OSPF-Based Adaptive and Flexible Security-Enhanced QoS Provisioning

Zhen Qin, Roberto Rojas-Cessa, and Nirwan Ansari

***Abstract*— Networks with quality of service (QoS) provisioning rely on QoS routing schemes to select paths between end-to-end hosts that can satisfy the requested service requirements. In this paper, we focus on networks that use a new service paradigm as a nested DiffServ model, which has been described as service vectors in combination with the explicit end-point admission control (EEAC) scheme [1]. The main objectives of service vectors are to improve the QoS granularity and service routing flexibility. These improvements enhances both network utilization and user benefits. However, QoS routing schemes need to consider service vectors to achieve such improvements. In this paper, we discuss a new set of requirements that are added into QoS routing schemes to avoid false routing and low network utilization when using service vectors. The OSPF protocol, as the widely used protocol, guarantees feasible deployability of service vectors in existing networks. Furthermore, we introduce a network architecture that integrates security into the set of QoS parameters and show how security-enabled QoS (SQoS) can also use the OSPF protocol for SQoS routing.**

***Index Terms*— Quality of Service, Service Vector, Network Security, OSPF, k Shortest Paths**

I. Introduction

IntServ and DiffServ are well-known alternatives for QoS provisioning. These two paradigms differ in the level of accuracy on service provisioning and QoS granularity for a scalable implementation, and therefore neither one can satisfy a large number of service requirements. Whether IntServ over DiffServ model can fill this void remains an open issue in which case it is necessary to map each individual flow's end-to-end QoS requirements from IntServ model to the DiffServ model. Hence, flows mapped to a single service class get similar end-to-end QoS guarantees, even in the case when the requirements of each individual flow differ. Therefore, the QoS granularity is decreased, thus depriving users' benefits and lowering network utilization.

One way to solve this problem is by using a nested DiffServ model, where each group of flows can have a subset of requirements. This model can be combined with the explicitly endpoint admission control (EEAC) scheme that represents the nested-DiffServ service levels as service vectors (SV) [1]. Consider n service classes $S = (S_0, S_1, \ldots, S_{n-1})$ provided by each link in a network. One flow, going from the source to the destination via m nodes or $m-1$ links, may choose service $s_i (s_i \in S)$ at router i. The service at s_i, may be different from service s_j selected by router j. The service vector hereby is defined as $s = (s_0, s_1, \ldots, s_{m-1}, s_m)$. The aim of service vectors is to find the suitable service classes in a single path so as to maximize

$$G = \max(U - C) \tag{1}$$

where U is the utility function and C is the cost. This combination of service vectors decouples the provisioning of end-to-end QoS at each router, thus resulting in an intermediate level of granularity and complexity between per-flow and per-group levels. The EEAC scheme can be performed in two phases: the probing (or exploring) phase, to determine link state, and the data transmission phase, which is performed after the probing (and call acceptance) processes. In the probing phase, the end host sends probing packets to the destination host to collect the SV information, which includes the service states of the routers along the end-to-end path. After receiving feedback from the end server and retrieving the state from the probing packets, the end host compares all possible service class combinations for this specific path, and computes the utilization and cost to find the most suitable service classes to be used at each router per flow basis. The selected service vector is marked in the data packets during the data transmission phase. Each router checks the vector, and provides the cost of the corresponding QoS service in it. This EEAC-SV model improves the QoS granularity to $O(p^q)$, where p is the number of routers and q is the number of service classes in the network (or end-to-end path). The flexibility feature increases the probability of minimizing the cost for the user and network utilization for the service provider. However, this scheme, as other endpoint admission control (EAC) models, assumes that the path is pre-selected so that the probing path and data transmission path are always fixed. This assumption simplifies the analysis, but it may not be accurate in the case of considering a real network. The reason is that in routing mechanisms, the above QoS provisioning scheme may not be able to provide the flexibility achievable by EEAC if SVs are not considered in the path calculation.

Here, we consider the open shortest path first (OSPF) [2] as the widely used QoS routing model. In Section II, we present two examples to show that the combination of OSPF and EEAC may cause false routing, which results in low utilization of the network resources and in high cost. To solve this problem, we propose using OSPF to select SVs during the path selection phase. Because a link is the connection between two routers, the different service classes available of a link can be detected by the neighboring routers and disseminated by OSPF in a timely fashion. Furthermore, the concept of security-enabled QoS concept (SQoS) and a solution to achieve the optimal path selection are also presented in this paper.

The reminder of this paper is organized as follows. Section II describes some of the drawbacks of the EEAC approach for path selection. Section III describes our proposed OSPF-provisioning approach. Section IV discusses the consideration of security issues. Section V present the analysis of the path selection algorithm. Section VI presents our conclusions.

This work is supported in part by National Science Foundation under Grant Awards 0435250 and 0423305.

The authors are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, University Heights, Newark, NJ 07102. Email : {zq4, rrojas, ansari}@njit.edu

II. Drawbacks of EEAC with Path Selection

Without loss of generality, OSPF can be used to select a path for the EEAC scheme. However, at the present, some vendors simply set the weight of links in OSPF to be inversely proportional to the capacity of the link. This configuration cannot reflect the accurate QoS state of the network. Some other setting methods reported in the literature [3] [4] are weights mapped to the combination of QoS parameters like delay, available bandwidth. However, currently there is no prominent definition of weight setting according to QoS as the multi-dimensional feature of QoS results in a complex weight setting. Figure 1 shows an example of a simple network with routers $N = (N_1, N_2, N_3, N_4, N_5, N_6)$. Assume the delay of the path is determined by the QoS requirement. The service class $S = (S_1, S_2, S_3, S_4)$ is thus categorized by the delay of the link as shown below. The cost C of the service is represented as:

$$S = \begin{cases} S_1 & (delay < 5ms) \\ S_2 & (5ms \leq delay < 10ms) \\ S_3 & (10ms \leq delay < 20ms) \\ S_4 & (delay \geq 20ms) \end{cases}$$

$$C = \begin{cases} 6 & (S = S_1) \\ 3 & (S = S_2) \\ 2 & (S = S_3) \\ 1 & (S = S_4) \end{cases}$$

Here, $W_i, i \in (1, 10)$, as shown in the graph, is the weight of each link recorded by OSPF. Now let us assume that a flow from N_1 to N_6 has a delay request of less than 25 ms. OSPF will select shortest path as $P_1 = (N_1, N_2, N_6)$ according to the addition of weights on the path. The EEAC-SV scheme then executes the probing processes along P_1, but no SV may satisfy the delay request for less than 25ms, so that the request is denied. However, the rest of the paths, such as $P_2 = (N_1, N_5, N_6)$ and $P_3 = (N_1, N_3, N_5, N_6)$, can satisfy the user's request with the proper service class selection. Furthermore, it is easy to see that the SV (S_2, S_2) with P_2 and with the lowest cost ($C = 3 + 3 = 6$) is the optimal solution. Therefore, the EEAC-SV scheme suffers of false routing in this case.

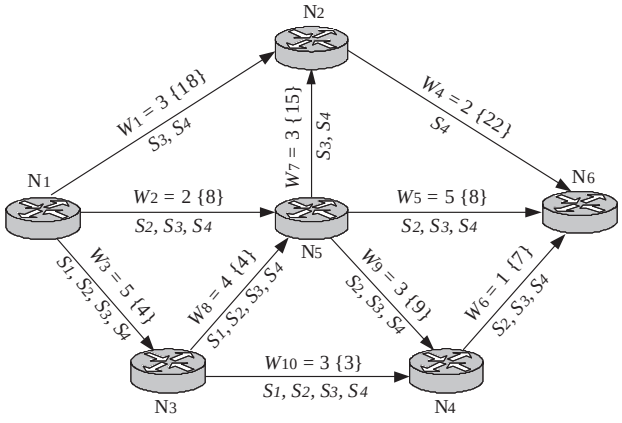

Fig. 1. Network topology of the first example

As another example, we set the link weight as the function of delay, as shown in the brackets in Figure 1, or:

$$W_i = f(delay) = \lceil delay \rceil .$$

Then $P_4 = (N_1, N_3, N_4, N_6)$ is the shortest path found by OSPF, which makes EEAC select (S_1, S_2, S_2) or (S_2, S_1, S_2) as the solution (where the cost is 12). However, the optimal answer in this case is $P_2 = (N_1, N_5, N_6)$ with service (S_2, S_2) in tandem (where cost is 6). The non optimal solution increases the cost and diminishes the network utilization.

III. OSPF-based Adaptive and Flexible QoS Provisioning

To overcome the above problem, we propose a new architecture based on OSPF. In our description, the weight of link i (W_i) is proposed to be represented as a vector that contains all the available service classes the link can provide, documented as:

$$Wi = (S_1, S_2, ...S_k) \quad s.t. \quad (S_1, S_2, ...S_k) \in S \tag{2}$$

Generally, the service class in a QoS model is defined in function of various QoS parameters. We note the service class as $S_i = (Q_i^1, Q_i^2, \ldots, Q_i^k)$, where $Q_i^j, j \in (1, \ldots, k)$ is the j_{th} QoS component in the i_{th} service class. For instance, a service class may be defined as (delay, jitter, packet loss, available bandwidth). In this way, different service classes share the common network resource, so they can be compared by the QoS parameters and sorted in a linear order. In our model, only the highest available service class needs to be marked as the weight of the link. However, the services of the lower classes can also be provided. Here, the amount of data used to represent the state class is reduced, that is, the overhead of link state update is decreased. Same as in the original OSPF protocol, the link states are exchanged by link state advertisement (LSA) packets among the routers in the network, so that every router has the same QoS link state database. When the user's request comes, the edge router, i.e. source node, selects the shortest path that satisfies 1. All the service classes lower than or equal to the weight of the link are candidates for selection. The edge router here is different from that in generic QoS routing in the sense that the router not only selects the path the data go through but also the service classes of the link as requested. If the user's requirement can be satisfied, the SV as the selected service classes are marked into the data packets during data transmission. The traversed routers read the service class from each packet and provide the corresponding service. If no SV can be found to fulfill users' demand, the request is denied.

As for the link-state update, the router estimates the state of the connected link and launches the state update mechanism when any of the states change. However, the estimation of the performance of each service cannot be accurate because if the state is updated too frequently these updates generate large traffic overhead. Besides, to prevent the LSA packets and the following data packets from increasing their overhead, we use divide the values of the QoS parameters in each service class into several service levels, so that $\lceil \log_2 M \rceil$ bits can represent M service classes.

OSPF sets a link to disseminate its state information every 30 minutes. This update interval may be too large for our architecture if high data rate flows are considered, because the dynamic changes of the QoS parameters may cause the state already known to other routers to become outdated. To overcome this problem, the update is triggered when the state of a link crosses a boundary of service level, which is called as a class-based triggering mechanism [5].

IV. Combination of Security and QoS

It is well known that the security level of a given communication depends on the individual user. Therefore, it is difficult to evaluate security uniformly and classify its service level. In order to shape the problem, we estimate the security protection capability of the router and linearly map it to the level of security satisfaction of users. The most widely considered security capabilities of the router can be listed as encryption, denial-of-service (DoS) detection, authentication, and virus filtering. Besides these, more security components can be extended in this framework. To the best of our knowledge, no quantitative benchmark of network security in the literature has been proposed. Thus, in this paper we create the following criterion to evaluate them, but other standards are equally applicable:

- Encryption K_e: measured by the bit-length of the encryption key (e.g., 64 bits, 128 bits) and strength of cryptographic algorithm (e.g., RSA, DES).
- Virus scanning K_v: measured by the number of viruses and worms that the anti-virus software can detect and the false-alarm ratio.
- DoS detection K_d: measured by the false positive ratio of intrusion detection system (IDS) under uniform DoS attack testing.
- Authentication K_a: measured according to the robustness of the authentication mechanism. It might contain a weak or strong password, biometric, and smart cards with onboard display and input interfaces.

Here, link security state is expressed as a vector K_a, K_v, K_d, K_a. From the user's viewpoint, the security of link i is the addition of the above four components:

$$K_i = a_1 K_e^i + a_2 K_v^i + a_3 K_d^i + a_4 K_a^i \tag{3}$$

Let us assume there are n links in the path under study, and that the security level of the path is the link with minimum security:

$$K_p = \min(K_1, K_2, \ldots, K_n) \tag{4}$$

$a_i,\ i \in (1,2,3,4)$ is the sensitivity weight of individual security component, as a user is concerned about the different security components that are specific to a given situation. As security protection capability mentioned here is considered to change at lower rates than the other QoS parameters, the security values are updated with a low frequency, such as once every 24 hours. During each update interval, this value is considered to remain constant in each service class. This decreases the computation work of routers and produces no increases in the complexity of the link state update.

As we mentioned above, equation 1 is utilized for service vector selection. The cost of the security service S_i in each router is related to the processor occupancy time and strength level $C_p(S_i)$, the occupied memory $C_m(S_i)$, the bandwidth $C_b(S_i)$, and the disk space $C_d(S_i)$ for a database to store virus or DoS attack patterns. The cost function of security service is the sum of all of them:

$$C_{security}(S_i) = C_p(S_i) + C_m(S_i) + C_b(S_i) + C_d(S_i). \tag{5}$$

V. Path Selection Algorithm Analysis

Generally data flows can specify their QoS requirements in terms of four parameters: the available bandwidth B_{req}, the maximum jitter request J_{req}, the maximum delay request

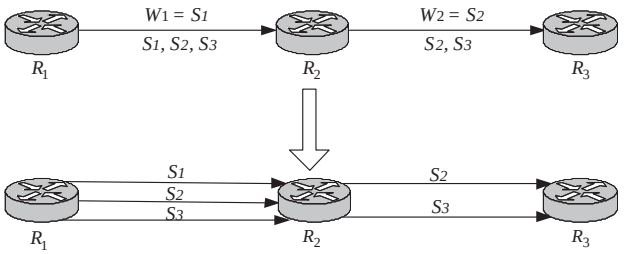

Fig. 2. The illustration of virtual links with service class.

D_{req}, and the minimum security requirement K_{req}. The path and SV selection problem thus can be described as the problem to maximize 1 as long as it is eligible for the selected path p_j,

$$\begin{aligned} B_p(P_j) \geq B_{req}, \quad D_p(P_j) \leq D_{req} \\ J_p(P_j) \leq J_{req}, \quad K_p(P_j) \geq K_{req} \end{aligned} \tag{6}$$

From the user perspective, the utility function U reflects the degree of users' satisfaction to the QoS service. Users' QoS requirement can be elastic or inelastic. With elastic demand the user can tolerate some degree of service deterioration if QoS provisioning is lower than the user expected constraint; while inelastic ones means otherwise. In this paper, we focus on inelastic QoS requirements so U is either 1 or 0. Maximizing (1) is equivalent to minimize cost function C (i.e. the multi-constrained least cost routing with multi-service is selectable), which is an NP-complete problem [6]. Let us firstly consider the several service classes in each link. To convert their multiple-to-one relationship to one-to-one mapping, each service class is regarded as a virtual link, as Figure 2 shows.

To simplify the above problem, we categorize users' constraints into two different classes and analyze the algorithm with each type of constraints. The combination of those algorithms will give the solutions but that is beyond the scope of this paper. One class is called concave or bottleneck constrained, such as the cases for available bandwidth, and security. This can be solved by using an extension of the Dijkstra algorithm as in 3. Assume a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of links. s and d are the source node and destination node, respectively. For the mth service class and $|E|$ links in the graph G, the time complexity of the pre-process part is $O(m\,|E|)$; the time complexity of main-selection part is $O(n^2)$. Thereby the total complexity of this algorithm is $O(n^2)$, which has the same order of complexity as the original Dijkstra algorithm.

The other class is called as additive constrained, such as delay and jitter are. Assume the set of feasible paths from node s to d is F. The cost function is C. The, the problem is defined as:

$$\begin{aligned} C = \min\{F\} \\ s.t. \quad \forall p_j \in F,\ D_{p_j} \leq D_{req}\ (or\ J_{p_j} \leq J_{req}) \end{aligned}$$

With respect to the lowest cost, the problem is known as Delay-Constrained-Least-Cost (DCLC). Many mechanisms are proposed to solve it in polynomial time, among which k-shortest paths (KSP) is a good solution. We propose our KSP based on Jimenez and Marzal's Recursive Enumeration Algorithm (REA) [7]. The idea is to list k shortest paths from s to d with increasing costs of weight in a directed graph.

QoS-Routing (G, s, d, K_{req})
Pre-Process Part
for each $e \in E$
$temp = \phi$
Set m service classes
for each $k \in m$
if $K_{e^k} \geq K_{req}$ **then**
$temp := temp \cup \{k\}$
Recode virtual links with lowest service class
$E := E - \{e^k\}, \forall k \in m$ except $k = \min\{temp\}$

Main Selection
$Dijkstra(E, s, d)$

Fig. 3. The path selection algorithm for concave constraint

The algorithm first invokes Dijkstra's shortest path algorithm to build the shortest path tree. Each path from s to current node v is the concatenation of the path from s to $pre(v)$ and the link $(pre(v), v)$, while $pre(v)$ is the adjacent predecessor node. The k_{th} shortest path $\pi^k(v)$ is thus selected from the candidate set $C^k(v)$ according to the following generalized Bellman's equation:

$$C^k(v) = \begin{cases} \pi^1(u) \cdot v \forall u \in pre(v) & if\ k = 1\ \&\ v \neq s, \\ & or\ k = 2\ \&\ v = s; \\ (C^{k-1}(v) - \pi^g(u).v) \cup \pi^{g+1}(u) \cdot v & otherwise; \end{cases}$$

$$\pi^k(v) = \begin{cases} s & if\ k = 1\ \&\ v = s; \\ argmin_{\pi \in C^k(v)} L(\pi) & otherwise; \end{cases}$$

where $\pi^{k-1}(u) = \pi^g(u) \cdot v$ and $L(\pi)$ is the weight of the path π.

The above recursive computation to obtain the k shortest paths solution is finished in $O(m + Kn\log(m/n))$ time. To avoid the worst case when K is large, we provide the following algorithm for the DCLC problem. Once the delay constraint is above the threshold, we track the kth longest path based on the longest path tree instead of the shortest one.

Build Shortest and Longest Path Tree
$path_{shortest} = Dijkstra(V, E)$;
$path_{longest} = Dijkstra'(V, E)$;
$threshold = h * (Delay(path_{shortest}) + Delay(path_{longest}))$;

K Paths Selection
if $Delay_{constraint} < Delay(path_{shortest})$ **then**
Request is denied
else if $Delay_{constraint} \leq threshold$
invoke k shortest path algorithm
else if $Delay_{constraint} \geq threshold$
invoke k longest path algorithm
$cost = \min(cost(\pi^i(d)), \forall i \in k)$

Fig. 4. The path selection algorithm for additive constraint

We simulated the algorithm in Figure 4 in the 32-node bidirectional network in [8] by running 10,000 requests. Each link is replaced by three virtual links to represent the service classes. Without loss of generality, the delay of the virtual link is uniformly distributed from 1 to 500. The source and destination node is 1 and 30. Here, h is set to 0.5. The delay constraint set is from 150 to 1500 with 50 between intervals. Figure 5 shows the average number of iterations k, where k is found without setting the upper bound of k and considering that there are optimal feasible paths. The figure shows that the number of iterations is not large and has not unlimited increase when the delay constraint increases. In reality, as the service class is distributed uniformly among links, the variety of construction paths decreases. This means that the number of k is actually smaller than that shown in the figure.

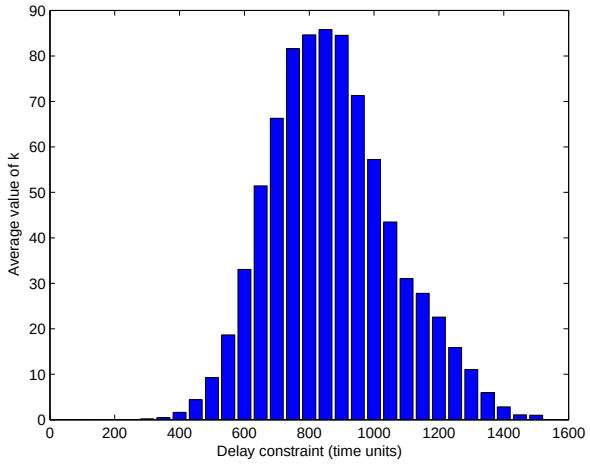

Fig. 5. Average number of iterations of our algorithm in 32-node network

VI. Conclusions

In this paper we proposed an extended OSPF framework where SVs and path selection are integrated into one phase to provide the flexibility of QoS and high utilization of the network. An efficient algorithm as well as the combination of security and QoS into what is called SQoS are introduced. To overcome the complexity of deployability of new mechanisms, we considered the to embed SVs into OSPF routing to guarantee feasible deployability into existing networks. Our future work includes the development of the efficient routing algorithm with multi-additive constraints and other constraints, such as number of hops, the extension of our architecture to wireless network, and link-state update mechanisms for accurate and practical QoS routing.

References

[1] J. Yang, J. Ye, and S. Papavassiliou, "A Flexible and Distributed Architecture for Adaptive End-to-End QoS Provisioning in Next Generation Networks," *IEEE JSAC*, Vol. 23, No. 2, pp. 321-333, Feb. 2005.
[2] J. Moy, "OSPF version 2," in *IETF*, RFC2328, Apr. 1998.
[3] G. Apostolopoulos, et al, "QoS Routing Mechanisms and OSPF Extensions," *IETF*, RFC 2676, Aug. 1999.
[4] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," *IEEE JSAC*, Vol. 20, No. 4, pp. 756-767, May 2002.
[5] G. Apostolopoulos, et al, "Quality-of-service based routing: A Performance Perspective," *ACM SIGCOMM Computer Communication Review*, Vol. 28, No. 4, pp. 17-28, Oct. 1998.
[6] Z. Wang and J. Crowcroft, "Quality of Service Routing for Supporting Multimedia Applications," *IEEE JSAC*, Vol. 14, No. 7, pp. 1228-1234, Sep. 1996.
[7] V. Jimenez and A. Marzal, "Computing the k Shortest Paths: A New Algorithm and an Experimental Comparison," *Lecture Notes in Computer Science: 1668*, pp. 15-29, 1999.
[8] S. Chen and K. Nahrsted, "On Finding Multi-Constrained Paths," *IEEE ICC*, Vol. 2, pp. 874-899, June 1996.