Chapter 15
An On-Demand QoS Routing Algorithm in Multiservice Scenarios

Yanjing Li and Li Li

Abstract The trend that Internet applications are becoming more various puts forward higher request on Quality of Service (QoS) of coexisting multiservices in modern networks. An on-demand QoS path selection algorithm (marked as BW-cost algorithm) satisfying constrained conditions of both bandwidth and cost is proposed innovatively in this paper, and its usage procedures in OSPF is claimed accordingly. The core idea of the algorithm is choosing the best route satisfying certain bandwidth requirement with smallest cost in order to achieve the goal of splitting streams and balancing load. Simulation results shows the BW-cost algorithm can successfully guarantee multiservices' QoS demands as well as has much better performance in some targets such as throughput and network utility, comparing to traditional Dijkstra algorithm. It also exceeds en existing on-demand QoS algorithm.

Keywords OSPF · Qos routing · Dijkstra algorithm · Multiservice scenario

15.1 Introduction

A multiservice scenario has become the mainstream study scene in today's networks as IP networks are demanded for more satisfying data services. Routing protocols which are using traditional algorithms to provide best-effort services can no longer meet the user's QoS needs any more. For instance of the Dijkstra algorithm used in OSPF protocol, which is one of the most classic shortest path

Y. Li · L. Li (✉)
Xitucheng 10, Haidian District, Beijing, China
e-mail: lili66@bupt.edu.cn

first algorithms, it always leads to the overuse of calculated paths with minimum hop counts and consequently the congestion at this best path when some high-quality-demanding data services trying to pass through. So some QoS mechanisms must be used to balance the load of the network and improve the transmission efficiency.

QoS routing has been studied for a while. In [1] a cheapest path algorithm from one source to all destinations when links have two weights (cost and delay) is presented. Some other solutions use source routing along with shortest path routing to achieve the goal [2]. Anirudha presented a load sensitive routing (LSR) algorithm based on Dijkstra's shortest path algorithm [2]. In [3], Karima et al. claimed that MPLS and traffic engineering provide indeed an adequate mean to establish constrained routes which satisfy application requirements like bandwidth. A novel two-phase load balanced shortest path routing (LB-SPR) is proposed in [4], where each phase uses the standard SPR protocol. In [5], authors described a path precomputation selection algorithm. A feature of it is that in each iteration, the intermediate results must be stored in the QoS routing table, which inevitably increases the expenses in terms of storage.

Another algorithm authors put forward is an on-demand BW-hop based QoS algorithm feasible when number of requests for QoS routes is limited [6]. They described how a standard Dijkstra algorithm can, for a given destination and bandwidth requirement, generate a minimum hop path that can accommodate the required bandwidth.

Through analysis OSPF protocol and current QoS algorithms, this paper puts forward a routing strategy capable of ensuring QoS (bandwidth requirement mainly). The core idea of the algorithm is choosing the best route satisfying certain bandwidth requirement with smallest cost. Simulation results show that the improved on-demand BW-cost routing algorithm proposed in this paper solves the QoS routing problem better than the algorithm in [6] while much exceeds the traditional Dijkstra algorithm.

The rest of this paper is organized like this: in Sect. 15.2, the BW-cost algorithm and its operation procedure in OSPF are illustrated in details; in Sect. 15.3, simulation results among different scenarios are analyzed and compared; eventually a conclusion is drawn in Sect. 15.4.

15.2 An On-Demand BW-Cost QoS Algorithm

15.2.1 QoS Routing Procedures Design

A routing starter computes the QoS route if there is a resource reservation request. If not, it remains calculating routes using normal Dijkstra algorithm. Elaborate procedure operated when QoS routing is applied is illustrated as below:

Each router sets up a bandwidth request queue, a current bandwidth request, a value of request threshold and a variable of the state of request queue. The request threshold means the maximum number of acceptable bandwidth requests supported by one router and should be identical for all routers in an OSPF area. The state of request queue is set to be REQ_FREE initially.

If there is no bandwidth request from upper applications, the original Dijkstra algorithm is applied directly. Every time the initiating router receives a bandwidth reservation request, it looks up the standard routing table to determine whether the destination address is reachable. If the destination address is unreachable, then the router denies this request. If the destination is reachable while the state of request queue is REQ_FULL at the same time, the router should deny the request as well. Otherwise resource request ID adds up by 1, which starts from 1, and the request is pushed into the bandwidth request queue. If the number of requests in the bandwidth request queue is already equal to the threshold value, the state of it is turned to REQ_FULL.

Whenever a router receives a request LSA, it firstly determines whether the state of the bandwidth request queue is empty. If empty, it directly sends out bandwidth response LSA and regards this received bandwidth request as current request. Then it checks if the state of request queue is REQ_FULL. If so, it compares the priority of the latest request to the most inferior request in the queue. If the priority of the latest request is higher, it discards the most inferior request in the queue and inserts the latest one into the queue based on the priority mechanism. Otherwise, it discards the latest request.

A router receives a bandwidth response LSA. If the router is the initiating router and the LSA is the response to current request, it puts the LSA into the database, which is used to store link states in OSPF, otherwise discards it simply. Whenever the database has contained bandwidth response LSAs from all the routers in the domain, the router applies the improved BW-cost QoS algorithm proposed in this paper, based on the information in the database. During the computation process, only links able to provide adequate bandwidth are taken into account, otherwise the link is thought unreachable. If the output of the algorithm is empty, then the router will deny the resource reservation request. If not, the router generates bandwidth allocation LSAs and then distributes them.

If a router receives a bandwidth allocation LSA, it checks itself if is part of the QoS route. If the answer is positive, it constructs a QoS route and reserves required bandwidth at related interfaces.

A router processes the bandwidth request with highest priority in the queue if it is not empty. If the router is the initiating one of the request, it sends out the bandwidth request LSA. Otherwise, it sends corresponding bandwidth response LSA. If the state of request queue is REQ_FULL, it is turned to REQ_FREE.

When a router receives a bandwidth release LSA, it checks itself if is part of the QoS route based on information in the LSA. If the answer is positive, it deletes related QoS routes and releases bandwidth at related interfaces.

If some OSPF link is broken, standard OSPF procedures are responsible for flooding all the messages through the whole domain. The initiating router examines all the requests which have already utilized QoS routes with participation of the broken link and sends corresponding bandwidth release LSAs. After that, it reapplies for required bandwidth to construct a new QoS route.

15.2.2 QoS Path Selection Algorithm

The core idea of the algorithm is to perform a minimum cost path computation on a preprocessed graph whose links without enough bandwidth for services are deleted from the topology. The pseudo code of the algorithm is illustrated in the following.

```
Inputs:
V = set of vertices, L = set of edges, s = source vertex (at which the algorithm is executed), d = destination
Struct tab_entry:
{hops = integer,
prevnode = integer 1..N,
ontree = boolean.}
Variables:
TT[1..N]: topology table, its (n) entry is a tab_entry record; S: list of candidate vertices; v: vertex under consideration; b(n,m): available bandwidth on edge (n,m); N: the number of paths being stored; cost(n,m): cost value of link (n,m).
Algorithm begins:
Function qos_djk:{
for n = 1 : N do/* Initialization */
TT[n].hops = Inf;
TT[n].prevnode = null;
TT[n].ontree = FALSE;
TT[s].hops = 0;
v = s;
while v ! == d do
begin:
TT[v].ontree = TRUE;
for all edges (v,m) in L and b(v,m) >= B{
if (!TT[m].ontree & TT[m].hops > TT[v].hops+1)
{
S = S union {m};
TT[m].hops = TT[v].hops + 1;
TT[m].prevnode = v;
}
}
if S is EMPTY
v = d; path = −1;/* This will end the algorithm */
else
v = S(1); S = S - {v};
end./*for while*/}/*end for function*/
  - - - - - - - - - - - - - - -
if (path!==−1)then store path;
id = 1;/*initialization, used to indicate which node is checked now*/
while path!==−1 do
begin:
w=path(id);/*element under consideration*/
k=path(id+1);/*next element on path*/
if (for any node m, m!==w & m!==k & b(w,m)>=B){
b(w,k)=0;/*fail the original selecte link*/
re-do Function qos_djk;
if (path!==-1)then store path;
id=1;/*reset to the first node*/
}/*end for if*/
else
id++;/*check next node on path*/
end;/*for while*/
cost=Function sum(path(1));
path_final=path(1);/*initialization*/
for x=2:N do{
if(cost>Function sum(path(x));) then
path_final=path(x);
}/*find the least costy path from all paths*/
- - - - - - - - -
Function sum:{
cst=0;/*initialization, used as a temp value for cost*/
for z=1:(length(path)-1)
cst=cost(path(z),path(z+1))+cst;
}/*The function is used to sum the costs*/
```

15.3 Simulation and Analysis

In the simulation, the network is constructed in OPNET. The topology is shown in Fig. 15.1. Four traffic flows are considered during the simulation, the ID number is also the bandwidth requirement value of a specific traffic with unit Kbps. This design also adapts to the following sections when network parameters are collected for traffic statistics.

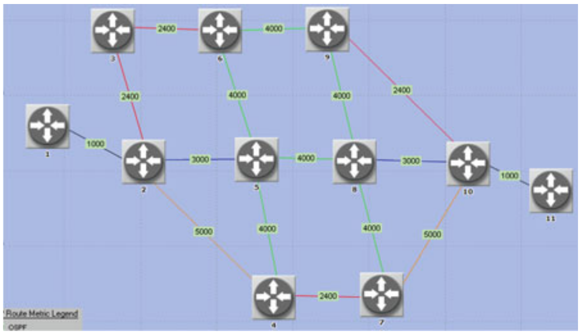


Fig. 15.1 Network topology

Three algorithms, the original Dijkstra algorithm (marked as standard Djk), the QoS algorithm proposed in [7] (marked as on-demand QoS Djk), and the algorithm proposed in this paper (marked as BW-cost QoS Djk) are simulated.

15.3.1 Simulation Results of QoS Routes

The path selection results of three algorithms are shown in Table 15.1.

From Table 15.1, we can get the conclusion that the improved BW-cost algorithm proposed perfectly balances load of multiple services from one shortest cost path to several QoS routes with second minimal cost value satisfying bandwidth demands at the same time.

15.3.2 Simulation Results of Throughput

The simulation results of throughput are shown in Fig. 15.2. From it we can see that when routers adopting either on-demand QoS algorithm, all 4 services' bandwidth requirements are guaranteed (light & dark green bars); while under the SPF algorithm circumstances, only two services with ID 850 and ID 550 are delivered successfully, huge traffic loss can be found for other two services as presented as red bars.

Table 15.1 Selected paths in different scenarios with various bandwidth requirements

	ID 1100	ID 850	ID 550	ID 350
Standard Djk	2-5-8-10	2-5-8-10	2-5-8-10	2-5-8-10
On-demand QoS Djk	2-5-8-10	2-4-7-10	2-3-6-9-10	2-4-7-10
BW-cost QoS Djk	2-5-8-10	2-3-6-9-10	2-3-6-9-10	2-4-7-10

Fig. 15.2 Throughput comparison

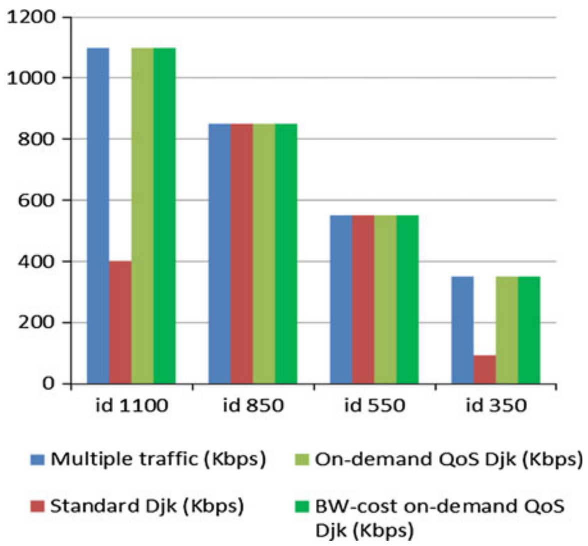
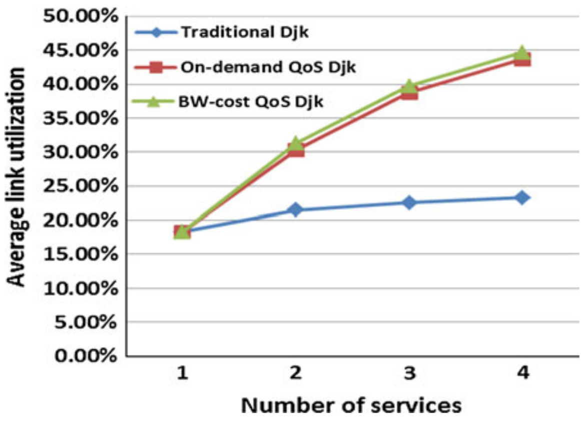


Fig. 15.3 Average link utilization



15.3.3 Simulation Results of Average Link Utilization

The computing method of average link bandwidth utilization is given as the following formula [7]:

$$U = \sum_{(i,j)\in L} \frac{u(i,j)}{|L|} \tag{15.1}$$

Where u (i,j) is ratio of occupied bandwidth to initial link bandwidth on link (i,j), L is the set of all links in the network. The results are shown in Fig. 15.3.

From Fig. 15.3, we can see that when adopting the traditional SPF algorithm, the link utilization of the whole network does not improve obviously (no higher than 25 %). When it comes to the on-demand QoS scenarios, the average utilization rises significantly as the number of services increases gradually and the new BW-cost QoS algorithm proposed in this paper has 1 % higher utilization than the existing on-demand QoS algorithm at multiple stages.

15.4 Conclusion

In this paper, elaborate illustrations are put on the proposed on-demand BW-cost QoS algorithm and its according working procedures to make QoS metrics compatible with OSPF flooding mechanism. Eventually, the simulation results show that QoS routing extension to OSPF protocol we proposed in this paper is a more effective mechanism to guarantee services' quality in multiservice scenario than the existing one.

Acknowledgments The research work is supported by the Fundamental Research Funds for the Central Universities (2012RC0120).

References

1. Goel A, Ramakrishnan KG, Katatria D, Logothetis D (2001) Efficient computation of delay-sensitive routes from one source to all destinations. Proc IEEE Infocom 12(5):683–687
2. Sahoo A (2002) An OSPF based load sensitive QoS routing algorithm using alternate paths. J IEEE 43(43):345–348
3. Maalaoui K, Belghith A et al (2005) Performance evaluation of QoS routing algorithms. J IEEE 32(3):742–749
4. Antic M, Maksic N et al (1997) Two phase load balanced routing using OSPF. IEEE J Selected Areas Commun 28(1):64–69
5. Guerin RA, Orda Williams A (1997) QoS routing mechanisms and OSPF extensions. Proceed IEEE GLOBECOM 19(9):67–73
6. Apostolopoulos G, Network working group (1999) RFC 2676. QoS routing mechanisms and OSPF extensions 24(2):37–42
7. Yanwen Hua, Songrong Qian (2006) Research on OSPF extensions supporting QoS routing mechanism. Comput Eng Des 27(3):415–417