Principles of resource allocation in networks

Devavrat Shah (MIT) and Damon Wischik (UCL)

ABSTRACT

Much research in networking, in areas such as TCP congestion control, wireless MAC protocols, fair sharing of data centers, and optimization of distributed algorithms, falls under the general heading of *resource allocation*. The network modeling community has developed a general-purpose language for describing resource allocation problems, and canonical mechanisms for solving them. We believe these should be taught as fundamental principles to graduate students in networking. This will encourage them to apply ideas from one area of networking to another, it will help them to distinguish which parts of a problem need hands-on systems-level work and which do not, and it will make them think more deeply about economic and social questions such as network neutrality.

1. INTRODUCTION

At the 2011 NSDI conference, 19% of papers and 26% of posters could be described as having network resource allocation as their main focus; an additional 33% of papers and 35% of posters were concerned in some way with the control plane for resource allocation problems. The resources in question ranged from Internet bandwidth to data center machines to wireless spectrum.

From our experience of teaching network modelling to graduate students, both lecturing and also giving advice to students doing their masters projects, we believe it is useful to teach general-purpose principles that can be applied to many different problems to do with resource allocation. By teaching general principles illustrated by a small number of applications, rather than going through a set of papers, students are given the tools they need to derive answers for themselves and compare their answers to published work, which is the best way to learn. We also believe it helps students to make connections between different application areas, reinforcing their knowledge. Finally, it allows us to cover more ground in a fewer lectures.

In Sections 2–4, we outline the general principles and give illustrations, to indicate what material could be covered in such a course. The general approach has three steps:

Step 1: identify the feasible region.

The first step is to identify the feasible region, i.e. the set of flow rates that the network can support. In Section 2 we give examples of feasible regions for Internet bandwidth, wireless bandwidth, data center resource allocation, and distributed algorithms.

Step 2: choose the desired operating point.

The second step is to decide where in the feasible region we would like the network to operate. It is important to make students realize that there is always a normative decision to be made ("the network *should* choose this feasible allocation rather than that feasible allocation").

In Section 3 we describe a general-purpose objective—weighted α-fair utility maximization—and illustrate how it applies to TCP and to data centers. This objective subsumes other goals including max-min fairness, TCP friendliness, throughput maximization, and Pareto efficiency, and this makes it a good tool to make students think about tradeoffs in network resource allocation. To put it bluntly, students need to face up to complexities of networks beyond what can be captured by average throughput and Jain's fairness index.

The mathematical prerequisite for this step is the ability to understand an optimization problem with constraints. We have found that this can be taught to students with high-school mathematics, in one or two lectures, helped by interactive animations in Mathematica.

Step 3: reach the desired operating point.

The third step is to design an algorithm. There are canonical distributed algorithms for finding the optimum operating point, which we outline in Section 4. These canonical algorithms must then be translated to the network in question; when they are translated to Internet congestion control, for example, we obtain TCP-like, RCP-like and backpressure-like algorithms.

When students understand that resource allocation is a general-purpose problem, and that there are canonical solutions, they will be better equipped to extract

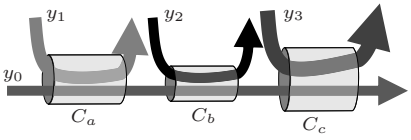

Figure 1: Bandwidth allocation for TCP

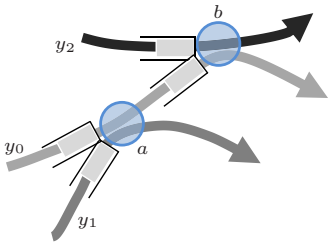

Figure 2: Job arrival rates for a distributed algorithm

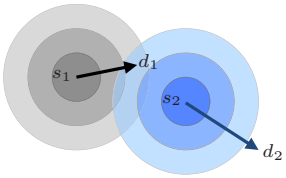

Figure 3: Wireless transmission with interference

general-purpose mechanisms from a paper in one area and apply it to a different area. We also believe it can help students working on projects to understand which parts of their work have generic solutions (to save them from re-inventing the wheel), and which parts require hands-on domain-specific systems work.

To fully understand the canonical solutions, it is necessary to be able to form the dual of a constrained optimization problem, and to understand the role of dual variables. We have found this is accessible to some students and inaccessible to others, depending on their mathematical background. The latter students can be taught the canonical solutions as recipes.

Taking it further

A formal introduction to resource allocation, as outlined above, can be the starting point for informed discussion in two directions:

The first direction is about the general architecture of control planes for resource allocation. The canonical algorithms from Step 3 involve signals from congested resources, and end-system response to those signals. This leads to the question: what sorts of congestion signals do different types of network expose, and to whom, and what sort of resource allocation outcomes can be achieved from these signals?

The second direction for discussion is about policy. Step 2 forces students to confront the fact that the network designer has to make a normative decision about resource allocation. How does this fit with the idea of network neutrality? In 'adversarial' networks, how can a policy about resource allocation be enforced?

We discuss these directions briefly in the conclusion, Section 5.

Background and related work

The general-purpose approach to resource allocation has been developed in the network modelling community over the past 20 years. Notable contributions are by Tassiulas and Ephremides [13] and Kelly et al. [5]. Bandwidth allocation in the wired Internet has received the most attention (see for example [8], the book by Shakkottai and Srikant [12], and the tutorial by Le Boudec [7]), though one of our goals in this position paper is to point out that the ideas apply much more widely than just bandwidth allocation. The work builds on general theory about multi-commodity flows on graphs, which has a long history in the operations research community, see for example [1]. Much of this work is written by and for the network modelling community, and it may not be very accessible to systems-oriented graduate students.

2. THE FEASIBLE REGION

The *feasible region* is defined to be the set of flow rates that can be supported, given the available capacity in a network. To flesh this out, here are some examples. The first four examples are of static scenarios, e.g. allocation of bandwidth to long-lived TCP flows. The last two examples are of dynamic scenarios, e.g. rate of arrivals of TCP flows of given average size.

Example 1 (data center) This is a toy example from [3], intended to illustrate resource sharing in a data center. Consider a system with 9 CPUs and 18 GB of RAM, and two users, where an instance of user 1's task requires 1 CPU and 4 GB of RAM, and an instance of user 2's task requires 3 CPUs and 1 GB of RAM. We will allow fractional tasks, e.g. if user 1 is allocated 1/2 CPU and 2 GB RAM then his/her instance runs at half speed. Let y_1 be the number of instances allocated to user 1, and let y_2 be the number of instances allocated to user 2. The feasible region is the set of all $(y_1, y_2) \geq 0$ such that

$$y_1 + 3y_2 \leq 9 \quad \text{(CPU constraint)}$$
$$4y_1 + y_2 \leq 18 \quad \text{(RAM constraint)} \qquad \diamondsuit$$

Example 2 (TCP) Consider a network with three links with capacities C_a, C_b and C_c Mb/s, and four TCP flows, as shown in Figure 1. Let y_s be the throughput of flow $s \in \{0, 1, 2, 3\}$. The feasible region is the set of all $(y_0, y_1, y_2, y_3) \geq 0$ such that

$$y_0 + y_1 \leq C_a,\ y_0 + y_2 \leq C_b,\ y_0 + y_3 \leq C_c. \qquad \diamondsuit$$

Example 3 (multipath TCP) This example is from [14, Figure 2]. Consider a network with three links with

capacities C_a, C_b and C_c. Let there be three multipath flows: flow 1 can use path 1 through a or path 2 through b and c, flow 2 can use path 3 through b or path 4 through a and c, flow 3 can use path 5 through c or path 6 through a and b. Let x_r be the throughput on path $r \in \{1, \ldots, 6\}$, and let y_s be the total throughput of flow $s \in \{1, 2, 3\}$. The feasible region is the set of all $(y_1, y_2, y_3) \geq 0$ such that there exist $x_1, \ldots, x_6 \geq 0$ for which

$$y_1 = x_1 + x_2, \quad y_2 = x_3 + x_4, \quad y_3 = x_5 + x_6,$$
$$x_1 + x_4 + x_6 \leq C_a, \quad x_2 + x_3 + x_6 \leq C_b,$$
$$x_2 + x_4 + x_5 \leq C_c.$$

After some careful linear algebra, one discovers that the x_r can be eliminated: the feasible region is the set of all (y_1, y_2, y_3) such that

$$y_1 + y_2 \leq C_a + C_b, \; y_2 + y_3 \leq C_b + C_c, \; y_1 + y_3 \leq C_a + C_c.$$

(Students with good linear algebra may be able to follow the proof in [4, Section 3.3] that the feasible region always involves linear constraints on the y_s. This particular calculation is taken from that reference.) ♢

Example 4 (wireless capacity) The Shannon-Hartley theorem says that the capacity of a wireless channel is $B \log_2(1 + S/N)$ where B is the channel bandwidth, S is the power of the signal, and N is the power of noise. Consider the wireless network in Figure 3, with two transmitters and two receivers. Let $h_{s,d}$ be the fading from transmitter s to receiver d. Suppose that each transmitter chooses to transmit at power $x_s P_s$ for some $x_s \in [0, 1]$, where P_s is the maximum power of transmitter s, and to send data at rate $y_s \geq 0$, and suppose that each receiver attempts to decode only the signal intended for it (i.e. the receivers do not use SIC), and that there is background noise of power N. The feasible region is the set of all $(y_1, y_2) \geq 0$ such that there exist $x_1, x_2 \in [0, 1]$ such that

$$y_1 \leq B \log_2\big(1 + x_1 h_{1,1} P_1 / (N + x_2 h_{2,1} P_2)\big),$$
$$y_2 \leq B \log_2\big(1 + x_2 h_{2,2} P_2 / (N + x_1 h_{1,2} P_2)\big).$$

After some algebra, one finds that the x_s can be eliminated, and the feasible region can be rewritten in terms of inequalities involving only y_1, y_2, and the constants h, P and n. ♢

Example 5 (distributed algorithm) This is a toy example from [11], intended to illustrate job scheduling in a distributed algorithm such as a distributed hash table. Consider a system with two machines and three types of jobs, as shown in Figure 2. Suppose both machines can handle C jobs per second. Let y_s be the rate at which jobs arrive of type $s \in \{0, 1, 2\}$, and suppose jobs of type 0 use machine a then b, jobs of type 1 use only machine a, and jobs of type 2 use only machine b. The feasible region is the set of all $(y_0, y_1, y_2) \geq 0$ such that

$$y_0 + y_1 \leq C, \text{ and } y_0 + y_2 \leq C. \qquad ♢$$

Example 6 (flow-level TCP) Consider first a single bottleneck link used by TCP flows, and take the TCP flows to have average size m Mb. Let the average rate at which new flows arrive be y per second (perhaps one flow starts as soon as the last one finishes, perhaps new flows arrive independent of what has arrived previously). Let the link speed be C Mb/s. The feasible region is simply $\{y \geq 0 : ym \leq C\}$. If y lies outside this region then demand (measured in bits per second) arrives quicker than it can be transmitted, so the number of active flows increases steadily. If y lies inside this region then (assuming that TCP manages to keep the link fully utilized) demand is served at the same average rate it arrives, and the link is stable.

Here is a network example of the same general sort, taken from [10]. Consider a network consisting of two links, link a with capacity C_a and link b with capacity C_b. Suppose some TCP flows use just link a, some use just link b, and some use both links; call these flows type 1, type 2 and type 0 respectively. Suppose the flows have finite size; let the average size of flows of type s be m_s. Depending on the number of active flows of each type, it may be that link a is the bottleneck, or link b, or both jointly may form the bottleneck. Let y_s be the average rate at which new flows of type s arrive. The feasible region is the set of all $(y_0, y_1, y_2) \geq 0$ such that

$$y_0 m_0 + y_1 m_1 \leq C_a, \text{ and } y_0 m_0 + y_2 m_2 \leq C_b. \quad (1)$$

To see that these inequalities are required for stable operation, simply count up the total demand (in bits per second) to be transmitted across each link. It is not true in general that these inequalities are sufficient to ensure stable operation—for example, if there are alternating flows of type 1 and type 2, and one flow starts as soon as the previous flow finishes, then stable operation requires $y_1 m_1 / C_a + y_2 m_2 / C_b \leq 1$, which is more restrictive than (1). However, it has been shown [10] that if flow arrivals are a Poisson process, then (1) is sufficient to ensure stable operation, using an idealized model of TCP. ♢

3. CHOOSING THE OPERATING POINT

The network has to operate with some allocation in the feasible region. The choice of which allocation depends both on the algorithms for the control plane of the network, and on user behaviour. We will first give some examples, then review the standard metrics for evaluating resource allocation, then specify a general-purpose metric which encompasses all the examples and the standard metrics.

Example 1 (data center) The proposal in [3] is that the data center should pick an allocation (y_1, y_2) such that the vector $(w_1 y_1, w_2 y_2)$ is max-min fair, where $w_1 = 4/18$ and $w_2 = 3/9$. The idea is to weight tasks according to how much capacity they consume at the resource at which they consume their largest share: for example one unit $w_1 y_1 = 1$ corresponds to $y_1 = 18/4$ instances of task 1, which consume 100% of the RAM and 50% of the CPUs, and one unit $w_2 y_2 = 1$ consumes 100% of the CPUs and 17% of the RAM. A max-min fair allocation of $(w_1 y_1, w_2 y_2)$ attempts to equalize consumption at each flow's dominant resource. ♢

Example 2 (TCP) Let p_a, p_b and p_c be the loss rates at the three links in Example 2 above, and let RTT_0 etc. be the round trip times. According to the TCP throughput formula, the throughputs achieved by the four flows are

$$y_0 = \frac{\sqrt{2}}{\mathsf{RTT}_0\sqrt{p_a + p_b + p_c}}, \quad y_1 = \frac{\sqrt{2}}{\mathsf{RTT}_1\sqrt{p_a}},$$
$$y_2 = \frac{\sqrt{2}}{\mathsf{RTT}_2\sqrt{p_b}}, \quad y_3 = \frac{\sqrt{2}}{\mathsf{RTT}_3\sqrt{p_c}},$$

assuming that loss rates are small enough that the loss rate experienced by flow 0 is $1 - (1 - p_a)(1 - p_b)(1 - p_c) \approx p_a + p_b + p_c$. It is not hard to show that these throughputs are exactly the solution to the optimization problem

$$\text{maximize} \quad \frac{-2}{y_0\mathsf{RTT}_0^2} + \frac{-2}{y_1\mathsf{RTT}_1^2} + \frac{-2}{y_2\mathsf{RTT}_2^2} + \frac{-2}{y_3\mathsf{RTT}_3^2}$$
$$\text{such that } (y_0, y_1, y_2, y_3) \text{ is feasible.}$$ ♢

Example 5 (Distributed algorithm) In Example 5 above, it would be reasonable to treat the arrival rates of jobs as fixed by user demand, and outside the control of the distributed system. If this is so, either the arrival rates are inside the feasible region and all jobs can be handled, or the arrival rates are outside the feasible region and some must be dropped. In other words, the resource allocation problem is one of admission control rather than rate allocation. Even the decision not to implement explicit admission control, and to simply drop jobs when buffers fill up, is a form of admission control—a foolish form of admission control, since it may lead to congestion collapse, i.e. wasting effort by serving a job that will turn out to be dropped downstream.

A simple objective is to maximize the net throughput. In the context of this simple example, this means picking rates $\hat{y}_0, \hat{y}_1, \hat{y}_2$ so as to

$$\text{maximize } \hat{y}_0 + \hat{y}_1 + \hat{y}_2 \quad \text{over } (\hat{y}_0, \hat{y}_1, \hat{y}_2) \text{ feasible,}$$
$$\text{such that } \hat{y}_0 \le y_0,\ \hat{y}_1 \le y_1,\ \hat{y}_2 \le y_1.$$

In this problem, y_s is the arrival rate of jobs of type s, taken to be fixed, and $\hat{y}_s$ is the rate at which those jobs are admitted, taken to be under the control of the distributed algorithm. Alternatively, we might imagine that there is a revenue associated with each completed job, say revenue w_s for jobs of class s, and aim to pick job admission rates so as to maximize $w_0 y_0' + w_1 y_1' + w_2 y_2'$. For example, if the revenues in Figure 2 are $w_0 = \$5$, $w_1 = \$4$ and $w_2 = \$6$, and the offered rate y_2 is high enough, then it is optimal for machine a to reject all \$5 jobs and only serve \$4 jobs. ♢

Here are some of the standard ways to evaluate a resource allocation.

Jain's fairness index measures how equal the flow rates are. It is given by $(\sum y_s)^2/(n\sum y_s^2)$ where n is the number of flows. It lies in the range $[0, 1]$, and if all flow rates are equal it has value 1.

Pareto efficiency is a more appropriate way to judge an allocation in networks with more than one bottleneck, when it may not be possible to achieve a Jain's fairness index of 1. An allocation vector $\mathbf{y}$ is Pareto efficient if there does not exist any other feasible allocation $\mathbf{y}'$ such that $\mathbf{y}' \ge \mathbf{y}$ and $y_s' > y_s$ for at least one flow s.

Max-min fairness specifies one particular allocation out of the many possible Pareto-efficient allocations. An allocation is max-min fair if increasing the allocation of any user must be at the cost of reducing the allocation of some other less fortunate user. Formally, $\mathbf{y}$ is max-min fair if for any other feasible allocation $\mathbf{y}'$ with $y_s' > y_s$ there exists t with $y_t' < y_t \le y_s$. This can be generalized by attaching weights to each user, as in Example 1 above.

Throughput maximization is another intuitively appealing goal. The trouble with max-min fair allocations is that they may have low total throughput. For example, in Example 2 (Figure 1), if all links have equal capacity, then the max-min fair allocation gets 2/3 of the maximum possible throughput. As described in Example 5, revenue maximization is the same as weighted throughput maximization.

A general-purpose metric.

All these examples and metrics can be subsumed into a single goal: given weights $w_s > 0$ and a parameter $\alpha \ge 0$, choose the allocation vector $\mathbf{y}$ which solves

$$\text{maximize} \quad \sum_s w_s \frac{y_s^{1-\alpha}}{1-\alpha} \tag{2}$$
$$\text{over all feasible allocations } \mathbf{y}.$$

(If $\alpha = 1$ then maximize $\sum_s w_s \log(y_s)$ instead.) This metric (2) is known as weighted α-fair utility, and it was introduced by Mo and Walrand [9]. Let us stress straight away that this metric is intended as a conceptual aid; it is not intended that students should attempt

to solve it in anything more than toy examples, and it is certainly not something that should ever be explicitly computed in a live network.

The links with the earlier examples and metrics are as follows. The solution to this optimization problem is always Pareto efficient, for any weights and any $\alpha \geq 0$. It can be shown that as $\alpha \to \infty$, the solution approaches the weighted max-min fair allocation. At $\alpha = 0$, we recover the weighted throughput-maximizing allocation. At $\alpha = 2$ and with weights $w_s = 2/\mathsf{RTT}_s^2$ we recover the allocation achieved by TCP (Example 2).

The pedagogical benefits of teaching resource allocation via weighted α-fair utility maximization are:

- It is a general-purpose metric which subsumes all the other standard metrics and examples. Resource allocation is not a hodge-podge of different ideas.
- Different types of resource allocation are parameterized very simply, by adjusting weights and the α parameter. This makes it easy to systematically explore the problem space.
- It makes it clear that there is a continuum between efficiency (small α) and fairness (large α), and there is in general no sweet spot which is both max-min fair and throughput-maximizing.
- Most importantly, there are canonical distributed algorithms for finding an allocation which maximizes this metric, which are the topic of the next section.

4. CANONICAL ALGORITHMS

We have seen that resource allocation can be viewed as a constrained optimization problem. This means there is a wide range of numerical techniques for computing the solution. What is interesting to network systems researchers is that some of these techniques translate into distributed algorithms.

We will give one detailed illustration here, for the example of Internet congestion control. The key point to bear in mind, though, is that resource allocation problems all have the same basic structure, given by (2), and so algorithms that work for one application should be translatable to other applications.

Example 2 We noted that TCP solves the optimization problem

$$\text{maximize} \quad \sum_{s\in\{0,1,2,3\}} \frac{-2}{y_s \mathsf{RTT}_s^2} \quad \text{over } y_0, y_1, y_2, y_3 \geq 0$$

$$\text{such that} \quad y_0 + y_1 \leq C_a,\ y_0 + y_2 \leq C_b,\ y_0 + y_3 \leq C_c.$$

A simple heuristic algorithm for solving such an optimization problem is steepest ascent with penalty functions. Invent a penalty function for each of the constraints, say $P_a(y_0 + y_1)$ etc., with the property that it is 0 if $y_0 + y_1 \ll C_a$, and it increases rapidly at $y_0 + y_1 \approx C_a$. It doesn't much matter what the exact penalty function is. Next, consider the unconstrained optimization problem, to pick $y_0, y_1, y_2, y_3 \geq 0$ so as to maximize

$$L(y_0, y_1, y_2, y_3) = \sum_{s\in\{0,1,2,3\}} \frac{-2}{y_s \mathsf{RTT}_s^2} - P_a(y_0 + y_1) - P_b(y_0 + y_2) - P_c(y_0 + y_3).$$

Pick some arbitrary initial values for the y_s. The natural way to update the y_s is to take a small step in the direction of steepest ascent, which is given by the partial derivatives of L with respect to each of its variables. Writing $p_a = P_a'(y_0 + y_1)$ etc., the direction of steepest ascent is

$$dy_0 = \frac{2}{y_0^2 \mathsf{RTT}_0^2} - (p_a + p_b + p_c), \quad dy_1 = \frac{2}{y_1^2 \mathsf{RTT}_1^2} - p_a,$$
$$dy_2 = \frac{2}{y_2^2 \mathsf{RTT}_2^2} - p_b, \quad dy_3 = \frac{2}{y_3^2 \mathsf{RTT}_3^2} - p_c.$$

TCP actually controls window size rather than rate. If we express these updates in terms of window sizes, e.g. $w_0 = y_0 \mathsf{RTT}_0$, then

$$dw_0 = \Big(2/w_0^2 - (p_a + p_b + p_c)\Big)\mathsf{RTT}_0. \qquad (3)$$

In fact, TCP increases its window at rate $1/\mathsf{RTT}_0$ in the absence of drops, and it decreases its window by $w_0/2$ when there is a drop, and drops occur at rate $(p_a + p_b + p_c)w_0/\mathsf{RTT}_0$, giving

$$dw_0 = 1/\mathsf{RTT}_0 - (p_a + p_b + p_c)w_0^2/(2\mathsf{RTT}_0) \qquad (4)$$

where p_j is the loss rate at link j. This is proportional to the formula (3), but multiplied by the factor $w_0^2/(2\mathsf{RTT}_0^2)$. It is still a small step of ascent, but it is not steepest ascent. ♢

There are several important lessons to draw from this example.

- Once we have decided on the optimization problem, the most naive optimization method possible (i.e. steepest ascent) suggests to us a distributed algorithm.
- The difference between (3) and (4) is the intellectual content of systems research. There is little merit in cranking a handle and producing a canonical solution. There is substantial merit in testing the canonical solution, finding its practical shortcomings, and fixing it. In this case, the difference is that TCP's adjustements are smaller when RTT is larger, which is a way of preventing instability.
- The canonical algorithm tells us how to adapt at the level of flow rates. To turn this into an actual packet-level algorithm also takes systems-level inspiration.

An optimization problem like the one we started with can be written in a variety of ways. Here is a rough outline of ways of writing it, and of the corresponding distributed algorithms. This is not the place to give details; a fuller and rather technical account of how they apply to congestion control is given in [12].

- The above formulation, in which the variables are the flow rates, leads to TCP-style congestion control.
- The dual optimization problem, in which the variables are dual variables at the resources, leads to RCP-style congestion control.
- If we extra variables for flow rates on each link, e.g. $y_{0,a}$, $y_{0,b}$ and $y_{0,c}$, and add flow conservation constraints $y_{0,a} = y_{0,b} = y_{0,c}$, we obtain backpressure-style congestion control.

For the application to distributed algorithms see [11].

5. DISCUSSION

We now consider two directions for further discussion with students. The first is about design principles for control planes for resource allocation, and the second is about policy.

Design principles for a resource control plane.

The most important guidance from the canonical algorithm in Section 4 is that it tells us a signalling architecture: it tells us that there are capacity constraints, and each needs to send a signal to the users who are using it, and users need to respond to their aggregate feedback. The other styles of canonical algorithm (mentioned at the end of that section) correspond to different signalling architectures, namely signals from users to resources for RCP-style algorithms, and signals between neighbouring resources for backpressure-style algorithms.

The signalling architecture does not depend on the weights nor on α, in the general-purpose objective (2). In other words, the outcome you want the network to achieve (whether it is max-min fairness, maximum efficiency, or something in between) has no bearing on the fundamental signalling architecture. Conversely, if the right signals are not there, it will be hard to achieve any sensible resource allocation.

This can lead to an informed discussion about signalling mechanisms. For example, what are the signalling mechanisms in DOCSIS, or wifi, or overloaded distributed hash tables? Do they measure all the constrained resources? In networks with strict isolation between users, what sort of resource allocation outcomes are possible?

Policy and network neutrality.

The most important lesson from Section 3 is that there is always a normative decision about who gets how much. The simple-minded view of network neutrality, often seen expressed on popular sites such as `slashdot.org`, is "I paid for 8 Mb/s, so I should get 8 Mb/s." Where in the feasible region is this operating point? Is it a useful way to think about network neutrality? See also [2].

To achieve a given resource allocation outcome, there needs to be either cooperation, or incentives to cooperate, or policing. This can lead to a discussion about control in various networks. What specific resource allocation was Comcast trying to achieve, by using DPI to throttle bittorrent traffic? How could users of a data center such as [3] cheat by hiding their true identities? This also ties in nicely with the conex working group at the IETF.

References

[1] D. Bertsekas and R. Gallager. *Data networks*. Prentice Hall, 1987.

[2] Bob Briscoe. Flow rate fairness: dismantling a religion. *ACM/SIGCOMM CCR*, 2007.

[3] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinsku, Scott Shenker, and Ion Stoica. Dominant resource fairness: fair allocation of multiple resource types. In *Proc. NSDI*, 2011.

[4] F. P. Kelly. Loss networks. *Annals of Applied Probability*, 1, 1991.

[5] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49, 1998.

[6] Fabius Klemm, Jean-Yves Le Boudec, and Karl Aberer. Congestion control for distributed hash tables. In *Proc. IEEE NCA*, 2006.

[7] Jean-Yves Le Boudec. Rate adaptation, congestion control and fairness: a tutorial. Retrieved 28 April 2011, 2008. URL `http://ica1www.epfl.ch/PS_files/LEB3132.pdf`.

[8] Laurent Massoulié and James Roberts. Bandwidth sharing: objectives and algorithms. *IEEE/ACM Transactions on Networking*, 2002.

[9] J Mo and J Walrand. Fair end-to-end window-based congestion control. *IEE/ACM Transactions on Networking*, 2000.

[10] J. W. Roberts and L. Massoulié. Bandwidth sharing and admission control for elastic traffic. In *Proc. ITC Specialist Seminar*, 1998.

[11] Devavrat Shah and Damon Wischik. Fluid models of congestion collapse in overloaded switched networks. Submitted, 2011. URL `http://www.cs.ucl.ac.uk/staff/d.wischik/Research/overload.html`.

[12] Srinivas Shakkottai and R. Srikant. Network optimization and control. *Foundations and trends in networking*, 2007.

[13] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 1992.

[14] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath TCP. In *Proc. NSDI*, 2011.