QoS Based Routing Algorithm with Crank-Back Ability

Somphone Kanthavong*, Prakit Tangtisanon* and Mayuree Lertwatechakul*

* Department of Information Engineering, Faculty of Engineering,King Mongkut's Institute of Technology, Bangkok, Thailand
(Tel : +66-2-737-3000; E-mail: klmayure@kmitl.ac.th)

Abstract: This paper proposes an improvement of a QoS based routing protocol. The hierarchical source routing algorithm[4] was improved by including of the Crank-Back algorithm[2]. The Crank-Back capability is the ability to solve the problem of selecting the wrong route because of using unsynchronized routing information in the ingress router. In order to reduce time of reversing the call-setup process back to the ingress router, spare-route information is included with a call-setup packet. Spare-route information could be used by every router to find the next appropriated link itself when insufficient resource of the selected link was found. The proposed algorithm improves the performance of the source routing call-setup process significantly.

Keywords: QoS based routing, crank-back ability, MPLS, source-routing

1. INTRODUCTION

Nowadays, many new packet switching network technologies were emerged to support individual requirement of any users. Mechanisms those were used to provide QoS services are including controlling and management of routing, path-setup, forwarding, marking and policing processes. Among these mechanisms, the most interesting one is how to know the appropriate routes and how to set-up the route with less control traffics and route set-up time.

Previous works on call-setup process could be divided into 2 groups: the distributed routing and the centralized routing. The distributed routing requires every router or switch to be has the ability to find an appropriated route. Such kind of the algorithm often uses call-setup flooding process to increase success probability of a call-setup process. Additional to increasing success probability, the algorithm also helps reduce time of call-setup retrying process in case of required resources are not available. On the other hand, flooding of call-request packet usually injects a lot of control packets into the network and this may suffer routers to keep so many call-setup requests' status and have too much unnecessary work to do.

By using the centralized routing algorithm, a central node determines the most appropriate route for a call request and then puts source-routing information into a call-setup packet. The packet will be sent through the network along the specified route in order to reserve network's resources it wants. If there is a node along the route could not provide the required resources, the call-request would be rejected.

Call-request rejection may suffer users when they need to access through the network. The network may absorb the effect by providing the retrying process or the crank-back ability. The crank-back ability is the ability to fix the failure of a call-setup process in case that the residual resource along the selected route could not be serviced. A router with crank-back ability can reverse the call-setup process to the preceding node when it finds that the request could not be served. This can be done only in the distributed routing network but it does not work in the centralized routing network. The reason here is that the routers or switches within the centralized routing network do not contain any routing information and have not any capability to compute them. So they have no other choices to be chosen, the call-setup process must be reverse back to the first step (in the central node) again.

As to reduce the control traffic and control process overhead of call-request flooding by distributed routing and to reduce time used in retrying process, this paper proposes a QoS based routing protocol with crank-back ability. The protocol is adapted from [4] which is a hierarchical source routing scheme. We added more spare-route information into the call-setup packet and let the router to be has an ability to choose a new route from the provided route information when the selected route is not available. The proposed scheme could be used to reduce controlling traffic and processing load of the network while still consumes less time than the ordinary source-routing processes.

2. QOS BASED ROUTING

Iata and Fujita [4] have proposed a call-setup process to be used in MPLS network. Routers of the network are classified into 2 types: Label Switching Router (LSR) and Area Border Router (ABR). ABR is a LSR that has its own routing information and can update them according to the recently status of links in the subnetwork that have received. Call-setup process of [4] was broken into each subnet processes. The process is implemented in hierarchical manner that very helpful in decreasing a number of routing information of which a router has to store and process. An ingress ABR of a call-setup packet will determine the most appropriate route for a call-request. The routing could be considered into 2 steps, in the first step, a set of routes are determined in the subnet level while in the second step, routes are determined in the router level. After the first step, the ABR will know which ABRs are eligible to access the target subnet. Once a target subnet is selected the second step will begin as to find the most appropriated route within the current subnet to reach the egress ABRs those connect the current subnet and the target subnet.

A set of routes may be pre-computed for the second step by using the QOSPF algorithm.[3],[8] Once a route has been selected, the ABR will put the route information into the call-setup packet. The packet will be sent to the next LSR on the specified route. The LSR only update their translation table and resources table corresponding to the allocated resources if they are available. The call-setup process will be continue the same way along the selected route until the call-setup packet has reached to the egress ABR. The processes like this will be happened again in the target subnet. On the other hand, if a LSR has found that the required resources are not match the call-setup process would be reversed back to the ingress ABR of the subnet. This may be happened

because of the routing information within the ingress ABR may be not synchronized with the actual resources information available at that time. Once the ABR has received a reject message, the ABR will try to find the other appropriate route from the pre-computed routes or try to compute for a new route using up-to-date routing information that the unavailable link has been truncated.

3. THE PROPOSED SCHEME

Even though mechanism [4],[7] produces less control packets and less processing time of a call request than [1] and [2] which are distributed mechanisms, but the routing information within ABRs at the edges of a subnet may not be synchronized with the actual resources information of the system. The unsynchronized problem would effect to an ABR to make a wrong decision in selecting an appropriated path for a call-setup request. The mistake will take effect to a LSR to reverse the call-setup process back directly to the ABR. The reversing process could be happened many times and this produces more control overhead and also wastes the user's time.

This paper, we propose a routing scheme in order to reduce the mentioned problem that may be occurred in [4]. By using the proposed scheme, ABR will add one or more spare-routes into a call-setup packet along with the selected route. The spares-path will be sorted inside the packet by cost or how optimum they are. The scheme needs LSRs to have a little more ability to read multiple routes' information in a call-setup packet and to modify them if necessary. LSRs must know where is the location of the next LSR address specified in the selected route and how to retrieve a new route from route information within a call-setup packet when the selected route could not be served.

In case that the available resources are enough, a call-setup process would follow the normal operation by forwarding a call-setup packet hop-by-hop through the specified route. A call-setup process will be success within a subnet when its call-setup packet has reached to the decided egress ABR. That means all LSRs along the route have reserved their resources according to the requested amount. On the contrary, if the requested resources of the specified link were not found, the LSR has to select a new route from the provided set of spare-routes. When the LSR found that one or more spare-routes pass through itself and the available resources are matched, the LSR would select the first route, reserve the resources and then forward the call-setup packet right through the new selected path. Before the call-setup packet will be transmitted to the new subroute, the routing information within the packet has to be modified by remove a set of routes that constructed with the unavailable link.

But if the LSR cannot find any available route passes through itself, the LSR would send back a reject message of the call-setup packet to the preceding LSR. In order to response of receiving a reject message, a LSR will consider all the routing information within the reject message as to find the next route that fit to the required QoS and then take action same as the processes described above. Once the call-setup packet is received at the egress ABR, it means the subtask of a call-setup within a subnet has been succeeded. On the other hand, if a reject message of a call-setup has reached to the ingress ABR, it would cause the ABR to compute for a set of new routes again. The new routes would be recalculated from the routing information which exists at that time and to be truncated by the set of known unavailable links retrieved from the reject message.

By using the algorithm, we could reduce call-setup time in the call-setup process which uses the source routing algorithm during a failure has occurred. Because the existing algorithms need a call reject message to be sent back to the ingress ABR for select the new route, while our algorithm a LSR may select the new route itself and then the call-setup process would be recovered. This causes the call-setup recover process would be happened much more quickly than the previous one. And that is the reason why our methodology could improve the average call-setup latency significantly as to be shown in the next sections.

4. THE ANALYSIS

This section, we would like to show how to analyze the advantage of the QoS based routing algorithm with crank-back ability compare to the preceding proposed algorithm via mathematical model. We have formed many equations to represent behavior and effective factors of both schemes. And the equations could be use to analyze the advantage and disadvantage of the schemes in the clear picture.

4.1 performance analysis of the existing scheme

Definition

Let N^{k_r} be a number of routers along a selected route and we consider the sequence number of a router along a subroute as 1, 2, 3,.., N^{k_r} according to their position placed in the route.

As to calculate the minimum delay that caused by a call-setup process, excluding for queuing delay, we could use the following equation.

$$D_{setup}\left(N_{k_r}\right)= \sum_{i=1}^{N_k-1} \tau_{ij}\Big|_{j=i+1} + \sum_{i=1}^{N_k-1}\delta_i + \sum_{i=2}^{N_k}\rho_i + \eta_{k_r} \qquad (1)$$

where

τ_{ij} is the propagation delay of a link *i,j* that connects node *i* to node *j*

δ_i is the transmission delay of a call-setup packet through link *i,j*

ρ_i is processing delay happened to a call-setup packet within LSR node *i* to consider the required resources of the specified link of the requested path.

η_{k_r} is time used by an ingress ABR to find an appropriated route and construct the call-setup packet by using the source-routing information of the selected path.

We assume that τ_{ij} , δ_i and ρ_i are constant for every node *i* and *j*. And given that $\overline{N}$ is an average number of routers per route.

The minimum time consumed by a call-setup process may be given by:

$$\overline{D}_{setup} = \overline{N}.\left(\overline{\tau}+\overline{\delta}+\overline{\rho}+\overline{\eta}\right) \qquad (2)$$

Where bar notation "$\overline{\ }$" represents the average value of a variable within a considered subnet.

If the resources of a specified link of the node M_{k_r} where $2 \le M_{k_r} \le N_{k_r}$ could be not satisfied the call request. Node M_{k_r} would send a reject message of the call-setup packet back to the ingress ABR. We could calculate the minimum time used by the process of sending a call-setup packet hop-by-hop to

node M_{k_r} along the selected route and node M_{k_r} has found that the required resources are not available and then node M_{k_r} would send the reject message back to the ingress ABR by using Eq.(3)

$$D_{CB}(M_{kr}) = \sum_{i=1}^{M_{k_r}-1}\tau_{ij}\Big|_{j=i+1} + \sum_{i=1}^{M_{k_r}-1}\delta_i + \sum_{i=2}^{M_{k_r}}\rho_i + \eta_{k_r} + \sum_{j=2}^{M_{k_r}}\tau_{ij}\Big|_{j=i+1} + \sum_{j=2}^{M_{k_r}}\delta_i + \sum_{j=2}^{M_{k_r}}\psi_i \quad (3)$$

where

ψ_j is the processing time used to process a reject message.

And we could represent the probability of a call request that may be accepted from node *i* through node *j-1* and would be rejected at node *j* of the route *r* as the following equation.

$$B_{k_r}(j) = \beta_{k_r}(j)\prod_{i=1}^{j-1}(1-\beta_{k_r}(i))\Big|_{j\le N_{k_r}} \quad (4)$$

where

$\beta_{k_r}(j)$ is the call blocking probability of node *j* for a call request *k*.

By using the above definitions then we could get the expectation of the sequence number of the blocking node along a route from:

$$E(M_{k_r}) = \sum_{i=1}^{N_{k_r}} i\cdot B_{k_r}(i) + \sum_{i=N_{k_r}+1}^{\infty} i\cdot B_{k_r}(i) \quad (5)$$

In order to achieve the approximation, we utilize the expectation equation. Normally, the number of actual nodes along a route is finite so we use the average value $\overline{\beta}_{k_r}(i)$ instead of $\beta_{k_r}(i)$ to calculate $B_{k_r}(i)$ which appears in the second term of Eq.(5) where $i > N_{k_r}$. This method is very helpful to decrease the error of approximation.

By using $E(M_{k_r})$ we could get the approximation of the delay caused by call-setup process of a call request *k* through route *r* from node no.1(an ingress ABR) to node no. M_{k_r} and the rejecting process from node no. M_{k_r} back to node no.1 again from:

$$\hat{D}_{CB}(M_{k_r}) = \frac{E(M_{k_r})\cdot D_{CB}(N_{k_r})}{N_{k_r}} \quad (6)$$

And the average of $\hat{D}_{CB}$ a call request *k* from Eq.(6) over every possible route could be defined as:

$$\overline{D}_{CB^k} = \frac{\sum_{i=1}^{R}\hat{D}_{CB}(M_{k_i})}{R} \quad (7)$$

While the probability of a call request k may be block ed by at least one node along route r

$$\beta_{k_r} = 1-\prod_{i=1}^{N_{k_r}}(1-\beta_{k_r}(i)) \quad (8)$$

And if $\overline{\beta}_k$ is the average of β_{k_r} for every route in the subnet and we assume that call blocking probability for a call request *k* is almost unchanged within a very short time interval. Then probability of a call request *k* may be blocked *f* times and to be succeed at *f+1* trying could be get from:

$$\zeta_k(f) = (\overline{\beta}_k)^f(1-\overline{\beta}_k) \quad (9)$$

So we could approximate a mean of blocking times within a subnet by using:

$$E_k(f) = \sum_{f=1}^{\infty} f\cdot\zeta_k(f) \quad (10)$$

And an average call-setup latency of a call request k within a subnet from:

$$\overline{D}_{total} = \overline{D}_{setup} + E_k(f)\cdot\overline{D}_{CB} \quad (11)$$

4.2 performance analysis of the proposed scheme

As we described before that a call-setup process which uses source-routing scheme may has to retry a call-setup process at the ingress ABR in case that the blocking occurred. This could attend the call request response time to the users or applications dramatically. The scheme presented in this paper proposes that if every LSR has the ability to resume a call-setup process by choosing a new route from a set of provided spare-routes within a call-setup packet, this may be helpful in decreasing signaling and resume the call-setup process time at the ingress ABR. As to show the performance of the proposed scheme we have defined a set of equations as follows:

Definition

Let we define m_{k_r} as a sequence number of a router node within a subroute *r* that has not enough resources to service a call request *k*.

A subroute means a part of a route which could be used to construct a QoS path for a call request. Within a subroute *r*, we consider the sequence number of a router along a subroute as 1, 2, 3, .., *n* same as the definition of sequence number of routers along a route that has defined in section 4.1. where node 1 of the subroute $\tilde{r}$ is the node y_{k_r} of the preceding subroute *r* which is the merging point to the subroute $\tilde{r}$. The system could resume the call-setup process without help of the ingress ABR only if there is at least one node *i,* with serviceable probability $P_{k_{\tilde{r}}}(i)$ within a joined subroute $\tilde{r}$, exists along route *r* from node 1 to node m_{k_r}.

where $P_{k_{\tilde{r}}}(i)$ is defined as:

$$P_{k_{\tilde{r}}}(i) = \max\{(1-\beta_{k_j}(i))\}\Big|_{j\ne r} \quad (12)$$

And if we break a call-setup latency into a progressing part and a reversing part. We could represent a total time used by a call-setup process which success at *f+1* times as:

$$D'_{total}(f) = \sum_{r=1}^{f+1} D_{progress} + \sum_{r=1}^{f} D_{back} \quad (13)$$

Where

$$D_{back} = \sum_{j=y_{k_r}+1}^{m_{k_r}} \tau_{ij}\Big|_{i=j-1} + \sum_{j=y_{k_r}+1}^{m_{k_r}} \delta_j + \sum_{j=y_{k_r}+1}^{m_{k_r}} \psi'_j \quad (14)$$

$$D_{progres} = \sum_{i=1}^{m_{k_r}} \tau_{ij}\Big|_{j=i+1} + \sum_{i=1}^{m_{k_r}} \delta_i + \sum_{i=2}^{m_{k_r}} \rho_i + \eta_{k_r} \quad (15)$$

Call-setup latency D'_{Total} is one of the most effective parameter to estimate the performance of a call-setup process. We could approximate D'_{Total} by using m_{k_r} as a number of progressing steps and using S_{k_r} as a number of reversing steps. Where the expectation of S_{k_r} could be calculated from:

$$E(S_{k_r}) = \sum_{i=1}^{m_{k_r}} G_{k_{\tilde{r}}}(i, m_{k_r}) \cdot (m_{k_r} - i + 1) \quad (16)$$

where

$G_{k_r}(y_{k_r}, m_{k_r})$ is the probability of a call request k is may be blocked at node m_{k_r} and there is a node located at position y_{k_r} along the same subroute r that could service the call request onto the other subroute $\tilde{r}$ and be defined as:

$$G_{k_{\tilde{r}}}(y_{k_r}, m_{k_r}) = P_{k_{\tilde{r}}}(y_{k_r}) \cdot \prod_{i=y_{k_r}+1}^{m_{k_r}} (1 - P_{k_{\tilde{r}}}(i)) \quad (17)$$

Where

$$\hat{y}_{k_r} = m_{k_r} - E(S_{k_r}) \quad (18)$$

$$n_{k_{r+1}} = N_{k_{r+1}} - \sum_{i=1}^{r} y_{k_i} - r + 1 \quad (19)$$

We could use Eqs. (16)~(18) to calculate $m_{k_{r+1}}$, $S_{k_{r+1}}$, $y_{k_{r+1}}$ and $n_{k_{r+1}}$ again and again in the next resumption process. Normally we have found that $n_{k_{r+1}} \le n_{k_r} \le n_{k_{r-1}} \le .. \le N_{k_1}$ except for the case that $N_{k_{r+1}} >> N_{k_r} >> N_{k_{r-1}}$ and $P_{k_{\tilde{r}}}(i)$ is very small. And this shows that smaller n_{k_r} usually give smaller β_{k_r} because of the reason that if every node in a subnet has the same blocking probability $\beta_{k_r}(i)$ for a call request k, a route constructed with the smallest number of router will be the route with the smallest blocking probability β_{k_r}. We could prove this by using Eq.(8)

5. SIMULATION AND RESULTS

Performance of any call-setup process could be measured in terms of call-setup latency, success probability, controlling traffic quantity, network utilization, the complexity of the algorithm, scalability, optimality of the route, and so on. The ability to select the most appropriated route that is matched to the user's requirement depends on routing algorithm and also how up-to-date the resources information was. In the study, we would like to focus only on the factor of call-setup latency through the numerical analysis.

By applying the equation described in the previous section onto the network topology as shown in Fig.1 whereas the blocking probability of an interesting call-request k at node i along route r is defined as the examples in the Table.1. For every calculation, we have defined a destination node $\underline{D}$ which has an average distance from a source node S equal to $\underline{N}$ nodes. We have computed the total time spent for a call-setup from node S(node1) to each node D under the defined circumstance.

The new destination D has been changed in order to increase the distance from the source node S.

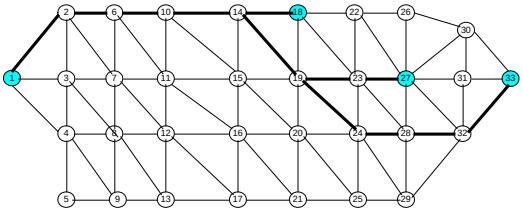

Fig. 1 The simulation Topology

Table 1 The examples of $\beta_{k_r}(i)$ of node i within Fig. 1

Node ID	Blocking probability at node i			Node ID	Blocking probability at node i		
	Route1	Route2	Route3		Route1	Route2	Route3
1	0.17	0.28	0.28	19	0.36	0.28	0.36
2	0.17	0.28	0.36	23	0.36	0.36	0.36
6	0.28	0.28	0.36	24	0.17	0.28	0.36
10	0.17	0.28	0.36	27	0.28	0.36	0.36
14	0.17	0.28	0.28	28	0.17	0.28	0.28
18	0.28	0.36	0.28	32	0.17	0.28	0.17

The result of simulations has been shown in Fig. 2 and Fig. 3. The results in Fig. 2 came from the scenario described in the preceding paragraph. It shows that call-setup latency has been spent in a subnet will be increased corresponding to the distance of a route. Without using spare-route information in source routing call-setup process, a call request must spend much longer time than using spare-route information about 58 percent.

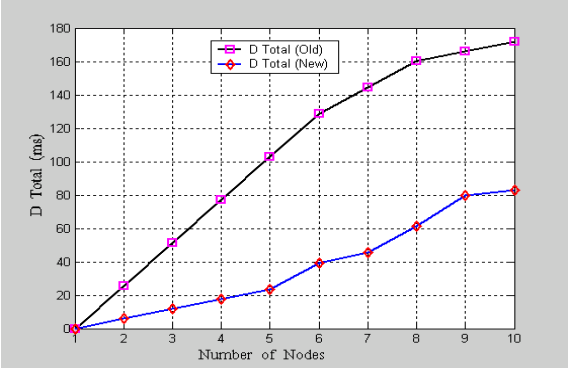

Fig. 2 Comparison between D_total of the proposed scheme D_total (New) and the existing scheme D_total(Old)

Fig. 3 shows the results of the other scenario. The scenario here was setup by defining that every router has the same blocking probability for a call request k onto every route. But the blocking probability $\beta_{k_r}(i)$ of every node i would be changed in order to show how the blocking probability may take effect to the call-setup latency. And we were also interested in how the both scheme response to the increasing

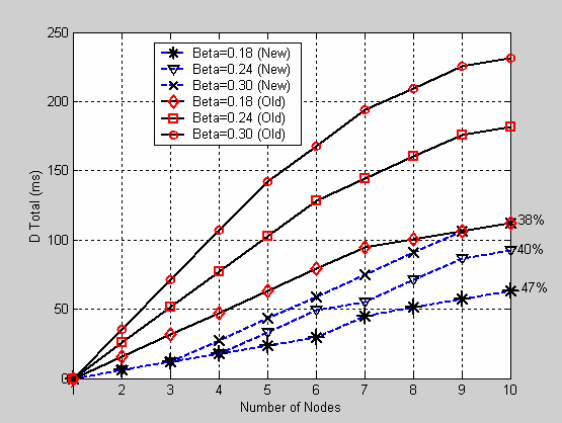

Fig. 3 Comparison performance of the proposed scheme and the existing scheme when blocking probability was changing

of blocking probability. The $\beta_{k_r}(i)$ in the scenarios has been changed from 0.18 to 0.24 and 0.30 consequently.

The results show that the higher blocking probability will increase call-setup latency in the both schemes. By using an ordinary source routing call-setup process, the call-setup latency would be increased dramatically even if the blocking probability was slightly increasing. Comparing to the proposed scheme, using spare-route information could help resume call-setup process much more quickly than the ordinary one. And the graph may show us more on the effect of increasing blocking probability that has been occurred to the proposed scheme. Call-setup latency of the new scheme has been slightly changed compare to the existing scheme when the blocking probability has been switched. At $\beta_{k_r}(i)$ equal to 0.18 the average call-setup time spent by the proposed scheme was 47 percent of time used by the preceding scheme. But when $\beta_{k_r}(i)$ has been switched to 0.24 and then 0.30 whereas ours take only 40% and 38% of time spent by the existing scheme. That means the source-routing call-setup process with the modified crank-back ability will show more on its performance against the blocking probability that was increasing.

6. CONCLUSIONS

Call-setup process that utilizes the source-routing algorithm may reduce LSR's task in collecting and computing for a set of possible routes. But the scheme is ineffective in a big and busy subnet which a route may take long distance and the blocking probability will be high. Using unsynchronized routing information may take much more bad effect to the success probability and time spent by a call-setup process. Our proposed scheme needs to add only a few capability to the LSRs to determine the routing information within a call-setup packet and select the new route when the specified route could not be supported. The proposed scheme reduces wasteful time used for a call-setup recovering process by the ordinary source-routing call-setup scheme significantly. Especially for the case that as the blocking probability of a subnet is getting higher, the performance of the QoS based routing algorithm with crank-back ability will be more clearly shown.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the AUN/SEED-Net Program contributed by JICA.

REFERENCES

[1] A. Fei and M. Gerla, " Smart Forwarding Technique for Routing with Multiple QoS Constraints", *Proc. Of the Global Telecommunications Conference 2000*, (GLOBECOM '00), Vol.1, pp.599–604, 27 Nov.-1 Dec. 2000.

[2] I. Cidon, R. Rom and Y. Shavitt , "Multi-path routing combined with resource reservation", *Proc. Of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies*, (INFOCOM '97), Vol.1, pp.92–100, 7-11April 1997.

[3] P. Kaewyongphang, C. Pornavalai, R. Varakulsiripunth, G. Chakraborty and N.Shiratori, "On Precomputing of QoS Paths", *Proc. Of the IEEE International Workshop on Intelligent Signal Processing and Communication System*. (ISPACS'99), pp.17-20, December 1999.

[4] A. Iata and N. Fujita, "A Hierarchical Multilayer QoS Routing System with Dynamic SLA Management", *IEEE Journal on Selected Areas in Communication*, Vol.18, No.12, pp.2603-2616, December 2000.

[5] T. Sekiguchi,Y. Koyama, K. Fujikawa,Y. Okabe, K.Iwama," Hierarchical path QoS on a QoS-based multicast protocol SRSVP," *High Performance Switching and Routing,2002. Merging Optical and IP Technologies. Workshop*, pp.334–339,26-29 May 2000

[6] S. Chen and K. Nahrstedt,"Distributed QoS routing with imprecise state information", *Computer Communications and Networks, 1998. Proceedings.7th International Conference* , pp.614–621, 12-15 Oct. 1998

[7] C. Hsien-Kang, C. Ben-Jye and H. Ren-Hung, "Reducing crankback overhead in hierarchical routing in ATM networks," *Communications, 2001. ICC 2001. IEEE International Conference*, pp.125–3129, vol.10, June 2001

[8] A. Orda and A. Sprintson, " QoS routing: the pre-computation perspective," *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings.* IEEE , Vol.1 , pp.128 –136, 26-30 March 2000