A New Fault Tolerant System for Multimedia QoS Guarantee on the Internet

OMAR S. ESSA
Computer Science Dep., Faculty of Science,
Menoufia University,
EGYPT

NIKOS E MASTORAKIS
WSEAS European Office
Agiou Ioannou Theologou 17-13,
15773, Zografou, Athens,
GREECE
dromar_essa2005@yahoo.com
mastor@wseas.org
http://www.wseas.org/mastorakis

Abstract: - Multimedia streams travel across a path that connects the source to the destination. This path should provide the multimedia streams with all required Quality of Services (QoS). Suppose that the determined multimedia path gives the multimedia streams all the needed services. Consequently; the multimedia streams flow in this path to reach the destination. In this situation, some sort of ambiguous question arises. The question is, what will be done if, during the multimedia streams flow in the path, that path is failed? This state causes a big problem especially when the multimedia streams are transported under the User Datagram Protocol (UDP). This paper demonstrates the answer about the mentioned question by introducing analysis and design of a system that creates an alternative path in the case of original path failure. The proposed system considers the resulted network problems in the alternative path creation. Also, the paper introduces an implementation of the suggested system using a network simulator called NS2. Performance evaluation of the system also is demonstrated.

Key-Words: - Multimedia Protocols, TCP/IP, QoS, Fault Tolerance, Internet Protocols

1 Introduction
The Internet protocol is datagram oriented with no notion of connections whatsoever. The connections needed for communication are formed by the hosts involved, not the network between them. The Internet Protocol (IP)-networks are also without support for characterizing different types of traffic by the quality of service they need. All users get "best effort" traffic regardless of their real needs. New multimedia applications demand more bandwidth and are less tolerant to jitters, delays and lost packets than the traditional data applications that have so far dominated the networks. Real-time QoS of the network is essential for the applications on low capacity links and over long hauls. These additions to the network help to keep the involved programs simpler too. In order to a network to cover the QoS needs for the applications, its resources must be reserved and controlled [1], [2].

Current routing protocols find the shortest path from the source to the destination, then the Resource Reservation Protocol (RSVP) must try to reserve the resources from this path. There is no way to reserve resources from another path, in case of a default path failure [3], [4]. So, the solution is to create an alternative path and change the multimedia streams away to flow in the new path. Or, all the streams that are failed in the transmission process are retransmitted. The second solution is so difficult (if not impossible). This is because the quantity of lost multimedia streams may be too huge to be retransmitted. So, the only available solution is to create another alternative path and complete the transmission process. To determine an alternative path, we face two open questions. The first question is: How a free path, that will transport the multimedia streams to the same destination, is created? The second question that may be put forward after the path creation is; Can the created path provide the required QoS that was found in the failed one?

This paper is organized as follows. In section 2, we illustrated the related work that contains the RSVP analysis and DiffServ && MPLS evaluation; in section 3, we introduced the problem definition; in section 4, we demonstrated our system; in section 5, we introduced a detailed simulation and evaluation of our system. Finally, we illustrated the conclusion and the future work.

2 Related Work
2.1 RSVP Analysis
In this subsection, we introduced a simple analysis to clarify how the reservation and QoS processes are done using RSVP. This analysis helps to extract and identify the multimedia QoS problem(s). The RSVP analysis comprises the following: how the RSVP deals with routers or routing protocols? and how the RSVP model operates?

2.1.1 RSVP soft state implementation
In the context of an RSVP, a soft state refers to a state in routers and end nodes that can be updated by certain RSVP messages. The soft state characteristic permits the RSVP network to support dynamic group membership changes and adapt the changes in routing. In general, the soft state is maintained by the RSVP-based network to enable a network to change the states without consultation with end points. This contrasts with a circuit-switch architecture in which an end point places a call and, in the event of a failure, places a new call. RSVP protocol mechanisms provide a general facility for creating and maintaining a distributed reservation state across a mesh of multicast and unicast delivery paths.

To maintain a reservation state, RSVP tracks a soft state in router and host nodes. The RSVP soft state is created, then periodically refreshed by a path and the reservation-request messages. The state is deleted if no matching refreshes messages arrive before the expiration of a cleanup timeout interval. The soft state can be deleted also as a result of an explicit teardown message. RSVP periodically scans the soft state to build a forward path and the reservation-request refresh.

When a route changes, the next path message initializes the path state on the new route. Future reservation-request messages establish a reservation state. The state on the now-unused segment is timed out [5]. (The RSVP specification requires initiation of new reservations through the network two seconds after a topology change.)

2.1.2 RSVP operational model
The RSVP resource-reservation process initiation begins when an RSVP daemon consults the local routing protocol(s) to obtain some routes. A host sends Internet Group management Protocol (IGMP) messages to join a multicast group and RSVP messages to reserve resources along the delivery path(s) from that group. Each router that is capable of participating in resource reservation passes incoming data packets to a packet classifier and then queues them as necessary in a packet scheduler. The RSVP packet classifier determines the route and QoS class for each packet. The RSVP scheduler allocates resources for transmission on the particular data link layer medium used by each interface. If the data link layer medium has its own QoS management capability, then the packet scheduler is responsible for negotiation with the data-link layer to obtain the QoS requested by RSVP.

The scheduler itself allocates packet-transmission capacity on a QoS-passive medium, such as a leased line, and can also allocate other system resources, such as CPU time or buffers. A QoS request, typically originating in a receiver host application, is passed to the local RSVP implementation as an RSVP daemon.

The RSVP protocol passes the request to all the nodes (routers and hosts) along the reverse data path(s). At each node, the RSVP program applies a local decision procedure called admission control to determine whether it can supply the requested QoS. If the admission control succeeds, then the RSVP program sets the parameters of the packet classifier and scheduler to obtain the desired QoS. If the admission control fails at a node, then the RSVP program returns an error indication to the application that originated the request, see Fig. (1) [5].

2.2 DiffServ and MPLS
The DiffServ and MPLS are the trials protocols that provide some sort of satisfactory in fault tolerance problem. In the following subsections, we will give a brief discussion and evaluation of DiffServ & MPLS.

2.2.1 Brief background
MPLS (Multi-Protocol Label Switching) [6], which is regarded as a core technology for migrating to next generation Internet can support various high-speed and valuable services by high-speed switching
and traffic engineering. The goal of MPLS QoS has been to establish parity between the QoS features of IP and MPLS, not to make MPLS QoS somehow superior to IP QoS. One of the main reasons that MPLS supports the IP QoS model is that MPLS, unlike IP, is not an end-to-end protocol.

IntServ (Integrated Services) and DiffServ (Differentiated Services) are the methods that provide QoS in IP networks. Since both IntServ and MPLS use RSVP (Resource Reservation Protocol) as the signaling protocol to reserve resources, supporting IntServ over MPLS is not difficult. DiffServ does not reserve resources for each flow but classifies flows depending on their properties. It determines the Behavior Aggregation (BA) for each class at edge routers and forwards or drops the packets according to their BAs at core routers.

The DiffServ model is appropriate for MPLS, which makes routing decisions at edge routers and forwards packets at core router. DiffServ defines traffic profiles that have to be satisfied by packets being forwarded. For EF (Expected Forwarding), resources for the peak data rate are reserved whereas resources for the committed data rate are reserved for AF (Assured Forwarding) [7].

2.2.2 DiffServ and MPLS limitations

After study and analysis of the DiffServ and MPLS, we found these drawbacks and limitations that make them not an ideal (general purpose) solution for fault tolerance problem:

1. No provisioning methods
2. No signaling (as RSVP)
3. Works per hop (i.e. what to do with non-DS hop in the middle?)
4. No per-flow guarantee.
5. No end user specification.
6. Large number of short flows work better with aggregate guarantee.
7. Long flow (voice and Video) and flows with high bandwidth (as 1Mb of video) need per flow guarantee.
8. Difficult to design end-to-end service if the guarantee is only in the hop boundaries (Only EF will work)
10. How to ensure resources availability inside the network?
11. DiffServ is unidirectional – no receiver control
12. Works only on the IP layer.
13. For over-provisioned networks, each flow will receive its target rate, but with unfair sharing of the excess bandwidth.

14. For under-provisioned networks, the high RTT flows will be further away from the target flow

3 Problem Formulation

The routing and resource reservation protocols must be capable to adapt a route change without failure. When new possible routes pop up between the sender and the receiver, the routing protocol may tend to move the traffic onto the new path. Unfortunately, there is a possibility that the new path can’t provide the same QoS as the previous one, regardless the functional path could. To avoid these situations, it has been suggested that the resource reservation protocol should be able to use a technique called the route pinning [8]. This would deny the routing protocol the right to change such a route as long as it is viable. Route pinning is not as easy to implement as it sounds. With technologies such as Classless Inter-Domain Routing (CIDR) [9], a pinned route can use as much memory from a router as a whole continent! Also, this problem may be occurred if a path station can’t provide the multimedia streams with the same required QoS during a transmission operation. At this situation, the multimedia streams should search about an alternative path to complete the transmission process.

4 The Proposed System

From the problem definition and the RSVP analysis, it is obvious that the elements of the resource reservation and QoS are RSVP, routing protocol, sender, and receiver. Also, it is notable that the recourse reservation process is occurred before the multimedia transmission. At the beginning of the multimedia streams transmission (i.e. after the recourses are reserved for the multimedia), the relations between the QoS elements are disjoint. So, if a change is occurred in the reserved path during the multimedia streams transmission operation, the previous stated problem may be occurred. If the connections between the QoS elements are reinstalled during the multimedia streams transmission, then the QoS problems may be solved. The reinstallation process is accomplished by three additive components that are called the proposed system components.

4.1 The proposed system components

The proposed system comprises three additive components in addition to the old system components. The additive components are 1- Connector. 2- Analyzer. 3- Detector. In the
following subsections, the definition and the functions of each additive component are demonstrate.

4.1.1 Connector
This component is fired at the transmission starting and can be considered as a software class(s). The connector has more than one task for helping the system to accomplish its target. The main function of the connector is to install the connections between QoS elements in a problem occurrence case.

4.1.2 Analyzer
This component, located at the receiver, is considered also as a software class(s). The main function of the analyzer is to extract the failed station(s) and its alternative(s). Also, the analyzer connects to RSVP at the receiver site to extract a QoS request or a flow description of the new path. Also the analyzer uses some features of DiffServ and MPLS to acquire an alternative simple path with full QoS requirements. The DiffServ provides the system with simplest path and push the complexity to the network edges. The MPLS provides our system with next hope for each packet and to perform traffic conditioning on traffic streams flow in different domains (paths).

4.1.3 Detector
The detector and the connector are fired simultaneously. The detector proceeds the connector in visiting the multimedia path’s stations. The detector visits each path station to test the required QoS. If the detector notes a defect in the QoS at any station (i.e. the station can’t provide the required QoS), then it sends to the connector an alarm message containing the station IP address and the failed required QoS.

4.2 System approach
After the recourse reservation processes have been done, the multimedia streams begin the flood across the predetermined path. The connector accompanies the multimedia streams at every station. When the connector receives an error message from the detector, the connector starts to install the connections between the QoS elements.

4.3 System messages
To complete the connections between proposed system components, we have to demonstrate the structure of each used message. The proposed system contains five new messages that can be stated as follows.
1- From the connector to the sender.
2- Between the connector and the routing protocol (router) (Request and Reply).
3- Between the connector and the analyzer (Request and Reply).
4- Between the analyzer and RSVP at the receiver site (Request and Reply).
5- From the detector to the connector.
4.3.1 From the connector to the sender
This message joins the connector with the multimedia sender. This message is sent when the connector receives the QoS request from the analyzer. This message structure is like the RSVP reservation-request message but with the connector ID (This field is used in case of more than one connector in the proposed system) [5].

4.3.2 Between the connector and the routing protocol (Request and Reply).
This message joins the connector with the router or the routing protocol. This message is fired when the detector alarms the connector that a QoS failure is occurred at a station in the multimedia path. The connector needs this message to access the alternative path (or station) that takes place the failed path (or station). There are two types of this message, the request message and the reply message. The request message comprises the failed path and the reply message contains the alternative path. The request message has the following fields, 1) Message type, 2) Container ID, and 3) Old path. The reply message has the following fields, 1) Message type, 2) Connector ID, and 3) Alternative path(s).

4.3.3 Between the connector and the analyzer (Request and Reply).
This message is used to communicate the connector and the analyzer. This message is fired when the connector needs a QoS request for the new path. The message has two types, the request message and the reply message. The request message contains a new path that is accessed from the router. The reply message contains the QoS request that is extracted after the analysis operation. The request message contains the following fields 1) Message type, 2) Container ID, and 3) Alternative path. The reply message contains the following fields 1) Message Type, 2) Connector ID, and 3) QoS request.

4.3.4 Between the analyzer and RSVP at the receiver (Request and Reply)
This message is used to complete the dialog between the analyzer and the RSVP at the receiver site. The analyzer handles the old path and its alternative(s) to extract the failed station(s) and its corresponding station(s) in the new path. The analyzer needs it to construct a QoS request for the new path (s). This message has two types, the request message and the reply message. The request message contains the new path that was sent by the connector. The reply message contains the QoS request that is extracted by the RSVP. The request message contains the following fields, 1) Message type, 2) Analyzer ID, and 3) Alternative path. The reply message contains the following fields, 1) Message type, 2) Analyzer ID, and 3) Required QoS.

4.3.5 From the detector to the connector
This message can be used to alarm the connector with new event occurrence. If the detector finds a fault at a station in relation to QoS, then it sends this message to the connector asking to start its function for solving the problem. The message contains the following fields, 1) Message Type, 2) Connector ID, 3) QoS request, and 4) Address of the failed station.

5 Performance Study
In this section, we study the performance of the suggested multi-resource reservation system. We used in our simulation a network simulator called NS2 [10]. We simulate a distributed reservation-enabled environment, with multiple distributed services deployed and multiple clients requesting these services. In particular, for runtime computation of end-to-end multi-resource reservation plans, we compare the performance of the proposed system with the best effort communication system (old system). The key performance metrics in our simulations are: 1) Successful reservation, 2) End-to-end QoS level, 3) Packet loss probability, 4) Delay jitter, and 5) efficiency of additive components. These parameters are evaluated for an increasing network load. The network load dictates the amount of cross traffic being generated at the intermediate nodes. Also, in our simulations, we compare between our system and the DiffServ & MPLS [6], [7].

5.1 Simulation setup
We demonstrate, in this subsection, the network infrastructure that is used in our simulation. Also, we introduce the structure of the service’s model that creates the inputs and output qualities of compound services.

5.1.1 Simulated Network Infrastructure
The simulated environment is shown in Fig. (4). There are four high performance computers H1 to H4, as well as a number of client machines (1000) in eight different domains D1 to D8. These hosts are connected by high speed network links (L1 to L14). Four different distributed services (S1 to S4) are deployed in our environment.

For simplicity, the simulator assumes the presence of three alternative paths between the host
and the destination. These paths have one, two, three, and four routers along them respectively. Each node is interconnected by a 1Mbps link. The propagation time between each node is assumed to be 3 ms. Queue length at each link is 0.9% of link capacity. Each intermediate node assumes the presence of 20 cross traffic generators. Each of these cross traffic generators generates traffic, which is categorized into three types: multimedia, control, and the miscellaneous. We assign the highest priority to multimedia traffic followed by control and other traffic. The control and the miscellaneous traffic are generated using the Poisson distribution.

We simulate the detector using a token that travels across the path before the connector. The danger flag, found in this token, is tested at each path station. If this flag value is changed to one, then the detector alarms the connector with failed QoS at the visited station. The connector, in our simulation, is an agent. This agent is fired at the beginning of multimedia streams transmission. The connector agent may be destroyed when the last packet in the multimedia streams is received. Also, in our simulation the analyzer is an agent. The analyzer agent is fired when the connector value is changed to one and it is destroyed at the receiver when the multimedia streams transmission is finished. Also, the old system in our simulation environment is the RSVP system. This system contains a sender and a receiver at multimedia session. The old system is controlled by the RSVP for QoS and recourses reservation guarantee [11]. For further details of the simulator, its implementation and the simulation parameters, see [10].

5.1.2 QoS Resource Model for each service

For service $S_i$ ($1 \leq i \leq 4$), the main server is $H_i$. The dependency graph of $S_i$ involves a chain of three service components $ciS \rightarrow ciP \rightarrow ciC$: $ciS$ is the server-side service component running on host $H_i$; $ciC$ is the client side service component running on the requesting client; and $ciP$ is a proxy service component running on host $H_j$ ($1 \leq i \leq 4$, and $j = i$) depending on which domain the client is from. For example, in Fig. (4), if a client in domain $D_1$ requests service $S_4$, then the service session will involve service components $c_4S$ on $H_4$, $c_4P$ on $H_1$, and $c_4C$ on the client itself.

* For each service component $ciS$, $ciP$, or $ciC$, Fig. (5) and Fig. (6) show the Quality input ($Q_{in}$) and Quality output ($Q_{out}$) levels and the corresponding resource requirements: Fig. (5) is for services $S_1$ and $S_4$; while Fig. (6) is for services $S_2$ and $S_3$. The service components require four end-to-end resources: $ciS$ requires resource $hS$, which is a local resource of the server; $ciP$ requires resources $hP$ and $ISP$, which represents the local resource of the proxy ($hS$ and $hP$ assumed to be of the same type), and the network link between the server and the proxy, respectively. $ciC$ requires resource $IPC$, which is the network link between the proxy and the client. We also assume that the end-to-end QoS levels are ranked as $Q_{P} > Q_{Q} > Q_{r}$ in Fig. (5) and $Q_{1} > Q_{m} > Q_{n}$ (Fig. (6)). In the same order, we denote the QoS levels as level 3, level 2, and level 1, respectively.

In our simulation, service requests from different clients generate service sessions. More specifically, a service session is generated by a client from a randomly selected domain among $D_1$ and $D_8$. The type of service is selected among the four services except $S_i/2$ ($i$ is the index of the domain where the
client is from). The service sessions are highly heterogeneous in their resource requirements and duration. For resource requirement heterogeneity, Fig. (5) shows the “base” resource requirement for normal service sessions. However, there are also “fat” service sessions whose resource requirement is N times the values, Fig. (5). N is either 2 or 10; and the ratio between normal service sessions and “fat” service sessions is 1 : 2. The ratio between “long” service sessions and “short” service sessions is 1 : 2. The service sessions are generated according to a Poisson process. We performed a lot of runs of the simulation, each run with a different average generation rates – from 60 sessions per 60 Time Units (TUs) to 240 sessions per 60 TUs. Each run takes a total of 10800 time units. The initial total amount of each resource is randomly set between 1000 and 4000 units. During each run, we also dynamically change the probability that each service is requested. Therefore, the overall demand for each individual resource changes over time. We created these conditions to test our system’s adaptively in dynamically identifying bottleneck resource(s) and select different end-to-end reservation plans.

5.2 Simulation results

5.2.1 Reservation success rate
This metric scales the efficiency of the proposed system regarding to the resource reservation. The curve in Fig. (7) shows the overall success reservation rate vise the average generation of services sessions per time unit. It is observed that the rate of success reservations in our system increases the success reservations in the old system. This increasing is due to the efficiency of the detector in fault detection at any resource before it is used. In addition, the efficiency of the connector is in finding and handling the alternative solution. Also, we can notice the decreasing of the success reservation rate when the number of service sessions is increased. This decreasing is justified by the huge number of connectors that caused some sort of links congestion (this congestion will be handled in a future work).

5.2.2 End-to-end QoS
This metric scales the efficiency of our system as regarding the complete reservation of the resources that provide required quality of the compound service. The curve in Fig. (8) shows the relation between the average end-to-end QoS level vs the average generation service sessions. It is notable that our system’s curve is approximately 3 while the old system curve is approximately 2.3. This indicates to the old system failure in dealing with the compound service. The value 3, determined for our system, indicates that the level of our system reaches at the compound service extraction, see Fig. (6). The decrease, noticed in the old system curve, is due to its weakness to complete the reservation processes for the input services that construct the compound one.

5.2.3 Packet loss
This metric demonstrates the probability of packet loss that occurred in the proposed system and the old system. The plot found in Fig. (9) demonstrates the packet loss probability vise the network load. Its obvious that the number of packet loss probability in our system and old system is diverged as the network load increasing. This divergence is justified by the following: the increasing in the network load which means the increasing in the network hosts and required services with different qualities. When the number of services and resources increases, the old
system efficiency decreases hence; the number of packet loss increases. Unlike the old system, our system uses the detector, the connector, and the analyzer, to handle a fired failure that occurred in the old system and promotes its efficiency. Hence; the probability of packet loss is approximately fixed especially after the network load equals to 0.4 (From 0.4 to 1 network load, the system has a huge number of services).

This metric shows the efficiency of our system additive components (the connector, the analyzer, and the detector). The efficiency of the connector is scaled by the number of successful connections in relation to the number of stations that can not provide their QoS. The efficiency of the analyzer is scaled by the number of successful QoS requests extraction in relation to the number of its connections with the connector. The efficiency of the detector is scaled by the number of failed points detection in relation to the number of failed points in the new system during the simulation time. For accuracy, all the components efficiency are scaled under different network loads, see Fig. (11).

5.2.4 Delay jitter
This metric is introduced to make sure that the additive components didn’t affect the multimedia packets delay jitter. The delay jitter regarding to the multimedia streams is a very important QoS parameter. The plot in the Fig. (10) describes the relation between the delay jitter and the first 500 packets sent by the new system. In the new system’s curve, it is obvious that the delay jitter is less than the old system’s curve in the most simulation time. So, the additive components operate without affecting the delay jitter of the multimedia packets.

5.2.5 System efficiency

Figure (9): Packet Loss.

Figure (10): Delay Jitter.

Figure (11): Efficiency of each system component.

Figure (12): Multimedia streams loss probability
5.2.7 End-To-End delay
Fig. (13) displays the end-to-end delay that may result from our computations at the buffer and during test the path if it can provide a required services or not. Its clear that our system computations didn’t affect the delay time. This is because the computations are done during multimedia transmission even a path failure is detected. Also, our system plot is better than the old one as the old one uses the rerouting technique when find a failure at any path station. The rerouting operations loads the old system with more computations that will increase the time delay.

![Figure 13: End-To-End delay.](image)

5.3 DiffServ and MPLS Vs Our System
The target of this simulation part is to compare between our system and DiffServ & MPLS package as regards the following parameters:
1- Delay time.
2- Bandwidth utilization.
3- Link failure case (DiffServ)
4- Rerouting case (MPLS).

![Figure 14: Delay of our system and DiffServ.](image)

![Figure 15: Bandwidth utilization of our system and DiffServ.](image)

![Figure 16: Our system and DiffServ at link failure.](image)

![Figure 17: Our system and MPLS at rerouting.](image)

Fig. (14) shows that our system didn’t cause an additional delay when compared with DiffServ model. This is because our system computations are done in parallel with multimedia streams transmission. Fig. (15) demonstrates the bandwidth utilization parameter that is optimized and stable for our system than DiffServ. This is because our system detects an error in a path station and find its alternative before the multimedia streams visit this station. In addition, our system messages are simple and didn’t consume high bandwidth as demonstrated in the system plot.
Fig. (16) displays the results of a simulation run with DiffServ enabled and our system but with link failure in the link between two nodes. This shows that even with DiffServ enabled, traffics are negatively impacted. The IP layer rerouting will act as a weak solution here this is because the rerouting time or duration for the rerouting to take place depends upon the IP routing protocol being followed in the network. About our system, it clear that with huge traffic generated by TCP and UDP, our system keeps the bandwidth usage stable. This is because it didn’t use the full rerouting technique as a solution, but it uses a partial rerouting operation that may substitute only a failed station with a new one. Fig. (17) shows the case of MPLS base with rerouting using the pre-established routing table.

Summarizing, no more improvements appeared in the average success reservations and the delay jitter parameters under lighter network load. For the packet loss and end to end QoS level parameters, our system improves the efficiency of the old system at the light and heavy network load. In addition, our system additive components work in high efficiency without bad effects on the old system approximately. The main overhead behind using MPLS is the signaling aspect. Depending upon the signaling protocol used, this could be an area of concern. RSVP, with its soft-state signaling causes high overhead. The MPLS signaling processing overhead and memory storage overhead at the LSRs (Label Switched Path) for maintaining per-LSP state information needs to be considered before deciding on MPLS and DiffServ integration.

6 Conclusion
We demonstrated in this paper, a brief analysis for the RSVP, DiffServ, and MPLS. Also, the QoS problems that may be occurred during the multimedia transmission have demonstrated. A new system to solve the QoS problem is introduced. The proposed system adds new three additive components, called connector, analyzer, and detector, over the old RSVP system to accomplish its target. A simulated environment is constructed and implemented to study the proposed system performance. A network simulator called NS2 is used in the environment simulation. Finally, detailed comments are demonstrated to clarify the extracted simulation results.

7 Future Work
We plan to solve the problems resulted by the system scalability (like a congestion problem, components’ synchronization, overhead communication). Hence; we can transform the proposed system to a new application layer protocol used for solving the multimedia QoS problems.

References:


Appendix A

A.1 Assumptions:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW</td>
<td>Detector visited station address.</td>
</tr>
<tr>
<td>SC</td>
<td>Connector visited station address.</td>
</tr>
<tr>
<td>TR</td>
<td>Time spent to reach any station.</td>
</tr>
<tr>
<td>TC</td>
<td>Connector visiting time.</td>
</tr>
<tr>
<td>L, J</td>
<td>Counters.</td>
</tr>
<tr>
<td>H, K</td>
<td>Two used variables.</td>
</tr>
<tr>
<td>PFS</td>
<td>Failed station position.</td>
</tr>
<tr>
<td>N</td>
<td>Number of stations in the old path.</td>
</tr>
<tr>
<td>M</td>
<td>Number of stations in the new path.</td>
</tr>
<tr>
<td>Old[]</td>
<td>Array used to keep the old path stations addresses.</td>
</tr>
<tr>
<td>New[]</td>
<td>Array used to keep the new path stations addresses.</td>
</tr>
<tr>
<td>Same[]</td>
<td>Array used to keep the similar stations found in the two paths.</td>
</tr>
<tr>
<td>Diff1[]</td>
<td>Array used to keep the different stations found in the old path.</td>
</tr>
<tr>
<td>Diff2[]</td>
<td>Array used to keep the different stations found in the new path.</td>
</tr>
</tbody>
</table>

Table A.1: Used symbols

A.2 Connector Algorithm

1- While the number of multimedia packets < > Null
  2-1 Begin
  2-2 The multimedia starts the transmission operation
  2-3 The connector agent is fired with the starting of the transmission operation.
  2-4 For I = 1 To N.
    2-4-1 Begin
    2-4-2 The connector agent tests the stored detector flag value.
    2-4-3 If the flag value is changed to one.
    2-3-3-1 Go to the step number 3
    2-4-4 Else

A.3 Analyzer Algorithm

1- If the stored connector flag is changed to one
  2-1 The analyzer receives an old and a new paths from the connector.
  2-2 The analyzer compares between the two paths and separates the similar stations and the different ones.
  2-3 The analyzer keeps the similar stations in a table (called same) and keeps the different stations in another two tables (called Diff1 and Diff2).
  2-4 The analyzer constructs a mapping in relation to the QoS in the tables of different stations, see step 2.
  2-5 The analyzer cooperates with the RSVP to extract the QoS request of a new path.
2-6 The analyzer capsulizes the results in a message and sends it to the connector.

2- The analyzer handling and mapping operations

2-1 For \( I = 1 \) to old[N].
   2-2-1 Begin
   2-2-2 If the old[I] = New[I]
      2-2-2-1 Begin.
      2-2-2-2 old[I] = Same[K]
      2-2-2-3 K = K + 1
      2-2-2-4 End IF.
   2-2-3 Else
      2-2-3-1 Begin.
      2-2-3-2 old[I] = Diff[H].
      2-2-3-3 old[I] = Diff[H].
      2-2-3-4 H = H + 1
      2-2-3-5 End Else.
   2-2-4 If H = K
      2-2-4-1no changing in the old QoS request.
   2-2-5 For \( J = 1 \) to H
      2-2-5-1 Begin
      2-2-5-2 Diff2 [J] = Construct a QoS request.
      2-2-5-3 End J For Loop.
   2-2-6 End I For Loop

3- End of the analyzer Algorithm.

A.4 Detector Algorithm

1- While the number of multimedia packets < > Null
   1-1 Begin
   1-2 If the QoS test value = 1
      1-2-1 Begin
      1-2-2 The detector multicasts an alarm message including the connector ID.
      1-2-3 The detector changes the test value to 0.
      1-2-4 The detector tests another succeed stations.
      1-2-5 End IF.
   1-3 End of the While Loop
   1-4 QoS test value = 0.

2- End of the detector algorithm.

A.5 Data stored by the container, the analyzer, and the detector.

A.5.1 Connector stored data
- Connector ID
- Address of each path station.
- Time of each visiting station.
- Analyzer ID.
- Analyzer Address.
- Stream ID.
- Detector flag value (default value =0)
- RSVP connections

A.5.2 Analyzer stored data
- Analyzer ID.
- Connector ID
- Connector address
- RSVP connections
- Similar table.
- Different tables
- The connector flag value (default value =0)

A.5.3 Detector stored data
- Detector ID.
- Connector ID.
- Connector address.
- QoS required from each path station.
- Path structure.
- The connector flag value (default value =0)
- QoS test value (default value =0)

A.6 Special comments related to our system implementation.

1- The old path is calculated starting from the failed station to the destination station.
2- The connector deletes the station address as it moves to another station.
3- The connector receives the QoS of the failed station(s) from the detector message.
4- The communication disjoint between the connector and other system components is occurred using the same RSVP messages.
5- The analyzer uses any multicast protocols like IGMP to join a multicast group.
6- The analyzer uses its multicast address for sending its messages to the connector using the connector ID.
7- The detector sends an alarm message to the connector like the analyzer did in step 6.
8- The connector, the analyzer, and the detector are software packages and can be installed as an arbitrary drives.
9- The connector moves to the middle station in the path to manage the system. The address of the middle station is sent with an acknowledge message in the first reservation process.
10- The algorithm of the connector, the analyzer, and the detector can be applied not only on the failed QoS case, but also on the new path creation case. The router, at which the path is changed, is considered as the station that can not provide the required QoS in the failed QoS case.

11- Before the routing protocol (at any router) makes a decision to change the multimedia flow path, it should inform the connector to be ready and run its functions. This will provide the connector with an enough chance to solve the QoS problems and prevent the occurrence of other problems like a congestion problem.