Practice Problem 1) (Computer Networking: A Top-Down Approach 6th Edition: Chapter 4 P1)

In this question, we consider some of the pros and cons of virtual-circuit and datagram networks.

a. Suppose that routers were subjected to conditions that might cause them to fail fairly often. Would this argue in favor of a VC or datagram architecture? Why?

**With a connection-oriented network, every router failure will involve the routing of that connection. At a minimum, this will require the router that is “upstream” from the failed router to establish a new downstream part of the path to the destination node, with all of the requisite signaling involved in setting up a path. Moreover, all of the routers on the initial path that are downstream from the failed node must take down the failed connection, with all of the requisite signaling involved to do this.**

**With a connectionless datagram network, no signaling is required to either set up a new downstream path or take down the old downstream path. We have seen, however, that routing tables will need to be updated (e.g., either via a distance vector algorithm or a link state algorithm) to take the failed router into account. We have seen that with distance vector algorithms, this routing table change can sometimes be localized to the area near the failed router. Thus, a datagram network would be preferable. Interestingly, the design criteria that the initial ARPAnet be able to function under stressful conditions was one of the reasons that datagram architecture was chosen for this Internet ancestor.**

b. Suppose that a source node and a destination require that a fixed amount of capacity always be available at all routers on the path between the source and destination node, for the exclusive use of traffic flowing between this source and destination node. Would this argue in favor of a VC or datagram architecture? Why?

**In order for a router to maintain an available fixed amount of capacity on the path between the source and destination node for that source-destination pair, it would need to know the characteristics of the traffic from all sessions passing through that link. That is, the router must have per-session state in the router. This is possible in a connection-oriented network, but not with a connectionless network. Thus, a connection-oriented VC network would be preferable.**

c. Suppose that the links and routers in the network never fail and that routing paths used between all source/destination pairs remains constant. In this scenario, does a VC or datagram architecture have more control traffic overhead? Why?

**In this scenario, datagram architecture has more control traffic overhead. This is due to the various packet headers needed to route the datagrams through the network. But in VC architecture, once all circuits are set up, they will never change. Thus, the signaling overhead is negligible over the long run.**

Practice Problem 2) (Computer Networking: A Top-Down Approach 6th Edition: Chapter 4 P2)

Consider a virtual-circuit network. Suppose the VC number is an 8-bit field.

a. What is the maximum number of virtual circuits that can be carried over a link?

**Maximum number of VCs over a link = 28 = 256.**

b. Suppose a central node determines paths and VC numbers at connection setup. Suppose the same VC number is used on each link along the VC’s path. Describe how the central node might determine the VC number at connection setup. Is it possible that there are fewer VCs in progress than the maximum as determined in part (a) yet there is no common free VC number?

**The centralized node could pick any VC number which is free from the set {0,1,…,28-1}. In this manner, it is not possible that there are fewer VCs in progress than 256 without there being any common free VC number.**

c. Suppose that different VC numbers are permitted in each link along a VC’s path. During connection setup, after an end-to-end path is determined, describe how the links can choose their VC numbers and configure their forwarding tables in a decentralized manner, without reliance on a central node.

**Each of the links can independently allocate VC numbers from the set {0,1,…,28-1}. Thus, a VC will likely have a different VC number for each link along its path. Each router in the VC’s path must replace the VC number of each arriving packet with the VC number associated with the outbound link.**

Practice Problem 3) (Computer Networking: A Top-Down Approach 6th Edition: Chapter 4 P6)

In the text we have used the term connection-oriented service to describe a transport-layer service and connection service for a network-layer service. Why do we use different terminology?

**In a virtual circuit network, there is an end-to-end connection in the sense that each router along the path must maintain state for the connection; hence the terminology connection service. In a connection-oriented transport service over a connectionless network layer, such as TCP over IP, the end systems maintain connection state; however the routers have no notion of any connections; hence the terminology connection-oriented service.**

Practice Problem 4) (Computer Networking: A Top-Down Approach 6th Edition: Chapter 4 P10)

Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link interfaces as follows:

|  |  |
| --- | --- |
| Destination Address Range | Outgoing Link Interface |
| 11100000 00000000 00000000 00000000  through  11100000 00111111 11111111 11111111 | 0 |
| 11100000 01000000 00000000 00000000  through  11100000 01000000 11111111 11111111 | 1 |
| 11100000 01000001 00000000 00000000  through  11100001 01111111 11111111 11111111 | 2 |
| Otherwise | 3 |

a. Provide a forwarding table that has five entries, uses longest prefix matching, and forwards packets to the correct link interfaces.

|  |  |
| --- | --- |
| **Prefix Match** | **Link Interface** |
| **11100000 00** | **0** |
| **11100000 01000000** | **1** |
| **1110000** | **2** |
| **11100001 1** | **3** |
| **otherwise** | **3** |

b. Describe how your forwarding table determines the appropriate link interface for datagrams with destination addresses:

11001000 10010001 01010001 01010101

11100001 01000000 11000011 00111100

11100001 10000000 00010001 01110111

**Prefix match for first address is 5th entry: link interface 3**

**Prefix match for second address is 3nd entry: link interface 2**

**Prefix match for third address is 4th entry: link interface 3**

Practice Problem 5) (Computer Networking: A Top-Down Approach 6th Edition: Chapter 4 P13)

Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3. Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17/24. Also suppose that Subnet 1 is required to support at least 60 interfaces, Subnet 2 is to support at least 90 interfaces, and Subnet 3 is to support at least 12 interfaces. Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints.

**Lots of possibilities. Must be able to support the minimum number given in problem. Some examples:**

**Subnet 1: 223.1.17.128/26**

**Subnet 2: 223.1.17.0/25**

**Subnet 3: 223.1.17.192/28**

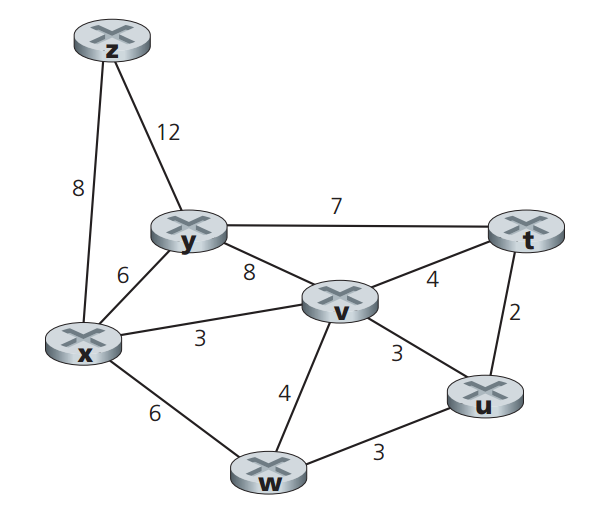
Practice Problem 6) (Computer Networking: A Top-Down Approach 6th Edition: Chapter 4 P15)

In Problem 4 (P10) you are asked to provide a forwarding table (using longest prefix matching). Rewrite this forwarding table using the a.b.c.d/x notation instead of the binary string notation.

|  |  |
| --- | --- |
| **Destination Address** | **Link Interface** |
| **11100000 00 (224.0.0.0/10)** | **0** |
| **11100000 01000000 (224.64.0.0/16)** | **1** |
| **1110000 (224.0.0.0/8)** | **2** |
| **11100001 1 (225.128.0.0/9)** | **3** |
| **otherwise** | **3** |

Practice Problem 7) (Computer Networking: A Top-Down Approach 6th Edition: Chapter 4 P26)

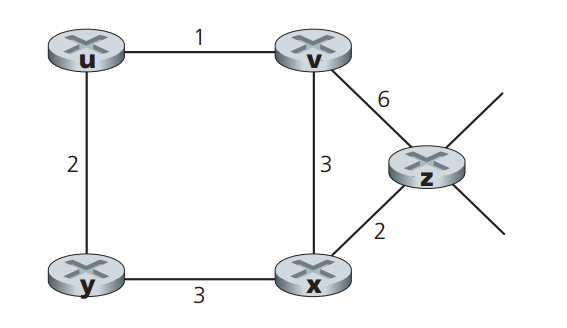
Consider the following network. With the indicated link costs, use Dijkstra’s shortest-path algorithm to compute the shortest path from x to all network nodes. Show how the algorithm works by computing a table similar to Table 4.3.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Step** | **N’** | **D(t),p(t)** | **D(u),p(u)** | **D(v),p(v)** | **D(w),p(w)** | **D(y),p(y)** | **D(z),p(z)** |
| **0** | **x** | **∞** | **∞** | **3,x** | **6,x** | **6,x** | **8,x** |
| **1** | **xv** | **7,v** | **6,v** | **3,x** | **6,x** | **6,x** | **8,x** |
| **2** | **xvu** | **7,v** | **6,v** | **3,x** | **6,x** | **6,x** | **8,x** |
| **3** | **xvuw** | **7,v** | **6,v** | **3,x** | **6,x** | **6,x** | **8,x** |
| **4** | **xvuwy** | **7,v** | **6,v** | **3,x** | **6,x** | **6,x** | **8,x** |
| **5** | **xvuwyt** | **7,v** | **6,v** | **3,x** | **6,x** | **6,x** | **8,x** |
| **6** | **xvuwytz** | **7,v** | **6,v** | **3,x** | **6,x** | **6,x** | **8,x** |

Practice Problem 8) (Computer Networking: A Top-Down Approach 6th Edition: Chapter 4 P28)

Consider the network shown below, and assume that each node initially knows the costs to each of its neighbors. Consider the distance-vector algorithm and show the distance table entries at node z.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **u** | **v** | **x** | **y** | **z** |
| **v** | **∞** | **∞** | **∞** | **∞** | **∞** |
| **x** | **∞** | **∞** | **∞** | **∞** | **∞** |
| **z** | **∞** | **6** | **2** | **∞** | **0** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **u** | **v** | **x** | **y** | **z** |
| **v** | **1** | **0** | **3** | **∞** | **6** |
| **x** | **∞** | **3** | **0** | **3** | **2** |
| **z** | **7** | **5** | **2** | **5** | **0** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **u** | **v** | **x** | **y** | **z** |
| **v** | **1** | **0** | **3** | **3** | **5** |
| **x** | **4** | **3** | **0** | **3** | **2** |
| **z** | **6** | **5** | **2** | **5** | **0** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **u** | **v** | **x** | **y** | **z** |
| **v** | **1** | **0** | **3** | **3** | **5** |
| **x** | **4** | **3** | **0** | **3** | **2** |
| **z** | **6** | **5** | **2** | **5** | **0** |