Online-routing on the butterfly network

Probabilistic analysis

Andrey Gubichev

Ferienakademie im Sarntal 2008
Saint-Petersburg State University

September 2008

Overview

Overview

The Butterfly

The r-dimensional butterfly consists of

- $(r + 1)2^r$ nodes and
- $r2^{r+1}$ edges

The Butterfly

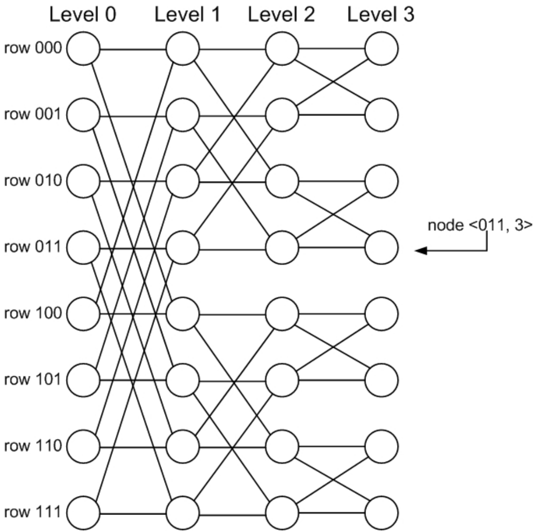
The r-dimensional butterfly consists of

- $(r+1)2^r$ nodes and
- $r2^{r+1}$ edges

such that

- node is $\langle w, i \rangle$: i is a level, w - r-bit number of row
- $\langle w, i \rangle$ and $\langle w', i' \rangle$ are linked $\Leftrightarrow$ (w=w' OR w and w' differ in ith bit) AND i'=i+1

Example: three-dimensional butterfly



Routing Problem

- routing N packets
- start — node $\langle u, 0 \rangle$ on level 0
- destination — node $\langle \pi(u), \log N \rangle$ on level $\log N$
  - $\pi : [1, N] \longrightarrow [1, N]$ is a permutation
- on-line algorithms: no global controller

Greedy algorithm

- the unique path of length $\log N$ from $\langle u, 0\rangle$ to $\langle \pi(u), \log N\rangle$ — *greedy path*
- *greedy routing algorithm*: each packet follows its greedy path

Greedy algorithm

- the unique path of length $\log N$ from $\langle u, 0 \rangle$ to $\langle \pi(u), \log N \rangle$ — *greedy path*
- *greedy routing algorithm*: each packet follows its greedy path
- main problem: routing many packets in parallel $\Rightarrow$ many greedy paths might pass through a single node or edge: *Congestion*!

Overview

Fact: greedy algorithm efficiency

The algorithm that chooses greedy paths, can solve any routing problem in $O(\sqrt{N})$

Overview: worst-case behavior

- if π is the bit-reversal permutation:

$$\pi(u_1 \cdots u_{\log N}) = u_{\log N} \cdots u_1$$

then the greedy algorithm will take $O(\sqrt{N})$ steps (and congestion $C \geq \sqrt{N}/2$)

Overview: worst-case behavior

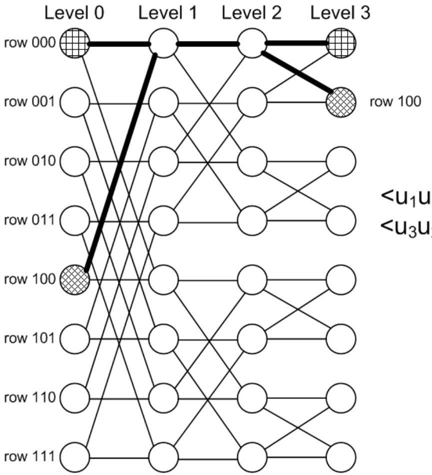
- if π is the bit-reversal permutation:

$$\pi(u_1 \cdots u_{\log N}) = u_{\log N} \cdots u_1$$

then the greedy algorithm will take $O(\sqrt{N})$ steps (and congestion $C \geq \sqrt{N}/2$)
- the same result holds for transpose permutation

$$\pi(u_1 \cdots u_{\frac{\log N}{2}} u_{\frac{\log N}{2}+1} \cdots u_{\log N}) = u_{\frac{\log N}{2}+1} \cdots u_{\log N} u_1 \cdots u_{\frac{\log N}{2}}$$

Example: bit-reversal permutation



$<u_1u_2u_3,0> \rightarrow$
$<u_3u_2u_1,3>$

Average-case behavior: problem statement

- we need to route packets in the butterfly
- all packets start at level 0
- each packet has a destination at level $\log N$,considered as *random*

Average-case behavior: problem statement

- we need to route packets in the butterfly
- all packets start at level 0
- each packet has a destination at level $\log N$,considered as *random*
- p is the number of packets at each input
  - if $p = 1$: standard N-packet routing problem
  - if $p = \log N$: network is more heavily loaded

Plan of analysis

- obtain bounds on congestion
- obtain bounds on running time

Overview

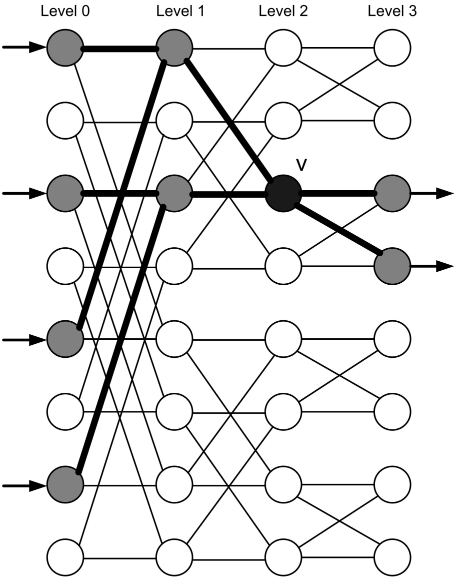
Upper bound on $P_r(v)$

- $P_r(v) =$ Probability(r or more packet paths pass through node v on level i), $r > 0$, $0 \leq i \leq \log N$
  - we are randomizing routing problems!

Upper bound on $P_r(v)$

- $P_r(v) =$ Probability(r or more packet paths pass through node v on level i), $r > 0$, $0 \leq i \leq \log N$
  - we are randomizing routing problems!
- at most $p2^i$ packets pass through v
- there are $2^{\log N - i}$ choices of destinations that will cause these packets to pass through v
  $\Longrightarrow$ each of $p2^i$ pass through v with probability 2^{-i}

Example: choices of inputs and outputs



$i = 2$

2^i inputs = 4

$N2^{-i}$ outputs = 2

Upper bound on $P_r(v)$

$$P_r(v) \leq \binom{p2^i}{r}(2^{-i})^r \leq \left(\frac{p2^i e}{r}\right)^r 2^{-ir} = \left(\frac{pe}{r}\right)^r$$

Notes about upper bound on $P_r(v)$

$P_r(v) \leq \left(\frac{pe}{r}\right)^r$

- The bound does not depend on v or on i

Notes about upper bound on $P_r(v)$

$P_r(v) \leq \left(\frac{pe}{r}\right)^r$

- The bound does not depend on v or on $i \Longrightarrow$ for any random routing problem r or more packets pass through any node with probability $\leq N \log N \left(\frac{pe}{r}\right)^r$

Notes about upper bound on $P_r(v)$

$P_r(v) \leq \left(\frac{pe}{r}\right)^r$

- The bound does not depend on v or on $i \Longrightarrow$ for any random routing problem r or more packets pass through any node with probability $\leq N \log N \left(\frac{pe}{r}\right)^r$
- we can make this probability be very low by choosing large r

Two particular cases of upper bound on $P_r(v)$: case 1

- if $p \geq \frac{\log N}{2}$, we choose $r = 2ep = O(p)$:

$$N \log N \left(\frac{pe}{r}\right)^r \leq N \log N \left(\frac{1}{2}\right)^{e \log N} = N^{1-e} \log N \leq 1/N^{3/2}$$

Two particular cases of upper bound on $P_r(v)$: case 2

- if $p \leq \frac{\log N}{2}$, we choose $r = \frac{2e \log N}{\log\left(\frac{\log N}{p}\right)}$ and omit technical details:

$$N \log N \left(\frac{pe}{r}\right)^r \leq 1/N^2$$

Result: outline of the analysis

- bound for $P_r(v)$
- it does not depend on v and $i \Rightarrow$ bound for all nodes
- it decreases when r increases
- choose r (for different p) large enough to make the bound small: $1/N^{3/2}$

Result: bound on congestion

For all but at most a $1/N^{3/2}$ fraction of the possible routing problems at most C packets pass through each node during a greedy routing where

$$C = \begin{cases} 2ep, \text{ if } p \geq \dfrac{\log N}{2} \\ 2e \log N / \log\left(\dfrac{\log N}{p}\right), \text{ if } p \leq \dfrac{\log N}{2} \end{cases}$$

Result: simple form of a bound

With high probability the congestion in a random problem is at most

$$C = O(p) + o(\log N)$$

Generalization of the result

Corollary. For any $\alpha > 0$, the congestion of all but $1/N^{\alpha}$ of the possible routing problems with p packets per input in a $\log N$-dimensional butterfly is at most $O(\alpha p) + o(\alpha \log N)$

Two special cases

- $p = 1$: the maximum number of packets that pass through any node is $O(\log N / \log\log N)$ with high probability
  - compare this bound with the worst case congestion: $O(\sqrt{N})$

Two special cases

- $p = 1$: the maximum number of packets that pass through any node is $O(\log N / \log\log N)$ with high probability
  - compare this bound with the worst case congestion: $O(\sqrt{N})$
- $p = \Theta(\log N)$: at most $O(\log N)$ packets will pass through any node with high probability

Conclusion: first bound on running time

The time needed to deliver every packet to its destination is at most $(C-1)\log N$ in most routing problems, where

$$C = O(p) + o(\log N)$$

.

Conclusion: first bound on running time

The time needed to deliver every packet to its destination is at most $(C-1)\log N$ in most routing problems, where

$$C = O(p) + o(\log N)$$

.

Now we will show that the running time is $\log N + O(p) + o(\log N)$ for almost all routing problems.

Overview

Random-rank protocol: motivation

If two or more packets are waiting to exit a node, we need to specify a protocol for deciding which packet will move forward out of the node first.

Random-rank protocol: details

- random priority key $r(P) \in [1, K]$ for each packet P
- define total order on the packets: $t(P)$ is the rank of packet P

Random-rank protocol: details

- random priority key $r(P) \in [1, K]$ for each packet P
- define total order on the packets: $t(P)$ is the rank of packet P
- define $w(P) = (r(P), t(P))$
- order w(P):
  if $P \neq P'$ we say that $w(P) < w(P') \Leftrightarrow (r(P) < r(P'))$ OR $(r(P) = r(P')$ AND $t(P) < t(P'))$

Random-rank protocol: details

- random priority key $r(P) \in [1, K]$ for each packet P
- define total order on the packets: $t(P)$ is the rank of packet P
- define $w(P) = (r(P), t(P))$
- order w(P):
  if $P \neq P'$ we say that $w(P) < w(P') \Leftrightarrow (r(P) < r(P'))$ OR $(r(P) = r(P')$ AND $t(P) < t(P'))$
- the packet with smallest w exits the node first

Random-rank protocol: naive question

Why do we need both r and t?

Random-rank protocol: naive question

Why do we need both r and t?

- r is random $\Rightarrow$ sometimes not unique
- t is not random
- $w(P) = (r(P), t(P))$ is random and unique

Example: random-rank protocol

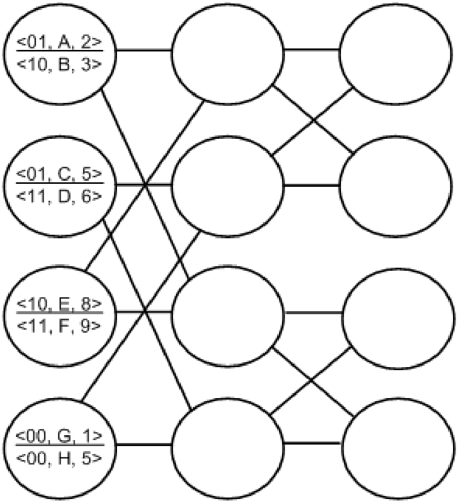


Figure: initial configuration: ⟨destination, name, random key⟩

Example: random-rank protocol

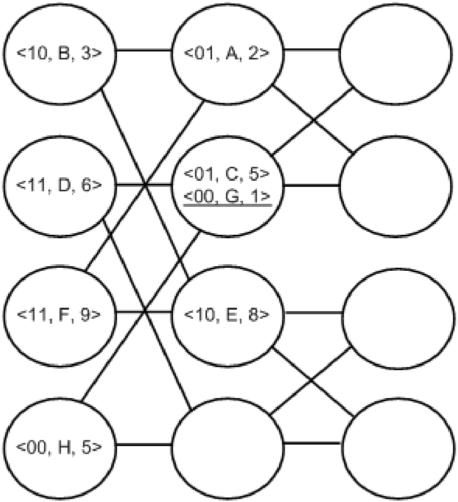


Figure: after step 1

Example: random-rank protocol

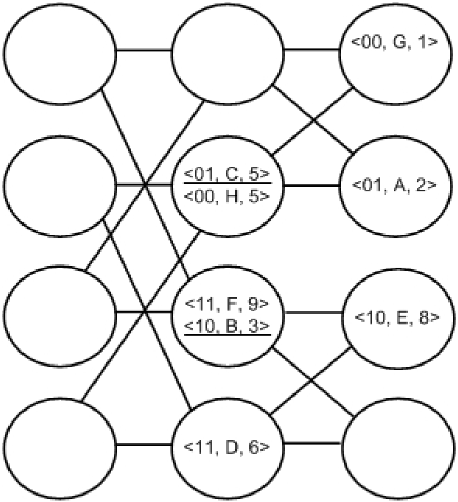


Figure: after step 2

Example: random-rank protocol

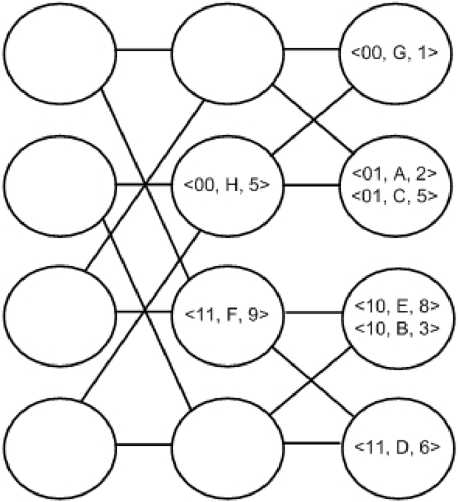


Figure: after step 3

Example: random-rank protocol

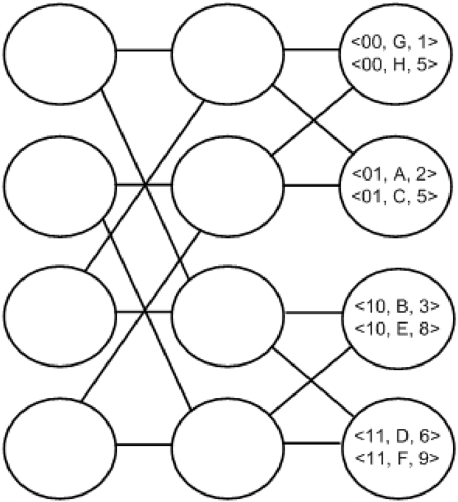


Figure: after step 4

Theorem about running time

If we use random-rank protocol, the congestion equals C, then the running time is T with probability at least $1 - 1/N^7$, where

$$T = \begin{cases} O(C), \text{ if } C \geq \dfrac{\log N}{2} \\ \log N + O(\log N / \log\left(\dfrac{\log N}{C}\right)), \text{ if } C \leq \dfrac{\log N}{2} \end{cases}$$

Proof: preliminaries

We consider routing problem with congestion number C, random keys $r(P)$ and running time T. We will show that T satisfies the bound from the theorem.

Delay path

- P_0 is the last packet to reach its destination v_0, it was last delayed at the node v_1, l_0 is the number of steps in the path $v_1 \longrightarrow v_0$

Example: delay path

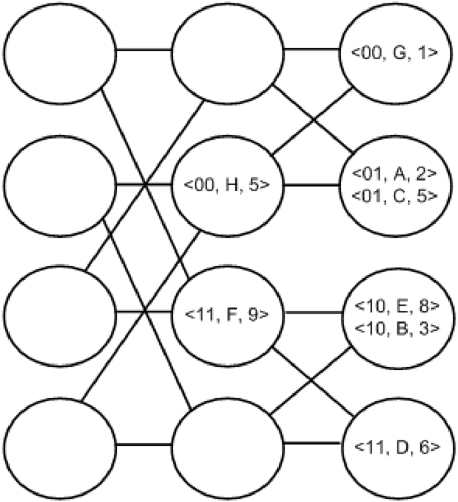


Figure: $P_0 = F$, $v_0 = \langle 11, 2 \rangle$

Delay path

- P_0 is the last packet to reach its destination v_0, it was last delayed at the node v_1, l_0 is the number of steps in the path $v_1 \rightarrow v_0$
- P_1 is the packet responsible for delaying P_0. P_1 itself was delayed at the node v_2, l_1 is the number of steps in the path $v_2 \rightarrow v_1$

Example: delay path

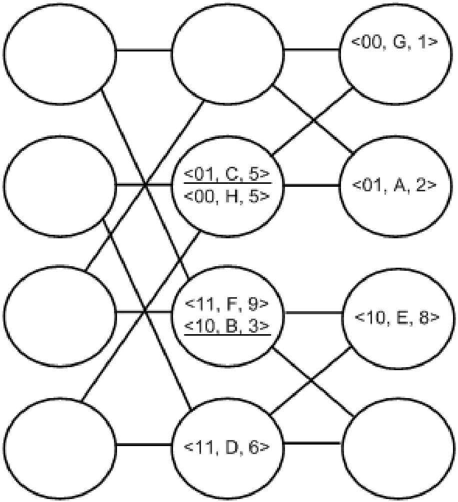


Figure: $P_1 = B$, $v_1 = \langle 10, 1 \rangle$

Delay path

- P_0 is the last packet to reach its destination v_0, it was last delayed at the node v_1, l_0 is the number of steps in the path $v_1 \to v_0$
- P_1 is the packet responsible for delaying P_0. P_1 itself was delayed at the node v_2, l_1 is the number of steps in the path $v_2 \to v_1$
- we proceed in a similar fashion until the sequence of delays ends at v_s

Example: delay path

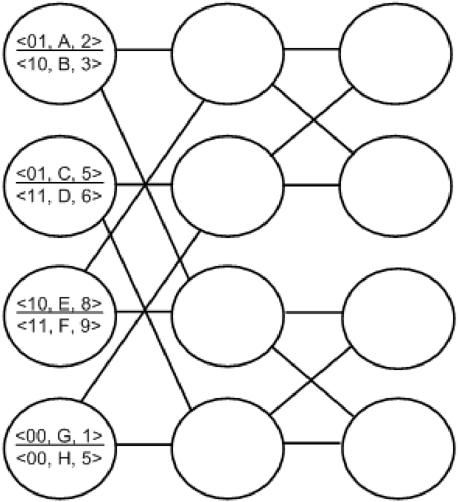


Figure: $P_2 = A$, $v_2 = \langle 00, 0 \rangle$

Delay path

- P_0 is the last packet to reach its destination v_0, it was last delayed at the node v_1, l_0 is the number of steps in the path $v_1 \to v_0$
- P_1 is the packet responsible for delaying P_0. P_1 itself was delayed at the node v_2, l_1 is the number of steps in the path $v_2 \to v_1$
- we proceed in a similar fashion until the sequence of delays ends at v_s. P_s moves forward from v_s during step 1.
- $\mathbf{P} = v_s \to \ldots \to v_1 \to v_0$ is the delay path

Delay path and running time

$T - l_0 - l_1 - \ldots - l_{s-1} - (s-1) = 1$ and
$l_0 + \ldots + l_{s-1} = \log N \Rightarrow s = T - \log N$

Delay sequence

A delay sequence consists of

- a delay path **P**

Delay sequence

A delay sequence consists of

- a delay path **P**
- integers $l_0 \geq 1, l_1 \geq 0, \ldots, l_{s-1} \geq 0$, $l_0 + \ldots + l_{s-1} = \log N$

Delay sequence

A delay sequence consists of

- a delay path **P**
- integers $l_0 \geq 1, l_1 \geq 0, \ldots, l_{s-1} \geq 0$, $l_0 + \ldots + l_{s-1} = \log N$
- nodes v_0,v_1,...,v_s: v_i is the node of **P** on level $\log N - l_0 - \ldots - l_{s-1}$

Delay sequence

A delay sequence consists of

- a delay path **P**
- integers $l_0 \geq 1, l_1 \geq 0, \ldots, l_{s-1} \geq 0$, $l_0 + \ldots + l_{s-1} = \log N$
- nodes v_0,v_1,...,v_s: v_i is the node of **P** on level $\log N - l_0 - \ldots - l_{s-1}$
- different packets $P_0, P_1, \ldots, P_s$: the greedy path for P_i contains v_i

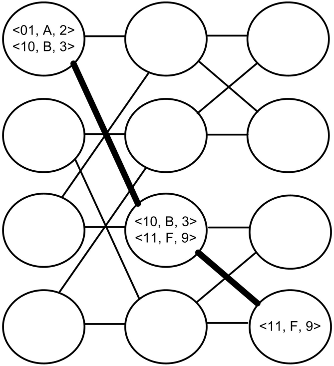
Delay sequence

A delay sequence consists of

- a delay path **P**
- integers $l_0 \geq 1, l_1 \geq 0, \ldots, l_{s-1} \geq 0$, $l_0 + \ldots + l_{s-1} = \log N$
- nodes v_0,v_1,...,v_s: v_i is the node of **P** on level $\log N - l_0 - \ldots - l_{s-1}$
- different packets $P_0, P_1, \ldots, P_s$: the greedy path for P_i contains v_i
- keys $k_0, k_1, \ldots, k_s$ for the packets: $k_s \leq k_{s-1} \leq \ldots \leq k_0$, $k_i \in [0, K]$.

A delay sequence is *active*, if $r(P_i) = k_i$ for $0 \leq i \leq s$.

Example: active delay sequence



$$P = \langle 00,0\rangle \rightarrow \langle 10,1\rangle \rightarrow \langle 11,2\rangle$$

$$l_0 = 1,\ l_1 = 1$$

$$v_0 = \langle 11,2\rangle,\ v_1 = \langle 10,1\rangle,\ v_2 = \langle 00,0\rangle$$

$$P_0 = F,\ P_1 = B,\ P_2 = A$$

$$k_0 = 9,\ k_1 = 3,\ k_2 = 2$$

Main property of an active delay sequence

$$\Pr(T \leq s + \log N) \leq$$

$$\leq \Pr(\text{there is an active delay sequence with } s+1 \text{ packets})$$

Number of possible delay sequences N_d

There are many possible delay sequences!

- N^2 choices for delay path **P**

Number of possible delay sequences N_d

There are many possible delay sequences!

- N^2 choices for delay path **P**
- $\binom{s+\log N-2}{s-1}$ choices for $l_0 \geq 1, l_1 \geq 0 \ldots, l_s \geq 0, \sum l_i = \log N$

Number of possible delay sequences N_d

There are many possible delay sequences!

- N^2 choices for delay path **P**
- $\binom{s+\log N-2}{s-1}$ choices for $l_0 \geq 1, l_1 \geq 0 \ldots, l_s \geq 0, \sum l_i = \log N$
  - Why?

Combinatorial explanation

There is one-to-one correspondence between choices for l_i and $(s + \log N - 2)$-bit binary string t with $s - 1$ zeros:

- l_i is the number of "1" between $(i + 1)$st and $(i + 2)$nd zeros in the string $01t0$

Combinatorial explanation

There is one-to-one correspondence between choices for l_i and $(s + \log N - 2)$-bit binary string t with $s - 1$ zeros:

- l_i is the number of "1" between $(i + 1)$st and $(i + 2)$nd zeros in the string $01t0$
- if $\log N = 3, s = 5, t = 001100$, then

$$01t0 = 010011000$$

and $l_0 = 1, l_1 = 0, l_2 = 2, l_3 = 0, l_4 = 0$

Number of possible delay sequences N_d

There are many possible delay sequences!

- N^2 choices for delay path **P**
- $\binom{s+\log N-2}{s-1}$ choices for $l_0, \ldots, l_s$

Number of possible delay sequences N_d

There are many possible delay sequences!

- N^2 choices for delay path **P**
- $\binom{s+\log N-2}{s-1}$ choices for $l_0, \ldots, l_s$
- after that $v_0, \ldots, v_s$ are completely determined and there are at most C choices for each P_i. Hence, at most C^{s+1} ways to choose $P_0, \ldots, P_s$.

Number of possible delay sequences N_d

There are many possible delay sequences!

- N^2 choices for delay path **P**
- $\binom{s+\log N-2}{s-1}$ choices for $l_0, \ldots, l_s$
- after that $v_0, \ldots, v_s$ are completely determined and there are at most C choices for each P_i. Hence, at most C^{s+1} ways to choose $P_0, \ldots, P_s$.
- $\binom{s+K}{s+1}$ ways to choose $k_0, \ldots, k_s$, $k_s \le k_{s-1} \le \ldots \le k_0$, $k_i \in [0, K]$

Number of possible delay sequences N_d

There are many possible delay sequences!

- N^2 choices for delay path **P**
- $\binom{s+\log N-2}{s-1}$ choices for $l_0, \ldots, l_s$
- after that $v_0, \ldots, v_s$ are completely determined and there are at most C choices for each P_i. Hence, at most C^{s+1} ways to choose $P_0, \ldots, P_s$.
- $\binom{s+K}{s+1}$ ways to choose $k_0, \ldots, k_s$, $k_s \leq k_{s-1} \leq \ldots \leq k_0$, $k_i \in [0, K]$
  - Why?

Combinatorial explanation

There is one-to-one correspondence between choices for k_i and $(s + K)$-bit binary string u with $s + 1$ zeros:

- k_i is the number of "1" to the left of the $(s + 1 - i)$th zero in the string $1u$

Combinatorial explanation

There is one-to-one correspondence between choices for k_i and $(s+K)$-bit binary string u with $s+1$ zeros:

- k_i is the number of "1" to the left of the $(s+1-i)$th zero in the string $1u$
- if $s+1=6, K=1, u=000110010$, then

$$1u = 1000110010$$

and $k_0 = 1, k_1 = 1, k_2 = 1, k_3 = 3, k_4 = 3, k_5 = 4$

Number of possible delay sequences N_d

$$N_d = N^2 \binom{s + \log N - 2}{s - 1} C^{s+1} \binom{s + K}{s + 1}$$

Probability to find an active delay sequence

$$N_d \Pr(r(P_i) = k_i \text{ for all } i) = N_d K^{-(s+1)}$$

Proof: results

This probability becomes smaller than $o(N^{-7})$, when the number of packets is

$$s + 1 = \begin{cases} O(C), \text{ if } C \geq \dfrac{\log N}{2} \\ O(\log N / \log\left(\dfrac{\log N}{C}\right)), \text{ if } C \leq \dfrac{\log N}{2} \end{cases}$$

Proof: final details

With probability $1 - o(N^{-7})$

$$T \leq s + \log N = \begin{cases} O(C) + \log N, \text{ if } C \geq \dfrac{\log N}{2} \\ \log N + O(\log N / \log\left(\dfrac{\log N}{C}\right)), \text{ if } C \leq \dfrac{\log N}{2} \end{cases}$$

Can we use another contention-resolution protocol?

Nonpredicting Contention-Resolution Protocols

Contention is resolved by a deterministic algorithm based on the history of contending packets, it doesn't depend on information about destinations.

Nonpredicting Contention-Resolution Protocols

Contention is resolved by a deterministic algorithm based on the history of contending packets, it doesn't depend on information about destinations.

- FIFO

Nonpredicting Contention-Resolution Protocols

Contention is resolved by a deterministic algorithm based on the history of contending packets, it doesn't depend on information about destinations.

- FIFO
- random-rank protocol is *not* non-predictive
- if we use a specific setting for random keys in RRP, it is non-predictive

History of edge activity

- R — routing problem
- Q — non-predictive contention-resolution protocol
- $H(R, Q) = \{(e, t) |$ packet traverses edge e at step $t\}$

Properties of $H(R, Q)$ (1/2)

Lemma 1. Q; R and R' with p packets per input. $H(R, Q) = H(R', Q)$ for steps in $[1, T]$ $\Rightarrow$ the location of packets after T steps of R is the same as the location of packets after T steps of R'

Proof.

Properties of $H(R, Q)$ (1/2)

Lemma 1. Q; R and R' with p packets per input. $H(R, Q) = H(R', Q)$ for steps in $[1, T] \Rightarrow$ the location of packets after T steps of R is the same as the location of packets after T steps of R'

Proof.

- $T = 0$: done

Properties of $H(R, Q)$ (1/2)

Lemma 1. Q; R and R' with p packets per input. $H(R, Q) = H(R', Q)$ for steps in $[1, T] \Rightarrow$ the location of packets after T steps of R is the same as the location of packets after T steps of R'

Proof.

- $T = 0$: done
- $T - 1 \mapsto T$:the same packets move forward the same direction for R and R'

Corollary. $R \neq R' \Rightarrow H(R, Q) \neq H(R', Q)$

Properties of $H(R, Q)$ (2/2)

Fact. Q, Q'; R with p packets per input $\Rightarrow \exists R'$ with p packets per input: $H(R, Q) = H(R', Q')$

Running time of non-predictive protocol

Theorem. $n_T(Q)$ — number of problems for which the greedy algorithm runs in T steps using Q. Then $n_T(Q) = n_T(Q')$ for any $T > 0, Q, Q'$.

Proof.

- N^{pN} different routing problems with p packets per input

Running time of non-predictive protocol

Theorem. $n_T(Q)$ — number of problems for which the greedy algorithm runs in T steps using Q. Then $n_T(Q) = n_T(Q')$ for any $T > 0, Q, Q'$.

Proof.

- N^{pN} different routing problems with p packets per input
- N^{pN} different histories

Running time of non-predictive protocol

Theorem. $n_T(Q)$ — number of problems for which the greedy algorithm runs in T steps using Q. Then $n_T(Q) = n_T(Q')$ for any $T > 0, Q, Q'$.

Proof.

- N^{pN} different routing problems with p packets per input
- N^{pN} different histories
- the set of all histories is the same for any Q' as it is for Q

Running time of non-predictive protocol

Theorem. $n_T(Q)$ — number of problems for which the greedy algorithm runs in T steps using Q. Then $n_T(Q) = n_T(Q')$ for any $T > 0, Q, Q'$.

Proof.

- N^{pN} different routing problems with p packets per input
- N^{pN} different histories
- the set of all histories is the same for any Q' as it is for Q
- each history defines the running time $\Rightarrow n_T(Q) = n_T(Q')$ for any $T > 0$

What does it mean?

- the distribution of running time T is the same for any nonpredictive protocol
- the average time is the same

Can we use another protocol?

We can set priority keys in RRP such that T will be at most $\log N + O(p) + o(\log N)$
$\Rightarrow$ greedy algorithm has the same average time T for any nonpredictive protocol.

Conclusion

- "Typical" routing problem (in a mathematical sense) is likely to have reasonable running time
- "Typical" routing problem (in practice: with bit-reversal and transpose permutations) has very bad estimation of running time