**Project 1**

**Evaluation of Adaptive Routing using Dijkstra’s**

***By, Urvi Gada***

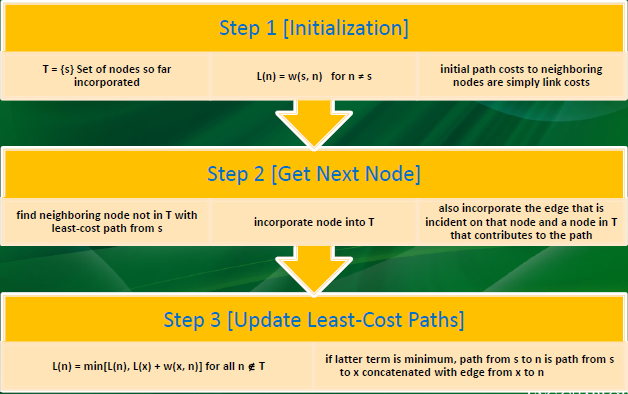
***Student ID-801029135 Email ID- ugada@uncc.edu***

**Objective:**

The objective of the project is to apply Dijkstra’s Algorithm to a network with adaptive link costs, to evaluate the effect of the adaptation rate on the network stability and performance.

**Theory:**

1. **Dijkstra’s Algorithm:** This algorithm is used to find the shortest path between two nodes. The algorithm works as follows:



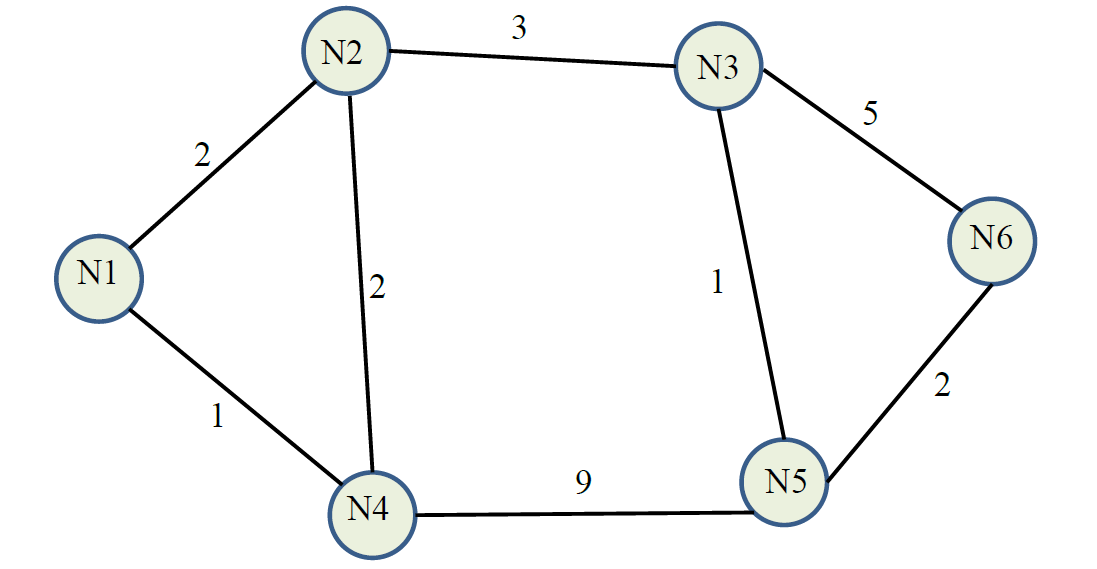
1. **Adaptive Routing:**

In adaptive routing, the routing decisions that are made change as the conditions on the network change. The two major factors that need to be considered are failure and congestion. When a link or a node fail, they can no longer be a part of the route. When a portion of the network becomes heavily congested, the packets must then be route through another portion of the network. This makes routing decisions complicated. However, adaptive routing greatly enhances the performance and aids congestion control. Thus, almost all packet switching networks use some sort of adaptive routing technique.

1. **Characteristics of the Network:**
2. **Network Stability-** A network is said to be stable when it can absorb large shocks and does not build up fragility from within. Using adaptive routing, makes the network more adaptable to changes. It determines how fast or slow the network must adapt, to avoid oscillations or uselessness. Thus, enhancing the stability.
3. **Network Performance-** The network performance determines how efficiently can the network transfer packets from one node to another without any errors, need of retransmission etc. Use of an adaptive routing algorithm must not only adapt to changing conditions but also enhance the overall performance of the network.

**Experiment Performed:**

The network on which Dijkstra’s Algorithm with adaptive link costs need to be applied is as follows:

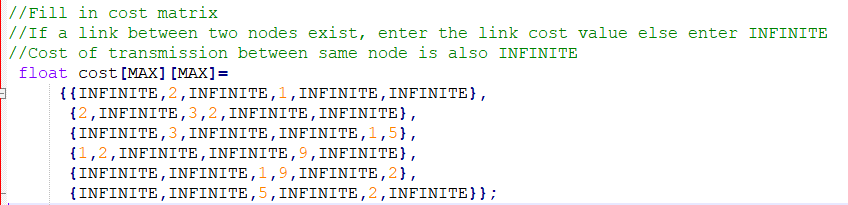


Here, the link costs are assumed to be the same in both directions.

1. The given C code for Dijkstra’s Algorithm calculates least path for a 7-node network. The network given above has 6 nodes. So, the following changes must be done in the code to make the code applicable for the given 6-node network:

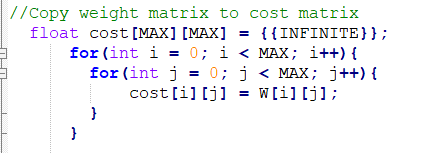


And,



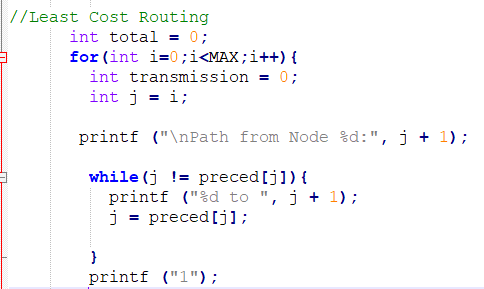
The Max macro needs to be changed which defines the number to nodes in the network. So, in this case it will be 6. Next, the cost matrix needs to be modified as per the given network. If a link exists between two nodes, the cost of that link is entered. Else enter the INFINITE value. Also, for the matrix entry where source and destination are the same node, INFINITE value is entered. E.g. (1,1).

1. The next task is to develop a program that can evaluate the adaptive routing that changes the link cost based on the current load on the link. The following steps needs to be executed:
2. **Initialization:** Here, a link cost matrix is defined as follows:



Here, W[i][j] represents initial link costs from node i to j. And the cost matrix is made to adapt as per the changing load on the link.

1. **Least Cost Routing:** Here, the given Dijkstra’s algorithm is applied to get the least-cost routes of all nodes to node 1 in the given network. The following is the code snippet for the same:

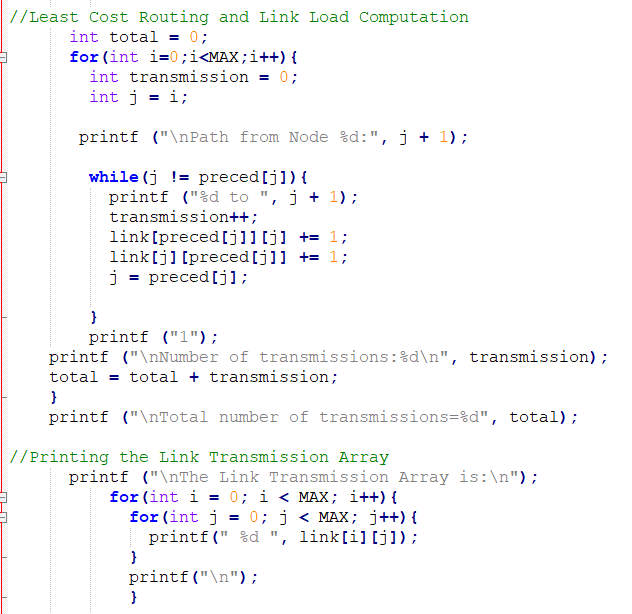


Here, we use the preced matrix to determine the shortest route from the source to the destination which in this case is node 1. The algorithm for this is as follows:



The preced matrix for the first iteration will be as above. The preced matrix contains the next node with the least cost that can route to Node 1 which is the destination. Thus, to determine the shortest route, we check until the value of preced[j] become equal to j. If it enters the loop, we print the (j+1) value which gives us the intermediate node. At the end, we print 1 which our destination node.

1. **Link Load Computation:** Here, for the set of routes obtained in the above step, we calculate the number of transmissions on each link L(i,j) assuming each node transmits one unit of data to be forwarded to Node 1. We use the same loop as above to determine the number of transmissions.



The preced matrix for the first iteration will be as below.



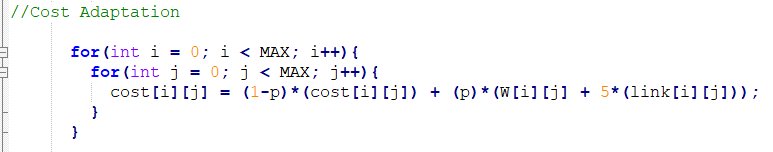
The preced matrix contains the next node with the least cost that can route to Node 1 which is the destination. Thus, to determine the shortest route, we check until the value of preced[j] become equal to j. If it enters the loop, we add 1 to the number of transmissions for that link. Simultaneously, we also update the Link matrix by using:

link[preced[j]][j] += 1;

link[j][preced[j]] += 1;

This will fill two positions in the matrix i.e. L[i][j] and L[j][i] because L[i][j] = L[j][i] as the costs for that link in both directions is the same. At the end, we calculate the total number of transmissions by adding all the individual transmissions and print the Link matrix.

1. **Cost Adaptation:** The link cost is adapted based on a variable parameter p as follows:



This adaptation is done for different values of p i.e. (0.2, 0.4, 0.6, 0.8, 1.0) and for each value of p we perform the above three steps 20 times in order to achieve a stable pattern.

**Observations:**

1. All the above steps were repeated multiple times for different values of p in order to obtain the number of transmissions and calculated costs for Link 3-2 and Link 5-4.

* Number of Transmissions for Link 3-2 and 5-4:

From the above graph we can observe the variation in the number of transmissions on Link 3-2 for p equal to 0.2, 0.4, 0.6, 0.8 and 1.0. For smaller values of p the variation for each iteration is small. However, for values of p from 0.6 to 1.0 we observe that the variation is larger.

For Link 5-2 we observe a similar trend however the pattern of variation differs. This indicates that each link in the network adapts differently to the changing network conditions independent of the other links. However, all changes are dependent on the p value which determines the amount of change in the network conditions.

* Calculated Costs for Link 3-2 and 5-4:

The above graph shows the variation in the calculated costs for Link 3-2. We can see that the variation is not very large for values of p equal to 0.2 and 0.4. However, the larger values of p show large variation in the calculated costs for each iteration. This indicates that the network conditions are changing very quickly.

For Link 5-4, we observe similar trends, however the range of variation is smaller than Link 3-2 as shown below:

* From the above 4 graphs we observe that the network conditions change depending on the value of p. A smaller p value indicates less variation in the network conditions and therefor, less variation in the calculated costs and vice versa.
* If the calculated cost for that iteration for a given value of p is small, then the number of transmissions on that link for that iteration will be higher. Similarly, if the calculated cost is large, then the number of transmissions on that link will be smaller.
* When the variation high, the network nodes must adapt very quickly to these changes. The chances of errors, retransmissions etc. also increase. However, in a network with fewer changing conditions, the nodes are not burdened to adapt too quickly. The performance of the system is better, and the network is also more stable as the probability of successful transmissions increases and need of retransmission, errors and overhead reduces. However, a rapidly changing system may reduce the performance and the stability is also affected as seen. The link costs and so the transmissions throughout that link have large variations.

1. For each value of p, the graph of the total number of transmissions in the network for each iteration is as follows:

From the above graph, we see that the variation in the total transmissions for smaller values of p is larger. This indicates, that when the network conditions change slowly, the network can adapt better and thus for a few iterations, the number total transmissions reduce. The shortest route is determined, and the number of hops from source to destination reduces. Thus, the transmission requires less power and fewer resources. However, for higher p values, there is no variation. The total number of transmissions is constant and has a higher value. Thus, faster changing conditions lead to larger number of transmissions in the network.

1. Additional adaptation method to improve the stability and performance:

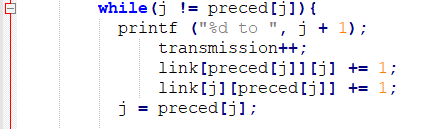
The above adaptation method assumes that the measured packet delay on a link is a good predictor of the link delay that is encountered when all nodes reroute traffic based on the reported delay. This can only be effective when there is some correlation between reported values and those that are experienced. This, correlation tends to be higher when the load is light, however, under higher loads, this correlation tends to be very small. When the load on one part of the network increases, all nodes simultaneously will route the packets through another portion thereby increasing the load on the other portion. This leads to oscillations.

The ARPANET third generation technique can be used to improve performance and stability. Instead of all nodes trying to find the best path to the destination, under heavy loads, the goal is to give average route a good path instead of best paths to all. Thus, the overall routing algorithm need not be changed, only the function that calculates the link costs must be changed. Thus, using the ARPANET third generation algorithm will help to improve the stability and performance of the above network.

1. Algorithm if the link costs are not symmetrical i.e. cost(i,j) ≠ cost(j,i).

The same code can be used to determine the routes and costs for asymmetrical link costs with the following changes in the code:

* Currently we are using only one preced matric for node 1 in one direction. For asymmetric link costs, we will have to calculate the preced matrices for each node as destination in both the directions.
* Since the link costs are the same in both directions, the link matrix can be filled in the same while loop as follows:



However, to make the code work for different costs, we will have to check the direction and then have a separate if-else condition in the while loop to fill in the Link matrix.

* Finally, a switch case will be needed, to determine the destination node, and the above logic will be executed on that node’s preced matric and the link costs, routes, link matrix etc. will be determined.

**Conclusions:**

The following conclusions can be derived from the various experiment steps performed:

* Most of the packet-switched networks use adaptive routing. The nodes are thus made to adapt to the changes in the network.
* The link cost adaptation is dependent on the value of p.
* For rapidly changing network conditions, the link cost variations are higher, and thus the number of transmissions on those links also show higher variations.
* For slowly changing network conditions, the link cost variations are smaller, and thus the number of transmissions on those links also show smaller variations.
* The total number of transmissions also depend on the changing network conditions. For a slowly changing network the total transmissions show a variation from high to low values. But for rapidly changing network conditions, there is no variation, and the total number of transmissions is high.
* ARPANET third generation algorithm can be used to further enhance the performance and stability of the network.

**References:**

1. Data & Computer Communications, by William Stallings, Tenth Edition, Prentice Hall. ISBN 9780133506488, 2013