

# Quality-of-Service Routing with Path Information Aggregation

Ka-Chung Leung, King-Shan Lui, Ka-Cheong Leung

September 16, 2007

## 1 Introduction

Routing is a process of finding a network path from a source node to a destination node. In order to deal with the scalability problem, large networks are often structured hierarchically by grouping nodes into different domains. Each domain that participates in the Internet is an *Autonomous System*. In a hierarchical network, different routing protocols can be used at different levels. In the Internet, a path-vector routing protocol, which is enhanced from the distance-vector protocol, called BGP, is used for *inter-domain (inter-AS) routing* that finds a path from one domain to another. Each AS, on the other hand, selects its own *intra-domain (intra-AS) routing protocol* to find a path from one node to another within the same domain. OSPF (a link-state routing protocol) and RIP (a distance-vector routing protocol) are examples of IETF-standardized intra-domain routing protocols.

A good routing protocol should find the “best path” from a source to a destination. In most standard routing protocols, the “best path” usually refers to the shortest path in terms of a single metric. This single metric may be the number of hops, delay, bandwidth, cost (such as administrative cost), and so on. Both link-state and distance-vector protocols can find a path, say, with

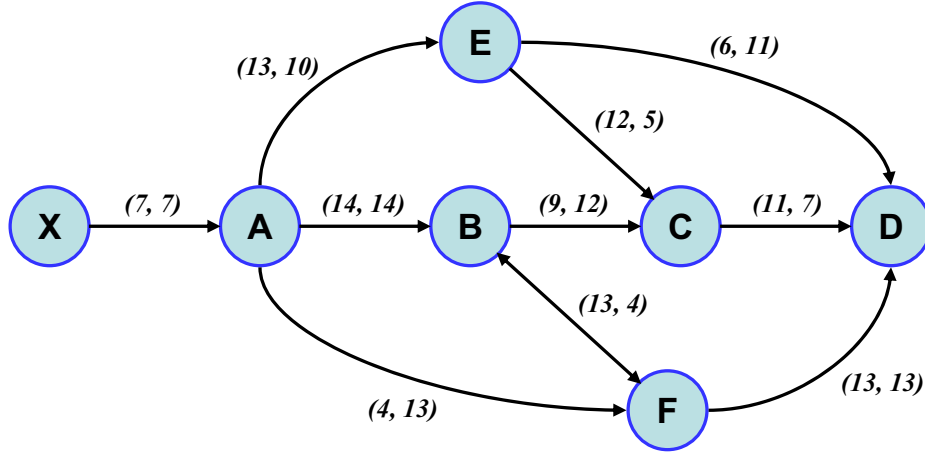


Figure 1: A simple network where  $(x, y)$  represents the QoS metrics of security level and bandwidth, respectively.

the minimum hop count, minimum delay, maximum bandwidth, or minimum cost, from one node to all the other nodes in the network. Unfortunately, when it comes to two or more metrics, the problem of *finding the best path* is not trivial anymore.

Consider the simple network in Figure 1. The tuple  $(x, y)$  associated with each edge represents the QoS metrics of security level and bandwidth, respectively. Note that the network contains both unidirectional edges and bidirectional edges. The QoS of the path  $A \rightarrow E \rightarrow D$  is  $(\min(13, 6), \min(10, 11)) = (6, 10)$  while the QoS of path  $A \rightarrow B \rightarrow C \rightarrow D$  is  $(9, 7)$ . The former is better in terms of bandwidth and the latter is better in terms of security level. No matter which path is selected as the “best” and kept in the routing table, some feasible QoS requests will not be admitted. Suppose that Node  $A$  decides to keep path  $A \rightarrow B \rightarrow C \rightarrow D$  in its routing table and then receives a routing request to  $D$  that requires 8 units of bandwidth value. By checking its routing table,  $A$  thinks that there is no path from itself to  $D$  having 8 units of bandwidth and rejects the request. However, the request is actually feasible since it is supported by path  $A \rightarrow E \rightarrow D$ .

If the parameters of all the paths from a source to a destination are plotted on a bandwidth-security level plane geometrically, the region of supported services forms a staircase. Figure 2 illustrates the idea. Points  $(4, 4)$ ,  $(4, 13)$ ,  $(6, 10)$ ,  $(9, 7)$ ,  $(11, 5)$ , and  $(13, 4)$  refer to the parameters

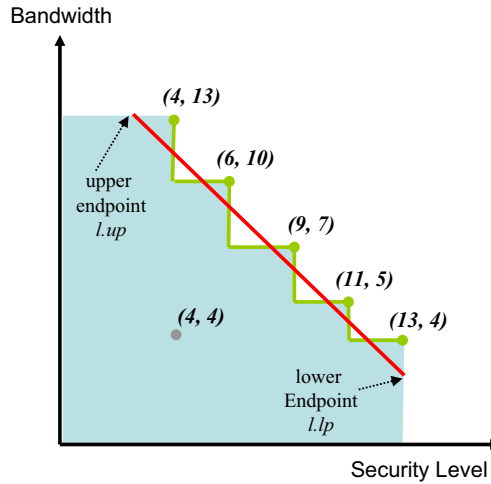


Figure 2: Parameters on bandwidth-security level plane.

of the paths from Node  $A$  to  $D$ .  $(9, 7)$  is definitely better than  $(4, 4)$  since it is better in both security level and bandwidth. However,  $(9, 7)$  is neither better than  $(6, 10)$  nor  $(11, 5)$ . The shaded area represents the feasible requests that can be supported by at least one path. To see why the area is a staircase, we look at the region supported by a certain path, say  $(9, 7)$ .  $(9, 7)$  can support all requests of security level requirement not larger than 9 units and the bandwidth requirement not larger than 7 units. The supported security level and bandwidth values of that path fall into the lower left quadrant of  $(9, 7)$ . The region supported by all paths is the union of the regions of all paths, forming a staircase as shown in Figure 2. This staircase can be represented by the points on the convex corners of the stairs and these points are referred to as *representative points* that can be identified in polynomial time [1].

Although the region of supported services can be represented by a set of representative points, it is not scalable to advertise the whole staircase to neighbors for routing. For example, in Figure 1 with the distance-vector routing protocol,  $A$  should advertise to  $X$  the QoS information from itself to  $D$ . It would be too expensive if the whole staircase is advertised. The authors in [1] have developed a *line segment* approach to solve the problem. A line, as shown in Figure 2, is used to approximate a staircase. Since a line can be uniquely represented by two points, advertising line segments reduces the size of the disseminated information dramatically so as to make it relatively

inexpensive.

Unfortunately, as the supported services represented by a line is not the same as the staircase, some QoS information may be lost. The optimum line is the one that minimizes the area between itself and the staircase. Referring to Figure 2, this area is the sum of the areas of the triangles formed between the staircase and the line segment. [1] uses linear regression to find such a line, without dealing with the more complex objective function to minimize the captioned area. The line segment approach is further studied as part of a *topology aggregation* scheme in hierarchical networks. Topology aggregation refers to the process of summarizing the internal topology information of a domain and is defined by the ATM PNNI standard. A link-state based routing protocol is developed to work with the line segment representation in [1]. In [2], the authors study how the line segment technique can be adopted for distance-vector based routing protocols when delay and bandwidth are considered.

In this paper, we study how to apply the line segment representation when bandwidth and security level are considered. As there are not many studies on how to put QoS information into the context of inter-domain routing, it is worth expanding the QoS routing research onto this area. The rest of the paper is organized as follows: Section 2 presents our network model and Section 3 presents the notations being used in our paper. Section 4 describes our main findings, followed by our simulation results in Section 5.

## 2 Network Model and Problem Statement

A large network consists of a set of domains and links that connect them as shown in a simple network in Figure 3. It is modelled as a directed graph where link metrics can be asymmetric in the two directions. A node is called a *border node* (black nodes in Figure 3) if it connects to a node of another domain.

A domain is modelled as a tuple  $(V, B, E)$ , where  $V$  is the set of nodes in the domain,  $B \subseteq V$

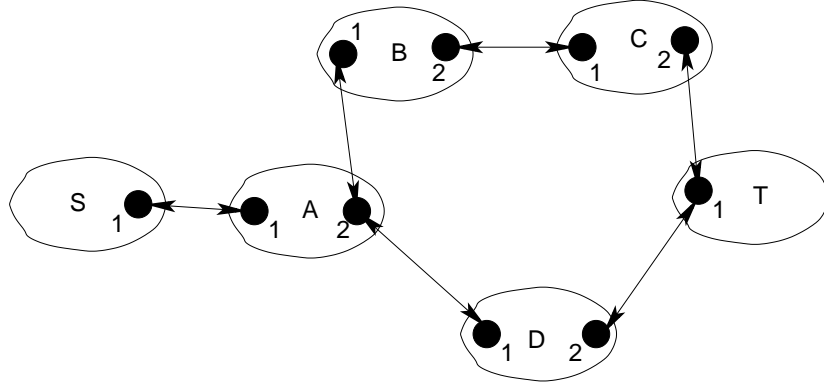


Figure 3: A simple two-level hierarchical network.

is the set of border nodes, and  $E$  is the set of directed links among the nodes in  $V$ . To simplify our discussion, we refer border node  $i$  of domain  $d$  as  $d.i$ . In Figure 3,  $A.1$  and  $A.2$  are border nodes of domain  $A$ . The entire network is modelled as  $(G, L)$ , where  $G = \{g_i | g_i = (V_i, B_i, E_i), 1 \leq i \leq N\}$  is the set of  $N$  domains and  $L$  is the set of links connecting domains.

The links in  $L$  and  $E_i, 1 \leq i \leq |G|$ , are called *physical links*. Note that a “physical link” in  $L$  is basically a layer-3 link. The QoS parameter of a physical link is denoted as a pair  $(S, W)$ , where  $S$  is the security level and  $W$  is the link bandwidth. Each pair of  $(S, W)$  represents a single point on the bandwidth-security level plane. A *physical path* from node  $v_0$  to node  $v_k$ , which is denoted by  $(v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_{k-1} \rightarrow v_k)$ , consists of a set of directed links  $(v_i, v_{i+1}) \in \cup_{1 \leq i \leq |G|} E \cup L$ , for  $0 \leq i < k$ . Let  $(S_{i \rightarrow i+1}, W_{i \rightarrow i+1})$  be the QoS parameter of link  $(v_i, v_{i+1})$ , where  $S_{i \rightarrow i+1}$  is the security level and  $W_{i \rightarrow i+1}$  is the bandwidth of  $(v_i, v_{i+1})$ . The security level of the path from  $v_0$  to  $v_k$  is  $\min_{i=0}^{k-1} \{S_{i \rightarrow i+1}\}$  and the bandwidth is  $\min_{i=0}^{k-1} \{W_{i \rightarrow i+1}\}$ . For example, if  $k = 3$  and the parameters of  $(v_0, v_1)$ ,  $(v_1, v_2)$ , and  $(v_2, v_3)$  are  $(3,5)$ ,  $(5,4)$ , and  $(6,4)$ , respectively, then the security level of  $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3$  is  $\min\{3, 5, 6\} = 3$  and the bandwidth is  $\min\{5, 4, 4\} = 4$ . In the geometrical representation, like in Figure 2, the QoS parameter pair of a physical path is denoted by a point in the bandwidth-security level plane.

A border router running the BGP protocol advertises its path information to neighboring border routers. By applying the distance-vector routing protocol mechanism, every domain can construct

a routing table that specifies the next hop neighbors on the shortest path tree for all other domains. For example,  $C.2$  knows that  $T.1$  is its direct neighbor and it relays this information to  $C.1$ . Since  $C.1$  and  $C.2$  are in the same domain, even if they may be several hops away, they are regarded as direct BGP neighbors. When  $C.1$  gets the information from  $C.2$ ,  $C.1$  knows that it can reach  $T.1$  through  $C.2$ . It then informs  $B.2$  that it can reach  $T.1$ . If a single QoS parameter is considered, say bandwidth, border nodes should also advertise the bandwidth information. There are two paths from  $A.2$  to  $T.1$ . Suppose that the bandwidth value of the lower path is larger. Then,  $A.2$  should specify in its routing table that the next hop neighbor leading to  $T.1$  is  $D.1$ . Furthermore,  $A.2$  advertises only the lower path's bandwidth value to  $A.1$ .

However, when two (QoS) parameters are considered, the distance-vector calculation and advertisement procedure are not trivial. In this paper, we identify and solve the three main problems that directly follow from routing with security level and bandwidth as the metrics:

1. How can  $B.1$  find the QoS from itself to  $T.1$ ? We assume that the QoS from  $C.1$  to  $C.2$  and the QoS from  $B.1$  to  $B.2$  are represented by line segments as discussed in Section 1. Although the QoS of the inter-domain links  $C.2$  to  $T.1$  and  $B.2$  to  $C.1$  are both  $(S, W)$  tuples, we have to join these QoS parameters together to find the QoS from  $B.1$  to  $T.1$ . This is described in Section 4.1.
2. Also, when  $A.2$  advertises to  $A.1$  the QoS information leading to  $T$ , it may not be possible to determine whether the upper path or the lower path is better. Therefore, in order to embed as much of the underlying QoS information of the paths as possible in advertisement, we have to *aggregate* the QoS of the two paths together. This is described in Section 4.2.
3. Finally, how should the routing table of an individual node be changed in order to cope with the new routing mechanism so that the node can determine how to forward a packet upon receiving a QoS request? This is described in Section 4.3.

### 3 Notation

This section describes the notations being used in our proposed work in subsequent sections.

When the parameters of all the paths from source to destination are plotted on the bandwidth-security level plane, a staircase can be drawn such that the supported service area lies on the bottom-left quadrant of the staircase as already mentioned in Section 1. An example is shown in Figure 2. Since it would be too expensive for a node to advertise the whole staircase, a line segment is constructed to approximate it. Such line segment is used to represent the QoS of multiple physical paths between a source-destination pair. Given the QoS parameters, we follow the *line segment* approach developed by [1] to find such a line.

The security level and bandwidth of a QoS point  $p$  on the bandwidth-security level plane are denoted as  $p.s$  and  $p.w$ , respectively, so a QoS point  $p$  can be written as a tuple  $(p.s, p.w)$ . In our discussion, a line segment  $l$  is denoted as  $[l.lp, l.up]$  where  $l.lp$  and  $l.up$  are the lower and upper endpoints of the line, respectively. Thus, the two endpoints of a QoS line can be written as  $(l.up.s, l.up.w)$  and  $(l.lp.s, l.lp.w)$ . Moreover, we denote the security level of a line segment  $l$  at bandwidth  $bw$  as  $l.SLatBW(bw)$ , and the bandwidth of the line segment at security level  $sl$  as  $l.BWatSL(sl)$ .

**Definition 1** A point  $p$  is more representative than a point  $p'$  if and only if

- $p.s \neq p'.s$  or  $p.w \neq p'.w$ , and;
- $p.s \geq p'.s$  and  $p.w \geq p'.w$ .

**Definition 2** Given a set  $S$  of points on the bandwidth-security level plane,  $(x, y) \in S$  is a representative point of  $S$  if and only if there does not exist any other point  $(x', y') \in S$  which is more representative than  $(x, y)$ . Otherwise  $(x, y)$  is a non-representative point.

Any staircase can be uniquely specified by a set of representative points. Note that if  $S$  contains the parameters of all physical paths from the source to the destination, then *stair*, which consists of

the representative points only, is as descriptive as including all points  $(x, y) \in S$  on the bandwidth-security level plane, since non-representative points can simply be ignored without affecting the supported service area as illustrated by the point (4, 4) in Figure 2. The representative points of *stair* are denoted as  $rp_1, rp_2, \dots, rp_n$  from the bottom-right corner to the top-left corner and we can represent a staircase as  $stair = (rp_1, rp_2, \dots, rp_n)$  where every element is a representative point. When there are multiple staircases  $stair_1$  and  $stair_2$  to be referenced, the representative points of  $stair_1$  and  $stair_2$  are denoted as  $rp_1^1, rp_2^1, \dots, rp_{n_1}^1$  and  $rp_1^2, rp_2^2, \dots, rp_{n_2}^2$ , respectively. If a particular segment of  $stair_1$  is to be referred, we use the notation  $stair_{1_1}, stair_{1_2}, \dots, stair_{1_m}$ , etc. A *null staircase* means there is no representative point in the "staircase".

Consider two QoS points,  $pt_1$  and  $pt_2$ . If  $pt_2$  is more representative than  $pt_1$ , we represent it as  $pt_2 \succ pt_1$  or  $pt_1 \prec pt_2$ . If  $pt_2$  is not more representative than  $pt_1$ , we represent it as  $pt_2 \not\succeq pt_1$  or  $pt_1 \not\prec pt_2$ .

We make use of the variable  $qos_n$ , where  $n = 1, 2, 3, \dots$ , to denote either a QoS point *or* a QoS line. We define a *QoS join operation* on two QoSs  $qos_1$  and  $qos_2$  as a QoS finding process for a path  $(v_a \rightarrow v_b \rightarrow v_c)$  where the two concatenating paths  $(v_a \rightarrow v_b)$  and  $(v_b \rightarrow v_c)$  are of  $qos_1$  and  $qos_2$ , respectively. The join operation is denoted using the symbol " $\oplus$ ". Thus, if  $qos_3$  is the QoS of the path  $(v_a \rightarrow v_b \rightarrow v_c)$ , then  $qos_3 = qos_1 \oplus qos_2$ .

The union operation of multiple QoSs  $(qos_1, qos_2, \dots, qos_n)$  is defined as the total supported service region supported by those QoSs. This union operation is denoted as  $(qos_1 \cup qos_2 \cup \dots \cup qos_n)$ .

## 4 QoS Routing Mechanism

### 4.1 QoS Join Operation

A logical inter-domain or intra-domain path  $(v_x \rightarrow v_y)$  can be a QoS point or a QoS line. To find the join of two paths  $(v_a \rightarrow v_b)$  and  $(v_b \rightarrow v_c)$ , three cases are possible — a point joining another

point, a point joining a line, and a line joining another line. For the case of a point joining another point ( $rp_1 \oplus rp_2$ ), the join result is a QoS point at  $(\min(rp_1.s, rp_2.s), \min(rp_1.w, rp_2.w))$ . The explanation is trivial. If a routing path is the concatenation of two physical links, then the combined supported QoS is the composition of the worst case of every metric supported in the two links.

In the following subsections, we prove and describe the join operations of a point against a line and a line against another line.

#### 4.1.1 Joining a point and a line

To provide a joining mechanism of a QoS point  $p$  and a line  $l$ , we shall derive it by considering the original staircase  $stair$  with representative points  $rp_1, rp_2, \dots, rp_n$  that will be approximated by the line  $l$ . Assume  $p$  is the QoS of the physical link  $(v_a, v_b)$  and  $stair$  is QoS of the logical path  $(v_b \rightarrow v_c)$ . Since every  $rp_i$  for  $1 \leq i \leq n$  represents a physical link  $(v_b, v_c)$ , the join operation can be viewed as combining the QoS of  $p$  with every  $rp_i$  in  $stair$ . In other words, the QoS of the path  $(v_a \rightarrow v_b \rightarrow v_c)$  is the union of the individual  $p \oplus rp_i$ . Therefore, we have:

$$\begin{aligned}
p \oplus stair &= (p \oplus rp_1) \cup (p \oplus rp_2) \cup \dots \cup (p \oplus rp_n) \\
&= (\min(p.s, rp_1.s), \min(p.w, rp_1.w)) \cup (\min(p.s, rp_2.s), \\
&\quad \min(p.w, rp_2.w)) \cup \dots \cup (\min(p.s, rp_n.s), \min(p.w, rp_n.w)) \\
&= \bigcup_{i=1}^n (\min(p.s, rp_i.s), \min(p.w, rp_i.w)) \tag{1}
\end{aligned}$$

Given staircase  $stair$  on the bandwidth-security level plane, the plane can be divided into seven regions, as shown in Figure 4. The mechanism of the joining is different when point  $p$  is located in different regions on the plane. Mathematically,  $p$  is said to be in a particular region according to the following:

- *Region I*: there exists some points  $rp_z \in stair$  such that  $rp_z$  is more representative than  $p$ .

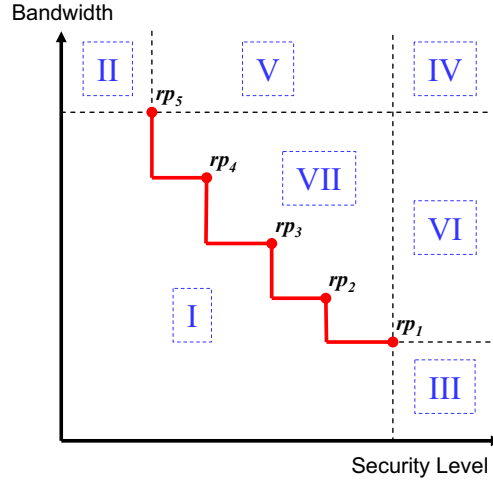


Figure 4: The seven regions on the bandwidth-security level plane when a staircase is given.

- *Region II*:  $p.s < rp_n.s$  and  $p.w \geq rp_n.w$ .
- *Region III*:  $p.s \geq rp_1.s$  and  $p.w < rp_1.w$ .
- *Region IV*:  $p.s \geq rp_1.s$  and  $p.w \geq rp_n.w$ .
- *Region V*:  $rp_n.s \leq p.s < rp_1.s$  and  $p.w \geq rp_n.w$ .
- *Region VI*:  $p.s \geq rp_1.s$  and  $rp_1.w \leq p.w < rp_n.w$ .
- *Region VII*:  $p.s < rp_1.s$  and  $p.w < rp_n.w$  and there does not exist any point  $rp_z \in \text{stair}$  such that  $rp_z$  is more representative than  $p$ .

Now, we derive the result of joining for  $p$  in each region. The graphical representation of each case is shown in Figure 5.

1. *Region I*:

The physical meaning of this situation comes directly from the fact that the physical link  $(v_a, v_b)$  constraints both the bandwidth and security level of the aggregated path  $(v_a \rightarrow v_b \rightarrow v_c)$ . By definition of Region *I*, there exists a tuple of  $(\min(p.s, rp_m.s), \min(p.w, rp_m.w))$ ,  $1 \leq m \leq n$ , in Equation 1 that is equal to  $(p.s, p.w)$ . Every other tuple  $i$ ,  $1 \leq i \neq m \leq n$ , of

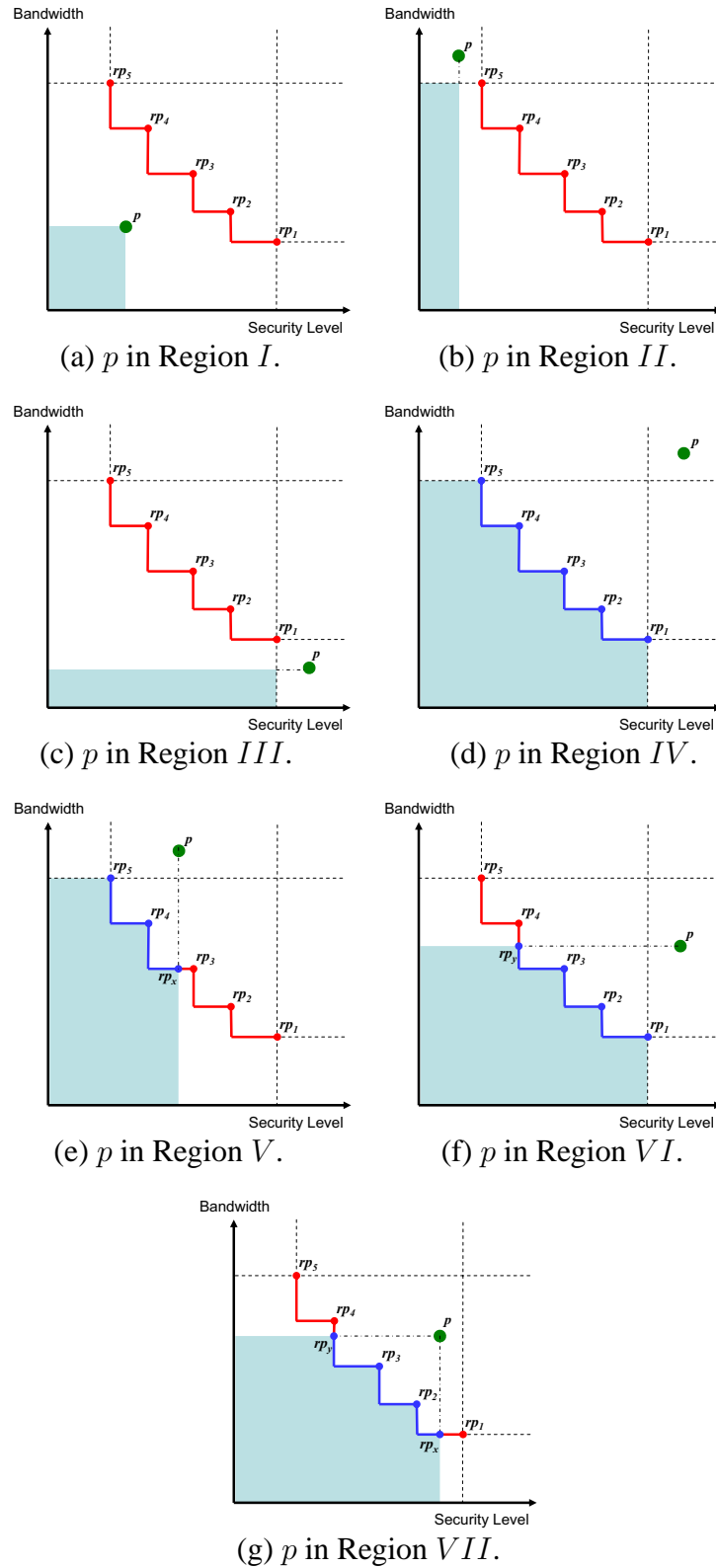


Figure 5: Graphical representation of all cases for a QoS point joining a QoS stair (aggregated service area denoted as the shaded region).

the join yields  $(p.x_i, p.y_i)$  where  $p.x_i \leq p.s$  or  $p.y_i \leq p.w$ . Therefore, except  $(p.s, p.w)$  all other joined tuples are non-representative points. The resulting QoS for the join operation is a single point at  $(p.s, p.w)$ , as shown in Figure 5(a).

## 2. Region II:

By evaluating Equation 1, we have

$$p \oplus stair = (p.s, rp_1.w) \cup (p.s, rp_2.w) \cup \dots \cup (p.s, rp_n.w) \quad (2)$$

The physical link  $(v_a, v_b)$  constraints the security level of the aggregated path.  $(p.s, rp_n.w)$  is more representative than  $(p.s, rp_1.w), (p.s, rp_2.w), \dots, (p.s, rp_{n-1}.w)$  because it has the largest bandwidth. The resulting QoS for the join operation is a single point at  $(p.s, rp_n.w)$ , as shown in Figure 5(b).

## 3. Region III:

By evaluating Equation 1, we have

$$p \oplus stair = (rp_1.s, p.w) \cup (rp_2.s, p.w) \cup \dots \cup (rp_n.s, p.w) \quad (3)$$

The physical link  $(v_a, v_b)$  constraints the bandwidth of the aggregated path.  $(rp_1.s, p.w)$  is more representative than  $(rp_2.s, p.w), (rp_3.s, p.w), \dots, (rp_n.s, p.w)$  because it has the largest security level. The resulting QoS for the join operation is a single point at  $(rp_1.s, p.w)$ , as shown in Figure 5(c).

## 4. Region IV:

By evaluating Equation 1, we have

$$p \oplus stair = (rp_1.s, rp_1.w) \cup (rp_2.s, rp_2.w) \cup \dots \cup (rp_n.s, rp_n.w) \quad (4)$$

In this situation, the physical link  $(v_a, v_b)$  does not put any constraint on bandwidth nor security level no matter what path for the route  $(v_b \rightarrow v_c)$  is chosen. Therefore, the result of the join returns the union of all the original representative points in  $stair$ , so the resulting QoS for the join is the original staircase  $stair$ , as shown in Figure 5(d).

#### 5. Region V:

In this case, we solve the problem by breaking the staircase  $stair$  into  $stair_1$  with representative points  $rp_1, rp_2, \dots, rp_k$  and  $stair_2$  with representative points  $rp_{k+1}, rp_{k+2}, \dots, rp_n$  such that  $rp_k.s > p.s \geq rp_{k+1}.s$ . Note that  $stair = stair_1 \cup stair_2$ . Therefore, we have

$$\begin{aligned}
p \oplus stair &= p \oplus (stair_1 \cup stair_2) \\
&= (p \oplus stair_1) \cup (p \oplus stair_2), \text{ where} \\
p \oplus stair_1 &= p \oplus (rp_1 \cup rp_2 \cup \dots \cup rp_k), \text{ and} \\
p \oplus stair_2 &= p \oplus (rp_{k+1} \cup rp_{k+2} \cup \dots \cup rp_n)
\end{aligned} \tag{5}$$

For the case  $p \oplus stair_1$ ,  $p$  is in fact in Region II of  $stair_1$ . Following the discussion above, the resulting QoS is therefore a point at  $(p.s, rp_k.w)$ . For the case  $p \oplus stair_2$ ,  $p$  is in fact in Region IV of  $stair_2$ . Following the discussion above, the resulting QoS is therefore the original staircase  $stair_2$ . Thus, the aggregated QoS of the join operation is a new staircase  $stair'$  composed of  $rp_x = (p.s, rp_k.w)$  and all the representative points in  $stair_2$ , as shown in Figure 5(e). Note that the point  $rp_x$  must be a representative point in  $stair'$  because its security level is greater than every point in  $stair_2$ .

#### 6. Region VI:

In this case, we follow similar approach by breaking the staircase  $stair$  into  $stair_1$  with representative points  $rp_1, rp_2, \dots, rp_k$  and  $stair_2$  with representative points  $rp_{k+1}, rp_{k+2}, \dots, rp_n$  such that  $rp_k.w \leq p.w < rp_{k+1}.w$ . Therefore, Equation 5 still applies.

For the case  $p \oplus stair_1$ ,  $p$  is in fact in Region *IV* of  $stair_1$ . Following the discussion above, the resulting QoS is therefore the original staircase  $stair_1$ . For the case  $p \oplus stair_2$ ,  $p$  is in fact in Region *III* of  $stair_2$ . Following the discussion above, the resulting QoS is therefore a point at  $(rp_{k+1}.s, p.w)$ . Thus, the aggregated QoS of the join operation is a new staircase  $stair'$  composed of  $rp_y = (rp_{k+1}.s, p.w)$  and all the representative points in  $stair_1$ , as shown in Figure 5(f). Note that the point  $rp_y$  must be a representative point in  $stair'$  because its bandwidth is greater than every point in  $stair_1$ .

### 7. Region *VII*:

The proof of this case is the combination of that in Region *V* and Region *VI* above. The staircase is broken down into:

- $stair_1$  with representative points  $rp_1, rp_2, \dots, rp_j$
- $stair_2$  with representative points  $rp_{j+1}, rp_{j+2}, \dots, rp_k$
- $stair_3$  with representative points  $rp_{k+1}, rp_{k+2}, \dots, rp_n$

with the criteria that  $rp_j.s > p.s \geq rp_{j+1}.s$  and  $rp_k.w \leq p.w < rp_{k+1}.w$ .

Note that  $stair = stair_1 \cup stair_2 \cup stair_3$ . Therefore,

$$\begin{aligned}
p \oplus stair &= p \oplus (stair_1 \cup stair_2 \cup stair_3) \\
&= (p \oplus stair_1) \cup (p \oplus stair_2) \cup (p \oplus stair_3), \text{ where} \\
p \oplus stair_1 &= p \oplus (rp_1 \cup rp_2 \cup \dots \cup rp_j), \\
p \oplus stair_2 &= p \oplus (rp_{j+1} \cup rp_{j+2} \cup \dots \cup rp_k), \text{ and} \\
p \oplus stair_3 &= p \oplus (rp_{k+1} \cup rp_{k+2} \cup \dots \cup rp_n) \tag{6}
\end{aligned}$$

For the case  $p \oplus stair_1$ ,  $p$  is in fact in Region *II* of  $stair_1$ , so the resulting QoS is therefore a point at  $(p.s, rp_j.w)$ . For the case  $p \oplus stair_2$ ,  $p$  is in fact in Region *IV* of  $stair_2$ , so

the resulting QoS is therefore the original staircase  $stair_2$ . For the case  $p \oplus stair_3$ ,  $p$  is in fact in Region *III* of  $stair_3$ , so the resulting QoS is therefore a point at  $(rp_{k+1}.s, p.w)$ . Thus, the aggregated QoS of the join operation is a new staircase  $stair'$  composed of  $rp_x = (p.s, rp_j.w)$ ,  $rp_y = (rp_{k+1}.s, p.w)$  and all the representative points in  $stair_2$ . This is shown in Figure 5(g).

In the above, we have proved the result of joining for  $p$  being located in different region of  $stair$ . However, in real situation, a line  $l$  used to approximate the staircase is advertised to a neighbor instead of the staircase itself. Thus, we have to approximate  $rp_1$  with the lower endpoint of  $l$ ,  $l.lp$ , and  $rp_n$  with the upper endpoint of  $l$ ,  $l.up$ .  $l.lp$  may have its bandwidth value different from that of  $rp_1$  while  $l.up$  may have its security level value different from that of  $rp_n$ . Also, we have to believe that there are infinite number of representative points along  $l$ , so the indicated service area will be different from the area covered by the original staircase. In this way, Figure 4 has to be modified to deal with such approximations, as redrawn in Figure 6. It can be observed that except Region *IV*, the bound of all other regions could have been shifted. Mathematically,  $p$  is said to be in a particular region according to which condition below is satisfied:

- *Region I*: there exists some points  $rp_z \in l$  such that  $rp_z$  is more representative than  $p$ .
- *Region II*:  $p.s < l.up.s$  and  $p.w \geq l.up.w$ .
- *Region III*:  $p.s \geq l.lp.s$  and  $p.w < l.lp.w$ .
- *Region IV*:  $p.s \geq l.lp.s$  and  $p.w \geq l.up.w$ .
- *Region V*:  $l.up.s \leq p.s < l.lp.s$  and  $p.w \geq l.up.w$ .
- *Region VI*:  $p.s \geq l.lp.s$  and  $l.lp.w \leq p.w < l.up.w$ .
- *Region VII*:  $p.s < l.lp.s$  and  $p.w < l.up.w$  and there does not exist any point  $rp_z \in l$  such that  $rp_z$  is more representative than  $p$ .

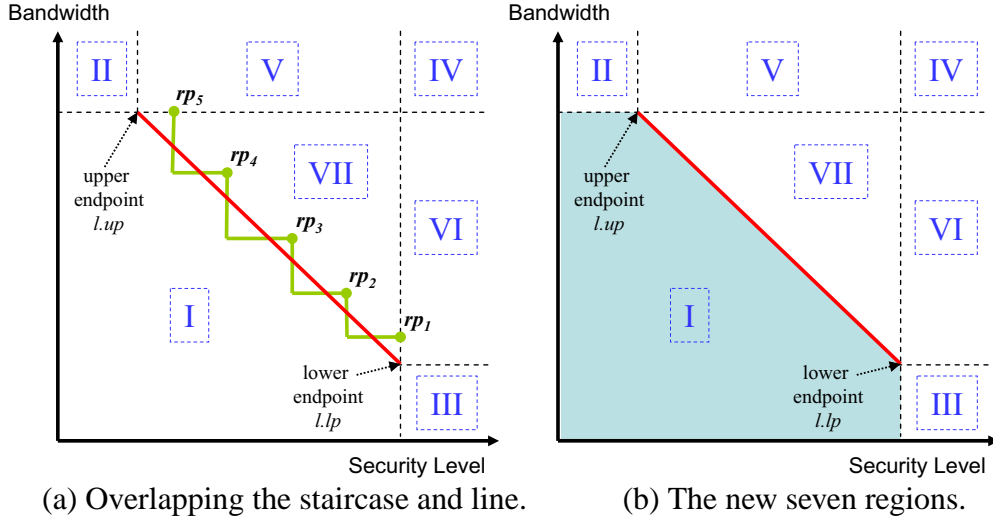


Figure 6: The seven regions on the bandwidth-security level plane when the staircase is approximated by a line.

As the line has been treated as if there are infinite many representative points, the proof of a point joining a staircase mentioned above will still hold. Any  $rp_x$  and  $rp_y$  generated will stick to the new line obtained after the join operation because the representative points are continuous in space. Now, when a point  $p$  is joined with a line  $l$ , we first determine the region that  $p$  is in. According to the derivation above, a new set of join result directly follows:

$$p \oplus l = \begin{cases} (p.s, p.w), & \text{if } p \in \text{Region I}; \\ (p.s, l.up.w), & \text{if } p \in \text{Region II}; \\ (l.lp.s, p.w), & \text{if } p \in \text{Region III}; \\ [(l.lp.s, l.lp.w), (l.up.s, l.up.w)], & \text{if } p \in \text{Region IV}; \\ [(p.s, l.BWatSL(p.s)), (l.up.s, l.up.w)], & \text{if } p \in \text{Region V}; \\ [(l.lp.s, l.lp.w), (l.SLatBW(p.w), p.w)], & \text{if } p \in \text{Region VI}; \\ [(p.s, l.BWatSL(p.s)), (l.SLatBW(p.w), p.w)], & \text{if } p \in \text{Region VII}. \end{cases}$$

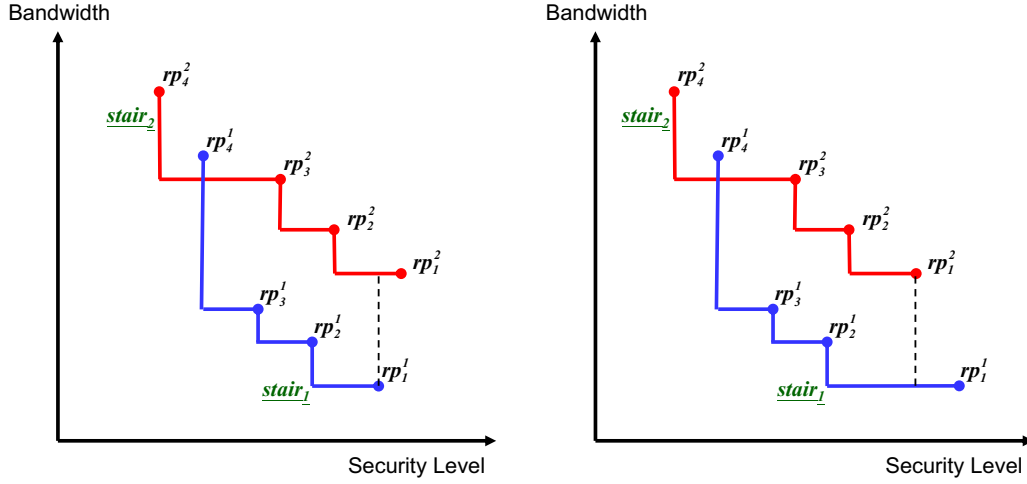


Figure 7: Naming of staircase.

#### 4.1.2 Joining a line and another line

Similar to the previous subsection, to provide a joining mechanism of a QoS line  $l_1$  and another line  $l_2$ , we shall derive it by considering the original staircase  $stair_1$  with representative points  $rp_1^1, rp_2^1, \dots, rp_n^1$  and  $stair_2$  with representative points  $rp_1^2, rp_2^2, \dots, rp_n^2$  that are approximated by the lines  $l_1$  and  $l_2$ , respectively. Given two staircases in the bandwidth-security level plane, the correct naming, as described in Definition 3, of the staircases  $stair_1$  and  $stair_2$  is important for the latter proof.

**Definition 3** Naming of staircase (illustrated in Figure 7):

Let  $s' = \min(rp_1^x.s, rp_1^y.s)$ . If  $stair_x.BW_{atSL}(s') \leq stair_y.BW_{atSL}(s')$ , then  $rp_1^x$  (and therefore all other points in  $stair_x$ ) is on  $stair_1$ . Otherwise,  $rp_1^x$  is on  $stair_2$ .

Assume  $stair_1$  is the QoS of the logical path  $(v_a \rightarrow v_b)$  and  $stair_2$  is the QoS of the logical path  $(v_b \rightarrow v_c)$ . As every  $rp_i^1$  (where  $1 \leq i \leq n_1$ ) combined with every  $rp_j^2$  (where  $1 \leq j \leq n_2$ ) represents the path from  $v_a$  to  $v_c$  via  $v_b$ , the join operation can be viewed as combining the QoS of every  $rp_i^1$  on  $stair_1$  with every  $rp_j^2$  on  $stair_2$ . By taking union of all the service areas that are supported by different paths, the QoS of the path  $(v_a \rightarrow v_b \rightarrow v_c)$  can be found and a new staircase  $stair_3$  will be obtained. Therefore, we have

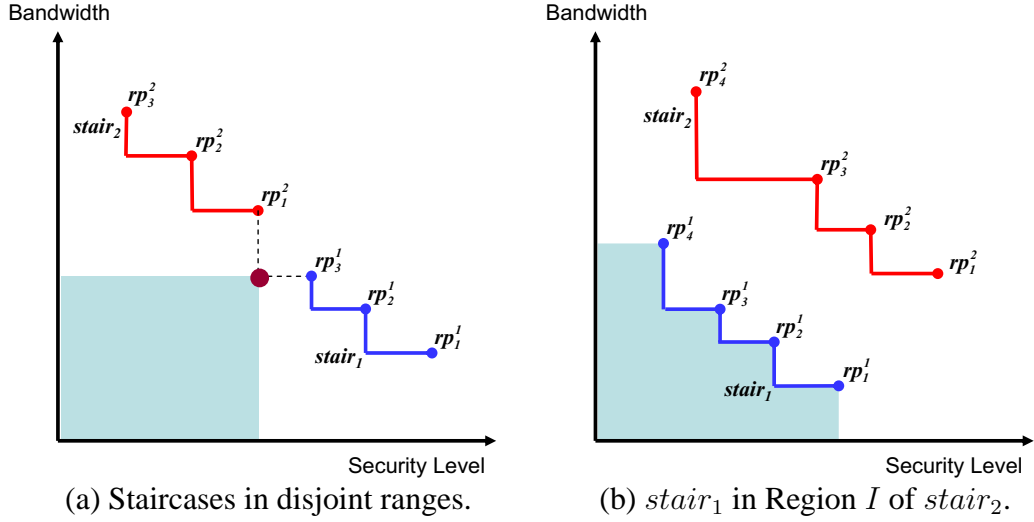


Figure 8: Two fundamental cases of  $stair_1$  and  $stair_2$ .

$$\begin{aligned}
stair_3 &= stair_1 \oplus stair_2 \\
&= (rp_1^1 \cup rp_2^1 \cup \dots \cup rp_{n_1}^1) \oplus (rp_1^2 \cup rp_2^2 \cup \dots \cup rp_{n_2}^2) \\
&= (rp_1^1 \oplus stair_2) \cup (rp_2^1 \oplus stair_2) \cup \dots \cup (rp_{n_1}^1 \oplus stair_2) \\
&= \bigcup_{i=1}^{n_1} (rp_i^1 \oplus stair_2)
\end{aligned} \tag{7}$$

Then, we can enumerate every element  $(rp_i^1 \oplus stair_2)$  in Equation 7 using Equation 1 and finally obtain the union of  $n^2$  tuples.

To find the join result of arbitrary  $stair_1$  and  $stair_2$ , we first need to define and formulate the join result of two fundamental cases, as shown in Figure 8. Case (1) is when  $stair_1$  and  $stair_2$  have disjoint bandwidth and security level ranges with  $rp_{n_1}^1.s > rp_1^2.s$ . Case (2) is when every representative point in  $stair_1$  is in Region I of  $stair_2$ . With these two fundamental cases, we will be able to derive the join result of any geometrical placement of  $stair_1$  and  $stair_2$ .

- Fundamental case (1):

In  $stair_1$ , all representative points are positioned in Region III of  $stair_2$ . Applying the

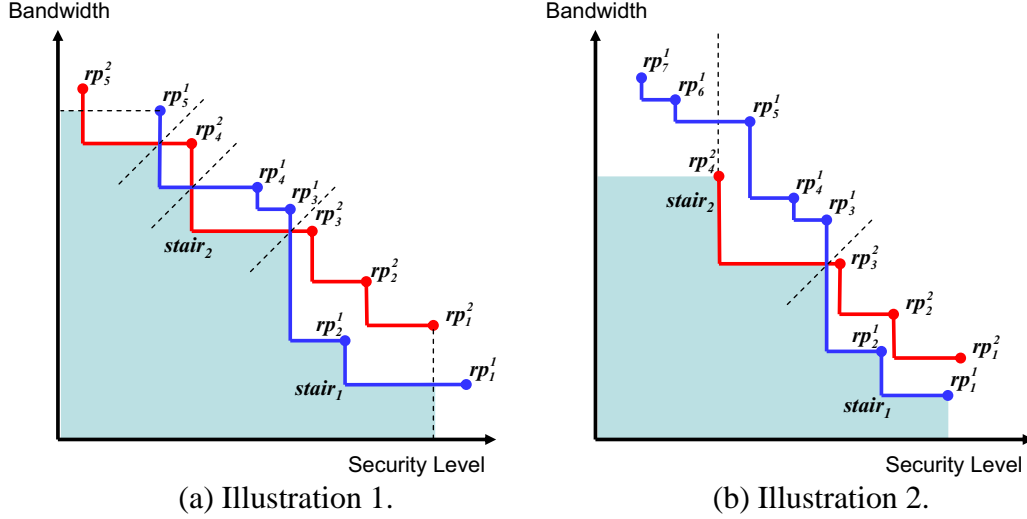


Figure 9: Segmentation of  $stair_1$  and  $stair_2$ .

result of Section 4.1.1, the joining of every  $rp_i^1$  ( $1 \leq i \leq n_1$ ) with  $stair_2$  results in a single point at  $(rp_1^2.s, rp_i^1.w)$ . Since the point  $(rp_1^2.s, rp_{n_1}^1.w)$  has the largest bandwidth value, it must be the only representative point that is formed by joining  $stair_1$  with  $stair_2$ . This case is shown in Figure 8(a).

- Fundamental Case (2):

When every representative point in  $stair_1$  is in Region I of  $stair_2$ , the joining of every  $rp_i^1$  (where  $1 \leq i \leq n_1$ ) with  $stair_2$  results in the original point at  $(rp_i^1.s, rp_i^1.w)$  according to the result from Section 4.1.1. Finally, the original staircase  $stair_1$  will remain after the joining. This case is shown in Figure 8(b).

With the above two cases proved, we can return to the general situation. The approach is like this. Given any  $stair_1$  and  $stair_2$ , if they entirely belong to one of the above fundamental cases, the derivation is finished. Otherwise, we first need to break the staircases into segments. Whenever there is an intersection point between  $stair_1$  and  $stair_2$ , the two staircases are being segmented by a breaking line. Two examples of segmentation are shown in Figure 9. Then, we separately join every segment in  $stair_1$  to  $stair_2$  to obtain the join result. Between the breaking lines (dotted line

in Figure 9) there is a portion of  $stair_1$  and a portion of  $stair_2$ . We refer them as a *segment group*.

Note the special cases regarding the starting and ending point of the staircases. Three cases are to be considered:

1. If  $rp_1^1.s > rp_1^2.s$  (an example is shown in Figure 9(a)), then we have to identify a portion of  $stair_{1_1} = (rp_1^1, rp_2^1, \dots, rp_a^1) \in stair_1$  such that  $rp_a^1.s > rp_1^2.s$  and  $rp_{a+1}^1.s \leq rp_1^2.s$ .  $stair_{1_1}$  itself will form an additional segment group despite there is no intersection point between the staircases. Joining of  $stair_{1_1}$  to  $stair_2$  is our fundamental case (1). The result will be a QoS point at  $rp_x = (rp_1^2.s, rp_a^1.w)$ .
2. If  $rp_{n_1}^1.s < rp_{n_2}^2.s$  and  $rp_{n_1}^1.w > rp_{n_2}^2.w$  (an example is shown in Figure 9(b)), then we have to identify a portion of  $stair_{1_r} = (rp_{b+1}^1, rp_{b+2}^1, \dots, rp_{n_1}^1) \in stair_1$  such that  $rp_{b+1}^1.s < rp_{n_2}^2.s$  and  $rp_{b+1}^1.w > rp_{n_2}^2.w$ . Now,  $stair_{1_r}$  itself will form an additional segment group despite there is no intersection point. In  $stair_{1_r}$ , all representative points are positioned in Region II of  $stair_2$ . Applying the result of Section 4.1.1, the joining of every  $rp_i^1$  ( $b+1 \leq i \leq n_1$ ) with  $stair_2$  results in a single point at  $(rp_i^1.s, rp_{n_2}^2.w)$ . Since the point  $rp_y = (rp_{b+1}^1.s, rp_{n_2}^2.w)$  has the largest security level value, it must be the only representative point that is formed by joining  $stair_{1_r}$  with  $stair_2$ .
3. If  $rp_{n_1}^1.s \geq rp_{n_2}^2.s$  and  $rp_{n_1}^1.w < rp_{n_2}^2.w$  (an example is shown in Figure 9(a)), then we have to identify a portion of  $stair_{2_t} = (rp_{c+1}^2, rp_{c+2}^2, \dots, rp_{n_2}^2) \in stair_2$  such that  $rp_{c+1}^2.s \leq rp_{n_1}^1.s$  and  $rp_{c+1}^2.w > rp_{n_1}^1.w$ .  $stair_{2_t}$  itself will form an additional segment group that does not include any portion of  $stair_1$ .

At the stage, we have correctly identified all the segment groups given two staircases. Assume we have  $g$  segment groups, the remaining step is to join the segments  $stair_{1_i}$  ( $1 \leq i \leq g$ ) to the whole  $stair_2$  and combine the results altogether. In fact, there are only two possible staircase organizations within each segment group no matter how many segment groups exist in total:

1. Every representative point in  $stair_1$ 's portion ( $stair_{1_x}$ ) is in Region  $I$  of  $stair_2$ :

In this case, when we join  $stair_{1_x}$  in the segment group to  $stair_2$ , fundamental case (2) directly applies. The join result is the original segment  $stair_{1_x}$ . Note that all points in  $stair_{1_x}$  will remain as representative points after fully joining  $stair_1$  with  $stair_2$  since no other pair of points joining together can yield a QoS point which is more representative than any point in  $stair_{1_x}$ . If  $stair_{1_x}$  is a null staircase (for example the segment group that contains only the point  $rp_4^2$  in Figure 9(a)), then nothing has to be joined.

2. Every representative point in  $stair_2$ 's portion ( $stair_{2_y}$ ) is in Region  $I$  of  $stair_1$ 's portion ( $stair_{1_x}$ ):

In this case, we divide the joining into three steps.

- $stair_{1_x}$  joining with  $stair_{2_y}$ ;
- $stair_{1_x}$  joining with  $stair_{2_i}$  ( $y + 1 \leq i \leq g$ ), and;
- $stair_{1_x}$  joining with  $stair_{2_j}$  ( $1 \leq j \leq y - 1$ ).

For the first step, every representative point in  $stair_{2_y}$  is in Region  $I$  of  $stair_{1_x}$ , so the joining of  $stair_{2_y}$  with  $stair_{1_x}$  results in the original staircase  $stair_{2_y}$ . This is proved in Section 4.1.1.

For the second step, we merge  $stair_{2_i}$  ( $y + 1 \leq i \leq g$ ) into a single staircase  $stair'_2$ .  $stair_{1_x}$  joining with  $stair'_2$  is in fact our fundamental case (1). The join result is therefore a point exactly at the upper intersection point of the segment group concerned.

For the third step, we merge  $stair_{2_j}$  ( $1 \leq j \leq y - 1$ ) into a single staircase  $stair''_2$ .  $stair_{1_x}$  joining with  $stair''_2$  is a variant of the fundamental case (1), with the position of the two staircases being swapped. Trivially, the join result is a point exactly at the lower intersection point of the segment group concerned.

As  $stair_{1_x} \oplus stair_2 = (stair_{1_x} \oplus stair_{2_y}) \cup (stair_{1_x} \oplus stair'_2) \cup (stair_{1_x} \oplus stair''_2)$ , the

combined join result is therefore  $stair_{2_y}$  together with two intersection points. Note that these points will remain as representative points after fully joining  $stair_1$  with  $stair_2$  since no other pair of points joining together can yield a QoS point which is more representative than any of such points.

If  $stair_{2_y}$  is a null staircase (for example the segment group that contains only the points  $rp_3^1$  and  $rp_4^1$  in Figure 9(a)), then the combined join result will simply be the two intersection points after we skipped through the first step mentioned above.

Since  $stair_1 = stair_{1_1} \cup stair_{1_2} \cup \dots \cup stair_{1_g}$ , the problem of joining the two staircases can be broken down into the joining of  $stair_1$ 's individual staircase segment with  $stair_2$ . The above derivation can therefore effectively covers any geometrical placement of  $stair_1$  and  $stair_2$  in the bandwidth-security level plane. Intuitively, the result can be interpreted as when a staircase segment  $stair_{x_z}$  has all its representative points less representative than some representative points in another staircase  $stair_y$ , then all representative points in  $stair_{x_z}$  will become representative points after the join operation.

In the above, we have derived the result of joining  $stair_1$  with  $stair_2$ . However, in real situation, the staircases are unobtainable and two lines  $l_1$  and  $l_2$  are used to approximated the staircases. For  $stair_1$ , we have to approximate  $rp_1^1$  with  $l_1.lp$  and  $rp_{n_1}^1$  with  $l_1.up$ . For  $stair_2$ , we have to approximate  $rp_1^2$  with  $l_2.lp$  and  $rp_{n_2}^2$  with  $l_2.up$ .  $l_1.lp$  and  $l_2.lp$  may have their bandwidth values different from that of  $rp_1^1$  and  $rp_1^2$ , respectively, while  $l_1.up$  and  $l_2.up$  may have their security level values different from that of  $rp_{n_1}^1$  and  $rp_{n_2}^2$ , respectively. Also, we have to believe that there are infinite number of representative points along  $l_1$  and  $l_2$ , so the indicated service area will be different from the original staircases.

**Definition 4** Naming of QoS line:

Let  $s' = \min(l_x.lp.s, l_y.lp.s)$ . If  $l_x.BWatSL(s') \leq l_y.BWatSL(s')$ , then  $l_x.lp$  (and therefore all other points) is on  $l_1$ . Otherwise,  $l_x.lp$  is on  $l_2$ .

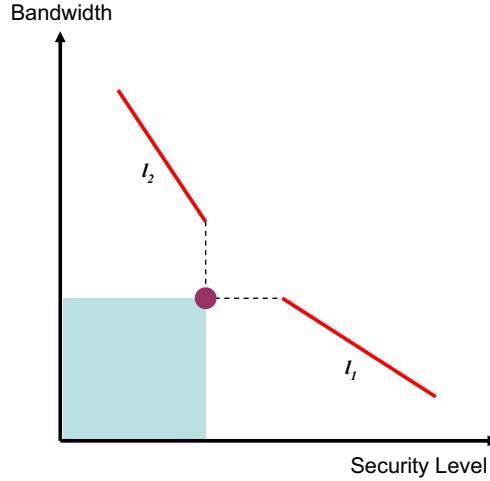


Figure 10: Graphical representation of joining  $l_1$  and  $l_2$  if  $l_1.up.s > l_2.lp.s$  and  $l_1.up.w < l_2.lp.w$  (aggregated service area denoted as the shaded region).

As the lines  $l_1$  and  $l_2$  have been treated as if there are infinite many representative points, the proof of a staircase joining another staircase mentioned above will still hold for line joining. An observation is that  $l_1$  and  $l_2$  will only cross each other at most once, compared to the general case that the staircases may cross each other infinitely many times. Furthermore, depending on the starting and ending point of the QoS lines, at most two additional QoS points  $rp_x$  and  $rp_y$  will be formed as discussed in staircase joining before. In the line joining case, these two representative points, if exist, will stick to the starting / ending point of the QoS segments after the joining.

Finally, according to the proof above, a new set of join result directly follows.

Consider the case where  $l_1.up.s > l_2.lp.s$  and  $l_1.up.w < l_2.lp.w$ , then  $l_1$  and  $l_2$  will have disjoint bandwidth *and* security level ranges. This is analogous to our fundamental case (1). In this case, the resulting QoS after join operation is a point at  $(l_2.lp.s, l_1.up.w)$ . This is shown in Figure 10.

Otherwise, eight different cases will be resulted, as shown in Figure 11. In the latter four cases where  $l_1$  and  $l_2$  cross each other, the resulting QoS is composed of a portion of  $l_1$  and a portion of  $l_2$  as inferred from the proof in the staircase scenario. Such QoS is no longer a straight line. However, it can be uniquely identified by three points in the bandwidth-security level plane. We

can therefore apply linear regression on these three points to find a line approximating the two line segments. The eight distinct cases are all shown in Figure 11.

Finally, the results are summarized as below:

If  $l_1.up.s > l_2.lp.s$  and  $l_1.up.w < l_2.lp.w$ ,  $l_1 \oplus l_2 = (l_2.lp.s, l_1.up.w)$ .

Otherwise, if  $l_1$  and  $l_2$  have no intersection point,

$$l_1 \oplus l_2 = \begin{cases} [(l_1.lp.s, l_1.lp.w), (l_1.up.s, l_1.up.w)], \\ \text{if } l_2.up.w \geq l_1.up.w \text{ and } l_2.lp.s \geq l_1.lp.s; \\ [(l_2.lp.s, l_1.BWatSL(l_2.lp.s)), (l_1.up.s, l_1.up.w)], \\ \text{if } l_2.up.w \geq l_1.up.w \text{ and } l_2.lp.s < l_1.lp.s; \\ [(l_1.lp.s, l_1.lp.w), (l_1.SLatBW(l_2.up.w), l_2.up.w)], \\ \text{if } l_2.up.w < l_1.up.w \text{ and } l_2.lp.s \geq l_1.lp.s; \\ [(l_2.lp.s, l_1.BWatSL(l_2.lp.s)), (l_1.SLatBW(l_2.up.w), l_2.up.w)], \\ \text{if } l_2.up.w < l_1.up.w \text{ and } l_2.lp.s < l_1.lp.s. \end{cases}$$

Otherwise, if  $l_1$  and  $l_2$  have one intersection point  $p_{int}$ ,  $l_1 \oplus l_2$  is the linear regression of three points, namely

$$\begin{cases} (l_2.up.s, l_2.up.w), (p_{int}.s, p_{int}.w), (l_1.lp.s, l_1.lp.w), \\ \text{if } l_2.up.w < l_1.up.w \text{ and } l_2.lp.s \geq l_1.lp.s; \\ (l_2.SLatBW(l_1.up.w), l_1.up.w), (p_{int}.s, p_{int}.w), (l_1.lp.s, l_1.lp.w), \\ \text{if } l_2.up.w \geq l_1.up.w \text{ and } l_2.lp.s \geq l_1.lp.s; \\ (l_2.up.s, l_2.up.w), (p_{int}.s, p_{int}.w), (l_2.lp.s, l_1.BWatSL(l_2.lp.s)), \\ \text{if } l_2.up.w < l_1.up.w \text{ and } l_2.lp.s < l_1.lp.s; \\ (l_2.SLatBW(l_1.up.w), l_1.up.w), (p_{int}.s, p_{int}.w), (l_2.lp.s, l_1.BWatSL(l_2.lp.s)), \\ \text{if } l_2.up.w \geq l_1.up.w \text{ and } l_2.lp.s < l_1.lp.s. \end{cases}$$

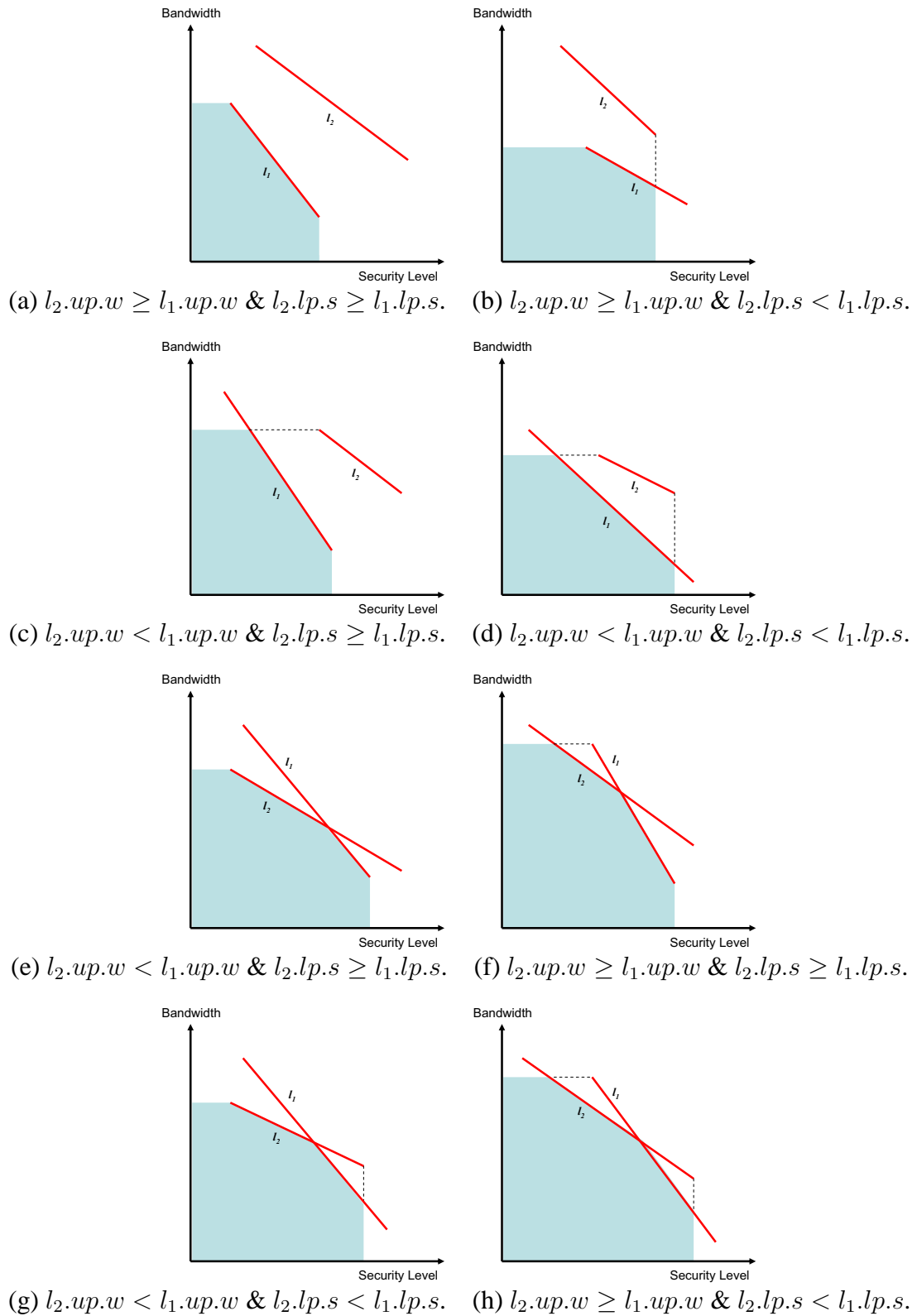


Figure 11: Graphical representation of joining  $l_1$  and  $l_2$ , with (e) - (h) having line intersections (aggregated service area denoted as the shaded region).

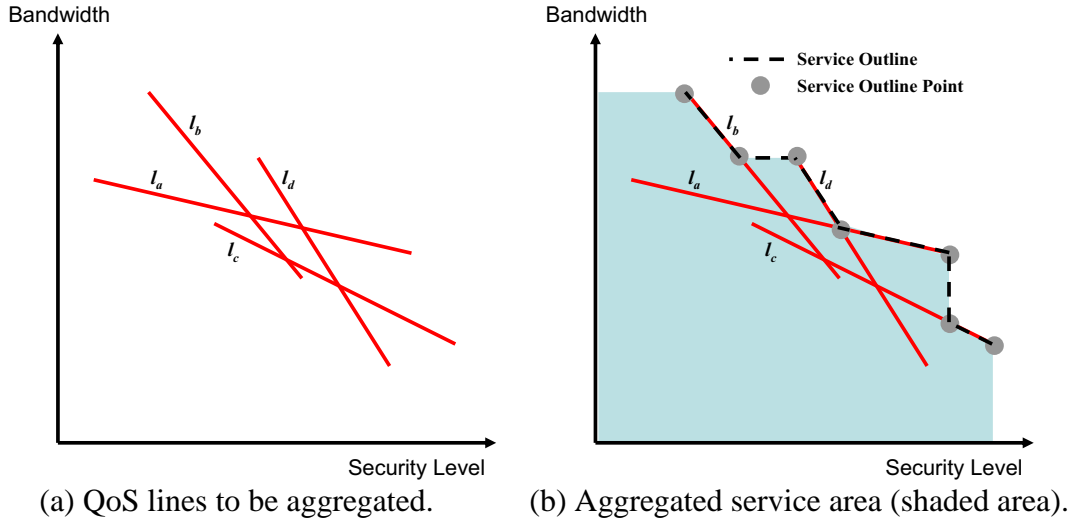


Figure 12: Line aggregation mechanism.

## 4.2 Line Aggregation

In order to enable the use of QoS line segment in distance-vector routing, another issue we need to consider is the mechanism for a node to determine how it should advertise the QoSs of several paths. Consider the case of Node  $A.2$  in Figure 3. The node knows the QoSs of the two paths leading to  $T$ , the upper path  $A.2 \rightarrow B \rightarrow C \rightarrow T$  and the lower path  $A.2 \rightarrow D \rightarrow T$ , and now it is its turn to advertise this information to its neighbor  $A.1$ . Which QoS should  $A.2$  advertise? If one QoS is better than the other,  $A.2$  can simply tell  $A.1$  the better QoS.

However, if the QoSs that a node possesses are the ones shown in Figure 12(a), it is difficult to tell which one is “better”. Our approach to the problem is to *aggregate* the QoS of the paths, representing the aggregated QoS with one new line segment. By doing so, we aim to embed as much of the QoS information as possible into one line segment in an advertisement and reduce the chance of rejecting feasible QoS requests.

In the bandwidth-security level plane, the QoS supported by an arbitrary line segment is the lower-left quadrant of the line. When a node receives multiple line segments for the same destination, the aggregated QoS is the union of the services supported by those lines. This is illustrated by the shaded area in Figure 12(b). In order to embed as much of the QoS information as possible, a

node should advertise the aggregated QoS instead of broadcasting one of the QoSs only. As shown in the figure, such service outline in general does not form a straight line but a polyline. The QoS polyline can be uniquely identified by specifying its service outline points (i.e., the dots in Figure 12(b)). As the number of service outline points grows linearly with the number of QoS lines to the same destination, advertising all service outline points is not scalable in the Internet. In order to solve this problem, we again use linear regression on the service outline points to approximate the service outline of the aggregated QoS with one line segment. This is a tradeoff between scalability and accuracy in route information.

We apply the *Method of Least Squares* on the service outline points. By such method, we attempt to obtain a straight line that best fits those points, and use this line as an approximation of the aggregated QoS to advertise to the direct neighbors.

To calculate the approximated line, we first define  $(x_1, y_1), (x_2, y_2), \dots, (x_{n_{pt}}, y_{n_{pt}})$ , be the set of service outline points, where  $n_{pt}$  is the number of service outline points and  $x_i$  and  $y_i$  are the security level and bandwidth of the  $i^{th}$  ( $1 \leq i \leq n_{pt}$ ) service outline point, respectively. The finding of the aggregated line consists of two steps:

1. Find the formula  $y = mx + b$  of the aggregated line using the Method of Least Squares, and
2. Find the two endpoints of it by substituting into the formula the known maximum security level and bandwidth supported by the aggregated QoS.

For the first step, we apply the following two formulae from the Method of Least Squares:

$$\begin{aligned}
 m &= \frac{(n_{pt} \times \sum_{i=1}^{n_{pt}} x_i y_i) - (\sum_{i=1}^{n_{pt}} x_i \times \sum_{i=1}^{n_{pt}} y_i)}{(n_{pt} \times \sum_{i=1}^{n_{pt}} x_i^2) - (\sum_{i=1}^{n_{pt}} x_i)^2} \\
 b &= \frac{\sum_{i=1}^{n_{pt}} y_i - m \times \sum_{i=1}^{n_{pt}} x_i}{n_{pt}}
 \end{aligned} \tag{8}$$

For the second step, we let  $l.lp$  and  $l.up$  be the lower and upper endpoints of the line  $l$ , respec-

tively. We also let  $max_s$  and  $max_w$  be the maximum security level and bandwidth supported by the aggregated QoS, respectively. As we want to restrict that  $l.lp.s = max_s$  and  $l.up.w = max_w$ , we find the two endpoints  $l.lp = (max_s, l.lp.w)$  and  $l.up = (l.up.s, max_w)$  by using the following formulae:

$$\begin{aligned} l.lp.w &= m \times max_s + b \\ l.up.s &= \frac{max_w - b}{m} \end{aligned} \tag{9}$$

Finally,  $l.lp = (max_s, l.lp.w)$  and  $l.up = (l.up.s, max_w)$  are advertised to the neighbors as the aggregated QoS to the destination.

### 4.3 Routing Updates

A router keeps two tables — a distance table and a routing table. Traditionally, the distance table records the link cost to each supported destination domain via each of its neighbors, and the routing table is constructed based on the distance table, specifying for each destination domain the corresponding next hop neighbor a request should be forwarded to. As we now have our new QoS joining and aggregation mechanism ready, we need to enhance the distance-vector routing protocol to embed QoS information in those tables of the routers in order to support QoS routing.

Consider a QoS request in the form  $(req_s, req_w)$ , where  $req_s$  and  $req_w$  are the required minimum security level and minimum bandwidth for this request. Our goal is that the router should accept any feasible request and forward the packet to the “best” next hop neighbor. There are two schemes to define, among all neighbors supporting the request  $(req_s, req_w)$ , the best neighbor:

1. if multiple neighbors support the required bandwidth  $req_w$ , it is desirable for routing through the neighbor with the maximum security level.

Distance Table

Domain	QoS to Domain via		
	A.1	B.1	D.1
$B$	(0, 0)	$pt_{A.2 \rightarrow B.1}$	(0, 0)
$C$	(0, 0)	(0, 0)	(0, 0)
$D$	(0, 0)	(0, 0)	$pt_{A.2 \rightarrow D.1}$
$S$	(0, 0)	(0, 0)	(0, 0)
$T$	(0, 0)	(0, 0)	(0, 0)

Table 1: Initial distance table of  $A.2$ .

2. If multiple neighbors support the required security level  $req_s$ , it is desirable for routing through the neighbor with the maximum bandwidth.

#### 4.3.1 New Distance Table

The distance table keeps one row for each supported domain and one column for each neighbor. The entry in row  $i$  and column  $j$  of the table kept in node  $N$  specifies the supported QoS, in terms of a line segment or a single QoS point, from  $N$  to domain  $i$  via neighbor  $j$ . If the number of supported domain be  $u$  and the number of direct neighbor be  $v$ , the distance table will be  $u \times v$  in size.  $N$  initializes its distance table with the QoS of directly attached links. The rest of the entries are initialized as (0, 0), meaning that  $N$  does not support any QoS for these neighbor-destination pairs. For example, the initial distance table of  $A.2$  in Figure 3 is given in Table 1.  $A.2$  knows only domains  $B$  and  $D$  through its direct neighbors. The QoS of the link from  $A.2$  to  $B.1$  is  $pt_{A.2 \rightarrow B.1}$  and this is recorded in row  $B$  column  $B.1$ . Row  $D$  column  $D.1$  is initialized in a similar manner. All entries of column  $A.1$  are (0, 0) because  $A.2$  does not aware of any path leading to  $B$  and  $D$  through  $A.1$ . After initializing the distance table, the routing table can be built as described in Section 4.3.2.

Let the QoS of  $N$  to neighbor  $j$  be  $l_{N \rightarrow j}$ . When  $N$  receives a QoS advertisement  $l$  from  $j$  to domain  $T$ ,  $N$  updates the entry of column  $T$  row  $j$  in the distance table by  $l_{N \rightarrow j} \oplus l$ .  $N$  should also

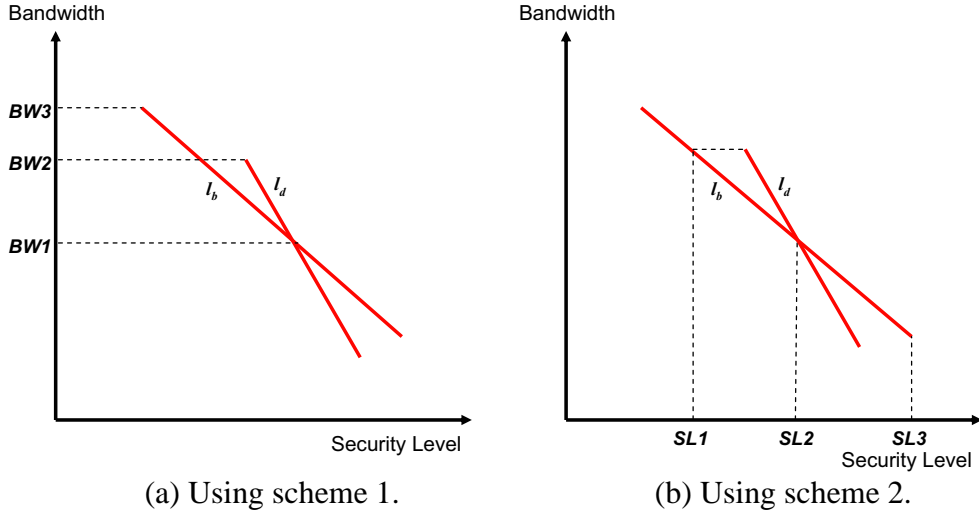


Figure 13: Routing table updates.

check whether this change affects the routing table and update it if necessary.

$N$  should also send its QoS information to its neighbors. Let  $j_1, j_2, \dots, j_m$  be the neighbors of  $N$  and let  $l_k$  be the QoS in column  $j_k$  and row  $T$  of the distance table.  $N$  advertises to  $j_k$  the aggregated QoS from itself to  $T$  through other neighbors except  $j_k$ . This is to prevent the formation of routing loop. Therefore, the aggregated QoS is  $\cup_{1 \leq i \leq m, i \neq j} l_i$  and the mechanism discussed in Section 4.2 is used to find such a single line segment for advertisement. Intuitively,  $N$  should send out an advertisement whenever its distance table changes. However, this may lead to too frequent table updates and is not desirable. We thus study two advertisement suppression mechanisms and they will be described in Section 4.3.3.

### 4.3.2 New Routing Table

With multiple QoS metrics, the routing table is no longer a table with  $u$  rows specifying a definite next hop neighbor for a request with destination  $i$ . The new routing table specifies for each destination, the neighbor that offers the largest bandwidth for different security level values, or the neighbor that offers the largest security level for different bandwidth values, depending on the scheme for the “best” neighbor we have chosen.

Routing Table

Domain	Outgoing Link	
	request bandwidth $req_w$	neighbor
$T$	$0 < req_w \leq BW1$	$B.1$
	$BW1 < req_w \leq BW2$	$D.1$
	$BW2 < req_w \leq BW3$	$B.1$

Table 2: Routing entry for  $T$  of  $A.2$ .

Using our simple network in Figure 3 as an example. When  $A.2$  receives, for destination  $T$ , the QoS lines  $l_b$  from  $B.1$  and  $l_d$  from  $D.1$  respectively, how should  $A.2$  update its routing table entry for destination  $T$ ?

First, assume we implement scheme 1. For each destination, the service outline can be constructed based on the entries on the distance table in which the concave corners and the highest point in the service outline will define the bandwidth ranges. For instance, in Figure 13(a), three ranges are formed. They are  $[0, BW1]$ ,  $(BW1, BW2]$ , and  $(BW2, BW3]$ . For the bandwidth ranges  $[0, BW1]$  and  $(BW2, BW3]$ ,  $B.1$  offers better security level while  $D.1$  is better for the bandwidth range  $(BW1, BW2]$ . The routing entry for domain  $T$  in  $A.2$  based on Figure 13(a) is shown in Table 2. The updates of the routing table are triggered by distance table updates since the routing table depends on the distance table.

Now, assume we implement scheme 2 under the same network topology and parameters. As shown in Figure 13(b), three ranges are also formed. They are  $[0, SL1]$ ,  $(SL1, SL2]$ , and  $(SL2, SL3]$ . For the security level ranges  $[0, SL1]$  and  $(SL2, SL3]$ ,  $B.1$  supports higher bandwidth. However,  $D.1$  in turn supports higher bandwidth for the security level range  $(SL1, SL2]$ . Therefore, similar routing table updates can be carried out.

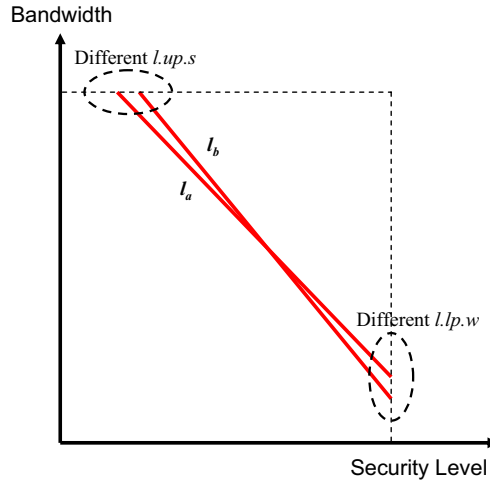


Figure 14: Checking of  $lp.d$ ,  $lp.w$ ,  $up.d$  and  $up.w$  for TC and AHC.

### 4.3.3 Convergence

Distance-vector routing mechanism suffers from the problems of slow convergence and routing to infinity. To ensure the stability of the routing protocol, we restrict the generation of new advertisements. A node should not send an advertisement to its neighbor if this advertisement is *similar* to the previous one it sent earlier. In order to check the similarity of the old and the potential new advertisements, we study two approaches: (1) threshold checking (TC), and (2) advertisement history checking (AHC).

#### 1. Threshold checking:

In this approach, the old and new QoSs are compared by checking the security level and bandwidth of the two endpoints as illustrated in Figure 14. If  $lp.s$  and  $up.w$  of the two QoSs are the “same” (i.e. varying within a very small value like 1%) while their  $lp.w$  and  $up.s$  vary within a certain threshold (in terms of a percentage), the two QoSs are regarded as similar. Let  $l_a$  and  $l_b$  be the two QoS lines and take  $lp.w$  as an example. The checking is performed as:

$$\begin{aligned}
variation &= |l_a.lp.w - l_b.lp.w| \\
allowed\_variation &= \min(l_a.lp.w, l_b.lp.w) \times threshold
\end{aligned} \tag{10}$$

If  $variation \leq allowed\_variation$ , the two QoS lines are said to have similar  $lp.w$ . The thresholds for  $lp.s$  and  $up.w$  are stricter than those on  $lp.w$  and  $up.s$ . It is because the majority of the information distortion introduced by line segment aggregation and join operation is on  $lp.w$  and  $up.s$ . Under this mechanism, when the new QoS for advertisement, which is computed for the same neighbor and destination domain, varies in a small level from the last advertised one, the new QoS is ignored and no new advertisement is generated. To facilitate the checking against with the last advertised QoS, each border node employing this approach has to keep a table, which has exactly the same dimension as that of the distance table, to record the last sent advertisement for every domain-and-neighbor pair.

## 2. Advertisement history checking:

With advertisement history checking, a node checks the newly computed but not yet advertised QoS against the recent history record. To enable this checking, each node maintains an advertisement history for each neighbor and destination pair. When there is any route information update triggered by a new advertisement arrival, a node updates its distance and routing table, and computes the new QoSs preparing for advertisement. Before advertising these newly computed QoSs, a node looks into its history record to see if similar advertisements have been made before. If there are several matchings, say two, found for a QoS in the record, it means that this QoS value has been updated for the third time and most probably routing information oscillation occurs. In this case, the new QoS advertisement is suppressed.

This measure allows a stricter comparison threshold on  $lp.w$  and  $up.s$  than the previous measure due to the use of matching count. By matching the current advertisement with the history table, AHC is able to detect route information oscillation more accurately, but with a tradeoff of longer detection time and larger history table size.

## 5 Simulation

### 5.1 Simulation Overview

In this section, we present the performance evaluation of our protocol obtained from simulation. We evaluated our findings using a self-written C++ network simulator, with the network topologies and costs generated by the BRITE software. The simulator performs both intra-domain and inter-domain routings with two-level hierarchical networks so as to emulate Internet routing. At this stage, we simulate a static network with no link cost changes throughout the route information exchange process. Afterwards, numerous QoS requests are being served. By launching simulations on various network topologies, the accuracy of our protocol in estimating the actual route information and how well it is serving various QoS requests are investigated. In the following sub-sections, we discuss our performance metrics, simulation testbed, and finally the results of our simulation.

### 5.2 Performance Metrics

Three performance metrics are used in our simulation: (1) feasible request ratio, (2) success ratio and (3) crankback ratio.

1. Feasible Request Ratio

In our simulation, intra-domain link, inter-domain link and QoS requests are all within the same integer range. Since both security level and bandwidth are constraint metrics, many

QoS requests are actually infeasible to be served. We partition the generated QoS requests to be feasible or infeasible by running the Dijkstra's algorithm in a flat topology (i.e., without hierarchical structure and with all link costs known to the source node). Then, we define the feasible request ratio as:

$$\text{feasible request ratio} = \frac{\text{total number of feasible requests}}{\text{total number of requests}} \quad (11)$$

## 2. Success Ratio

Success ratio is defined as:

$$\text{success ratio} = \frac{\text{total number of accepted feasible requests}}{\text{total number of feasible requests}} \quad (12)$$

It measures how well our routing protocol is able to serve feasible requests. Referring to Figure 2, requests that fall in the covered area of the staircase (shaded area) are feasible requests and those fall in the covered area by the approximated line are accepted requests. Therefore, requests that fall in the intersection of the two areas are accepted feasible requests.

## 3. Crankback Ratio

The word “crankback” refers to the situation that accepted requests cannot be served successfully. There are two types of crankback situation. The first type of crankbacked requests are infeasible requests that are accepted by our protocol due to line approximations. The second type of crankbacked requests are requests that are feasible, but our protocol finds a routing path which either forms a routing loop or does not fulfill the requested QoS parameters due to route information distortion. Finally, crankback ratio can be defined as:

$$\text{Crankback ratio} = \frac{\text{total number of crankbacked requests}}{\text{total number of accepted requests}} \quad (13)$$

From the above definitions, it is easy to see that a good QoS routing protocol should have high success ratio and low crankback ratio.

## 5.3 Simulation Testbed

### 5.3.1 Networks Generation

We evaluated our protocol described in Section 4 using BRITE topology randomly generated by the BRITE software (version 2.1b Java generator). The BRITE topology provides connectivity to intra-domain and inter-domain nodes. As all links generated by the BRITE software are non-directional, we purposely assigned different bandwidth and security level in the forward and backward directions to evaluate the performance in asymmetrical networks. The link costs are all random integer values in the same range for all links.

In our simulation, we generated networks with 10 domains, each of which contains 50 nodes. The link-state algorithm has been used for intra-domain routing while the asynchronous distance-vector algorithm with modifications as described in Section 4.3 has been used for inter-domain routing. We implemented Scheme 1 in the routing tables as described in Section 4.3.2 onto our network simulator, and then evaluated the performance using threshold checking and advertisement history checking.

Further parameters have to be configured into our self-written network simulator and the BRITE software. Such parameters are summarized in Tables 3 and 4.

### 5.3.2 QoS Request Generation

To evaluate the performance of our protocol in serving the QoS requests, we generated a new set of QoS requests for each network topology under simulation after the route information exchange between nodes has been completed. Each QoS request is a 5-order tuple, which is denoted as  $(SrcDomain, SrcNode, DestDomain, req_s, req_w)$  where  $req_s$  and  $req_w$  are integers in the range

Parameter	Value
Number of Domains	10
Number of Inter-Domain Links	40
Number of QoS Requests	5000
Networks Simulated	10
Domain Size	50 nodes
Link Security Level	$Z^+ \in [5, 10]$
Link Bandwidth	$Z^+ \in [5, 10]$
Advertisement Transmission Delay (Unit)	$R^+ \in (0.0, 2.0]$
TC Thresholds (lp.s, lp.w, up.s, up.w)	(10%, 35%, 35%, 10%)
AHC Thresholds (lp.s, lp.w, up.s, up.w)	(10%, 10%, 10%, 10%)

Table 3: Summary on general simulation parameters.

Parameter	Value
AS Model (inter-domain)	Waxman
Router Model (intra-domain)	Waxman
Size of Main Plane (HS)	1000
Size of Inner Plane (LS)	100
Node Placement	Random
Growth Type	Incremental
alpha	0.15
beta	0.2
Number of Neighbors Connected (m)	2

Table 4: Summary on BRITE parameters.

[5, 9]. Requests with  $req_s$  or  $req_w$  equal 10 were not simulated because such requests will generally lead to infeasible requests. The routing is considered complete when we eventually route a packet to any one of the destination domain's border node.

We generated one QoS request for every possible "source node to destination domain" pair. This contributes to  $50 \times 10 \times 10 = 5000$  QoS requests.

### 5.3.3 QoS Request Serving

Our network simulator reads in and serves QoS requests one by one and all requests are served independently from each other. To facilitate our following discussion, we denote the source node and the destination domain of a request as  $req_{src}$  and  $req_{dst}$ , respectively. The serving process for each QoS request consists of three stages:

1. Request Acceptance Judgement

This stage determines whether or not the QoS request should be accepted by our protocol. It is done by  $req_{src}$ , which checks the QoS of the request against its aggregated QoS to  $req_{dst}$ . If  $qos_a$  is the aggregated QoS line segment from  $req_s$ 's distance table for  $req_{dst}$  and that  $(req_s, req_w)$  falls within the covered region of  $qos_a$ , the request is accepted and the following stages follow. Otherwise, the request is rejected and the latter stages will be skipped.

2. Skeleton Inter-Domain Path Finding

After a request has been accepted, the simulator finds the actual path that our protocol would use to serve the request. The skeleton path finding is done by recursively looking up the next hop border for  $req_w$  from borders' routing tables, starting from  $req_{src}$  until  $req_{dst}$  is reached. The retrieved path is a skeleton path because it does not have its intra-domain route information filled.

3. Intra-Domain Route Filling

This stage fills in the intra-domain route information missed out from the inter-domain path

found in stage 2. For each domain along the skeleton path, intra-domain routing is launched to find the path with the largest security level that serves  $req_w$ . After merging the obtained inter-domain and intra-domain route information, the overall security level  $actual_s$  of the whole path is retrieved. It is then compared with the requested security level  $req_s$ . If  $actual_s \geq req_s$ , the request is said to be served successfully, otherwise it is crankbacked.

However, due to the fact that both QoS metrics are constrictive parameters, exchanging routing information through our line segment approximation approach can create certain amount of routing loops. In light of this, the following approach is adopted: When serving a QoS request for destination domain  $i$ , the packet's AS-path from source border to the current border is carried through as the packet traverses. If a current border, upon scanning the AS-path information, discovers the path of a packet contains the identity of itself, the current border will retrieve the next hop border  $j$  from its routing table, remove the line segment at distance table entry  $(i, j)$ , and then re-calculate the service outline for domain  $i$ . Finally it updates its routing table and continues to route the packet using the updated routing table. With this approach, no routing loop can be formed.

Now, for every QoS request, we record whether it is feasible, accepted and / or crankbacked, and then obtain the performance metrics as described in Section 5.2. The results are presented in the following subsection.

## 5.4 Simulation Results

The simulation results are shown in Figure 15. We simulated 10 different networks using both threshold checking and advertisement history checking modes, and all results shown below are values averaged over 10 domains.

From simulation, the average feasible request ratio is only 40.7%. The value differs if we choose another security level and bandwidth ranges for the QoS requests. The average success ratio is over 90% for both TC and AHC modes, showing that the protocol is good at estimating the

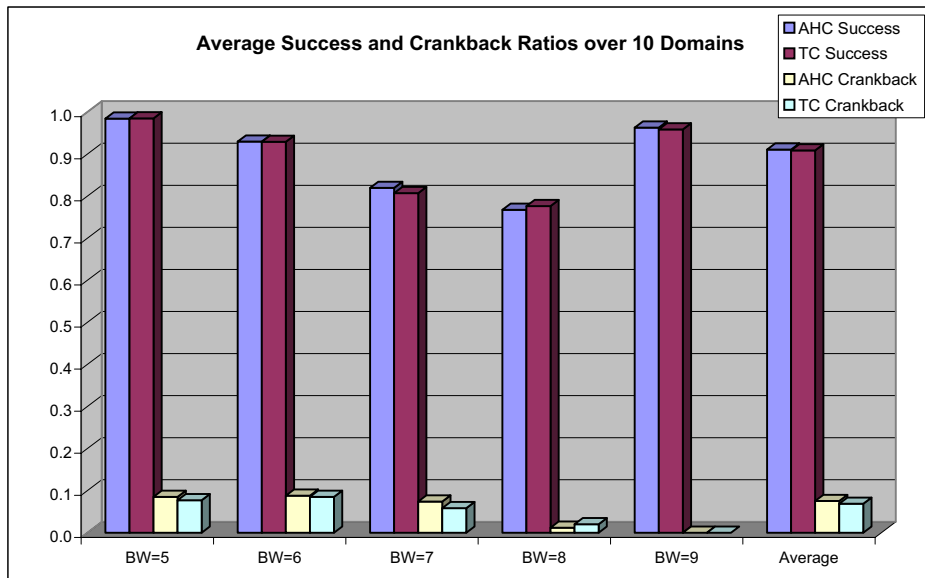


Figure 15: Simulation results for QoS routing with two concave constraints.

actual QoS of the path towards destination. The average crankback ratio is about 7% for both TC and AHC modes, which is formed by inaccuracies introduced in the route information exchange process.

The protocol's convergence time depends greatly on the threshold values chosen to suppress generating new advertisement packets. Define 1 time unit to be the average transmission time of a new advertisement packet. Using the simulation parameters shown in Table 3, the average convergence time for TC and AHC mode are 84 units and 113 units, respectively.

## References

- [1] K.-S. Lui, K. Nahrstedt, S. Chen, Routing with topology aggregation in delay-bandwidth sensitive networks, *IEEE/ACM Trans. Netw.* 12 (1) (2004) 17–29.
- [2] W.-Y. Tam, K.-S. Lui, S. Uludag, K. Nahrstedt, Quality-of-service routing with path information aggregation, *Elsevier Computer Networks Journal* 51 (12) (2007) 3574–3594.