CHAPTER 2.

SYSTEM'S PRINCIPLES

In this chapter the basic principles for constructing the system are presented, and its principles of operations are discussed. The concepts which will be defined and developed in this chapter will be used throughout the thesis.

2.1. The Scope of the System

The system, in the context of this research, is defined as the network topology plus the interface functions, as shown in Figure 2.1. The shared buses of the hosts are the boundaries of the system. Each host bus may be shared by several computing units. In general, the units which reside beyond the host bus are beyond the scope of this thesis. From the host point of view, the network is a resource, like other memory resources. Via the shared host bus, a computing unit writes or reads a message to or from a buffer in the network's interface.

A node is defined as a point in space from which there is an access to the network, via a host bus. In the following discussion the term "node" refers to the interface functions at some point.

2.2. The Network Topology

The network topology is a hypergraph, which is formally defined as [Berg73]: Let $X = \{x_1, x_2, \cdots, x_p\}$ be a finite set of nodes, and let $F = \{E_i \mid i \in I\}$ be a family of

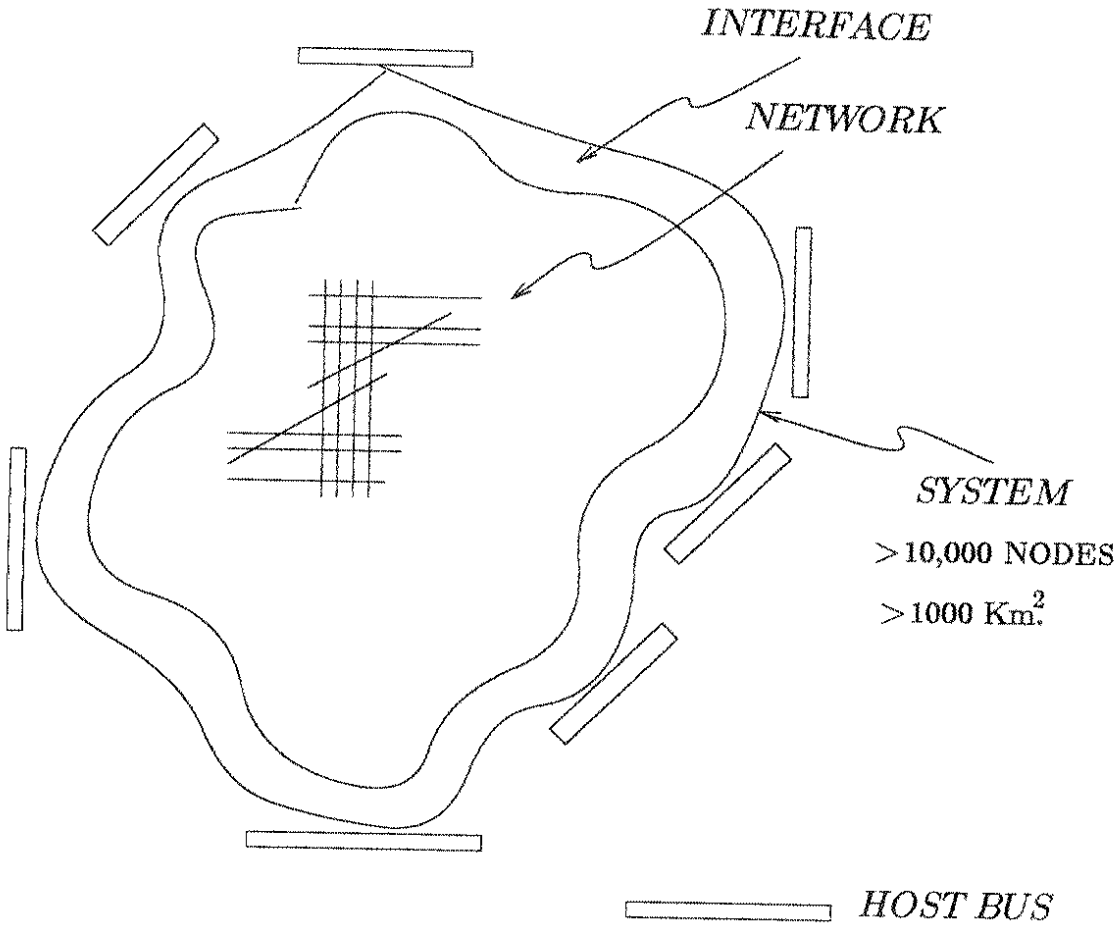

Figure 2.1: The System Description

subsets of X. The family F is said to be a hypergraph on X if (1) $E_i \neq \phi$, and (2) the union of all E_i is equal to X. The couple $E = (X,F)$ is called a *hypergraph*. The elements $x_1, x_2, \cdots, x_p$ are called the *vertices*. The sets $E_1, E_2, \cdots, E_m$ are called the *edges of the hypergraph*.

In this research four different hypergraph configurations were considered.

(1) Two-dimensional regular hypergraph (2D-R) – each edge (net) E_i has exactly n vertices (nodes), such that $n^2 = p$ (p is the total number of vertices in the hyper-

graph), and each vertex belongs exactly to two edges (E_i). Figure 2.2a is an example of a 4–by–4 regular hypergraph in the $x-y$ plane; each edge has exactly 4 vertices and the total number of vertices is 16. Figure 2.2b is a representation of a 3–by–3 2D–R hypergraph, such that each net is realized by a centralized optical star.

(2) Two–dimensional, partial hypergraph (2D–P) – the set of p vertices is divided into $2k$ subsets. Each E_i has exactly n vertices, such that (i) k of its vertices belong to exactly two edges, (ii) a vertices belong only to one edge, and (iii) $k+a=n$. The total number of vertices in the partial network is

$$p = k^2 + 2ka = n^2 - a^2$$

Described in Figure 2.3 the different possible layouts of 2D–P. Note that since

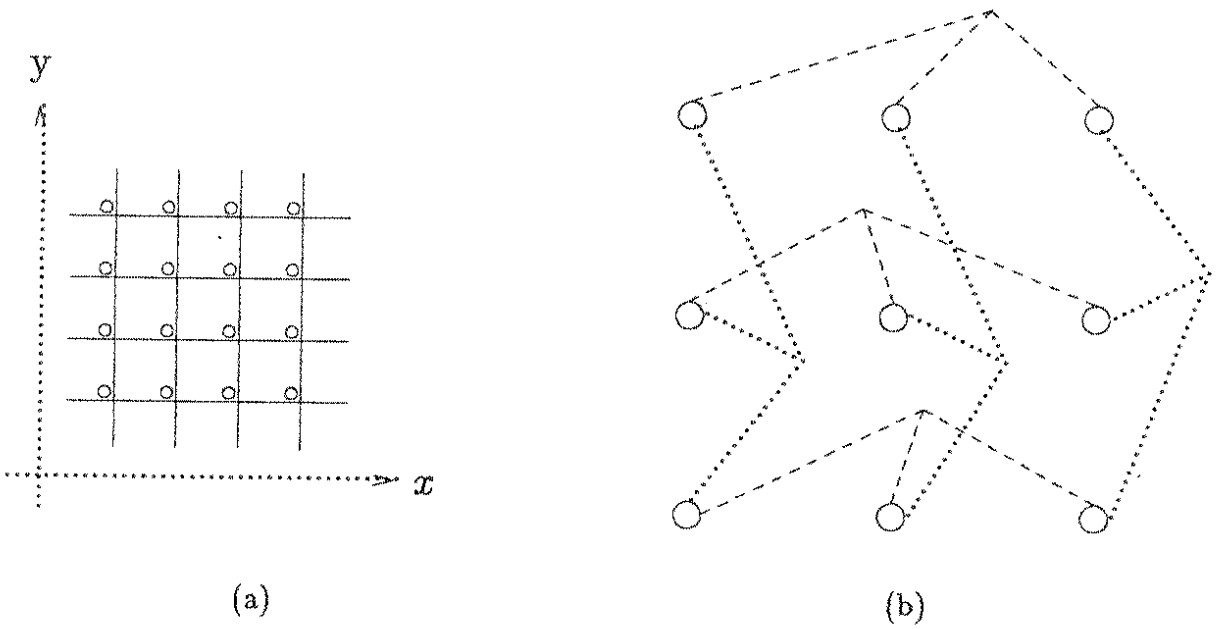

Figure 2.2: Two–Dimensional Regular Hypergraph

every edge is a centralized optical star, the nodes (vertices) are **indistinguishable** with respect to the star's center. Therefore, the a vertices of each net, which belong only to one edge, can always be rearranged as shown in Figure 2.3b or Figure 2.3c.

(3) Three–dimensional, regular hypergraph (3D–R) – each edge (net) E_i has exactly n vertices (nodes), such that $n^3 = p$ (p is the total number of vertices in the hypergraph), and each vertex belongs exactly to three edges (E_i). Figure 2.4 is an example of a 3–by–3–by–3 regular hypergraph in $x-y-z$ space; each edge has exactly 3 vertices, and the total number of vertices is 27.

(4) Three–dimensional, partial hypergraph (3D–P) – the set of p vertices is divided into $3k^2$ subsets. Each E_i has exactly n vertices, such that (i) k of its vertices

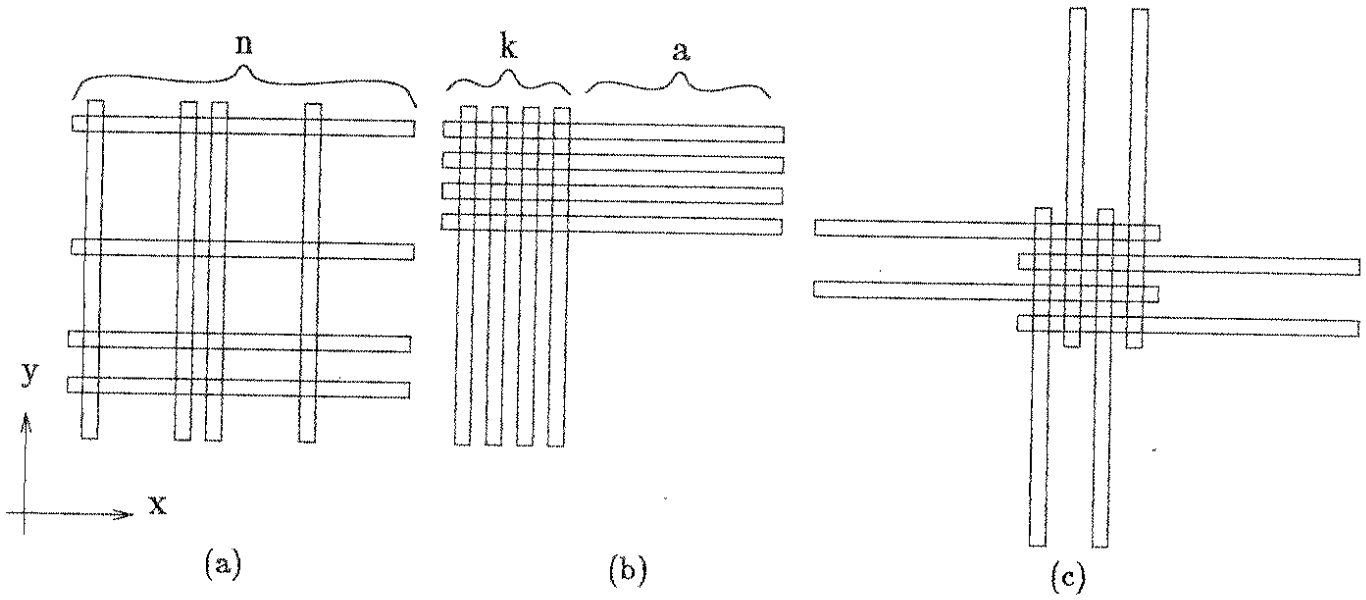

Figure 2.3: 2D–2P Partial Hypergraph

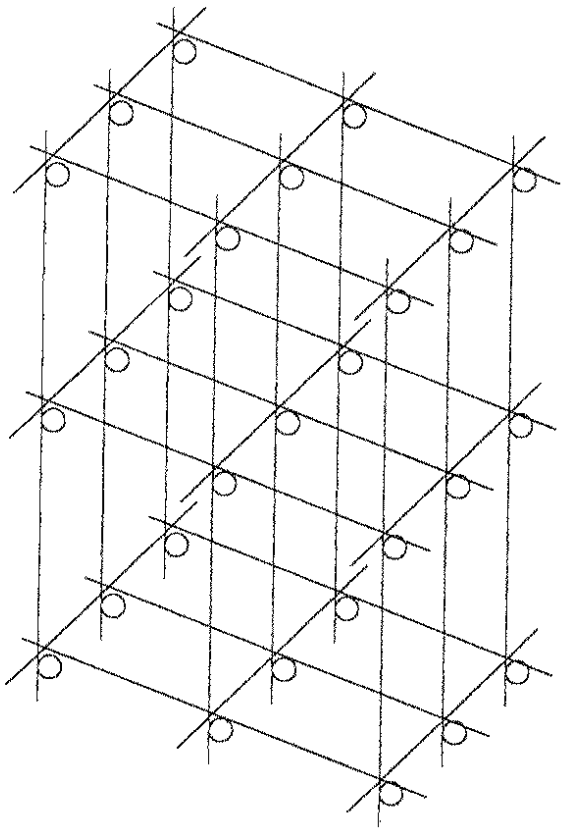

Figure 2.4: Three–Dimensional Hypergraph

belong to exactly three edges, (ii) a vertices belong only to one edge, and (iii) $k+a = n$. The total number of vertices in the 3D–P network is

$$p = k^3 + 3k^2a = n^3 - a^3 - 3a^2k$$

2.3. The Optical Star

Each net is a passive optical star, which is realized by centralized passive optical couplers, as shown in Figure 2.5, which is an array of single mode, optical couplers. A general discussion on star couplers is found in [Marh84]. It is assumed that the optical star is built in a very small area in space, so the communication on the net merges to one

point in space (called the center of the star), and then broadcasts back to all the nodes of the net. The optical hypergraph is composed of multiple optical stars, as shown in Figure 2.2b, where there are 9 nodes and 6 optical stars, with each node having access to two optical stars via its two ports.

The single–mode, optical coupler is a device with two input terminals (C^A and C^B), and two output terminals (C^C and C^D).

Let $G(A,D)$ – be the transmittance from point A to point D, representing the fraction of power from A which reaches D.

The optical coupler has the following reciprocity property:

$$G(C^A,C^D) = G(C^B,C^C)$$

$$G(C^A,C^C) = G(C^B,C^D)$$

The major consequence of this property is that it is impossible to send from both inputs more than half of the energy to only one of the outputs. For example, if

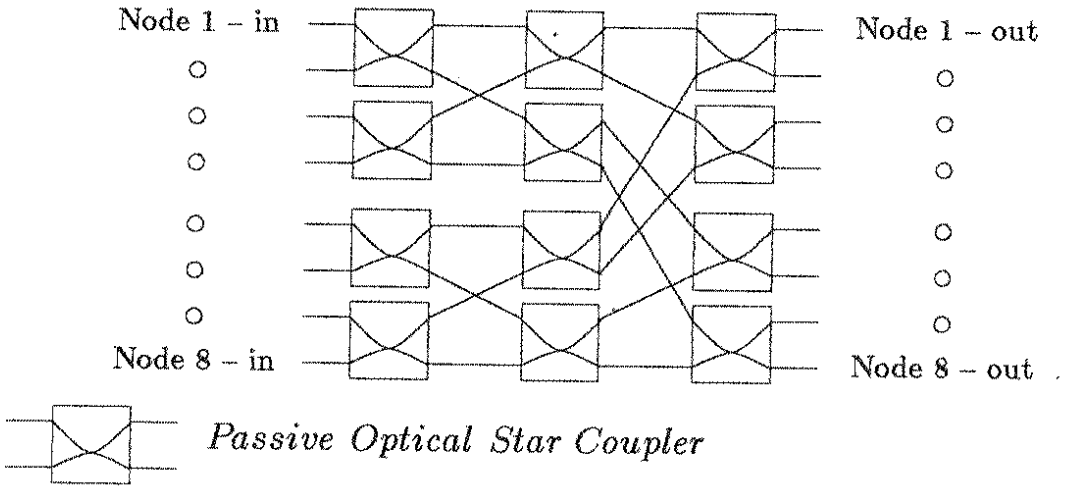

Figure 2.5: Optical Star

$G(C^A,C^D)=\frac{3}{4}$, then $G(C^A,C^C)=\frac{1}{4}$, and hence, $G(C^B,C^D)=\frac{1}{4}$. Therefore, for practical reasons it can be assumed that the transmittance of the optical coupler is exactly one half, i.e., the coupler attenuates the signal by $3db$.

In [NTM85], Tobagi et al. analyze different net configurations which use passive optical couplers. Their results demonstrate the feasibility of constructing a passive fiber optic net with about 100 nodes, within an area of more 1000 square kilometers.

The net is constructed of single–mode fibers with very low losses, and $3db$ optical couplers (only symmetric couplers are considered). The following is a comparison among three passive topologies: star, binary–tree and ring (or linear bus).
Given that (a) the transmitter emits a unit of energy into the net, (b) the net has N nodes (N is power of 2), and (c) the only losses are by splitting the energy by the optical couplers. *What is the minimum energy that each node receives?*

(1) Star topology –

$$E_r=\frac{1}{2^{\log_2 N}}=\frac{1}{N}$$

(2) Tree topology –

$$E_r=\frac{1}{2^{(2\log_2 N)-1}}=\frac{2}{N^2}$$

(3) Ring or linear bus topology –

$$E_r=\frac{1}{2^N}$$

Clearly, the passive star topology distributes the energy most efficiently. Detailed discussion of the energy issue, with respect to all losses, can be found in [NTM85].

2.4. Time Partitioning

The timing relationship among the system's nodes and nets has a very important role in the implementation of high–level system functions, e.g., distributed operating systems and voice communication.

There are three main reasons for time slotting:

(1) To be able to transmit a single bit of information into the centralized net, almost every bit period. Thus, the upper bound on the net utilization is almost 100%. Note that as the bandwidth gets higher the size of the packet in the serial link gets shorter. In a large area network with a bandwidth over one gigabit/sec it is reasonable to assume that several packets can be in the channel at the same time. Without slotting, the next one to transmit must wait until it sees the tail of the current packet. Thus, as the bandwidth gets higher more communication capacity is wasted.

(2) To be able to broadcast multiple short control messages successively from multiple nodes.

(3) To support a global system clock, which maintains global event synchronization of all the system's nodes, and enables time stamping of all communication and computation events.

(4) To have a well–defined global state (or instantaneous description), and well–defined global state transitions.

(5) To reduce the probability of collision, as in the case of slotted Aloha versus pure Aloha [Abra77].

The system is synchronized by each time slot. Each node's interface has a slot counter for measuring the time. All the system's slot counters are synchronized and have the same reading (in Chapter 6 there is a detailed description and analysis of the mechanism for achieving global event synchronization).

The slots are grouped into frames with f slots per frame. The duration of one time slot and the value of f are determined such that one time frame duration is larger than the maximum delay of any node to the center of the net and back. The duration of a slot is measured in bit periods, and is about the size of a typical packet or page in the system.

2.4.1. Periodic exchange of state information

Each slot is subdivided into $r+1$ minislots. The first r minislots are very short control minislots (CMSs), and the last one is a data minislot (DMS). The minislots are used by different nodes, so during each time slot $r+1$ different nodes are transmitting over the net.

The main reason for dividing the slot into control and data parts is for functional and physical partition between the communication (or computation) controls, and the actual data transfers (or processing).

2.4.2. Parcels for voice communication

During the DMS, which occupies most of the time slot, one packet of data is sent from one node. The data packet may be subdivided into p parcels. Each parcel may have a different destination or destinations over the net. The use of parcels enables the

node to mix small and large data messages in an arbitrary fashion. This makes the communication over the net flexible and dynamically adaptable to different types of communication. As a result, the use of the communication resource of the net becomes more efficient. The use of parcels, together with global event synchronization, enables the simple integration of voice and data communication (see Chapter 9).

2.5. Basic Principles of Operation

The operation of this system follows some general principles that are defined in this section.

2.5.1. Global synchronization and total event ordering

The time on each of the system's nets is partitioned in the same way, and with equal slot duration. Global synchronization means that the time patterns on all nets cannot differ by more than half a time slot (see Chapter 6). Each node interface has a slot counter (local event clock). All the system's slot counters are synchronized, such that at the end of each slot all the system's slot counters have the same reading for a period of time which is greater than zero. With this value, the current events at each node are time stamped. Thus, if all the events in the system are time stamped, then the system preserves total temporal event ordering.

2.5.2. Synchronous and asynchronous operation

There are two possible views on this system, the microview and the macroview. In the microview, the system is asynchronous, i.e., every node has its own local clock,

which is used for transmission and for the interface's finite state controllers. In the macroview, the system is synchronous, and its basic event is a single time slot. The synchronous event is three or four orders of magnitude longer than the asynchronous event, a single bit period.

2.5.3. Open mode operation

Global event synchronization and total event ordering enable the system to perform distributed computing functions in an *open mode* fashion. For example, a process can send a request with a time stamp to all the other cooperating processes. The request will be granted, without an explicit acknowledge after the two conditions are true: (i) the process waits the necessary required time for the request to reach all the other cooperating processes, and (ii) the requesting process does not receive similar requests, with an earlier time stamp, during its waiting period.

2.5.4. State information and common knowledge

Control and state information are transmitted during the control minislots (CMSs), so that all nodes periodically exchange state information. This information is considered **local common knowledge** after it has reached all the net's nodes. Once this information has reached all the **system's** nodes, it is then considered **global common knowledge.**

It is not required that **common knowledge** information be kept explicitly in the node. The interface can perform real–time computation on the information, incorporate the result n some local parameters, and discard the original data. So, the inter-

face does not contain a large number of replicated parameters.

For example, the state information, which is exchanged periodically, can be used to integrate, in a uniform manner, the following functions:

(i) Data Communication

(ii) Distributed Computation

(iii) Distributed Data Base Management

(iv) Real–Time Voice Communication

2.5.5. Round–robin scheduling

Round–robin scheduling is very simple to implement in real–time, and can guarantee fairness. Moreover, for a homogeneous system, round–robin scheduling is efficient. In principle, other scheduling algorithms can be implemented, but for the presentation, understanding, and analysis round–robin is simpler. An important condition for implementing any other scheduling scheme is to be able to execute it under real–time constraints, which become tighter as the net's bandwidth increases.