Homework Problems and Discussion Questions

Chapter 4

Review Questions

Sections 4.1-4.4

1) What are the two main functions of a datagram-based network layer? What additional functions does a VC-based network layer have?

2) List and describe the ATM network service models.

3) Compare and contrast link-state and distance-vector routing algorithms.

4) Discuss how a hierarchical organization of the Internet has helped to scale to millions of users.

5) It is necessary that every autonomous system use the same intra-autonomous routing algorithm? Why or why not?

Section 4.5

6) What is the decimal equivalent of the IP address 223.1.3.27 ?

7) Consider a LAN to which ten host interfaces and three router interfaces are attached. Suppose all three LANs use class C addresses. The IP addresses for the 13 devices will be identical in which of the first 32 bits?

8) Consider a router with three interfaces. Suppose all three interfaces use class C addresses. Will the IP addresses of the three interfacess necessarily have the same first 8 bits?

9) Suppose there are three routers between source and destination hosts. Ignoring fragmentation, an IP segment sent from source host to destination host will travel over how many interfaces? How many routing tables will be indexed to move the datagram from source to destination?

10) Suppose an application generates chunks 40 bytes of data every 20 msec, and each chunk gets encapsulated in a TCP segment and then an IP datagram. What percentage of each datagram will be overhead and what percentage will be application data?

11) Consider sending a 3000 byte datagram into a link that has a MTU of 500 bytes. Suppose the original datagram is stamped with the identification number 422. How many fragments are generated? What are their characteristics?

12) Consider Figure 4.5-2. Starting with the original table in D, suppose that D receives from A the following advertisement:

destination network	next router	number of hops to destination
30	C	10
1	--	1
10	--	1
....	....	....

Will the table in A change? If so how?

13) Contrast and compare the advertisements used by RIP and OSPF.

14) RIP advertisements typically announce the number of hops to various destinations. BGP updates, on the otherhand, announce the _____ (fill in the blank) to the various destinations.

15) Why are different inter-AS and intra-AS protocols used in the Internet?

Section 4.6

16) Describe three different types of switching fabrics commonly used in packet switches.

17) Why are buffers needed at the output ports of switches? Why are buffers needed at the input port of switches?

Section 4.7

18) Compare and contrast the IPv4 and the IPv6 header fields. Do they have any fields in common?

19) It has been said that IPv6 tunnels through IPv4 routers, IPV6 treats the IPv4 tunnels as link layer protocols. Do you agree with this statement? Why or why not?

Section 4.8

20) What is an important difference between implementing the multicast abstract via multiple unicasts, and a single network (router) supported multicast group.

21) True or False: when a host joins a multicst group, it must change its IP address to be that of the multicast group it is joining.

22) What are the roles played by the IGMP protocol and a wide-area multicast routing protocol?

23) What is the difference between a group-shared tree and a source-based tree in the context of multicast routing?

24) True or False: In reverse path forwarding, a node will receive multiple copies of the same packet. True or False: In reverse path forwarding, a node may forward multiple copies of a packet over the same outgoing link.

25) Classify each of the following multicast routing algorithms as either a source-baed tree approach or a group-shared tree approach: DVMRP, MOSPF, CBT, PIM Sparse Mode, PIM Dense Mode.

Problems

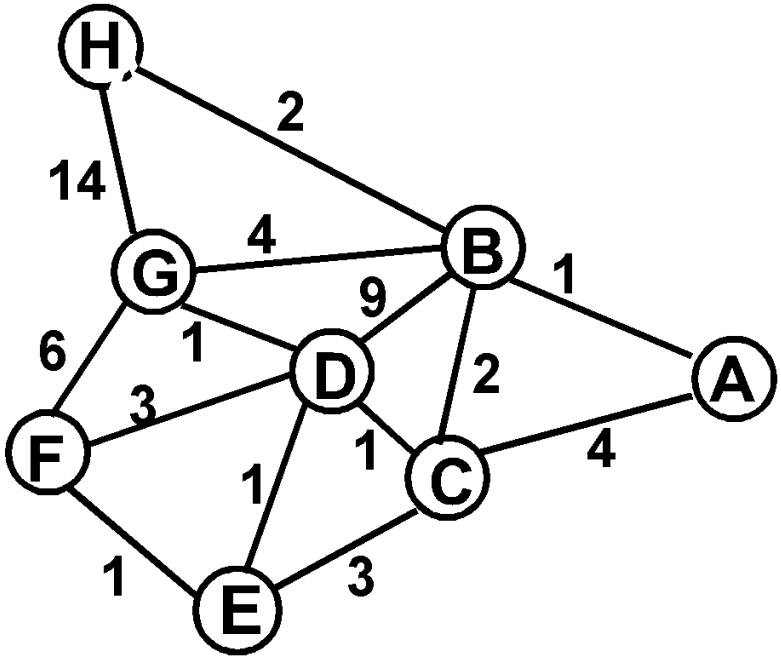
1) Let us consider some of the pros and cons of a connection-oriented versus connectionless architecture.

> a) Suppose that in the network layer, routers were subjected to "stressful" conditions that might cause them to fail fairly often. At a high level, what actions would need to be taken on such router failure. Does this argue for a connection-oriented or a connectionless environment?
>
> b) Suppose that in order to provide a *guarantee* regarding the level of performance (e.g., delay) that would be seen along a source-to-destination path, the network requires a sender to declare its peak traffic rate. If the declared peak traffic rate and the existing declared traffic rates that have been declared are such that there is no way to get traffic from the source to the destination that meets the required delay requirements, the source is not allowed access to the network. Would such a approach be more easily accomplished within a connection-oriented or connectionless paradigm?

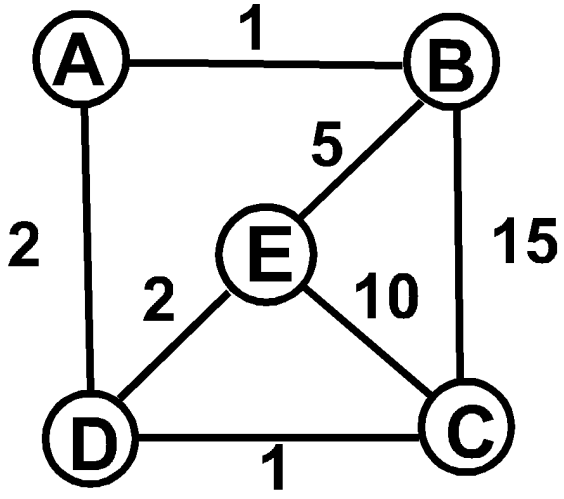
2) In Figure 4.2.1, enumerate the paths from A to F that do not contain any loops.

3) Consider the network shown below, with the indicated link costs. Use Dijkstra's shortest path algorithm to compute the shortest past from F to all network nodes. Show how the algorithm works by

computing a table similar to Table 4.2.1.



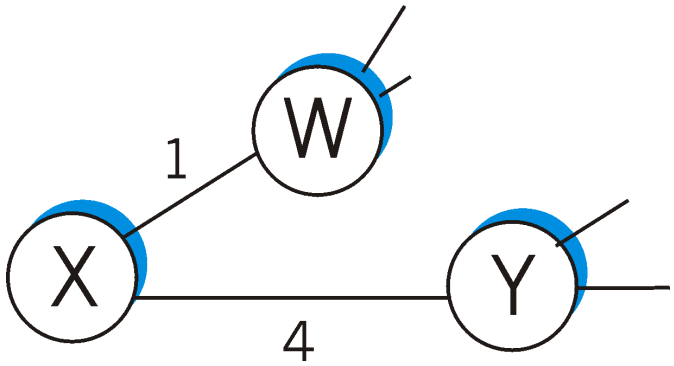
4) Consider the network shown below and assume that each node initially knows the costs to each of its neighbors. Consider the distance vector algorithm and show the distance table entries at node E.



5) Consider a general topology (i.e., not the specific network shown above) and a synchronous version of the distance vector algorithm. Suppose that at each iteration, a node exchanges its minimum costs with its neighbors and receives their minimum costs. Assuming that the algorithm begins with each node

knowing only the costs to its immediate neighbors, what is the maximum number of iterations required until the distributed algorithm converges? Justify your answer.

6) Consider the network fragment shown below. X has only two attached neighbors, W and Y. W has a minimum cost path to destination A of cost 5 and Y has a minimum cost path to A of 6. The complete paths from W and Y to A (and between W and Y) are not shown. All link costs in the network have strictly positive integer values.



a) Give X's distance table (row) entries for destinations X, Y and A.
b) Give a link cost change for either c(X,W) or c(X,Y) such that X will inform its neighbors of a new minimum cost path to A as a result of executing lines 15 and 24 of the distance vector algorithm.
c) Give a link cost change for either c(X,W) or c(X,Y) such that X will *not* inform its neighbors of a new minimum cost path to A as a result of executing lines 15 and 24 of the distance vector algorithm.

7) Compute the distance tables for X, Y and Z shown in rightmost column of Figure 4.2-4. After the computation of the new distance tables, which nodes will send which updated values to which neighbors?

8) Consider the three node topology shown in Figure 4.2.4. Rather than having the link costs shown in Figure 4.2-4, the link costs are c(X,Y)=5, c(Y,Z)=6, c(Z,X)=2. Compute the distance tables after the initialization step and after each iteration of a synchronous version of the distance vector algorithm (as we did in our earlier discussion of Figure 4.2-4.)

9) Consider the 8-node network (with nodes labeled A-H) above. Show the minimal cost spanning tree rooted at A that includes (as end hosts) nodes C, D, E, and G. Informally argue why your spanning tree is a minimal cost spanning tree.

10) We saw in Section 4.8 that there is no network layer protocol that can be used to identify the hosts participating in a multicast group. Given this, how can multicast applications learn the identities of the

hosts that are participating in a multicast group?

11) Consider the two basic approaches identified towards achieving multicast: unicast emulation and network-layer-multicast. Consider a single sender and 32 receivers. Suppose the sender is connected to the receiver through a binary tree of routers. What is the cost of sending a multicast packet in the case of unicast emulation and network-layer multicast for this topology? Here, each time a packet (or copy of a packet) is sent over a single link, it incurs a unit of "cost". What topology for interconnecting the sender, receivers, and routers will bring the cost of unicast emulation and true network-layer-multicast as far apart as possible.? You can choose as many routers as you'd like.

12) Design (give a pseudocode description of) an application-level protocol that maintains the host addresses of all hosts participating in a multicast group. Specifically identify the network service (unicast or multicast) that is used by your protocol, and indicate whether your protocol is sending messages in-band or out-of-band (with respect to the application-data flow among the multicast group participants), and why.

13) Consider the topology from Figure 4.8-8. Suppose the link cost from B to D changes from 1 to 10. Find the Steiner tree that connects all of the shaded routers. (Note: you are not being asked here to program a solution to the Steiner tree problem. Instead, you should be able to construct the minimum costs tree by inspection and informally convince yourself that it is the minimum costs tree). If you were asked (you are *not* being asked to actually do so!), how would you prove that your tree is indeed a minimum cost tree?

14) Center-based routing. Consider the topology shown in Figure 4.8-8. Suppose node C is chosen as the center in a center-based multicast routing algorithm. Assuming that each attached router in the multicast group uses its least cost path to node C to send join messages to C, draw the resulting center-based multicast routing tree. Is the resulting tree a minimum cost Steiner tree? Justify your answer.

15) Least unicast-cost path routing. Consider Figure 4.8-8. Suppose that node E is chosen as the source. Compute the least unicast-cost path multicast routing tree from E to multicast routers A, B, and F.

16) Reverse path forwarding. Consider the topology and link costs shown in Figure 4.8-8 and suppose that node E is the multicast source. Using arrows like those shown in Figure 4.8-11, indicate links over which packets will be forwarded using RPF, and links over which packets will not be forwarded, given that node E is the source.

17) Suppose that the cost of a transmitting a multicast packet on a link is completely independent of the cost of transmitting a unicast packet on a link. Will reverse path forwarding still work in this case? Justify your answer.

18) Traffic concentration in center-based trees. Consider the simple topology shown in Figure 4.8-8. Suppose that each of the multicast routers receive one unit of traffic per unit time from an attached host.

This traffic must be forwarded to the other three multicast routers. Suppose that node C is chosen as the center node in a center-based multicast routing protocol (see homework problem above). Given the resulting routing tree, compute the rate of traffic on each link in the topology. (Compute the total amount of traffic on each link, regardless of the direction of the traffic flow). Suppose next that RPF is used to build four source-specific routing trees rooted at each of the routers A, B, E, F. Recompute the rate of traffic on each of the links in this second scenario. In this example, does a center-based tree or source-specific trees tend to concentrate traffic?

19) Suppose that a network has G multicast groups, each with S group members (hosts), each of which can be a sender. Under DVMRP, each router must thus maintain up to S pieces of routing information (the outgoing link on the shortest reverse path to the sender, for each of the S senders) for each group. Thus, in the worst case, each router must maintain S*G pieces of routing information, when taking all groups into account. What is the worst case amount of routing information needed by MOSPF, PIM Sparse Mode and PIM Dense Mode? Justify your answers.

20) Birthday problem. What is the size of the mutlicast address space. Suppose now that two different multicast groups randomly choose a multicast address. What is the probability that they choose the same address? Suppose now that 1000 multicast groups are ongoing at the same time and chose their multicast group addresses at random. What is the probability that they interfere with each other?

21) Recall that in our discussion of multicast tunneling, we said that an IP multicast datagram is carried inside of a IP unicast datagram. How does the IP router at the end of the multicast tunnel know that the unicast datagram contains an IP multicast datagram (as opposed to simply being an IP unicast datagram that should be forwarded along)?

Discussion Questions

1) Suppose AS X and Z are not directly connected but instead connected by AS Y. Further suppose that X has a peering agreement with Y, and that Y has a peering agreement with Z. Finally, suppose that Z wants to transit all of Y's traffic but does not want to transit X's traffic. Does BGP allow Z to implement this policy?

2) In Section 4.7 we indicated that deployment of IPv6 has been slow to date. Why has it been slow? What is needed to accelerate its deployment? (See article by L. Garber.)

3) In Section 4.8.1 we saw that the multicast abstraction can be implemented by having a sender open an individual connection to each of the receivers. What are the drawbacks of this approach compared to the approach that provides native multicast support at the network layer? What are the advantages of this approach?

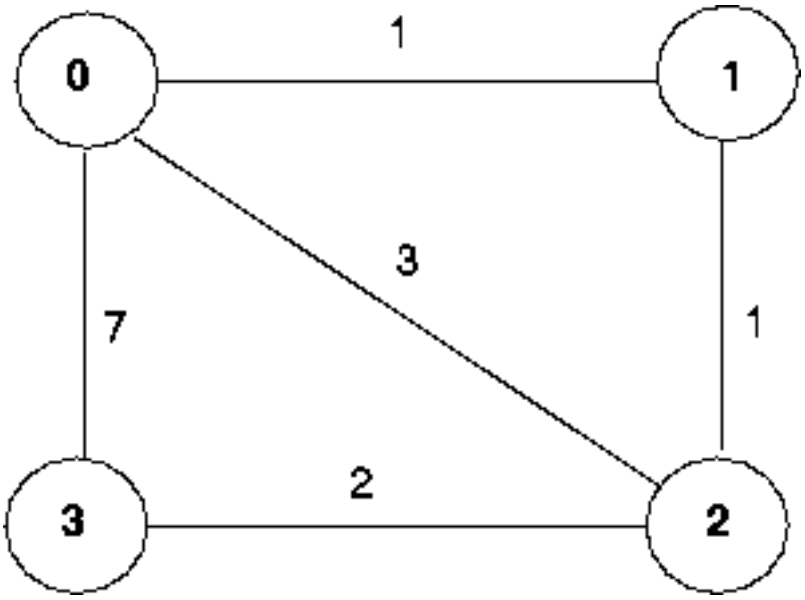
4) In Section 4.8 we identified a number of multicast applications. Which of these applications are well-suited for the minimalist Internet multicast service model? Why? Which applications are not

particularly well-suited for this service model?

5) Given the CBT soft state mechanism for maintaining a tree, why do you think there is a separate FLUSH_TREE message? What would happen if the FLUSH_TREE message were lost?

Programming Assignment

In this third programming assignment, you will be writing a ``distributed" set of procedures that implement a distributed asynchronous
distance vector routing for the network shown below:



You are to write the following routines that will ``execute" asynchronously within the emulated environment provided for this assignment. For node 0, you will write the routines:

- *rtinit0()* This routine will be called once at the beginning of the emulation. *rtinit0()* has no arguments. It should initialize your distance table in node 0 to reflect the direct costs of 1, 3, and 7 to nodes 1, 2, and 3, respectively. In the figure above, all links are bi-directional and the costs in both directions are identical. After initializing the distance table, and any other data structures needed by your node 0 routines, it should then send its directly-connected neighbors (in this case, 1, 2 and 3) the cost of it minimum cost paths to all other network nodes. This minimum cost information is sent to neighboring nodes in a routing update packet by calling the routine *tolayer2(),* as described in the full assignment. The format of the routing update packet is also

described in the full assignment

- *rtupdate0(struct rtpkt *rcvdpkt)*. This routine will be called when node 0 receives a routing packet that was sent to it by one if its directly connected neighbors. The parameter **rcvdpkt* is a pointer to the packet that was received. *rtupdate0()* is the ``heart'' of the distance vector algorithm. The values it receives in a routing update packet from some other node *i* contain *i*'s current shortest path costs to all other network nodes. *rtupdate0()* uses these received values to update its own distance table (as specified by the distance vector algorithm). If its own minimum cost to another node changes as a result of the update, node 0 informs its directly connected neighbors of this change in minimum cost by sending them a routing packet. Recall that in the distance vector algorithm, only directly connected nodes will exchange routing packets. Thus nodes 1 and 2 will communicate with each other, but nodes 1 and 3 will not communicate with each other.

Similar routines are defined for nodes 1, 2 and 3. Thus, you will write 8 procedures in all: *rtinit0(), rtinit1(), rtinit2(), rtinit3(), rtupdate0(), rtupdate1(), rtupdate2(), rtupdate3().*These routines will together implement a distributed, asynchronous computation of the distance tables for the topology and costs shown in the figure above.

You can find the full details of the programming assignment, as well as C code that you will need to create the simulated hardware/software environment at http://gaia.cs.umass.edu/kurose/network/programming_assignment.htm

Lab: Implemented a distributed, asynchronous distance vector routing algorithm

Overview

In this lab, you will be writing a ``distributed" set of procedures that implement a distributed asynchronous distance vector routing for the network shown in Figure Lab.4-1.

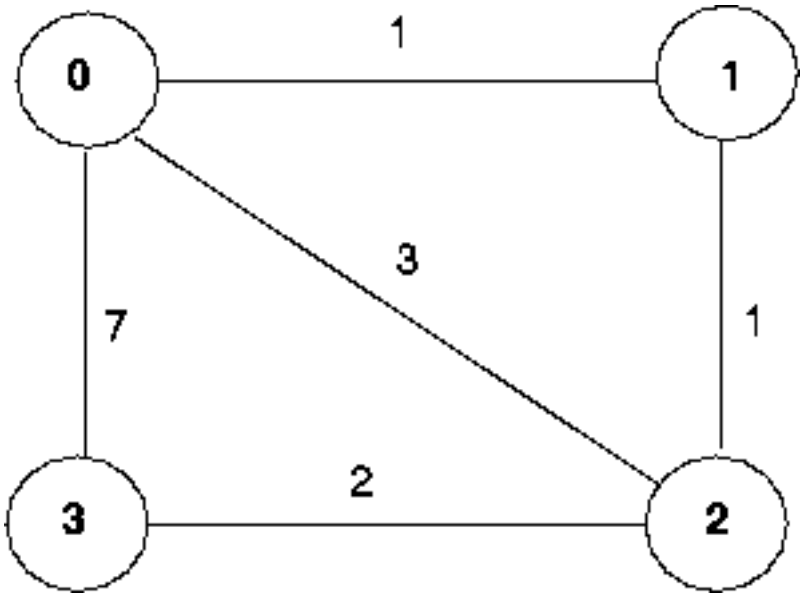


Figure Lab.4-1: Network topology and link costs for DV routing lab

The Basic Assignment

The routines you will write For the basic part of the assignment, you are to write the following routines which will ``execute" asynchronously within the emulated environment that we have written for this assignment.

For node 0, you will write the routines:

- `rtinit0()` This routine will be called once at the beginning of the emulation. `rtinit0()` has no arguments. It should initialize the distance table in node 0 to reflect the direct costs of 1, 3, and 7 to nodes 1, 2, and 3, respectively. In Figure 1, all links are bi-directional and the costs in both directions are identical. After initializing the distance table, and any other data structures needed by your node 0 routines, it should then send its directly-connected neighbors (in this case,

1, 2 and 3) the cost of it minimum cost paths to all other network nodes. This minimum cost information is sent to neighboring nodes in a *routing packet* by calling the routine `tolayer2()`, as described below. The format of the routing packet is also described below.

- `rtupdate0(struct rtpkt *rcvdpkt)`. This routine will be called when node 0 receives a routing packet that was sent to it by one if its directly connected neighbors. The parameter `*rcvdpkt` is a pointer to the packet that was received.

`rtupdate0()` is the ``heart'' of the distance vector algorithm. The values it receives in a routing packet from some other node *i* contain *i*'s current shortest path costs to all other network nodes. `rtupdate0()` uses these received values to update its own distance table (as specified by the distance vector algorithm). If its own minimum cost to another node changes as a result of the update, node 0 informs its directly connected neighbors of this change in minimum cost by sending them a routing packet. Recall that in the distance vector algorithm, only directly connected nodes will exchange routing packets. Thus nodes 1 and 2 will communicate with each other, but nodes 1 and 3 will node communicate with each other.

As we saw in class, the distance table inside each node is the principal data structure used by the distance vector algorithm. You will find it convenient to declare the distance table as a 4-by-4 array of `int`'s, where entry `[i,j]` in the distance table in node 0 is node 0's currently computed cost to node i via direct neighbor j. If 0 is not directly connected to *j*, you can ignore this entry. We will use the convention that the integer value 999 is ``infinity.''

Figure Lab.4-2 provides a conceptual view of the relationship of the procedures inside node 0.

Similar routines are defined for nodes 1, 2 and 3. Thus, you will write 8 procedures in all: `rtinit0(), rtinit1(), rtinit2(), rtinit3(),rtupdate0(), rtupdate1(), rtupdate2(), rtupdate3()`

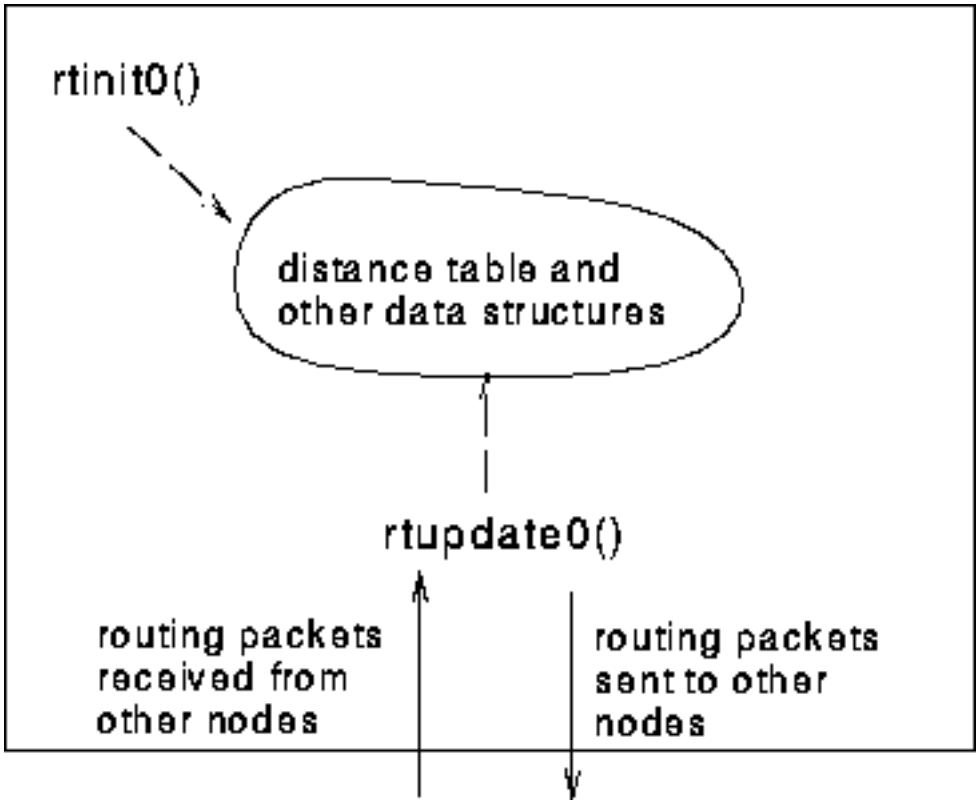


Figure Lab.4-2: Relationship between procedures inside node 0

Software Interfaces

The procedures described above are the ones that you will write. We have written the following routines that can be called by your routines:

`tolayer2(struct rtpkt pkt2send)`
where `rtpkt` is the following structure, which is already declared for you. The procedure `tolayer2()` is defined in the file prog3.c

```
extern struct rtpkt {

  int sourceid;       /* id of node sending this pkt, 0, 1, 2, or 3
*/

  int destid;         /* id of router to which pkt being sent

                         (must be an immediate neighbor) */

  int mincost[4];    /* min cost to node 0 ... 3 */
```

```
};
```

Note that `tolayer2()` is passed a structure, not a pointer to a structure.

`printdt0()`
: will pretty print the distance table for node 0. It is passed a pointer to a structure of type `distance_table`. `printdt0()` and the structure declaration for the node 0 distance table are declared in the file `node0.c`. Similar pretty-print routines are defined for you in the files `node1.c, node2.c node3.c`.

The simulated network environment

Your procedures `rtinit0(), rtinit1(), rtinit2(), rtinit3()` and `rtupdate0(), rtupdate1(), rtupdate2(), rtupdate3()` send routing packets (whose format is described above) into the medium. The medium will deliver packets in-order, and without loss to the specified destination. Only directly-connected nodes can communicate. The delay between is sender and receiver is variable (and unknown).

When you compile your procedures and my procedures together and run the resulting program, you will be asked to specify only one value regarding the simulated network environment:

- **Tracing.** Setting a tracing value of 1 or 2 will print out useful information about what is going on inside the emulation (e.g., what's happening to packets and timers). A tracing value of 0 will turn this off. A tracing value greater than 2 will display all sorts of odd messages that are for my own emulator-debugging purposes.

  A tracing value of 2 may be helpful to you in debugging your code. You should keep in mind that *real* implementors do not have underlying networks that provide such nice information about what is going to happen to their packets!

The Basic Assignment

You are to write the procedures `rtinit0(), rtinit1(), rtinit2(), rtinit3()` and `rtupdate0(), rtupdate1(), rtupdate2(), rtupdate3()` which together will implement a distributed, asynchronous computation of the distance tables for the topology and costs shown in Figure 1.

You should put your procedures for nodes 0 through 3 in files called node0.c, ... node3.c. You are **NOT** allowed to declare any global variables that are visible outside of a given C file (e.g., any global variables you define in `node0.c.` may only be accessed inside `node0.c`). This is to force you to abide by the coding conventions that you would have to adopt is you were really running the procedures in four distinct nodes. To compile your routines: `cc prog3.c node0.c node1.c node2.c node3.` Prototype versions of these files are here: node0.c, node1.c, node2.c, node3.c. You can pick up a copy of the file prog3.c at http://gaia.cs.umass.edu/kurose/network/prog3.c.

This assignment can be completed on any machine supporting C. It makes no use of UNIX features.

As always, most instructors would expect you to hand in a code listing, a design document, and sample output.

For your sample output, your procedures should print out a message whenever your `rtinit0(), rtinit1(), rtinit2(), rtinit3()` or `rtupdate0(), rtupdate1(), rtupdate2(), rtupdate3()` procedures are called, giving the time (available via my global variable `clocktime`). For `rtupdate0(), rtupdate1(), rtupdate2(), rtupdate3()` you should print the identity of the sender of the routing packet that is being passed to your routine, whether or not the distance table is updated, the contents of the distance table (you can use my pretty-print routines), and a description of any messages sent to neighboring nodes as a result of any distance table updates.

The sample output should be an output listing with a TRACE value of 2. Highlight the final distance table produced in each node. Your program will run until there are no more routing packets in-transit in the network, at which point our emulator will terminate.

The Advanced Assignment

You are to write two procedures, `rtlinkhandler0(int linkid, int newcost)` and `rtlinkhandler1(int linkid, int newcost)`, which will be called if (and when) the cost of the link between 0 and 1 changes. These routines should be defined in the files `node0.c` and `node1.c,` respectively. The routines will be passed the name (id) of the neighboring node on the other side of the link whose cost has changed, and the new cost of the link. Note that when a link cost changes, these routines will have to update the distance table and may (or may not) have to send updated routing

packets to neighboring nodes.

In order to complete the advanced part of the assignment, you will need to change the value of the constant LINKCHANGES (line 3 in `prog3.c`) to 1. FYI, the cost of the link will change from 1 to 20 at time 10000 and then change back to 1 at time 20000. Your routines will be invoked at these times.

We would again **STRONGLY** recommend that you first implement the undergraduate assignment and then extend your code to implement the graduate assignment. It will **not** be time wasted. (Believe me, I learned this the hard way!)

Q&A

When we've taught this lab in our introductory neworking course, students have posed versious questions. If you are interested
in looking at the questions we've received (and answers), check out
http://gaia.cs.umass.edu/kurose/network/programming_assignment_QA.htm