Randomized Routing on Fat-Trees

Ronald I. Greenberg
Charles E. Leiserson

MIT
**Fat-Trees**

- Concentrator switch
- 2 channels
- Processor

$n$ processors

capacity of channel $\text{cap}(c)$ = no. of wires
Universality

Any bounded-degree network can be efficiently simulated by a fat-tree of comparable hardware cost.

efficiently = polylog time degradation

hardware cost = VLSI area

Leiserson [85]: Off-line simulations

Leiserson & Greenberg: Randomized on-line simulation
Proof of Universality

Assumption: At most $O(A)$ bits can pass through a surface of area $A$ in unit time.

Communication bandwidth at $i$th level: $O\left(\left(\frac{V}{2^i}\right)^{2/3}\right)$
Universality Proof (continued)

Balance the decomposition tree.
Associate with FT with exponentially growing capacities.
Message set delivered in time $t$ by competition puts $O((t \log n) \cdot \text{cap}(c))$ messages on channel $c$. 
Load Factor

arbitrary network

message set $M$

$$\text{load}(M,c) = \# \text{messages crossing } c$$

$$\text{cap}(c) = \text{bandwidth of } c$$

$$\lambda(M) = \max_{c} \frac{\text{load}(M,c)}{\text{cap}(c)}$$

The load factor $\lambda(M)$ is a lower bound on the time to deliver $M$. 
Our Result

With high probability, e.g. $1 - O(\frac{1}{n})$, a message set $M$ can be delivered in the following no. of delivery cycles:

<table>
<thead>
<tr>
<th>load factor</th>
<th>delivery cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 \leq \lambda(M) \leq 1$</td>
<td>$O$</td>
</tr>
<tr>
<td>$1 &lt; \lambda(M) \leq \frac{\log n}{\log \log n}$</td>
<td>$O(\log n \log (\lambda(M)))$</td>
</tr>
<tr>
<td>$\frac{\log n}{\log \log n} \leq \lambda(M) \leq \text{poly}(n)$</td>
<td>$O(\lambda(M))$</td>
</tr>
</tbody>
</table>

(always $O(\lambda(M) + \frac{\log n}{\log \log n})$)

- single algorithm
- no restriction to permutation routing
- no assumptions about statistical distribution of messages

Each delivery cycle takes $\Theta(1.3^2 n)$ time.
A Communications Model

- synchronous
- bit-serial
- batched (delivery cycles)
- no buffering at switches

![Diagram]

- must drop message
- can deliver both messages in one cycle

- successful deliveries acknowledged
- perfect switches ($\log_2 n$ depth)
Routing Algorithm

Must deal with polynomial congestion.

Randomize in the choice of messages to send in each delivery cycle.

- no randomization in basic paths taken

- switch randomization unnecessary
The Basic Idea

In each delivery cycle, choose a random subset of messages to try routing.

Idea: Suppose we know $\lambda(M)$. Send each message with probability $\frac{1}{r \lambda(M)}$ for some $r > 1$.

The expected number of messages sent through channel $c$ is

$$\frac{\text{Load}(M, c)}{r \lambda(M)} \leq \frac{1}{r} \cdot \text{Cap}(c)$$

Intuition: About $\frac{M}{r \lambda(M)}$ messages will be delivered. Total no. of delivery cycles should be about $r \lambda(M)$. 
The Halving Lemma

**Lemma:** Let $U$ be the set of currently un routed messages, and suppose $\lambda(U) \leq \lambda$ initially. Then $O(\max\{\lambda, \log n\})$ cycles suffice to yield $\lambda(U) \leq \frac{1}{2} \lambda$ with high probability.
Routing with Known Load Factor

**CASE 1:** \( \lambda(M) \geq \log n \log \log n \)

\( O(\lambda(M)) \) cycles suffice

**CASE 2:** \( \lambda(M) \leq \log n \log \log n \)

\( O(\log n \log(\lambda(M))) \) cycles suffice

**Why:** Halve the load factor \( \log(\lambda(M)) \) times

**Recall:** Each halving takes

\( O(\max\{\lambda(U), \log n\}) \) cycles.

\[
\lambda(M) + \lambda(M)/2 + \lambda(M)/4 + \cdots + \log n + \log n + \cdots + \log n
\]

\( \sum \)

\( = O(\lambda(M)) \)

\( = O(\log n \log \log n) \)

**CASE 2**
Routing with Unknown Load Factor

Make guesses in two phases.

**Phase 1:** Square guesses. Work = $O(\lg n / \lg \lambda)$.  
**Phase 2:** Double guesses. Work = $O(\lambda)$.  

---  

**Guesses for $\lambda$:**  
1  
2  
4  
16  
256  

**Work:**  
1  
$\lg n$  
$2 \lg n$  
$4 \lg n$  
$8 \lg n$  

**Phase 1:**  
- $\lg n$  
- $\lg n \cdot \lg n$  
- $2 \lg n \cdot \lg n$  
- $4 \lg n \cdot \lg n$  
- $\lambda(M)$  

**Phase 2:**  
- $\lg n \cdot \lg n$  
- $\lg n \cdot \lg n$  
- $2 \lg n \cdot \lg n$  
- $4 \lg n \cdot \lg n$  
- $\lambda(M)$
The Halving Lemma

Lemma: Let $U$ be the set of currently unrouted messages, and suppose $\lambda(U) \leq \lambda$ initially.

Then $O(\max\{\lambda, \log n\})$ cycles suffice to yield $\lambda(U) \leq \frac{1}{2} \lambda$ with high probability $\left(1 - O\left(\frac{1}{\sqrt{n^2}}\right)\right)$

Idea of Proof:

Defn.: A $p$-subset of $M$ is a subset formed by independently choosing each message in $M$ with probability $p$.

Send $\frac{1}{r\lambda}$-subsets of $M$ for a suitable constant $r$. 
Proving the Halving Lemma

- Analysis of a single channel in a single cycle
- A single message in a single cycle
- A single channel over many cycles
Channel Congestion

**Lemma:** Let $M'$ be a $p$-subset of $M$. Then the probability is at most
$(e^\lambda(M))^{\text{cap}(c)}$ that a given channel $c$ is congested by $M'$, i.e. that $\text{load}(M',c) > \text{cap}(c)$.

**Proof:** Chernoff bound on binomial distribution $B(s,t,p) \leq \left(\frac{ept}{s}\right)^s \rightarrow$ probability of:
$s$ successes in $t$ independent Bernoulli trials with probability $p$ of successes.
Plug in $s \leftarrow \text{cap}(c)$, $t \leftarrow \text{load}(M,c)$. 
**Congestion Parameter**

**Definition:** The congestion parameter $r$ of a fat-tree is the smallest positive value such that for each simple path of channels $c_1, c_2, ..., c_P$ in the tree, \( \sum_{k=1}^{P} \left( \frac{e}{r} \right) \text{cap}(c_k) \leq \frac{1}{2} \).

**Note:** $r$ is constant for universal fat-trees, or for fat-trees which have capacities that are all $\Omega(\lg^{2} n)$. 
**Single Message/Single Cycle**

**Lemma:** Consider a delivery cycle in which a \( p \)-subset \( M' \) of \( M \) is sent. The probability that a given \( m \in M \) is delivered is \( \geq \frac{1}{2}p \).

**Proof:** Suppose \( m \in M' \) must go through \( c_1, c_2, \ldots, c_k \). Then the probability that at least one of these channels is congested is at most \( \sum_{k=1}^{2} \frac{1}{k} \cdot \text{cap}(c) \leq \frac{1}{2} \). Thus, \( \Pr\{m \in M \text{ delivered}\} = \Pr\{m \in M'\} \cdot \Pr\{m \text{ not lost} \mid m \in M'\} \geq p \cdot \frac{1}{2} \).
Sketch of Proof of Halving Lemma

Recall:
- $\lambda(U) \leq \lambda$ initially
- run $z = \max\{k_1 \lambda, k_2 \log n\}$ cycles
  sending $\frac{1}{r \lambda}$-subsets of $U$.
- $\lambda(U) \leq \frac{1}{2} \lambda$ at the end.

Pf. sketch:
- Consider arbitrary channel.
- Assume there are a lot of unrouted messages before each cycle.
- Show that a lot of messages get routed during the $z$ cycles.
Proof Sketch continued

- Assume there are always $\frac{1}{2} \lambda \text{cap}(c)$ undelivered messages which must pass through channel $c$.
- Expected # messages delivered on a given cycle is $\geq \frac{1}{2} \cdot \frac{1}{r} \cdot \frac{1}{2} \lambda \text{cap}(c) = \frac{\text{cap}(c)}{4r}$.
- Probability that fewer than $\frac{\text{cap}(c)}{8r}$ messages delivered on a given cycle is $< 1 - \frac{1}{8r}$. (bad cycle)
- For $z = \Omega(\log n)$, it is unlikely a large constant fraction of cycles are bad.
- For $z = \Omega(\lambda)$, $\Omega(z)$ good cycles yields $\Omega(\lambda \text{cap}(c)) = \Omega(\text{load}(U,c))$ deliveries.
Why not Greedy

**Greedy approach:** Just send all undelivered messages. Keep resending any messages which do not reach their destinations.

**Switches:**
- greedy: drop minimum number of messages
- oblivious: decisions on what to drop based on no knowledge of message set other than presence of messages on input lines.

Sending more messages doesn't always help.

Greedy strategy requires $\Omega(\lambda(M) \log n)$ cycles.
Other Results

• Good off-line schedules exist

• If the channel capacities are all \( \Omega(\log n) \), then \( O(\lambda(M)) \) cycles suffice with high probability

• Improved fat-tree
Research Directions

1. Greedy strategy
2. Lower bounds
3. Different communication models
4. Different ways of handling congestion
5. Comparison with other routing schemes
6. Better universal networks