumed to have available for forming a message an alphabet S of size M. The *self-information* [28], $I(s_i)$, is defined as the information gain when a symbol s_i has occurred,

$$I(s_i) = -log_2\, p(s_i) \;\; bits, \tag{1.1}$$

where $p(s_i)$ is the probability of occurrence of the *ith* symbol s_i.

The *source entropy* ($H(S)$), which is defined in [8] as the average amount of information obtainable per symbol from the source (over the alphabet S), equals to [28]

$$\begin{aligned} H(S) &= \sum_{i=1}^{M} p(s_i)\, I(s_i) \;\; bits, \\ &= -\sum_{i=1}^{M} p(s_i)\, log_2 p(s_i) \;\; bits, \end{aligned} \tag{1.2}$$

which has the following properties [28]:

- $0 \leq H(S) \leq log_2 M$.
- $H(S) = 0$ – if and only if all the probabilities are 0 except for one, which must be 1.

- $H(S) = log_2 M\ bits$ – if and only if all the probabilities are equal to $1/M$.

The information destination has also available an alphabet R of size N. For each symbol s_i sent from the source, symbol r_j is selected at the destination. Similar to the source entropy, the *destination entropy* [28] can be defined as,

$$H(R) = -\sum_{j=1}^{N} p(r_j)\ log_2 p(r_j)\ \ bits, \tag{1.3}$$

Preceding reception, the probability of selecting the input symbol s_i for transmission is $p(s_i)$. This is called *"a priori probability"* of s_i. After reception of r_j, the probability that the input symbol s_i becomes $p(s_i|r_j)$ (the conditional probability that s_i is sent given that r_j is received). This is called *"a posteriori probability"* of s_i. The change in the probability indicates (or measures) how much the receiver deduced from the reception of the r_j [28].

Therefore, the information gained, known as the *mutual information* [28], $I(s_i; r_j)$, due to the reception of r_j is defined as the logarithmic difference between the a priori probability and the a posteriori probability; that is

$$I(s_i; r_j) = log_2[p(s_i|r_j)/p(s_i)]\ \ bits. \tag{1.4}$$

Additionally, the *average* mutual information $I(S; R)$, which is an information measure of the whole communication system, is given by

$$I(S; R) = \sum_{i=1}^{M}\sum_{j=1}^{N} p(s_i, r_j)\ I(s_i; r_j), \tag{1.5}$$

where, $p(s_i, r_j)$ is the joint probability defined as

$$\begin{aligned} p(s_i, r_j) &= p(s_i|r_j)\ p(r_j), \\ &= p(r_j|s_i)\ p(s_i). \end{aligned} \tag{1.6}$$

Subsequently, the joint (or system) entropy, $H(S, R)$, is defined as

$$H(S, R) = -\sum_{i=1}^{M}\sum_{j=1}^{N} p(s_i, r_j)\ log_2 p(s_i, r_j), \tag{1.7}$$

Statistically, S and R, typically, are not independent, then $H(S,R)$ can be written as

$$\begin{aligned} H(S,R) &= H(S) + H(R|S), \\ &= H(R) + H(S|R), \qquad (1.8) \end{aligned}$$

where $H(R|S)$ and $H(S|R)$ are called conditional entropies and defined as follows:

$$\begin{aligned} H(R|S) &= -\sum_{i=1}^{M}\sum_{j=1}^{N} p(s_i, r_j)\ log_2 p(r_j|s_i), \\ H(S|R) &= -\sum_{i=1}^{M}\sum_{j=1}^{N} p(s_i, r_j)\ log_2 p(s_i|r_j) \ \ bits, \qquad (1.9) \end{aligned}$$

Based on all the above equations, it can be shown that

$$\begin{aligned} I(S;R) &= H(S) + H(R) - H(S,R), \\ &= H(S) - H(S|R), \qquad (1.10) \\ &= H(R) - H(R|S). \qquad (1.11) \end{aligned}$$

Equation 1.10 states that, after the transmission is through, the average information gained regarding the message corresponds to the average source information less the average uncertainty that still remains about the message. That is, the $H(S|R)$ represents the average information necessary at the destination after reception to sufficiently formulate the message sent. In other words, $H(S|R)$ presents the information lost in the channel.

Likewise, equation 1.11 shows that the information transmitted corresponds to the destination entropy less some information that is not from the source. So, the expression $H(R|S)$ can be viewed as a noise entropy added in the channel [28]. Figure 1.2 illustrates, pictorially, the relationship between the different aforementioned information measurements and entropies.

The preceding discussion assumes that the information source transmits a single symbol. However, in practice, the interest is rather placed on an information source

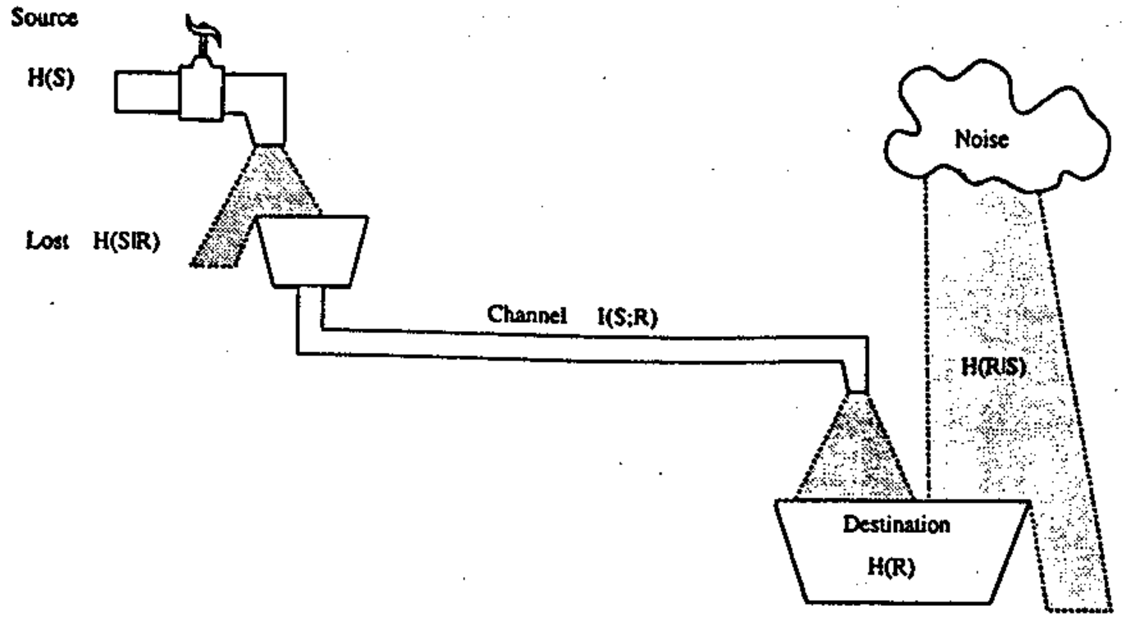

Figure 1.2: Communication system entropies relationship

which transmits a sequence of symbols. Interrelationships exist between symbols in a transmitted message. There are two classes of information sources based on the interrelationships present between the input symbols, namely, *zero-memory source* and *mth-order Markov source* [28].

Zero-memory Information Source

In this case, each symbol in the sequence occurred independently (i.e. transmitted symbols are statistically independent). The entropy of a sequence of q symbols (denoted by S^q, where S is the source alphabet) corresponds to q times the source entropy, that is

$$H(S^q) = qH(S) \ \ bits/sequence. \tag{1.12}$$

Thus, the average amount of information for all sequences are obtained by multiplying the corresponding symbol measure by the number of symbols in the sequence [28].

mth-order Markov Information Source

In practice, for an information source, it is usually assumed that the occurrence of a given symbol depends on some number m of last preceeding symbols. Hence, it is called mth-order Markov source. In the following $m = 1$ is considered (i.e. first-order Markov source) which means that the probability of selecting a symbol depends on the previous selected symbol.

For a first-order Markov source, there are M (alphabet size) possible symbols $s_i (i = 1, \ldots, M)$ can proceed a given symbol position. For any selection out of those symbols, the source is assumed to move from state q_i to state q_j with a probability $p(q_j|q_i) = p_{ij}$, where q_i and q_j are in $Q = \{q_1, q_2, \ldots, q_M\}$. The set of all such conditional probabilities is compressed by the transition matrix T (Markov matrix), where

$$T = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1M} \\ p_{21} & p_{22} & \cdots & p_{2M} \\ \vdots & & & \\ p_{M1} & p_{M2} & \cdots & p_{MM} \end{bmatrix}$$

For each state, an (state) entropy $H(q_i)$ can be defined by

$$H(q_i) = -\sum_{j=1}^{M} p(q_j|q_i)\ log(q_j|q_i). \tag{1.13}$$

For the first-Markov source each state is precisely specified by one symbol of the source alphabet, hence $p(q_j|q_i) = p(s_j|s_i)$. Therefore, equation 1.13 can be written as

$$H(q_i) = H(s_i) = -\sum_{j=1}^{M} p(s_j|s_i)\ log_2(s_j|s_i). \tag{1.14}$$

Finally, the source entropy $H(S)$ can be expressed as

$$\begin{aligned} H(S) = H(Q) &= -\sum_{i=1}^{M} p(s_i) H(s_i) \\ &= -\sum_{i=1}^{M}\sum_{j=1}^{M} p(s_i) p(s_j|s_i) log(s_j|s_i) \\ &= -\sum_{i=1}^{M}\sum_{j=1}^{M} p(s_i, s_j) log(s_j|s_i) \\ &\leq log_2 M \quad bits/symbol. \end{aligned} \tag{1.15}$$

1.1.2 Signal's Spectrum and Bandwidth

The *spectrum* of a signal is the range of frequencies that it contains. In other words, it is the difference between the upper and the lower frequency ranges of the components that make up the signal. The width of the spectrum is called the *absolute bandwidth* of a signal. In many cases, signals have an infinite absolute bandwidth. The relatively narrow band of frequencies, which contains most of the energy in the signal, is called *effective bandwidth,* or just *bandwidth,* see Figure 1.3 [23,9].

1.1.3 Channel capacity

The data rate of a signal is limited by, besides others, its bandwidth. That is, the greater the bandwidth of a signal, the higher the data rate which can be transmitted using that signal. The *channel capacity* refers to the (maximum) rate at which information can be transmitted over a given communication channel without error, under given conditions [23,18]. The channel capacity, C, is related to the average mutual information I(S;R) described in Section 1.1.1:

$$C = max_{p(s_i)} I(S;R) \;\; bits/sec. \tag{1.16}$$

Nyquist, in 1924, derived an equation expressing the capacity, C, of noiseless channel with a limited bandwidth:

$$C \;=\; 2B \; log_2 L \;\; bits/sec, \tag{1.17}$$

where B is the bandwidth of the signal, and L is the number of signaling levels [26,23].

The channel capacity is not only limited by the bandwidth, but also by the noise added to the transmitted signal. The amount of (white) noise present is measured by the ratio of the signal power to the noise power (i.e. the signal-to-noise ration; S/N). In 1948, Shannon, who is considered the father of the information theory, extended Nyquist's work to the case of a noisy channel. Shannon's major result about noisy channels is that the capacity, C, of any channel whose bandwidth is B Hz, and whose signal-to-noise ratio is S/N, is given by the following upper bound expression [21,26]:

$$C = B \ log_2(1 + S/N) \ \ bits/sec. \tag{1.18}$$

The relationship between the number of signaling levels (i.e. the number of bits per cycle) and the S/R can be found by combining the two equations 1.17 and 1.18 together. Thus, L can be expressed on the N/S as

$$L = \sqrt{1 + S/N} \ \ levels/signal. \tag{1.19}$$

1.2 Electrical Properties

Passband

The frequency spectrum of a communications channel is partitioned into two sections: *passband* and *stopband*, see Figure 1.3. Frequencies within the passband are transmitted with little or without any distortion, whilst those in the stop-band are rejected [8].

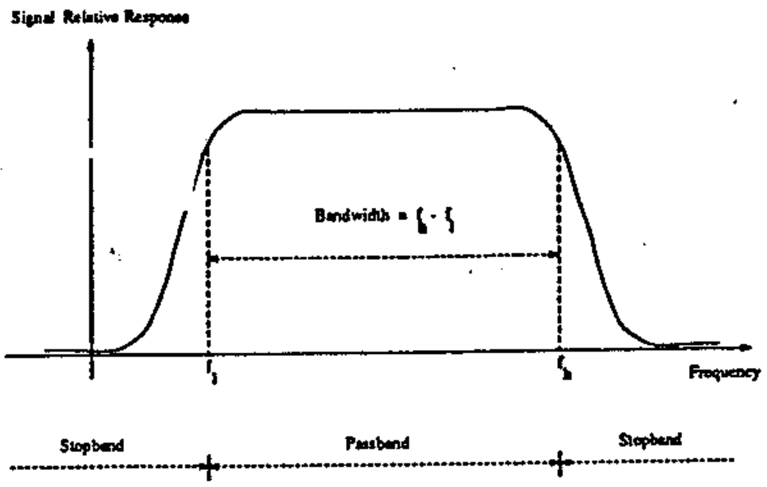

Figure 1.3: Signal frequency response

Signal Distortions

Distortions are undesirable modification in a signal [18]. There may be two distinguishable types of signal distortions corresponding to the amplitude ($A(\omega)$) and phase ($\phi(\omega)$) properties of the channel transfer function $H(\omega) = A(\omega)e^{j\phi(\omega)}$. If $A(\omega)$ is not a constant with the frequency, ω, then it causes an *amplitude distortion*, and if $\phi(\omega)$ is not linear with frequency, then a *phase (delay) distortion* is occurred [8,20].

- *Attenuation Distortion:* The amplitude distortion has two forms: gain or loss (attenuation). Here the emphasis will be placed on the latter. Attenuation is, as defined by Martin [18], "a decrease in magnitude of current, voltage, or power of a signal in transmission between points". In other words, the strength of a signal drops down with distance over any transmission media. The attenuation of the transmitted signal is not equal for all frequencies, i.e. the attenuation varies as a function of frequency [23].

  The attenuation has normally a logarithmic relationship with the distance between the transmitter and receiver over guided media, but has a more complex (e.g. a square) function of distance in case of unguided media. Repeaters or amplifiers are used to boost the signal at definite distances to overcome the attenuation [23].

- *Delay distortion:* There is a certain delay between the transmission of a signal at the source and its reception at the destination, typically, over a guided transmission medium. Usually, the signal is delayed more at some frequencies than at others [18,23]. That is, higher frequencies are not delayed as much as lower frequencies. Thus, the received signal is distorted due to variable frequency delay. It is notably in digital data transmission, where some of the signal frequency components of one bit position mix up with other bit position, resulting *intersymbol interference* [23].

Propagation Delay

It is the time needed to transmit a signal from the source to the destination. On land lines (e.g. guided media) this delay is about 6 to 10 $\mu s/km$, and on a satellite link, it is around 250 to 300 *ms/satellite hop* [9]. Propagation time, at high transmission speed, affects block size determination in error control techniques.

Noise

It is the undesirable electrical signals, inserted due to channel property components or natural disturbances, which tend to reduce the performance of communications channels [18]. That is, the received signal will correspond to the transmitted signal, changed by the different distortion introduced by the transmission system, and additional undesirable signal (noise) that are inserted somewhere between transmission and reception. Normally, there are various sources of noise. Nevertheless, four of them are discussed, namely, white noise, intermodulation noise, crosstalk, and impulse noise [23].

1. *White Noise:*

   Most of the electrical devices and transmission media possess it. It is known also as thermal noise which is the outcome of thermal agitation of electrons in a conductor. White noise is not a function of frequency, but rather temperature. The white noise power for a bandwidth 1 *Hz* equals to

   $$kT \quad watt/Hz, \tag{1.20}$$

   where, k is the Boltzmann's constant ($1.3803 \times 10^{-23}\ J/^{o}K$), and T is the temperature, degrees Kelvin.

   White noise can not be removed, hence it places an upper bound on communications system performance as shown by Shannon's equation 1.18.

2. *Intermodulation Noise:*

   It is the result of using a shared medium, i.e. multiple signals at different frequencies share the same transmission medium (e.g. frequency division multiplexing). Thus, signals at a frequency which is the difference or sum of the original frequencies or their multiples are produced. Those produced signals may interfere with the exist one.

3. *Crosstalk:*

   It is undesirable crossing between signal channels. For example, it can appear by electrical coupling between adjacent twisted pair. Crosstalk is of the equal order of magnitude as the white noise.

4. *Impulse Noise:*

   It is noncontinous, consisting of unsmoothed pulses (noise spikes) of short duration and of relatively high amplitude. The duration of the impulses can be rather long relative to the data transmission rate. Usually, a noise impulse removes or inserts two or more adjacent data bits. There are many sources of impulses noise, it may occur from within the communication channel itself or from a source external to the channel. It is the primary source of error in digital data communication.

1.3 Examples: Hard- and Soft-guided Channels

The characteristics and quality of data transmission are determined both by the property of the signal and the property of the medium. Transmission media could be classified as *guided (hard-guided)* or *unguided (soft-guided).* With the guided media the waves are guided along a physical path; examples are twisted pair, coaxial cable, and optical fiber. Unguided media offer means for transmitting electromagnetic waves but do not guide them; examples are propagation through air, vacuum, and seawater. In the case of guided media, the medium itself is more important in determining the limitation of transmission. Whereas for unguided media the spectrum or frequency band of the signal produced by the transmitting antenna is more important than the medium in determining transmission characteristics [23].

The following is a brief characteristics overview of six types of transmission media; three examples for guided media (twisted pair, coaxial cable, and optical fiber) and three for unguided media (radio, terrestrial microwave, and satellite microwave). Most of this section's contents are obtained from [23,18].

1. **Twisted Pair**

- A pair of insulated metallic wires (e.g. copper) which are twisted to minimize the electromagnetic (low-frequencies) interference between pairs.
- The media is quite susceptible to interference and noise because of its easy coupling with electromagnetic fields. Impulse noise also easily intrudes into twisted pair. Shielding the wire with metallic braid or sheathing reduces interfaces, and twisting itself reduces the susceptbility.
- Used for transmitting analog and digital signals.
- A typical bandwidth of approximately 250 *kHz* is possible.
- Attenuation is about 1 *dB/km* (at 0 *Hz* and it becomes great at high frequencies, see Figure 1.4.
- Repeaters (or amplifiers) are necessary every 2 to 10 *km*.
- Data rates reach up to 4 *Mbps*.
- 12 to 24 voice channels can be carried.
- A multiple wiring (i.e. a number of twisted pairs are packed into one cable) increases crosstalk. The use of different twist lengths in adjacent pairs and a balance transmission line reduces crosstalk.
- An average propagation time is about 10 $\mu sec/km$. The velocity propagation of a signal differs widely with frequency (i.e. the lower frequencies will

arrive later than the higher frequencies). The longer the line, the grater the delay will be, and so the greater the distortion.

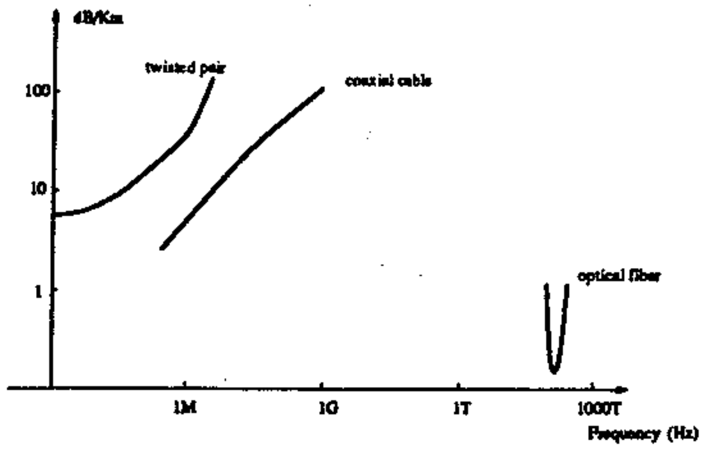

Figure 1.4: Attenuation of guided media

2. **Coaxial Cable**

- It consists of a hollow outer cylindrical conductor which surrounds a single inner wire conductor. The space between the cylindrical shell and the inner conductor is filled with an insulator (e.g. plastic or air).
- It can transmit much higher frequencies and operate in wider ranger of frequencies than twisted pair.
- Much less sensitive to interference and crosstalk than twisted pair.
- A coaxial tube can carry 3,600 to 10,800 voice channels.
- Attenuation does not become severe until very high frequencies Figure 1.4.
- It has more preferable frequency characteristics than twisted pair, i.e. it can be used effectively at higher frequencies and data rates.
- Digital and analog signals can be transmitted.
- In case for analog signaling, bandwidth is about 400 *MHz*.
- In case for digital signaling, data rates are up to 800 *Mbps*.

- Repeaters (or amplifiers) are spaced every 1 to 10 *km*.
- The velocity propagation is approximately equal to the light velocity (for frequencies above 4 *kHz*). It varies slightly with frequency, hence giving very little delay distortion.

3. **Optical Fiber**

   - It is a thin (2 to 125 μm), made of glasses or plastics, flexible medium. It consists of one ore more fibers, each is surrounded by its own cladding, which are grouped together in a jacket.
   - It transmits a ray of light by means of internal reflection. Two types of light sources are used: a light emitting diode and an injection laser diode.
   - It has grater bandwidth (than coaxial cable or twisted pair) around 2 *GHz*, and higher data rates upto 2 *Gbps*.
   - Only analog signals are transmitted.
   - It has lower attenuation, 0.2 to 2 *dB/km*, see Figure 1.4.
   - Not effected by interference, crosstalk, and impulse noise.
   - Repeater spacing is greater (100 *km*).
   - Light propagation is about the light speed.

4. **Radio**

   - Radio is omnidirectional.
   - Covers VHF and part of UHF bands: 30 *MHz* to 1 *GHz*. It is highly effective for broadcast communications.
   - Less sensitive to attenuation from rainfall, comparing to microwave. Radio waves have less attenuation because of longer wavelength. The amount of attenuation as a result of the distance follows

$$Attenuation = 10\ log\ (4\pi d/\lambda)^2\ \ dB, \tag{1.21}$$

where, d is the distance between the transmitter and receiver, and λ is the wavelength.

- The main source of impairment is multipath interference which is the result of reflections from land, water, and human-made objects.
- Maximum distance, d, between the transmitter and receiver anttenas follows the following expression:

$$d = 7.14\sqrt{kh}\ \ km, \tag{1.22}$$

where h is the antenna height in meters, and k is an adjustment constant (~ 0.75).

- Data rate is low in order of *kbps*.
- Bandwidth is about 100 *KHz*.

5. **Terrestrial Microwave**

- The antenna is fixed rigidly and focuses a narrow beam to achieve line-of-sight transmission to the receiving antenna.
- Relay towers are usually spaced 10 to 100 *km* apart, and amplifiers are at each relay point. Hence, fewer amplifiers are used than for coaxial cable with the same distance.
- The maximum distance between antennas, d, follows the expression given in equation 1.22.
- Its transmission covers a frequency rang of 2 to 40 *GHz*. The most common bands for typical carrier long-haul communications are the 4 and 6 *GHz* bands.

- It has a bandwidth about 7 to 220 *MHz*, and data rates around 12 to 274 *MHz*.
- The primary use is in long-haul telecommunication services.
- Attenuation is the prime source of loss. It follows the formula given in equation 1.21. The loss correlates with the square of the distance, not as the twisted pair or coaxial cable which are linear. Hence, repeaters (or amplifier) may placed farther apart for microwave systems.
- Microwave radio is scattered by hills and other objects. Weather (temperatures and humidity) can cause fading. Finally, the main problem with microwave is radio interference.

6. **Satellite Microwave**

- A communication satellite serves as a form of microwave relay over large distances.
- It appears stationary above the Equator at height approximately 36,000 *km*.
- To avoid interference with other satellites, 4° or 3° spacing is required in the 4/6 or (12/14) *GHz* band, respectively.
- Earth stations are located outside cities (80 *km* away) to avoid interferences with terrestrial microwave links.
- Propagation delay (one way) is about 240 to 300 *ms* between the two stations in the earth via the satellite.
- It is the leading medium for high-usage international trunks.
- The optimum frequency range is about 1 to 10 *GHz*.
- Bandwidth is 500 *MHz*.

1.4 Modelling

In the following, a review of various mobile radio channel models is briefly presented. The models are at very high level of the channel models. They have been used to evaluate different data flow control procedures of the data link layer (e.g. ARQ). Most of them consider the channel as an on/off channel, called threshold models (see Figure 1.5). That is, at signal strengths above a certain threshold communication is successful. Also, at signal strengths below this threshold, it is assumed unsuccessful.

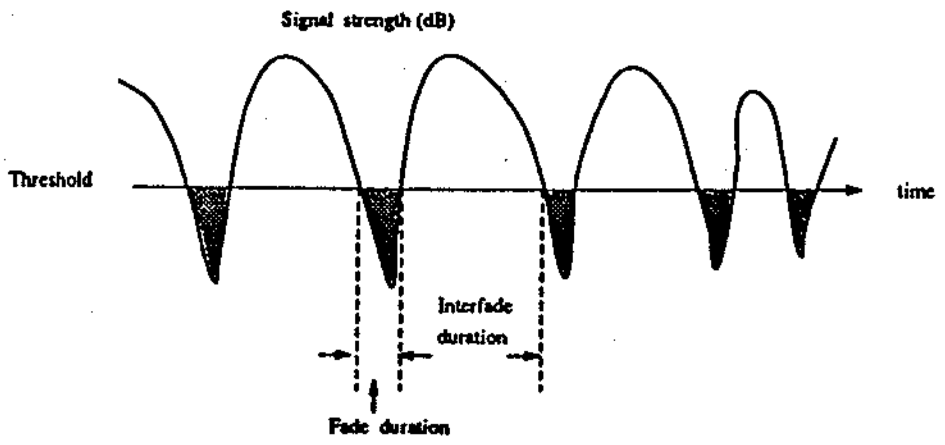

Figure 1.5: Channel threshold model

Karim [11] presented an analytical approach to compute the upper bound of the probability of transmission failures in term of fade statistics over a Rayleigh fading channel. A Rayleigh fading hardware simulator was used to obtain the distribution of the number of fades in an interval of time. He stated that the probability P_e of the transmission failure is given by

$$P_e = \sum_{n=1}^{\infty} P(n) \times P_f(n),$$

where $P(n)$ is the probability that there are n fades in the interval that a complete message sequence occupies and $P_f(n)$ is the probability that the transmission scheme fails assuming n fades. Karim in his other paper [12] used a computer simulation of the Rayleigh fading channel. It was found that the total number of error-free packets depends on the expected value of the interfade interval (D_{av}). Then, the transmission efficiency (η) was given by

$$\eta \leq \frac{1}{(1 + d_{av}/D_{av})^2},$$

where d_{av} is the average fade duration.

Likewise, Siew and Goodman [22] used the fade and interfade duration distributions and presented a mathematical model for the fading mobile radio channel characteristics to develop formulas for data link throughput and delay. They showed that the probability of successful transmission (s) depends on the packet duration (p), the carrier wavelength, the speed of the mobile, and the fade margin. They defined the transmission efficiency as

$$\eta = r(1 - \frac{a}{a + u}),$$

where a is the additional information, u is the user information, and r is the probability of a new packet is generated in each p-second timeslot ($\sim s$).

In 1984, Comroe and Costello [4] presented their work which was based on the threshold model of a land mobile radio channel with the multipath property. They stated that the probability of successful communication (P_s) can be identified as the probability that the signal (strength) is above threshold, i.e.

$$P_s = 1 - F(r) = e^{-0.963r^2},$$

where r is the threshold to median signal ratio and $F(r)$ is the probability distribution of r. In more desirable outcome, they gave the probability of failure (P_f) for a frame (data block) of certain length,

$$P_f = 1 - e^{-0.693r^2} e^{-f_r L},$$

where L is the frame length in time units and f_r is the fading rate which depends on the Doppler frequency f_D ($= V/\lambda$; for a vehicle speed V and carrier wavelength λ).

Finally, Chuang [3] and Chang [2] in separate papers simulated and estimated, respectively, the throughput of a multipath channel using fade- and interfade-duration statistics. In the simulation model given by Chuang, frame errors generated by comparing a random number with the frame-error ratio (P_f) given by

$$P_f = 1 - (1 - P_e(t))^{L-1} - (P_e(t))^L,$$

where P_e is the bit-error ratio and L is the frame length in bits. The frame is erroneous if the generated random number is smaller than (P_f). On the other hand, Chang used the threshold model to estimate the throughput. The probability (P_s) that the transmission is successful was given by

$$P_s = \sum_{t>0} P_i(t) P_I(f_D, t),$$

where $P_i(t)$ is the probability that the entire frame is transmitted within an interfade of duration t, and $P_I(f_D, t)$ is the probability that the interfade duration, at Doppler frequency f_D, is t.

All the previous models considered the Rayleigh fading, but not the shadow fading cased, for example, by hand-off in cellular communications.

Chapter 2

Data Link Control Protocols

While the physical layer provides only a raw bit stream service, the *data link layer* tries to make the physical link reliable. The main objective of this layer is to offer a reliable means to control the flow of data on a physical link. The fundamental service provided by the link layer to the higher layers is that of error control. Thus, with a satisfactorily functional data link layer protocol, the next higher layer may assume virtually error-free transmission over the link [23,9]. Although the data link protocol can be used in multipoint links, only a point-to-point link is considered in this report.

2.1 Basic Properties

In this section some of the basic properties of data link protocols are discussed. For instance, message framing, flow control, windowing, error detection, and error recovery.

2.1.1 Message Framing Schemes

Data link protocols may be divided into three types base on the message framing schemes used. Those are character oriented, byte count oriented, and bit oriented protocols. A character oriented protocol uses certain characters, such as STX to identify the beginning of a message, and ETB to identify the end of a block of data. The block of data is manipulated as a sequence of characters (usually 8-bit characters). IBM's Binary Synchronous Protocol, known as BISYNC, represents a good example of a character oriented protocol [23,19].

Byte count oriented protocols use header which includes, beside other control information, a beginning special character followed by a count that determines how many characters follow in the data field of the message. As in character oriented protocols, the block of data is considered as a sequence of characters. DEC's Digital Data Communication Message Protocol (DDCMP) is an example of this type of protocol [19].

In a bit oriented protocol, the block of data is not viewed as a sequence of character, rather a sequence of bits. A special bit pattern (flag), such as 01111110, signals the beginning of the block. Bit stuffing is used when its needed to differentiate between the actual data and the flag. This type of protocol is widely used today in modern data link protocols. For example, HDLC, IBM's SDLC, and LAP-B [23,19].

2.1.2 Duplexity

The *duplexity* of a link refers to the direction and timing of signal flow. A *half-duplex (HDX)* link, often is known as *either-way* link, can transmit and receive, but not at the same time. On a *full-duplex (FDX)* link, known as *both-way* link, two stations can simultaneously send and receive data from each other [9,23].

Generally, the data is divided into blocks (called *frames*) which are transmitted one after the other. After each frame or a group of frames is transmitted, the receiver sends an acknowledgement. A *positive* acknowledgement (ACK) is sent if the frame was received without any error being detected. Otherwise, a *positive* acknowledgement (NAK) is sent [9].

In the case of a half-duplex system, the sender waits for an acknowledgment from the receiver before transmitting the next frame. Subsequent to receiving ACK, the sender is free to continue with the next transmission, but if NAK is received, the sender retransmits the last frame. Though, typically, there is a limited number of retransmission of a frame [9].

In full-duplex system, frames are numbered and can be sent one after the other without waiting for an acknowledgement. If a frame is received with an error, the receiver sends NAK. Subsequently, the sender either transmit the erroneous frame plus all frames that had transmitted since the erroneous frame, or only the erroneous frame, depending on what an ARQ scheme is used (see Section 2.1.4.2) [9].

2.1.3 Flow Control Techniques

Flow control sequences manage the flow of information (frames) across the data link. They allow a receiver to supervise the quantity and speed of information flowing into its system so as to avoid deadlock or overflow its capacity to accept and process the incoming data [7].

One form of flow control, known as *stop and wait*, acts as follows: The receiver signals its willingness to accept data by sending a poll or responding to a select. The sender then transmits its data. Following reception, the receiver must again signals its willingness to accept data before more are sent [23].

Another widely used form of flow control is known as *sliding window* protocol (which is used with error recovery techniques such as Go-Back-N and selective-repeat ARQs; see Section 2.1.4.2) and where multiple frames are allowed to be in transit at one time. The nature of all sliding window protocols is that at any moment of time, the sender maintains a list of consecutive sequence numbers (*sending window*) corresponding to frames it is allowed to send. Equivalently, the receiver also maintains a *receiving window* corresponding to frames it is permitted to accept. The sending window and the receiving window require not be of the same size. Note that a receiving window of size 1 means that the receiver only accepts frames in order [26].

2.1.4 Error Control

Error control is divided into two basic strategies: *error detection* and *error correction.* The former is the process of determining whether a received frame has an error. Whereas the latter is the process to fix this error. The error detection is discussed in Section 2.1.4.1, and error correction is discussed in Section 2.1.4.2.

The error-detecting and error-correcting characteristics of a code relys on its Hamming distance, which is the minimum number of bit positions in which any two code-words differ. Thus, to detect d errors, a Hamming distance $d+1$ code is required so that there is no way that d single-bit errors can alter a valid code-word into another legitimate code-word. Analogously, to correct d errors, a Hamming distance $2d+1$ code is needed due to the fact that even with d changes, the valid code-word can be deduced from the changed (invalid) code-word [26].

2.1.4.1 Error Detection

Error detection in data communication systems influences the use of redundancy. The more redundancy, the more reliable the technique of error detection, and the less channel throughput. Therefore, most error detection systems are a compromise between the size of redundancy needed and the percentage of error detected [9].

Most of the error detection methods perform on the following manner: For a given frame of bits, additional bits that incorporates an error-detecting code (widely know as frame check sequence; FCS) are appended by the sender. The receiver does the same calculation and compares the two outcomes. The frame is declared valid if they match, otherwise, it is declared erroneous and corrective action can be taken (see Section 2.1.4.2). A number of particular techniques follow this general rule. For example,

1. **Parity Bit:**

   It is the simplest bit error detection which appends a parity bit to the end of each character (word) in the frame. A good example is the ASCII code, in which a parity bit is added to each 7-bit ASCII character. The parity bit can be either odd or even, and its value is chosen so that the encoded character has an odd or even number of 1's respectively. The parity checking routine is not powerful because it can only detect changes in an odd number of bits [23,9,26].

2. **Two-Coordinate Parity Checking:**

   A significant enhancement can be accomplished by using a second dimension (vertical) of parity bits. Here, the frame is regarded as a block of characters organized into two dimensions. Each character has its own (horizontal) parity bit. Moreover, at the end of the data block (the information field) a block check character (BCC) is generated, in which the *ith* bit of BCC is a parity check for the *ith* of all other characters in the block. Hence, any bit in this block of data has two parity checks being performed on it. These calculations (horizontal and vertical) act as complementary checks on each other to enhance the over all error detection capability. A single bit error can be easily detected and located by BCC and may be corrected, two or any odd number of errors can be detected but not located. Yet, some patterns of even number of errors survive undetected. Additionally, in the case of burst error, BCC can detect a single burst of length as much as the character (code-word) length. This scheme is useful for character-oriented transmission systems, but it incorporates a fair amount of overhead in the form of one bit for each character plus one extra character (the BCC) at the end of the block [23,9].

3. **Cyclic Redundancy Check (CRC):**

   CRC is becoming highly used, particularly in bit-oriented data link protocols. It is based upon handling a bit string (frame) as a single binary number representing a polynomial with binary coefficients. This polynomial will be divided (modulo 2 division) by another (generator) polynomial, and the remainder is appended to the data block as CRC. The transmitter and receiver must agree upon a generator polynomial. Most CRCs are 16 bits long, however, 32-bit CRCs are becoming widely used particularly as an option in various point-to-point data link protocols [23,9]. For example, here are some examples of the 16-bit generator polynomials:

CRC-16:	$x^{16} + x^{15} + x^1 + 1$
CRC-CCITT:	$x^{16} + x^{12} + x^5 + 1$

The 16-bit CRC detects all single and double errors, all errors with an odd number of bits, all burst errors of length 16 or less, 99.997% of 17-bit error bursts, and 99.998% of 18-bit and longer bursts [26].

2.1.4.2 Error Correction

After an error is detected, it is better to perform certain action that will attempt to fix the error. There are two main procedures which have been used widely and they may sometimes exist together jointly. Those are *forward error correction (FEC)* and *automatic repeat request (ARQ)*.

Forward Error Correction

Besides the error-detection codes, there are as well error-correction codes, known as *forward error correction*, which contain enough redundant information to correct detected errors at the receiver. Even though FEC is infrequently used in data transmission because of its high overhead, it is used in some cases where retransmission is inefficient or impossible; for example, in one-way transmission to mobile receivers or data broadcasting techniques. In some cases, to obtain a satisfactory degree of error correction, the size of the code have to be approximately as much as the size of data. Hence, decreasing the effective data rate to 50% [23,9].

Automatic Repeat Request

Retransmission is by far the most common means of error correction. It is mainly based on two functions: Error detection, such as CRC (see Section 2.1.4.1), and Automatic Repeat Request (ARQ). When an error is detected, the receiver requests the transmitter that the frame be retransmitted [23,9]. There are three types of ARQ [23]:

1. **Stop-and-wait ARQ**

   It uses the simple stop and wait acknowledgement scheme (no windowing), see Section 2.1.3. The transmitter sends a single frame and then must await an acknowledgement. No other data frames can be sent until the receiver acknowledges this frame. The receiver sends a positive acknowledgement (ACK) if the frame is correct and a negative acknowledgement (NAK) otherwise. If no recognizable acknowledgement is received before the frame *time out* expires, then the same frame is sent again. Note that this technique needs that a copy of the transmitted frame be kept until an ACK is received for that frame. Frames are alternately numbered with 0 or 1 and positive acknowledgements too. The primary advantage of stop-and-wait ARQ is its simplicity, and the primary disadvantage is that this is an inefficient protocol for large propagation time compared to transmission time.

2. **Go-Back-N**

   One variant of continuous ARQ is known as *Go-Back-N ARQ.* In this technique, a transmitter may send a sequence of frames limited by window size. If the receiver detects an error in a frame, it sends NAK for that frame, and discards all subsequent incoming frames until the frame in error is correctly received. Therefore, on receiving NAK, the transmitter must retransmit the frame in error plus all succeeding frames. With Go-Back-N ARQ, it is not essential that each single frame be acknowledged. Windowing scheme is used with a window of size $2^n - 1$ for 2^n sequence number.

3. **Selective-repeat ARQ**

   This technique provides a more efficient approach than Go-Back-N. The only frames retransmitted are those that receive NAK. On the other hand, the receiver must have storage to save post-NAK frames until the error frame is transmitted, and the logic for reinserting the frame in the proper order. The transmitter, too, will require more complex logic to be able to send frames out of sequence. Because of such complications, the Go-Back-N approach is commonly used. The window-size requirement is more restrictive for selective-repeat than for go-back-N. For selective-repeat ARQ, the maximum window size should be no more than half the range of sequence numbers so that there is no overlap between the sending and receiving windows.

2.2 Examples with Rules

2.2.1 HDLC: High-level Data Link Control

This section provides an overview (Section 2.2.1.1) and an operational (HDX and FDX) examples (Section 2.2.1.2) of HDLC.

2.2.1.1 An Overview

High-level data link control (HDLC) can be used on point-to-point or multidrop lines. In the following discussion, the normal-mode (primary/secondary relationship) of HDLC will be examined for a point-to-point link.

HDLC is a bit-oriented data link protocol which transmits pure binary data capsulated within a frame. The general frame layout is shown in Figure 2.1. The different

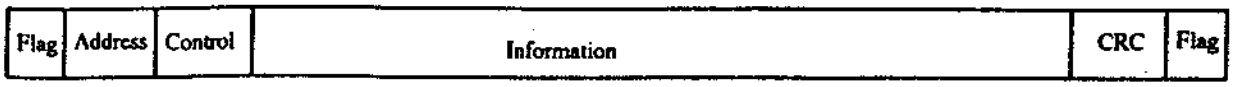

Figure 2.1: An HDLC frame format

fields that make up the frame are as follows [23,9,26]:

- **Flag Fields:**

  Flag fields delimit the frame at both ends with a unique eight-bit pattern 01111110. Additionally, *bit stuffing* is used.

- **Address Field:**

  The address field is an expandable eight-bit field to indicate the destination (secondary) station. It is usually used in the multidrop lines mode, but always present in the frame.

- **Control Filed:**

  The control field is either 8 or 16-bit field depending in the window size. The contents of this field determine the frame type, see Figure 2.2. There are three type of frames:

  1. *Information (I-frame):* The information frame contains the data to be transmitted. The control field of the I-frame contains the sequence number ($N(S)$) of the transmitted frame, the *piggybacked* positive acknowledgment ($N(R)$), and the *poll/final (P/F)* bit. This bit is used as poll by the

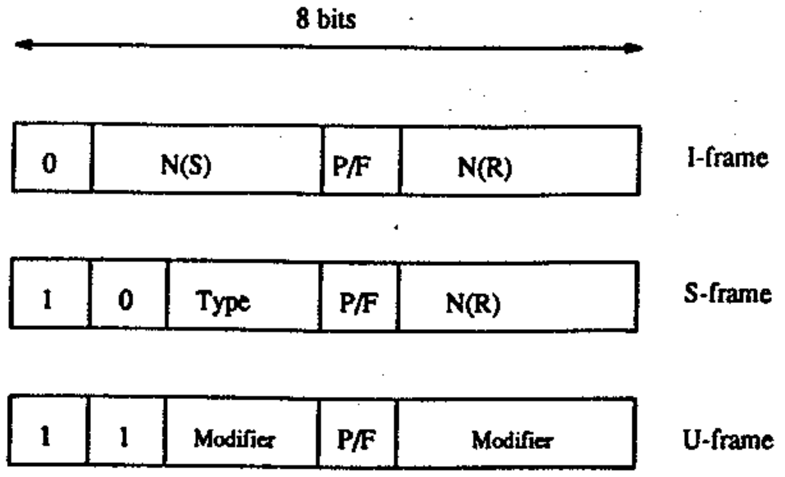

Figure 2.2: Control field

primary station and as a final by secondary station. The $(NS)/N(R)$ sequence numbers can range from 0 to 7 (8-bit control field), or 0 to 127 (16-bit control field).

2. *Supervisory (S-frame):* It is used for flow and error control. There are four kinds of S-frames determined by the two-bit *type* field in the control field:
   - *Receive Ready (RR)* This is a positive acknowledgement frame with the expected next frame number equals $N(R)$.
   - *Reject (REJ)* It is a negative acknowledgement frame requesting retransmission of all input frames starting with the frame number $N(R)$.
   - *Receive Not Ready (RNR)* The RNR frame is like the RR frame, but it requests the sender to stop sending until a subsequent S-frame.
   - *Selective Reject (SREJ)* It is a negative acknowledgement frame which requires retransmission for only the specified frame number $N(R)$.
3. *Unnumbered (U-frame):* It is used for various number of control commands such as HDLC mode setting.

- **Information Filed:**

The information field is used by I-frames and some other U-frames. Its length is variable, and practically is limited by the implementation. Further, it can contain any bit pattern.

- **Frame Check Sequence Field:** The cyclic redundancy check (CRC) is used, see Section 2.1.4.2.

2.2.1.2 HDLC–Operational Examples

The following examples illustrate a number of typical (one-way data transmission) message exchanges using HDLC. Note that, HDLC provides group acknowledgement which enhances the transmission efficiency, particularly in HDX mode. They are taken from Housley's book [9], and presented in form of diagrams. In those diagram, messages are symbolized as follows:

type,N(S),N(R),P/F

Where *type* will be I for I-frame or RR, REJ, RNR, or SREJ for the corresponding S-frame. *N(S)* and *N(R)* are the send and receive sequence numbers. *P/F* indicates if a poll or final bit is sent. Note that initialization and termination phases of the link is not included in these examples.

1. **HDX - error free data transfer**

   In this example (see Figure 2.3.(a), a row of four frames are sent before an acknowledgement is needed. Here, the one-way delay is 1/3 of the information frame transmission time.

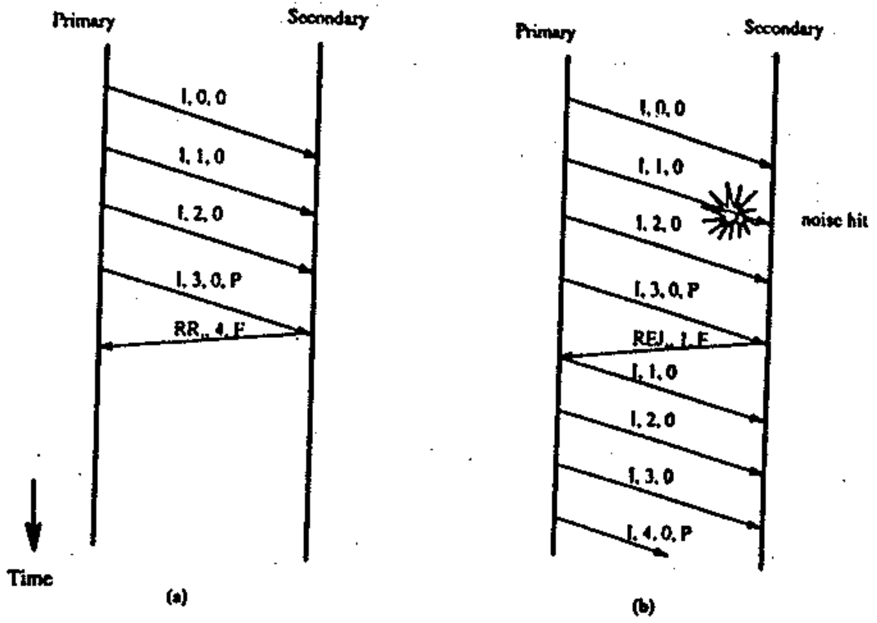

Figure 2.3: HDX data transfer: (a) error free, and (b) erroneous.

2. **HDX - erroneous data transfer**

   Assume that, for the four frames in example 1 above after they have been sent, frame 1 was corrupted. The secondary (receiver) will send the REJ-frame (REJ,,1,F) which identifies the last successful received frame was 0. Subsequently, the primary (sender) retransmits frames starting from frame number 1, see Figure 2.3.(b).

3. **FDX - error free data transfer**

   Figure 2.4.(a) depicts one-way data transfer over a error-free FDX line. Observe that the primary remains sending data after the poll (bit) has been transmitted.

4. **FDX - erroneous data transfer**

   In Figure 2.4.(b) illustrates erroneous data transmission. Frame 1 has been corrupted by a noise hit. When the poll is received, the secondary indicates that the next (in-sequence) frame it wishes to receive is frame 1. This makes

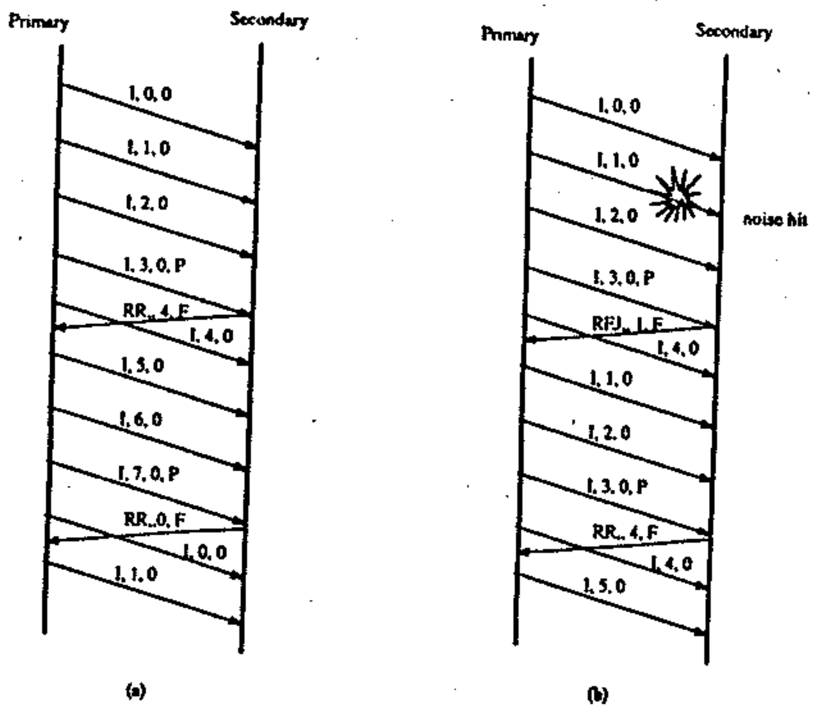

Figure 2.4: FDX data transfer: (a) error free, and (b) erroneous.

the primary retransmit frames starting from frame number 1.

2.2.2 Frame Relay

Frame relay offers a streamlined method for wide-area packet switching. Even though the emphasis in this report is placed on transmitting frames over point-to-point links, in this section the emphasis will be placed on transmitting frames over packet-switched network, where frames are transmitted over multiple hops.

In frame relaying, end-to-end flow and error control are the duty of a higher layer than data link layer. It is based in the impression that the recent digital facilities are very reliable. Hence, link-by-link flow and error control is not provided.

Figure 2.5 depicts the difference in transmitting a single data packet from the source to the destination and receiving back its acknowledgement packet using the X.25 (employs HDLC-like) and frame relay.

In the X.25 case, at every stage (link) of the frame's trip from the source to the destination, the data-link control protocol performs the transaction of data frame and

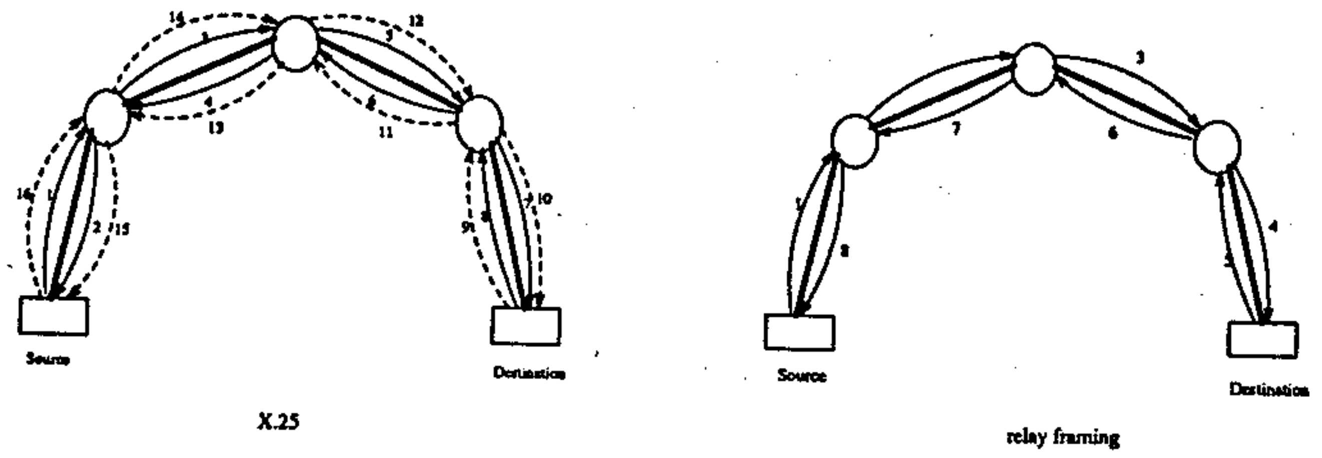

Figure 2.5: Frame relay vs. X.25

an acknowledgement frame. While in the case of frame relaying, one data frame is transmitted from the source to the destination, and an acknowledgement (which is generated at a higher layer) is sent back in a frame. Therefore, frame relay requires half the number of transmissions needed by X.25 to complete the transfer of the same data packet.

The benefit of frame relaying is that the communications process is streamlined. Though, the basic disadvantage of frame relaying is that there is no link-to-link flow and error control. This becomes critical if some of the data links are not reliable enough, so that many frames are retransmitted repeatedly between the two far ends (the source and destination). For more information see [24].

2.2.3 File Transfer Protocols

In a communications network, files are transferred from one station to another. A file transfer facility fits in the application layer of ISO-OSI model, which provides the interface to the user. Ideally, in file transmission, all the levels of OSI model at each end (source and destination) are participating somehow. However, levels 4 (transport) and 2 (data link) of OSI model are the major ruler in the file transfer. The transport layer provides a successful transmission of files by supporting blocking and block sequence numbers. Whereas the data link layer provides a successful transmission of blocks by supporting error recovery and data flow control.

Examples of file transfer protocols are Kermit, Xcopy, and FTP-DAP.

Chapter 3

Performance Measures

3.1 Overhead–Static

In case of physical and data link layers, *overhead* is the additional information (bits) as a result of delimiters, bit stuffing, synchronization, frame check sequence, control information, etc. Overhead can be defined as either the ratio between additional bits and user bits (i.e. the information field in a frame), or the ratio between addition information and the summation of user and additional information [9,23,15]. The former is considered in this chapter. That is, the overhead (h) is defined as

$$\begin{aligned} h &= \frac{additional\ \ bits}{user\ \ bits} \\ &= \frac{a}{u} \end{aligned} \tag{3.1}$$

Subsequently, the *bit efficiency* which is the ratio of the user information bits to the transmitted bits. Thus, the bit efficiency (η_b) can be expressed as

$$\begin{aligned} \eta_b &= \frac{u}{u+a}, \\ &= \frac{1}{1+h} \end{aligned} \tag{3.2}$$

3.2 Efficiency–Dynamic

In measuring the performance of a communications link while transmitting a message, different time delays should be considered. The *loop-back* (or just *loop*) delay is an easy to measure at one end (station) of the link. It is defined as the sum of the round-trip delays encountered during the transmission from one end of the link to the other and back [9].

In addition to the propagation delay (T_p), the loop delay (T_d) may include station (receive or transmit) reaction time, receiver delay, user reaction time, and ,in case of HDX link, request-to-send and clear-to-send delay (T_{rc}). In the examples to follow, for the simplicity, the loop delay is approximately defined as

$$T_d = 2(T_p + T_{rc}). \tag{3.3}$$

Let T_f and T_a denote the time needed to transmit a frame and its acknowledgement, respectively. Therefore, the total time (T_x) required to send a frame from one station and receive its acknowledgement from the other can be expressed as

$$T_x = T_f + T_a + T_d \tag{3.4}$$

Note that here T_d is an overhead. Thus, the throughput efficiency η_x is given by

$$\begin{aligned} \eta_x &= \frac{T_f + T_a}{T_f + T_a + T_d} && (3.5) \\ &\overset{T_a \ll T_f}{\approx} \frac{T_f}{T_f + T_d} && (3.6) \end{aligned}$$

Hence, the larger the frame the greater the throughput efficiency [15].

In the case of *group acknowledgements*, it is possible that the throughput efficiency reaches 100 %. To see that, consider the following assumptions for the next throughput efficiency analysis of group acknowledgements:

1. error-free FDX link,

2. one-way I-frame transmission,

3. no gaps between transmitted I-frames,

4. go-back-N ARQ with RR and REJ,

5. group of n frames immediately acknowledged by an S-frame.

Let T_c be the cycle response time which is the time between sending the first frame of the group until receiving their (group) acknowledgement, and it can be given by

$$\begin{aligned} T_c &= (nT_f + T_d/2) + (T_a + T_d/2) \\ &\overset{T_a \ll T_f}{\approx} nT_f + T_d. \end{aligned} \tag{3.7}$$

To maintain a 100% efficiency for FDX operation, the following condition must be satisfied:

$$T_c \leq NT_f, \tag{3.8}$$

where N is the ARQ window size. Substituting equation 3.7 into 3.8 gives

$$n \leq N - T_d/T_f \tag{3.9}$$

Thus, the throughput efficiency for the group acknowledgements can be expressed as

$$\eta_x(n) = \begin{cases} 1 & \text{if} n \leq N - T_d/T_f \\ NT_f/T_c & \text{otherwise.} \end{cases} \tag{3.10}$$

Henceforth, the FDX operation is maintained so long as the window size (N) is large, the loop delay (T_d) is small, group size (n) is small, or frame length (T_f) is large. Figure 3.1 shows the maximum throughput efficiency achievable for window size 7 and 127 as a function of frame length ($u + a$).

3.3 Error Recovery Efficiency

In this section, a very simple error analysis [15] is presented based on the stop and wait ARQ protocol 2.1.4.2. Prior to start the analysis, there are some assumptions that have been made to simplify the analysis. They are as follows:

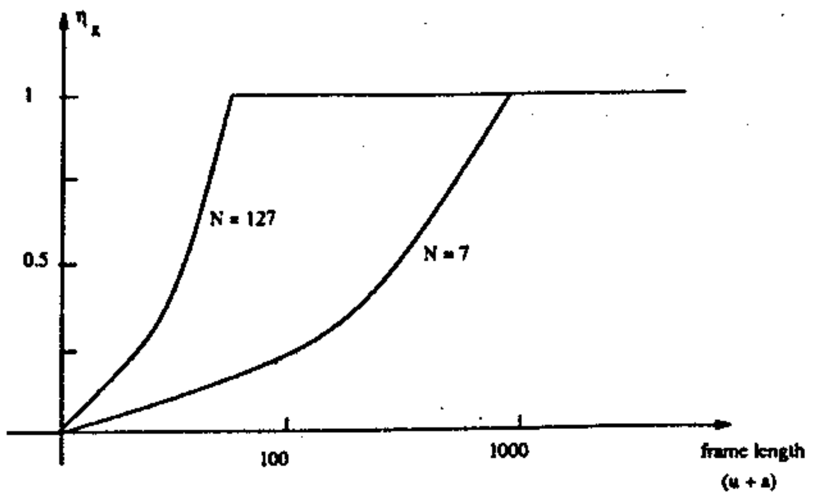

Figure 3.1: Throughput efficiency vs frame length (group acknowledgements)

1. Errors come in burst.
2. The time between error bursts (denoted T_e) are equal.
3. An error burst time period (Δe) is very small compared with T_e.
4. Each burst affect only one frame (i.e. $\Delta \ll T_f$).
5. The acknowledgement length is very small compared to the frame length; i.e. $T_a \ll T_f$.
6. The stop and wait ARQ protocol is used.

Let T_t denotes the total time required to send n frames and is defined as

$$T_t = T_c + T_r, \tag{3.11}$$

where, $T_c = nT_f$ is the time required to transmit the n frames without errors, and $T_r = n_r T_f$ is the time required to retransmit the erroneous n_r frames. Therefore, the total number of transmitted frames equals to $n + n_r$.

Since each error burst effects only one frame then there are T_t/T_e error burst. Thus, T_r can be written as

$$T_r = n_r T_f = (T_t/T_e)T_f$$

The actual time required to transmit the n frames is given by

$$\begin{aligned} T_c &= T_t - T_r \\ &= T_t - (T_t/T_e)T_f \\ &= T_t(1 - T_f/T_e) \end{aligned}$$

Thus, the error recovery efficiency is defined as follows

$$\begin{aligned} \eta_e &= \frac{T_c}{T_t} \\ &= 1 - T_f/T_e \end{aligned} \tag{3.12}$$

3.4 Optimization

Figure 3.2 shows the relationship between the length of the user information and each of the efficiencies discussed in the previous section; bit efficiency, throughput efficiency, and error recovery efficiency.

For the bit efficiency, the larger frame length, the greater its efficiency. It depends on the transmission mode (asynchronous or synchronous) and the link protocol (character or bit oriented). In the case of the throughput efficiency, it increases for large frame length. Also, it depends on the loop delay (e.g. η_x increases for FDX and a medium with a small propagation delay). Whereas the error efficiency decreases as the frame length increases.

One way to find the optimal frame length is to compute the overall efficiency which is the product of the above three efficiencies. That is, the overall efficiency η is given by

$$\eta = \eta_b \times \eta_x \times \eta_e. \tag{3.13}$$

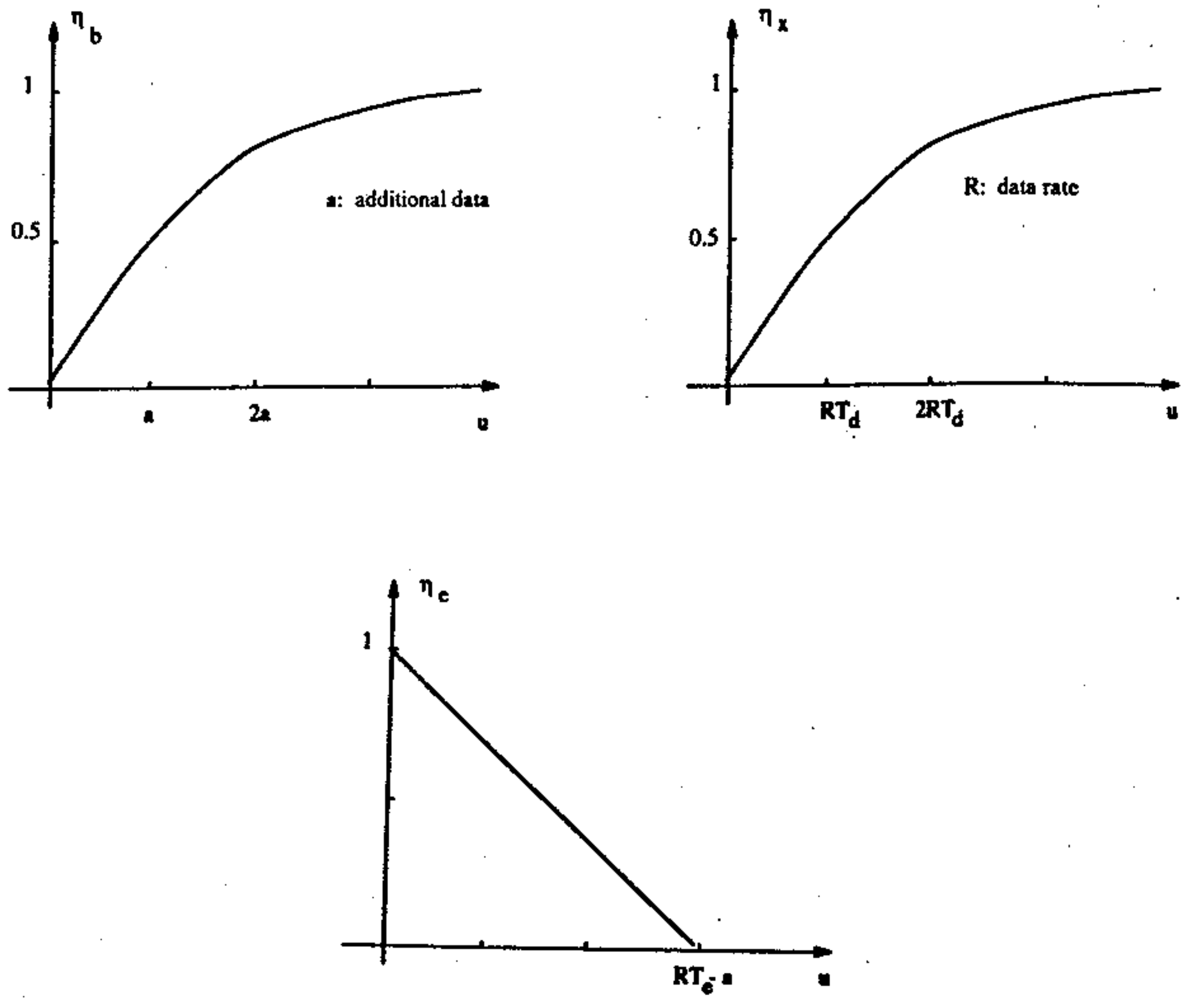

Figure 3.2: Relationship between efficiencies and the frame length

Then, the optimal frame length is obtained at where overall efficiency is maximal, see Figure 3.3.

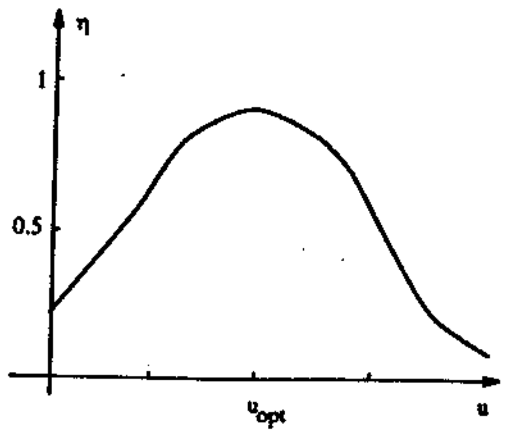

Figure 3.3: Overall efficiency vs frame length

3.5 Examples with Numerical Results

For the subsequent computations, consider the following parameters:

1. A point-to-point line.
2. Line1: satellite, one way propagation delay, $T_p = 250$ *ms.*
3. Line2: terrestrial cable, one way propagation delay, $T_p = 30$ *ms.* (3000 *km* at 10 $\mu sec/km$)
4. Request-to-send/Clear-to-send delay (HDX), $T_{rc} = 250$ *ms.*
5. Frame1: user information field size, $u = 80$ *bits.*
6. Frame2: user information field size, $u = 8000$ *bits.*
7. Line speed or data rate, $R = 4800$ *bps.*

8. HDLC frame consists of, besides user information field, two 8-bit flags, 8-bit address, 8-bit control field, and 16-bit CRC. Bit stuffing is used.

Computations

1. Total additional information: *a bits*

   - Frame1:

$$48 \leq a_1 \leq 70$$

   - Frame2:

$$48 \leq a_2 \leq 1654$$

2. Overhead (static): h [see equation 3.1]

   - Frame1:

$$\begin{aligned} \frac{48}{80} &\leq h_1 \leq \frac{70}{80} \\ 0.6 &\leq h_1 \leq 0.875 \end{aligned}$$

   - Frame2:

$$\begin{aligned} \frac{48}{8000} &\leq h_2 \leq \frac{1654}{8000} \\ 0.006 &\leq h_2 \leq 0.2068 \end{aligned}$$

3. Bit efficiency: η_b [see equation 3.2]

   - Frame1:

$$\begin{aligned} \frac{1}{1+0.875} &\leq \eta_{b1} \leq \frac{1}{1+0.6} \\ 0.5333 &\leq \eta_{b1} \leq 0.625 \end{aligned}$$

- Frame2:

$$\begin{aligned} \frac{1}{1+0.2068} &\leq \eta_{b2} \leq \frac{1}{1+0.006} \\ 0.8286 &\leq \eta_{b2} \leq 0.994 \end{aligned}$$

4. Loop delay: T_d *msec* [see equation 3.3]

- Line1 (HDX):

$$T_{d1} = 2(250+250) = 1000$$

- Line1 (FDX):

$$T_{d1} = 2(250+0) = 500$$

- Line2 (HDX):

$$T_{d2} = 2(30+250) = 560$$

- Line2 (FDX):

$$T_{d2} = 2(30+0) = 60$$

5. Frame and acknowledgement time: T_f and T_a *msec*

- Frame1:

$$\begin{aligned} \frac{80+48}{4.8} &\leq T_{f1} \leq \frac{80+70}{4.8} \\ 26.7 &\leq T_{f1} \leq 31.3 \end{aligned}$$

- Frame2:

$$\begin{aligned} \frac{8000+48}{4.8} &\leq T_{f2} \leq \frac{8000+1654}{4.8} \\ 1676.7 &\leq T_{f2} \leq 2011.25 \end{aligned}$$

- ACK:

$$\begin{aligned} \frac{48}{4.8} &\leq T_a \leq \frac{54}{4.8} \\ 10 &\leq T_a \leq 11.25 \end{aligned}$$

6. Throughput efficiency η_x [see equation 3.5]

- Line1 (HDX):
  - Frame1:

$$\begin{aligned} \frac{26.7+10}{26.7+10+1000} &\leq \eta_{x11} \leq \frac{31.3+11.25}{31.3+11.25+1000} \\ 0.0354 &\leq \eta_{x11} \leq 0.0408 \end{aligned}$$

  - Frame2:

$$\begin{aligned} \frac{1676.7+10}{1676.7+10+1000} &\leq \eta_{x12} \leq \frac{2011.25+11.25}{2011.25+11.25+1000} \\ 0.6278 &\leq \eta_{x12} \leq 0.6691 \end{aligned}$$

- Line1 (FDX):
  - Frame1:

$$\begin{aligned} \frac{26.7+10}{26.7+10+500} &\leq \eta_{x11} \leq \frac{31.3+11.25}{31.3+11.25+500} \\ 0.0684 &\leq \eta_{x11} \leq 0.0784 \end{aligned}$$

  - Frame2:

$$\begin{aligned} \frac{1676.7+10}{1676.7+10+500} &\leq \eta_{x12} \leq \frac{2011.25+11.25}{2011.25+11.25+500} \\ 0.7713 &\leq \eta_{x12} \leq 0.8018 \end{aligned}$$

- Line2 (HDX):
  - Frame1:

$$\begin{aligned} \frac{26.7+10}{26.7+10+560} &\leq \eta_{x21} \leq \frac{31.3+11.25}{31.3+11.25+560} \\ 0.0615 &\leq \eta_{x21} \leq 0.0703 \end{aligned}$$

- Frame2:

$$\begin{aligned} \frac{1676.7+10}{1676.7+10+560} &\leq \eta_{x22} \leq \frac{2011.25+11.25}{2011.25+11.25+560} \\ 0.7507 &\leq \eta_{x22} \leq 0.7831 \end{aligned}$$

- Line2 (FDX):
  - Frame1:

$$\begin{aligned} \frac{26.7+10}{26.7+10+60} &\leq \eta_{x21} \leq \frac{31.3+11.25}{31.3+11.25+60} \\ 0.3795 &\leq \eta_{x21} \leq 0.4149 \end{aligned}$$

  - Frame2:

$$\begin{aligned} \frac{1676.7+10}{1676.7+10+60} &\leq \eta_{x22} \leq \frac{2011.25+11.25}{2011.25+11.25+60} \\ 0.9656 &\leq \eta_{x22} \leq 0.9712 \end{aligned}$$

Part II

Simulation of Communication Protocols

Chapter 4

Probability, Statistics, and Queueing Theory

In this chapter a brief introduction in some concepts of probability, statistics, and queueing theory are presented. Firstly, the following significant terms are defined based on the definations found in [1,29].

Random Experiment:

It is an experiment whose outcome is not certain.

Sample Space:

The sample space of a random experiment is the *set* of all possible simple outcomes of the experiment. It can be classified either discrete (the number of the sample points is finite or countably infinite) or continous (the sample points consist of all the numbers on some finite or infinite interval of the real line).

Event:

An event is simply a collection of certain points, that is, a subset of the sample space.

4.1 Random Variables

In many random expermints, some number associated with the experiment is required rather than the actual output. Therefore, the interest is in a function that assigns a numerical value to each possible outcome of an experiment. This function is known as *a random variable.*

Formaly, a random variable X on a sample space S is a function $X : S \rightarrow R$ that assigns a real number $X(s)$ to each possible outcome $s \in S$ [29].

4.2 Distributions

The significance of the random variable approach depends on the ability to find out the probability that the values of the random variable exist in a given set of real numbers. Therefore, the most important attribute of a random variable X is the *probability distribution*, that is, how the probability associated with values of X is distributed over these values [1].

The Cumulative Distribution Function (CDF) of a random variable maps a given value x to the probability of a variable taking a value less than or equal to x:

$$F_X(x) = P(X \leq x).$$

The derivative $f(x) = dF(x)/dx$ of the CDF $F(x)$ is called the *probability density function* (pdf) of X. Given a pdf $f(x)$, the probability of X being in the interval (x_1, x_2) can also be computed by integration:

$$P(x_1 < X \leq x_2) = F(x_2) - F(x_1) = \int_{x_1}^{x_2} f(x)dx$$

For a discrete random variable, the CDF is not continous and, therfore, not differentiable. In such cases, the *probability mass function* (pmf) is used in place of pdf. Consider a discrete random variable X that can take n distinct values $x_1, x_2, .., x_n$

with probabilties $p_1, p_2, ..., p_n$ such that the probability of the *ith* value x_i is p_i. The pmf maps x_i to p_i:

$$f(x_i) = p_i.$$

4.2.1 Distribution Functions

In this section, a number of distribution functions are briefly surveyed and summarized. Most of them are somehow applicable to the fields of data communications and queueing theory. The contents of this section are mostly extracted from Jain's book [10].

Bernoulli

The Bernoulli disribution is the simplest discrete distribution. A Bernoulli variable can have only two values: success ($x = 1$), with probability p, and failure ($x = 0$), with probability $1 - p$. This distribution is used to model the probability of a binary result; for example, a frame trasferred over a data link reaches or does not reach the destination, or a bit in a packet is altered by noise and arrives in error.

1. **Parameters:** p, the probability of success, $0 \leq p \leq 1$
2. **Range:** $x = 0, 1$
3. **pmf:** $f(x) = \begin{cases} 1-p & if\ x = 0 \\ p & if\ x = 1 \end{cases}$
4. **Mean:** p
5. **Variance:** $p(1-p)$

Binomial

The binomial distribution refers to the number of successes x in a series of n Bernoulli experiments. It is uesd to model the number of successes in a string of n independent and identical Bernoulli trials. For instance, the number of packets that arrives at the distinations without loss, or the number of bits in a packet that are not altered by noise.

1. **Parameters:** p, probability of success in a trial, $0 \leq p \leq 1$
   $n > 0$, number of trials
2. **Range:** $x = 0, 1, \ldots, n$
3. **pdf:** $f(x) = \begin{pmatrix} n \\ x \end{pmatrix} p^x(1-p)^{n-x}$
4. **Mean:** np
5. **Variance:** $np(1-p)$

Geometric

The distribution of number of trials up to and including the first success in a sequence of Bermoulli trials is called a geometric distribution. It is a discrete equivalent of the exponential distribution, and used to model the number of attempts between successive failures (or successes). For instance, the number of packets successfuly transmitted between those requiring a retransmission, or the number of successive error-free bits in a packet received on a noisy link.

1. **Parameters:** p, the probability of success, $0 \leq p \leq 1$
2. **Range:** $x = 1, 2, \ldots, \infty$
3. **pmf:** $f(x) = (1-p)^{x-1}p$
4. **Mean:** $1/p$
5. **Variance:** $(1-p)/p^2$

Negative Binomial

In a sequence of Bernoulli attempts, the number of failures x before the mth success has a negative binomial distribution. For example, the number of retransmission for a message consisting of m packets, or number of error-free bits received on a noisy link before the m in-error bit.

1. **Parameters:** p, probability of success,
   $m > 0$, number of successes
2. **Range:** $x = 0, 1, \ldots, \infty$
3. **pmf:** $f(x) = \begin{pmatrix} m+x-1 \\ m-1 \end{pmatrix} p^m(1-p)^x$
4. **Mean:** $m(1-p)/p$
5. **Variance:** $m(1-p)/p^2$

Poisson

The Poisson distribution is a limiting form of the binomial distribution, and it is used extensively in queueing models. It is used to model the number of arrivals over a given interval. For example, number of requests to a server in a given time interval t, or number of transmitted error-free packets per unit time.

1. **Parameters:** $\lambda > 0$, mean
2. **Range:** $x = 0, 1, \ldots, \infty$
3. **pmf:** $f(x) = P(X = x) = \lambda^x \frac{e^{-\lambda}}{x!}$
4. **Mean:** λ
5. **Variance:** λ

Uniform

This is one of the simplest distribution to use. It is commonly used if a random variable is bounded and no further information is available; for instance, distance between source and destination on a network. In the case of a discrete random variable, a uniform distribution is used when it is believed that the value is equally likely over a bounded interval. For example, the source and destination nodes for the next packet on a network.

1. **Parameters:** a, lower limit (must be an integer if discrete)
   b, upper limit (must be an integer if discrete)
2. **Range:** $a \leq x \leq b$, if continuous
   $a, a+1, a+2, \ldots, b$, if discrete
3. **pdf:** $f(x) = 1/(b-a)$
4. **pmf:** $f(x) = 1/(b-a+1)$
5. **Mean:** $(a+b)/2$
6. **Variance:** $(b-a)^2/12$, if continuous
   $((b-a+1)^2-1)/12$, if discrete

Normal

The normal (Gaussian) distribution is the most important distribution in applied probability and statistics. The normal distribution is used whenever the randomness is caused by several independent sources acting additively. For instance, errors in measurment.

1. **Parameters:** μ, mean
   $\sigma > 0$, standard deviation
2. **Range:** $-\infty \leq x \leq \infty$
3. **pdf:** $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$

4. **Mean:** μ

5. **Variance:** σ

Exponential

The exponential distribution is used widely in queueing models. It is the only continuous distribution with the memoryless property. It is used to model the time between successive events; for example, the time between successive request arrivals to a device.

1. **Parameters:** $a > 0$, mean

2. **Range:** $0 \leq x \leq \infty$

3. **pmf:** $f(x) = (1/a)e^{-x/a}$

4. **Mean:** a

5. **Variance:** a^2

Gamma

The Gamma distribution is a generalization of the Erlang distribution (not included in this survey, for more information see [1]). It is used to model service times of services in queueing network models and for repair times.

1. **Parameters:** $a > 0$, scale parameter
   $b > 0$, shape parameter

2. **Range:** $0 \leq x \leq \infty$

3. **pdf:** $f(x) = \frac{(x/a)^{b-1}e^{-x/a}}{a\Gamma(b)}$

4. **Mean:** ab

5. **Variance:** a^2b

4.3 Basic Queueing Theory Notations and Defi- nations

Queueing theory plays a significant rule in the quantitative comprehension of computer communication networks. For example, the queueing theory aids in finding out how network load characteristics, such as message rates and message length distribution, affect performance.

Figure 4.1 depicts the basic elements of a queueing system. *Customers* from a

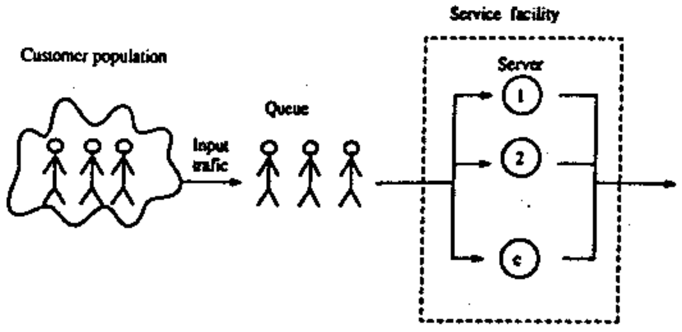

Figure 4.1: A queueing system

population enter a queueing system to receive some type of services [10]. A customer could be any entity that needs a service. For example, a program demanding an input/output service, or a packet that needs to be transmitted. The *server facility* of the queueing system contains one or more servers. A *server* is an object which provides the required service for a customer. If all servers are in use when a customer arrives to the queueing system, the customer must wait in a queue until a server is available [1]. A queueing system is generally specified by the following elements [10]:

1. **Population Size:**

   The total number of customers which can enter the queueing system. It can be finite or infinite.

2. **Arrival Pattern:**

   The "pattern" in which customers arrive to the queueing system is influential. Frequently, it is assumed that customers arrive at times $0 \leq t_0 < t_1 < t_2 < ... < t_n < ...$ The random variables $\tau_i = t_i - t_{i-1}$ are called ***interarrival times.*** It is assumed that the τ_i form a sequence of independent and identical distribution random varibales. The most common arrival pattern is Possion arrivals; when the interarrival time has an exponential distribution.

3. **Service Time Distribution:**

   The service time is the time each customer spends in the system. Usually, it is assumed that the service times are random variables which are independent and of identical distribution.

4. **System Capacity:**

   The maximum number of customers that can be in the queueing system may be limited. In most queueing systems the queue capacity is assumed to be infinite. In other words, every in coming customer is allowed to wait until a service can be given. Some other systems have a finite capacity, for example, a message buffering system.

5. **Number of Servers:**

   A single server system can serve only one customer at a time. However, when the service facility contains more than one identical server, then more than one customer may be served at the same time.

6. **Service Discipline:**

   The pattern in which customers are served is called the service disciplin. The most commom queue discipline is "First Come First Serve" (FCFS). Other queue disciplines include "Last Come First Serve", "Random Selection for Service", and "Priority Service".

In order to define a queueing system, the previous elements are used. A shorthand notation, called the Kendall's notation, in the form $A/B/c/K/m/Z$ is used, where A is the interarrival time distribution, B is the service time distribution, c is the number of servers, K is the system capacity, m is the population size, and Z is the service discipline.

Often, a shorter notation, $A/B/c$ is used for describing a queueing system with an infinite capacity, an infinite population size, and a FCFS service discipline. Furthermore, the distribution of the interarrival time and service time (i.e. A, B) are usually denoted by sysmbols, for example: G (general), M (exponential), E_k (Erlang with parameter k), and D (deterministic) [1,10].

Figure 4.2 illustrates the random variables used in the queueing analaysis. From the customer viewpoint, the queueing system consists of three stages. They are as follows with their variables [1,10]:

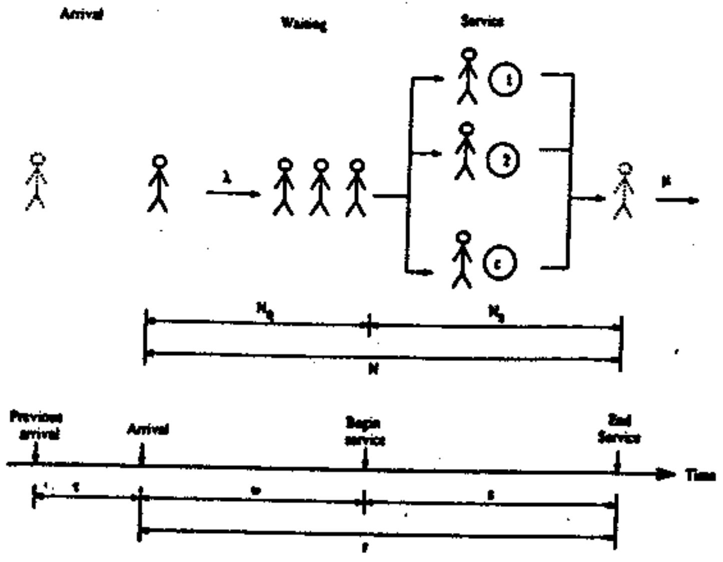

Figure 4.2: Random variables in a queueing system

1. *Arrival Stage*

λ : average arrival rate of customers.

τ : [Random Variable (RV)], interarrival time; i.e. the time between two successive arrivals.

2. *Waiting Stage*

   N_q : [RV], number of customers waiting to receive service.

   w : [RV], waiting time; i.e. the time a customer spends in the queue before receiving service.

3. *Service Stage*

   μ : average service rate per server.

   N_s : [RV], number of customers receiving service.

   s : [RV], service time per server.

The total number (N) of customers in the system equals $N_q + N_s$, and the respons time r (total time a customer spendse in the queueing system) equals $w + s$.

4.3.1 A Simple Queueing System

The simple queuing system has only one queue. It is usually used to analyze individual resources in a system. For example, if all frames waiting to be sent by a data link layer are kept in one queue, then the data link layer can be modeled using a simple queueing system. The state of a simple queuing system can be discribed by the number of the customers n in the system. The arrival or departure of a customer changes the system state to $n + 1$ or $n - 1$, respectively. This is known as a birth-death process. The state transition diagram of birth-death process is depected in Figure 4.3 [10].

4.3.2 Queueing Networks

There are a number of systems which consist of several services (queues). A customer may receive service at one or more queues, before exiting from the system. Such

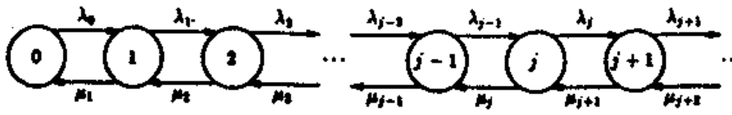

Figure 4.3: State transition diagram of a birth-death process

systems are modeled by queueing networks. The most used classification of queueing networks identifies two queueing networks: *open* or *closed.*

In an open queueing network, customers arrive from external sources and leave the queueing system after receiving service. Since the number of customers in the system varies with time, the goal is to find the distribution of the number of customers in the system. In a closed queueing network, the number of cutomers in the system is constant, and they keep circulating from queue to the next. Here the emphasis is on the system throughput.

4.4 Performance Measures

The following list of performance measures is for the steady-state behaviour of the queueing system [1,13]

- r the average response time.
- w the average waiting time in the queue.
- $\pi_r(90)$ the 90% of waiting time in the system (i.e. 90% of all cusomers spend less than this time in the system).
- $\pi_w(90)$ the 90% of waiting time in the queue.

- $E[N] = \lambda E[r]$ the average number of customers in the system (Littel's law).
- $E[N_q] = \lambda E[w]$ the average number of customers waiting in the queue.
- $p_n = P(N = n)$ the probability that there are n customers in the queueing system.

Chapter 5

Simulation Techniques

Basically, there are three ways to evaluate the performance of a given system. They are:

1. System measurements.

2. Simulation models.

3. Analytical models

The most frequently used approach for the solution of computer system and communications models is the simulation. Simulation is a helpful procedure for computer systems and communications performance analysis. Ferrari [5] defined simulation as "an evaluation technique which represents by a model the behavior of a system in the time domain." Also, it is defined by Biles in Chapter 3 of [13] as "the development of a mathematical-logical model of a system and the experimental manipulation of the model on a computer." In this chapter, topics related to simulation modeling are discussed; for example, simulation modeling approaches, simulation time, workload characterization, simulation languages and systems, experimental design validation, and output analysis.

5.1 Simulation Modeling Approaches

Commonly, there are two kinds of modeling used in (discrete-event) simulation: *the event scheduling approach* and *the process interaction approach.* The former allows the user to structure a system description by focusing on the moments in the time when state changes (events) occur; i.e., concentrating on events and their effect on system state. Whereas the latter requires a user to describe the flow of each entity through the system. Although the process interaction approach traditionally seems simpler than the event scheduling approach (to a model developer) , this simplicity comes at the sacrifice of programming control [6].

In the event scheduling approach, a single flowchart may include all events (state changes) that happen at the same time. An event contains all decision flows and updating that relate to the change in state the execution of the event requires. Since time elapses between events (but not within an event), a simulation can be viewed as the execution of a series of events ordered chronologically on desired execution times [6].

On the other hand, the process interaction approach provides a process for each entity in a system. A process is a time-ordered collection of events, activities, and delays that are somehow related to an entity [13]. The inclusion of time flow within the process interaction flowcharts distinguishes this approach from event scheduling [6].

5.2 Simulation of Time

In the event-scheduling simulation approach, an event queue is used to keep a list of scheduled events. When an event needs to be scheduled, an entry is created with the name of the event and its desired execution time as attributes and inserted into the event queue. The queue is ordered on the event execution time. Subsequent to the selection and removal of the first (header) event from the queue for execution, the *simulation time* is advanced to this event execution time. The technique of variable time advancing neglecting all the intervening time between events is known as *next event technique* [5,6,17].

In some simulators, simulated time is advanced by a small increment. At each advance, all of the current events that call for execution at that time are executed. This method for advancing the simulated time is known as *fixed-step technique* [5,6].

In summary, there are two approaches for advancing the simulated time [10]:

1. *Next event approach*

   The time is advanced automatically to the time of the next earliest occurring event.

2. *Fixed-step approach*

   The time is advanced by a small increment and then any events that can occur are checked.

5.3 Random Numbers

Random numbers are an important factor of simulation models. Most computer languages provide a subroutine that will generate random numbers. To be useful, a sequence of random numbers is required to be uniformly distributed and independent [13].

On a digital computer, the only way to generate *truly* random numbers is to save a table of such size to offer all numbers required during the simulation process. However, this way is not preferable because of the need of very large storage in main memory or many accesses to secondary storage. The substitute is to use a *pseudo-random* number generator which is deterministic; i.e., given the start value (seed), it is possible to predict the numbers in the sequence [10,13].

The preferable characteristics of the pseudo-random generator function are as follows [10,13]:

- It should be fast.
- It should not require large storage in main memory.
- It should have a large period (the length of the sequence until the series begins to repeat itself in the previous order).
- It should disallow continuously repeating the same numbers.
- The random numbers should be reproducible, given the same seed.
- The generated random numbers should be uniform and independent.

There are various techniques for generating random numbers, including (for further information see references [10,13]):

- Linear-congruential generators.
- Tausworthe generators.
- Extended Fibonacci generators.
- Combined (random number) Generators.

In the above discussions, random numbers are assumed to be distributed uniformly between 0 and 1. When a specific distribution (non-uniform) is used to generate random numbers, then it is called *random-variate* generation. There are a number of techniques used to generate random variates. Some techniques may be better than the others for a particular distribution. Here are some of those techniques (see [10,13] for more information):

- Inverse transformation.

- Rejection.
- Composition.
- Convolution.

5.4 Workload Characterizations

The *workload* of a computer system, as defined by Ferrari [5], "is the set of all inputs (program, data, commands) the system receives from its environment". There are two (workload-associated) operations to be done during the initial stage of an evaluation analysis. The first is the specification of the type of workload which is to operate or model the system. The second operation is the *characterization* of the workload for the system [5]. The workload characterization is an important step in simulation modeling [17] or any other performance evaluations, and it is the hardest technical problem to solve for the investigator [5]. This section is devoted to highlight the features of workload characterizations.

The workload can be characterized with different levels of detail. However, the main characteristics of a workload model are as follows [5,10]:

- Representativeness (accuracy).
- Flexibility (easy and inexpensive to modify).
- Simplicity of construction (collecting data and implementation).
- Compactness (the level of detail).
- Usage costs.
- System independence (portability).
- Reproducibility (repeatability).

- Compatibility (usability with the system model).

In a system performance evaluation, characterizing a workload requires determining which of its several parameters (measurable quantities) have an influence on the performance [5]. In choosing the parameters, it is desirable to use those parameters that depend on the workload rather on the system [10]. Some of the techniques used for workload characterizations are (for more detail see [10])

- Averaging,
- Specifying dispersion,
- Signal-parameter histograms,
- Multiparameter histogram,
- Principal-component analysis,
- Markov models, and
- Clustering.

5.5 Data Collection and Reporting

Data collection is gathering data and information that assist the development of the basic description of each of the system entities and developing probability distribution for the random variables used in the system model [16,13]. Data is collected at different times during the simulation, and it is used to produce simulation report. For example, a report may give the system utilization, mean busy period, mean queue length, etc. The report measures reflect operations completed from the start of the current measurement interval up to the time of the report [5].

5.6 Programming

In this section, three main topics in computer programming are discussed. They are debugging (or verification) techniques, deadlock, and common programming errors.

5.6.1 Debugging techniques

During the development, it should be ensured that the simulation model is correctly implemented *(verification process)* and that it is representative of the actual system *(validation process)*. Thus, verification (or debugging) is associated with the correctness of the implementation, and validation (discussed in Section 5.9) is related with the determining whether the simulation model is a correct representation of the actual system [16,10].

In simulation, as in common computer programming, there are various techniques can be used perhaps together for debugging as follows [16,10]:

1. **Top-Down Modular Design**

   Modularity expects that the simulation model be organized into modules (e.g., subroutines, procedures, or functions) that interact with each other through properly-defined interfaces. They can be developed, debugged and maintained independently. Furthermore, top-down design allows the development to follow a hierarchical structure of the model. Therefore, generally, it is desirable to begin with a moderately detailed model which is gradually made as complex required.

2. **Structured Walk-Through**

   More than one person can read the simulation program for debugging.

3. **Antibugging**

   Additional checks and outputs are incorporated in the simulation program that will signal possible bugs. For instance, if the probability for some events are expected to add up to 1, this may be checked and an error message printed if the sum is not equal to 1 within, perhaps, some accepted error.

4. **Deterministic Models**

   A traditional debugging approach is to permit the user to specify constant (deterministic) distributions. Thus, the output is easily determined.

5. **Run Simplified Cases**

   The model should be run, if possible, under simplifying assumptions for which true characteristics are known or can easily be computed.

6. **Trace**

   Tracing is one of the helpful techniques in debugging a discrete-event simulation program. In a trace, the state of the simulated system is printed out just after each event occurs, and later compared with known (computed) output. Tracing causes extra processing overhead, hence, switches that allow the traces to be turned on and off should be included in the simulation program. Furthermore, the user should be able to determine the level of detail in the trace.

7. **Animation Output**

   On-line graphic displays and traces provide a useful way to illustrate the changes of the system and debug the simulation program.

8. **Continuity Test**

   The simulation program is run several times under different settings of the input parameters. For any one parameter, a slight change in the input should normally cause merely a slight change in the output. Therefore, any prompt (unexpected) changes in the output should be investigated.

9. **Degeneracy Tests**

   The model is tested for extreme values of system, configuration, workload parameters (i.e., lowest or highest values allowed).

10. **Consistency Tests**

    The model is tested to determine if it has similar results for input parameter values that have similar effects.

11. **Seed Independence**

    The model should give similar results for different seed values.

5.6.2 Deadlock

Deadlock occurs when two ore more objects (processes, events, etc.) are competing for some resources or communicating with each other, resulting a permanent blocking of these objects. It is a problem that should be taken under consideration in the design phase. Then, at the implementation stage deadlock detection and recovery facilities should be provided.

Communication protocols usually involve more than one process for exchanging messages, and therefore, the possibility of deadlock occurrence. Therefore, most protocls incorporate a time-out property.

5.6.3 Common Programming Errors

It is infrequent to have a program without an error. There are various programming errors that are most often encountered while writing a computer program. The most common errors, as given by [27], are as follows:

1. *Errors in problem definition*

   When the programmers and the users do not understand each other or the users do not know what they want, then the written programs may produce unwanted results.

2. *Incorrect algorithm*

   The programmers may select a faulty or inadequate algorithm. Painfully, it is not usually possible to know that an undesirable algorithm was selected until it has been used.

3. *Errors in analysis*

   These errors consist of either overlooking possibilities (such as not considering negative values or small or large numbers) or incorrectly solving the problem (e.g., no variable initializations, incorrectly terminating or indexing a loop, etc.) That is, incorrect programming of the algorithm.

4. *Semantic errors*

   Failure to understand how to use a command or function.

5. *Syntax errors*

   Failure to follow the rules of the programming language.

6. *Execution errors*

   Failure to predetermine the probable ranges in calculation; e.g., division by zero or negative square roots.

7. *Data errors*

   Failure to project the ranges of data; e.g., performing a mathematical operation on an alphabetic data.

8. *Documentation errors*

   They occur when the user documentation does not match the program; e.g., the program is awkward, clumsy, or incomplete.

5.7 Simulation Languages and Systems

Today, there are several simulation languages and systems that are used for general purposes, or for particular problems or environments. This section reviews only three different simulation languages and systems. They are the *smpl* simulation language, the SMPL simulation system environment, and the SIMSCRIPT programming language.

5.7.1 smpl: A Simulation Language

smpl [17] is a discrete-event simulation language. It is a functional extension of a general purpose language. This addition comes in a form of a set library functions which, together with the selected general purpose language, construct an event-oriented simulation language. Different implementations of *smpl* are available for several programming languages such as Basic, Pascal, Fortran, C, and PL/1, and on various machines ranging from microcomputers to super-computers. The source code in C language of *smpl* is available in the public domain.

smpl views a simulated system of having three types of entities:

1. **Facilities:**

   A system consists of a number of interconnected facilities. Generally, a facility represents some resources of the system being modeled and provides some services. For example, a CPU in a computer system, or a channel in a communications network model. The interconnection between facilities is done implicitly through the model's routing of tokens between facilities. *smpl* provides functions to define, reserve, release, and preempt facilities.

2. **Tokens:**

   The dynamic progress of the system is modeled by the movement of tokens through a set of facilities. Thus, a token may represents, for instance, a task in a computer system, or a packet in a communications network model. *smpl* provides two basic operations for managing the flow of tokens through the simulated system: a token may reserve (preempt) facilities, and it may schedule events of different durations. However, when a token tries to reserve an engaged facility, it is queued until the facility becomes ready.

3. **Events:**

   An event is a change of state of any system entity. For example, a task completes a CPU execution interval, or a packet is sent over the channel in the communications network model. *smpl* provides functions for scheduling events and for selecting events for execution.

A *smpl* simulation program comprises an initialization routine, a control routine, and a number of event routines. There are two types of initialization routines that are provided by *smpl*. One, the smpl() function, initializes and clears data structures, and sets both simulation time and the start of the current measurement interval to zero. It is used to completely initialize the simulation system. The other function, reset(), clears all accumulated measurements, and is used usually for reinitialization when multiple simulation runs are required.

The control routine chooses the next event to occur and transfers control to the corresponding event routine which may schedule one or more other events and then return the control to the control routine. Scheduling an event implies that a record, containing the event number (identifier), the event occurrence time, and (in some cases) the token associated with this event, is created and inserted into the event list. The event list is ordered in increasing values of event occurrence times.

smpl provides a number of random-variate generator functions for different distributions such as uniform, exponential, Erlang, and hyperexponential.

5.7.2 SMPL: A Simulation Environment

SMPL [17] is a simulation environment which consists of the *smpl* simulation subsystem (see Section 5.7.1), additional tools for debugging, analysis, and reporting, and an interactive run-time interface called *mtr*. The objective of SMPL is to increase the speed of the development, testing, and use of simulation models. Figure 5.1 illustrate the SMPL simulation environment.

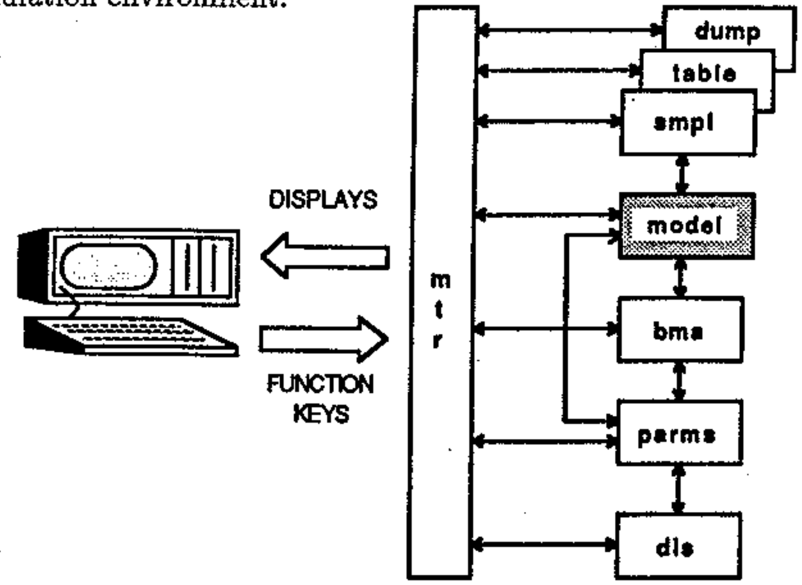

Figure 5.1: The SMPL Simulation Environment

Essential simulation operations are accomplished through simulation program calls to *smpl* functions. The interface between the user and the rest of the simulation environment is provided by *mtr*. Monitoring operations are obtained by an *mtr* function which is invoked by *smpl* whenever an event occurs. Besides these two components, the SMPL simulation environment includes some other modules such as *dump, table, bma, parms,* and *dis* modules.

The *dump* module displays the state of the simulated system. Functions for table definition, data entry, reporting, and plotting of the simulated system are provided by the *table* module. Tables are used to collect distributions of output parameters. The "match means analysis" *(bma)* module is used to estimate the simulation run length necessary to accomplish a certain accuracy for a model output parameter. The *parms* module offers facilities for defining and naming SMPL parameters. It provides the means of communication between the simulation program, SMPL, and the user. Plotting facility utilization, average or current queue length, or an SMPL parameter versus time or other parameter are provided by *dis* module using the computer's bit-mapped graphics display.

5.7.3 SIMSCRIPT Programming Language

SIMSCRIPT [14], is primarily a general programming language which provide the means to form discrete-event, continuous, or combined simulation models. It is a process-oriented and event-oriented simulation language. Due to its general process approach, its advanced and complex data structures, and its powerful control statements, SIMSCRIPT is usually used for large complicated simulation models. Its English-like and free-format syntax make it easy to read and nearly self-documenting. SIMSCRIPT is available, commercially, for computers ranging from microcomputers to mainframes. The microcomputer and workstation versions include the SIMGRAPHICS animation and graphics package [16].

In SIMSCRIPT, a system (statically) is characterized by entities, attributes, and sets (similar to queues). An entity is a program element (similar to a variable) that exists in a model system. Its values called attributes. Logical groupings of entities are referred to as set. A given entity may have any number of attributes, own any number of sets, and belongs to any number of sets [13,14].

The basic unit of action in a SIMSCRIPT simulation is an *activity*. The static structure (i.e., entities, etc.) discussed above is used for activity descriptions. The dynamic structure of the system is modeled by *events* which are caused by the start or stop of activities (i.e., changes of system state). [14].

To build a simulation model in SIMSCRIPT, a *preamble*, main program, and event routines should be implemented. The preamble provides a static description of the whole model and the necessary events, entities, and sets. It does not contain any executable statements. The main program is used to get input parameters for the simulation, to initialize the even list and state variables, and to start the simulation. Event routines develop the action needed when processing each event in the model [16,13].

5.8 Experimental Designs

In simulation, experimental design offers a way of determining prior to the runs are performed which particular configurations to simulate so that the wanted information can be achieved with the minimum amount of simulation [16]. Thus, the purpose of a decent experimental design is to achieve the maximum information with the minimum number of experiments [10]. Biles in Chapter 3 of [13] defines experimental design as "determining the alternatives that can be evaluated through simulations, choosing the important input variables and their appropriate levels, selecting the length of the simulation run, and the number of replications.

A good experiment design helps in isolating the effects of different variables that may affect the performance. It, also, permits determining if a variable has a critical effect or if the observed difference is merely because measurement errors [10].

There are several varieties of experimental designs. The most often used designs are either full or partial factorial designs [10]. A Full factorial design exercises each possible combination of all levels (values) of all factors (variables). Thus, a performance analysis with k factors, with the *ith* factor having n_i levels, expects $\prod_{i=1}^{k} n_i$

experiments. The usefulness of full factorial designs are, first, every possible combination of configuration and workload is examined. Second, the effect of each factor and its interaction with other factors can be found. Regardless, it consumes too much time and resources to undertake these many experiments [10].

Therefore, there are three methods to reduce the number of experiments, as stated by Jain [10]:

1. Reduce the number of levels for each factors; e.g., a full factorial design in which each of the k factors is used at *two* levels requires 2^k experiments.

2. Reduce the number of factors.

3. Use fractional factorial designs.

In the last alternative, a fraction of the full factorial design can be used due to the expense or the time required. However, the information obtained from a fractional factorial design is less than that obtained from a full factorial design.

5.9 Validation

Validation is part of the simulation model development, as it is stated in Section 5.6.1, and it depends upon the assumptions and the system being modeled. It is considered one of the most difficult problems facing a simulation analyst [16]. Jain [10] describes validation as to ensuring that the assumptions used in developing the simulation model are acceptable in that, if correctly implemented, the model would produce results close to that observed in real system.

Model validation includes validating the three major components of the model [10]:

1. Assumptions.

2. Input parameter values and distributions.

3. Output values and conclusions.

Furthermore, each of these three components may be eligible for a validity test by comparing it with that obtained by the following three possible sources [10]:

1. Expert intuition.
2. Real system measurement.
3. Theoretical results.

5.10 Output Analysis

5.10.1 Types of Simulations

There are mainly two types of simulations, with respect to the simulation output. One is called a *terminating (or transient)* simulation wherein the simulation run length is determined by the problem. That is, there is a "natural" event E that defines the length of each run (replication). The event E is specified before any runs are performed. It may be a measure in terms of the time needed to attain a particular set of activities required to reach a specified state, given initial conditions determined by the problem. Generally the initial conditions for a terminating simulation affects the chosen measures of performance, therefore, these conditions should be representative of those of the actual system. A terminating simulation stresses in how many times the simulation should be repeated (with different random-number streams) to achieve a certain accuracy [16,17].

The second type is called a *steady-state* simulation. Here the objective is to study the long-run behavior of a system in omission of the effect of the initial conditions of the model. Both the initial conditions and the length of the simulation are determined by the modeler, and output values should be independent of the initial conditions. Results of the initial part (transient state) of the simulation should not be considered in the final computations; otherwise the point estimation may be subject to initialization bias [10,17]. For further information in how to remove the transient effect see Section 5.10.5.

5.10.2 Types of Measures

The *mean (or average)* value of a simulation output parameter is the most often used as a performance measure. For a discrete-time process, the simulation generates a sequence of n sample values $X_1, X_2, \ldots, X_n$ whose *sample mean* would be

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

As n goes to infinity, $\overline{X}$ (sample mean) converges to the expectated value of X (denoted by $E[X] = \mu$), known as the *distribution* or *true* mean [17].

Another performance measure is the *variance* which is a measure of dispersion of distribution. The *sample variance* of a set of n sample values $X_1, X_2, \ldots, X_n$ is defined by

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2$$

Likewise, as n goes to infinity, s^2 converge to the expected variance, denoted by $E[(X-\mu)^2] = \sigma^2$.

The square root of the variance is the *standard deviation.* The ratio of the standard deviation to the mean is used as a performance measure and is called the *coefficient of variation* [17].

5.10.3 Confidence Intervals

In simulation, it is crucial to know how long run lengths have to be to achieve a sample mean arbitrary close to the distribution mean (μ). Therefore, the question of the simulation run length is answered by a measure called a *confidence interval* [17]. In this section, the confidence interval for a steady-state simulation is examined, since most simulations in computer system design environment are steady-state simulations.

Let the simulation output produce a set of N sample values $Y_1, Y_2, \ldots, Y_N$ from a distribution with true (but unknown) mean μ. Then the point estimator of μ (called sample mean) can be computed as

$$\overline{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

Also, let $1 - \alpha$ be the probability that the absolute value of the difference between the sample mean and the true mean is equal to or less than a particular value (H), called the confidence interval half-width. That is,

$$P[|\overline{Y} - \mu| \leq H] = 1 - \alpha$$

Therefore, the confidence interval for the mean is defined as

$$P[\overline{Y} - H \leq \mu \leq \overline{Y} + H] = 1 - \alpha \tag{5.1}$$

The interval $\overline{Y} - H$ to $\overline{Y} + H$ is known as the *confidence interval*, and $1 - \alpha$ is called the *confidence level*. Equation 5.1 declares that if a large number of experiments are performed and the confidence interval for each experiment is computed, then the percentage of intervals containing μ is $1 - \alpha$.

If the output sample values are independent random variables from a normal distribution, then H is given by

$$H = t_{\alpha/2,N-1}\frac{s}{\sqrt{N}},$$

where $t_{\alpha/2,N-1}$ is the upper $100(1-\alpha)\%$ point of a t distribution with $N - 1$ degree of freedom, and $s/\sqrt{N}$ is the standard deviation of the distribution of the sample mean [13,17].

5.10.4 Replication

A replication of a simulation run is a recording of the system's performance under a specified combination of parameters with different random number stream for each run [13]. Suppose k replications of a simulation run have been made, each producing m sample values with a sample mean Y_i (the mean of individual runs). Since for each run a various random number stream was used, the means $(Y_i, i = 1, 2, \ldots, m)$ are independent, and will be approximately uniformly distributed for large m. The over all sample mean (i.e. the mean of means) can be computed by

$$\overline{Y} = \frac{1}{k}\sum_{i=1}^{k} Y_i,$$

with a confidence interval $\overline{Y} \pm H$, where $H = t_{\alpha/2,k-1}s/\sqrt{k}$. To obtain the desired value of H, m (the simulation run length) or k (the replications number) can be increased [17].

Note that, long simulation runs are usually used to remove the effect of transients, and they give large numbers of observations which increase the confidence level in the estimate of the mean at the cost of additional computations. Replications are helpful for simulation analyses that are established to examine the transient rather than the steady-state behaviors of the system [13].

5.10.5 Warm-up Problem

The initialization of a simulation program may put the system in unusual state, and it may take some time for the model to "warm-up". The output sample values collected during this warm-up (transient) state may not be representative of steady-state behavior and, if the run length is not adequately large may bias the sample mean (Y_i). The task of identifying the end of this state is called *transient removal.* It is difficult, if not impossible, to specify precisely what constitutes the transient state and when it ends [10,17]. There are several techniques for transient removal including the following (for more information see [10]):

1. Long runs.

2. Proper initialization.

3. Truncation.

4. Initial data deletion.

5. Moving average of independent replications.

6. Batch means.

5.10.6 Comparing alternative designs

Commonly interest is focused on the statistical analyses of the output from several different simulation models rather than a single model. Law [16] states that an essential condition for using many statistical techniques for comparing alternative configurations is the capability of gathering IID (independent and identically distributed) samples with expectation equal to the desired measure performance. This is easy to achieve in the case of terminating simulations by making independent replications. But it is complicated in the case of steady-state, because it is difficult to obtain IID observations having expectations equal to the desired steady-state measure performance. This section describes a special case of one superior technique, called "paired comparison". The special case evaluates two alternative designs on the basis of their means.

Let x be the output parameter on which the comparison is be based. Also, let X_{ij} (for $i = 1, 2$ and $j = 1, 2, \ldots, k_i$) be a sample of k_i observations from system i, and $\mu_i = E[X_{ij}]$ be the expected response of interest. The smaller of k_1 and k_2 is used for the comparison, and let that be equal K. Now, X_{1j} and X_{2j} can be combined to define the difference $D_j = X_{1j} - X_{2j}$. A confidence interval for the mean difference can be computed by

$$\begin{aligned} \overline{D} &= \frac{1}{K}\sum_{j=1}^{K} D_j \\ s^2 &= \frac{1}{K-1}\sum_{j=1}^{K}(D_j - \overline{D})^2 \end{aligned}$$

$$ConfidenceInterval \quad : \quad \overline{D} \pm t_{\alpha/2,K-1} s/\sqrt{K},$$

where, $1-\alpha$ is the confidence level. The advantage of this method is that, X_{1j} and X_{2j} do not have to be independent and their variances need not to be the same.

Chapter 6

Simulation of Communications Protocols

6.1 Modeling of Protocols

It is common to view the structure of a communication architecture as a hierarchy of different protocol layers. A good example is the ISO-OSI seven-layered model. Each layer provides certain services and also needs certain services from the neighboring layers. Moreover, each layer views the neighboring layers as "black boxes" which provide specific services, but does not concern with how the neighboring protocols provide them. This way of viewing an interface between adjacent layers establishes the service specification of a protocol. Also, this specification should be "abstract" in the sense that it specifies the interface's commands and responses, but not their implementations [25].

There are, basically, two protocol specification methods that have customly been used: *state machines* and *programs*. Actually, there is no fundamental difference between these approaches. The state machine model is inspired by recognizing that protocols may be practically considered as rules describing the responses or outputs of a protocol to each command or input. On the other hand, program specifications are influenced by noticing that protocols are simply a special set of information processing procedures [25].

A number of variations of these basic protocol specifications have been used including the following [25]:

- **Finite State Automata**

  An FSA is a simple form of state machine models, since it has only a single state variable which takes on a relatively small range of values. States (in the case of protocols) usually correspond to different steps in operations; e.g., idle, data transfer, etc. An FSA can be represented graphically or in tabular form. The restriction of having a single state variable is a serious obstacle particularly when a protocol has features such as sequence numbers or message texts.

- **Abstract Machine Model**

  This is an enhancement of an FSA achieved by having multiple state variables of different types. The main distinction is that, in the case of the abstract machine, the "state" becomes a vector of variables instead of a single variable in the case of the FSA. A problem with abstract machine models comes from their purely passive behavior (i.e., they define the effects of invoked operations, but do not incorporate any concept of an active operator that may initiate events or invoke operations).

- **Formal Languages**

  Formal languages and grammars which formulate them are another variation of the state machine type model. One of the limitations of formal grammar models is that they do not differentiate between inputs and outputs. Additionally, a formal language model has similar limitations to an FSA in representing the text of messages or values of sequence numbers.

- **Sequencing Expressions**

  Sequencing expressions are an effort to define a sequence of protocol operations directly. The major operations are repetitions, follows, and alternatives. A good example of sequencing expressions is the well-known regular expressions that conform to FSA. Similar to grammars, sequencing expressions do not formally distinguish between inputs and outputs.

- **Petri Nets**

  Petri nets are another graphical tools used for protocols specifications. In general, Petri nets have the same limitations as FSA. Some extensions to Petri nets are more powerful than abstract machines by including timing constraints to the transitions.

- **Buffer Histories**

  They are equivalent to formal languages and sequencing expressions in their effort to formulate the input/output behavior of the system without any reference of explicit states. The basic model is a process (an active computing entity) that has a set of buffers which facilitate the exchanges of items of specified types with others. The main part of the specifications are relations between the input and output items read from or written to various buffers by different processes. Buffer histories seem useful when the behavior being specified in suitably expressed as a relationship between sequences of items.

- **Abstract Data Types**

  ADT is a form of encapsulating data with their operations. It is practically a formalization of the concept of the abstract machine. But, data types used as elements of the state vector may themselves be user-defined abstract data types. In abstract data type models, exception handling is a serious problem.

- **Programs and Program Assertion**

  One of the oldest approaches of describing operations of a system are programs. Here, based on the program implementations, protocols specifications are obtained. The main disadvantage of programs as specifications comes from their shortcoming of abstraction.

  State deltas are a technique in specifying the system behavior with least explicit programs. A state delta represents the basic unit of specifications, giving a pre-condition, modification list, and a post-condition. It states that if the pre-condition becomes true at some time, then eventually later the post-condition will be true, and during that period of time only the variables listed in the modification list may change.

  The difference between program and state machine specifications is not major. (Programs can be converted into an abstract machine model by considering the program counter as the main state variable).

6.2 Statistical Models of Point-to-Point Communication Channels

A general model of a point-to-point communication channel comprises two stations (transmitter and receiver), a channel, and probably a noise source, see Figure 6.1 for the block diagram of the model. A typical operation of this model is as follows: Each frame (or packet) is processed by a station (the transmitter), propagates (and perhaps injected by a noise) through the channel to the other station, and is processed by the receiver. The receiving station checks that the packet is error free and then send an acknowledgment. The erroneous packets must be retransmitted. The complexity of the model depends on the data link control protocol; i.e., the flow control used (e.g., Stop-And-Wait, Go-Back-N ARQ, or Selective-Repeat ARQ) and on the transmission

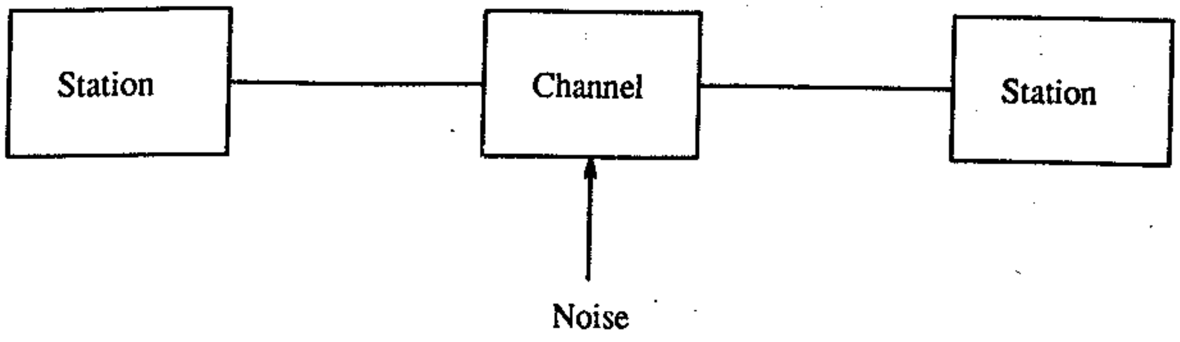

Figure 6.1: A Channel Model

mode (HDX or FDX). The following are some of the possible model parameters:

1. The channel propagation time.
2. Packet interarrival times (distribution and mean)
3. Window size; in case of ARQ.
4. The noise properties:
   (a) Probability of bit error.
   (b) Probability of block (packet) error.
   (c) Burst noise distribution.
5. Properties of data link level protocol.
   (a) Type of data link protocol.
   (b) Flow control type.
   (c) Transmission mode.
   (d) Frame size.

(e) Window size; in case of ARQ.

(f) Maximum number of retansmissions.

6. and some more other parameters

The analysis studies of this model result mostly in two output:

1. Mean throughput rate.

2. Packet delay statistics.

Bibliography

[1] Arnold O. Allen. *Probability, Statistics, and Queueing Theory With Computer Science Applications.* Academic Press, 1978.

[2] Li Fung Chang. Throughput estimation of ARQ protocols for a ryleigh fading channel using fade- and interfade-duration statistics. *IEEE Transactions on Vehicular Technology*, 40(1):223–229, February 1991.

[3] Justin C. I. Chuang. Comparison of two ARQ protocols in a Rayleigh fading channel. *IEEE Transactions on Vehicular Technology*, 39(4):367–373, November 1990.

[4] Richard A. Comroe and Daniel J.Costello. ARQ schemes for data transmission in mobile radio systems. *IEEE Journal on Selected Areas in Communication*, SAC-2(4):472–481, July 1984.

[5] Domenico Ferrari. *Computer System Performance Evaluation.* Prentice-Hall, Inc., 1978.

[6] George S. Fishman. *Principles of Discrete Event Simulation.* John Wiley and Sons, 1978.

[7] Paul E. Green. *Computer Network Architectures and Protocols.* Plenum Press, 1982.

[8] Simon Haykin. *Communication Systems.* John Wiley & Sons, 1983.

[9] Trevor Housley. *Data Communications and Teleprocessing Systems.* Prentice-Hall, 1987.

[10] Raj Jain. *The Art of Computer Systems Performance Analysis.* John Wiley and Sons, 1991.

[11] M. R. Karim. Transmission of digital data over a Rayleigh fading channel. *IEEE Transactions on Vehicular Technology*, vt-31(1):1–6, February 1982.

[12] M. R. Karim. Packet communications on a mobile radio channel. *AT & T Technical Journal*, 65(3):12–20, May/June 1986.

[13] Naim A. Kheir. *Systems Modeling and Computer Simulation.* Marcel Dekker, Inc., 1988.

[14] P. Kiviat, H. Markowitz, and R. Villanueva. *SIMSCRIPT II.5 Programming Language.* CACI, Los Angeles, 1973.

[15] B. Kurz. Course notes: CS 5865 data networks, 1989.

[16] Averill M. Law and W.D̃avid Kelton. *Simulation Modeling and Analaysis, Second Edition.* McGraw-Hill, Inc., 1991.

[17] M. Macdougall. *Simulating Computer Systems Techniques and Tools.* The MIT Press, 1987.

[18] James Martin. *Telecommunication and computer.* Prentice-Hall, 1976.

[19] John E. Mcnamara. *Technical Aspects of Data Communication.* Digital Press, 1982.

[20] Martin S. Roden. *Digital and Data Communication Systems.* Prentice-Hall, 1982.

[21] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication.* The University of Illinois Press, 1964.

[22] Chee Kheong Siew and David J. Goodman. Packet data transmission over mobile radio channels. *IEEE Transactions on Vehicular Technology*, 38(2):95–101, May 1989.

[23] William Stallings. *Data and Computer Communications.* Macmillan Publishing Company, 1988.

[24] William Stallings. Faster packet networks. *Byte*, 16(12):173–181, November 1991.

[25] Carl A. Sunshine. Formal modeling of communication protocols. In S. Schoemaker, editor, *Computer Networks and Simulation II*, pages 53–75. North-Holland Publishing Co., 1982.

[26] Andrew S. Tanenbaum. *Computer Networks.* Prentice-Hall, 1981.

[27] Dennie Van Tassel. *Program Style, Design, Efficiency, Debugging, an Testing.* Prentice-Hall Inc., Englewood Cliffs, NJ, 1978.

[28] John B. Thomas. *An Introduction to Statistical Communication Theory.* John Wiley and Sons, 1969.

[29] Kishor S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications.* Prentice-Hall, Inc., 1982.