January 15, 2018

Problems that are assigned to me as per rule of formula

Last Two Digits: : 05
k = 5%5 = 0
k $\rightarrow k+1$ = 1
k $\rightarrow k+2$ = 2
Hence, I have to solve Q1 and Q2 in each category

1 About Interconnection Networks

1.1 Q1: Prove that torus is node-symmetric

Answer:
Torus Meshes with the "WrapAround" connections are called Torus. Torus can be 1D,2D,3D,..,**N**D. with degree of 2n+2 where n=0,1,...,N-1 respectively.
Node-Symmetry: Node-Symmetry means that every node in the graph has same degree or every node in the graph is connected to same number of other nodes or every node has same number of neighbors.
We have to prove that the torus of any dimension **N** is a node-symmetric or every node of N torus has same number of neighbors. We have to prove the following.
Deg(ND - Torus) = 2n+2 where n= N-1 (i)
The above equation (i) means that the increasing dimension **N** of torus the number of neighbors for each node increases by 2.
We can prove this fact by induction on the dimension of **N** of torus the number of neighbors for each node increases by 2.

We can prove this fact by induction on the dimension **N** of torus. And prove that (i) is true for each dimension.

Proof by Induction:

Base Case: We can start the proof by taking two base cases $N = 1$ and $N = 2$. From (i) we get,

when N = 1 then n = 0

Deg (1D - Torus) = 2(0) + 2 = 2

when N = 2 then n = 1;

Deg (2D - Torus) = 2(1) + 2 = 4

Induction: Assume that (i) is true for N=k, then we have to prove that it is also true for , N = k+1

When N=k then n=k-1;

Deg (kD - Torus) = 2(k-1)+2 = 2k

Result: N = k +1 then n = k;

Deg([k+1] D-Torus) = 2(k+1-1) + 2 = 2k + 2

Above equation has same form as (i) when we applied our induction hypothesis i.e n=k, which means our induction hypothesis is true so we can say that torus is node-symmetric with

Deg(ND-Torus) = 2n+2

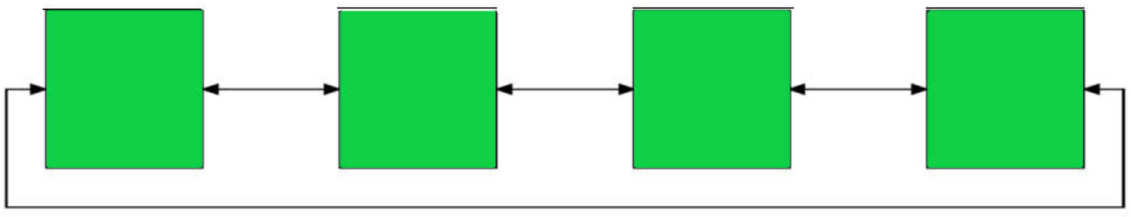
Proof by Contradiction:

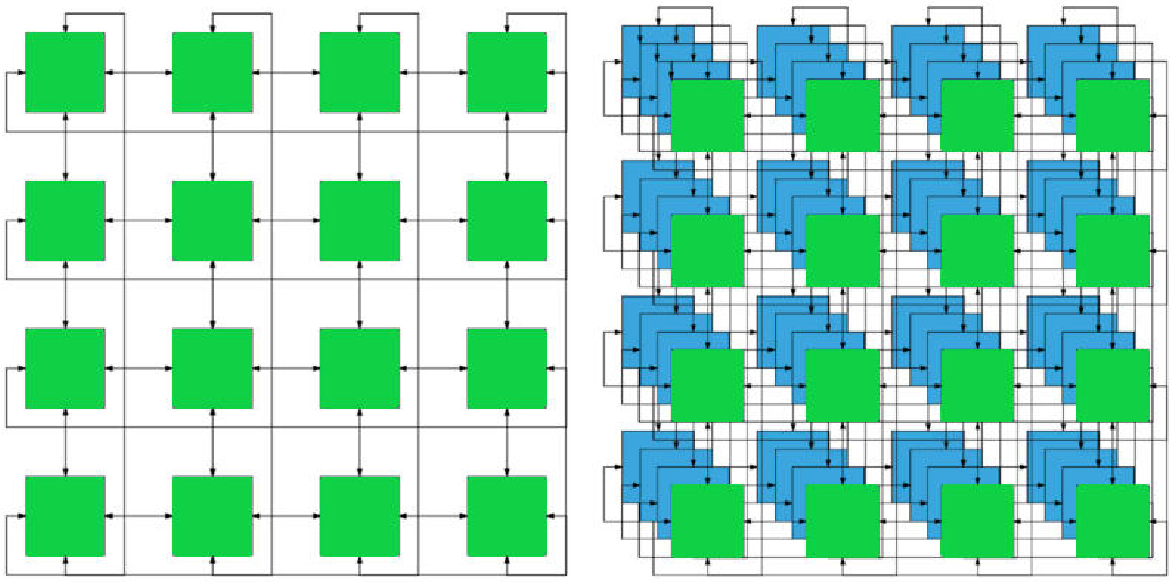
Torus is not node-symmetric.

if torus is not node symmetric than the different nodes will have different number of neighbors. But for a graph to be torus each node should have

Degree = 2N

which makes the graph symmetric. if the torus is not node symmetric then that would be some other graph that doesn't follow the properties of a torus. Some examples of torus are given below.





1D, 2D and 3D torus

So accourding to the above arguments we can say that our contradiction is false and torus is always node symmetric.

1.2 Prove that k-dimensional hypercube contains (2^{k-1}-1)-node complete binary tree as a subgraph, but does not contain (2^k-1)-node complete binary tree.

Answer:

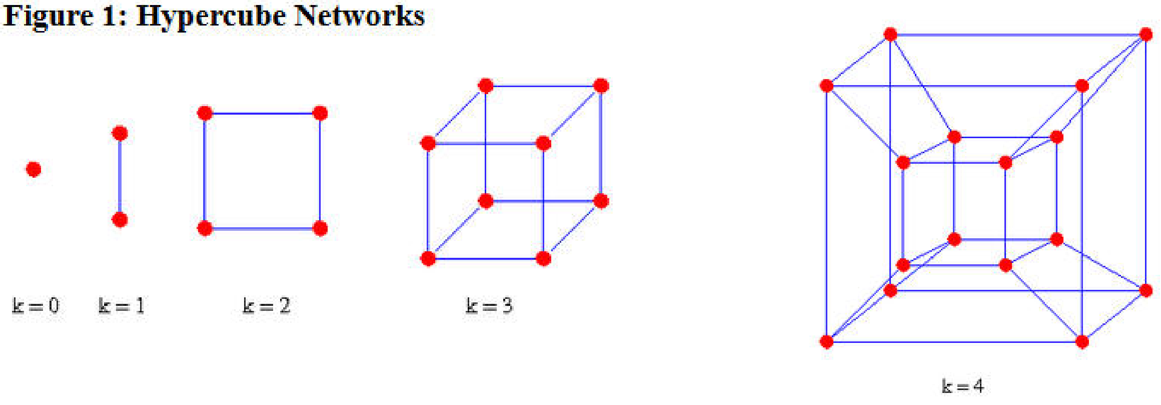
The brief **introduction** about the binary hypercube:

The binary hypercube is emerging as one of the most popular network architectures for parallel machines. This is due partly to the facts that the hypercube has a simple recursive structure that there are simple algorithms for message routing on the hypercube that work well in practice.

The Binary k-cube

A binary k-cube, often referred to as a hypercube, connects N = 2^k network nodes in the form of a cube constructed in k-dimensional space. The corners of this cube represent the nodes, and the edges represent the inter-nodal connections. More formally, if the nodes are numbered from $2^k - 1$, nodes whose binary numbering differs in exactly one position have connections between them. Figure 1 shows how binary k-cubes are constructed for k in the range 0 to 4.

Figure 1: Hypercube Networks

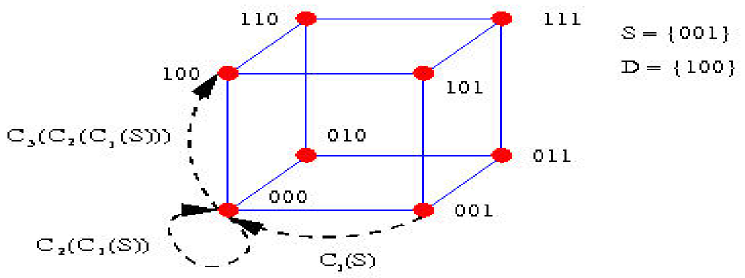


The binary k-cube therefore has k routing functions, C_i $0 \leq i \leq k-1$, one routing within each dimension, defined thus

$C_i(x) = \epsilon_i | I(x)$

Informally, for each dimension either an exchange permutation (ϵ_i) or an identity permutation (I) is applied to x in order to establish a route from any source to any destination node. A route from any source to any destination label can be found by starting at the source node and then comparing each bit in the source and destination labels in turn. If the bits are the same, then the identity permutation is applied to the source label and the route is not extended. If the bits are different, then the exchange permutation is applied to the source label, and the route extends along the link connecting the current node to a new node with a label equal to ϵ_i (current label). Such a route is illustrated in Figure 2.

Figure 2: Hypercube Routing



It is also apparent that since the maximum number of bits required to identify n processors uniquely is k = $[log_2 n]$, the path length between an arbitrary pair of nodes is at most k.

It is immediately apparent that the binary k-cube has a very rich interconnection structure, with a total of k. 2^{k-1} bidirectional connections, and k communication links per node. One possible problem, which could limit the number of nodes in an k-cube network, is the number of communication links required per node, and hence the physical complexity of the whole network. In fact, it is the length of the interconnecting wires which poses the most serious problem for networks with large values of k. This can be shown by examining the rate of growth of the volume of the network.

The rate of growth of the inter-nodal distances in a binary k-cube depends on the

length of one side of the machine. Since most machines are constructed physically in three-dimensional space, one side of a machine must have length which is O($N_{1/3}$). Consequently, the time delay associated with the transmission of messages across the most significant dimension of the network will also be equal to O($N_{1/3}$).
If the system is synchronous then the clock speed of the machine must decrease in proportion to this increasing delay; alternatively, if each processor runs at O(1) instructions per second then the interval between each communication event must increase in proportion to the increased transmission delay. The net effect of increasing wire length is that the communication bandwidth per node decreases as the system becomes larger. This is essentially a problem of physical scalability.

Proof by Contradiction: We need to use contradiction to prove that k-dimensional hypercube contains ($_{2k-1}$-1)-node, if k-dimensional hypercube contains (2_k-1)-node complete binary tree, then we can also assume that the it can also find the 3D (3-Dimesional) hypercube. There is a 7-Node complete Binary Tree, but it's a contradiction factor for because it also contains a 3-Node complete binary tree.
Now we can use a K-Dimensional hypercube that contains 2_{k-1} node a complete binary tree, we use gray code to find out that there exists a 2_{k-1} node complete binary tree.
We choose one to be the left node or the right node and consider the level to be the root node at 0^{th}. When we choose the child node at j^{th} level, for the left child node.We also just revert the
$(j-1)^{th}$, j^{th} and $(j+1)^{th}$ bits.
In the next round, when we choose the $(j+1)^{th}$ level, we just invert the bits which are j^{th} bit of parent in the j-level. Which makes sure that the child node is different from the parent node and previous one.
Since we calculate the number of 3 bit group, it's obvious that there are $k-2, 3$ bits group, Since the total number of complete binary tree is 2^{k-1}.

References:
[1]SANDEEP N.BHATT, FAN R. K. CHUNG, F. THOMSON AND ARNOLD L. ROSENBERG, EFFICIENT EMBEDDINGS OF TREES IN HYPERCUBES*,1992 Society for Industrial and Applied Mathematics ,012 ,SIAM J. COMPUT. Vol. 21, No.1,pp. 151-162, February 1992
http://www.math.ucsd.edu/∼fan/mypaps/fanpap/108hyperembed.pdf [2] The Binary k-cube
http://homepages.inf.ed.ac.uk/cgi/rni/comp-arch.pl?Networks/k-cube.html,Networks/k-cube-f.html,Networks/menu.html

2 About P2P Networks

2.1 Compare unstructured P2P networks and structured P2P networks, and list their pros and cons.

Factors	**Unstructured P2P Networks**	**Structured P2P Networks**
Overlay Network	In unstructured P2P, overlay has no particular structure or design. Peers are connected randomly.	In structured P2P, overlay network is organized into a specific topology.
Joining of a new peer	A new peer that wants to join the network randomly forms connection to the existing nodes but copying existing links of other nodes and then form its own links over time.	In structured P2P network, when a new peer wants to join the network, it has to inform the other peer of its joining and protocol ensures that new peer joins the network efficiently.
File ownership	In unstructured P2P network, files are randomly distributed on machines across network and request need to be broadcast to find the file, which can cause flooding of requests in the network.	Structured P2P network, implements a distributed hash table (DHT) to assign ownership of each file to a particular peer. This enables peers to search for resources on the network using a hash table.
File Searching	Due to unstructured overlay network, when the peer wants to find a desired piece of data in the network, the search query must be flooded through the network to find as many peers as possible that share the data.	In structured P2P, the protocol ensures that any node can efficiently search the network for a file/resource, even if the resource is extremely rare. Consistent hash table is used for searching file on the network.

Routing	Unstructured P2P usually uses broadcasting to find the file, when the file is found on the particular machine, then file is directly sent back to the original machine.	Structured P2P networks usually use variant of consistent hashing as routing protocol where searching, addition, deletion and transferring of file takes place efficiently.
Examples	Gnutella, Gossip, and Kazaa are examples of unstructured P2P protocols	Bit Torrent, the Kad network, the Storm botnet, YaCy, the Coral, Chord, Kademlia, PAST storage utility, P-Grid, CAN and Coop Net are examples of structured P2P networks.

Pros and Cons

Un-Structured P2P Networks

Pros:

1. Because there is no structure globally imposed upon them, unstructured networks are easy to build.
2. Unstructured P2P network allow for localized optimizations to different regions of the overlay.
3. Role of all peers in the network is equal.
4. Unstructured networks are highly robust in the face of high rates of ”churn”.
5. No centralized server, so there is no bottleneck or single point of failure.
6. Self-organizing.
7. No hardware cost, uses memory and processing of the peers.
8. Distributed storage and search.

Cons:

1. Flooding of requests in the network.
2. Flooding causes high amount of signaling traffic in the network, uses more CPU/memory to process all the search queries.

3. There is no correlation between peer and the content managed by it.
4. There is no guarantee of finding the file.
5. Unstructured overlay network makes it less resilient.
6. Security and trust issues.
7. Routing attacks, malicious users can join network and cause different types of attacks.
8. Routing protocol is not efficient.
9. Scalability of unstructured network is not very good because of the lack of file tracking and proper routing protocol.

Structured P2P Network

Pros:

1. In structured peer-to-peer networks the overlay is organized into a specific topology.
2. The protocol ensures that any node can efficiently search the network for a file/resource, even if the resource is extremely rare.
3. Structured P2P use distributed has table (DHT), to assign ownership of each file to a particular peer.
4. Peers search for a file in the network using hash table which enables the efficient and fast searching of file.
5. No centralized server, so there is no bottleneck or single point of failure.
6. Self-organizing.
7. No hardware cost, uses memory and processing of the peers.
8. Distributed storage and search.
9. Dynamic and Resilient.
10. Very good scalability because of proper routing algorithm.

Cons:

1. In order to route traffic efficiently through the network, nodes in a structured overlay must maintain lists of neighbors that satisfy specific criteria. This makes them less robust in networks with a high rate of churn.
2. High cost of processing for discovering resources throughout the network.

3. Leaving and joining of nodes causes extra processing on the neighbor machines.
4. Security and trust issues.
5. Vulnerable to malicious attacks.
6. Higher maintenance overhead to keep routing tables up to date.

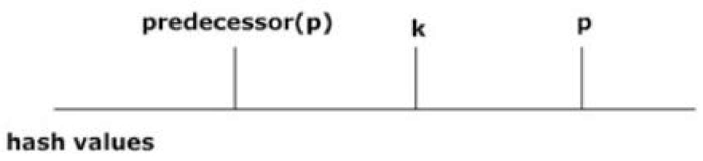
2.2 In Chord, answer how the system updates finger tables when a node leaves.

Answer:

Introduction Peer-to-peer systems and applications are distributed systems without centralized control or hierarchical organization, in which each node runs software with equivalent functionality.
The chord Protocol supports just one operation: given a key, it maps the key onto a node. Depending on the application using Chord, that node might be responsible for storing a value associated with the key. Chord uses consistent hashing to assign keys to chord nodes. Consistent hashing tends to balance load, Since each node recieves roughly the same number of keys, and requires relatively little movement of keys when nodes join and leave the system.

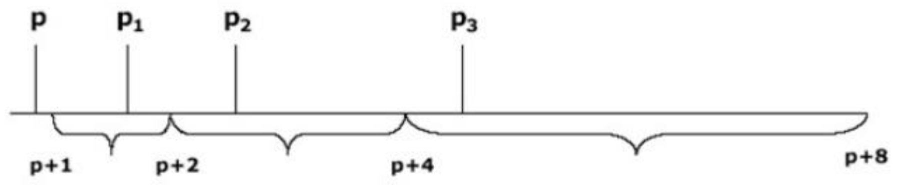
1. Based on a hashing of search keys and peer addresses on binary keys of length **m**

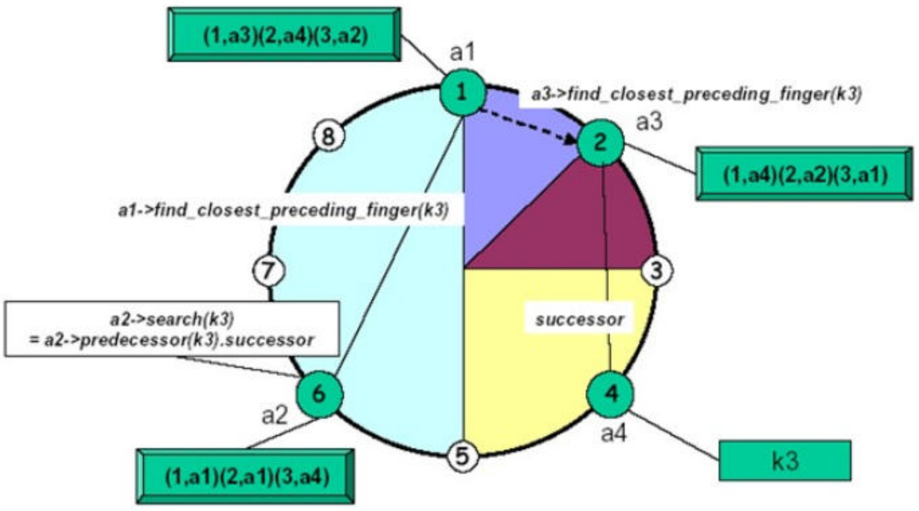
2. Each peer with hashed identifier p is responsible to store values associated with the key for all hashed keys k such that
   $k \in]$ predecessor (p) , $p]$



3. Each peer p stores a ”finger table” consisting of the first peer with hashed identifier

4. A search algorithm ensures the reliable location of the data- Complexity O(log n), n nodes in the network.




(1,a3)(2,a4)(3,a2)
a1
a3->find_closest_preceding_finger(k3)
a3
(1,a4)(2,a2)(3,a1)
a1->find_closest_preceding_finger(k3)
a2->search(k3)
= a2->predecessor(k3).successor
successor
a2
a4
(1,a1)(2,a1)(3,a4)
k3
1
2
3
4
5
6
7
8

Finger Tables update When a node Leaves Chord is a scalable protocol for lookup in a system which uses a variant of consistent hashing. When having a closer examination to the structure of Chord, a distributed hash table stores key-value pairs by assigning keys to different computers. At this, a node stores all the keys for which it is responsible. Chord requires each node to keep and maintain finger table for efficient routing.

The i_{th} entry of node n will contain the address of successor $((n + 2^{i-1}) mod\ 2^m)$

When a node leaves the Chord network, it gives all the keys that it is responsible for to it's successor. As a next step, We have a closer examination on the finger tables of all existing nodes in the Chord network. All nodes that have entries in their finger tables pointing to the leaving node have to be updated. Thus, the finger table entry has to be changed to the node identifier of the leaving node's successor. The cost for updating the finger tables when a node leaves is

$O(log^2 n)$

3 About Data Center Networks

3.1 Explain the difference between computing center and data center

Answer:

1. The terms computing center and data center are often used interchangeably, but

there is a difference between the two terms. Following are the differences which set data center and computing center apart. Data center is power protected, secure and environmentally controlled space used for housing servers, computer and network equipment.

2. Data center is the operating theatre for an enterprises network service delivery, a data center site may utilize the entire site and building shell.
3. Data centers include thousands of server racks, network centers, cooling equipment, power equipment etc.
4. On the other hand, computing center is a place in data center where actual computing or processing takes place. Network, power, and cooling resources are controlled from the computing center.
5. Security is monitored through computing center and damage control monitoring is done through computing centers. Actual data processing tasks are done in computing centers which use data stored in data centers.
6. Computing center may or may not be linked to data center. Sometime computing center is only used to control network of an organization or to monitor security, without any need of data center.
7. But computing center is an essential part of data center to control the data center itself.

3.2 Besides Dcell and Bcube, do you know any other networks proposed for data centers?

Answer: Data center network is constructed to better accommodate dynamic virtualized servers and storage environments, while the DCN architecture is regarded as one of the most important determinants of network performance, and it plays a significant role in meeting the requirements of could services as well as the agility and dynamic reconfigurability of the infrastructure for changing application demands. As a result, many novel proposals, such as **Fat-Tree** , **OSA** , **DCell** , **BCube** , **VL2** , **c-Through** , **Helios** , **FiConn** , **MDCube** , **HyperBcube**, **Portland** , **FlatNet** , **SprintNet** , **CamCube** , **Small-World** , **NovaCube** have been proposed aiming to efficiently interconnect the servers inside a data center to deliver peak performance to users.
Generally, the DCN topologies can be roughly classified into four categories:

1. Tree-based topology

   VL2, Portland

2. Recursive-defined topology

    DCell, BCube, FiConn, FlatNet, SprintNet

3. Hybrid Network

    c-Through, Helios

4. Direct Network

    CamCube, Small-World

Networks that we can consider as Alternative to DCell and Bcube

1. Portland and UCSD08

    Rearrangeable non-blocking Clos network

    No server change needed

    Destination addr based routing

2. VL2

    Reduces cables by using 10G Ethernet

    Leveraging existing OSFP and ECMP

    Randomized Valiant routing

   Details:
   VL2 is an agile and cost effective network architecture, Which is built from numerous switches arranged into a clos topology. VL2 employs Valiant Load Balancing (VLB) to spread traffic across network paths, and uses address resolution to support large server pools. Besides, VL2 applies flat addressing to eliminate fragmentation of resouces and allow any service to be assigned to any server anywhere in the Data Center. However, the directory sustem may become a bottleneck in the case of heavy network load.

3. Bidimensional Compound Network (BCN) for data centers
   Details:
   A BCN for data centers inherits the advantages of HCN. BCN is a level-i irregular compound graph which recursively defined in the first dimension for
   $i \geq 0,$
   and a level one regular compound graph in the second dimension. In each dimension, a high-level BCN employs a one lower level BCN as a unit cluster and connects many such clusters by means of a complete graph.

References:
[1]Deke Guo, Tao Chen,Dan Li, Mo Li, Yunhao Liu, Guihai Chen, Expandable and Cost-Effective Network Structures for Data Centers Using Dual-Port Servers, IEEE Transactions on Computers, Vol. 62, No. 7, 1303-1317, July 2013
[2] http://cs.sjtu.edu.cn/∼gchen/course/acn/Notes/Bcube.pptx —Slide#19
[3] Rethinking the Data Center Networking: Architecture, Network Protocols, and Resource Sharing.
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6990724