Distributed and Adaptive Routing Based on Game Theory

Baptiste Jonglez, Bruno Gaujal

Routage distribué et adaptatif fondé sur la théorie des jeux

Baptiste Jonglez[1] et Bruno Gaujal[1]

[1] *Univ. Grenoble Alpes, Inria, CNRS, LIG, F-38000 Grenoble France*

Dans cet article, nous présentons un algorithme de routage distribué multi-flots permettant de choisir des chemins de bout en bout dans un réseau. Celui-ci converge vers une configuration dans laquelle aucun flot ne peut améliorer son délai de bout en bout en changeant de chemin (équilibre de Nash).

Notre algorithme est robuste aux erreurs de mesures, tolère les mesures obsolètes, et ne nécessite pas de synchronisation d'horloge. Notre preuve de concept est implémentée sous forme d'un contrôleur OpenFlow, et nous l'évaluons sur une plate-forme d'émulation utilisant Mininet.

1 Introduction

A common challenge in current and future communication networks is to exploit path diversity to increase performance and reliability. This problem can be modeled as a *multi-commodity flow problem* [1]. Given a number of concurrent source-destination flows, the problem is to assign these flows to network paths, while respecting capacity constraints and optimizing a performance metric. Our goal is to provide a *distributed* solution to this problem, that requires neither cooperation between sites nor knowledge of the network topology and performance parameters.

Adaptive routing algorithms base the routing decisions on dynamic properties of the network (such as real time link load, end-to-end latency, or packet loss). Despite years of research, adaptive routing techniques did not see wide adoption. The main reason is the presence of potential instabilities and routing oscillations, which could make the cure worse than the disease. Indeed, early experiments with delay-based routing in the ARPANET resulted in severe stability issues under high load, rendering the network close to unusable [2].

Contributions In this work, we present a novel algorithm for adaptive routing in packet-switched networks, mapping source-destination flows to paths in the context of *atomic non-splittable routing games*. We claim that it provides a viable and stable solution to adapt to traffic conditions, and effectively avoids congestion. Our algorithm is based on strong theoretical grounds from game theory, while our proof-of-concept uses Openflow to ease implementation. Our routing algorithm is endowed with the following desirable properties for efficient implementation:
1) It is fully distributed: only local information is needed, and it requires no explicit coordination between routers. 2) It is oblivious to the network topology. 3) It is robust to outdated and noisy measurements. 4) There are no endless oscillations. 5) It does not require clock synchronization between routers, or between routers and end hosts.
Proofs and additional material are available in a companion research report [3].

2 Distributed Routing over a Network

Let (V,E) be a communication network over a set V of nodes and a set E of bi-directional links, over which we consider the following *multi-commodity flow problem*. A set $\mathcal{K}$ of *flows* of packets must be routed over the network. Each flow $k \in \mathcal{K}$ is characterized by a source a_k, a destination b_k and a nominal arrival rate of packets, λ_k. Also, each flow is affected a set $\mathcal{P}_k$ of possible paths in the network from its source to its destination, with $|\mathcal{P}_k| = P_k$. A *configuration* is a choice of one path per flow. Our objective is to find a configuration that minimizes the *end-to-end average delay* of each flow.

We design a *learning algorithm* that allows each flow to discover a path, such that the global configuration is a Nash equilibrium of the system: no flow can improve its delay by changing its path. The challenge is to consider a realistic scenario in which no flow has information about the choices of the others or even knows about their presence. The only information that a flow can get from the network is a measure of the end-to-end delay of the packets it sends over its current path.

The Optimal Path Selection (OPS) Algorithm 1 is based on a mirror-descent algorithm for general potential games, presented in [4]. For one flow, say k, we call $d_k(p_1,\cdots,p_k,\cdots,p_K)$, the end-to-end *average delay* for packets of flow k under the configuration where flow 1 uses path p_1 among its possible paths, flow 2 uses path p_2, and so forth. The algorithm is probabilistic and maintains two vectors, both of size P_k (denoted P).

The *probabilistic choice* vector $\mathbf{q} = (q_1 \dots q_P)$ gives the probability to choose each path p.

The *score vector* $\mathbf{Y} = (Y_1 \dots Y_P)$ maintains a (negative) score for each path, to be optimized, where Y_p depends on the average delay for packets of flow k on path p.

Algorithm 1: Optimal Path Selection (OPS) Algorithm for flow k.

Initialize: $n \leftarrow 0$; $\mathbf{q} \leftarrow (\frac{1}{P},\dots,\frac{1}{P})$; $\mathbf{Y} \leftarrow (0,0,\dots,0)$;
repeat
  When local timer ticks for the nth time;
  Select a new path p w.r.t. probabilities $\mathbf{q}$;
  Send packets on path p and measure their delay D;
  Update score of path p: $Y_p \leftarrow \left(Y_p - \gamma_n(D+\tau Y_p)/q_p\right) \vee \beta_n$;
  foreach *path* $s \in \mathcal{P}_k$ **do**
    update probability: $q_s \leftarrow \frac{\exp(Y_s)}{\sum_\ell \exp(Y_\ell)}$;
until *end of time*;

The OPS algorithm uses 3 parameters: $\tau > 0$ is a discounting factor over past scores. The bounding sequence β_n (in the algorithm, $\vee$ denotes the maximum operator) is such that $\beta_n \to -\infty$ and $|\beta_n| \leqslant C_1 n + C_2$ for some constants C_1 and C_2. The decreasing sequence of discretization steps γ_n is in L_2 ($\sum_n \gamma_n^2$ converges), but not in L_1 ($\sum_n \gamma_n$ diverges). Typically, $\gamma_n = 1/n^\alpha$ with $1/2 < \alpha \leqslant 1$ works.

Theorem 1 (Convergence to equilibrium)**.**
If delay measurements have no bias and all flows have the same rate[†]*, for all* $\varepsilon > 0$*, there exists* $\tau > 0$ *such that under discounting factor* τ*, Algorithm 1 converges for all flows to an* ε*-optimal configuration, in the following sense:*
For each flow k, the probability vector $\mathbf{q}$ *converges almost surely to a near degenerate probability:* q_p *becomes smaller than* ε *for all* $p \in \mathcal{P}_k$ *except for one path, say* p_k^**, for which it grows larger than* $1-\varepsilon$*.*
Furthermore, under configuration $(p_1^*,\dots,p_K^*)$*, no flow can unilaterally reduce its delay: for all k and* $\forall p \in \mathcal{P}_k$, $d_k(p_1^*,\dots,p,\dots,p_K^*) \geqslant d_k(p_1^*,\dots,p_k^*,\dots,p_K^*)$.

The proof of convergence is given in [3] and also shows the additional properties of OPS described in the introduction. However two questions remain.

Price of Anarchy In general, a Nash equilibrium (NE) does not provide any guarantee on its global performance. In the worst case, the performance of a Nash equilibrium can be arbitrarily far from an optimal configuration. However, the price of anarchy (ratio between the performance of a Nash equilibrium versus the optimal configuration) is bounded when delays are smooth [6], and is known to converge to one for large networks or in heavy traffic [7].

[†] This assumption is needed for the underlying game to have a potential [5]. This is an essential ingredient to prove convergence. When flows do not have the same arrival rate, one can still use the OPS algorithm. It can be shown (not reported here) that if OPS converges, it finds a Nash equilibrium. An alternative is to split flows into subflows, all with the same rate. In that case the subflows from the same flow may end up using different paths from source to destination.

Speed of convergence The speed of convergence of similar algorithms is of order $1/n$ [8] in a centralized context (with a single flow) and with strictly convex objective functions. Neither condition is true here. Up to our knowledge, no theoretical result exists today to bound the speed of convergence for our algorithm, and we evaluated it experimentally in the following section.

3 Implementation and Experimentations

The OPS algorithm is completely distributed: it requires only local measures, local choices, and no coordination is needed between routers. In practice, however, a router cannot select full paths from source to destination. Therefore, we implement OPS as an SDN controller running on *gateways*, that can only select among several next-hop routers. Core routers simply use their regular routing protocols.

As for the measurement of delays, we use ACK packets to measure the one-way delay experienced by each flow, and eliminate clock offset between hosts by keeping track of the minimum measured delay. Finally, clock ticks (line 3 in Algo 1) are replaced by packet arrivals: the SDN controller "ticks" every T packets for each flow, and computes the empirical mean of the last S packets among those T, as a non-biased estimate of the average delay.

We evaluate our OPS algorithm in the network displayed in Figure 1, using Mininet [9] on Linux.

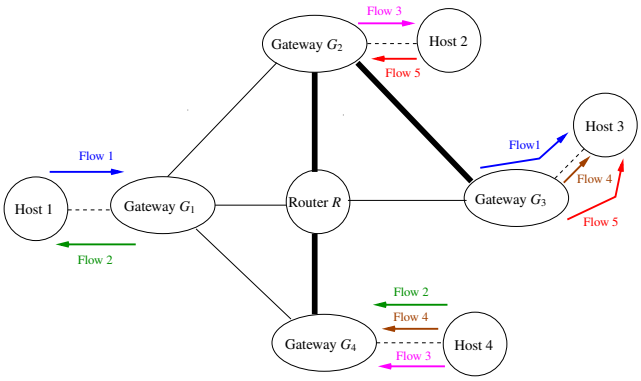

Figure 1: Network used in the following experiments. Thin lines represent links with capacity of 4 Mbit/s, while thick lines represent links with a capacity of 8 Mbit/s. All links have a transmission delay equal to 5 ms. Dashed lines between hosts and their gateway are links with unrestricted capacity. Five flows using the network are represented. Each flow consists in UDP packets with a constant throughput λ, with λ varying from 2000 to 3900 Kbit/s in the experiments. The *load* is $\rho = \frac{\lambda}{4000\ \text{Kbit/s}}$ ($\rho = 1$ means that a single flow uses the full capacity of a 4 Mbit/s link).

Flow	Equilibrium 1	Equilibrium 2
Flow 1	$G_1 \to G_2 \to G_3$	$G_1 \to G_2 \to G_3$
Flow 2	$G_4 \to G_1$	$G_4 \to R \to G_1$
Flow 3	$G_4 \to R \to G_2$	$G_4 \to R \to G_2$
Flow 4	$G_4 \to R \to G_3$	$G_4 \to G_1 \to R \to G_3$
Flow 5	$G_2 \to G_3$	$G_2 \to G_3$

As long as $\rho < 1$, there exists at least one stable configuration of the network, *i.e.* a choice of path for each flow that satisfies capacity constraints on all links. Additionally, for $\frac{2}{3} < \rho < 1$, there are only two Nash equilibria given in the table.

Since we have four gateways, we run four instances of our Ryu-based Openflow controller, each controlling a different gateway. We modified `udpmt` to generate UDP packets for each flow with a timestamp of the date of emission (μTP header). The destination host for each flow runs `udptarget`, suitably modified to reply back with small UDP packets containing μTP timestamps. This allows the controllers to measure the one-way delay of each flow.

This setup has been run a large number of times, to reduce variability. Each experiment lasts for 2400 seconds, to allow most executions to converge. To parallelize the execution, we used between 5 and 45 identical machines.

In the experiments, the load varies from 0.75 to 0.975. Figure 2 shows the evolution of Flow 1. Gateway G_1 has three possible next-hop routers: G_2, R, or G_4. We display the probability, over time, that G_1 selects each of the three next-hop routers. The other flows are also being forwarded concurrently, but this is not displayed here. G_2 gets selected most of the time, after a transient period. This choice is consistent with both equilibria.

In Figure 3, the convergence time is expressed as the number of iterations of our algorithm necessary so that all flows use a given route more than 80 % of the time. When the load approaches 1, we expect that

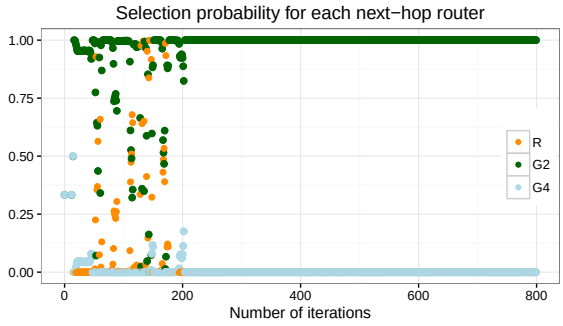

Figure 2: Probability of selecting each next-hop router over time, for Flow 1 (see Figure 1). Here, the load is equal to $\rho = 0.875$, and the parameters are $T = 500$ packets and $S = 5$ packets.

the convergence time goes to infinity, because the average delays cannot be reliably estimated for nearly unstable networks.

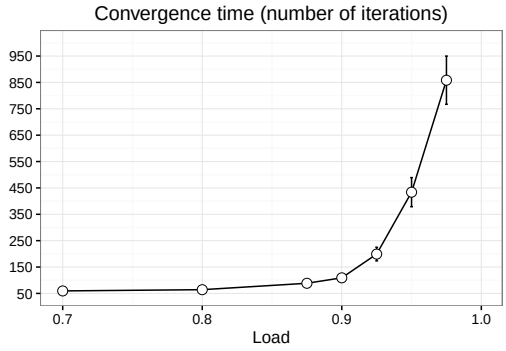

Figure 3: Average convergence time of the algorithm as a function of the load, with 95 % confidence intervals, for the network in Figure 1.

References

[1] T. C. Hu, "Multi-commodity network flows," *Operations Research*, vol. 11, no. 3, pp. 344–360, 1963.

[2] A. Khanna and J. Zinky, "The revised arpanet routing metric," *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 4, pp. 45–56, 1989.

[3] B. Jonglez and B. Gaujal, "Distributed Adaptive Routing in Communication Networks," Research Report RR-8959, Inria ; Univ. Grenoble Alpes, Oct. 2016.

[4] P. Coucheney, *et al.*, "Penalty-Regulated Dynamics and Robust Learning Procedures in Games," *Mathematics of Operations Research*, vol. 40, no. 3, pp. 611–633, 2015.

[5] A. Orda, *et al.*, "Competitive routing in multiuser communication networks," *IEEE/ACM Trans. on Networking*, vol. 1, no. 5, pp. 510–521, 1993.

[6] G. Christodoulou and E. Koutsoupias, "The price of anarchy of finite congestion games.," in *37th Annual ACM Symposium on Theory of Computing (STOC)*, 2005.

[7] R. Colini-Baldeschi, *et al.*, "On the price of anarchy of highly congested nonatomic network games," in *9th International Symposium of Algorithmic Game Theory (SAGT)* (Springer, ed.), no. 9928 in LNCS, pp. 117–128, 2016.

[8] A. S. Nemirovski and D. B. Yudin, *Problem Complexity and Method Efficiency in Optimization*. New York, NY: Wiley, 1983.

[9] B. Lantz, *et al.*, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 19, ACM, 2010.