Online Routing of Virtual Circuits

– CT Network Theory Seminar –

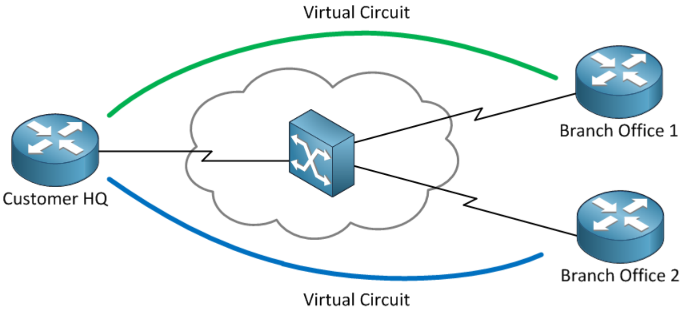
Wenkai Dai

Communication Technologies Group
Computer Science Department
University of Vienna

April 12, 2021

Outlines

Introduction



- Permanent: no rerouting once a route assigned unless failures

The Online Virtual Circuits Routing Problem

Input:

- An edge-weighted graph $G = (V, E, u)$, where $|V| = n$, $|E| = m$, and a capacity function $u : E \mapsto \mathbb{R}^+$;
- Each request r_i: $r_i = (s_i, t_i, p(i))$, is to request a path from s_i to t_i in G using bandwidth $p(i)$;
- Normalization: for each request r_i, $\forall e \in E$, $p_e(i) = \frac{p(i)}{u(e)}$;
- *A sequence of requests*: $\sigma = (r_1, r_2, \ldots, r_k)$.

Definitions:

- Online algorithm $\mathscr{A}$ assigns routes $\mathscr{P} = \{P_1, P_2, \ldots, P_k\}$;
- Offline algorithm $\mathscr{A}^*$ assigns routes $\mathscr{P}^* = \{P_1^*, P_2^*, \ldots, P_k^*\}$;
- Given a set of routes $\mathscr{P}$, *relative load* after the first j requests:

$$\ell_e(j) = \sum_{\substack{i:e\in P_i \\ i\le j}} p_e(i), \qquad \ell_e^*(j) = \sum_{\substack{i:e\in P_i^* \\ i\le j}} p_e(i)$$

The Online Virtual Circuits Routing Problem (2)

Definitions:

- Online algorithm $\mathscr{A}$ assigns routes $\mathscr{P} = \{P_1, P_2, \ldots, P_k\}$;
- Offline algorithm $\mathscr{A}^*$ assigns routes $\mathscr{P}^* = \{P_1^*, P_2^*, \ldots, P_k^*\}$;
- Given a set of routes $\mathscr{P}$, *relative load* after the first j requests:

$$\ell_e(j) = \sum_{\substack{i:e\in P_i \\ i\leq j}} p_e(i), \qquad \ell_e^*(j) = \sum_{\substack{i:e\in P_i^* \\ i\leq j}} p_e(i);$$

- Maximum Load:

$$\lambda(j) = \max_{e\in E} \ell_e(j), \qquad \lambda^*(j) = \max_{e\in E} \ell_e^*(j);$$

- Abbreviations: $\lambda = \lambda(k)$ and $\lambda^* = \lambda^*(k)$.

Objective of $\mathscr{A}$: Minimize λ/λ^*

Algorithm 1 Algorithm ASSIGN-ROUTE

procedure ASSIGN-ROUTE($p, s, t, G, \vec{\ell}, \Lambda, \beta$)
  /*Λ: current estimate of λ^*;
  /*β: designed performance guarantee of the algorithm;
  $\forall e \in E, p_e := \frac{p}{u(e)}$; ▷ Normalize each given bandwidth
  $\forall e \in E, \tilde{p}_e := \frac{p_e}{\Lambda}$; ▷ Normalization by Λ
  $\forall e \in E, \tilde{\ell}_e := \frac{\ell_e}{\Lambda}$; ▷ Normalization by Λ
  $\forall e \in E : c_e := a^{\tilde{\ell}_e + \tilde{p}_e} - a^{\tilde{\ell}_e}$;
  Let P be the shortest path from s to t in G w.r.t. costs c_e;
  if $\exists e \in P : \ell_e + p_e > \beta\Lambda$ **then**
    $b := fail$
  else
    $\forall e \in P : \ell_e := \ell_e + p_e$; $b := success$;
return $(P, \vec{\ell}, b)$;

Competitive Ratio Analysis for ASSIGN-ROUTE

Theorem 1.1

If $\lambda^* \leq \Lambda$, then there exits $\beta = O(\log n)$ such that Algorithm ASSIGN-ROUTE *never fails. Thus, the relative load on an edge never exceeds $\beta\Lambda$*

Proof.

- Assume we know Λ, s.t., $\lambda^*(j) \leq \lambda^* \leq \Lambda$
- Consider state after j request, where $1 \leq j \leq k-1$
- Define potential function:

$$\Phi(j) = \sum_{e \in E} \alpha^{\tilde{\ell}_e(j)} \left(\gamma - \tilde{\ell}_e^*(j)\right), \text{ where constants: } \alpha, \gamma > 1;$$

- First to show Φ is not increasing if $\lambda^* \leq \Lambda$

Competitive Ratio Analysis for Assign-Route

Proof (Cont.)

- Let online (offline) algorithm assign the route P_{j+1} (P^*_{j+1}) to the $(j+1)$-th request;

$$\Phi(j) = \sum_{e \in E} \alpha^{\tilde{\ell}_e(j)} \left(\gamma - \tilde{\ell}^*_e(j)\right)$$

- Compute $\Phi(j+1) - \Phi(j)$ as follows:

$$\sum_{e \in P_{j+1}} \left(\gamma - \tilde{\ell}^*_e(j)\right)\left(\alpha^{\tilde{\ell}_e(j+1)} - \alpha^{\tilde{\ell}_e(j)}\right) - \sum_{e \in P^*_{j+1}} \alpha^{\tilde{\ell}_e(j+1)} \tilde{p}_e(j+1)$$

- By $(x + \Delta(x))(y + \Delta(y)) = y\Delta(x) + (x + \Delta(x))\Delta(y)$
- Let $x = \alpha^{\tilde{\ell}_e(j)}$, $y = (\gamma - \tilde{\ell}^*_e(j))$
- $\Delta(x)$ and $\Delta(y)$ denote the changed values of x and y between $\Phi(j)$ and $\Phi(j+1)$.

Competitive Ratio Analysis for Assign-Route

Proof (Cont.)

- Let online (offline) algorithm assign the route P_{j+1} (P^*_{j+1}) to the $(j+1)$-th request;
- Bound $\Phi(j+1) - \Phi(j)$ as follows:

$$\sum_{e \in P_{j+1}} \left(\gamma - \tilde{\ell}^*_e(j)\right)\left(\alpha^{\tilde{\ell}_e(j+1)} - \alpha^{\tilde{\ell}_e(j)}\right) - \sum_{e \in P^*_{j+1}} \alpha^{\tilde{\ell}_e(j+1)} \tilde{p}_e(j+1)$$

$$\leq \sum_{e \in P_{j+1}} \gamma\left(\alpha^{\tilde{\ell}^*_e(j)+\tilde{p}_e(j+1)} - \alpha^{\tilde{\ell}_e(j)}\right) - \sum_{e \in P^*_{j+1}} \alpha^{\tilde{\ell}_e(j)} \tilde{p}_e(j+1)$$

$$\leq \sum_{e \in P^*_{j+1}} \left(\gamma\left(\alpha^{\tilde{\ell}^*_e(j)+\tilde{p}_e(j+1)} - \alpha^{\tilde{\ell}_e(j)}\right) - \alpha^{\tilde{\ell}_e(j)} \tilde{p}_e(j+1)\right)$$

$$= \sum_{e \in P^*_{j+1}} \alpha^{\tilde{\ell}_e(j)} \left(\gamma\left(\alpha^{\tilde{p}_e(j+1)} - 1\right) - \tilde{p}_e(j+1)\right).$$

Competitive Ratio Analysis for Assign-Route

Proof (Cont.)

$$\Phi(j+1) - \Phi(j) \leq \sum_{e \in P^*_{j+1}} \alpha^{\tilde{\ell}_e(j)} \left(\gamma \left(\alpha^{\tilde{p}_e(j+1)} - 1 \right) - \tilde{p}_e(j+1) \right)$$

- Offline assigns the route P^*_{j+1} for the $(j+1)$-th request, then
$$\forall e \in P^*_{j+1} : 0 \leq \tilde{p}_e(j+1) \leq \lambda^* / \Lambda \leq 1.$$
- To prove $\Phi(j+1) - \Phi(j) \leq 0$, we need to show $\gamma(\alpha^x - 1) \leq x$ for $x \in [0, 1]$. It is true for $\alpha = 1 + 1/\gamma$
- Clearly, $\Phi(0) = \gamma m$;
- Recall $\Phi(j) = \sum_{e \in E} \alpha^{\tilde{\ell}_e(j)} \left(\gamma - \tilde{\ell}^*_e(j) \right)$ and $\tilde{\ell}^*_e(j) \leq 1$
- Thus, $\Phi(j) = \sum_{e \in E} \alpha^{\tilde{\ell}_e(j)} \left(\gamma - \tilde{\ell}^*_e(j) \right) \leq \gamma m$
- Since $\gamma > 1$, then
$\max_{e \in E} \ell_e(k) \leq \Lambda \cdot \log_a \left(\frac{\gamma m}{\gamma - 1} \right) = O(\Lambda \log n)$.

□

Competitive Ratio Analysis for Assign-Route

- How to decide Λ: easy to approximate Λ by at most four;
- First stage: $\Lambda = \min_e \tilde{p}_e(1) = \min_e p(1)/u(e)$
- Iterates: A new stage starts when Assign-Route fails, and double the value of Λ and ignore all requests routed in previous phases.
- In the final stage: we can have $\Lambda \leq 4\lambda^*$

Theorem 1.1

Algorithm Assign-Route *can achieve* $O(\log n)$*-competitive ratio with respect to load.*

Lower Bound $\Omega(\log n)$ for Routing

- Directed network: $\exists u, v \in V$, s.t., $c(u, v) \neq c(v, u)$;
- Lower bound $\Omega(\log n)$: there exists a special case for directed network, where the load of online algorithm cannot be better than optimal algorithm by the factor $\log n$
- Oblivious adversary: generate requests independently of the outcome of online algorithm
- The lower bound also holds randomized algorithm against an oblivious adversary

Proof of Lower Bound $\Omega(\log n)$

- The source $s \in V$, and n nodes $V' = \{v_1, \ldots, v_n\}$, s.t., $(s, v_i) \in E$, where $v_i \in V'$;
- For each $1 \le i \le \log n$, divide V' to 2^{i-1} sets $V_{i,j}$:
  - for each $j = 1, \ldots, 2^{i-1}$, there is a sink $S_{i,j}$, and each node in the current set $V_{i,j}$ has a link to $S_{i,j}$
- Each link has a unit capacity.
- For each phase $1 \le i \le \log n$, $n/2^i$ requests from s to a sink $S_{i,j}$ for $1 \le j \le 2^{i-1}$ and each request needs a unit bandwidth.
- To show: online algorithm causes $\log n/2$ load on some edge (s, v_j), where $v_j \in V'$, but optimal (offline) algorithm has the load one on all edges

Proof of Lower Bound $\Omega(\log n)$ (2)

- Offline: there are at most n requests, we could find n edge-disjoint paths.
- Claim for online: at the end of phase i, for the sink $S_{i,j}$, the average of expected load for (s, v_l), where $v_l \in V_{i,j}$, is $\geq i/2$
    - For $1 \leq i \leq \log n$, it holds for $i = 1$
    - Assume it holds for the phase i, s.t., the average expected load for nodes in $V_{i,j}$ is $i/2$, where $|V_{i,j}| = n/2^{i-1}$.
    - Now, at the phase $i+1$, let $V_{i,j} = V_{i+1,j'} \cup V_{i+1,j'+1}$, where $|V_{i+1,j'}| = |V_{i,j}|/2 = n/2^i$;
    - There are $n/2^{i+1}$ requests for nodes in $V_{i+1,j'}$ (sink $S_{i+1,j'}$)
    - Thus, the average expected load for nodes in $V_{i+1,j'}$: $i/2 + 1/2 = (1+i)/2$ (claim is true)
    - Finally, for the last phase $i = \log n$, then the load is at last $\log n/2$

Doubling Approach

Theorem 3.1

For any load balancing problem, let ALG_Λ *be a parameterized online algorithm satisfying* $\mathrm{OPT}(\sigma) \leq \Lambda \implies \mathrm{ALG}_\Lambda(\sigma) \leq c \cdot \Lambda$*. Then there is an algorithm* ALG *s.t., for all* σ*,*
$\mathrm{ALG}(\sigma) \leq 4c \cdot \mathrm{OPT}(\sigma)$*.*

- ALG executes in stages, each stage correspond to the most recent estimate of Λ.
- Stage 0, $\Lambda_0 = \mathrm{OPT}(j = 0)$, easy to compute the optimal for the first job.
- Each stage j, ALG uses ALG_Λ to assign jobs until it fails and start a new stage by doubling Λ (ignoring previous stages for assigning jobs)
- Stage k, $\Lambda = 2^k \Lambda_0$

Proof of Doubling Approach

Proof.

- To prove $\mathrm{ALG}(\sigma) \leq 4c \cdot \mathrm{OPT}(\sigma)$ for any sequence σ
- Suppose ALG terminates at the stage h.
- If $h = 0$, it is clear $\mathrm{ALG}(\sigma) \leq c \cdot \mathrm{OPT}(\sigma)$
- Let r be the first job for the stage h, and σ_j denotes the subsequence processed in stage j
- Clearly, stage $h-1$ failed on $\sigma_{h-1}r$, while ALG_{Λ} has $\Lambda = 2^{h-1}\Lambda_0$
- Thus, $\mathrm{OPT}(\sigma) \geq \mathrm{OPT}(\sigma_{h-1}r) > 2^{h-1}\Lambda_0$
- Moreover

$$\mathrm{ALG}(\sigma) = \sum_{j=0}^{h} \mathrm{ALG}(\sigma_j) \leq \sum_{j=0}^{h} c \cdot 2^j \Lambda_0 = c\left(2^{h+1} - 1\right)\Lambda_0 \quad \square$$

Questions & Answers

Thanks for your Listening, and welcome your questions!