Distributed Algorithm Excersie 1 (Group 06)

By *Group 06*
1) Darshan Hingu 380584
2) RaviPrasad Marike Ramesh 387219
3) Swamy, Seema Narasimha 384418
4) Yuchun Chen 387275

1.1

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

The connection between parallel, distributed, and concurrent systems is that they're all terms that describe systems made up of computers and software that are doing more than one thing at a time. The differences lie in why and how they do it. Parallelism, or parallel code, or parallel systems talks about how to take a given system, and make it run faster by breaking into pieces that can run simultaneously. So suppose you want to do something really complicated. It's got three steps, A, B, and C. A and B each prepare things for C, but they don't interact with each other at all. Then you can run that by running A, waiting until it's done, then running B, waiting until it's done, and then running C. Or, if you've got a spare processor, you could run A and B simultaneously, and then when they're done, run C. When you're making a program run parts at the same time in order to make it run faster, then you're doing parallelism.

Distribution is talking about systems that are made up of multiple physical parts connected by a communication network. It's fundamentally a study of how to build systems where the system itself is broken into physical pieces, which may be located in different places, have a variety of constraints on communication, etc. If your system is is specifically designed to be run as multiple programs running simultaneously on on many different pieces of hardware, but behaving in some sense as a single system, then you're doing a distributed system.

The example of each:

- Weather forecasting software is usually parallel code. Doing the computational fluid dynamics work to generate accurate predictions requires an enormous amount of computation, and dividing it up among many processors makes it run at a (more) reasonable rate.
- An example of a distributed system would be a piece of software like writely, which is a word processor that runs inside of a web browser. In writely, you can edit a document in your web-browser, and you can share editing with multiple people – so you can have three or four web browsers all editing the same document. In terms of the system, the web browsers are each running little Java applications that talk to the server and each other by messaging; and they have absolutely none of the code for actually changing the document. The server has no code for doing things like rendering the UI, but it communicates with the clients to get receive and process edit commands, and send out updates so that all of the UIs are rendering the same thing. The entire design of the system is built around the idea that there are these multiple pieces, each running on different machines.

Distributed systems are a collection of autonomous computers linked by a network and using software to produce an integrated computing facility. Size is a distributed system? Local Area Network (10's of hosts) Metropolitan Area Networks (100's of hosts) Wide Area Networks (internet) (1000's or 1,000,000's of hosts) Example: Wide area network applications email - electronic mail WWW - world wide web netnews - group discussions on single subject gopher - text retrieval service multimedia/teleconferencing over networks Multimedia information access and conferencing applications bbs - bulletin board systems If it does not complete during its time slice, it is paused, another computer begins or resumes, and then later the original system is resumed. Here execution doesn't happen at the same instant. Parallel systems are collection of computers, in which if one computer is doing one aspect of some computation, other can work on another aspect. All of them can share same set of data, but work proceeds parallel. Advantage of this kind of system is to circumvent physical and mechanical constraints.

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

1. Scale – We will never get away from how large data sets are becoming these days. With consumers creating so many different sets of data from purchase, viewing, or play history, we are all creating data at break-neck speed. Any large-scale employer will need top-notch developers who possess the ability to create systems to process and store this data – and make sense of it.
2. Live Site – How do you make your systems or platforms fault-tolerant? Who handles the situation when datacenters, containers, or machines go down? How long does it take to release a build? Who is on call? When you're not live and online, you cease to exist. You cease to monetize. The needs for expertise in this field are deep: the process, escalation path, SLA's. The general "know-how" on how to get everything back up, triage, and solve tough problems. As a result, these Developer positions are always critical to any business or service that is online and has a large cloud infrastructure.
3. Next Generation/v2 – You think that service is cool now, but the team who shipped it is already hard at work on v2, fixing bugs and improving the customer experience. Building large-scale platforms to test or experiment these new features or services requires a special Developer who understands the importance of analysis and experimentation. In essence: making sure new features work in production, out in the real world.
4. Latency – Everyone has experienced latency. You're waiting for your movie to load, or the replay of the Super Bowl isn't streaming as fast or as clearly as you want. Page load time affects customers' experience. Companies will look for Developers to help them trim down latency: It's critical to ensure they have the top-of-the-line containers and equipment, and can focus on other efficiencies.
5. BI – Business Intelligence platforms – You're processing, storing, and performing transactions. What do you do with the data from what you ship? You want to create solutions to make sense of the data, isolate patterns or trends, and make business decisions based on the data you analyzed. It takes a Developer to create these elegant, scalable solutions. The demand for this type of Developer will continue to grow as the data, and demand for data analysis, grows.

iii. What are the three V´s for BigData, name and explain them.

Big Data – 3 Vs of Big Data – Volume, Velocity and Variety.

Volume

We currently see the exponential growth in the data storage as the data is now more than text data. We can find data in the format of videos, musics and large images on our social media channels. It is very common to have Terabytes and Petabytes of the storage system for enterprises. As the database grows the applications and architecture built to support the data needs to be reevaluated quite often. Sometimes the same data is re-evaluated with multiple angles and even though the original data is the same the new found intelligence creates explosion of the data. The big volume indeed represents Big Data.

Velocity

The data growth and social media explosion have changed how we look at the data. There was a time when we used to believe that data of yesterday is recent. The matter of the fact newspapers is still following that logic. However, news channels and radios have changed how fast we receive the news. Today, people reply on social media to update them with the latest happening. On social media sometimes a few seconds old messages (a tweet, status updates etc.) is not something interests users. They often discard old messages and pay attention to recent updates. The data movement is now almost real time and the update window has reduced to fractions of the seconds. This high velocity data represent Big Data

Variety

Data can be stored in multiple format. For example database, excel, csv, access or for the matter of the fact, it can be stored in a simple text file. Sometimes the data is not even in the traditional format as we assume, it may be in the form of video, SMS, pdf or something we might have not thought about it. It is the need of the organization to arrange it and make it meaningful. It will be easy to do so if we have data in the same format, however it is not the case most of the time. The real world have data in many different formats and that is the challenge we need to overcome with the Big Data. This variety of the data represent Big Data.

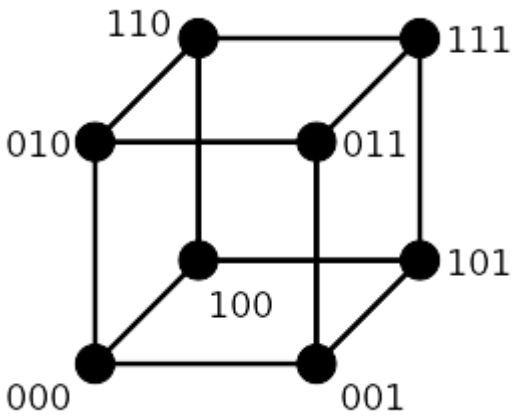
Exercise 1.2: Topologies

Exercise 1.2 Topologies

Part i:

According to Hammings distance the number of shortest path between arbitary nodes can be calculated as follows:

Lets consider a 3 Dimension Hypercube



Lets consider our 2 arbitary farthest diagonally nodes as `010` and `101`.

So according to Hamming's Distance we have to perform XOR operation between these two nodes which gives us i.e `010 ^ 101 = 111`

The paths between our 2 assumpt nodes.

```
010 => 011 => 001 => 101
010 => 110 => 111 => 101
010 => 000 => 001 => 101
010 => 000 => 100 => 101
010 => 110 => 100 => 101
010 => 000 => 001 => 101
010 => 011 => 111 => 101
```

So, we found that in 3-D Hypercube there are 7 different shortest path between 2 arbitary nodes.

The general formulation for finding shortest path using Hamming's Distance for n-dimensional hypercube is as below:

If the two vertices bit strings differ in k-spot the number of shortest paths will be equal to ***2^k - 1***.

Parti ii:

Lets consider the path length **'k'** as 3 between 2 nodes. In the bit string of vertex V If 3 bits are varying then according to Hamming distance lenght between two nodes will be 3. So based on Hamming distance there will be 2^3 - 2 = 6 different node pairs for an 3-D Hyperbcube. In generalized manner it is ***2^k-2***.

Part iii:

Each node in the hypercube of dimension D has exactly D partner nodes that it is directly connect to, therefore the number of possible direction for the initiating node is D, the second node only has D-1 possible direction left.(d partners nodes the sendinbg direction. d (d -1) .... * (d 1 (d -1)) i.e ***D!***

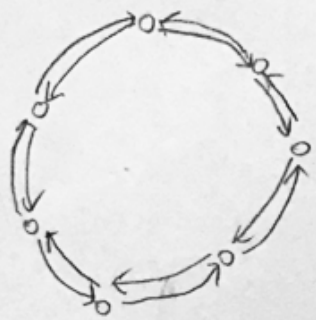
Part iv:

Yes, it is possible to do multiple broadcast in parellel on the spanning tree produced on Hypercube. The unit time model for this ***(2^d -1) / d***.

Part III

(a) Bidiredtional ring with n nodes

Traditional echo

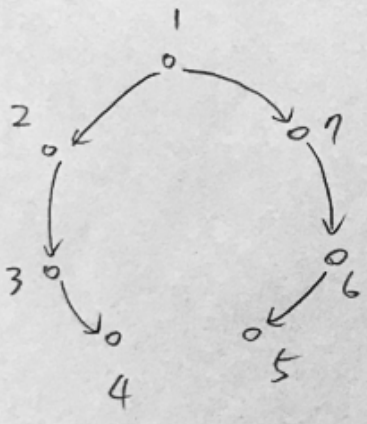


2n messages

Include confirmed messages

⇒ 2(2n) = 4n messages

Improved echo



$Z = \{1, 2, 7, 3, 6, 4, 5\}$

n-1 messages

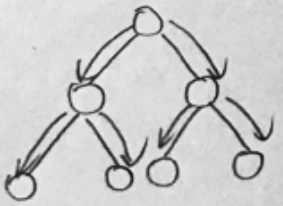
Include confirmed messages

⇒ 2(n-1) messages

Improved echo help for message reduction on Bidiredtional ring case

(b) Binary X-tree of height (with 2h+1-1 nodes)

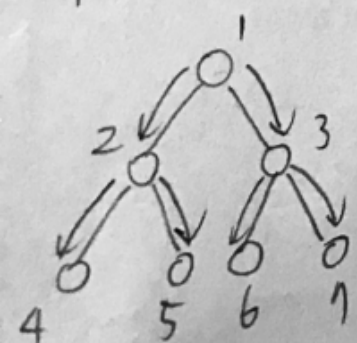
Traditional echo



6messages ⇒ n-1 messages

Include confirmed messages ⇒ 2(n-1) messages

Improved echo



$Z = \{1, 2, 3, 4, 5, 6, 7\}$

6 messages ⇒ n-1 messages

Include confirmed messages ⇒ 2(n-1) messages

Improved echo does not help for message reduction on Binary X-tree case