Robert Gallager's Minimum Delay Routing Algorithm Using Distributed Computation

Timo Bingmann and Dimitar Yordanov

Decentralized Systems and Network Services Research Group
Institute of Telematics, Universität Karlsruhe

January 29, 2007

DSN

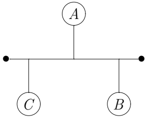
Road Map

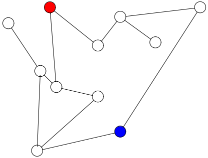
1 Introduction
2 Model
3 Algorithm
4 Conclusion

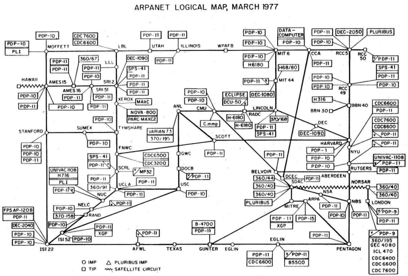
Introduction: Routing Algorithms

What are they?

Why do we need them?







Goals of Routing Algorithms

Primary Goal

Achieve "good" or even *optimal routing*.

- How to measure routing quality?
  $\longrightarrow$ Routing metrics

Other Aims

- little network overhead
- stability and reliablity
- adapt to changes
- quickly converge to optimal state
- scale well

Characteristics

Route Calculation Time

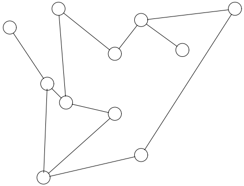
- Static routing algorithms
- Dynamic routing algorithms
- Quasi-static routing algorithms

Characteristics

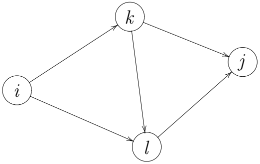
Other Characteristics

- Single-Path vs. Multi-Path Algorithms
- Centralized vs. Distributed Algorithms
- User vs. System Optimization

Model

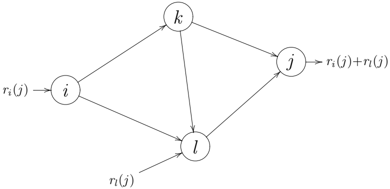


Model



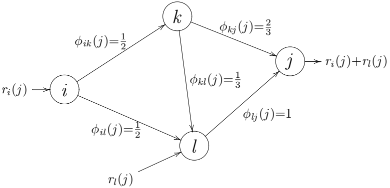
Set of n nodes enumerated by $\{1, 2, \dots, n\}$
Set of links: $\mathcal{L} := \{(i, j) \text{ is existing link}\}$

Model



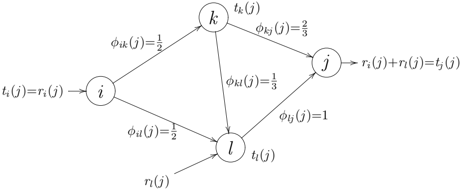
Input traffic entering at i and destined for j: $r_i(j)$.
e.g. in kbit/s

Model



Routing variables $\phi_{ik}(j)$:
Fraction of traffic destined for j travelling link (i, k).

Model



Sum over all traffic at node i destined for j: $t_i(j)$.

Constraints on ϕ

1. No traffic on non-existing links and no loopback traffic

$$\phi_{ik}(j) = 0 \quad \forall\,(i,j) \notin \mathcal{L} \text{ or } i = j$$

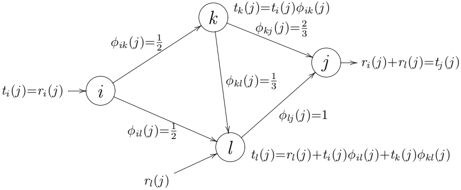
2. No loss of traffic is allowed.

$$\sum_{k=1}^{n} \phi_{ik}(j) = 1 \quad \forall\, i, j$$

3. All nodes are inter-connected.

$$\phi_{ik}(j) > 0, \phi_{kl}(j) > 0, \ldots, \phi_{mj}(j) > 0$$
$$\exists\, i, k, l, \ldots, m, j \;\forall\, i, j$$

Model



$$t_i(j) = r_i(j) + \sum_{l=1}^{n} t_l(j)\phi_{li}(j)$$

Variables

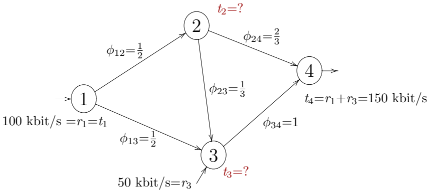
- Set of n nodes enumerate by $\{1, 2, \ldots, n\}$
- Set of links: $\mathcal{L} := \{(i,j) \text{ is existing link}\}$
- Input traffic set $\boldsymbol{r} := \{r_i(j)\}$
- Node flow set $\boldsymbol{t} := \{t_i(j)\}$
- Routing variable set $\boldsymbol{\phi} := \{\phi_{ik}(j)\}$.

Theorem 1

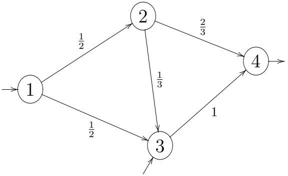
The routing variable set $\boldsymbol{\phi}$ will actually guide the network's flow.

Formally: An input set $\boldsymbol{r}$ and a routing variable set $\boldsymbol{\phi}$ uniquely define a network flow set $\boldsymbol{t}$.

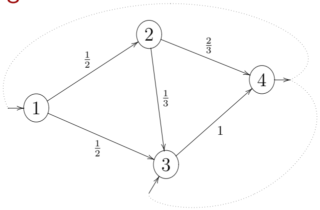
Routing Variables



Routing Variables

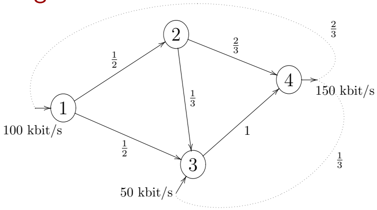


Routing Variables



Find steady state by introducing imaginary links which transfer traffic back to its source node.

Routing Variables



$$\phi_{ji}(j) := \frac{r_i(j)}{\sum_k r_k(j)}$$

Markov Transition Matrix

$$\Phi = (\phi_{ik}(j))_{i,k} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 0 & 1 \\ \frac{2}{3} & 0 & \frac{1}{3} & 0 \end{pmatrix}$$

The second constraint on ϕ and $\phi_{ik}(j) \geq 0$ are the defining properties of a stochastic matrix.

Markov Equation

With $\phi_{ji}(j) := \frac{r_i(j)}{\sum_k r_k(j)}$ the aggregation equation

$$t_i(j) = r_i(j) + \sum_{l=1}^{n} t_l(j)\phi_{li}(j)$$

can be contracted to

$$t_i(j) = \sum_{l=1}^{n} t_l(j)\phi_{li}(j) \quad \Leftrightarrow \quad \bar{t} = \bar{t}\Phi$$

Equilibrium Distribution

$$\bar{t} = \bar{t}\Phi$$

Is the equation of a Markov chain in an equilibrium state.

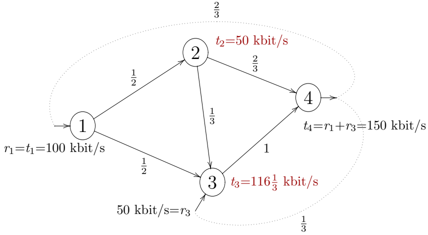
From Markov chain theory: If the transition matrix is irreducible, then exactly one equilibrium distribution $\bar{t}$ exists.

Equilibrium in the Example

$$\Phi = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 0 & 1 \\ \frac{2}{3} & 0 & \frac{1}{3} & 0 \end{pmatrix} \qquad \lim_{n\to\infty} \Phi^n = \begin{pmatrix} \frac{6}{25} & \frac{3}{25} & \frac{7}{25} & \frac{9}{25} \\ \frac{6}{25} & \frac{3}{25} & \frac{7}{25} & \frac{9}{25} \\ \frac{6}{25} & \frac{3}{25} & \frac{7}{25} & \frac{9}{25} \\ \frac{6}{25} & \frac{3}{25} & \frac{7}{25} & \frac{9}{25} \end{pmatrix}$$

$$\Rightarrow \bar{t}' = \begin{pmatrix} \frac{6}{25} \\ \frac{3}{25} \\ \frac{7}{25} \\ \frac{9}{25} \end{pmatrix}^{\top} \quad \Rightarrow \quad \bar{t} = \begin{pmatrix} 100 \\ 50 \\ 116\frac{1}{3} \\ 150 \end{pmatrix}^{\top} \text{kbit/s}$$

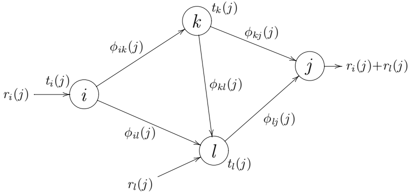
Equilibrium in the Example



Delay

Currently the model only describes traffic flow.

Now introduce delay.



Traffic and Delay

First define total traffic f_{ik} on a link (i, k)

$$f_{ik} = \sum_j t_i(j)\phi_{ik}(j)$$

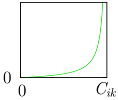
Traffic and Delay

Then calculate link delay $D_{ik}(f_{ik})$ from the traffic.

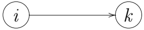
Only requirements of D_{ik}: convex and increasing.

For example

$$D_{ik}(f_{ik}) = \frac{f_{ik}}{C_{ik} - f_{ik}}$$



with link capacity C_{ik}.



Total delay

Finally define total delay D_T

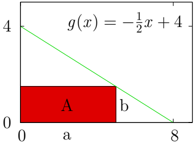
$$D_T = \sum_{i,k} D_{ik}(f_{ik})$$

Goal: Minimize D_T by setting optimal $\phi_{ik}(j)$.

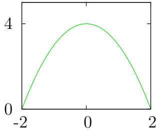
Use same general method as with maximizing rectangle area function in school.

General Method

Problem:
Find $a, b = g(a)$ with maximum area A.
Set first derivative to zero.



$$A(a) = a \cdot b = a \cdot g(a)$$
$$= -\tfrac{1}{2}a^2 + 4a$$
$$A'(a) = -a^2 + 4$$
$$A'(a) = 0 \text{ for } a = \pm\sqrt{4}$$
$$\Rightarrow b = 5$$



Derivative of D_T

Method: Determine the derivative of D_T and find a root.

But derive D_T by which parameter?

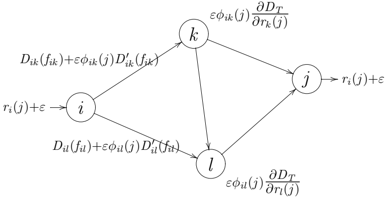
D_T is the sum of all delays D_{ik}.
Each D_{ik} is a function of the link traffic f_{ik}.
f_{ik} is somehow determined by $\boldsymbol{r}$, $\boldsymbol{t}$ and $\boldsymbol{\phi}$.

$$D'_{ik}(f_{ik}) = \frac{\mathrm{d}D_{ik}(f_{ik})}{\mathrm{d}f_{ik}}$$

Partial Derivatives of D_T

Easier: Determine partial derivative $\dfrac{\partial D_T}{\partial r_i(j)}$

How does more input traffic change total delay?

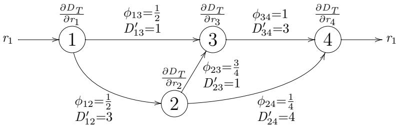


Partial Derivatives of D_T

Partial derivative regarding input traffic:

$$\frac{\partial D_T}{\partial r_i(j)} = \sum_k \phi_{ik}(j) \left(D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \right)$$

Calculate marginal (incremental) delay in this example:

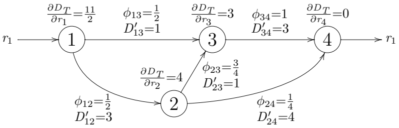


Partial Derivatives of D_T

Partial derivative regarding input traffic:

$$\frac{\partial D_T}{\partial r_i(j)} = \sum_k \phi_{ik}(j) \left(D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \right)$$

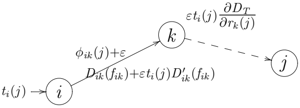
Calculate marginal (incremental) delay in this example:



Partial Derivatives of D_T

However a future algorithm should change routing variables $\phi_{ik}(j)$.

So determine their change to delay: $\frac{\partial D_T}{\partial \phi_{ik}(j)}$



Finding a Root

$$\frac{\partial D_T}{\partial \phi_{ik}(j)} = t_i(j) \left(D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \right)$$

Find a stationary point of D_T regarded as a function of $\phi_{ik}(j)$ in which all $\frac{\partial D_T}{\partial \phi_{ik}(j)} = 0$ $(\nabla D_T(\boldsymbol{\phi}) = 0)$.

However $\boldsymbol{\phi}$ has the three constraints
$\Rightarrow$ Lagrange multipliers are required.

Lagrange Multipliers

Formalize the constraints into a function $g(\boldsymbol{\phi}) = 0$, with $\nabla g(\boldsymbol{\phi}) \neq 0$.

Introduce Lagrange multipliers λ and solve:

$$\nabla D_T(\boldsymbol{\phi}) = -\lambda g(\boldsymbol{\phi})$$
$$g(\boldsymbol{\phi}) = 0$$

Lagrange Multipliers

Result:

$$\frac{\partial D_T}{\partial \phi_{ik}(j)} \begin{cases} = \lambda_{ij}, & \phi_{ik}(j) > 0 \\ \geq \lambda_{ij}, & \phi_{ik}(j) = 0 \end{cases} \quad \forall\, i \neq j \;\forall\, (i,k) \in \mathcal{L}$$

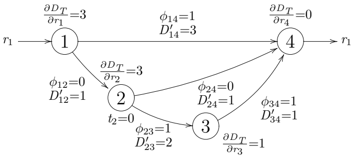
Note that the λ_{ij} do not depend on k.

$\Rightarrow$ All used links must have same marginal delay.
Unused must have greater marginal delay.

Only Necessary

However this condition is not sufficient.

Counter-example:

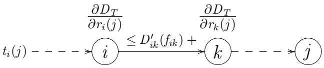


Sufficient Condition

Brilliant idea of Gallager: remove the factor $t_i(j)$

$$\frac{\partial D_T}{\partial r_i(j)} \leq D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)}$$

Intuitive reduction of delay:

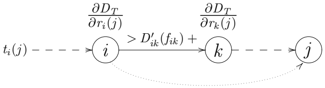


Sufficient Condition

Brilliant idea of Gallager: remove the factor $t_i(j)$

$$\frac{\partial D_T}{\partial r_i(j)} \leq D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)}$$

Intuitive reduction of delay (Contraposition):



Transformation into Algorithm

$$\frac{\partial D_T}{\partial r_i(j)} \leq D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)}$$

transformed into an iterative version useful for the future algorithm

$$D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \geq \min_{(i,m)\in\mathcal{L}} \left(D'_{im}(f_{im}) + \frac{\partial D_T}{\partial r_m(j)} \right)$$

The Algorithms Main Goal

- Calculate new routing variables (ϕ_{ik})
  - increase ϕ_{ik} on links with small marginal delay
  - decrease ϕ_{ik} on links with large marginal delay
- During iterative distributed computation:
  - stable state is reached
  - optimal solution is found
  - no deadlock occurs

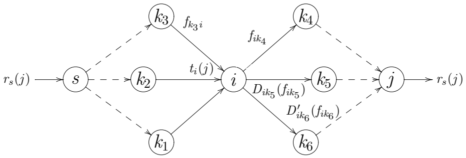
The Algorithm

1. Determine the necessary variables: $\frac{\partial D_{ik}}{\partial r_i(j)}$ and $D'_{ik}(f_{ik})$
2. Calculate new routing variables ϕ^1
   - main challenge: keep ϕ loop free

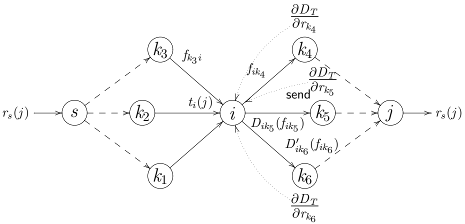
Variables Available to a Specific Node

- A node knows:
  - its incoming and outgoing links
  - its neighbors
  - the amount of traffic flow (can be measured)
  - its routing variables for all links and destinations

Variables Available to a Specific Node



Variables Available to a Specific Node

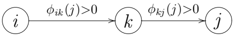


Determine Marginal Delay

- D_{ik} can be calculated or measured
- D'_{ik} can be calculated from D_{ik}
- D'_{ik} more often measured
- Still missing $\frac{\partial D_{ik}}{\partial r_i(j)}$

Downstream Concept

- Each node becomes $\frac{\partial D_{ik}}{\partial r_i(j)}$ from its downstream neighbors
- Node k is downstream from i with respect to destination j, if there is a path from i to j through k and all routing variables on the way down to j are positive (i.e. $\phi_{il_1}(j) > 0$ … $\phi_{l_n j}(j) > 0$)



Routing Variables Calculation

- Calculate new variables in three steps.
- Determine the best link (lowest marginal delay)
- Difference between each link k and the best link:

$$a_{ik}(j) = \underbrace{D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)}}_{\text{on link } k} - \underbrace{\left(D'_{ib}(f_{ib}) + \frac{\partial D_T}{\partial r_b(j)}\right)}_{\text{on the best link}}$$

Routing Variable Reduction

$\Delta_{ik}(j)$: the reduction of routing variable $\phi_{ik}(j)$

$$\Delta_{ik}(j) = \min\left\{\phi_{ik}(j),\ \frac{\eta}{t_i(j)} a_{ik}(j)\right\}$$

with a small scale factor η.

The New Routing Variables

$$\phi^1_{ik}(j) = \begin{cases} \phi_{ik}(j) - \Delta_{ik}(j), \\ \quad \text{if } (i,k) \text{ is not the best link} \\ \phi_{ib}(j) + \sum\limits_{\substack{(i,m)\in\mathcal{L} \\ m\neq b}} \Delta_{im}(j), \\ \quad \text{if } (i,k) \text{ is the best link} \\ \quad \text{and therefore } k = b \end{cases}$$

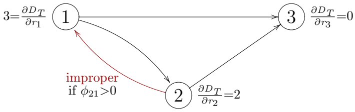
Blocked Set

- Blocked set $B_i(j)$: restrict flow from node i
  - require: $\phi_{ik}(j) = 0 \; \forall k \in B_i(j)$
- Nodes included in $B_i(j)$
  - nodes, which do not have link to node i
  - neighbors, which have downstream paths containing a loop

Improper Routing Variables

A routing variable $\phi_{ik}(j)$ is defined as improper if

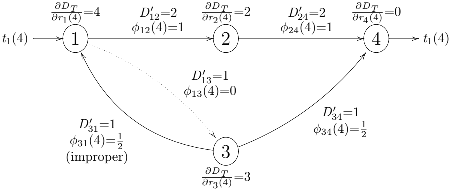
$$\phi_{ik}(j) > 0 \quad \text{and} \quad \frac{\partial D_T}{\partial r_i(j)} \leq \frac{\partial D_T}{\partial r_k(j)}$$



Blocked Set Definition

Formally $B_i(j)$ includes all nodes k, for which $\phi_{ik}(j) = 0$ and k can route packets to j over a path that contains some link (l, m) with *improper* $\phi_{lm}(j)$ and $\phi^1_{lm}(j) > 0$.

Example



Theorem 5

For every $D_0 > 0$
there exists a scale factor η for the algorithm A, such that if ϕ^0 satisfies $D_T(\phi^0) \leq D_0$, then

$$\lim_{m\to\infty} D_T(A^m(\phi)) = \min_{\phi} D_T(\phi)$$

Proof is done via seven lemmas over four pages (of twelve) in the paper.

Outline of Proof

Say $\phi^1 := A(\phi)$ and f^1 the new link flow.

First goal: calculate $D_T(\phi^1) - D_T(\phi)$.

Gallager uses auxiliary function ($0 \leq \lambda \leq 1$):
$D_T(\lambda) = \sum_{i,k} D_{ik}(f_{ik}^\lambda)$ with $f_{ik}^\lambda = f_{ik} + \lambda(f_{ik}^1 - f_{ik})$

and applies Taylor's remainder theorem in Lagrange form:

$$D_T(\phi^1) - D_T(\phi) = \left(\frac{\mathrm{d}D_T(\lambda)}{\mathrm{d}\lambda}\right)(0) + \frac{1}{2}\left(\frac{\mathrm{d}^2 D_T(\lambda)}{\mathrm{d}\lambda^2}\right)(\lambda^*)$$

Outline of Proof

Lemmas 1 to 4 are used to upper bound $\frac{\mathrm{d}D_T(\lambda)}{\mathrm{d}\lambda}$ and $\frac{\mathrm{d}^2 D_T(\lambda)}{\mathrm{d}\lambda^2}$.

Concluding in lemma 5:

For D_0 say $M := \max_{i,k} \max_{f:D_{ik}(f)\leq D_0} D''_{ik}(f)$

and let $\eta := \dfrac{1}{Mn^6}$, then

$$D_T(\phi^1) - D_T(\phi) \leq -\frac{1}{2\eta(n-1)^3} \sum_{i,j} \Delta_i^2(j)\, t_i^2(j)$$

Outline of Proof

In lemma 6 the last lemma is used to show a strict monotony criterion.

Let ϕ be routing variables with $D_T(\phi) < D_0$ but not the minimum.

Then $\exists\, \varepsilon > 0$ and m with $1 \leq m \leq n$:

$$\forall\, \phi^* : |\phi - \phi^*| < \varepsilon : D_T(A^m(\phi^*)) < D_T(\phi)$$

Proof includes a detailed analysis of the algorithm's steps for improper links and blocked nodes.

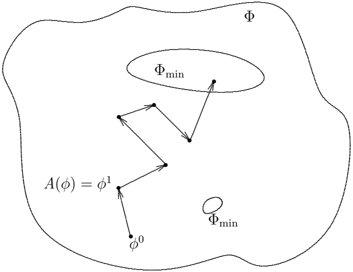
Outline of Proof

Let $\Phi \subseteq \mathbb{R}^n$ compact euclidean space of routing variables.

Then algorithm is a mapping $A : \Phi \to \Phi$, and $D_T : \Phi \to \mathbb{R}$ a real function.

Let $D_{\min}$ minimum of D_T over Φ and $\Phi_{\min}$ set of ϕ with $D_T(\phi) = D_{\min}$.

Outline of Proof



Outline of Proof

Because Φ is compact the sequence $\{A^m(\phi)\}$ has a convergent subsequence $\{\phi^l\}$.

Let $\phi' = \lim_{l\to\infty} \phi^l$, and since D_T is continuous $D_T(\phi') = \lim_{l\to\infty} D_T(\phi^l)$.

Left to prove: $D_T(\phi') = D_{\min}$.
Follows from $D_T(A^m(\phi)) < D_T(\phi)$.

Problems

- First drawback: required scale parameter η
- How can the start state be determined?
- What if links or nodes are dropped or added?
- Adapting to changing input traffic statistics.

Conclusion

- Rigorous mathematical approach
- Well designed mathematical model:
  - describe the minimum total delay problem
  - conditions for achieving global optimization
- Iterative, distributed routing algorithm
  - proved in detail that the algorithm will always progress into a network state with total minimum delay
- 209 citations on Google Scholar, 55 on Citeseer.