THE USE OF COMPUTER SIMULATION TO ILLUSTRATE DYNAMIC ROUTING ALGORITHMS IN AN EDUCATIONAL SETTING

Peter Vial & Parviz Doulai
School of Electrical, Computer and Telecommunications Engineering
University of Wollongong
Wollongong, Australia

Abstract

Computer simulation plays an important role in the educational experience of technologies involved with telecommunication engineering. At the University of Wollongong a computer simulation oriented laboratory class was designed and implemented to illustrate dynamic routing algorithms at undergraduate level. This paper outlines the design and purpose of the laboratories in which ARENA was used and highlights how Visual Basic modules can be incorporated in the design of telecommunications based simulations.

1. Iintroduction

In engineering education, computer simulations are used to represent the essential features of a real system so that learners can test their analytical and design skills in a convenient and safe environment. For instance, in basic electrical engineering education, computer simulation provides a unique way of creating stimulation and challenge along with an opportunity to work on realistic case studies that could not be achieved otherwise. In engineering education, a successful implementation of a simulation-based teaching and learning approach requires extensive course material and laboratory support documents development accompanied by supplementary lecture material including specially tailored assignments, tutorial questions and assessment tasks.

During the academic year of 2001 a new subject was offered for the first time in conjunction with the University of Wollongong new undergraduate program in Internet Technology program. This subject was called Internet Technology 2, and is a core second year subject for students doing Bachelor of Internet Science and Technology (BIST) and optional for students doing Computer Science.

This paper reports on design and implementation of a computer simulation-based learning environment aiming to help students to understand Internet's dynamic routing algorithms in a visually rich environment. This paper also looks at the actual students' usage of the system. Preliminary evaluation of the project indicate that the simulation-based learning environment for complex engineering concepts such as dynamic routing algorithms directly contributes towards the student improved learning.

2. Course Structure

The Faculty of Informatics and the School of Electrical, Computer and Telecommunications Engineering have cooperated together to offer a degree called the Bachelor of Internet Science and Technology (BIST). Students who study under the Technology strand are able in second year to undertake a new subject called Internet Technology 2. These students are required to participate in an advanced laboratory program that has lent heavily upon simulation using a program called ARENA. It is a telecommunications based subject, studying aspects of Internet Technology which include Dynamic Routing Algorithms. Associated with this subject was an advanced laboratory in which the second year undergraduate students were to participate. As this laboratory was primarily concerned with reinforcing concepts encountered within the subjects curriculum, part of the laboratory program used simulation.

The simulation package chosen was ARENA. This package has been used within the School of

Electrical Computer and Telecommunications Engineering to teach classical telecommunications queueing theory at undergraduate and postgraduate level [10]. The simulations in the previous subjects were trivial in scope compared to the simulations undertaken for this subject. This paper outlines and gives examples of how ARENA was used to provide simulations of basic dynamic routing techniques. The ARENA simulations used will be explained and full (unabridged) copies of the simulations have been provided to the conference organizers for inclusion with the proceedings and are available at the first authors web page [4].

3. Dynamic Routing Algorithms

The Internet uses dynamic routing algorithms that come under two classical types:

1. Distance Vector
2. Link State

Typical examples of Internet protocols which use Distance Vector are RIPv1 (Routing Information Protocol), RIPv2 and BGPv4 (Border Gateway Protocol) [1]. Two examples of Internet protocols, which use the Link State algorithm, are OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System) protocols [1][2].

Distance Vector algorithms work on the principle that all paths have equal value. Here, each individual link from one router to another is called a 'hop'. The number of hops between a source network or host and a destination network or host is used as the statistic to determine the minimum hop count. There are many deficiencies with the use of the Distance Vector algorithm, such as:

- The minimum hop count may not be the most appropriate path through the Internet. For example, choosing links which may have a lower bandwidth than another set of links with a larger hop count but much higher bandwidth for the intervening transmission links is not an optimal choice [1]
- Each router has no knowledge of the interfaces that other routers are connected to. The only information that each router knows is the hop count of its neighbourhood routers from other reported networks. As a result, the Distance Vector algorithm is also known as *routing by rumor*. [3]
- These algorithms are slow to converge, for example if using the legacy distance vector protocol RIP, it uses 30 seconds between routing table updates and thus for a medium sized network (one with about 100 routers), it may take many minutes before convergence is achieved.
- These algorithms are prone to routing loops once convergence has been achieved. This occurs because each router has absolutely no knowledge about the surrounding network topology. When a link failure occurs it is possible that loops can occur as outlined in [5][6].

The other dynamic routing algorithm is the link state algorithm. This is based on Dijkstra Shortest Path First (SPF) algorithm [7]. It requires that the entire network topology be known, which in practice is usually defined in terms of Areas or Autonomous Systems (Systems controlled by a single organisation). The Internets main protocol for Interior Routing is OSPF. As indicated it separates the local network into Areas, with the backbone network being designated Area 0. The link state algorithm works by sending out link state messages (called LSA's in OSPF) which contain all of the network subnets and interfaces known by the individual routers within a Area or domain. The individual routers then store the information from various Link state messages into a topological database. From this database the Shortest path first tree can be derived. The root of the tree is then taken as the router that is forming the database and all branches are attached from this point to the form the shortest path tree as shown in Figure 3.

Starting at the root of the tree, which in Figure 2 is Router A, each branch is revealed and the minimum cost path is chosen. For example in Figure 2, the

Advertiser ID	Network ID	Cost Of Route	Neighbour ID
A	1	2	E
A	2	3	F
A	10	1	B
B	10	4	A
B	12	2	C
C	12	5	B
C	15	2	D
D	15	5	C
D	61	3	E
E	1	3	A
E	61	2	D
F	2	2	A
F	22	3	-

Figure 1: Example of a generic link state topological database

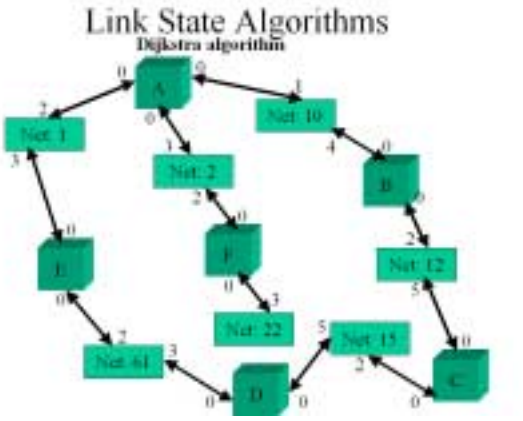
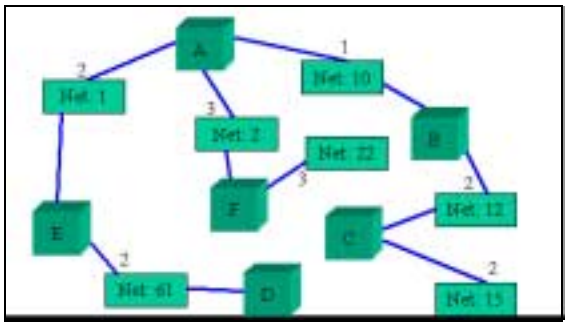


Figure 2: The topological database in tree format showing all possible linkages with Router A as the root.

first minimum path is actually to **Net 10**, so it is made permanent in the Shortest Path Tree and then Router B is revealed. This process is continued, always choosing the link that costs the less until the Shortest Path Tree is formed and all networks in the Area have had a path found to them. Note that the cost to Router B is zero, as the cost going into any Router using Dijkstra's algorithm is always zero. Applying the algorithm in this case leads to the routing or forwarding table and Shortest Path Tree shown in Figure 3, where the link between **Net 15** and **Router D** shown in Figure 2 has been removed. [7]

These dynamic algorithms were modeled using ARENA which allowed students to monitor the



Net ID	Link Cost	Next Router From A (root)
1	2	-
2	3	-
10	1	-
12	3	B
15	5	B
22	6	F
61	4	E

Figure 3: The Shortest Path Tree after applying the Dijkstra algorithm and the resultant routing or forwarding table.

underlying protocol mechanisms. This is not apparent when using a real network as the mechanisms are transparent to network managers. This is , in fact, one of the advantages of using simulation over building an expensive physical network.

4. ARENA in Modeling Telecommunications systems

ARENA is a modeling package that will run on any Windows based Personal Computer. It is a graphical package which is based on the SIMAN's modeling language [13][16][17]. ARENA comes with many predefined modules such as **Arrive**, **Depart** and **Server**. The **Arrive** module allows an arrival process to be modeled. It is possible to use many different built in random processes such as exponential (expo()), uniform (unif()) or gaussian (norm()) distribution. The Server module allows a entity (usually modeled in the routing algorithm as representing a message or datagram or packet) to be queued and serviced given some random or fixed distribution.

The ARENA simulation, like many other network simulators (for example, OPNET [9] or ns-2 [8]), can be run, stepped and paused. This allows network wide variables such as routing tables to be examined at different time epochs. This is not normally possible with a physical network. One difference between ARENA and these other simulators is that it is more generic, and not specifically aimed at simulating telecommunication subsystems. It has to be modified to simulate aspects of such systems.

ARENA allows the network modeler to use sub-models to represent embedded systems within the model. It also allows the use of global variables which can be used, in the case of dynamic routing, to mimic and show individual routing tables throughout the network (regardless of whether using Distance Vector or Link state algorithms). These features were utilised in the design of all three simulators. Where possible standard modules were used, such as ARRIVE, or TALLY to collect statistics such as how many packets or entities have been successfully routed through the network. Also, in the case of Distance Vector a entity can have an attribute associated with it. This attribute may be a

simple hop count (which is implemented in the first two ARENA simulations) which allows the student to measure the hop count before and after the routing algorithm has converged. Such illustrations are difficult to achieve with a laboratory full of routers and switches and even if these were available, the cost would be much more than that incurred by using the simulator.

ARENA provides the ability for the student to setup and watch the values of variables within the simulation using the **watch** window. This is utilised in these simulations so that the student can observe the slow change in routing tables using the Distance Vector examples and the calculated routing tables using the link state database and shortest path calculations of Dijkstra.

The following provides basic information on selected experiments that were designed and implemented in this course.

5. Distance Vector Algorithms

The ARENA simulator was used to illustrate simple aspects of Distance Vector routing algorithms. The first experiment showed how the routing algorithm converges. The second ARENA simulation started by setting up a converged network and then by generating an entity that indicated that the link to a destination network had fallen over, which resulted in a simulated routing loop.

One of the first Distance Vector routing protocols was RIP (Routing Information Protocol). It was developed as a result of research at Xerox PARC (Palo Alto Research Center) [12]. This protocol defaults to Routing table updates every 30 seconds and this was the default value set up in the simulation for routing updates. The basic operation of the RIP protocol was implemented within the ARENA simulation called "RIPv1DV.doe".

In order to demonstrate the concepts involved in Distance Vector routing convergence, a network with fifteen interior routers, seven ingress nodes (routers) and three egress nodes (routers) was chosen. The configuration is shown in Figure 4. A set of global variables were setup to represent individual routing tables of each router in the network and initialised in a Variables module as shown in Figure 5. This was achieved by using an array for each routing table, for example the routing table for Router B (referred to as RB in

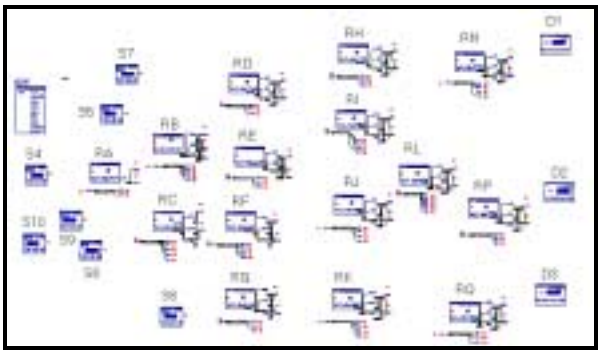


Figure 4: Chosen Network Topology for Distance Vector simulation (shows the look of simulation while it is being run).

Figure 4 above) was assigned to the array variable **BrouteTable**. This variable has 3 rows and 3 columns. Table 1 shows what each element represents and which variable in the **watch** window represents its value.

Figure 5: Variables module in ARENA simulation RIPv1DVc.doe

Figure 6: Initial values for Router B's, routing table from RIPv1DVc.doe

The routing table for RB is shown in Table 1. This was used in the laboratory instructions to show how the watch variables needed to be interpreted and thus understand how the individual routing tables were changing. The initial values for the Routing table for Router B were as shown in Figure 6. The first three values {1,2,3} showed the destination identifiers for the three destinations used in the simulation. The next three values {16,16,16}

showed the current number of hops to the destination. In RIP a hop count greater than 15 indicates that the destination is unreachable. The simulator also uses this as an indication that the destination is currently unreachable. The last set of values indicates which interface should be used to send out packets for particular destinations. In this case the values are set to zero and the built in default paths are used. When the simulation runs and actual routing paths are discovered, these will change from zero to one , two or three to indicate which outgoing interface is to be used. The concept of an interface could include an ethernet card or a serial communications port depending on the underlying Data Link and Network layer protocols being used in a real network.

Figure 7 shows a section of the simulation. In it, there are connections going from the individual interior routers (Routers A and B are shown here). Associated with each router is a sub-module which has connection points connected to green paths. These sub-modules generate and receive the individual routing messages generated every time an update occurs. This is the only way that the individual routing tables can be changed. Once these tables stop changing the dynamic routing algorithm is said to have converged and the destinations can be reached via a minimum number of hops.

Figure 8 shows a section of the sub-module associated with Router B. This code processes the received routing message, comparing the contents of the neighbours routing table (which is encoded in attributes inside the entity as would occur in a real packet based network) with the routers current routing table. If a quicker route to a particular destination is discovered this processing will update Router B's routing table before the routing message is disposed of in the Dispose module at the end.

The students record the routing tables before the dynamic routing algorithm starts, and then every routing update after this until they have identified where the routing tables stop changing at which point the Distance Vector algorithm has converged. The simulator allows the time at which the algorithm starts to be modified and also the time between routing messages may be varied.

ARENA simulation RIPv1DVLoops.doe is based on RIPv1DVc.doe, using the same number of interior routers (15), ingress routers (7) and egress routers (3). However, the sub-modules were dramatically re-designed so that all routers connected to the sub-module could provide routing update messages to it. This would normally be the case, but in RIPv1DV.doe only those routers in the forward (towards the destination) direction were able to feedback their routing tables.

	Column 1	Column 2	Column 3
	Destination	Hops	Interface Out
Row 1	{BRouteTable(1,1)} 1	{BRouteTable(1,2)} 16	{BRouteTable(1,3)} 0
Row 2	{BRouteTable(2,1)} 2	{BRouteTable(2,2)} 16	{BRouteTable(2,3)} 0
Row 3	{BRouteTable(3,1)} 3	{BRouteTable(3,2)} 16	{BRouteTable(3,3)} 0

Table 1: Shows how routing table information was assigned to arrays within the Arena simulation of the Distance Vector routing algorithm (RIPv1DVc.doe)

In the RIPv1DVLoop.doe simulation, the egress Router 5 (destination 2) sends a message to the interior Router P that the link has ceased to work. Router P adjusts its routing table accordingly, setting the entry for destination 2 from one hop to a hop count of sixteen, indicating it is unreachable. The surrounding routers are notified of this at the next update, but one of them tells P that it can get to destination 2 via another route. This route may be through Router Q, which has now been told its route is unreachable. The routing loop starts from this point and the hop count continues to increase till the hop count gets to 16 in all routers by which time the network knows there is a problem and packets destined for Destination 2 are disposed of at the ingress router. This is simulated by using the hop count at Router P (which is a global variable) and allowing packets destined for destination 2 to only proceed if this count is less than 16. The laboratory instructions included tables which the students were expected to fill in by observing the routing tables at each update epoch.

This simulation also was modified slightly to allow similar operation to the Border Gateway Protocol version 4 (BGPv4) such that when the link to Destination 2 was broken, Router P (acting as a BGP router) told all the other routers at exactly the same time of this event and, of course, no routing loop occurred.

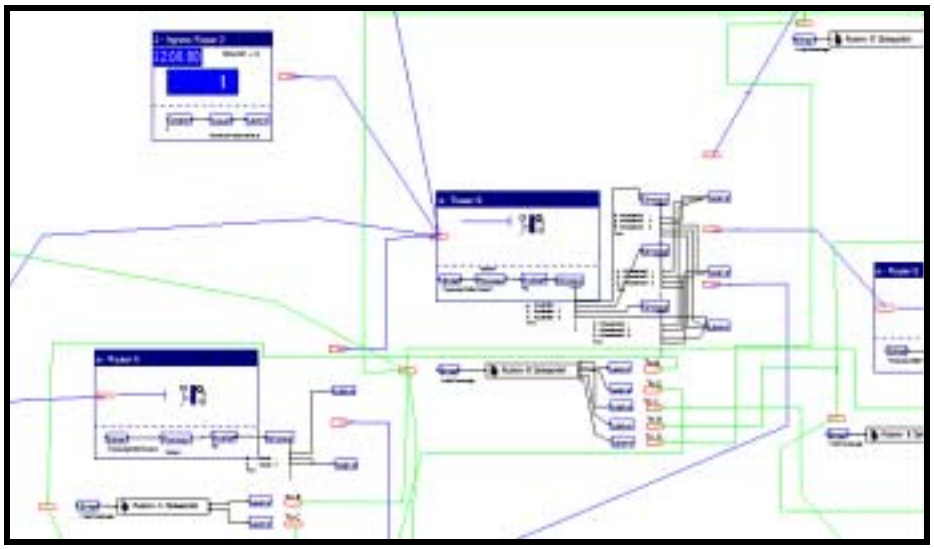


Figure 7: A snap shot of the simulator showing Routers A and B and the paths upon which packets and routing messages will travel. The green lines are the paths followed by routing messages and the blue lines are the paths followed by actual packets destined for one of the three destinations.

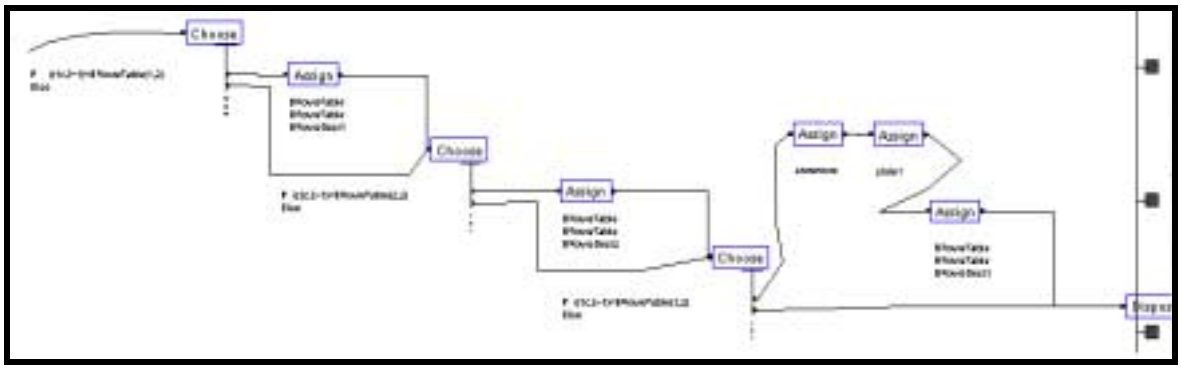


Figure 8: A section of the sub-module associated with Router B in the simulator RIPv1DVc.doe which updates the routing table by carrying out a series of comparisons and assignments before the received routing message is disposed of at the end of the routing messages processing.

	Column 1	Column 2	Column 3	Column 4
	Destination	Metric Cost	Interface Out	Next Hop
Row **1**	{r1_table(1,1)} 11	{r1_table(1,2)} 3	{r1_table(1,3)} 1	{r1_table(1,4)} 0
Row **5**	{r1_table(5,1)} 15	{r1_table(5,2)} 4	{r1_table(5,3)} 3	{r1_table(5,4)} 3
Row **7**	{r1_table(7,1)} 17	{r1_table(7,2)} 2	{r1_table(7,3)} 1	{r1_table(7,4)} 7

Table 8: Routing Table for link state simulator linkStateModel.doe, this example is for Router 1, Networks 11, 15 and 17 (in actuality the routing table has 9 rows for the 9 possible network destinations but only Network 9 has been set up as a destination in the simulator).

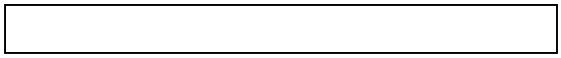
6. Link State Algorithms

The ARENA simulator was also used to illustrate simple aspects of typical Link State routing protocols, like OSPF. Unlike the Distance Vector simulation, each node is expected to form a topological database of the surrounding network. To do this calculation using cascaded modules, as implemented with the Distance Vector simulations, would have involved very many such modules and would have been very hard to test and debug. One of the available features built into ARENA 3.5 is the ability to use Visual Basic modules. ARENA has its own Visual Basic module designer and these modules can be activated on simulation events such as RunBeginSimulation or upon the entry of a entity into a VBA module.

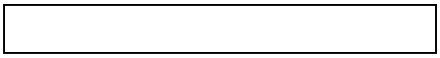
Every VBA module has a unique ID associated with it, and the name of the module in the Visual Basic editor window is automatically allocated. For example, the Visual Basic module associated with VBA Block identifier 1 is "VBA_Block_1_Fire()" and that associated with VBA Block indentifier 2 is "VBA_Block_2_Fire()" [13].

Visual Basic modules are used in many Windows based software packages. For example, Visual Basic modules can be used with Office 97 suite of products such as ACCESS '97 An example of this in which the primary author was involved in 2001 was a Power Quality database where Visual Basic modules were used to analyse the acquired data [14]. As ARENA is a Windows based product, it can also use Visual Basic modules to provide extra functionality that is not built into the normal ARENA modules. The built-in Visual Basic editor (accessible pressing the ALT and F11 keys simultaneously) provides a fully functional debugging environment with the ability to place break points and then monitor local variables and step through the Visual Basic code.

There are examples of using Visual Basic in ARENA provided in [13]. These were used to form the basic structure of the link state modules. Within the Visual basic editor it is necessary to import the global variables within the simulation into the VBA modules local variables and after calculating the resultant routing tables using the Shortest Path Algorithm of Dijkstra, export these back into the simulation so that packets can be routed through the simulator. To do this many local variables were declared in the Visual basic module. For example, to get a reference to the array cell r2_table(1,2) required the code:



To write this value back out to the simulation so that routing of packets for router 2 could be changed, the following code was required:



An array called **topo** was set up in the ARENA simulation which contained information about the topological database for the simulated network. This array had 18 rows and 5 columns. The third column of the **topo** array represented the metric cost for the corresponding link. The link identification was provided in a topological tree representation of Figure 9 as shown in Figure 10. The laboratory instructions indicated what set of values the students should set these link costs to during the demonstration. This was done in the Variables module so that different metric costs could be assigned to individual paths through the network. The network configuration chosen for the simulation is shown in Figure 9. It allows for five possible paths that the packets can be routed on. By changing these metric costs the routing algorithm would calculate the least total cost metric path through the network and this would be used.

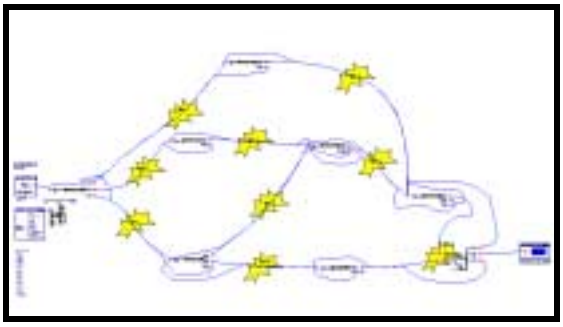


Figure 9: The Network used for the link state routing algorithms demonstration.

A additional feature also implemented in the simulator was a flag, called "LoadBalanceEnabled" which when set to '1' would allow the simulator to choose paths (at strategically pre-selected routers) to load balance the packets through the network in a similar manner to that which could be used in an actual OSPF implementation.

Within the lectures for the associated subject, the students had been instructed in how to use the Dijkstra algorithm to calculate the routing tables of individual routers. The laboratory instructions

asked the students to record in tables the resultant routing tables that were produced by the simulation for a given metric cost configuration. The structure of the resultant routing tables was similar to that shown in Table 1, except it had four columns instead of three. The first column was the Network destination identifier. For example, Net 1's identifier was 11 (all the Network identifiers ranged from 11 through to 19, for Net 9 and the Routers ranged from 1 through to 7, Router 1's identifier was '1' and Router 7's identifier was '7'). The second column shows the metric cost from the router it is in (example Router 1's routing table). The third column shows the interface number that the packet should be forwarded to and the fourth column shows the identification number of the next router. These are shown in Table 2.

The students were asked to verify that the computer simulations results were the same as that which the Dijastra algorithm would produce (as they were).

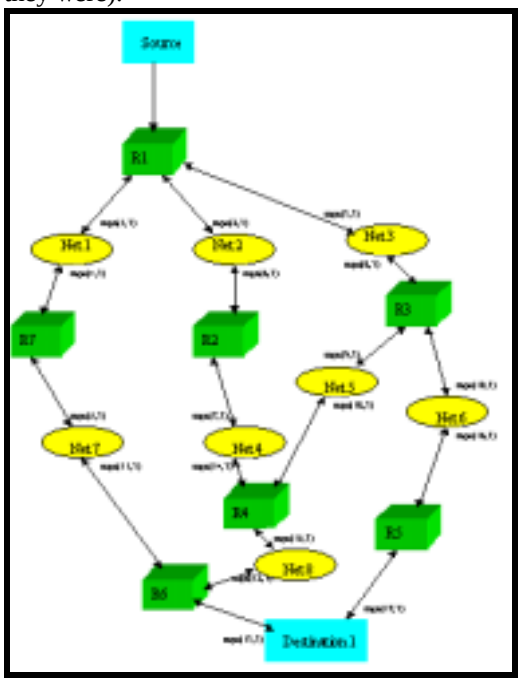


Figure 10: The topological tree showing the meaning of the third column of the topological database, topo.

The use of Visual Basic was very appropriate for implementing the Dijkstra algorithm. The temporary database was placed in an array called Tentative and then each branch starting at the router that the VBA module was located in with minimum cost was made permanent. This cycle was repeated until all nodes (routers and networks) were accounted for. The code itself does this two times per loop. The first section should always deal with routers (like R4) and the second section always deals with networks (like Net4). Initially it was found that sometimes, when link costs for a router and network were the same the first section would choose a Network link as the permanent path. This caused the code to never converge and it also resulted in erroneous routing tables. To solve this problem, the first section had extra code added which basically checks to see if there is a router with the same minimum cost as a network node, and if so, the router path is made permanent instead of the network nodes path.

7. Observations

The use of computer simulation was effective in teaching and demonstrating the dynamic routing algorithms for both Distance Vector and Link state routing. In the second laboratory (RIPv1DV.doe) the students were asked to create the tables in their laboratory notebooks. While this was effective, in the third and fourth laboratories an excel spreadsheet was used as a template. This allowed the students to concentrate on the material that was being illustrated. The feedback directly received from some of the students indicated that this was a more effective laboratory session than the second one had been because the structure was available. In teaching theory this is called scaffolding [15]. In the next incarnation of these demonstrations a excel spreadsheet template will also be used for this second laboratory demonstration.

8. Final Remarks

Computer Simulation is an effective way to educate undergraduates in dynamic routing algorithms. Its use allows concepts that may be difficult to explain and show through measurement to be illustrated. The computer Simulation ARENA which has been developed primarily for process simulation can also be used to model and illustrate telecommunications sub-systems such as those associated with dynamic routing. This was used in undergraduate teaching laboratories in spring session of 2001 within the Bachelor of Internet Science and Technology teaching programme. It was found to be an effective way to convey understanding of underlying dynamic routing protocols and their operation.

REFERENCES

1. Uyless Black, 2000, "IP Routing Protocols: RIP, OSPF, BGP, PNNI & Cisco Routing

Protocols", Prentice Hall Series in Advanced Communications Technologies
2. Paul Cernick, Mark Degner, and Keith Kruepke, 2000, "Cisco IP Routing Handbook", Professional Middleware, IDG books, pp 197-198
3. Paul Cernick, Mark Degner, and Keith Kruepke, 2000, "Cisco IP Routing Handbook", Professional Middleware, IDG books, pp 18-19
4. http://www.elec.uow.edu.au/people/staff/p.vial/
5. Paul Cernick, Mark Degner, and Keith Kruepke, 2000, "Cisco IP Routing Handbook", Professional Middleware, IDG books, pp 20-21
6. Uyless Black, 2000, "IP Routing Protocols: RIP, OSPF, BGP, PNNI & Cisco Routing Protocols", Prentice Hall Series in Advanced Communications Technologies, pp. 112-113
7. Behrouz A Forouzan, 2000, "Data Communications and Networking", McGraw-Hill, 2nd Edition pp. 633-640
8. "The Network Simulator ns-2", http://www.isi.edu/nsnam/ns/, visited October 2001
9. "OPNET", http://www.mil3.com/, visited October 2001
10. Mischa Schwartz, 1987, "Telecommunication Networks: Protocols, Modeling and Analysis", Addison-Wesley, pp. 212-223
11. Two postgraduate sessional thesis projects in the spring of 2001 were using ARENA to model metering aspects of SNMPv3 (Simple Network Management Protocol version 3) and simple flow aspects of WDM (Wave Division Multiplexing) Optical Routers. Both students did these projects under Mr P J Vial (BE (Hons 2 div 1) ME (Hons) DipEd Png) supervision as part of their Master of Engineering Studies program in Spring Session 2001, Mr Karthik Vilapakkam Nagarajan and Mr Frank Nordhal on Optical Routing.
12. Uyless Black, 2000, "IP Routing Protocols: RIP, OSPF, BGP, PNNI & Cisco Routing Protocols", Prentice Hall Series in Advanced Communications Technologies, p. 104
13. W David Keltan, Randall P Sadowski, Deborah A Sadowski, 1998, "Simulation with ARENA", McGraw Hill, covers ARENA 3.0
14. P.J.Vial, V W Smith, P J Vial, V J Gosbell & B S Perera, "Database Design for Power Quality Survey", AUPEC 2001
15. M.D.Roblyer, Jack Edwards, Mary Anne Havriluk, 1997, "Integrating Educational Technology into Teaching", Prentice Hall
16. David A Takus, David M Profozich, "ARENA Software Tutorial", Proceedings of the 1997 Winter Simulation Conference
17. Deborah Sadowski, Vivek Bapat, Glenn Drake, "The ARENA Product family: Enterprise modeling solutions", Proceedings of the 1998 Winter Simulation Conference