CHAPTER 7.

TWO–DIMENSIONAL HYPERGRAPH

7.1. General

The physical layout of a 2D hypergraph is mapped onto a two dimensional square grid, as shown in Figure 7.1a, which is an example of a 5–by–5 2D regular hypergraph. For regular hypergraphs, each node is identified by an ordered pair (x,y):

$$node(x,y) \quad such\ that \quad 1 \leq x \leq n, \quad 1 \leq y \leq n$$

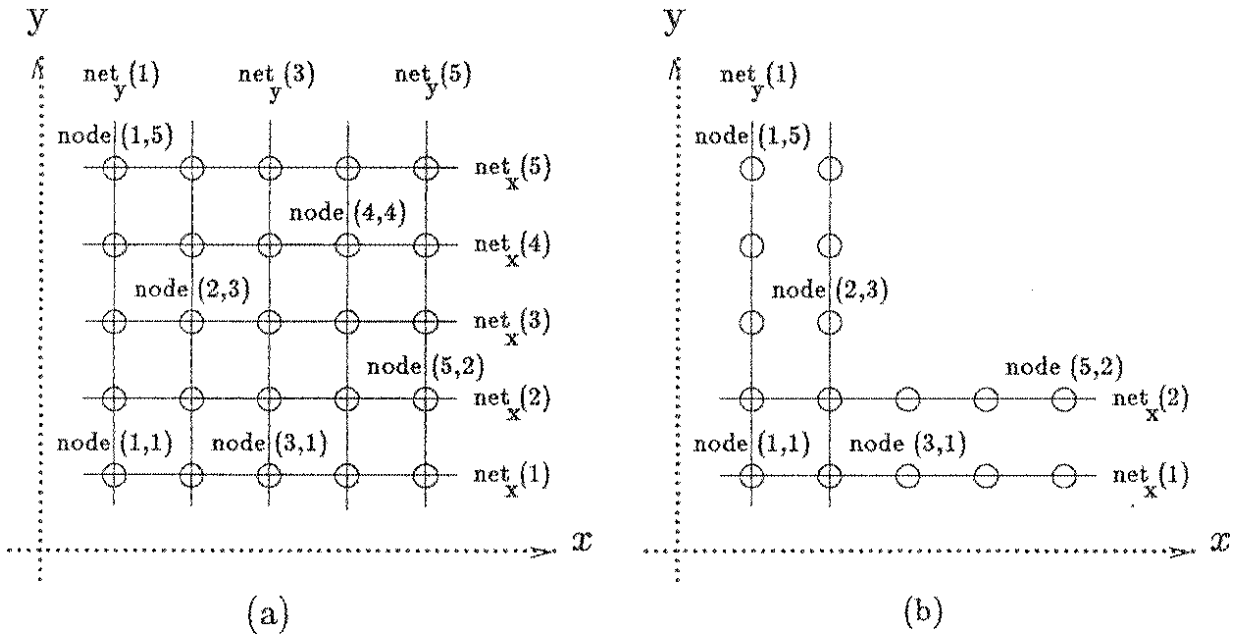

Figure 7.1: The Layout of a 2D Hypergraph

For a partial hypergraph, the nets in the x direction are indexed sequentially from 1 to k. The nets in the y direction are indexed sequentially from 1 to k. Each net has k nodes with two ports and a nodes with one port, such that $(n=k+a)$. A two–port node is identified by an ordered pair (x,y), a one–port node by either (x,∞) or (∞,y). Since the n nodes of each net are indistinguishable with respect to the center of the star, it is always possible to rearrange the network such that all nodes with two ports are within one square region close to the $x-y$ axis origin, as shown in Figure 7.1b.

In the discussion the following notation is used:

- $net_x(s)$ is the s^{th} net in the x direction. Thus, $node(r,s)$ is at the intersection of $net_x(s)$ and $net_y(r)$.
- **commonly known** refers to state information which has been broadcast from a known origin and has reached and been decoded by all the nodes of a net.
- $L_x(s)$ is the total communication load of $net_x(s)$, defined as the number of full buffers that are ready to be sent over the net. It includes only **commonly known** state information.
- $L_x(r,s)$ is the communication load of $node(r,s)$ via the $port_x(r,s)$ to $net_x(s)$; it includes only load information which is already **commonly known**.
- $L_y(r,s)$ is the load of $node(r,s)$ via the $port_y(r,s)$ to $net_y(r)$; it includes only load information which is already **commonly known**.

7.2. Two–Dimensional Node Interface

The node interface to a 2D network consists of two ports, buffers, a control unit, and a host interface, as shown in Figure 7.2. Each of the two ports is full–duplex and is

designed as described in Chapter 4. The control unit is the heart of the node interface and includes the functions buffer manager, routing controller, and synchronizer.

The buffers at each node are organized as one uniform finite collection of m units. Each buffer can contain one packet of data. Any of these can be attached for read or write operations to the host bus interface, or $port_x$, or $port_y$. Each buffer can function as a FIFO or a RAM and, in the FIFO mode, it can be attached to two interfaces, for simultaneous read and write operations.

7.3. The Delay for Updating State Information

One of the design objectives is to minimize the delay of updating state information and thereby improving the distributed algorithm performance. The 2D regular and partial hypergraph are analyzed in this sectio, in order to find out the average– and worst–case propagation delay of state information in the system. It is assumed that each net has n nodes, and each slot has r control minislots. In the following discussion it is assumed that the state information should propagate via two nets in 2D–R hypergraph and via three nets in 2D–P hypergraph.

It is assumed that the control messages sent during the CMSs have a predefined structure, with a specific field (few bits) for each parameter. Therefore, a state parameter (e.g., load) which arrives at the node via $port_x$ can be transmitted via $port_y$ during its next CMS.

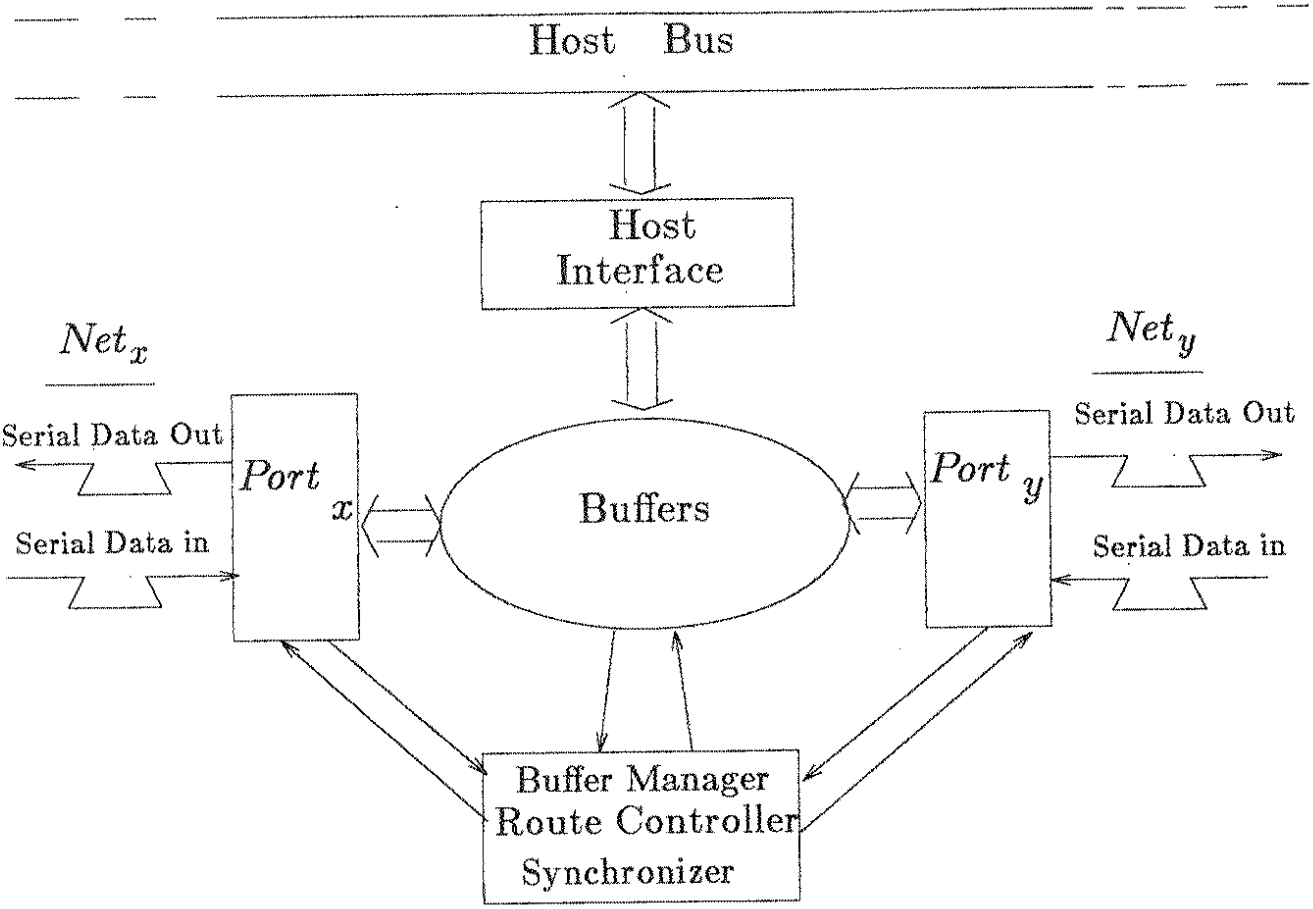

Figure 7.2: Functional Description of 2D Interface

7.3.1. The average delay

If a uniform use of the control minislots is assumed, then a node sends control information via its CMS every

$$l = \left\lceil \frac{n}{r} \right\rceil \ slots.$$

Thus, on average, new state information is sent with a delay of $\frac{1}{2}l$ and will reach to all the nodes on a net after an additional $f+1$ slots. The maximum distance between

any two nodes of 2D–R is two nets. Therefore, the average delay for state information to reach all the 2D–R nodes is

$$t_{prop-2D-R} = 2(\frac{1}{2}\, l + f + 1) = l + 2(f+1) \;\; slots.$$

In the 2D–P the information should propagate via three nets; therefore, the average delay is

$$t_{prop-2D-P} = 3(\frac{1}{2}\, l + f + 1) = \frac{3}{2}\, l + 3(f+1) \;\; slots.$$

7.3.2. Upper–bound on the delay

In the worst case, the state information is transmitted after l slots; therefore, the upper–bound of the propagation delay through the system is $2l + 2(f+1)$ slots for 2D–R, and $3l + 3(f+1)$ slots for 2D–P.

The upper–bound parameter is important for the design of protocols that operate in an open loop mode, i.e., distributed algorithms which use time stamps as a substitute for explicit acknowledgements, as in the mutual exclusion example of Section 6.4.

7.3.3. Number of messages

The total number of control messages that are exchanged is an important criterion in the evaluation of a distributed algorithm. Since the system is globally synchronized by events, and assuming that each event is time stamped, then the order of the distributed events is preserved. Therefore, in many cases the control messages and their timing information can be sufficient for the distributed algorithm. One aspect of the complexity of distributed algorithms, which are based on **common knowledge**, can be measured by counting how many control messages are exchanged.

In order that a state parameter will reach all the system's nodes, the parameter is sent in a control message over one net and then over all the nets which are orthogonal to the original net. Thus, the number of control messages which are needed for propagating a certain state information (e.g., mutual exclusion request) to all the system's nodes is $n+1$ for 2D–R. For 2D–P the maximum distance is three, and a parameter propagates first via one net, then on k orthogonal nets, and then on $k-1$ parallel nets ($1+k+k-1 = 2k$ nets).

All the nodes of a net monitor the activity on the net in the same way and in a synchronous manner. Thus, the distributed activity can be measured by the total number of nets rather than by the total number of nodes. Therefore, the complexity of the synchronous hypergraph is twice the square root of the total number of nodes ($2\sqrt{total-nodes}$). This complexity is much lower than a point–to–point network, which is on the order of the total number of nodes. Thus, it exhibits the potential of the synchronous hypergraph for the implementation of complex distributed algorithms.

7.4. Routing Algorithms for a 2D Regular Network

The basic parameter for routing is the total load, which is **commonly known** on a net, rather than the total load of a node. An optimal routing algorithm will balance the load among all nets and thereby maximize the communication capacity of the system. The use of the net load as a parameter decreases the complexity of the routing algorithm relative to algorithms that use the load of each individual node. For example, the number of different parameters for an algorithm which considers global information of

nets is only on the order of the $\sqrt{total-nodes}$.

Several methods for routing algorithms will be discussed in this section. These methods use the load information that is periodically exchanged by the control messages. One of the design objectives is to make these algorithms transparent to the host, i.e., the routing algorithms are performed by the network interface, independent of the system's software. Each node monitors its neighborhood (the activity on its two nets), and performs the routing algorithm without exchanging special messages with other nodes in the system.

The different routes on a 2D regular network may be classified into two types:

Primary route is the path with the least number of nets from one node to another. For the 2D–R hypergraph, the primary path has a length of one or two nets. For example, to get from $node(r,s)$ to $node(u,v)$ (which do not have a common net) there are two possible ways: (i) via $net_y(r)$ to $net_x(v)$, or (ii) via $net_x(s)$ to $net_y(u)$.

Secondary route is any route which is not a primary route. For the 2D–R hypergraph, a secondary route has a length of at least three nets.

7.4.1. The local balancing routing algorithm

This algorithm uses only the primary routes. Between every two nodes, in the plane of 2D–R, there are at most two primary routes. The routing algorithm determines which route to select. If the nodes have a common net the routing is via this net; otherwise, the criterion for selecting the primary route is by locally balancing the load of the two orthogonal nets, i.e., the data packet is sent via the net with the lesser load.

An active node is a node which uses its CMS, and by this asserts that "*I am Alive.*" Let $node(r,s)$ be the source node, and $node(u,v)$ be the destination node, then there are four cases:

Case 1 – $r \neq u$ and $s \neq v$, and $node(r,v)$ and $node(u,s)$ are active.

If $L_x(s) \geq L_y(r)$, otherwise route is via $net_y(r)$ and $net_x(v)$, else the route is via $net_x(s)$ and $net_y(u)$.

Case 2 – $r = u$ or $s = v$, the source and destination have ports to the same net, the packet is sent directly via this net.

Case 3 – $r \neq u$ and $s \neq v$, and only one of the intermediate nodes, $node(r,v)$ or $node(u,s)$, is active, then the packet is routed via the active node.

Case 4 – both intermediate nodes, $node(r,v)$ and $node(u,s)$, are not active, then a secondary route is selected.

The implementation of local load balancing is simple. In order to compute the total load of a net, the nodes report the relative change in their load. Thus, the computation of the total load of one net is done by adding (or subtracting) the load changes.

7.4.2. The class of secondary routing algorithms

The class of secondary routing algorithms considers both primary and secondary routes. The objective of these algorithms is to balance the load over all the network's nets, and as a result to maximize the utilization and to minimize the average delay. With each control message, additional load parameters are transferred, which give information on the load of the orthogonal net. Via its two ports, the node monitors

the activity on each of its two nets, and the information seen by $port_x$ is then transmitted over net_y, and vice versa.

In order to maximize the flow in the system and to minimize the average packet delay, the sum of the net's load along the route should be minimized.

Theorem: Maximum route length on 2D–R –

On 2D–R hypergraph, the secondary routing algorithm should not consider routes which are longer than three nets.

Proof:–

Let $node(r,s)$ be the source node and $node(u,v)$ be the destination node. The destination node can be accessed either via $net_x(v)$ or via $net_y(u)$. Assume that from some considerations $net_x(v)$ was selected; since the network is 2D–R hypergraph, a packet from the source node will intersect with $net_x(v)$ after traveling via one or two orthogonal nets, and the total length will not exceed three nets. The same argument is used if the $net_y(u)$ is selected. Thus, any route of length greater than three nets will just increase the sum of the loads along the route, Q.E.D.

In general, each port monitors the activity over its net and extracts three parameters:

(1) L – the current load on the net.

(2) L^{ave} – the average net load during the past h slots (h is to be determined by simulation of an actual system).

(3) U^{ave} – the average net utilization during the past h slots, which is the ratio between the full DMSs and h.

These three parameters are broadcast with the control messages via the other port onto the orthogonal net. As a result, all nodes receive these parameters from all the nets of the system. For maintaining global information, each node of the 2D–R should store $3(2n) = 6n$ parameters, and for 2D–P network, only $6k$ parameters. Thus, maintaining global routing information for 2D partial network can be quite reasonable.

7.5. 2D Partial Hypergraph as a Centralized Switch

In this section the partial hypergraph is viewed as a centralized switching network, such that the nodes with two ports, constituting the switch, can be located in one building, and the nodes with one port are distributed within an area of a few kilometers around the switch.

Figure 7.3 is an example of a partial hypergraph of four nets (NET A, NET B, NET C, and NET D); each net is a passive optical star and is operated as described in Chapter 5. This is the same as the operation of the nets of a 2D regular hypergraph network. Two nodes of each net have two ports and are placed in one area, constituting the centralized switch. Note that this centralized arrangement is done for practical purposes (such as maintenance), and not because of different operational principles. The nets in Figure 7.3 can be viewed as local area networks, and the centralized switch is the means for merging them together. Thus, 2D–P hypergraph is a method for connecting local area networks (optical nets) together. This approach is significant, since it enables the construction of a small system which can be gradually extended.

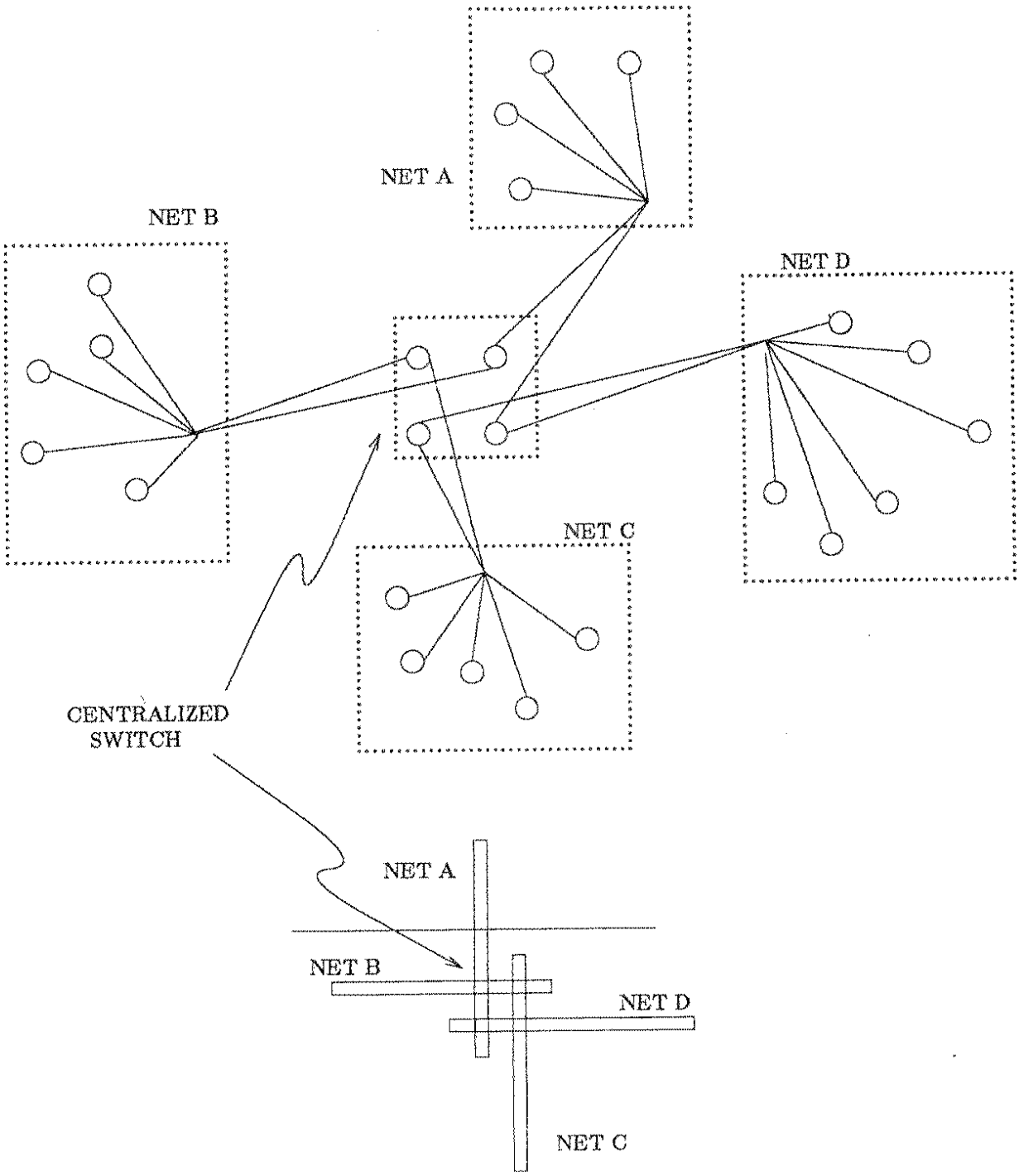

Figure 7.3: 2D–P Modeled as a Centralized Switch

In the analysis of a partial hypergraph as a centralized switch, three main issues are considered: (i) the bottlenecks and overflow prevention, (ii) conditions for ensuring maximum communication flow, and (iii) the effective bandwidth of a net.

7.5.1. Bottlenecks and overflow prevention

The nodes with two ports (the switch nodes) are potential bottlenecks, since a large portion of the communication traffic merges into them. Bottlenecks can occur under heavy load: i.e., when the number of full buffers which are **commonly known** is at least one. A bottleneck may result in buffer overflow and the loss of packets. In the following discussion it will be shown how overflows can be prevented.

In order to achieve these objectives the following conditions are determined, and then a theorem is proved.

Buffers Condition 1:

At a switch node, the set of buffers ($\{B_1, B_2, B_3, \ldots, B_m\}$) is divided such that no more than $\left\lfloor \frac{m}{2} \right\rfloor$ can be in the queue of $port_x$, and no more than $\left\lfloor \frac{m}{2} \right\rfloor$ can be in the queue of $port_y$.

Periodic Exchange Condition:

Each switch node can broadcast state information via its two ports using the control messages during its CMS, at most every l' time slots.

Overflow Hazard Flag Condition:

The control message has a flag which indicates overflow hazard. When a port turns the overflow hazard flag on, the other nodes on this net will not transmit

any more packets to it. The maximum latency or delay for turning on the overflow hazard flag, and for this information to get to all the nodes on the net, is

$$HAZARD\text{-}FLAG_{delay} = l' + f + 1,$$

which is the time until the next CMS, plus the time for this information to propagate through the net.

Buffers Condition 2:

The total number of buffers at each switch node is greater than $2l' + 4f + 2$, or that

$$\left\lfloor \frac{m}{2} \right\rfloor > l' + f + 1 + f,$$

the expression $l' + 2f + 1$ is the maximum number of packets which can arrive at that port from the time it has been decided locally (at the node interface), until the time in which this state information is **commonly known** on the net. Note that the second f is for the possible f packets which can be on their way, from the time the new state is **commonly known.**

The Algorithm for Turning On the Overflow Hazard Flag:

A switch node will turn on the overflow hazard flag when it broadcasts via $port_x$ over net_x, if the queue of full buffers ready to be sent via $port_y$ over net_y is greater or equal to $\left\lfloor \frac{m}{2} \right\rfloor - l' - 2f - 1$, and similarly for the overflow hazard flag which is broadcast over net_y.

Theorem:

The above conditions and algorithm are sufficient for preventing overflow.

Proof:

Whenever any of the two overflow hazard flags are turned on, the switch node can control the traffic which is generated locally by its host, by preventing the host from filling up buffers. Assume that, in the worst case, all the traffic from one net should be switched to the other net. Thus, no more than $l' + 2f + 1$ packets can arrive from the moment the hazard flag is locally turned on until this state is effective at all of the net's nodes. Therefore, using the above algorithm is sufficient to prevent overflow.

7.5.2. Ensuring maximum flow

The use of the overflow hazard flag may cause reduction in the performance of the system, i.e., the above conditions can reduce the effective communication capacity of the network. To prevent this additional condition is introduced:

Maximum Flow Condition:

When an overflow hazard flag is turned on over net_x, there should be a minimum number of full buffers in the queue to net_y, such that this queue will not become empty before the overflow hazard will be turn off. This minimum number is $l' + 2f + 1$, and the following inequality holds:

$$\left\lfloor \frac{m}{2} \right\rfloor - l' - 2f - 1 > l' + 2f + 1 \quad or \quad \left\lfloor \frac{m}{2} \right\rfloor > 2(l' + 2f + 1).$$

Theorem:

The above condition is sufficient to ensure maximum flow.

Proof:

Clearly, a reduction may happen if the queue of $port_x$ is empty, while the traffic over net_y to this node is halted by the overflow hazard flag. The time for turning off the overflow hazard flag is $l' + f + 1$ slots, and additional f slots are required for a packet to arrive at this node. Thus, $l' + 2f + 1$ full buffers ensure that the node will be able to send packets continuously.

The value of l' determines the total number of buffers in the interface. Therefore, in order to reduce this number, the ports of a switch node should have more access to a CMS, i.e., the switch node should have priority over other nodes in the use of the CMS. This will also improve the response time of the network.

7.5.3. Effective bandwidth

In this analysis, the system is modeled in order to find the number of nets required for constructing a system with some given parameters. In the following model the system is homogeneous; i.e., all the nodes are identical in their behavior, and all the nets have the same number of nodes.

The following parameters are defined:

- n_{net} – the number of nodes on each net ($n_{net} = k + a$).
- n_{sys} – the total number of nodes in the system $n_{sys} = {n_{net}}^2 - a^2 = k^2 + 2ka$.
- BW_{node} – the node's bandwidth, or the *average* traffic which the host of each node generates (it is the same for nodes with one or two ports).
- BW_{net} – the net's bandwidth, the *maximum* traffic which the net can transfer.
- b – the ratio between BW_{net} and BW_{node} $b = BW_{net} / BW_{node}$.

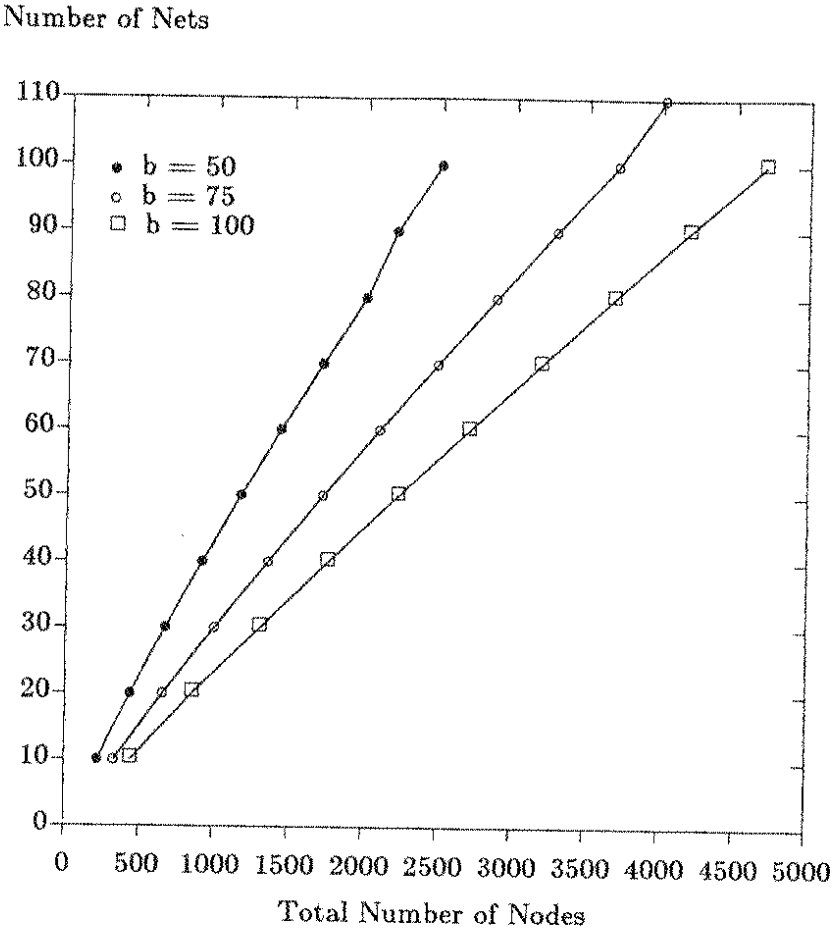

Figure 7.4: The Number of Nets as a Function of the Total Number of Nodes (b is fixed)

• d_{ave} – the average distance between two nodes of the network (measured in the number of different nets in which a message should travel). In the 2D–P network the possible distances are 1, 2 or 3 nets.

In order to support the average communication in the system, the following inequality should be satisfied:

$$2kBW_{net} \geq n_{sys} BW_{node} d_{ave},$$

i.e., the total communication capacity should be greater than or equal to the total traffic generated by the system's nodes, which should be multiplied by the average

distance (number of nets) each packet travels.

The average distance in the homogeneous 2D–P is

$$d_{ave} = \frac{2ka\left[1(k+a-1)+2k(k+a-1)+3a(k-1)\right] + k^2\left[1(2k+2a-1)+2((k+a-1)^2-a^2)\right]}{n_{sys}^{\ 2}}$$

Using the above expressions, it is possible to show various relationships among the different parameters (see Figures 7.4 and 7.5). The parameter b represents the

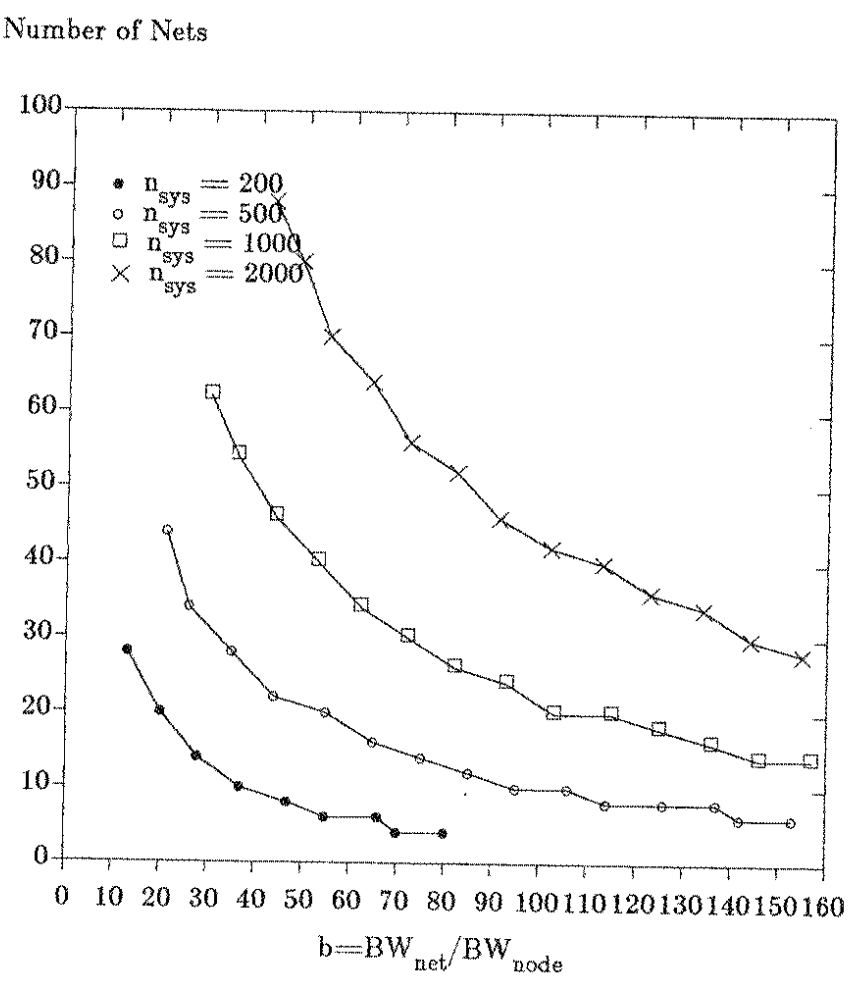

Figure 7.5: The Number of Nets as a Function of the Ratio $\frac{b=BW_{net}}{BW_{node}}$

state of a *given technology*, and one can assume that its value will increase in the future, with the result that a single net will be able to support the communication of more or busier nodes. For a larger value of b and a in a fixed size system, the number of nets is smaller. Also for a fixed value of b, as the system size enlarges, the net size enlarges as well. It is also apparent that the number of ports on each net of the partial network is larger than $\frac{b}{2}$ but smaller than b.