1 Introduction

1.1 Parallel systems

A *parallel system* consists of a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of *processors* and a *communication mechanism*. We distinguish between three types of communication mechanisms:

- **Communication via a shared memory**

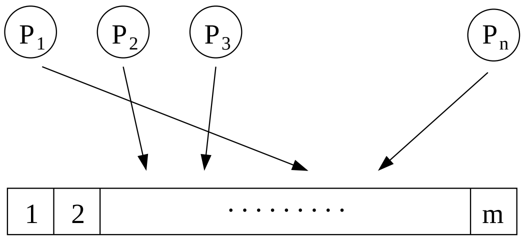

Figure 1: An example of a PRAM.

We are given a shared memory of m cells, numbered from 1 to m. Every processor has read and write access to any cell of the shared memory. Each access takes one time step. We call such a system a *parallel random access machine* or *PRAM*. Its advantage is that it is very easy to develop programs within this model. However, building a PRAM may be extremely difficult and inefficient.

- **Communicaton via links**

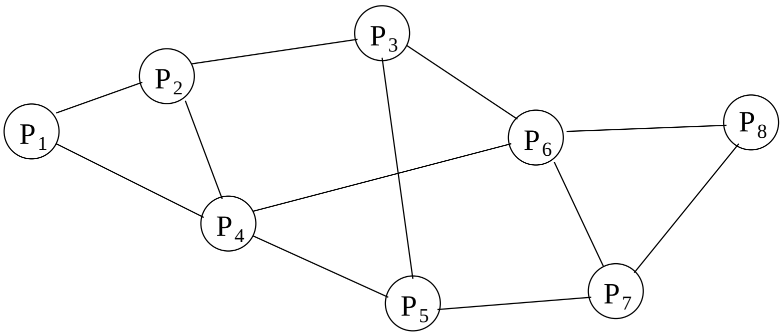

Figure 2: An example of a communication network.

In this case the processors are connected via a *communication network* whose topology can described as an undirected *communication graph* $G = (\mathcal{P}, E)$. If $\{P_i, P_j\} \in E$, then there is a communication link between P_i and P_j which allows P_i to send a message to P_j and vice versa. Unless stated otherwise, we will always assume that all messages are of the same size (in this case they are usually called *packets*). Furthermore, every link needs one time step to forward a message and can transport at most message in each direction at a time.

The number of links that can be in a network of n processors is $\binom{n}{2}$. Thus, for 1000 processors there can be up to approximately 500.000 links. Since it is unrealistic to assume that so many links are available, it may not be possible to send a packet directly from its source to its destination. Instead, the packet may have to traverse several links to do so.

- **Communication via a bus**

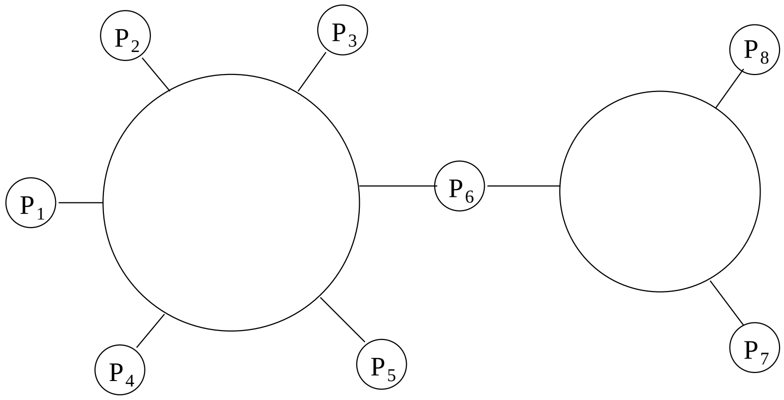

Figure 3: An example of a bus network.

In a *bus network* the processors may be connected via several buses. A bus is a communication line that can be shared by two or more processors. Only one processor is allowed to send a packet to a bus at a time step. However, all processors connected to a bus are able to reveive the packet placed on the bus at any time. We will not consider this model in this lecture.

1.2 The communication model

We will always assume that the processors work in a *synchronous* way, that is, all processors start their next step at the same time. In each time step a processor P is able to perform the following tasks:

- create new packets (if any)
- send out packets
- receive incoming packets
- absorb packets whose destination is P

We will not count internal computations.

1.3 Overview of the lecture

In this lecture, we will concentrate on the following problems:

- **Routing**
  The problem of sending messages in a network from their sources to their destinations is called *routing*. Routing involves two basic disciplines:
  - **path selection problem:** how to select paths for the packets?
  - **packet switching problem:** in which order to send the packets along their paths?
- **Simulation between networks**
  How efficiently can one network simulate a step of the other?
- **Simulation of a shared memory**
  How efficiently can programs written for PRAMs be simulated on networks?

1.4 Routing in a cycle

A *cycle* is a graph $G = (\mathcal{P}, E)$ with

$$E = \{\{P_i, P_{i+1}\} : \ i \in \{1, \ldots, n-1\}\} \cup \{\{P_n, P_1\}\} \ .$$

Suppose that we are given a set of m packets represented as source-destination pairs (s_i, d_i), $i \in \{1, \ldots, m\}$. Initially, all packets are stored at their sources. In order to send the packets to their destinations, the nodes choose the shortest path and use the furthest-to-go (or FTG) rule. FTG means that if two packets contend to use the same link in the same direction at the same time step, then the one that has the most edges to go is preferred. If several packets have the same number of edges to go, then the one of maximum index i wins. For this strategy we can show the following result.

Theorem 1.1 *For any set of m source-destination pairs our strategy needs at most $m+\lfloor n/2 \rfloor - 1$ time steps to send all packets to their destinations.*

Proof. Let $P = (p_1, \ldots, p_m)$ denote the set of all packets, and let the *rank* $\text{rank}_v(p_i)$ of packet p_i at node v be defined as the number of edges p_i still has to traverse from v plus $i/(m+1)$. According to our preference rules it must hold for any two packets p, q at some node v where p is preferred against q that

$$\text{rank}_v(p) > \text{rank}_v(q) \ .$$

We will need this observation later.

Now, let us use a proof method called *backwards analysis*. We start with the last packet that reached its destination. Let this packet be called $q_1 \in \{p_1, \ldots, p_m\}$, and let its destination be called v_0. We follow q_1 backwards in time till it was delayed by some other packet, say q_2, at some node v_1. Let ℓ_1 be the number of edges traversed from v_0 to v_1. We continue to follow q_2 backwards in time until it was delayed by some other packet, say q_3, at some node v_2. Let ℓ_2 be the number of edges traversed from v_1 to v_2. In general, we will follow packet q_i backwards in time until it was delayed by some other packet, say q_{i+1}, at some node v_i. Let ℓ_i be the number of edges traversed from v_{i-1} to v_i. If we reach a packet q_i that has not been delayed by any other packet, we simply follow it backwards in time until it reached its source node, called v_i.

Assume now that we touched s packets $q_1, \ldots, q_s$ in the course of this argument. Using our observation above, we can state the following three facts:

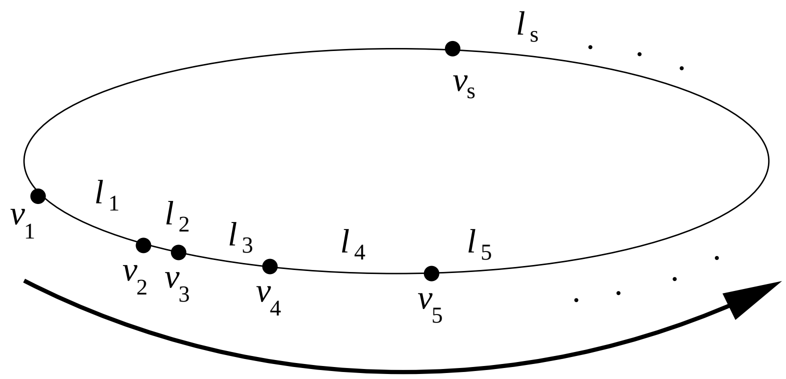

Figure 4: A possible outcome of the backwards analysis.

1. $\text{rank}_{v_0}(q_1) \geq 0$
2. $\text{rank}_{v_i}(q_{i+1}) > \text{rank}_{v_i}(q_i)$
3. $\text{rank}_{v_i}(q_i) = \text{rank}_{v_{i-1}}(q_i) + \ell_i$

We will use these facts to prove the following lemmas.

Lemma 1.2 $\sum_{i=1}^{s} \ell_i \leq \lfloor n/2 \rfloor$.

Proof. Using facts (1) to (3), we obtain that

$$\text{rank}_{v_s}(q_s) \geq \sum_{i=1}^{s} \ell_i$$

However, since $\text{rank}_{v_s}(q_s)$ must be smaller than $\lfloor n/2 \rfloor + 1$ due to a maximum path length of $\lfloor n/2 \rfloor$ and since the ℓ_i are integers, the lemma follows. $\sqcap$

Lemma 1.3 $s \leq m$

Proof. Due to fact (2) and the fact that if $\text{rank}_v(p) > \text{rank}_v(q)$ for two packets p and q then this relationship also holds for all other nodes visited by p and q, it holds that no packet can appear more than once in the backwards argument. Hence, $s \leq m$. $\sqcap$

Obviously, the total runtime of our strategy is equal to the time covered by the backwards argument. Since the number of time steps covered by this argument is exactly $\sum_{i=1}^{s} \ell_i + s - 1$, the two lemmas above imply a runtime of at most

$$\lfloor n/2 \rfloor + m - 1 \ .$$

$\sqcap$

Theorem 1.1 is worst case optimal, since a routing problem can easily be set up that requires $m + \lfloor n/2 \rfloor - 1$ steps (see Figure 5).

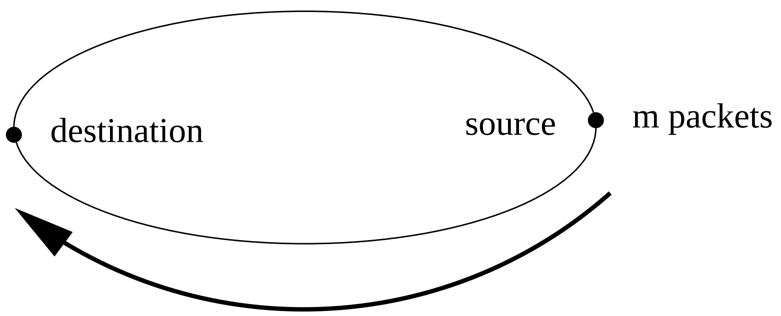

Figure 5: A worst case situation.

1.5 References

- C. Scheideler. Universal Routing Strategies for Interconnection Networks. Lecture Notes in Computer Science 1390, Springer, 1998.