Journal of the Operations Research
Society of Japan
2007, Vol. 50, No. 4, 428-443

# MINIMUM UNDERFLOW ROUTING FOR STREAM-TYPE COMMUNICATION SERVICES

Masaki Aida Chisa Takano Shunichi Osawa
*Tokyo Metropolitan University*

(Received October 31, 2006; Revised July 12, 2007)

*Abstract* The Internet of today supports various types of communication services including not only conventional data communication services but also stream-type communication services. Typical examples of stream-type services include interactive voice such as Voice over IP (VoIP) and live video delivery. Stream services require real-time packet transmission, and the continuous playback of packets at the receiver side. As a result, they are not only sensitive to the absolute value of the delay, but also sensitive to delay variations. This paper addresses the problem of optimal routing for stream-type communication services. Optimality is discussed in terms of the continuous playback of packets. From network observations, we assume that the end-to-end delay statistics conform to a normal distribution. We model a network as a weighted graph with its link weights representing link delays. In the course of the analysis, we show that this type of routing optimization problem can be formulated as a process of searching for a specific point in a coordinate system defined by the mean and variance of the end-to-end delay. This paper presents an efficient algorithm for finding the optimal point in this coordinate system.

**Keywords**: Telecommunication, routing, underflow, stream traffic, VoIP

## 1. Introduction

The Internet of today has been regarded as a multimedia network that supports various types of communication services. In particular, stream-type communication services, including Voice over IP (VoIP) and live video delivery, are very different from conventional data communication services. Stream-type services require real-time packet transmission, and the continuous playback of packets at the receiver side.

In the past, various ways have been offered for routing in the Internet or, in general, packet switched networks. The main goal is to minimize the summation of a certain quantity associated with each link along a path, for example, the number of hops or the mean value of end-to-end delay across the network [6]. This class of problem is the shortest path problem and is quite solvable. It can be efficiently solved by applying the well-known Dijkstra's algorithm [3].

Associated with each stream-type communication service, there exists the maximum permissible delay (or allowable delay) that should be guaranteed to obtain satisfactory quality of service (QoS) [1, 4]. Let us take interactive voice communication as an example [see Figure 1]. Voice packets received by the listener side terminal are stored temporarily in a buffer before they are played back, to smooth out the packet delay jitter. The buffer is called the playout buffer. Playback is performed under the control of a continuous clock source. The required amount of storage must be large enough to make the probability of buffer overflow/underflow reasonably small — a situation that may occur with excessive packet delay variation. Furthermore, packets that experience delays longer than a certain

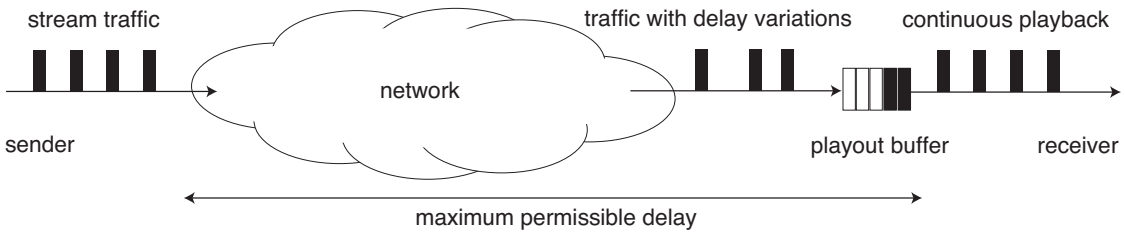


Figure 1: System model

value will be too late for play-back and must be discarded at the receiving buffer.

Thus, to obtain a satisfactory service quality, the routing should be designed to take the distribution of delay into account. In this paper, we investigate end-to-end delay distribution by means of observations of an actual network (Sec. 2). Based on these observations, we formulate the routing problem as the minimization of the probability that packets exceed the maximum permissible delay (Sec. 3). We model a network as a graph with its link weights representing delays and assume that the end-to-end delay statistics follow a normal distribution. Our problem is regarded as a stochastic version of the shortest path problem on a weighted graph. This class of problem requires a different formulation to that which may be used for optimal routing based simply on minimization of mean delay. As will be described later, Dijkstra's algorithm cannot be applied directly. Ichimori et.al.[5] consider a related problem which can be dealt with using a minimum spanning tree. We will show that the treatment proposed in their paper can be efficiently used as one tool to solve our problem (Sec. 4). However, their treatment is not sufficient to solve the stochastic shortest path problem by itself. A shortest path problem is a more difficult class of problem than the minimum spanning tree problem and requires some extensions. Next, we show that the optimization of routing in our problem is equivalent to searching for a specific point in a coordinate system determined by the mean and variance of the end-to-end delay (Sec. 5). Using this formulation, we construct a simple and efficient algorithm for solving the problem (Sec. 6) and show that the residual estimation error is bounded (Sec. 7). Finally, we describe the simulation results (Sec. 8).

## 2. Measurements of End-to-End One-Way Delay

### 2.1. Measured data

We conducted observations of the end-to-end delay of packets by using a real network. The path from the PC of an asymmetric digital subscriber line (ADSL) customer to another PC in a company's LAN was used in the experiments, where both PCs were GPS-synchronized.

The LAN for the ADSL customer and the company's LAN were connected via two ISPs and the path between the LANs consisted of 15 hops. 64-byte UDP packets were sent from the ADSL customer to the PC in the company's LAN according to a Poisson process and their one-way end-to-end delays were monitored for 900 seconds for each experiment period. The experiment was conducted on a weekday in December 2003.

Five typical results are listed in Table 1. The load of the network was light in the first two cases and heavy in the last two cases. Measurements #1 and #2 were conducted at night, and #4 and #5 were in busy hours. Measurement #3 was the result when there was a medium load on the network.

Table 1: Results of measurement of end-to-end one-way delay (ms)

| reference ID | measurement period | mean delay | min. delay | max. delay | usage of network |
|---|---|---|---|---|---|
| #1 | 00:00–00:15 | 5.319 | 4.942 | 11.948 | light |
| #2 | 02:30–02:45 | 5.279 | 4.944 | 9.948 | light |
| #3 | 10:00–10:15 | 42.258 | 4.967 | 138.136 | medium |
| #4 | 10:45–11:00 | 130.590 | 28.516 | 187.936 | heavy |
| #5 | 11:45–12:00 | 116.767 | 27.746 | 191.342 | heavy |

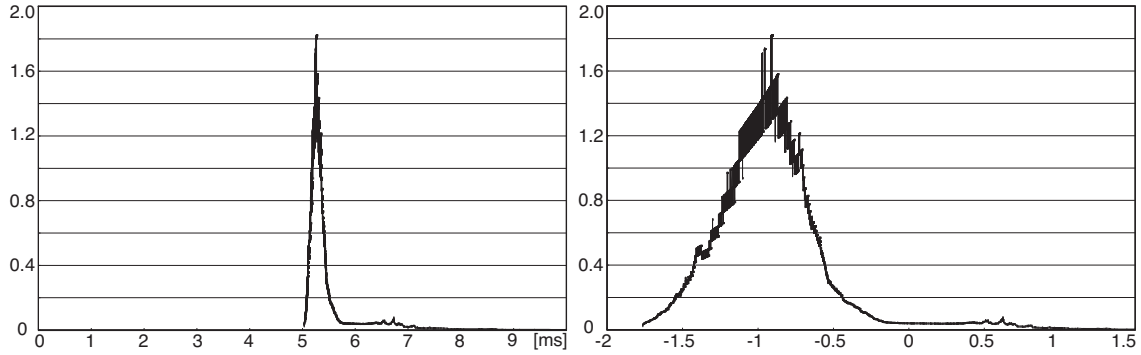


Figure 2: Distributions of end-to-end one-way delay for measurement #1 (left: distribution of measured delay on a linear scale, right: distribution for the logarithm of variable component of delay)

### 2.2. Observed distribution of end-to-end one-way delay

First, we investigate the cases where the network load is light. The left-hand graph of Figure 2 shows a distribution of the measured delay for measurement #1. We subtracted 4.87 ms as the constant component of the delay and show the histogram for the logarithm of residual component of the delay in the right-hand graph of Figure 2. The right-hand graph implies end-to-end delay consisting of a constant component and a variable component that follows a lognormal distribution. Figure 3 shows a comparison between the measured, empirical distributions of the variable component of end-to-end one-way delay for measurements #1 and #2, and lognormal distributions. These results imply that the variable component of end-to-end delay follows a lognormal distribution when the network load is light.

Next, we examine the cases where the network load is heavy. Figure 4 shows a comparison between the empirical distributions of end-to-end one-way delay for measurements #4 and #5, and normal distributions. These results imply that the end-to-end delay follows a normal distribution when the network load is heavy.

Finally, we consider the case where the load of the network is medium. Figure 5 shows a comparison between the empirical distribution of the variable component of end-to-end one-way delay for measurement #3, and theoretical distributions, normal and lognormal. From this result, we cannot determine the distribution of the end-to-end delay when there is a medium load on the network.

Table 2 summarizes the above results. Since controlling the QoS of streaming services is important when the network is congested, we should consider the situation where the load of the network is high. So, we assume the end-to-end delay follows a normal distribution.

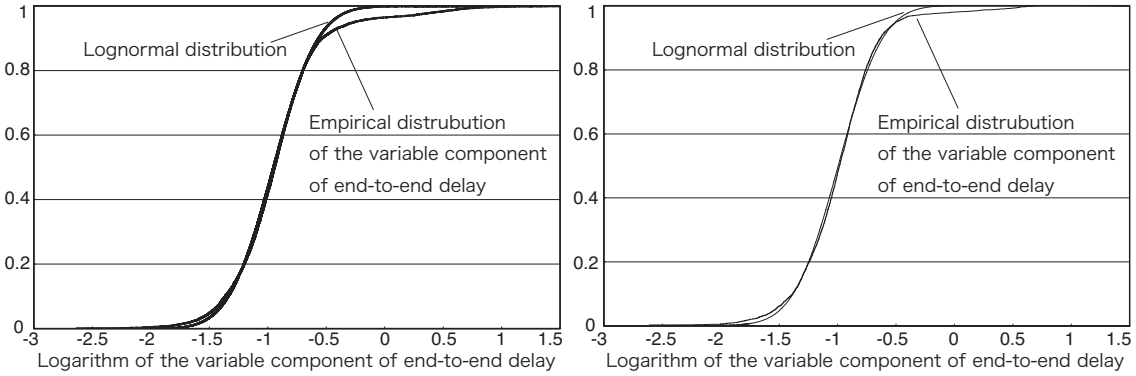


Figure 3: Comparisons between empirical distributions of the variable component of end-to-end one-way delay and lognormal distributions (left: measurement #1, right: measurement #2)

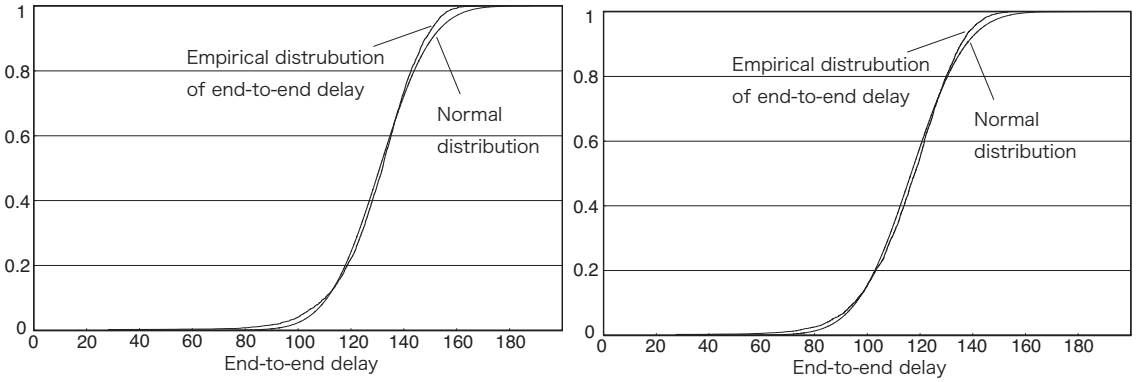


Figure 4: Comparisons between empirical distributions of end-to-end one-way delay and normal distributions (left: measurement #4, right: measurement #5)

Even if the assumption is not appropriate in all cases, we can assume that a light load on the network does not have a crucial effect on the QoS.

## 3. Optimal Routing Problem

We represent a network as a graph $G = G(V, E)$ where $V$ is the set of nodes and $E$ is the set of links connecting the nodes. Each link denoted by $i$ has a weight $d(i)$ that represents the delay experienced by packets on the link. In general, the value of $d(i)$ fluctuates dynamically. We assume that $d(i)$ is an independent random variable characterized by its mean $m(i)$ and variance $v(i)$. We further assume that the mean $m(i)$ and variance $v(i)$ are stable over time. We refer to $d(i)$ as a stochastic weight. We treat a shortest path problem between given two nodes in the following part of this paper.

Table 2: Distributions of end-to-end one-way delay

| load of the network | distribution |
|---|---|
| light | constant + lognormal |
| medium | ? |
| heavy | normal |

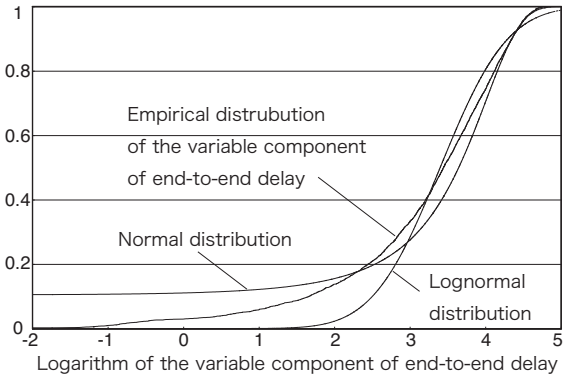


Figure 5: Comparisons between empirical distributions of end-to-end one-way delay and normal distributions (for measurement #3)

Before considering the shortest path problem between two given nodes, we briefly add supplemental comments about the assumptions of independence and stationarity of $d(i)$. We do not assume stability over a long time period; we are interested only in the limited period of the busy hour(s). Also in traditional telephone networks, stochastic properties of traffic are assumed to be stable over a limited period even if they change dramatically over a longer time period. We also assume independence of flows in the busy hour(s). In this period, when there is a high volume of traffic, a single flow has a relatively small impact on the total amount of traffic. Even if the assumption of independence is not true for link-by-link delay statistics, then we can reconsider the link delay as the delay of a segment of a network that is of a relatively large scale, e.g. the delay of each Internet Service Provider (ISP) network. This is because it is not easy to see how there could be short time dependence of stochastic properties between different large scale subnetworks.

Let $\pi$ denote a path between two given nodes in $G$ and $\Pi = \{\pi\}$ be the set of all paths between the two given nodes. Given any path $\pi$, based on the independence assumption, the path weight $D(\pi)$, its mean $M(\pi)$, variance $V(\pi)$, and standard deviation $S(\pi)$ can be expressed as:

$$\begin{aligned} D(\pi) &= \sum_{i\in\pi} d(i), \quad M(\pi) = \sum_{i\in\pi} m(i), \\ V(\pi) &= \sum_{i\in\pi} v(i), \quad \text{and} \quad S(\pi) = \sqrt{V(\pi)}. \end{aligned} \tag{3.1}$$

Here, we assume the end-to-end delay $D(\pi)$ follows a normal distribution

$$D(\pi) \sim N(M(\pi),\, V(\pi)). \tag{3.2}$$

Conventionally, routing problems have been formulated in terms of finding a particular path $\pi$ between two nodes from the set of paths $\Pi$ that minimizes the mean end-to-end delay. In this formulation, the problem is expressed in the form:

$$\min_{\pi\in\Pi} M(\pi). \tag{3.3}$$

This problem depends only upon the mean value of delay because (3.3) does not take the variance of delay into consideration. Previous studies show that this type of problem of finding the shortest path can be solved efficiently by applying dynamic programming techniques; e.g. Dijkstra's algorithm.

In contrast, our aim is to take the variance of delay into consideration, so we formulate our minimum underflow routing problem as follows:

$$\min_{\pi\in\Pi} \mathrm{P}\{D(\pi) > D_0\}, \tag{3.4}$$

where $D_0$ is a given positive real constant representing the maximum permissible delay. The purpose of our optimization is to minimize the probability that the end-to-end delay exceeds a predefined value. Here, we assume that $D_0$ satisfies

$$D_0 \geq \min_{\pi\in\Pi} M(\pi), \tag{3.5}$$

by considering real applications to communication networks. By standardization of $D(\pi)$, we can obtain an equivalent formulation:

$$\max_{\pi\in\Pi} \frac{D_0 - M(\pi)}{S(\pi)}. \tag{3.6}$$

Hereafter, we call the function to be maximized the "Evaluation function" of the path $\pi$:

$$E(\pi,\, D_0) := \frac{D_0 - M(\pi)}{S(\pi)}. \tag{3.7}$$

## 4. Application of Dynamic Programing

Unfortunately, the evaluation function $E(\pi, D_0)$ still has properties that make it hard to treat. This section explains what this difficulty is and how it can be overcome.

Imagine a situation where there are three nodes ($A$, $B$, and $C$) which are connected by three links as shown in Figure 6. Each link and each path have weights that are evaluated by $E(\pi, D_0)$. We further assume that link $c$ is the optimal path between nodes $B$ and $C$. It can easily be shown that, if the evaluation function $E(\pi, D_0)$ is used to evaluate the path/link weights, the optimal path between node $A$ and node $C$ does not always include link $c$ which is the optimal path between node $B$ and node $C$. We show an example using Figure 6. We assume that link $a$ has $(m(a),\, v(a)) = (1,\, 1)$. Similarly, links $b$ and $c$ have $(m(b),\, v(b)) = (7,\, 4)$ and $(m(c),\, v(c)) = (8,\, 1)$, respectively. In addition, we set $D_0 = 10$. Then, the values of the evaluation function of the paths between nodes $B$ and $C$ are

$$E(b,\, 10) = \frac{10-7}{\sqrt{4}} = 1.5, \quad \text{and} \quad E(c,\, 10) = \frac{10-8}{\sqrt{1}} = 2,$$

respectively. Since $E(b,\, 10) < E(c,\, 10)$, link $c$ is better. On the other hand, values of the evaluation function of the paths between nodes $A$ and $C$ are

$$E(a \to b,\, 10) = \frac{10-(1+7)}{\sqrt{1+4}} \simeq 0.894, \quad \text{and} \quad E(a \to c,\, 10) = \frac{10-(1+8)}{\sqrt{1+1}} = 0.707,$$

respectively. Since $E(a \to b,\, 10) > E(a \to c,\, 10)$, the best path between nodes $A$ and $C$ includes link $b$. It means that the principle of optimality [2], which is the basis for applying dynamic programming techniques, is not available when this evaluation function is used.

Hence, the well-known, Dijkstra's algorithm cannot be applied directly to our problem because the algorithm employs the optimality principle. To avoid the above difficulty, we introduce a new measure for the link and path weights. The following approach follows

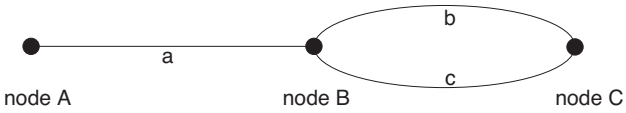


Figure 6: Example of network

Ichimori et.al. [5] whose work deals with the stochastic minimum spanning tree problem. This technique is also effective in solving our problem. We introduce a positive parameter $x$ (which has the dimension of 1/(time)) and define the weight of the individual link $i$ as:

$$w_x(i) := m(i) + x \cdot v(i). \tag{4.1}$$

Following this definition, the $x$-weight of a path $\pi$ can be expressed as:

$$W_x(\pi) = M(\pi) + x \cdot V(\pi). \tag{4.2}$$

With this formulation of link and path weights, an end-to-end path weight can be obtained by simply summing up the weights of all links that comprise the path, i.e.,

$$W_x(\pi) = \sum_{i \in \pi} m(i) + x \sum_{i \in \pi} v(i) = \sum_{i \in \pi} w_x(i). \tag{4.3}$$

Hereafter, we call $w_x(i)$ the $x$-weight of link $i$.

Next, we define the concept of the $x$-optimal path between a given pair of nodes. A path $\pi_{x\text{-opt}}(x)$ is $x$-optimal if its $x$-weight satisfied the following minimization for a given value of $x \geq 0$.

$$\min_{\pi \in \Pi} W_x(\pi). \tag{4.4}$$

In general, the configuration of the path $\pi_{x\text{-opt}}$ varies with the value of $x$. Let $\Pi_{x\text{-opt}}$ denote a set of such optimal paths and let $\pi_{\text{opt}}$ denote the real optimal path that satisfied the original problem (3.6).

The $x$-weight thus defined has a useful characteristic which allows our problem to be solved. We show this in the following theorem.

**Theorem 4.1** *The true optimal path $\pi_{\text{opt}}$ is included in the set $\Pi_{x\text{-opt}}$ of $x$-optimal paths.*

$$\pi_{\text{opt}} \in \Pi_{x\text{-opt}}. \tag{4.5}$$

**Proof**

We prove this by contradiction. Suppose the real optimal path $\pi_{\text{opt}}$ is not included in the set $\Pi_{x\text{-opt}}$. We then have

$$\begin{aligned} D_0 - M(\pi_{\text{opt}}) &= E(\pi_{\text{opt}},\, D_0)\, S(\pi_{\text{opt}}), \\ D_0 - M(\pi_{x\text{-opt}}(x)) &< E(\pi_{\text{opt}},\, D_0)\, S(\pi_{x\text{-opt}}(x)), \end{aligned}$$

from (3.7), which gives the following inequality.

$$M(\pi_{x\text{-opt}}(x)) - M(\pi_{\text{opt}}) > E(\pi_{\text{opt}},\, D_0)\, \left(S(\pi_{\text{opt}}) - S(\pi_{x\text{-opt}}(x))\right). \tag{4.6}$$

If we arbitrarily select the value of $x$ as $x' = E(\pi_{\text{opt}}, D_0)/(2S(\pi_{\text{opt}}))$, there should exist a corresponding $x$-optimal path $\pi_{x\text{-opt}}(x')$ in $\Pi_{x\text{-opt}}$ which should satisfy

$$W_x(\pi_{x\text{-opt}}(x')) < W_x(\pi_{\text{opt}}). \tag{4.7}$$

However, it follows that

$$\begin{aligned} &W_x(\pi_{x\text{-opt}}(x')) - W_x(\pi_{\text{opt}}) \\ &= M(\pi_{x\text{-opt}}(x')) + x' \cdot V(\pi_{x\text{-opt}}(x')) - (M(\pi_{\text{opt}}) - x' \cdot V(\pi_{\text{opt}})) \\ &> \frac{E(\pi_{\text{opt}}, D_0)}{2S(\pi_{\text{opt}})}(S(\pi_{\text{opt}}) - S(\pi_{x\text{-opt}}(x')))^2 \geq 0. \end{aligned} \tag{4.8}$$

Thus, a contradiction is derived. ■

This theorem enables us to take the following steps to solve our problem: first, we solve for the set of $x$-optimal paths for the $x$-weight problem to which the optimality principle holds and can be solved efficiently using Dijkstra's algorithm; then we search for the true optimal path from among the $x$-optimal paths.

## 5. Representation of Paths on $M$-$V$ Plane

The main difficulty with the above derived approach is how to find out the optimal path from all the $x$-optimal paths for a given graph. To overcome this difficulty, we introduce, in this section, a new coordinate system to analyze our problem. This offers a useful means of systematically searching for the x-optimal paths and the true optimal path.

Specifically, our problem is formulated as searching for a specific point in the plane.

In the following, we show how all $x$-optimal paths are represented as points in the plane. We then examine the geometrical properties of the points in the plane that can be used to shrink the domain where the target point should be searched for.

The $x$-weight defined by (4.1) is a linear function of $x$. In general, any linear function is completely determined by specifying two parameters, i.e., its gradient and the point of intersection with the vertical axis. Here, we introduce a plane that is defined by the $M$-axis and $V$-axis, representing, respectively, the intersection and the gradient of the $x$-weight function, which in turn represent the mean delay of the corresponding path and its variance. On this plane, referred to as the $M$-$V$ plane, each path $\pi$ is represented by a single point $P = (M(\pi), V(\pi))$ as shown in Figure 7.

Next let us examine the contours of the original evaluation function, (3.7), on the $M$-$V$ plane. For a given value $E$ of the evaluation function, the corresponding contour is expressed as

$$V = \frac{1}{E^2}(D_0 - M)^2. \tag{5.1}$$

The resulting shape of the contour is parabolic. Several examples of contours, for different values of $E$, are shown in Figure 7.

Following the above formulation, our purpose can be restated as finding the point with the largest value of $E$. The point is expected to be found towards the lower left corner on the $M$-$V$ plane.

To systematize this searching process, we first examine the basic relationship between two points, $P_1$ and $P_2$, on the $M$-$V$ plane with the following two lemmas. In the following discussion, a path $\pi_i$ is represented by the point $P_i$ whose Cartesian coordinates are given by $(M_i, V_i)$, where $M_i = M(\pi_i)$ and $V_i = V(\pi_i)$.

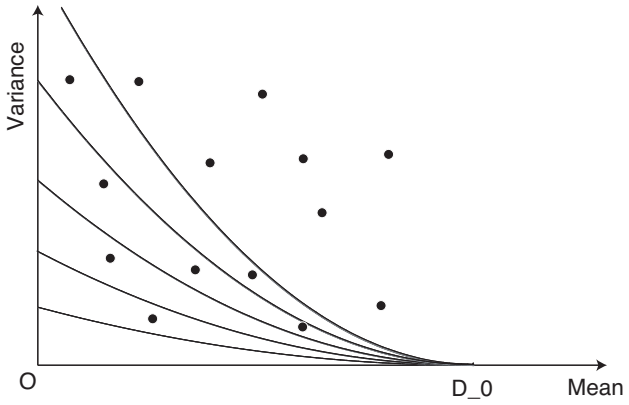


Figure 7: Distribution of paths and contours for evaluation function

**Lemma 5.1** *Let $\pi_1$ and $\pi_2$ be different paths belonging to $\Pi_{x\text{-opt}}$. Let points $P_1$ and $P_2$ on the $M$-$V$ plane represent the paths $\pi_1$ and $\pi_2$ respectively. If there is some $x = x_0 > 0$ that satisfies*

$$W_x(\pi_1) = W_x(\pi_2) \quad \text{for} \quad x = x_0, \tag{5.2}$$

*then the gradient of the line segment $P_1P_2$ is negative and its value is $-(1/x_0)$.*

**Proof**

By applying (4.2) to (5.2), $x_0$ may be calculated as

$$x_0 = -\frac{M_1 - M_2}{V_1 - V_2}. \tag{5.3}$$

Therefore the gradient of the line segment $P_1P_2$ is:

$$\text{gradient} = \frac{V_1 - V_2}{M_1 - M_2} = -\frac{1}{x_0} < 0. \tag{5.4}$$

■

**Lemma 5.2** *Let $\pi_1$ and $\pi_2$ be different paths belonging to $\Pi_{x\text{-opt}}$, which are the $x$-optimal paths for $x = x_1$ and $x = x_2$, respectively. If $x_1 < x_2$, there is some $x = x_0$ that satisfies*

$$W_x(\pi_1) = W_x(\pi_2), \quad (x = x_0), \tag{5.5}$$

*and*

$$x_1 < x_0 < x_2. \tag{5.6}$$

**Proof**

From the assumption, we have

$$\begin{aligned} &W_x(\pi_1) < W_x(\pi_2), \quad (x = x_1), \\ &W_x(\pi_1) > W_x(\pi_2), \quad (x = x_2). \end{aligned} \tag{5.7}$$

Since both $W_x(\pi_1)$ and $W_x(\pi_2)$ are linear functions of $x$, the two lines intersect, so there is some value of $x$ such that $W_x(\pi_1) = W_x(\pi_2)$, where $x_1 < x < x_2$. ■

We next examine the geometrical distribution of points $P_i$, representing the collection of $x$-optimal paths, over the $M$-$V$ plane. This distribution has an outstanding property described by the following theorem. Before entering into details, we should mention some properties that can be understood intuitively:

1. There exists only a finite number of $x$-optimal points.
2. Each point presents $x$-optimality for a certain range of $x$, i.e., for $x_i < x < x_{i+1}$. So, supposing that the complete set of $x$-optimal paths has been obtained, we can number the points such that $P_i$ is optimal for $x_i < x < x_{i+1}$, where $i = 1, 2 \ldots$

**Theorem 5.1** *Suppose that the complete set of the $x$-optimal points has been obtained and they are numbered such that a point representing the $x$-optimality with a larger value of $x$ is given a greater number. Connect these $P_i$ on the $M$-$V$ plane by straight lines in ascending order and the resulting shape is convex and monotonically decreasing [see Figure 8].*

**Proof**

Let $P_i$, $P_j$ and $P_k$ be different points on the $M$-$V$ plane, where they are determined to be $x$-optimal for the value $x$ of $x_i$, $x_j$ and $x_k$ respectively, and $0 < x_i < x_j < x_k$. Let gradient$(P_i, P_j)$ and gradient$(P_j, P_k)$ denote the gradients of line segments $P_iP_j$ and $P_jP_k$, respectively. Because these points correspond to paths in $\Pi_{x\text{-opt}}$, we can always find some real values $a$, $b$ for $x$ such that

$$\begin{aligned} W_x(\pi_i) &= W_x(\pi_j), \quad (x = a), \\ W_x(\pi_j) &= W_x(\pi_k), \quad (x = b). \end{aligned} \tag{5.8}$$

For these to be satisfied, from lemma 5.2, $a$ and $b$ must satisfy the relationship:

$$0 < x_i < a < x_j < b < x_k. \tag{5.9}$$

From lemma 5.1, we have

$$\text{gradient}(P_i, P_j) = -\frac{1}{a}, \quad \text{gradient}(P_j, P_k) = -\frac{1}{b}, \tag{5.10}$$

hence

$$\text{gradient}(P_i, P_j) < \text{gradient}(P_j, P_k) < 0. \tag{5.11}$$

That is, the gradient is less steep on the right side of the connected lines. This property holds true for any combination of $x$-optimal points and hence the shape is convex. ■

Using the above results, we now develop an algorithm that locates the optimal point in the $M$-$V$ plane. Our algorithm starts with a known set of $x$-optimal points on the plane. The main steps carried out in the algorithm consist of:

1. determining regions in which the candidate points are to be searched for,
2. determining the region in which the search process is to be applied first, and
3. shrinking the search regions based on the newly obtained $x$-optimal points.

Assume that $x$-optimal points $P_i$, $P_j$, $P_k$ and $P_l$ are known on the $M$-$V$ plane, where $x_i < x_j < x_k < x_l$. We consider the problem of searching for a new point $P_n$ for $x_j < x_n < x_k$. The domain where $P_n$ should be searched for is a triangle [see Figure 9] and its location is determined in the following manner. Two vertices of its triangular domain are found trivially to be $P_j$ and $P_k$. We define the new vertex $Q_{jk}$ as the intersection point of two extended lines, $P_iP_j$ and $P_kP_l$. The reason for this is easily understood by considering the fact that if $P_n$ is to exist for $x_j < x_n < x_k$, the connecting curve $P_jP_nP_k$ must be convex as dictated by theorem 5.1.

Within this triangle, the candidate point that has the potential to offer the highest value in terms of the evaluation function (3.7) is $Q_{jk}$. If the evaluation with $Q_{jk}$ gives a larger value than with the other two points, $Q_{ij}$ and $Q_{kl}$, the region is worth searching first.

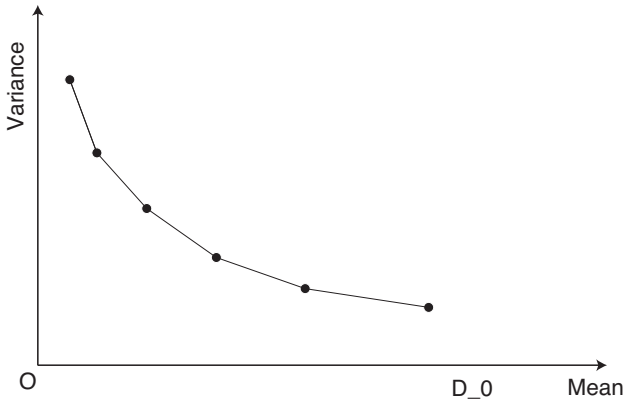


Figure 8: Distribution of $x$-optimal paths

For a given set of known points, we can identify as many triangles as there are line segments. Each newly found vertex must have a corresponding value of the Evaluation function. The triangle with the new vertex which is evaluated to have the largest value is the most promising place to search for a new $x$-optimal point, and this point is expected to yield the greatest value of the Evaluation function.

Once we find a new $x$-optimal point, we can restart the algorithm with the increased number of known points. In this case, following the same reasoning as the above, we can shrink the search domain as shown in Figure 10. Note that finding a single point results in the shrinkage of not only the original triangle but also the two adjacent triangles. As the cycles of the algorithm are repeated, the search domain continues to shrink until the point with the highest evaluation, or the true optimal point, is determined.

For the given triangle $(P_j, P_k, Q_{jk})$, we can find a new $x$-optimal point in the following manner. We use

$$x = -\frac{1}{\text{gradient}(P_j, P_k)} \tag{5.12}$$

to solve for the $x$-optimal path by applying Dijkstra's algorithm. The justification for this is given in Appendix A. If we can find a new path other that those corresponding to $P_j$ or $P_k$, this would be the one with the highest evaluation value. Otherwise, there are no $x$-optimal points in this region, and this region should be excluded from further investigation.

## 6. Formal Description of Algorithm

In this section, based upon the previous discussion, we introduce a formal description of our algorithm. As seen in Figures 9 and 10, each search region forms a triangle, two of the vertices or which are $x$-optimal paths and the other is the point that gives the upper-bound of the evaluation $E(\pi, D_0)$ in the triangle. In the following algorithm, $E_{\text{opt}}$ takes the value of the largest evaluation $E(\pi, D_0)$ among $x$-optimal paths that have already been searched for, and $E_{\text{ub}}$ takes the value of the upper-bound of the evaluation $E(\pi, D_0)$ for all $x$-optimal paths (the latter case means effectively finding the shortest path using weight $v(i)$ only). The algorithm is as follows:

**Step 1** Given two nodes on $G$, find the initial $x$-optimal paths between them, first with $x$ set to 0 and then with $x$ set to a very large value (this means that finding the shortest paths using weight $v(i)$ only). The path finding processes can be done using Dijkstra's algorithm. Calculate the corresponding evaluations $E(\pi, D_0)$ by applying (3.7) and set

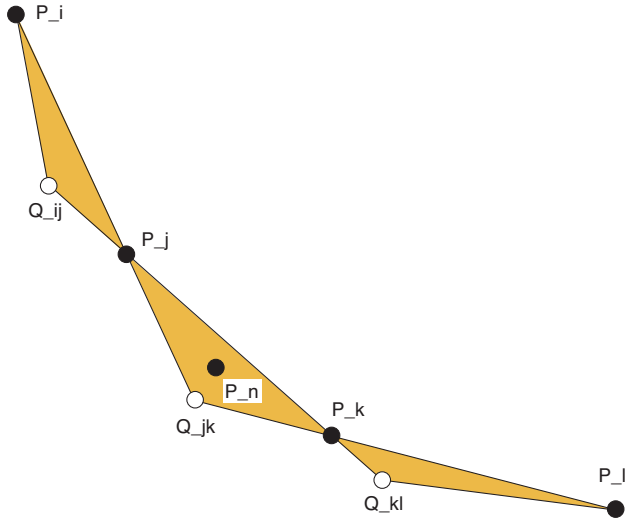


Figure 9: Search domain I

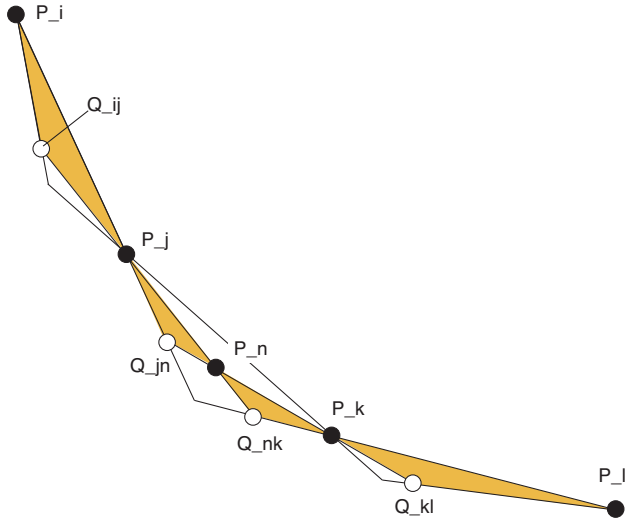


Figure 10: Search domain II

$E_{\mathrm{opt}}$ to the larger value. In addition, set $E_{\mathrm{ub}}$ to the largest evaluation for unsearched paths.

**Step 2** Determine search domains from a set of known $x$-optimal points, and evaluate $E(\pi, D_0)$ for each new vertex $Q_{jk}$. For the domain $(P_j, P_k, Q_{jk})$ with the highest evaluation, update $E_{\mathrm{ub}}$, and find a new $x$-optimal path applying Dijkstra's algorithm with $x$ set to $-(1/\mathrm{gradient}(P_j, P_k))$. If there remains no candidate domain, go to step 4.

**Step 3** If a new path exists, we plot the corresponding point $P_n$ on the $M$-$V$ plane. Calculate $E(\pi, D_0)$ for the new point $P_n$ and if it is larger than $E_{\mathrm{opt}}$, update $E_{\mathrm{opt}}$ to this value. Exclude from the candidate list those domains whose evaluation values are smaller than $E_{\mathrm{opt}}$. Otherwise no new path can be found, and we exclude the domain from the search domain list. In both cases, go back to step 2.

**Step 4** Terminate the algorithm.

Figure 11 depicts the initial steps of the algorithm. The left-hand figure shows the $M$-$V$ plane for Step. 1 after two points have been calculated. If a third point is found in the triangle triangle on the left, the new search region is restricted as shown in the right-hand

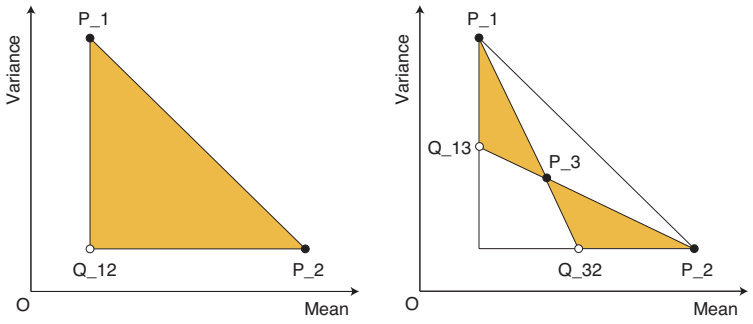


Figure 11: Initial steps in algorithm

figure.

## 7. Residual Error in the Algorithms

The main strength of our algorithm is that we can control the residual error caused by premature termination of the algorithm. Let us consider the following two probabilities, that end-to-end delay exceeds the maximum permissible delay;

$$p_{\text{opt}} = \frac{1}{\sqrt{2\pi}} \int_{E_{\text{opt}}}^{+\infty} \exp\left(-\frac{x^2}{2}\right) \mathrm{d}x, \tag{7.1}$$

$$p_{\text{ub}} = \frac{1}{\sqrt{2\pi}} \int_{E_{\text{ub}}}^{+\infty} \exp\left(-\frac{x^2}{2}\right) \mathrm{d}x, \tag{7.2}$$

where $E_{\text{opt}}$ is current optimal value and $E_{\text{ub}}$ is the upper-bound of the optimum value obtained from the algorithm. We define the error as the difference of the above probabilities, $(p_{\text{opt}} - p_{\text{ub}})$. The residual error can be calculated as:

$$\begin{aligned}(\text{the maximum residual error}) &< p_{\text{opt}} - p_{\text{ub}} \\ &= \frac{1}{\sqrt{2\pi}} \int_{E_{\text{opt}}}^{E_{\text{ub}}} \exp\left(-\frac{x^2}{2}\right) \mathrm{d}x. \end{aligned} \tag{7.3}$$

As the algorithm iterates, $E_{\text{ub}}$ is decreased and $E_{\text{opt}}$ is increased (or remains unchanged). Therefore the error decreases with each cycle in the algorithm and so we can stop the algorithm when the error is sufficiently small.

## 8. Simulation Results

This section describes the results of a simulation for investigating the efficiency of the minimum underflow routing algorithm and the number of iterations required for our algorithm derived in Sec. 6. Figure 12 shows the network model used in the simulation. The model has 100 nodes and they are configured as a $10 \times 10$ lattice topology. We are searching for the communication path is searched from the sender node 00 to the receiver node 99. The minimum number of hops is 18, which is comparable in size to a real network environment.

Each link has two different weights that correspond to the mean and variance of the link delay. The values of the weights are set in advance as independent uniform random numbers distributed in the range (0, 1).

We choose the maximum permissible delay as $D_0 = 12$, 13, or 14. For each value of $D_0$, we have conducted path searches 100 times and compared the paths obtained from

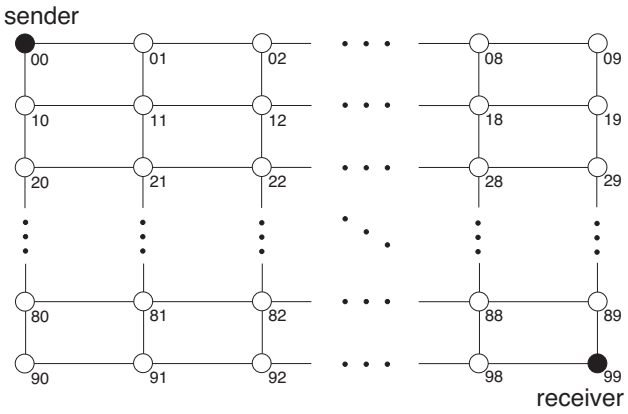


Figure 12: Simulation model

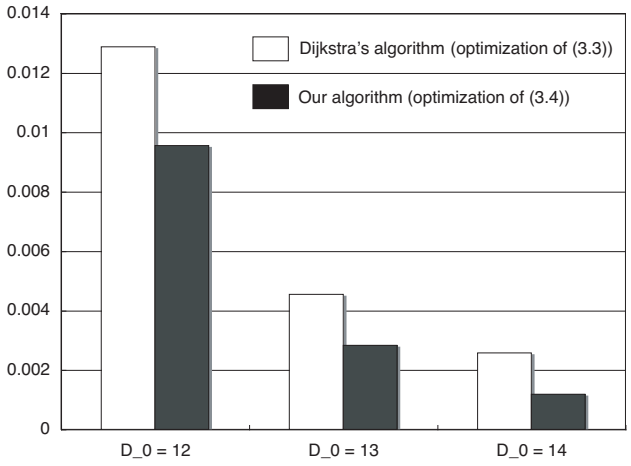


Figure 13: Simulation results: Probability that the end-to-end delay exceeds $D_0$

optimizations (3.3) and (3.4). First, Figure 13 sshows a comparison of the probabilities that the end-to-end delay exceeds the maximum permissible delay $D_0$. The probability values were obtained from the average of 100 path searches. From this figure it may be seen that the minimum underflow routing algorithm is most effective for larger values of $D_0$.

In addition, Table 3 shows the number of iterations, which were required to search for the optimal path using our algorithm. Here, the number of iterations means the number of applications of Dijkstra's algorithm before the optimal path is determined. The average number of iterations is around 6, and the maximum is 9. From these results, we can expect the algorithm to converge fairly rapidly in practical networks.

Table 3: Number of iterations

| maximum permissible delay | number of measurements | number of iterations | | |
|---|---|---|---|---|
| | | average | max. | min. |
| $D_0 = 12$ | 100 | 5.96 | 9 | 3 |
| $D_0 = 13$ | 100 | 6.28 | 9 | 4 |
| $D_0 = 14$ | 100 | 6.18 | 9 | 4 |

## 9. Conclusion and Discussion

The QoS of stream-type communication services like interactive voice is sensitive to delay variations. We have presented in this paper an efficient algorithm for solving the stochastic shortest path problem for which optimization function is given by (3.4). This is applicable to routing technologies for stream-type communication services to maintain their QoS. The primary results of this paper may be summarized as follows. First, we investigated the end-to-end delay distribution by measuring an actual network, and showed it follows a normal distribution when the network is congested. Next, we introduced the notion of $x$-weight to make the application of dynamic programming techniques possible, and we introduced the $M$-$V$ plane and re-state the problem in a graphical context, in which the problem is formulated as the process of searching for a specific point on the $M$-$V$ plane. Based on these developments, we proposed an efficient algorithm to search for the point on the $M$-$V$ plane. In this process, we demonstrated that the remaining estimation error is bounded. $(E_{\mathrm{ub}} - E_{\mathrm{opt}})$ decreases monotonically with the number of times the algorithm is applied and the residual error (7.3) decreases similarly. Therefore, we can stop the algorithm taking into account factors such as the network size, available processing power, etc., while controlling the remaining estimation error caused by the untimely termination of the algorithm. Finally we showed simulation results for this algorithm, which lead us to expect that the algorithm will converge fairly rapidly in practical networks.

## Acknowledgements

We would like to thank Mr. Teruyuki Kubo of NTT Software Corporation, Dr. Keisuke Ishibashi and Mr. Ichizo Nakamura of NTT Corporation, for their enlightening discussions and invaluable comments.

## A. Determination of the Value of $x$

The $x$-weight function of each path $P$, $W_x(P)$, is a linear function of $x$ and can be represented on the $x$-$W_x$ plane by a straight line. Suppose that two $x$-optimal paths, $P_1$ and $P_2$ have been found for $x = x_1$ and $x_2$, respectively, and the two lines intersect at $x = x_3$. If another $x$-optimal path $P_3$ is found for $x_1 < x < x_2$, then the corresponding line should fall within the shaded region in Figure 14. Thus, the corresponding $x$-weight function $W_x(P_3)$ should have the following property.

$$W_x(P_1) < W_x(P_3) \quad \text{for} \quad x = x_1, \tag{A.1}$$

$$W_x(P_2) < W_x(P_3) \quad \text{for} \quad x = x_2, \tag{A.2}$$

$$W_x(P_3) < W_x(P_2) \quad \text{and} \quad W_x(P_3) < W_x(P_1) \quad \text{for} \quad x = x_3. \tag{A.3}$$

It can easily be understood that we can determine the existence of such a path by calculating an $x$-optimal path with $x$ set to $x_3$. The value of $x_3$ is calculated as

$$x_3 = -\frac{1}{\mathrm{gradient}(P_1, P_2)}. \tag{A.4}$$

## References

[1] M. Aida, I. Nakamura, and T. Kubo: Optimal routing in communication networks with delay variations. *IEEE Infocom '92*, (1992), 153–159.

[2] R. Bellman: *Dynamic Programing* (Princeton University Press, Princeton, NJ, 1957).

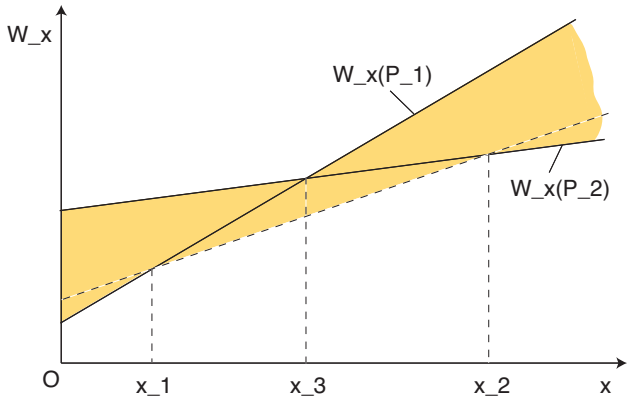


Figure 14: Path representation on $(x, W_x)$-plane

[3] D. Bertsekas and R. Gallager: *Data Networks* (Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987), 322–323.

[4] G. Carle and E.W. Biersack: Survey of error recovery techniques for IP-based audio-visual multicast applications. *IEEE Transactions on Networking*, **11** (1997), 24–96.

[5] T. Ichimori, S. Shiode, H. Ishii, and T. Nishida: Minimal Spanning Tree with Normal Variates as Weight. *Journal of Operations Research Society of Japan*, **24** (1981), 61–65.

[6] M. Schwartz and T. E. Stern: Routing Techniques Used in Computer Communication Networks. *IEEE Transactions on Communications*, **COM-28** (1980), 539–552.

Masaki Aida
Graduate School of System Design
Tokyo Metropolitan University
6-6 Asahigaoka, Hino-shi
Tokyo 191-0065, Japan
E-mail: `maida@sd.tmu.ac.jp`