Optimal congestion control with multipath routing using TCP-FAST and a variant of RIP

Enrique Mallada and Fernando Paganini

Universidad ORT
Montevideo, Uruguay

Abstract. This paper discusses an optimization-based approach for congestion control together with multipath routing in a TCP/IP network. In recent research we have shown how natural optimization problems for resource allocation can be solved in decentralized way across a network, by traffic sources adapting their rates and routers adapting their traffic splits, all using a common congestion measure. We present here a concrete implementation of such algorithms, based on queueing delay as congestion price. We use TCP-FAST for congestion control, and develop a multipath variant of the distance vector routing protocol RIP. We demonstrate through ns2-simulations the collective behavior of the system, in particular that it reaches the higher transfer rates available through multiple routes.

1 Introduction

The use of multiple paths between a source and destination of traffic is a natural choice to enhance the performance of a network, especially for bulk transfers that care mainly about the overall throughput. Flow could stream over many channel paths inside the network, adapting to whatever capacity is available at the moment. This ideal calls, of course, for a more active role for the network layer than that of current practice. In the prevailing situation, IP routing is insensitive to congestion and the only real-time control is done by the transport layer, designed to be single path; even if multiple TCP streams are used, there is limited control from the source as to how traffic travels inside the network. To fully explore the multipath options from the *source* (or an overlay at the edge of the network) as recently investigated in [7, 4], is not scalable given the exponential number of end-to-end paths.

A scalable alternative requires participation of the routers. Two main arguments against involving routers in real-time adaptation are (i) to keep them simple, and (ii) that congestion-based adaptation leads to instabilities such as route flaps. Tackling the second one first, it is well recognized that the main reason for such flaps is the use of single path routing, in which large bulks of traffic are suddenly switched. For multipath routing, stable methods based on gradual adaptation of the split of traffic have been known for a long time [3]. A recent proposal along these lines is [5], which uses a heuristic rule based on back-pressure signals to adapt routing splits. Other related work for wireless networks is [1, 12].

In [9] we proposed a way to combine multipath adaptive routing with the congestion control of TCP. We showed how to formulate global resource allocation optimization problems, and how to design local, scalable laws at sources and routers to solve them. In particular, we were able to give theoretical proofs of global convergence for arbitrary network topologies. These results are briefly reviewed in Section 2. Key to the overall solution is the propagation of a universal "congestion price" across the network.

The purpose of this paper is to develop an implementation of the ideas in [9]. We take as congestion price the queueing delay, which TCP variants such as FAST [6] estimate and use for congestion control, and routers can locally measure. This brings us back to the first point raised above: how complicated would it be for an IP router to perform multipath routing, and to handle and propagate these congestion signals? In Section 3 we show that the multipath routing algorithms of [9] fit well with distance vector routing protocols such as RIP (see [10]); we thus develop an enhancement of RIP to perform the necessary tasks. In Section 4 we present simulation work in ns2 to demonstrate the feasibility and properties of this implementation. We use the ns2 distribution of TCP-FAST [2], and developed our variant on the ns2 version of RIP. Conclusions are given in Section 5.

2 Combined congestion control and multipath routing

This section summarizes the ideas of [9] as to how to combine methods of congestion control with multipath routing, and the theory relating these algorithms to optimization.

2.1 Flow variables

Consider a set of nodes $\mathcal{N}$, indexed by i, j, connected by a set of directed links $\mathcal{L}$, each denoted by l or by (i,j). The network supports various flows indexed by $k \in \mathcal{K}$, between a source node $s(k)$ and a destination node $d(k)$, following possibly multiple paths. We introduce the following variables: x^k, external rate entering the network at the source; y_l^k, rate of flow k through link l; x_i^k, total rate of flow k coming into node i. These quantities are subject to the natural flow-balance constraints. The total flow on link l is denoted by y_l, and its capacity by c_l.

2.2 Multipath Routing

The router at node $i \in \mathcal{N}$ must decide on which of its outgoing links $(i,j) \in \mathcal{L}$ it will forward incoming packets with destination d. We introduce, for this purpose, routing fractions or " split ratios" $\alpha_{i,j}^d$ satisfying

$$\alpha_{i,j}^d \geq 0, \qquad \sum_{(i,j)\in\mathcal{L}} \alpha_{i,j}^d = 1.$$

This means the rates of flow k satisfy $y_{i,j}^k = \alpha_{i,j}^{d(k)} x_i^k$ for each $(i,j) \in \mathcal{L}$.

2.3 Congestion measure

Rates and routes will be controlled in response to a common congestion measure (called "price"). The basic price p_l is a scalar variable that measures the congestion state of each link $l \in \mathcal{L}$, depending on its total traffic y_l. In our implementation of Section 3, p_l will be the delay at the queue of link l.

The key to a scalable solution is the ability to summarize in a simple variable the congestion state of a portion of the network, using current routing patterns. We define, for this purpose, the node prices q_i^d, $i \in \mathcal{N}$, each representing the average price of sending packets from node i to destination d. Node prices must thus satisfy the recursion

$$q_d^d = 0, \qquad q_i^d = \sum_{(i,j)\in\mathcal{L}} \alpha_{i,j}^d [p_{i,j} + q_j^d], \quad i \neq d. \tag{1}$$

At the source node of flow k, the node price summarizes the congestion cost of the network for this flow. We denote it by

$$q^k := q_{s(k)}^{d(k)}.$$

2.4 Control

There are two main things to control in the network, source rates and router split ratios, based on the common congestion measure.

Control of source rates is typically done via the congestion window; a more macroscopic flow model takes the form of a "demand function" $x^k = f^k(q^k)$, in which rate responds to the congestion price q^k. In the case of TCP-FAST, this function takes the form

$$x^k = \frac{K_k}{q^k};$$

this is equivalent to the source maximizing its "consumer surplus" $U_k(x^k) - q^k x^k$ for the "utility function" $U_k(x^k) = K_k \log(x^k)$.

Routers must update the split ratios $\alpha_{i,j}^d$, based on congestion information, obtained locally or from its neighbors. Rather than instantaneously choosing the single least congested route, which causes route flaps, an idea that goes back to [3] is to gradually shift traffic to less congested routes. We impose the following conditions on the vector of changes $\{\Delta\alpha_{i,j}^d\}$:

- $\{\Delta\alpha_{i,j}^d\}$ depends on current ratios $\{\alpha_{i,j}^d\}$ and congestion prices $\{p_{i,j} + q_j^d\}$.
- $\{\Delta\alpha_{i,j}^d\}$ is negatively correlated with the route prices, and maintains node balance:

$$\sum_{(i,j)\in\mathcal{L}} \Delta\alpha_{i,j}^d (p_{i,j} + q_j^d) \leq 0, \qquad \sum_{(i,j)\in\mathcal{L}} \Delta\alpha_{i,j}^d = 0.$$

- Equilibrium is only reached when all outgoing links in use have equal price, $q_i^d = p_{i,j} + q_j^d$, and the rest have $\alpha_{i,j}^d = 0$.

2.5 Optimization interpretation

The above strategy for control of rates and routes can be interpreted in terms of distributed optimization. Indeed, in [9] the equilibrium points of such algorithms are shown to solve one of the following problems, which generalize those in [7, 8] to the case of arbitrary multipath routing.

Problem 1 (WELFARE). Maximize $\sum_k U_k(x^k)$, subject to link capacity constraints $y_l \le c_l$, and flow balance constraints.

Problem 2 (SURPLUS). Maximize $S := \sum_k U_k(x^k) - \sum_l \phi_l(y_l)$ subject to flow balance constraints.

Both these problems optimize aggregate utility of all sources, in the second case discounting a "traffic engineering cost". Which optimization applies depends mainly on the method used to generate link prices: Problem 2 applies to the case where the congestion measure is a marginal cost,

$$p_l = \phi_l'(y_l); \tag{2}$$

Problem 1 applies to a congestion measure satisfying the dynamics

$$\dot{p}_l = \gamma_l [y_l - c_l]^+_{p_l}. \tag{3}$$

In addition to an interpretation at equilibrium, dynamic properties of these algorithms are studied in [9]; it is shown that under certain assumptions of time-scale in the control, they converge globally to the optimal points.

Both of the models in (2-3) have been applied to queueing delay; the first (a static function of link rate) follows from queueing theory in steady state; the second from fluid-flow considerations. Regardless of these considerations, the main point is that taking queueing delay as our notion of congestion, the proposed methods for adaptation of routes and source rates have a rationale in terms of solving a global optimization.

3 Implementation

The formalism described in the previous section can be taken as a basis for more than one implementation, depending on the choice of the link congestion measure, the source utility function, and the method for sharing congestion information between routers and with traffic sources. In [9] we outlined some of the implementation issues in general terms, a discussion we continue here.

3.1 Discussion and Strategy

- A first point concerns the formation of node prices, which are used by routers to make decisions on traffic splits. Given link prices generated at each router, the corresponding node prices that satisfy (1) can be found iteratively: each node periodically updates q_i^d to the right-hand side of (1), based on announcements of neighboring nodes and its own link prices, and then announces its new price to its neighbors. Under the assumption of continued connectivity, it can be shown this recursion converges. The message passing is exactly the same as in distance-vector protocols such as RIP. The main change is that instead of taking hop-count as the default metric in routing announcements, we replace it by congestion price. Specifically, when router i announces to its neighbors it can reach destination d, it attaches as metric the corresponding congestion price q_i^d.

- Update of router split ratios: an implementation difficulty here, already identified in [3], refers of the possibility of generating routing loops during the transient phase (these disappear in equilibrium due to optimality). A blocking method was proposed in [3] to avoid this, and can be adapted here as follows: a loop can only occur if some packets are going "uphill" in price q_i^d, interpreted as a potential. This "improper" behavior cannot be completely banned without potentially leading to discontinuities in $\alpha_{i,j}^d$; however it can be flagged, and communicated to neighbors, warning them not to *start* routing new traffic in the direction of the improper-behaving node. In this way, starting from a loop-free configuration, this property is preserved.

- Communication of the price to the sources. A major point of discussion (see [11]) among congestion control implementations is whether to introduce *explicit* congestion signals between routers and sources, or to rely exclusively on implicit measures which can be estimated by sources. The latter alternative is usually favored for practical reasons of incremental deployment.

  In this paper we hit a middle ground on this issue. On one hand, we believe that explicit congestion control at the fastest time scale (a price in every packet) would be both burdensome to the routers, and not very useful in the multipath context. After all, for routers to find their correct node prices based on the RIP protocol takes some time, so there is no point in providing fast feedback of a quantity that may not have the correct value. So we will favor a congestion signal that sources can implicitly estimate, such as loss probability or queueing delay. In this way, sources that probe the network with large amounts of packets may be able to infer the current congestion measure faster than the time it takes the routers to become aware of it.

  Still, these implicit estimation methods may have biases and it is essential for the correct functioning of the overall system that, over the long run, sources and routers use compatible prices. Therefore we include, at the slower timescale of RIP announcements, an explicit portion of message passing between sources and routers which sources can use to calibrate their estimation. This requires the IP layer of a source node to listen to the RIP announcements, and pass the corresponding price q^k up to its TCP layer which is doing the price estimation.

- When choosing an implicit measure of congestion, to be useful in this scenario it should respect the recursion (1). From an end-to-end perspective, the price q^k must equal the average price experienced by packets over their respective routes, according to their corresponding routing fractions. In [9] we outlined how loss probability (or ECN marking probability) satisfied this requirement, to first order. Here we focus on queuing delay: if p_l is the delay of each link's queue, then q_i^d in (1) is the average or expected queueing delay experienced by a packet between node i and destination. So the price q^k at the source is the average queueing delay experienced by packets when probing all routes. What sources can explicitly measure is, however, not queueing delay but the average round-trip-time of their packets,
$$\overline{RTT} = \overline{D} + q^k,$$
where $\overline{D}$ is the average propagation/processing delay over all routes.
In single-path implementations such as TCP-FAST [6], the propagation delay is estimated through "BaseRTT", i.e. the minimum observed RTT, which assumes that the source has encountered an empty queue at least for one packet. This assumption can be criticized, especially for a source that starts transmitting on an already congested path, leading to biases and unfairness. In any event, this solution is not available for a multipath setting: if the paths have different propagation delays, the required "average BaseRTT" will be different from the minimum. The correct tracking of this quantity is one compelling reason for the use of some explicit prices from the IP layer at a slow time-scale, as described below.

3.2 Details and ns2 implementation

Multipath distance vector protocol This protocol is based on the Bellman-Ford distance vector algorithm, and its most well-known implementation, RIP (see e.g. [10]). The protocol learns routes to an IP destination address from its own locally connected networks, and from routes received from neighboring routers. But, as compared to RIP, our multipath protocol does not discard a route if it has a shorter (or cheaper) alternative; rather, it maintains in its routing table all possible next hops for a given destination. Each row in the routing table is accompanied with its metric, $p_{i,j} + q_j^d$, where $p_{i,j}$ is the queueing delay of the link, measured as the link queue divided by its capacity, and q_j^d is the metric learned from the downstream router. Also, each row has a variable that keeps track of the routing fraction $\alpha_{i,j}^d$ using this outgoing link. Forwarding decisions are made by choosing a pseudorandom number between 0 and 1, and going through the list of next hops until the sum of the $\alpha_{i,j}^d$ exceeds that number.

When the algorithm starts, it learns the routes from directly connected destinations, and assigns them cost $q_j^d = 0$. Since these are the first routes to be learned, they are assigned $\alpha_{i,j}^d = 1$: all traffic for this destination will initially be routed through this path. Analogously, every time a new destination is discovered it is assigned $\alpha_{i,j}^d = 1$; on the other hand, new routes learned for an already known destination are assigned $\alpha_{i,j}^d = 0$.

Routing announcements of the form (destination, metric, flag) are sent from each node to its neighbors. These are sent every Δ_t^r seconds, or also asynchronously if the node has received a notification that changes its routing table or metric. The first two fields are similar to RIP's, the metric is the weighted average q_i^d from (1). The additional flag indicates whether this is a proper or improper route, used for blocking loops: a route is announced as improper if at least one of the next hops j with $\alpha_{i,j}^d > 0$ satisfies either (i) $q_j^d > q_i^d$, or (ii) node j's last announcement had the improper flag on. We also included a version of RIP's "poison reverse" method: if node i is sending all its traffic to node j ($\alpha_{i,j}^d = 1$), the announcement from i to j carries infinite metric. This helps avoid trivial loops in the initial stages of the algorithm.

Periodically, with a separate period Δ_t^a , the adaptation of $\alpha_{i,j}^d$ takes place at node i. We describe this procedure, denoting for simplicity $\pi_j := p_{i,j} + q_j^d$, the metric seen by the node for each of its outgoing links:

- Identify the minimum metric $\pi_{min} = \min_j \pi_j$.
- For links which are more expensive than the minimum, update
$$\alpha_{i,j}^d(t + \Delta_t^a) := \alpha_{i,j}^d(t) + \beta\Delta_t^a(\pi_{min} - \pi_j);$$
  β is a system parameter that controls the speed of adaptation.
- The sum of all the decrements in $\alpha_{i,j}^d$ above is compensated by distributed increments in the cheapest links, except those which are *blocked*; a node is blocked if has the improper flag on and is receiving no traffic.

Modifications to TCP-FAST Associated with source nodes are TCP-FAST agents. These estimate average RTT and BaseRTT by running, for each received ACK, the updates

$$\overline{RTT} := (1 - a) * \overline{RTT} + a * currentRTT \tag{4}$$
$$BaseRTT := (1 - b) * BaseRTT + b(\overline{RTT} - q^k) \tag{5}$$

The RTT averaging equation (4) is standard: the parameter a can be made inversely proportional to the current window size. This makes the time constant of the filter to correspond to a certain number of RTTs. Equation (5) to estimate BaseRTT is modified from FAST, it is not based on the minimum RTT. Instead, we apply a lowpass filter, with parameter $b << a$ (so $BaseRTT$ evolves at a slower time-scale than $\overline{RTT}$), driven by the values ($\overline{RTT} - q^k$), where the prices q^k are computed by the routing agent. In ns2, every time the routing agent at a node computes a new price q_i^d, it checks whether there are any FastTCP agents at the same node with that destination. If so, it updates the q^k parameter used by that agent.

Another modification required on FastTCP is that, consistently with multipath routing, it no longer makes sense to consider the ordering of packet arrivals in decisions about congestion control. In particular, the duplicate ACK feature should be removed, and RTT averaging should be performed on all packets, not just those which come in order. In our current implementation, we sidestep these issues through the following means: on the receiving end, the receiver sends as ACK the sequence number of the received packet, rather than the usual of stating the next expected packet. All these become valid ACKs and are used in the RTT computation by the FastTCP source. On the other hand, we emulated on the source's side the receptor logic, so that the rest of the protocol did not require changes.

4 Simulation results

We present some results for the simple network topology shown in Figure 1. There are two sources at nodes 1 and 2, a common destination at node 4; both sources use TCP-FAST with parameter $K_1 = K_2 = 250$. This parameter represents the number of packets to be stored in network queues in equilibrium. There are two bottleneck links: (2,4) with capacity 37.5 Mbps, and (3,4) with capacity 25 Mbps; both have one-way propagation delay of 25ms. Links (1,2) and (1,3) have capacity of 100Mbps, and 50ms delay. Capacities and delays are the same in the reverse path. Packet size is 1040 bytes.

The following parameters were used in routers: $\beta = 0.5$, $\Delta_t^r = 500ms$, $\Delta_t^a = 500ms$. In TCP sources, we used $a^{-1} = 4 * cwnd$, $b^{-1} = 3000$ for the averaging of RTT and BaseRTT.

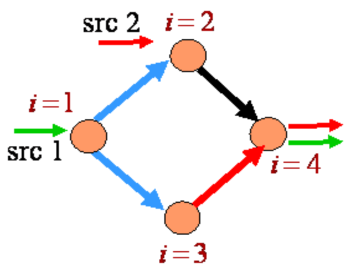
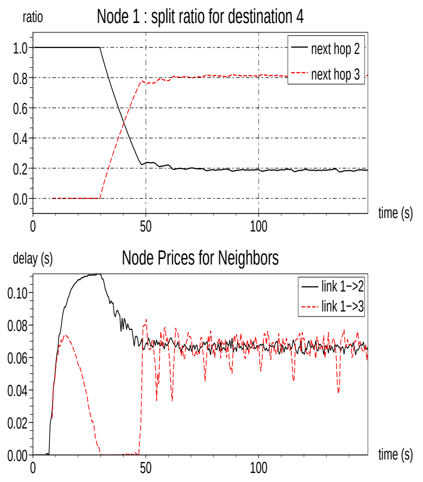


Fig. 1. Simulated network

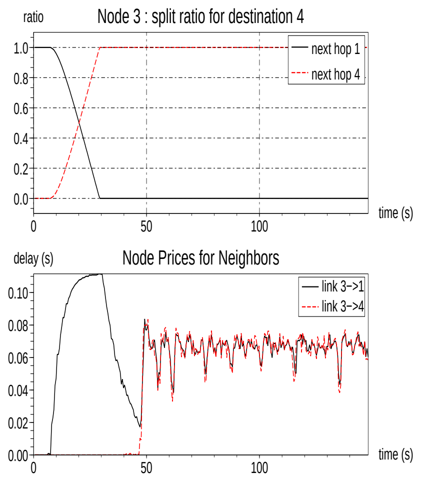
The simulation results are depicted below. Figure 2 contains split ratios and metrics (prices) for nodes 1 and 3. Figure 3 contains queueing delays for bottleneck links (2,4) and (3,4), and the rates of both sources in packets/second.

In the initialization process, for random reasons all nodes (1,2,3) discover first the top route, via node 2 to destination node 4. In particular, node 3 sets its split ratios initially to route via node 1, the longest route, and blocks the use of link (3,4) for node 1. Therefore, initially all traffic from both sources travels on the top route and reaches the destination through link (2,4). For about 7 seven seconds, both TCP-FAST sources ramp up their rates, source 2 having an initial advantage due to its smaller RTT. At that time, link (2,4) saturates and the sources react, converging around 30 seconds to sharing the bottleneck fairly.

By 30 seconds, node 3 has completed its transfer of routing to link (3,4). This unblocks the bottom route for node 1, and allows for source 1 to increase its total rate, eventually filling the bottom link as well around 45 seconds. At the same time, this allows source 2 to send more traffic through link (2,4).



(a) From node 1



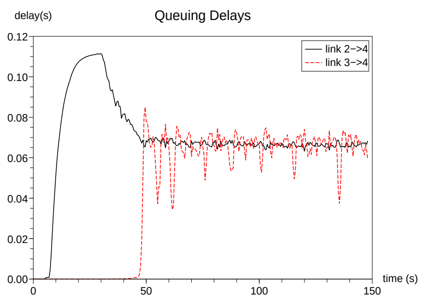
(b) From node 3

Fig. 2. Split ratios and prices for destination 4

After about 60 seconds, the system reaches an equilibrium with node 1 routing 20% of traffic from source 1 through the top path, and 80% through the bottom path. Node 2 routes all traffic from source 2 through link (2,4). Queueing delays (prices) at both bottlenecks equalize, so the sources with same demand curves equalize their rates to $(c_1 + c_2)/2 = 31.25$ Mbps or 3750 packets/second.

5 Conclusions

We have implemented distributed algorithms, at sources and network routers, which solve a distributed optimization problem, combining traffic engineering with congestion control as proposed in [9]. The implementation is based on queueing delay as a congestion price; routers measure local prices and exchange information with neighbors, following a multipath variant of a distance-vector routing protocol. Fast-TCP sources estimate this delay from their RTT measurements in real time, calibrating their propagation delay through periodic interactions with the IP layer. Our ns2 simulations over a simple network verify the expected behavior from the theory. Future work will involve more extensive



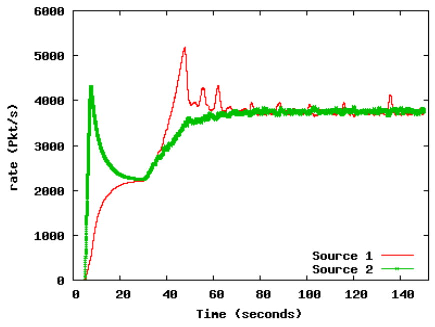


Fig. 3. Bottleneck queueing delays and source rates

tests and larger networks. Remaining challenges are partial deployment of this protocol and its compatibility with IP address summarizing.

Acknowledgment. This work was partially supported by grant FA9550-06-1-0511 of the US AFOSR, Latin American Initiative.

References

1. L. Chen, S. H. Low, M. Chiang and J.C. Doyle, "Cross-layer congestion control, routing and scheduling design in ad-hoc wireless networks", IEEE INFOCOM 2006.
2. T.Cui and L. Andrew, "FAST TCP simulator module for ns-2, version 1.1", available from `http://www.cubinlab.ee.mu.oz.au/ns2fasttcp`
3. R. G. Gallager, "A minimum delay routing algorithm using distributed computation", *IEEE Trans. on Communicactions*, Vol Com-25 (1), pp. 73-85, 1977.
4. H. Han, S. Shakkottai, C. V. Hollot, R. Srikant and D. Towsley, "Multi-Path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity on the Internet", in *IEEE/ACM Trans. on Networking*, Vol. 14, Issue 6, Dec. 2006.
5. I. Gojmerac, T. Ziegler, F. Ricciato, P. Reichl, "Adaptive Multipath Routing for Dynamic Traffic Engineering", Proc. GLOBECOM'03, San Francisco, Nov. 2003.
6. C. Jin, D. X. Wei and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance"; *IEEE Infocom*, March 2004.
7. F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability", *Jour. Oper. Res. Society*, vol. 49(3), pp 237-252, 1998.
8. S. H. Low and D. E. Lapsley, "Optimization flow control, I: basic algorithm and convergence", *IEEE/ACM Trans. on Networking*, vol.7, no.6, pp 861-874, 1999.
9. F. Paganini, "Congestion control with adaptive multipath routing based on optimization", *Conf. on Information Sciences and Systems*, Princeton, NJ, Mar 2006.
10. L. L. Peterson and B. S. Davie, *Computer Networks: a Systems Approach*, Morgan-Kauffman, San Francisco, 2003.
11. S. Shalunov, L.Dunn, Y. Gu, S. Low, I. Rhee, S. Senger, B. Wydrowski, L. Xu, "Design Space for a Bulk Transport Tool", http://transport.internet2.edu/
12. Y. Xi and E. Yeh, "Node-Based Distributed Optimal Control of Wireless Networks", *Conf. on Information Sciences and Systems*, Princeton, NJ, Mar 2006.