Control of multipath TCP and optimization of multipath routing in the Internet

Damon Wischik*, Mark Handley, and Costin Raiciu

UCL**

Abstract. There are moves in the Internet architecture community to add multipath capabilities to TCP, so that end-systems will be able to shift their traffic away from congested parts of the network. We study two problems relating to the design of multipath TCP. (i) We investigate stochastic packet-level behaviour of some proposed multipath congestion control algorithms, and find that they do not behave how we might expect from fluid modeling: they tend to flap randomly between their available paths. We explain why, and propose a congestion control algorithm that does not flap. (ii) We consider how the path choice offered by the network affects the ability of end-systems to shift their traffic between a pool of resources. We define a 'resource poolability' metric, which measures for each resource how easy it is for traffic to be shifted away from that resource e.g. in the event of a traffic surge or link failure.

Key words: multipath TCP, congestion control, resource pooling, fluid model, load balancing

1 Introduction

It has been argued that the natural next step in the evolution of the Internet is to harness the responsiveness of end systems to achieve better network-wide traffic management [1]. If end systems can spread their load across multiple paths in the right way, with the right reaction to the right congestion signals from the network, then traffic will quickly and automatically move away from congested or failed links in favour of uncongested links. This will relieve stress on the Internet's routing system (BGP), which is overwhelmed [2].

End-systems already do shape traffic to some extent: TCP backs off in response to congestion; peer-to-peer systems choose peers that give good throughput; content distribution networks route traffic to well-chosen server farms. These disparate mechanisms can pull in different directions, and they can conflict with the traffic management algorithms used by network operators [3]. There is now

* Computer Science department, Gower Street, London WC1E 6BT, UK. Telephone +44 20 76790442. Email d.wischik@cs.ucl.ac.uk

** This work arises from participation in the EU-funded Trilogy project, and particular thanks are due to Rolf Winter, Marcel Bagnulo and Pascal Merindol. Damon Wischik is supported by a university research fellowship from the Royal Society.

an opportunity to do things right: The Internet Engineering Task Force has recently begun to consider the practical design of a multipath version of TCP [4], and if the longevity of Jacobson's TCP is any indication then we will be living with the consequences of their decisions for several decades. This is a perfect opportunity for mathematical modeling to assist in the design process.

The fundamental challenge of relying on end-systems to manage network-wide traffic is this: how can the system as a whole achieve a desirable outcome, when the end-systems only have local knowledge? The key mathematical insight was provided by Kelly et al. [5], who showed that congestion control at end-systems can be thought of as a distributed control system for solving a network-wide optimization problem. They also realized that routing can be seen as an extension of congestion control—choosing route r_1 rather than r_2 is an extreme case of increasing traffic on the first route and reducing it on the second, and it should presumably be done in response to signals from the network about congestion levels along the two paths. There has since been a great deal of theoretical work on congestion control, some of it on multipath. There are six parts to the multipath-modeling research agenda:

(i) How does a fluid model arise from stochastic packet-level behaviour of a multipath congestion control algorithm?
(ii) Is the fluid model stable? [6, 7, 8]
(iii) What is the flow-level behaviour, assuming that congestion control works properly? [9, 10]
(iv) How should a flow learn which of many possible available paths it should use? What are the consequences for flow-level behaviour, both for multipath TCP[1] and for overlay networks such as peer-to-peer applications? [11, 13]
(v) What sort of path choice does the network need to offer?
(vi) What signals should the network use to affect the behaviour of multipath traffic? What will be the impact of end-system multipath on the peering and pricing contracts between network operators? [3]

This paper fills in some gaps in items (i) & (v). Section 2 is concerned with item (i). It turns out that the stochastic packet-level behaviour is quite surprising, and one needs to think carefully about what fluid models actually represent in order to understand the problem. This is important to get right, if one is to implement a reliable robust congestion control algorithm. We believe we have an algorithm that performs reliably enough to be deployed today, and we have a Linux implementation.

Section 3 is concerned with item (v). There are several technologies with which path choice might be offered: end-systems could set a few path selector

[1] Key et al. [11] propose a simple and appealing answer: 'if every end-system is given a choice of two paths, then flow-level behaviour is near optimal'. The theory behind this answer assumes that paths are chosen independently from a large collection of equivalent links. However the Internet's topology is likely to impose correlations between the choices, and this means that two paths might not be sufficient [12], unless the paths are well chosen—hence the need for (v).

bits in each packet header and the network could route according to what those bits say; or path choice could be achieved via multiple IP addresses at multi-homed end-systems; or it could be completely managed by overlay networks. It is vital to be able to judge the benefit of each of these mechanisms, so that Internet architects can decide if it is worth making wide-ranging changes e.g. to the Internet's routing system (BGP) or to the interpretation of the IPv4 'type of service' bits.

Terminology. We shall use the word flow to refer to a source of traffic. Each flow can send its traffic over one or more paths. The traffic it sends on a single path is called a subflow.

2 Designing a multipath congestion control algorithm

There have been four proposals for multipath congestion control algorithms: an original proof of concept by Kelly et al. [5], a translation of techniques from optimization and control theory by Wang et al. [6], and algorithms derived from fluid models by Kelly and Voice [8] and Han et al. [7]. In the latter three pieces of work, it was assumed that fluid models are an appropriate description of the system, and the work was to analyse the stability of the fluid models.

We simulated the algorithms suggested by these fluid models, and found surprising behaviour: even when the stability analysis says the system should be stable, the algorithms behaved erratically, flipping from sending almost all traffic on one path to sending almost all traffic on a different path, and the flips were non-periodic. In this section we describe this behaviour, and explain why it arises and how the fluid models should be interpreted. We will be concerned with the behaviour of an individual flow, not with aggregates.

Another issue is that the proposed fluid models are for an Internet in which a user's traffic rates are determined by the congestion he/she sees, whereas in the current Internet it is his/her window size that is determined by congestion, and traffic rates are determined by window size and round trip time. We describe how to adapt the multipath congestion control algorithm so that it plays nicely with today's protocols (or indeed with any other benchmark for fairness that we might set).

Notation. Suppose a flow can send its traffic over several paths, indexed by r. Suppose that the congestion control algorithm is window-based, like TCP, i.e. it maintains a window size w_r on each path and attempts to keep w_r packets in flight on path r. Congestion is controlled by adjusting the w_r. Let $w = \sum_r w_r$. The throughput i.e. traffic rate it gets on path r is $x_r = w_r/\mathsf{RTT}_r$, where RTT_r is the round trip time on that path. Assume that RTT_r does not vary with congestion; this is reasonable when the routers along the path all have small buffers, and a matter for further study when they do not. Let p_r be the packet drop probability (or the packet marking probability, if Explicit Congestion Notification is enabled).

2.1 Flappiness and resource pooling

Consider a very simple system consisting of a single flow with two paths, each path with constant packet drop probability p_r. Assume that both paths have equal round trip time, so that we might as well replace rates by window sizes in the fluid models. Also, assume that the drop probabilities are small so that $1 - p_r \approx 1$. A fluid model from [8, equation (21)[2]] is[3]

$$\frac{d}{dt}w_r(t) = \frac{w_r(t)}{\mathsf{RTT}_r}\Big[a - bw(t)p_r\Big]. \tag{1}$$

This corresponds to the congestion control algorithm that increases w_r by a whenever it receives an acknowledgement on path r, and decreases it by $bw(t)$ when it detects a drop. We will refer to this as Algorithm (1). A simulation is shown in the left column of Figure 1. The horizontal axis shows w_1 and the vertical axis shows w_2, and we plot lines to show the evolution of (w_1, w_2) between drops.

- If $p_1 > p_2$ then the fixed point of (1) is $\hat{w}_1 = 0$ and $\hat{w}_2 = a/(bp_2)$, and the simulation confirms that the algorithm uses path 2 almost exclusively.
- If $p_1 = p_2$ then any solution $(\hat{w}_1, \hat{w}_2)$ with $\hat{w} = \hat{w}_1 + \hat{w}_2 = a/(bp)$ is a fixed point of (1). The simulation shows however that the system flaps between $w_1 \approx 0$ and $w_2 \approx 0$. If we plot $w_1(t)$ as a function of t, we observe that the flaps occur at random (non-periodic) times. Note that multipath congestion control will tend to equalize congestion throughout the network, so the $p_1 = p_2$ case is generic.

We also simulated another algorithm, Algorithm (2), adapted from [7, equation (14)]:

$$\frac{d}{dt}w_r(t) = \frac{w_r(t)}{\mathsf{RTT}_r}\Big[a\frac{w_r(t)}{w(t)} - bw_r(t)p_r\Big]. \tag{2}$$

This algorithm has the same fixed point as (1) but it has gentler increases and decreases. We thought it might be less flappy but Figure 1 shows otherwise.

What causes flappiness? To understand how flappiness arises, consider a somewhat contrived scenario in which both paths use a single bottleneck link, and packet drops occur whenever $w_1 + w_2 = 100$, and the flow is using Algorithm (1) with $a = 1$ and $b = 1/4$. Starting from $w_1 = w_2 = 1$, both windows increase until $w_1 = w_2 = 50$. Suppose that path 1 experiences a drop and w_1 decreases. The two window sizes will then grow until $(w_1, w_2) = (33.3, 66.7)$. Just one more drop on path 1 is enough to push w_1 down to 8.3 packets. At this point it will take six consecutive drops on path 2 for the two windows to equalize again.

[2] The control-theoretic analyses in [7, 8] uses $x_r(t - \mathsf{RTT}_r)$ rather than $x_r(t)$, to reflect the fact that acknowledgements received at time t are for packets sent at time $t - \mathsf{RTT}_r$; but they also show that removing the lag to give (1) should only improve the stability of the dynamical system.

[3] It is understood in this and in all other fluid model equations that if $w_r = 0$ then we take the positive part of the right hand side.

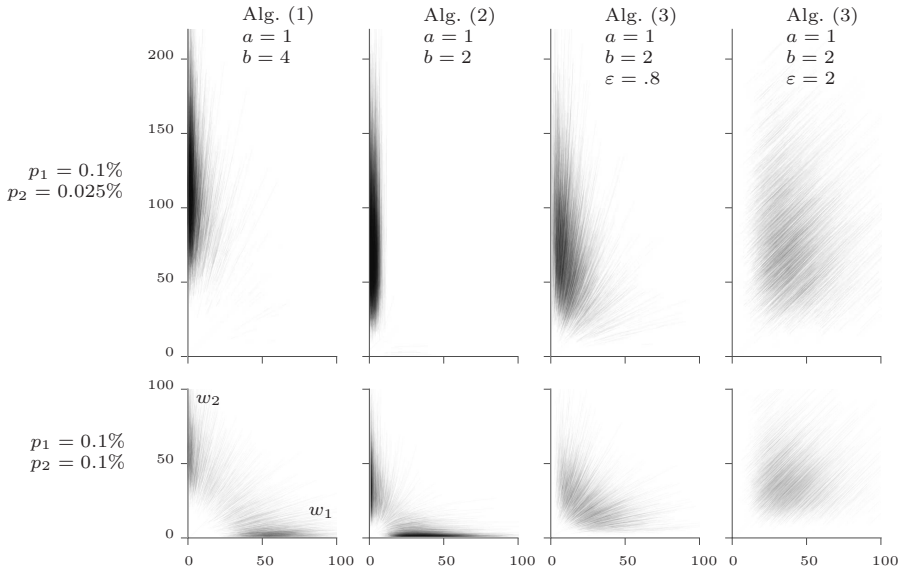

Fig. 1. Window size dynamics for a two-path flow. The axes are w_1 and w_2, and the plots show the increase phases of the process $(w_1(t), w_2(t))$.

There seem to be two drivers of flappiness. (i) Both algorithms move traffic off congested paths and onto uncongested paths. Even when two paths have the same drop probability, chance fluctuations will mean that from time to time one path suffers a few extra drops, so it will look momentarily more congested, so the flow will flip to the other path. To overcome this, it will be necessary either to accept less perfect resource pooling, or to use smoothed estimates of loss which will result in a more sluggish response. (ii) The second driver of flappiness in Algorithm (1) is the problem of capture: if flow 1 experiences a couple of drops, flow 2 needs to experience many more drops to bring the traffic rates back into balance.

We have simulated networks where the drop probabilities are not fixed but instead depend on the offered traffic, and still find flappiness.

Interpretation of fluid models. The issue with the fluid model is that it only holds in the limit as the number of flows tends to infinity, and in this limit $w_r(t)$ represents the average window size among a large ensemble of equivalent flows at time t. This was argued heuristically in [14], and proved rigorously for a simplified model in [15]. In a large ensemble of multipath flows, any linear combination $(\lambda, 1-\lambda)\hat{w}$ may appear as the average, if each individual flow flaps randomly between $(w_1, w_2) = (0, \hat{w})$ and $(w_1, w_2) = (\hat{w}, 0)$. We suspect that λ performs a symmetric random walk in $[0, 1]$, since by symmetry it is just as likely for a flow to flip from path 1 to path 2 as *vice versa*.

We note as an aside that the illustrations in this paper come from a semi-fluid simulator along the lines of [16], which we validated using a packet-level implementation. Our simulator solves a differential equation for the increase phase, e.g. $\dot{w}_r(t) = \big(w_r(t)/\mathsf{RTT}_r\big)\big(aw_r(t)/w(t)\big)$ for algorithm (2), and applies packet drops randomly according to an inhomogeneous Poisson process.

2.2 Alleviating flappiness

Consider Algorithm (3), the congestion controller corresponding to

$$\frac{d}{dt}w_r(t) = \frac{w_r(t)}{\mathsf{RTT}_r}\Big[\frac{a}{w(t)}\Big(\frac{aw_r(t)}{w(t)}\Big)^{1-\varepsilon} - bp_r w_r(t)\Big]. \tag{3}$$

At $\varepsilon = 0$ this is flappy, and if there are several paths with joint lowest drop probability then the fixed point is not unique. It is much like Algorithm (2), except for the $1/w(t)$ in the increase term which we put in so as to better reflect TCP's increase rule. At $\varepsilon = 2$ the subflows are uncoupled, and Figure 1 shows that this is completely unflappy and completely useless at shifting traffic away from the more congested path. At $\varepsilon = 0.8$ there is a reasonable compromise between flappiness and load-shifting.

For any $\varepsilon > 0$ it is easy to solve for the fixed point of (3) and to see it is unique[4]. The fixed point solves $w_r = (a^{2-\varepsilon}/bp_r\hat{w}^{2-\varepsilon})^{1/\varepsilon}$. Note that the total window size $\hat{w}$ is divided between paths in proportion to $1/p_r^{1/\varepsilon}$, so the smaller ε the greater the aversion to congestion. We can also use the equation $\hat{w} = \sum_r \hat{w}_r$ to solve for $\hat{w}$. It turns out that if there are several paths through a single bottleneck link, then $\hat{w}$ depends on the number of paths that the flow is using; this is clearly undesirable on grounds of fairness, and it was not the case for (1) or (2). In the next section we give a general-purpose method for removing unfairness.

2.3 Compensating for round trip time

Here is a simple way to design a multipath congestion control algorithm so that it fits in gracefully with other traffic, in particular with TCP and its dependence on round trip time. We set ourselves two goals. To state them, we first define $\hat{x}_r^{\mathsf{TCP}} = \sqrt{2/p_r}/\mathsf{RTT}_r$ to be the throughput that a single-path TCP flow would get if it experienced packet drop probability p_r and had round trip time RTT_r. Let $\hat{w}_r^{\mathsf{TCP}} = \sqrt{2/p_r}$ be the corresponding window size. Our goals are *(i) A multipath flow should not get more than $\hat{x}_r^{TCP}$ on any single path, though it may get less. This means that other flows cannot suffer, and may benefit, if I deploy multipath TCP. (ii) A multipath flow should get total throughput* $\max_r \hat{x}_r^{TCP}$. *This means that the more paths I have access to the more I benefit.* These goals explicitly use TCP as a reference, but the argument applies straightforwardly to any other reference throughput formula $hx_r^{\mathrm{ref}}(p_r, \mathsf{RTT}_r)$.

[4] There was also a $\varepsilon > 0$ parameter introduced in [7, equation (1)] to guarantee uniqueness of the fixed point; we however intend that ε should be a design parameter, say $\varepsilon \approx 0.8$, rather than a negligible term for making the maths tractable.

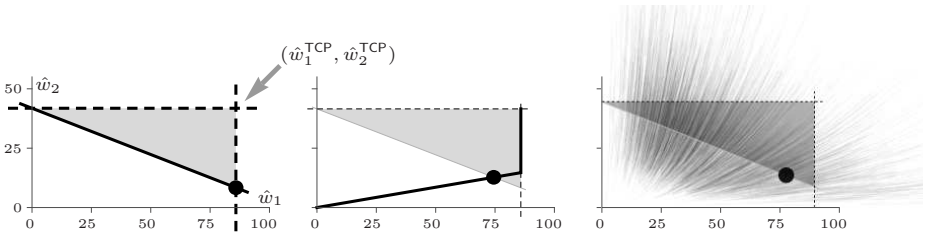

Fig. 2. The two design goals place constraints on what the equilibrium window sizes should be, and a can be chosen to meet them.

Our goals can be achieved using a congestion controller corresponding to the fluid model

$$\frac{d}{dt}w_r(t) = x_r(t)\Big[\frac{a}{w(t)}\Big(\frac{aw_r(t)}{w(t)}\Big)^{1-\varepsilon} \wedge \frac{1}{w_r(t)} - p_r\frac{w_r(t)}{2}\Big] \tag{4}$$

where $x \wedge y = \min(x, y)$. The $\wedge$ ensures that a window does not increase any faster than TCP would, and the decreases are the same as TCP, so goal (i) is satisfied by a coupling argument. To satisfy goal (ii), we want to choose a so that the equilibrium window sizes satisfy

$$\sum_r \frac{\hat{w}_r}{\mathsf{RTT}_r} = \max_r \frac{\hat{w}_r^{\mathsf{TCP}}}{\mathsf{RTT}_r}. \tag{5}$$

Figure 2 illustrates the constraints. The axes show $\hat{w}_1$ and $\hat{w}_2$ for a two-path flow with $p_1 = 0.025\%$, $p_2 = 0.1\%$, and $\mathsf{RTT}_1 = 2.5\mathsf{RTT}_2$. Goal (i) says that $(\hat{w}_1, \hat{w}_2)$ should lie below and to the left of the dashed lines. Goal (ii) says that it should lie on the sloping line. Since $p_1 < p_2$ we would ideally put as much traffic as possible on path 1, i.e. choose the black dot in the leftmost plot. In the middle plot, the bold line shows the fixed points $(\hat{w}_1, \hat{w}_2)$ that we can get by tuning a (with $\varepsilon = 0.8$); we propose to choose a to just meet goal (ii), i.e. to choose the black dot. To calculate a, first write out the fixed point equation for (4),

$$\frac{a}{\hat{w}}\Big(\frac{a\hat{w}_r}{\hat{w}}\Big)^{1-\varepsilon} \wedge \frac{1}{\hat{w}_r} = p_r\frac{\hat{w}_r}{2},$$

then rewrite it in terms of $\hat{w}_r^{\mathsf{TCP}} = \sqrt{2/p_r}$ to get

$$\hat{w}_r^{\mathsf{TCP}} = \hat{w}_r \vee \hat{w}_r^{\varepsilon/2}(\hat{w}/a)^{1-\varepsilon/2}$$

where $x \vee y = \max(x, y)$. Substituting into (5),

$$\sum_r \frac{\hat{w}_r}{\mathsf{RTT}_r} = \max_r\Big\{\frac{\hat{w}_r}{\mathsf{RTT}_r} \vee \frac{\hat{w}_r^{\varepsilon/2}(\hat{w}/a)^{1-\varepsilon/2}}{\mathsf{RTT}_r}\Big\}.$$

Equality cannot occur when the right hand side is equal to $\hat{w}_s/\mathsf{RTT}_s$ for some s, since that would require all the other $\hat{w}_r$ to be equal to 0, hence

$$\sum_r \frac{\hat{w}_r}{\mathsf{RTT}_r} = \max_r \frac{\hat{w}_r^{\varepsilon/2}(\hat{w}/a)^{1-\varepsilon/2}}{\mathsf{RTT}_r}.$$

Solving for a gives

$$a = \hat{w}\Big(\frac{\max_r \hat{w}_r^{\varepsilon/2}/\mathsf{RTT}_r}{\sum_r \hat{w}_r/\mathsf{RTT}_r}\Big)^{1/(1-\varepsilon/2)}.$$

From instant to instant the algorithm may not actually know the fixed point window sizes, so it cannot compute a exactly. We propose using the current window sizes $w_r(t)$ to estimate the fixed point window sizes $\hat{w}_r$, yielding an estimated value for a. Maybe it might be useful to smooth this estimate, but in our simulations it was not necessary. The rightmost plot in Figure 2 shows a simulation trace.

3 Resource poolability

In a multipath network, congestion at one resource can be alleviated by shifting traffic onto other resources. The extent to which this is possible depends on (i) how much of a flow's traffic is shifted between its paths in response to congestion, and (ii) which paths it has available. This section is concerned with the second point. we will explain what a resource pool is, and we will define a metric that measures the poolability of a resource.

Notation. Suppose the network comprises an interconnection of a set of flows S with a set of resources J. Each flow $s \in S$ identifies a unique source-destination pair. Associated with each flow is a collection of paths, each path being a set of resources. If path r belongs to flow s then we write $r \in s$. If a path r uses a resource j we write $j \in r$. (If two flows share a route, we deem there to be two paths that happen to use exactly the same resources.)

It is helpful to introduce matrices to succinctly express the relationships between flows, paths and resources. Let $A_{jr} = 1$ if $j \in r$ i.e. if path r uses resource j, and let $A_{jr} = 0$ otherwise. Let $H_{sr} = 1$ if $r \in s$ i.e. if path r serves flow s, and $H_{sr} = 0$ otherwise. The two 0–1 matrices A and H express all the details of the topology and multipath routing that we are concerned with.

Each path r has associated with it a traffic rate $x_r \geq 0$. The total traffic rate for flow s is $y_s = \sum_{r \in s} x_r$, and the total traffic at resource j is $z_j = \sum_{r:j \in r} x_r$. In matrix notation, $z = Ax$ and $y = Hx$. Also define the traffic intensity at j to be $\rho_j = z_j/C_j$.

3.1 Five optimization problems.

In the classic multicommodity flow problem, we imagine that there is a fixed demand $y \geq 0$, and that each resource j has a fixed capacity C_j, and we seek an

allocation x such that $Hx = y$ and $Ax \le C$. The optimization problem

$$\begin{aligned}&\text{FEAS}(y, C):\\ &\text{minimize} \quad \delta \quad \text{over} \quad \delta \in \mathbb{R}, x \ge 0\\ &\text{such that} \quad Ax \le C + \delta, \;\; Hx = y,\end{aligned}$$

indicates whether this is possible: there exists such an allocation when $\text{FEAS}(y, C) \le 0$. The dual problem is important—

$$\begin{aligned}&\text{POOL}(y, C):\\ &\text{maximize} \quad \sum_s y_s q_s - \sum_j C_j p_j \quad \text{over} \quad p \ge 0, q\\ &\text{such that} \quad \sum_j p_j = 1 \quad \text{and} \quad q_s \le \min_{r \in s} \sum_{j \in r} p_j \text{ for all } s.\end{aligned}$$

For the case of end-system congestion control, Kelly et al. [5] supposed that end-systems choose their traffic rates in response to congestion signals from the network. Let $p_j(z_j/C_j)$ be the drop probability (or marking probability, or price signal) at resource j when the load is z_j, and define $L_j(\rho) = \int p_j(\rho)\, d\rho$. Assume that $L_j(\cdot)$ is strictly convex. Suppose each flow s has a utility function $U_s(y_s)$ associated with its total traffic rate y_s, and consider the problem

$$\begin{aligned}&\text{SYSTEM}(C):\\ &\text{maximize} \quad \sum_s U_s(y_s) - \sum_j C_j L_j(z_j/C_j) \quad \text{over} \quad y \ge 0, x \ge 0, z \ge 0\\ &\text{such that} \quad Ax = z, \;\; Hx = y.\end{aligned}$$

In the case of single-path traffic (when H is the identity matrix), the equilibrium throughput of TCP is in fact the solution to this optimization problem with $U_s(y_s) = -1/(\mathsf{RTT}_s^2 y_s)$ where RTT_s is the round trip time for that flow. SYSTEM(C) is a natural generalization to multipath; [6, 7, 8] define multipath congestion control algorithms and show they have equilibrium throughputs which solve SYSTEM(C) or closely related problems. We can rewrite SYSTEM(C) as $\max_y \{\sum_s U_s(y_s) - \text{MINL}(y, C)\}$ where the latter optimization is

$$\begin{aligned}&\text{MINL}(y, C):\\ &\text{minimize} \quad \sum_j C_j L_j(z_j/C_j) \quad \text{over} \quad x \ge 0, z \ge 0\\ &\text{such that} \quad Hx = y, \;\; Ax = z.\end{aligned}$$

The dual to MINL(y, C) is

$$\text{OPTP}(y, C):$$

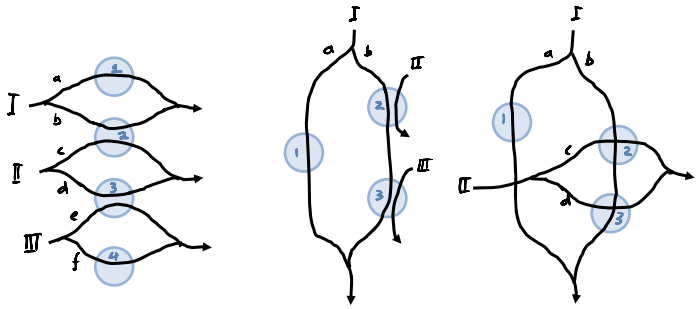

Fig. 3. Three scenarios for examining multipath flow problems. Flows are labelled I, II, III, paths are labelled $a, b, \ldots$ and resources are labelled $1, 2, 3, 4$.

$$\begin{aligned} &\text{maximize} && \sum_s y_s q_s - \sum_j C_j L_j^*(p_j) \quad \text{over} \quad p \geq 0, q \geq 0 \\ &\text{such that} && q_s \leq \min_{r \in s} \sum_{j \in r} p_j. \end{aligned}$$

Here $L_j^*(p)$ is the Fenchel-Legendre transform $L^*(p) = \sup_{\rho \geq 0} p\rho - L(\rho)$.

3.2 Resource pooling in the multicommodity flow problem

Here are some examples that give intuition about POOL. Consider the middle scenario in Figure 3. From explicit calculation, a flow y is feasible if and only if $y_{\mathrm{I}} + y_{\mathrm{II}} \leq C_1 + C_2$ and $y_{\mathrm{I}} + y_{\mathrm{III}} \leq C_1 + C_3$; it is like a single-path system with two 'resource pools' $\hat{C}_1 = C_1 + C_2$ and $\hat{C}_2 = C_1 + C_3$ where y_{II} uses $\hat{C}_1$, y_{III} uses $\hat{C}_2$ and y_{I} uses both. The two constraints reflect the two extreme feasible solutions $p = (1/2, 1/2, 0)$ and $p = (0, 1/2, 1/2)$ of POOL. Or consider the rightmost scenario in Figure 3. Here the single feasibility constraint is $2y_{\mathrm{I}} + y_{\mathrm{II}} \leq 2C_1 + C_2 + C_3$; it is like a single-resource system where y_{I} uses the resource twice and y_{II} only once. The constraint reflects the single extreme feasible solution $p = (1/2, 1/4, 1/4)$ to POOL. Laws [17] calls these 'generalized cut constraints', and gives several illuminating examples.

We tried running multipath congestion control on these three scenarios (and thereby solved SYSTEM and OPTP). In the rightmost scenario for example we found that flow II balances its traffic so that $p_2 = p_3$ i.e. so that congestion is balanced on those two resources, and flow I also balances its traffic so that $p_1 = p_2 + p_3$, resulting in $p_1 = 2p_2 = 2p_3$. This solution to OPTP somehow corresponds to the POOL solution $p = (1/2, 1/4, 1/4)$—which suggests that OPTP also tells us about resource pools. This is discussed further in Section 3.3.

POOL can also tell us about the effect of adding or removing capacity. Consider the leftmost scenario in Figure 3, and suppose $y = (6, 5, 5)$ and $C = (8, 2, 4, 3)$. This is not feasible, as exemplified by the dual variables $p = (0, 1/3, 1/3, 1/3)$ and $q = (0, 1/3, 1/3)$: we could add $\delta = q^{\mathsf{T}} y - p^{\mathsf{T}} C = 1/3$ unit of capacity to each

link to make it feasible. Alternatively, $d\delta/dC_j = p_j$, which means we could make y feasible by adding $\delta/p_4 = 1$ unit of capacity to link 4, for example. The corresponding analysis of OPTP is the basis for Section 3.4.

3.3 Finding resource pools

Given the similarity between POOL(y, C) and OPTP(y, C), we conjecture that the extreme optimal solutions to OPTP(y, C) tell us about which of the resource pools are tight, in the same way as do the extreme optimal solutions to POOL(y, C). In particular, based on simulation experiments, we conjecture the following:

The extreme optimal solutions to OPTP(y, C) *may be found as follows. Find the solution to* SYSTEM(C)*, for example by simulating the fluid model for a multipath congestion control algorithm of the sort described by Kelly and Voice [8]. Denote the optimal flows and drop probabilities by* $\hat{y}$ *and* $\hat{p}$*. All extreme optimal solutions* p *to* OPTP$(\hat{y}, C)$ *have the form* $p_j = \hat{p}_j 1_{j \in P}$ *for some set* $P \subseteq J$*. Call these sets* P *the resource pools.*

If this conjecture is correct, then one might employ heuristic techniques to discover the resource pools. One might then display them as a visualization aid, to assist a network operator in choosing alternative paths. For example, if some resource pool is a bottleneck then there is no point providing alternative paths that go through the same bottleneck. A well-connected network operator is likely to be able to find good alternative paths, but a poorly-connected operator is not. This is how network operators can provide value to their customers, even in a world of dumb pipes, intelligent end-systems and network neutrality.

3.4 Resource poolability matrix

In a multipath congestion control problem, what is the effect on OPTP(y, C) when we change the capacity of one of the resources? What is the effect on the drop probabilities? Intuitively, we might expect that if a resource's capacity is reduced then the drop probabilities of all the other resources in the same resource pool will increase, and other resources will not be affected. We might also expect that if a resource is in a large pool then drop probabilities are not much affected if the resource fails, but if it is in a small pool then failure has a much bigger impact.

Definition of poolability. For each resource j define its *poolable capacity* $\tilde{C}_j$ by

$$\tilde{C}_j = \frac{C_j}{\ddot{L}_j(\rho_j)}$$

where the dots refer to the second derivative with respect to ρ_j. Also define a $|J| \times |J|$ matrix Ψ, called the *resource poolability matrix*, and a $|S| \times |J|$ matrix Φ, called the *sensitivity matrix*, by

$$\begin{bmatrix} \Psi \\ \Phi \end{bmatrix} = \begin{bmatrix} \bar{A} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} M & -\bar{H}^{\mathsf{T}} \\ \bar{H} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \bar{A}^T \, \mathrm{d}(\tilde{C}^{-1}) \\ 0 \end{bmatrix}$$

Here $\bar{A}$ denotes the adjacency matrix A restricted to those paths with non-zero traffic, $\bar{H}$ denotes H restricted similarly, $M = \bar{A}^{\mathsf{T}}\,\mathrm{d}(\tilde{C}^{-1})\bar{A}$, and $\mathrm{d}(\tilde{C}^{-1})$ denotes the diagonal matrix with diagonal entry j given by $1/\tilde{C}_j$. If the inverse of the middle matrix does not exist, then use the Moore-Penrose pseudoinverse. It is shown in Section 3.5 that $dz_i/dC_j = \rho_j\Psi_{ij}$.

The resource poolability matrix lets us read off a variety of interesting quantities. For example, if the capacity of link j changes from C_j to $(1-\delta)C_j$, then how much does the traffic intensity at that link change? It changes by roughly

$$-\delta C_j \frac{d\rho_j}{dC_j} = -\delta C_j\Big(\frac{1}{C_j}\frac{dz_j}{dC_j} - \frac{z_j}{C_j^2}\Big) = \delta\rho_j(1-\Psi_{jj}).$$

What is the impact on drop probability at that link? It changes by roughly

$$-\delta C_j \frac{d}{dC_j}\dot{L}_j(\rho_j) = \frac{\delta C_j}{\tilde{C}_j}\rho_j(1-\Psi_{jj}).$$

From these two equations, we see that $\Psi_{jj} = 1$ means perfect resource pooling—if link j looses capacity then its traffic can be routed elsewhere and drop probability at j does not increase. If $\Psi_{jj} = 0$ then there is no resource pooling, and the answers are exactly what they would be for an isolated resource.

If the capacity of link j changes from C_j to $(1-\delta)C_j$, the total traffic at some other link i changes by roughly

$$-\delta C_j \frac{dz_i}{dC_j} = -\delta C_j\rho_j\Psi_{ij}.$$

Observe that $\delta C_j\rho_j$ is roughly the amount of traffic that has to move away from link j, hence $-\Psi_{ij}$ tells us what share link i takes of the knock-on traffic.

Examples of resource poolability. The resource poolability matrices for the three examples in Figure 3, assuming that all the shown routes are in use, are as follows. For the leftmost network Ψ is

$$\frac{1}{\sum_j \tilde{C}_j}\begin{bmatrix} \tilde{C}_2+\tilde{C}_3+\tilde{C}_4 & -\tilde{C}_1 & -\tilde{C}_1 & -\tilde{C}_1 \\ -\tilde{C}_2 & \tilde{C}_1+\tilde{C}_3+\tilde{C}_4 & -\tilde{C}_2 & -\tilde{C}_2 \\ -\tilde{C}_3 & -\tilde{C}_3 & \tilde{C}_1+\tilde{C}_2+\tilde{C}_4 & -\tilde{C}_3 \\ -\tilde{C}_4 & -\tilde{C}_4 & -\tilde{C}_4 & \tilde{C}_1+\tilde{C}_2+\tilde{C}_3 \end{bmatrix}$$

and for the middle and rightmost networks respectively Ψ is

$$\frac{1}{\sum_j \tilde{C}_j^{-1}}\begin{bmatrix} \tilde{C}_1^{-1} & -\tilde{C}_2^{-1} & -\tilde{C}_3^{-1} \\ -\tilde{C}_1^{-1} & \tilde{C}_2^{-1} & -\tilde{C}_3^{-1} \\ -\tilde{C}_1^{-1} & -\tilde{C}_2^{-1} & \tilde{C}_3^{-1} \end{bmatrix}, \quad \frac{1}{4\tilde{C}_1+\tilde{C}_2+\tilde{C}_3}\begin{bmatrix} \tilde{C}_2+\tilde{C}_3 & -2\tilde{C}_1 & -2\tilde{C}_1 \\ -2\tilde{C}_2 & 4\tilde{C}_1+\tilde{C}_3 & -\tilde{C}_2 \\ -2\tilde{C}_3 & -\tilde{C}_3 & 4\tilde{C}_1+\tilde{C}_2 \end{bmatrix}$$

There seems to be some sort of algebra here, akin to the algebra of electrical circuits in series and parallel, but we have not uncovered it.

3.5 Derivation of resource poolability

First write down the Lagrangian for MINL(y, C) or equivalently OPTP(y, C):

$$\mathcal{L}(x, z; p, q) = \sum_j C_j L_j(z_j/C_j) - \sum_j p_j \Big(z_j - \sum_r A_{jr} x_r\Big) + \sum_s q_s \Big(y_s - \sum_r H_{sr} x_r\Big).$$

The complementary slackness conditions are

$$\begin{aligned} p_j &= \dot{L}_j(\rho_j) \text{ for all } j, \text{ where } \rho_j = z_j/C_j \\ q_s &= \sum_j A_{jr} p_j \text{ for all paths } r \text{ in use by } s \\ z_j &= \sum_r A_{jr} x_r \text{ for all } j \\ y_s &= \sum_r H_{sr} x_r \text{ for all } s. \end{aligned} \tag{6}$$

The dot in $\dot{L}_j(\rho_j)$ refers to the derivative with respect to ρ_j. Substituting for p_j and z_j, the first three become

$$q_s = \sum_j A_{jr} \dot{L}_j\big(C_j^{-1} \sum_v A_{jv} x_v\big). \tag{7}$$

Now consider changing the capacity of resource C_i while leaving the other resources C and the total flow rates y unchanged. Differentiating (6) & (7) with respect to C_i we obtain

$$q_s' = \sum_j A_{jr} \ddot{L}_j(\rho_j) \Big\{ \frac{1}{C_j} \sum_v A_{jv} x_v' - \frac{\rho_j}{C_j} 1_{i=j} \Big\} \tag{8}$$

$$0 = \sum_r H_{sr} x_r'. \tag{9}$$

Here the primes q_s' and x_r' refer to derivatives with respect to C_i, the dots $\ddot{L}_j(\rho_j)$ refer as before to a double derivative with respect to ρ_j, and $1_{\{\cdot\}}$ is the indicator function, $1_{\text{true}} = 1$ and $1_{\text{false}} = 0$. Assume for now that all derivatives exist. Rearranging (8),

$$\rho_i A_{ir} \frac{\ddot{L}_i(\rho_i)}{C_i} = \sum_v x_v' \Big(\sum_j A_{jr} A_{jv} \frac{\ddot{L}_j(\rho_j)}{C_j} \Big) - \sum_s q_s' H_{sr}. \tag{10}$$

In matrix terms we can write (9) & (10) as

$$Hx' = 0 \quad \text{and} \quad \big[Mx' - H^{\mathsf{T}} q'\big]_r = \rho_i A_{ir}/\tilde{C}_i \text{ for each } r \text{ in use} \tag{11}$$

where $\tilde{C}_j = C_j/\ddot{L}_j(\rho_j)$ and $M = A^{\mathsf{T}}\, \mathrm{d}(\tilde{C}^{-1}) A$ and $\mathrm{d}(\tilde{C}^{-1})$ denotes the diagonal matrix with diagonal entry j given by $1/\tilde{C}_j$. These equations are easier to deal

with if we expand our focus and work simultaneously with derivatives with respect to each of the link capacities. It will also be convenient to rescale by ρ_i^{-1}. Accordingly, define the matrix $\tilde{x}'$ by $\tilde{x}'_{ri} = \rho_i^{-1} dx_r/dC_i$, and the matrix $\tilde{q}'$ by $\tilde{q}'_{si} = \rho_i^{-1} dq_s/dC_i$. With these definitions (11) becomes

$$\begin{bmatrix} M & -H^\mathsf{T} \\ H & 0 \end{bmatrix} \begin{bmatrix} \tilde{x}' \\ \tilde{q}' \end{bmatrix} = \begin{bmatrix} A^\mathsf{T} \,\mathrm{d}(\tilde{C}^{-1}) \\ 0 \end{bmatrix} \tag{12}$$

We want to know how drop probability is affected by a change in capacity, and this is straightforward to calculate from dz_j/dC_i, which it is convenient to rescale: let $\tilde{z}'_{ji} = \rho_i^{-1} dz_j/dC_i$. If we knew $\tilde{x}'$ we could simply compute $\tilde{z}' = A\tilde{x}'$, which is what we named as the poolability matrix Ψ. We named $\tilde{q}'$ as the sensitivity matrix Φ.

Existence and uniqueness. Now we must answer the questions: are $\tilde{x}'$, $\tilde{q}'$ and $\tilde{z}'$ uniquely determined? Do they even exist?

First, note that the original optimization problem MINL(y, C) always has a unique solution for z, because we assumed that L is strictly convex. It may be that $\tilde{z}'$ does not exist, in corner cases, e.g. when a path swaps in or out of use as capacities change. But if we are in a part of the capacity space where marginal changes in C do not alter the set of paths in use, $\tilde{z}'$ exists and is unique.

Second, observe that x may not be unique, for example when two flows have exactly the same choice of resources to use. However, if $\tilde{z}'$ exists then there must be some solution for $\tilde{x}'$ even if it is not unique. Therefore we might as well take the pseudo-inverse of the matrix in (12), since any solution for $\tilde{x}'$ is as good as any other for the purpose of computing $\tilde{z}'$.

Third, z determines p which determines q, so $\tilde{q}'$ is unique when it exists.

Final thoughts. Our analysis of resource poolability asks what happens when the network changes, assuming the total demand y does not change. This leads to clean maths. It also reflects a division of responsibilities—it is the role of the network to provide good paths, and the role of end-systems to decide how much traffic to send, and it is reasonable for the network to assume that end-systems will use low-congestion paths when they are available. We conjecture that our technique may be extended to other forms of load balancing that can be described by means of an optimization problem; congestion controllers such as (3) fall into this category.

Bibliography

[1] Wischik, D., Handley, M., Braun, M.B.: The resource pooling principle. ACM/SIGCOMM CCR **38**(5) (2008)
[2] Handley, M.: Why the Internet only just works. BT Technology Journal **24**(3) (2006)
[3] Acemoglu, D., Johari, R., Ozdaglar, A.: Partially optimal routing. IEEE Journal of selected areas in communications (2007)

[4] Ford, A., Raiciu, C., Handley, M., Barre, S.: TCP Extensions for Multipath Operation with Multiple Addresses. Internet draft, IETF (2009). URL `http://tools.ietf.org/html/draft-ford-mptcp-multiaddressed-01`
[5] Kelly, F.P., Maulloo, A.K., Tan, D.K.H.: Rate control in communication networks: shadow prices, proportional fairness and stability. Journal of the Operational Research Society **49** (1998)
[6] Wang, W.H., Palaniswami, M., Low, S.H.: Optimal flow control and routing in multi-path networks. Performance Evaluation **52**(2–3) (2003)
[7] Han, H., Shakkottai, S., Hollot, C.V., Srikant, R., Towsley, D.: Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the Internet. IEEE/ACM Transactions on Networking **14**(6) (2006)
[8] Kelly, F.P., Voice, T.: Stability of end-to-end algorithms for joint routing and rate control. ACM/SIGCOMM CCR **35**(2) (2005)
[9] Key, P., Massoulié, L., Towsley, D.: Combining multipath routing and congestion control for robustness. In: Proceedings of IEEE CISS. (2006) Expands on 2005 technical reports Microsoft TR-2005-111 and UMass CMPSCI 05-55.
[10] Kang, W.N., Kelly, F.P., Lee, N.H., Williams, R.J.: State space collapse and diffusion approximation for a network operating under a fair bandwidth sharing policy. Annals of Applied Probability (2009)
[11] Key, P., Massoulié, L., Towsley, D.: Path selection and multipath congestion control. In: Proceedings of IEEE INFOCOM. (2007) Also appeared in proceedings of IEEE ICASSP 2007.
[12] Godfrey, P.B.: Balls and bins with structure: balanced allocations on hypergraphs. In: Proceedings of ACM/SIAM SODA. (2008)
[13] Wang, B., Wei, W., Kurose, J., Towsley, D., Pattipati, K.R., Guo, Z., Peng, Z.: Application-layer multipath data transfer via TCP: schemes and performance tradeoffs. Performance Evaluation **64**(9–12) (2007) Expands on 2005 technical report UMass CMPSCI 05-45.
[14] Raina, G., Towsley, D., Wischik, D.: Part II: Control theory for buffer sizing. ACM/SIGCOMM CCR **35**(3) (2005) Summarizes [18].
[15] McDonald, D.R., Reynier, J.: Mean field convergence of a model of multiple TCP connections through a buffer implementing RED. Annals of Applied Probability **16**(2) (2006)
[16] Misra, V., Gong, W.B., Towsley, D.: Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. ACM/SIGCOMM CCR **30**(4) (2000)
[17] Laws, C.N.: Resource pooling in queueing networks with dynamic routing. Advances in Applied Probability **24**(3) (1992)
[18] Raina, G., Wischik, D.: Buffer sizes for large multiplexers: TCP queueing theory and instability analysis. In: Proceedings of EuroNGI conference on Next Generation Internet. (2005)