

# On General Quickest Path Problem and Path-Tables

Nageswara S.V. Rao  
Computer Science and Mathematics Division  
Oak Ridge National Laboratory  
Oak Ridge, Tennessee, USA

Nachimuthu Manickam  
Department of Mathematics  
DePauw University  
Greencastle, Indiana, USA

February 26 - March 2, 2001  
32nd Southeastern International Conference on  
Combinatorics, Graph Theory and Computing  
Baton Rouge, Louisiana

Research Sponsored by  
Laboratory Directed Research and Development Program  
Oak Ridge National Laboratory  
and  
Defense Advanced Research Projects Agency

## Outline

### 1. Introduction

- 1.1 QOS Requirements and Routing Problems
- 1.2 Bandwidth Reservation Framework

### 2. Generalized Quickest Path Problem

- 2.1 Quickest Path Problem
- 2.2 Minimum End-to-End Delay Algorithm

### 3. Path-Table

- 3.1 Quickest Paths
- 3.2 Path-Table Size Estimation

### 4. Conclusions

# End-to-End Requirements

## **Next Generation of Computer Networks:**

Users require routes with end-to-end delay/rate guarantees

### **Scenario 1:**

Physician retrieves x-ray or CATSCAN image from a remote site.

### **Scenario 2:**

Mobile robot team explore a building for radiation/humans

### **Scenario 3:**

Law enforcement officer retrieves all information of a person onto a hand-held computer from a repository.

### **Scenario 4:**

Movie/video on-demand over a computer network

## **Present-Day Networks:**

Routing methods do not provide deterministic guarantees on end-to-end delay

## **In a nutshell:**

New algorithms and mechanisms are needed to provide guarantees

# General Quickest Path Problem

**Given:** computer network  $G = (V, E)$   
available bandwidths  $b(e)$ , for link  $e \in E$   
link-delays  $d(e)$ , for link  $e \in E$   
queuing delay  $q_v(r)$ , for node  $v \in V$ , for message size  $r$

## **Message Transmission Problem:**

Compute a path to send message of  $r$  units  
from  $s$  to  $d$  with minimum end-to-end delay

## **Note:**

1. This is a simple but very important transmission problem
2. No polynomial-time algorithms are known  
— related problems are studied in  
networking, operations research, and transportation
3. Known algorithms:
  - (a) solve restricted versions
  - (b) provide soft bounds for more complicated tasks

# Bandwidth Reservation Framework

## Features:

### **Source-Based Algorithm:**

— Available bandwidth at all links is centrally known

### **Wait While Compute:**

— Bandwidth is put “on hold” while paths are computed

### **Guaranteed bandwidths:**

— Once reserved, bandwidth is held available for the period

## Requirement:

Time complexity of the routing algorithm must be low

## Justification:

1. Simple to implement
2. Can be naturally supported on ATM networks
3. One of the few mechanisms to provide end-to-end guarantees
4. Provides valuable insight into more complicated mechanisms

## Path Delays

**Simple Path:**  $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$ :

End-to-End delay for message size  $r$ :

$$t(r, P) = g\left(r, \min_{j=0}^{k-1} b(e_j)\right) + \sum_{j=0}^{k-1} d(e_j) + \sum_{j=0}^{k-1} q_{v_j}(r)$$

where  $e_j = (v_j, v_{j+1})$ ;

$g\left(r, \min_{j=0}^{k-1} b(e_j)\right)$  is delay due to bandwidth;

$\sum_{j=0}^{k-1} d(e_j)$  is delay due to link-delays; and

$\sum_{j=0}^{k-1} q_{v_j}(r)$  is the queuing delay.

**Notation:**

$$\text{delay: } d(P) = \sum_{j=0}^{k-1} d(e_j)$$

$$\text{bandwidth: } b(P) = \min_{j=0}^{k-1} b(e_j)$$

$$\text{queuing: } q(r, P) = \min_{j=0}^{k-1} q_{v_j}(r)$$

$$t(r, P) = g(r, b(P)) + d(P) + q(r, P)$$

## Quickest Path Problem

**Given:** computer network  $G = (V, E)$   
available bandwidths  $b(e)$ , for link  $e \in E$   
link-delays  $d(e)$ , for link  $e \in E$

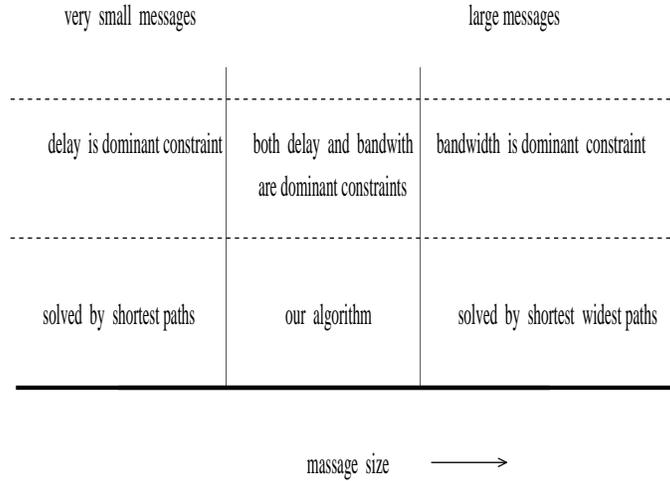
**Special case:**

No queuing delay:  $q_v(r) = 0$ , for all  $v \in V$   
Simple bandwidth:  $g(r, b(P)) = r/b(P)$

$$t(r, P) = r/b(P) + d(P)$$

**Note:**

1. Well-known problem; Chen and Chin (1990), Rosen et al (1991)
2. Solved with time complexity  $(cm + cn \log n)$   
 $c$ : number of distinct bandwidths



## Special Case: Effect of Message Size

**Important Special Case:**  $t(r, P) = r/b(P) + d(P)$

- simple bandwidth and delay constraints
- no queuing delay
- popularly known as the quickest path problem

**Low values of  $r$ :** MTP solved by shortest path  
 — total delay is dominated by delay alone

$$r < \min_{\text{distinct } P_1, P_2} \frac{B_1 B_2 (D_2 - D_1)}{B_2 - B_1}$$

**High values of  $r$ :** MTP solved by shortest-widest path  
 — total delay is dominated by delay alone

$$r > \max_{\text{distinct } P_1, P_2} \frac{B_1 B_2 (D_2 - D_1)}{B_2 - B_1}$$

**Main Point:**

For end-to-end delay *both* bandwidth and delay are important.

## Example

**Network:** Three disjoint paths only  $P_1, P_2$  and  $P_3$ ,

$$D_i = d(P_i) \text{ and } B_i = b(P_i), i = 1, 2, 3$$

$$D_1 < D_3 < D_2 \text{ and } B_1 < B_3 < B_2$$

Shortest path based only on link-delays:  $P_1$

Shortest-widest path:  $P_2$

—  $P_3$  is neither

Total delay is minimized by  $P_3$  for the message size

$$\frac{B_1 B_3 (D_3 - D_1)}{(B_3 - B_1)} < r < \frac{B_2 B_3 (D_2 - D_3)}{(B_2 - B_3)}$$

## Minimum End-To-End Delay

$$\begin{aligned}t(r, P) &= g(r, b(P)) + d(P) + q(r, P) \\ &= g\left(r, \min_{j=0}^{k-1} b(e_j)\right) + \sum_{j=0}^{k-1} d(e_j) + \sum_{j=0}^{k-1} q_{v_j}(r)\end{aligned}$$

### In comparison with quickest path problem:

— Queuing delays are non-zero;

— Bandwidth delays:

decrease with bandwidth and increase with message size  $r$

i.e.  $g(r, b)$  non-decreasing with  $r$  and non-increasing with  $b$

## Path Computation Algorithm

$\{b_1, b_2, \dots, b_c\}$ : distinct values of the bandwidths  $b(e)$ ,  $e \in E$ .

$G(a) = (V, E(a))$ : subnetwork where  $e \in E(a)$  if and only if  $b(e) \geq a$ .

*Augmented Delay* of an edge  $e = (v_1, v_2)$ :

$$d_A(e) = d(e) + q_{v_1}(r)$$

$s - d$  shortest path in  $G(a)$ :

shortest delay path based *only* on the augmented delay of edges

---

*algorithm* Min-Path( $r$ )

1. for  $j = 1, 2, \dots, c$ , compute  $s - d$  shortest path  $P_j$  in  $G(b_j)$ ;
  2. compute index  $k$  which minimizes  
 $\{g(r, b(P_j)) + d(P_j) + q(r, P_j) | j = 1, 2, \dots, c\}$ ;
  3. return  $P_k$  as the path with the minimum end-to-end delay;
- 

— Path with minimum end-to-end delay can be computed in

$O(cm + cn \log n)$  time

$c$ : number of distinct bandwidths

Same complexity as quickest path algorithm

## Path-Table: Quickest Path Problem

### Basic Idea:

Partition range  $[1, \infty]$  of  $r$  into intervals such that:  
each interval has a single path with minimum end-to-end delay

### Computation:

Path-table has  $q \leq m$  entries:

for any message size  $r$ , path with minimum end-to-end delay can be retrieved in  $O(\log n)$  time.

---

*algorithm* Compute-Table(  $P_L, r_L, P_R, r_R$  )

1. compute intersection size  $r_I$ ;
2.  $P_I \leftarrow$  minimum delay path for message size  $r_I$   
    computed using step 2 of algorithm Min-Path;
3. **if**  $[D(P_I) = D(P_L) \text{ and } B(P_I) = B(P_L)]$  or  
     $[B(P_I) = B(P_R) \text{ and } B(P_I) = B(P_R)]$  **then**
4.     Path $[r_L, r_I] \leftarrow P_L$ ; Path $[r_I, r_R] \leftarrow P_R$ ;
5. **else**
6.     Compute-Table( $P_L, r_L, P_I, r_I$ );
7.     Compute-Table( $P_I, r_I, P_R, r_R$ );

---

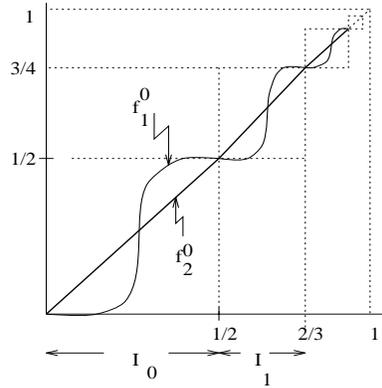
— Rao and Batsell (1997)

— Time complexity  $O(cq + cm + cn \log n)$

## Path-Table in General Case

### Path-Table:

- can be of infinite size for compact range for  $r$
- even when  $g(r, b)$  is monotone



# Smooth Delay Functions

## Under additional conditions:

- (i)  $t(\cdot, P)$ 's are continuous, and
- (ii) no two of them intersect in more than  $s$  points,

## **We have:**

- path-table has no more than  $\lambda_s(p^*)$ , the Davenport-Shinzel number,  
 $p^*$ : size of dominant path set

**Dominant Path Sets:**  $\mathcal{P}_b$ : set of all paths from  $s$  to  $d$  in  $G_b$

$N(P)$ : set of nodes of path  $P$ ;

For  $P_1, P_2 \in \mathcal{P}$ ,

$P_1$  dominates  $P_2$  if  $N(P_1) \subseteq N(P_2)$ ;

$\mathcal{P}_b^*$ : set of all paths of  $\mathcal{P}_b$  that are not dominated

$$p^* = \left| \bigcup_b \mathcal{P}_b^* \right|$$

For certain networks  $p^* = 2^{\lfloor n/2 \rfloor}$

## Finite Path-Tables

**Davenport-Shinzel numbers:** Sharir (1987)

$$\lambda_s(a) \leq \begin{cases} a & \text{if } s = 1 \\ 2a - 1 & \text{if } s = 2 \\ O(a\alpha(a)^{O(\alpha(a)^{s-3})}) & \text{if } s \geq 3 \end{cases}$$

where  $\alpha(\cdot)$  is the slow growing Ackermann's inverse.

—  $\lambda_s(a)$  is almost linear in  $a$  for small  $s$ .

For quickest path problem,  $p^* \leq m$ , and  $s = 1$   
path-table size:  $\lambda_1(m) \leq m$

## Conclusions

### Other Formulations:

1. Queuing delays are statistically estimated  
probabilistic guarantees can be given on the computed path  
— using a similar algorithm

### Research Issues:

1. Tighter bounds on size of dominant path set  
— typical numbers for real networks
2. Efficient path-table computation
3. Multiple paths