Chapter 4

Traffic Sensitive Dynamic QoS Routing Algorithm
Chapter 4

Traffic Sensitive Dynamic QoS Routing Algorithm

"Science rests on reason and experiment, and can meet an opponent with calmness; but a belief is always sensitive."

James A. Froude

4.1 Introduction

QoS can be represented by several parameters like throughput, delay, jitter, loss, etc, and there have been several studies and discussions for providing QoS in the Internet.

The current approach for providing QoS may be divided into two types as discussed in Chapter Three. One is the approach based on Resource Reservation Techniques (RSVP). IntServ (Integrated Services) is the framework of this approach [67]. In this framework, the Routing resources are reserved for each of the flows through RSVP in order to assure the communication quality of them. This method assures communication quality in a restricted manner, but there is a fatal problem of scalability in this approach. Another approach is called as DiffServ [64]. DiffServ is the framework in which communication flows are classified into several classes in the DiffServ domain, and in the domain, traffic of each class is treated differently at each router according to its corresponding Per Hop Behavior.

The studies and discussions related to QoS in DiffServ frameworks concentrates on how to assure quality of flows that are congested, or in other words, how to allocate network resources to particular classes that are congested [75]. But, since the network resources are limited, a resource reservation for some particular classes reduces the
resources for other classes of flows. In this Chapter, we propose an algorithm that will consider an alternate path when a particular path becomes too congested by the reduction of flows to a particular path. The proposed approach is called as "Traffic Sensitive Dynamic QoS Routing Algorithm". This method invokes a procedure that will reduce the traffic in the congested link. The proposed method is a dynamic method and there are two severe problems which affect the communication quality whenever dynamic changes occur in a network. They are 1) Routing Loop Problem and 2) Path Oscillation Problem [76].

4.2 Routing Loop Problem

The main reason for Routing loops in the Internet traffic is because of the inconsistencies in the routing state. This makes packets to be delayed or discarded in their traversal. The Routing loops make packets to be caught in a loop and to traverse to the same link in the network multiple times, and to show up as duplicates of the original packet. This is explained with an example below. The proposed algorithm overcomes the routing loop problem.

Consider that a particular link in the Network is congested. This problem can be solved by making packets to bypass that congested link. A congested link denotes that there is a large amount of traffic going through that link. Hence, if the cost of the congested link is raised, then some of the shortest paths will consider an alternate path in order to avoid a high cost link [77].
Fig. 4.1 illustrates this situation. The number of shortest paths that go through the link from B to D decreases as its cost increases. Initially, when the cost of the link B to D is 4, the traffic from A and B to D go through B to D. The cost of B to D increases to 6. Now, the node A forwards its packets to D through C. Next, when the cost of B to D increases to 8, B forwards its packets to D through C relieving the congestion in the link B,D. This simple idea has not been implemented widely. The main reason why it is not implemented is that it can create temporary loops of packets, often called as the routing loops, which cause major problems like loss of connections and packet losses.

Fig. 4.2 explains about routing loops and about why they appear. There are three nodes, A, B, and C. The three nodes are connected with the help of the links with different costs such as 9, 3, 3, and 5. In Fig. 4.2(a), the shortest paths from B to A and also C to A going through the link labeled as (B,A). Suppose the cost of the link (B,A) is increased from 5 to 13 as in Fig. 4.2(b), the shortest paths now change as shown in Fig. 4.2(c) which does not seem to cause any problem outwardly. However there is a problem. Fig. 4.2(b) shows the phase in between B and A in which node B has known that the cost of (B,A) has increased to 13 but C does not know this. This causes a packet whose destination is A to
move along a loop between B and C. After some time the information of the changed cost is also known to C and the loop disappears as in Fig. 4.2 (c) [77],[78]. This amount of time delay is unacceptable in the Internet, and a method to overcome this problem should be found.

The solution for this problem is as follows: Consider Fig. 4.2(b). Here the cost of the link between A and B is increased from 5 to 13. Instead of increasing the cost directly to 13, consider a situation when the cost between A and B being increased gradually to 9. This is shown in Fig. 4.2 (d). It is found that there are no loops created in the transition phase. After the transition phase, the shortest paths are fixed as in Fig. 4.2(d) and then the cost of (A,B) is increased to 13 which does not cause any routing loop.

4.3 The Path Oscillation Problem

The Path Oscillation problem is one of the well known problems on Traffic Sensitive Dynamic Routing. A path repeatedly changes among several paths and does not converge. Suppose a link is congested and the cost of the link is raised, some of the flows passing the link will change its path to the second shortest one. The traffic amount of a link in the alternative path is increased and may cause congestion. Consequently, path of the
flows return to the original one, and this process may be repeated. This is the path oscillation problem [79]. The main problem is that the oscillation consumes a lot of network resources such as CPU processing power and thus capability of routers significantly lessens. For this problem, the consumption of CPU resources is suppressed within an acceptable level by taking enough time intervals between cost change events in the proposed work.

4.4 Open Shortest Path First

Since OSPF (Open Shortest Path First) is used in this work on “Traffic Sensitive Dynamic QoS Routing”, a brief introduction of OSPF is presented here. OSPF is one of the most commonly used routing protocols in the Internet. OSPF is a link state type of a routing protocol, where each router manages the whole network topology and calculates the shortest path. The speciality of the OSPF algorithm is that it dynamically restores various topology changes including failures and cost changes. When such a topology change occurs, the change information is circulated in the network and then every router calculates a new shortest path [24].

4.5 The Traffic Sensitive Dynamic QoS Routing

In this section, the “Traffic Sensitive Dynamic QoS Routing Algorithm” into DiffServ framework is introduced. This aims to overcome one of the limitations present in the DiffServ Framework. DiffServ provides quality differentiation in per class fashion by applying a different Per Hop Behaviour to each traffic class. But, since a Per Hop Behaviour defines only a queuing behavior and not a routing behavior, the limitation of resources of links cannot be avoided. For example, when a link is congested, in order to
assure the quality of high precedence flows, DiffServ will drop the packets of low precedence flows. The number of drop packets cannot be decreased no matter how the queuing mechanisms are used. This is one of the current limitations of DiffServ.

The traffic in a link is controlled by changing the cost of the link in the proposed "Traffic Sensitive Dynamic QoS Routing Algorithm". By raising a link cost, the amount of traffic in the link can be reduced.

The same transition can be applied for the general case as follows: The traffic in a link can be flexibly reduced by raising its cost to a proper value. On the other hand, when traffic becomes less, then the link cost has to be reduced. In the proposed system, the traffic in every link is monitored by the Cost Manager in the DiffServ Framework and when it reaches the threshold value, the traffic is controlled by changing the link cost dynamically. In other words, the cost of a link is raised dynamically, whenever the traffic increases, and waits for the feedback effect in traffic. The cost to change must be within the safe range, but if the feedback effect is not enough, then the cost has to be raised again.

It is to be noted here that the change of path for each and every flow lessens the communication quality since the path switching operation invokes instability. Hence, the change of cost should be performed only when it is necessary. The cost is raised only in case of congestion. Also, the change of cost is done only for a low precedence class.

Since there is waiting time for a feedback effect in each step, and also because each step includes the period of path convergence of Open Shortest Path First, the traffic is slow.

The slow traffic is acceptable because of the two reasons. The first reason is on the path oscillation problem. The main problem related to the path oscillation problem is that
the oscillation consumes CPU processing power and thus the capability of routers considerably decreases. However, the slow path control in this system is able to keep the loss of capability in an acceptable level. The second reason is due to the average transition of network load. The traffic load transits often and it is usually not in a selected manner. The proposed work takes this transition into consideration. In the proposed work, only a single cost change can be done. If this is not done, then there is a possibility of routing loops. Hence, a server which prevents simultaneous cost changes is prepared by giving a permission of cost changes selectively. In the system, the server becomes the leader of parallel execution of the algorithm. It is to be noted that the algorithm can be executed independently per destination.

4.5.1 The Network Model

A network consisting of N nodes is considered as our model. A node can be a host or a router. The nodes in a network are connected by physical links along which all the packets can be transmitted. Node (i) and Node (j) are considered neighbors of each other if there is a physical link between them. The physical link between them is identified by the Link (i, j) and the Cost of transmitting a packet over Link(i, j) is denoted by Cost(i, j). Only Cost is considered here, since this is the only parameter considered in the proposed algorithm.

4.5.2 Analysis and Notifications

The analysis that is done before going into the development of the proposed algorithm is as follows.
1. Monitoring of Traffic: The algorithm must have mechanisms for determining that the current path is congested.

2. Identifying the problematic links: The algorithm must have mechanisms for alternate paths based on a threshold value whenever there is congestion between links.

A system related issue to be considered is the transaction between traffic control and path oscillation. In other words, the time interval between cost change events is to be found. If the interval is too short, the path oscillation goes high. On the other hand, if the interval is too long, traffic control will be too slow to catch up with the dynamic change of network traffic.

Also an upper limit of the link costs to be raised should be defined in order to prevent redundant path oscillations. Consider a situation when the network is saturated with packets and when some link is congested, the cost is raised and some flows passing the link bypass it. The congestion, however, is not resolved and then another link cost is raised in turn. If this process is repeated, costs will be raised infinitely. To resolve the problem, the maximum cost for each link should be defined. In almost all cases, a few times of cost changes are enough to control traffic properly. The upper limit for each link should be set to make the traffic well balanced.

4.5.3 Flowchart

Fig. 4.3 gives the Flowchart of the proposed Traffic Sensitive Dynamic QoS Routing Algorithm.
Figure 4.3. Traffic Sensitive Dynamic QoS Routing Algorithm
There are some Notifications used in the proposed algorithm. They are used to communicate the information from one node to its neighbors in order to facilitate in the execution of the algorithm. Following are the Notifications needed to implement the proposed algorithm

1. **Congestion Notification**: This is a Notification that is sent by a node to its neighbors, whenever it encounters congestion in one of its outgoing links. Each node in the network will have a threshold value set in order to detect congestion. Whenever the load on an outgoing link is above the set threshold value, then a congestion notification message is sent by the node to its neighbors. This message can be given as \(\text{CongNote}(i, j)\) which denotes that a congestion is experienced on the Link\((i, j)\) by node \(\text{Node}(i)\).

2. **Congestion Rectification Notification**: When a call for congestion rectification is made, it invokes a procedure which increases the cost of the link step by step which in turn automatically diverts the traffic that are using that link. This reduces congestion in that link.

3. **Alternate Path Request Notification**: Whenever a node receives congestion notification intimation, it may send an alternate path request notification to one of its neighboring nodes. Whenever this notification represented as \(\text{AltPathReq}(i, j, k, l)\) is sent from node \(\text{Node}(k)\) to node \(\text{Node}(p)\), then this message is meant to let \(\text{Node}(p)\) know that \(\text{Node}(k)\) is going to change the route of the packets to \(\text{Node}(p)\) from node\((l)\) instead of its OSPF neighbor \(\text{Node}(i)\), since congestion is reported in Link\((i, j)\).
4. Congestion Completion Notification: Whenever a link is no longer congested, the nodes connected to it send this message to its neighbors to inform about the change of state. This intimation is represented in the proposed algorithm as CongComp(i, j) which denotes that a congestion is no longer existing on the Link(i, j) by node Node(i).

5. Alternate Path Completion Notification: This is a Notification that is sent by a node whenever it receives CongComp intimation from its neighbor and whenever this node is temporarily routing packets through alternate path because of a congestion notification. If Node(k) was temporarily rerouting packets from Node(p) to Node(q) because of congestion in Link(i, j), then Node(k) will send out AltPathComp(i, j, k) to Node(q) whenever it receives CongComp(i, j) from its neighbor Node(i).

The flowchart given in Fig. 4.4 works as follows: Each link in the network has a threshold value in order to detect congestion. If the traffic crosses the threshold value then CongNote is sent to all neighbors except Q. A CongRect notification is sent which in turn increases the cost of the link with a time delay. A request for alternate path is made which calculates an alternate path based on OSPF and forwards packet through the alternate paths. Link P,Q is again checked for congestion and the process is repeated till there is no congestion in the link P,Q. When the congestion is relieved, a Congestion Completion notification is sent. AltPathComp Notification informs that there is no further need for the alternate path and the link P,Q can be used again. Then packets are transmitted through P,Q.
4.5.4 Proposed Algorithm

Start

While P detects congestion on a link (P,Q)

Do

{ 
    Send CongNote (P,Q) to all neighbors except Q

    CongRect (P,Q)

    { 
        Cost (P,Q) = Cost (P,Q) + d;

        Timer();
    }

    AltPathReq (P,Q)

    { 
        Calculate alternate path based on OSPF and forward packets through alternate path;
    }

    CongComp (P,Q)

    AltPathComp (P,Q)

} 

Transmit packets through congestion rectified path (P,Q).

Stop.

4.5.5 Experiments and Evaluations

The Evaluations are done using GNU (GNU’s Not Unix) Zebra [81],[82] a routing software package that provides TCP/IP based routing services. Zebra uses an advanced
software architecture to provide a high quality, multi server routing engine. It supports the existing routing protocols such as RIP, OSPF and BGP. Among these, OSPF is considered in this evaluation. GNU Zebra uses a VTY (Virtual TeletYpe interface) command line interface which allows a user to modify and control all parameters. In addition, Zebra itself has a command line interface for dealing with Zebra specific commands. The procedures are written to increase the cost of the congested link with a delay and to forward the packets through alternate paths. To allow access to the functions, new VTY commands are installed into the system. These VTY commands accept cost and delay as arguments and pass them to the internal functions. The internal function sends congestion note to neighboring nodes and calls for congestion rectification which increases the cost of the link, calculates alternate paths and forwards packets through the new path. EXPECT scripts [83] help to divert the traffic of the congested link. With this new set of functions within the OSPF and with simple EXPECT scripts, traffic in the congested link can be reduced in an iterative manner.

The Experiments and Evaluations are done for the “Traffic Sensitive QoS Routing Algorithm” with the Network model given in Fig. 4.4. This model network has five nodes and six links. The Routing of packets is done initially from node A to D.

![Figure 4.4 The Network model](image)
The inputs to the simulator are Model Network with five links and six nodes, Cost for each link, Threshold value for traffic in each link, Time delay in msec, cost to be increased and source-destination pair.

Figure 4.5 Traffic Sensitive Dynamic QoS Routing Algorithm Screen.

Fig. 4.5 shows the screen with the model network chosen for the evaluations. The screen also shows the “Send” and “Reset” options. The process followed by the simulator is when a source and a destination nodes are entered, the “Send” option calculates the shortest path based on OSPF.

The shortest path calculated by the “Traffic Sensitive Dynamic QoS Routing Algorithm” is displayed in green colour in Figures 4.6, 4.7, 4.8, and 4.9. The “Reset”
option sets the default values for the links of the network. The “Cost” displays the total cost of all the links to reach from source to destination.

Fig. 4.6 shows the shortest path calculation based on the proposed Traffic Sensitive Dynamic QoS Routing Algorithm with cost = 3 between the nodes B and D. Figures 4.7, 4.8 and 4.9 show the screen for cost = 5, 7 and 9 respectively.

Figure 4.6 Traffic Sensitive Dynamic QoS Routing Algorithm for cost = 3.
Figure 4.7 Traffic Sensitive Dynamic QoS Routing Algorithm for cost = 5.

Figure 4.8 Traffic Sensitive Dynamic QoS Routing Algorithm for cost = 7.
The Table 4.1 shows the values generated by the simulator for the number of paths passing through the links to the destination D from all other nodes. The evaluations are recorded for costs ranging from 3 to 9 in the link B-D. For example, when the cost of the link B-D is 3 then the number of paths passing through the link A-B is 1.

**Table 4.1** Number of paths passing through the links for cost 3, 5, 7 and 9.

<table>
<thead>
<tr>
<th>Cost / Link</th>
<th>A-B</th>
<th>B-D</th>
<th>A-C</th>
<th>B-C</th>
<th>D-E</th>
<th>C-E</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
Fig. 4.10 shows the graphical representation of how the network model behaves with respect to the cost changes ranging from 3 to 9 between the nodes B to D. It also clearly shows how the traffic reduces between the nodes B and D and at the same time how the traffic increases along the nodes D to E. Now the traffic in link D to E crosses the threshold value, and the proposed algorithm is invoked to reduce its congestion. This leads to path oscillation. However, since there is a delay between cost changes, this path oscillation problem is considerably reduced. This work is carried out on a small network and hence the path oscillation problem is obvious. But when the proposed algorithm is implemented in a large scale network, the path oscillation problem is solved.

![Graphical representation](image)

Figure 4.10 Behaviour of the Network model with respect to Cost Changes

The earlier methods mentioned in the literature survey, either reduces the traffic or increases the bandwidth of the congested link. In this proposed work, the traffic and bandwidth are not altered. Instead, an alternate path is found for the congested link by increasing the cost dynamically there by reducing the traffic in the congested link.
4.6 Conclusion

In this Chapter, The Traffic Sensitive Dynamic QoS routing into DiffServ Frameworks is proposed. Using this algorithm, an alternate path is found if a link is congested and also measures are taken to reduce the congestion in the link by increasing the cost in regular intervals. The problems associated with Dynamic Routing algorithms like Routing loops and Path Oscillation are solved using the proposed algorithm. The proposed algorithm is executed using GNU Zebra software and the screens are displayed. Even though time delay and Cost change in only one link is included in this work, there is still some path oscillation in the network. As a future work, the proposed algorithm can be extended by making changes collectively in bandwidth, cost, load and delay.