DISTRIBUTED ALGORITHM AND REVERSIBLE NETWORK

SHREEVATSA RAJAGOPALAN AND DEVAVRAT SHAH

ABSTRACT. Motivated to design a feasible optical network architecture for the future Internet, we address the question of scheduling (wavelength assignment) in an optical network. The key challenge in designing a scheduling algorithm lies in solving a combinatorial optimization problem under very stringent distributed constraints. Specifically, given R random variables $x_1, \ldots, x_R$ taking integer values, each variable is required to find its assignment so that collectively they maximize $\sum_r W_r x_r$ subject to some linear constraints that $Ax \leq C$. In doing so, each variable can only avail of two pieces of information (and nothing else): one, its own weight W_r, two, given its current values (and the values of other variables being unknown), can it increase it by $+1$ or not. As the main result of this paper, we present a randomized algorithm that solves this problem. Our algorithm builds upon classical theory of reversible networks. To the best of our knowledge, this is the first such algorithm for such a stringent distributed problem.

1. INTRODUCTION

The future Internet will be subjected to extremely stringent demand in terms of bandwidth by data-mongering, delay-sensitive next-generation applications. This necessitates the use of optical technology for network design, given the limitations of a purely electronic network. However, slow electronic memory and the lack of progress in designing all-optical routers suggest that a packet optical architecture is highly infeasible. On the other hand, the current electronic packet network architecture has been reasonably serving our purpose for current needs. Therefore, a canonical future network architecture must be a "hybrid" between an extremely high-bandwidth "optical core network" and a relatively low-bandwidth "electronic edge network". The electronic network works efficiently with packet-switched architecture; the only feasible way to operate an optical network seems to be a circuit-switched architecture. Therefore, in order to design such a hybrid opto-electronic architecture, it is necessary to design an "interface" between the packet-switched electronic and circuit-switched optical networks. In this paper, we consider a natural architectural solution for such an "effective interface" so that current end-to-end protocols, such as the congestion control protocol (TCP), operate seamlessly.

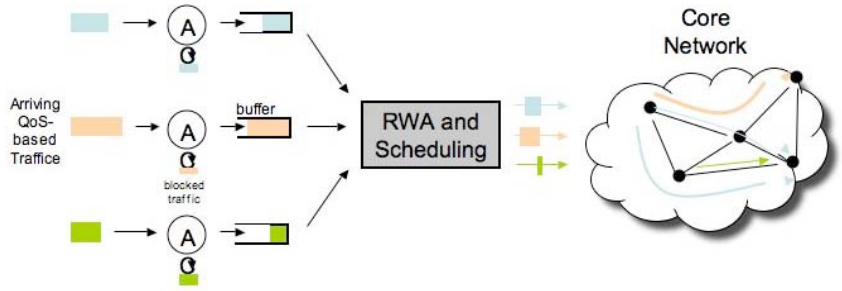

FIGURE 1. The hybrid interface between electronic edge and optical core network.

Both authors are at MIT. SR is with ORC and DS is with LIDS and EECS. Email: {vatsa, devavrat}@mit.edu. This work was supported by NSF FIND grant for Optical network design.

978-1-4244-2247-0/08/$25.00 ©2008 IEEE.

Figure 1 presents an illustration of the interface: packets arriving through electronic networks are gathered at the ingress of optical networks forming a reasonably long "flow"; the flows are buffered at the ingress port after admission control, which depends on the traffic shaping and buffer management schemes; the scheduling algorithm decides which flow gets to transfer its data by using available wavelengths along the path given by the routing mechanism. The three main algorithmic parts of this interface are: (a) admission control, (b) routing, and (c) scheduling. The admission control is likely to follow as a 'statistical police' given the routing and scheduling. The routing operates at a relatively much slower time-scale than scheduling, as the aggregate demands between various source-destination pairs (in the core network there are not too many such pairs and hence aggregation takes place) change slowly. Therefore, routing is based on learning statistics and using an offline optimization algorithm. On the other hand, the scheduling algorithm has to operate at the actual time-scale in order to react to the dynamics. The scheduling problem in such a setup is very challenging because: (i) an efficient scheduling algorithm boils down to solving a combinatorial (integer) optimization problem, and (ii) each ingress node needs to find this solution in a totally distributed manner, that is, each ingress node can only obtain information from the network as to whether wavelength on a certain path is available or not, but it can not obtain any information about other nodes' assignment or even their objective. Therefore, even finding a solution for this problem under above stringent constraints, even without worrying about efficiency, is challenging.

To the best of our knowledge, there is no solution known in the literature across communities: distributed algorithms, optimization and networks. As the main result of this paper, we design a novel distributed algorithm for solving this problem using the theory of *reversible networks*. We believe that the algorithm utilizes the network capacity to the fullest, and that it performs well in terms of the resulting buffer-sizes remaining small. A salient feature of our algorithm is the possibility of trading off the performance of the algorithm with the frequency of querying the network for availability—frequency of querying network is proportional to the control-information overhead in the operation of the network. Thus, our scheduling algorithm provides: (a) an efficient distributed solution, (b) tunability in terms of control-information and performance, and (c) the first ever solution to an important distributed algorithms problem.

1.1. **Organization.** In Section 2, we formulate the underlying mathematical problem and the necessary algorithmic problem that needs to be solved in distributed manner. In Section 3, we present a distributed algorithm for solving this problem. We present the proof of its correctness and computation time as well. In Section 4, we briefly describe how this algorithm can be used for scheduling.

2. Problem formulation

The network is a graph $G = (V, E)$. There are R routes, given by a $R \times |E|$ matrix A indicating the number of times a particular link occurs in a particular route. Arrivals are modeled as independent Poisson processes with arrival rate ν_r on route r. Each call simultaneously holds all links on its route, and holds them for a period that is exponentially distributed with unit mean. Calls are buffered at the ingress. A call can access the network if and only if it would not exceed the capacity of any link on its route.

We seek to devise a *scheduling policy*: for the buffers to make decisions, possibly based on queue sizes. A buffer may request to send one unit call on a particular route. If the request is granted, the buffer sends a call through. Our notion of cost here is the number of requests made. Note that each buffer has extremely limited information: the only information comes from queries about whether the network can accommodate an additional call on this route or not.

Following the classical "max-weight" philosophy would indicate that the set of routes served should be that which maximizes the quantity $\sum_r x_r f(B_r)$ where B_r is the size of buffer r, for some appropriate function f. This is an integer optimization problem, with an additional constraint enforced by the limited nature of the information available to each buffer. Thus we are led to a distributed optimization problem of the following form, which we shall call D-OPT:

D-OPT: Given variables $(x_1, \ldots, x_R)$; associated weights $(W_1, \ldots, W_R)$.
Find x that solves

$$\max \sum W_r x_r$$

such that

$$x \in \mathcal{X} = \left\{x \in \mathbb{N}^R \colon Ax \leq C\right\}$$

with the constraint that it must be done in a purely *distributed* way: At any instant, each variable x_r can only obtain information (from an oracle) as to whether it can itself be increased by 1 or not, but has *no* other information, such as the values of the other variables.

3. Algorithm

3.1. **Description.** We describe a continuous-time algorithm for D-OPT. In essence, we simulate the network. Each variable x_r has an independent Poisson clock, whose rate depends on W_r. Namely, it is $\exp(NW_i)$ for some large N. Each time the clock ticks, the oracle is asked whether the variable can be increased by 1. If it can, the variable is increased, else nothing happens. The variable is then decreased by 1 after a time that is exponentially distributed with unit mean. (This is analogous to the call entering and leaving the network.) We will prove that if this algorithm is run for a sufficiently long time, the solution to the optimization problem can then be simply read off as the values of the variables at that instant:

Theorem 1. *Let x^* be the solution to* D-OPT*, i.e., $x^* \in \arg\max(W^T x)$. Let $x(t) = (x_1(t), \ldots, x_r(t))$ be the state of the system at time*[1] *t. Then, for any t such that $t > \exp(7N \sum_r W_r x_r^*) \log(1/\delta)$,*

$$Pr[W^T x(t) \leq W^T x^* - \epsilon] \leq \delta + \frac{\log |\mathcal{X}|}{N\epsilon}$$

The above result suggests that by selection of $N = \log |\mathcal{X}| \delta^{-1} \epsilon^{-1}$, the algorithm finds solution within ϵ additive error of the optimal solution with probability at least $1 - 2\delta$. Note that the parameter, N of the algorithm is *independent* of weights W but depends only on the total problem size $|\mathcal{X}|$.

3.2. **Correctness.** We note some properties of the stationary distribution.

Lemma 1. *Suppose a function T is defined on a set $\mathcal{X}$. For any probability distribution μ on $\mathcal{X}$, define $F(\mu) = \mathrm{E}_\mu[T(x)] + H(\mu)$. Then, $F(\cdot)$ is uniquely maximized by the distribution π, given by*

$$\pi(x) = \frac{1}{Z} \exp(T(x))$$

where $Z = \sum_{x \in \mathcal{X}} \exp(T(x))$ is the appropriate constant of proportionality.

Proof. For any distribution μ,

$$\begin{aligned} F(\mu) = \mathrm{E}_\mu[T(x)] + H(\mu) &= \sum_x \mu(x) T(x) - \sum_x \mu(x) \log \mu(x) \\ &= \sum_x \mu(x)(\log \pi(x) + \log Z) - \sum_x \mu(x) \log \mu(x) \\ &= (\log Z) \sum_x \mu(x) + \sum_x \mu(x) \log \frac{\pi(x)}{\mu(x)} \\ &\leq \log Z + \log \sum_x \mu(x) \frac{\pi(x)}{\mu(x)} = \log Z \end{aligned}$$

where we use the concavity of the log function, with equality holding only when $\mu = \pi$. □

As a sanity check, note that when $T(x) = 0$, Lemma 1 says that the uniform distribution has the maximum entropy among all distributions on $\mathcal{X}$.

[1] Here, we measure time in terms of number of Poisson clock ticks.

Lemma 2. *Let $Z = \sum_{x\in\mathcal{X}} \exp(T(x))$ as above. Let f be any function we want to maximize on $\mathcal{X}$. Let $f(x^*)$ be the maximum possible value of $f(x)$, and let $K_\epsilon = \{x \in \mathcal{X}\colon f(x) \le f(x^*) - \epsilon\}$. Let π_N be the distribution given by $\pi_N(x) \propto \exp(T(x) + Nf(x))$ Then,*

$$\pi_N(K_\epsilon) \le \frac{\log Z}{\epsilon N}$$

Proof. Let μ^* be the distribution that assigns all probability on x^*, i.e.,

$$\mu^*(x) = \begin{cases} 1 \text{ if } x = x^* \\ 0 \text{ otherwise} \end{cases}$$

We know that π_N maximizes $\mathrm{E}_\mu[T(x) + Nf(x)] + H(\mu) = F(\mu) + N\,\mathrm{E}_\mu[f(x)]$, by lemma 1. Thus,

$$F(\mu^*) + Nf(x^*) \le F(\pi_N) + N\,\mathrm{E}_{\pi_N}[f(x)] \le F(\pi_N) + N(f(x^*) - \epsilon\pi_N(K_\epsilon)) \text{ and hence}$$

$$\pi_N(K_\epsilon) \le \frac{F(\pi_N) - F(\mu^*)}{\epsilon N} \le \frac{F(\pi)}{\epsilon N}$$

□

With these lemmas in hand, we now turn to our algorithm. The evolution of the "state" $(x_1, x_2, \dots)$ in our algorithm can be viewed as a Markov process, analogous to that in a loss network whose routes and capacities are given by A and C, so that the feasible states are $\{x \in \mathbb{N}^R\colon Ax \le C\}$, precisely the feasible set $\mathcal{X}$ for our problem D-OPT. (This is no surprise, as we formulated our problem based on a network of that form.) A loss network has independent Poisson arrival processes along each route, and call holding times are independently and exponentially distributed with unit mean.

From the theory of reversible networks ([1], [2]), we know that such a loss network is a reversible Markov process, with stationary probability of any state x being $\pi(x) \propto \prod_r \nu_r^{x_r}/x_r!$ where ν_r are the arrival rates.

Here, as the arrival (request) rates are $\nu_r = \exp(NW_r)$, the process has the stationary distribution

$$\pi_N(x) \propto \prod_r \frac{\exp(NW_r x_r)}{x_r!} = \exp(T(x) + Nf(x)) \tag{1}$$

where $T(x) = \log(\prod_r \frac{1}{x_r!})$ and $f(x) = \sum_r W_r x_r$ is precisely the objective function we wish to maximize.

Hence from the previous lemma, we can conclude the following:

Lemma 3. *Let $K_\epsilon = \{x\colon \sum W_r x_r \le \sum W_r x_r^* - \epsilon\}$, and let π_N be the stationary distribution of our algorithm. Then,*

$$\pi_N(K_\epsilon) \le \frac{\log|\mathcal{X}|}{\epsilon N}$$

Proof. $\exp(T(x)) = \prod_r \frac{1}{x_r!} \le 1$ for every x, so $Z = \sum_{x\in\mathcal{X}} \exp(T(x)) \le |\mathcal{X}|$, and it follows from lemma 2. □

3.3. **Cost of the algorithm.** We now give a (very loose) upper bound on the mixing time, and hence on the cost, of the algorithm.

In the Markov process that arises from our algorithm, the stationary probability $\pi_N(x)$ of a particular state $x = (x_1, \dots, x_i, \dots, x_R)$ is as given in equation 1. It transitions to a state like $(x_1, \dots, x_i + 1, \dots, x_R)$ at a rate proportional to $\nu_i = \exp(NW_r)$, and to a state like $(x_1, \dots, x_i - 1, \dots, x_R)$ at a rate proportional to x_i. Call the transition rate between states x and y as $q(x, y)$.

In order to apply known results about reversible discrete-time Markov chains, we look at the "jump chain" of the Markov process. This is a discrete-time Markov chain on the same set of states $\mathcal{X}$, with transition probabilities $p(x, y) = \frac{q(x,y)}{q(x)}$ where $q(x) = \sum_y q(x, y)$. With these transition probabilities, the Markov chain is reversible, with stationary distribution given by

$$\pi^J(x) = \frac{\pi(x)q(x)}{\sum \pi(x)q(x)}$$

Thus, for any two states x and y,

$$Q(x, y) = \pi^J(y)p(y, x) = \pi^J(x)p(x, y) = \frac{\pi(x)q(x)}{\sum \pi(x)q(x)} \frac{q(x, y)}{q(x)}$$

Define

$$h = \min_{\pi(S)\leq\frac{1}{2}} \frac{Q(S,S^c)}{\pi^J(s)} \quad \text{where} \quad Q(S,S^c) = \sum_{x\in S, y\in S^c} Q(x,y)$$

We know that the second largest eigenvalue β_1 of P satisfies $\beta_1 \leq 1 - \frac{h^2}{2}$. We therefore want a lower bound on h. We state a number of steps below leading eventually to a (loose) lower bound:

1. $\prod_r \frac{\exp(NW_r x_r)}{x_r!} \leq \prod_r \exp(NW_r x_r) = \exp(N\sum_r W_r x_r) \leq \exp(N\sum_r W_r x_r^*)$.
2. So $\sum_{x\in\mathcal{X}} \prod_r \frac{\exp(NW_r x_r)}{x_r!} \leq |\mathcal{X}| \exp(N\sum_r W_r x_r^*)$.
3. $\prod_r \frac{\exp(NW_r x_r)}{x_r!} \geq \prod_r \frac{1}{x_r!} \geq exp(-\sum_r x_r \log x_r) \geq \exp(-2\sum_r x_r)$.
4. $q(x,y) \geq 1$ as it is either x_i or $\exp(NB_i)$ for some i.
5. So $\pi(x)q(x,y) \geq \exp(-2\sum_r x_r)/(|\mathcal{X}| \exp(N\sum_r W_r x_r^*))$.
6. Going the other way, $\pi(x) \leq 1$.
7. And $q(x) = \sum_r q(x,y) \leq 2R \max \exp(NW_r)$.
8. So $\sum_x \pi(x)q(x) \leq |\mathcal{X}|(2R \max \exp(NW_r))$.

Finally, putting it all together,

$$h \geq \min_S Q(S,S^c) \geq \min_{x,y} Q(x,y) \geq \frac{1}{\exp(3N\sum_r W_r x_r^*)}$$

for sufficiently large N. Therefore ([3]), the δ-mixing time of the process is bounded above as

$$\tau(\delta) \leq \exp(7N \sum_r W_r x_r^*) \log(\frac{1}{\delta}).$$

This means that, after time $\tau(\delta)$, the probability of any event is at least the probability under stationary distribution minus δ. This will immediately imply the time bound and probability bound as claimed in Theorem 1. This completes the proof of Theorem 1.

4. Scheduling in optical networks

Having solved the optimization problem in a distributed setup, we return to the setting that motivated it. In the original network setting, the distributed constraint corresponds naturally to the nature of the network, and the oracle that each buffer requests for information is the network itself.

We are thus able to induce the network to be in feasible states that maximizes $\sum_r x_r W_r$, for weights W_r of our choosing. In particular, we can take the W_r to be some increasing function of (B_r), the size of the buffer at a particular instant. The rates of the Poisson clocks change whenever the size of the queue changes. We believe that, as in the maximum-weight scheduling paradigm, tuning the serving policy in this manner will lead to stable scheduling.

In particular, with W_r a function like $\log\log B_r$, the mixing time is $o(B_r)$. When the buffer sizes are large, this means that the change in the sizes while the algorithm approaches stationarity is relatively insignificant. Thus, we strongly believe that this constitutes a stable scheduling policy. Completing this proof will be our immediate future work.

References

1. F.P. Kelly, *Reversibility and stochastic networks*, Wiley, London, 1979.
2. ———, *Loss networks*, Annals of Applied Probability (1991), pp. 319–378.
3. R. Montenegro and P. Tetali, *Mathematical aspects of mixing times in markov chains*, Found. Trends Theor. Comput. Sci. **1** (2006), no. 3, 237–354.