|  |  |  |
| --- | --- | --- |
|  | Distributed Algorithms – Project 1  Winter term 2017/18 | Danh Le Phuoc  & Qian Liu  ODS |
| *Exercise Sheet 1* | | |

Kosta Ivancevic (397172)

Sebastian Schasse (318569)

Poonam Kumari Roy (387359)

Umang Ashoke Kumar (388193)

**Exercise 1.1 Definitions**

1. **What is the difference between a distributed system and a parallel computer?**

Distributed system is defined as a graph DS = (N, E) ; where N denotes set on nodes and E denotes set of edges in the graph. N corresponds to set of individual processing units and E corresponds to set of links, which enable the communication between different machines. Tasks which are solved by the distributed system are solved on multiple physically remote machines, communicating through message exchange.

On the other hand, parallel computer is tightly bound to one physical machine, which is able to have multiple processing cores, or individual processors. Local memory is the shared resource which is used for mutual communication between processors. The task is distributed to these cores, which solve individual fragments in parallel, with the paradigm much like distributed system.

By taking into account similar approach in “Divide and Conquer” paradigm (Divide et impera), we can conclude that parallel computer is a subset of distributed system. It is limited to one machine whereas, distributed systems are not limited with regards to number of machines. Parallel computer has local communication between processes, while distributed system has network enabling the machines to be physically remote. Finally, as a result of unlimited number of nodes and the possibility of introducing new, and excluding non-functional nodes, distributed systems have higher processing power, are more modular, and scalable and are more immune to individual failures.

1. **Why do we use distributed systems although they are complicated? Give examples?**

Distributed systems are convenient in multiple reasons some of which are listed below:

Processing power one of the main motivations for introducing such system. A graph comprised of a large number of nodes, each of which process data in parallel with others introduces large increase in computational power. Many difficult tasks are overcame using distributed systems, rather than traditional multicore and multiprocessor systems.

Modularity in this sense can have multiple meanings, each of which corresponds to different positive aspect of distributed systems. Adaptivity to large variety of different architectures is beneficial, as each architecture can be exploited. Also, dynamical change in the architecture is the concept, more and more present in todays distributed systems. Polymorphism, as this can be called brings adaptivity and modularity to a whole new level in terms of exploiting maximal computation power.

Scalability is vital aspect of any modern system. It enables changes in terms of nodes and connections, increases fault tolerance, and is good basis for system improvements.

1. **What are the differences between the synchronous model, the asynchronous model and the atom model?**

Synchronousmodel of a distributed system defines the upper and lower bounds for all the processes on individual nodes. This means that the information about the specific tasks (start time and end time) is available. The communication between nodes is also bounded, meaning that the delay time, needed for a message to be received is known in advance. Clock rates of individual nodes are known. Taking this into account, we are able to use timeouts as detecting failures. The expectancy of knowing all the time parameters of such complicated system is a long shot, so this model is mostly used in modeling and simulation.

Asynchronous model of a distributed system is contrary to Synchronousmodel. The delay in processing individual tasks is mutually independent and unknown. Most importantly, the messages, which carry information between nodes and enable parallelism and distribution, have nondeterministic delays, as a result of network traffic. Timeouts have no secure information. Reason for timeout may be one of the following:

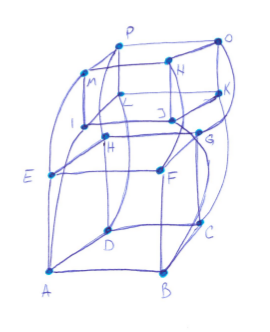
* The message is delayed longer than usual (traffic overflow on the network)
* The message got lost (packet is dropped in the network or corrupoted)
* The sending process has send the message later than usual (congestion control)
* The sending process has crashed before it could send the message

Asynchronous models make no assumptions on process execution speed, communication delays or clock drift rates.

Atom Model makes assumptions that Processing is synchronous and communication is asynchronous.

**Exercise 1.2: Topologies**

**Consider a hypercube with dimension d.**



1. **Given two arbitrary nodes u and v from the hypercube. How many shortest paths are there between u and v?**

One approach to this problem is to enumerate each edge of the hypercube in binary. As we can model each hypercube dimension d from multiple hypercubes dimension g (g<d), binary codding corresponds well, as can be seen from figure above. Weight of the bits corresponds to adequate hypercube.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | 0 | 0 | 0 | 0 |
| B | 0 | 0 | 0 | 1 |
| C | 0 | 0 | 1 | 1 |
| D | 0 | 0 | 1 | 0 |
| E | 0 | 1 | 0 | 0 |
| F | 0 | 1 | 0 | 1 |
| G | 0 | 1 | 1 | 1 |
| H | 0 | 1 | 1 | 0 |
| I | 1 | 0 | 0 | 0 |
| J | 1 | 0 | 0 | 1 |
| K | 1 | 0 | 1 | 0 |
| L | 1 | 0 | 1 | 1 |
| M | 1 | 1 | 0 | 0 |
| N | 1 | 1 | 0 | 1 |
| O | 1 | 1 | 1 | 1 |
| P | 1 | 1 | 1 | 0 |

If we take two edges, u and v, the length of a shortest path corresponds to the number of different bytes in the corresponding binary representation. For Example, let as assume that we want to know the length of a shortest path between P and C. They differ in 3 bits. With this in mid we can calculate the number of different shortest paths as 3! = 6. All sets of nodes in between are {(O,K),(L,K),(H,G),(H,D),(L,D),(O,G)}

In general, if the binary representations differ in n bits, then number of shortest paths is equal to n!

1. **How many different node pairs are there, that are connected with shortest path of length k?**

If a hypercube is dimension d, then the number of nodes is nodes. Each of these nodes is connected with d other nodes. The nodes are represented in binary, as given above. For each node, there is a shortest path to any other node, which can be measured through counting different bits in binary representation. If we are given a node and the shortest path k, we can compute the number of nodes which are on a k distance, by calculating the number of combinations of k elements from the set of d elements. We do this procedure for each node to get the number of nodes with shortest path k. It is important not to repeat node pairs, so we divide by 2.

where N = and represents number of nodes

1. **Given the broadcast algorithm from the lecture to produce spanning trees on the hypercube. How many different spanning trees could be generated from the same start node by varying the dimension used for sending?**

When the broadcast algorithm would be applicable in parallel to hypercubes I general,then it needs to be applicable to a hypercube of dimension one. Let’s examine that: We start with node 0 and send a message to node 1. In parallel we would now need to send a message from node 0 again to any neighbor. The problem is that we can only send the message to neighbor 1. The edge is already used by the previous broadcast. That means that even for the very simple hypercube with dimension one, sending broadcasts in parallel fails.

**Exercise 1.3: Distribution of Information**

1. **Implement the Flooding algorithm with acknowledgments using the simulation framework teachnet (provided at the ISIS course website). After finishing the implementation, test your algorithm on a ring topology and compare the amount of messages sent with the “Broadcast on Unidirectional Rings” using the same topology.**

The “Broadcast on Unidirectional Rings” sends n messages whereas the Flooding algorithm sends 4e-2n+2 messages. In a ring topology e==n, so that we can say it sends 2n+2 messages. That means in a ring topology, the Flooding algorithm is worse than the Broadcast.

1. **Implement the Echo algorithm using teachnet and evaluate the correctness of the amount of messages sent to be (2e) on various topologies. Highlight all edges that are part of the spanning tree.**

We evaluated Echo algorithm sending 2e messages for the following topologies: ring, star, tree, bintree, binxtree, 4x4 mesh, 4x4 torus, complete and hypercube.

1. **An improvement of the Echo algorithm has been introduced (see lecture) that sends a set of tabu node IDs together with an explorer. Examine the behavior of the algorithm compared to the classical Echo algorithm in terms of message reduction under the assumption of the following topologies:**
   * + 1. **Bidiredtional ring with n nodes**

As our implementation shows, on ring toplogies, we don’t save any explorer messages. The amount of messages is the same as with the original Echo algorithm.

* + - 1. **Binary X-tree of height h (with 2h+1 -1 nodes, cf. lecture 2, slide 29)**

Our implementation shows again, that the improvement breaks the algorithm for Binary X-trees. The reason is, that the improvement saves explorer messages, those explorer messages are missing in the counts, though. That means that no confirmation messages are sent. To correct that, you would need to change the specification of the explorer counts as well.

**Additional notes and assessment:**

* **important parts of the implementation have to be annotated with comments**
* **each exercise has to be completed handled in teams of 3-4 students**
* **the exercise sheet is successfully completed, if exercise 1 was presented and the solution was explained satisfactorily**