Calculating flow level performance under balanced fairness

T. Bonald[1] and J. Virtamo[2]

[1]France Telecom R&D
38-40, rue du Général Leclerc
92794 Issy-les-Moulineaux Cedex 9, France

[2]Networking Laboratory
Helsinki University of Technology
P.O. Box 3000, FIN-02015 HUT, Finland

May 22, 2003

Abstract

We consider a data network whose resources are shared by a dynamically varying number of elastic flows. It has been recently found that there exist allocations with the property that the stationary distribution of the number of flows in progress on different routes depends only on the traffic loads on these routes. Balanced fairness refers to the most efficient of such allocations. In this paper we present an efficient method for calculating performance metrics such as flow throughputs for networks obeying balanced fairness. The method, which notably applies to the practically interesting case of concentration tree networks, is based on a recursive algorithm for computing the normalization constant of the stationary distribution. This algorithm is extended to include the case where the rate of each flow is additionally limited by an external constraint representing the user's access line for instance. The method is also shown to be applicable to the case of a finite user population. Several examples are worked out.

Keywords: balanced fairness, recursion for normalization constant, flow throughput, concentration tree

1 Introduction

Most traffic in today's Internet is generated by the transfer of documents such as files or web pages. This traffic is elastic in that the duration of each transfer depends on network congestion. Each document is split into a sequence of packets, referred to as a flow, whose sending rate is adapted in response to congestion indications such as packet losses, typically under the control of TCP. The quality of the transfer then depends on the time required to successively transfer all the packets of the flow. In this sense, network performance for elastic traffic is mainly manifested at flow level and can be gauged by measures such as the average per flow throughput, i.e., the ratio of mean flow size to mean flow duration.

Performance studies for elastic traffic have traditionally focussed on the packet level behaviour of various congestion control algorithms, including TCP. While the design of an efficient congestion control is a critical issue, it may be argued that the average per flow throughput on a given network route depends more significantly on the number of flows sharing the corresponding links, which varies as new flows are generated and others cease. To analyze flow level behaviour in such a dynamic setting, idealized models are needed. An appropriate abstraction in this context entails entirely disregarding packet level phenomena and considering the flow content as a fluid which is transmitted as a continuous stream through the network. It is assumed in this fluid model that rate changes occur instantaneously and simultaneously on all links on every flow arrival or departure. We assume flows occur in sessions consisting of a succession of flows and separating "think times" and that sessions occur as a Poisson process.

As there are many flows with different routes being transmitted simultaneously, the evolution of the number of flows depends on how network resources are allocated. Most work has focussed on so-called utility based allocations, where bandwidth is shared so as to maximise some utility function of the instantaneous flow rates [10]. Examples of such allocations are classical max-min fairness [1] and Kelly's proportional fairness [9]. Crucially, the optimality of utility based allocations is defined for a static composition of flows. If the true random nature of traffic were taken into account, it would be necessary to define utility in terms of the performance of individual finite duration flows. In this case, it is not obvious that max-min or proportional fairness are optimal in any real sense. In random traffic, performance and therefore utility depend in general on the precise statistics of offered traffic and are virtually impossible to evaluate analytically.

An alternative notion of fairness, called "balanced fairness", has been introduced by Bonald and Proutière [2]. When flows share bandwidth with balanced fairness, performance is insensitive to detailed traffic characteristics such as the distribution of flow sizes and think-time durations. The name derives from the set of detailed balance relations satisfied by the instantaneous rates allocated to individual flows, which constitute necessary and sufficient conditions for insensitivity in the underlying stochastic networks [13, 3]. The insensitivity is such that the stationary distribution of the number of flows in progress, and consequently the average per flow throughput, depend only on the expected traffic offered on each route. Balanced fairness thus generalizes to a network context the insensitivity of bandwidth shar-

ing of an isolated bottleneck link introduced in [4].

Insensitivity is the key to simple and robust performance results. Network dimensioning rules can be developed based on traffic intensity forecasts only, independently of the complex traffic structure which is continually evolving as new applications gain popularity. Balanced fairness can thus be viewed as a bandwidth sharing objective to be realized by appropriate packet level mechanisms. Alternatively, one might consider the mere existence of an insensitive allocation as evidence that more readily realized allocations do not depend significantly on traffic characteristics beyond expected demand. Simulations show for instance that the performance of balanced fairness is generally close to that of max-min fairness [2]. This suggests that balanced fairness is in fact a good candidate to approximate the behaviour of elastic traffic in the Internet.

While the performance of balanced fairness is insensitive to detailed traffic characteristics, it is still a complex function of the capacity of all links and the traffic intensity on all routes. An efficient recursive algorithm to compute performance metrics for a certain class of topologies was presented in [6]. The algorithm has been implemented in Mathematica and readily provides numerical and symbolic evaluations. We review and extend this algorithm in this paper. In particular, we prove that a crucial technical assumption on which the method relies, while not generally valid, does hold for the practically interesting case of concentration tree networks. We extend the algorithm to include the case where the rate of each flow is additionally limited by an external constraint representing the user's access line for instance. The method is also shown to be applicable to the case of a finite user population where sessions do not arrive as a Poisson process.

The rest of this paper is organized as follows. Section 2 gives a brief review of the notion of balanced fairness and its basic properties. We prove in Theorem 1 that balanced fairness is Pareto efficient in concentration tree networks, which implies that the recursive algorithm developed in Section 3 indeed applies to this class of networks. In Section 4 several examples of the recursion are provided and numerical evaluations illustrate the accuracy of simple approximations. Sections 5 and 6 present the extensions of the algorithm to the cases of limited flow rates and a finite user population, respectively. The results are summarized in Section 7.

2 Balanced fairness

2.1 Basic notions

The following is a short summary of the notion of balanced fairness and the network model to which it pertains; for a full account readers are referred to [2, 5].

The network consists of a set of links $\mathcal{L} = \{1, \ldots, L\}$ where link l has a capacity C_l. A random number of flows compete for the bandwidth of these links. There are N classes of flows where each class i is characterized by a route $\mathcal{R}_i$ consisting of a set of links. When

link l is on route $\mathcal{R}_i$ we use the natural notation $l \in \mathcal{R}_i$. Conversely, defining $\mathcal{F}_l$ to be the set of flow classes going through link l we can write equivalently $i \in \mathcal{F}_l$. The mean volume of information offered by flows in class i per unit time, i.e., the load of class i, is denoted ρ_i. The network state is defined by the vector $x = (x_1, \ldots, x_N)$, where x_i is the number of class-i flows in progress.

The total capacity $\phi_i(x)$ allocated to class-i flows is assumed to be shared equally between these flows and to depend on the network state x only. The capacity allocation must satisfy the capacity constraints,

$$\sum_{i \in \mathcal{F}_l} \phi_i(x) \leq C_l, \qquad \forall\, l \in \mathcal{L}. \tag{1}$$

The allocation is said to be balanced if

$$\frac{\phi_i(x - e_j)}{\phi_i(x)} = \frac{\phi_j(x - e_i)}{\phi_j(x)}, \qquad \forall\, i, j,\ x_i > 0,\ x_j > 0,$$

where e_i is a N-vector with 1 in component i and 0 elsewhere. The balance property implies that there is a balance function $\Phi(x)$ such that

$$\phi_i(x) = \frac{\Phi(x - e_i)}{\Phi(x)}, \qquad \forall\, i,\ x_i > 0.$$

Basically any positive function $\Phi(x)$ defines a balanced allocation. As shown in [2] there is a unique balanced allocation such that for any network state x all the capacity constraints (1) are satisfied and at least one of them is satisfied as an equality, i.e., at least one network link is saturated. For this allocation, the balance function is obtained recursively from

$$\Phi(x) = \max_l \left\{ \frac{1}{C_l} \sum_{i \in \mathcal{F}_l} \Phi(x - e_i) \right\}. \tag{2}$$

This allocation is referred to as balanced fairness. Any link l that realizes the maximum in (2) is saturated in state x. There may be several links saturated in this state. We denote the set of all saturated links by $\sigma(x)$ and call it the saturation set of state x (under balanced fairness).

Assuming Poisson flow arrivals and exponential flow size distributions, it may readily be verified from the balance property that the invariant measure is given by

$$\pi(x_1, \ldots, x_N) = \Phi(x_1, \ldots, x_N) \rho_1^{x_1} \cdots \rho_N^{x_N}. \tag{3}$$

This result, however, has a much wider validity. As shown in [2], the bandwidth sharing network can be identified with a so-called Whittle network of processor sharing servers (cf. [13]). The insensitivity properties of Whittle networks allow us to conclude that the invariant measure (3) is valid for much more general traffic characteristics. Flow sizes and think time durations can have quite general distributions and need not be independent. The number of flows per session can be generally distributed. The only requirement is that sessions arrive as a Poisson process [2, 5].

An important role is played by the normalization constant,

$$G(\rho) = G(\rho_1, \ldots, \rho_N) = \sum_{x_1=0}^{\infty} \cdots \sum_{x_N=0}^{\infty} \Phi(x_1, \ldots, x_N) \rho_1^{x_1} \cdots \rho_N^{x_N},$$

where the traffic load vector is denoted $\rho = (\rho_1, \ldots, \rho_N)$. $G(\rho)$ may be identified as the generating function of the balance function $\Phi(x)$ and thus contains the same information. In particular, $\Phi(x)$ itself and the performance measures can be derived from $G(\rho)$. A key performance measure for class-i flows is the flow throughput, $\gamma_i = \mathrm{E}[S_i] / \mathrm{E}[T_i]$, where $\mathrm{E}[S_i]$ and $\mathrm{E}[T_i]$ are the mean flow size and sojourn time, respectively. Expanding by the flow arrival rate λ_i and applying Little's result we have $\gamma_i = \rho_i / \mathrm{E}[x_i]$. The denominator can be obtained by derivation yielding

$$\gamma_i = \frac{\rho_i}{\mathrm{E}[x_i]} = \frac{G(\rho)}{\frac{\partial}{\partial \rho_i} G(\rho)} = \frac{1}{\frac{\partial}{\partial \rho_i} \log G(\rho)}. \tag{4}$$

In the rest of the paper we concentrate on an efficient method for calculating the normalization constant and flow throughputs for particular network configurations.

2.2 Pareto efficiency

Expression (2) suggests a recursive method can indeed be applied to those network topologies for which it is possible to identify the saturation set $\sigma(x)$ of any state x. In the practically interesting case of concentration tree networks, the saturated links can be easily identified provided the allocation is Pareto efficient, i.e., for any class i and any state x such that $x_i > 0$, there is a saturated link on route $\mathcal{R}_i$. By a concentration tree we mean a connected network without any loop, with all the routes going through a specific node, called the root. The method is the following [6]. First allocate to the x_i class-i flows, $x_i > 0$, the capacity of their leaf link. Proceeding from the leaves towards the root, always apply the capacity constraint of the links to the aggregate flows, if any (with the convention that the constraint is also applied when the capacity of a link is equal to that allocated to the aggregate). The uppermost links that are constraining, and only those, are saturated. Based on this construction, it is easy to show that a Pareto efficient allocation in a concentration tree has the following properties:

1. For each state x there is a unique set of saturated links, denoted $\sigma^{\mathrm{P}}(x)$ and called the Pareto efficient saturation set of state x.

2. The Pareto efficient saturation set $\sigma^{\mathrm{P}}(x)$ consists of links l with the property: l can be saturated in state x whereas no link above l, i.e., closer to the root, can be saturated. In particular, the set $\sigma^{\mathrm{P}}(x)$ is uniquely defined given $\{i : x_i > 0\}$, i.e. the set of indices of the active classes.

3. The saturation set $\sigma^{\mathrm{P}}(x)$ defines a partition of the active flow classes such that for each i with $x_i > 0$ there is one and only one link $l \in \sigma^{\mathrm{P}}(x)$ such that $i \in \mathcal{F}_l$.

While utility based allocations are Pareto efficient, this property does not hold in general for balanced fairness, a notable counter example being provided by so-called homogeneous hypercycles, cf. [2]. On the other hand, the Pareto efficiency of balanced fairness has been proven for some simple topologies such as line and parking lot configurations, discussed later. In [6], it was conjectured that Pareto efficiency also holds for concentration trees. This result is now established by Theorem 1 below, from which we deduce that $\sigma(x) = \sigma^{\mathrm{P}}(x)$ for all x in view of property 1.

Theorem 1 *Balanced fairness is Pareto-efficient in concentration trees.*

The proof of the Theorem is given in Appendix A.

In section 5 we shall consider an extension where each flow of class i is additionally constrained by a rate limit a_i. For the analysis of the extended system, the following corollary will be needed:

Corollary 1 *Balanced fairness is Pareto-efficient in concentration trees with flow rate limits.*

Proof. For any finite x, a tree with limited access rates a_i is equivalent to a tree where each class i is divided into x_i subclasses, each connected to the tree by a link with capacity a_i (and only those states need to be considered, where the occupancy of each subclass is at most one).

3 Recursive algorithm

We first give the recursion for the normalization constant of the stationary distribution, then that for the throughputs.

3.1 Recursion for the normalization constant

Let $\mathcal{I}_k = \{i_1, \ldots, i_k\}$ be a k-tuple, $k \leq N$, of unequal indices, $1 \leq i_1 < \ldots < i_k \leq N$. Define a k-dimensional set of states $\Omega_{\mathcal{I}_k}$ as follows,

$$\Omega_{\mathcal{I}_k} = \{x : x_i > 0 \text{ if and only if } i \in \mathcal{I}_k\},$$

i.e., the set of states where there are active flows in each of the classes represented by the index set $\mathcal{I}_k$, and only in those. Specifically, we define $\mathcal{I}_0$ to mean the empty set $\emptyset$ and $\Omega_\emptyset$ to mean the set consisting solely of the zero state $\Omega_\emptyset = \{(0, \ldots, 0)\}$. The whole state space is decomposed as

$$\Omega = \sum_{k=0}^{N} \sum_{\mathcal{I}_k} \Omega_{\mathcal{I}_k}.$$

The decomposition Ω for a 3-dimensional case, $\Omega = \Omega_\emptyset + \Omega_1 + \Omega_2 + \Omega_3 + \Omega_{1,2} + \Omega_{1,3} + \Omega_{2,3} + \Omega_{1,2,3}$, is illustrated in Figure 1 (the set $\Omega_{1,2,3}$ not shown).

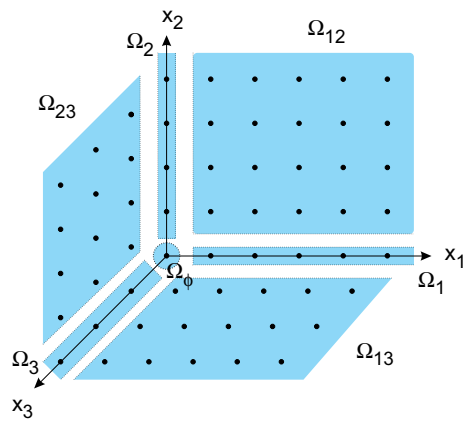


Figure 1: Decomposition of the state space.

Defining partial sums over the sets $\Omega_{\mathcal{I}_k}$,

$$G_{\mathcal{I}_k} = \sum_{x \in \Omega_{\mathcal{I}_k}} \Phi(x_1, \ldots, x_N) \rho_1^{x_1} \cdots \rho_N^{x_N},$$

the normalization constant $G(\rho)$ can be decomposed accordingly,

$$G(\rho) = \sum_{k=0}^{N} \sum_{\mathcal{I}_k} G_{\mathcal{I}_k}(\rho). \tag{5}$$

Now we introduce the central assumption that for each $\mathcal{I}_k$ it is possible to identify at least one link that is saturated in all the states of $\Omega_{\mathcal{I}_k}$ (the links that are saturated are always uniquely defined by (2)). Note that this assumption is not valid for any network topology so that the algorithms described below do not have general applicability. The assumption allows us to derive a recursion expressing $G_{\mathcal{I}_k}(\rho)$ in terms of the $G_{\mathcal{I}_{k-1}}(\rho)$, where $\mathcal{I}_{k-1} \subset \mathcal{I}_k$ (one index in the set $\mathcal{I}_k$ dropped). The number of the different $G_{\mathcal{I}_k}$ is 2^N, which leads to a manageable recursion up to $N = 10, \ldots, 15$. The number is anyhow far less than the number of states, n^N, to be evaluated by the basic recursion (2) when the state space is truncated in each direction at n (typically n is of the order $10, \ldots, 100$).

The recursion is best illustrated by an example. Consider a 2-dimensional set $\Omega_{i,j}$. By our assumption, in all states of this set a given link is saturated. Let the capacity of this link be C. One or both of flow classes i and j must go through the link. For the sake of discussion, we assume that both classes do so. In view of (2) and (3), we deduce:

$$\forall x \in \Omega_{i,j}, \quad \pi(x) = \frac{\rho_i}{C} \pi(x - e_i) + \frac{\rho_j}{C} \pi(x - e_j).$$

Each state x in set Ω_i contributes as a "source of recursion" to the state sum $G_{i,j}(\rho)$ an amount equal to its own measure $\pi(x)$ times the expression

$$\frac{\rho_j}{C} \cdot \frac{1}{1 - (\frac{\rho_i}{C} + \frac{\rho_j}{C})}.$$

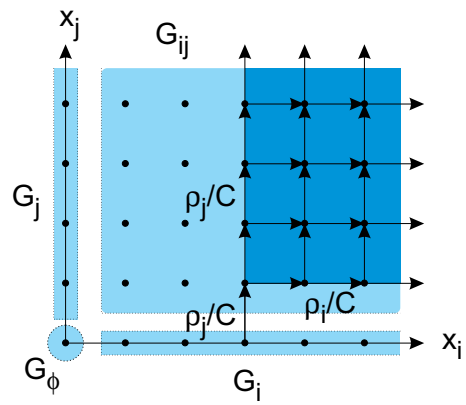


Figure 2: Recursion for the normalization constant.

The first factor comes from the "bridge" and the second factor represents the infinite sum, $S_{i,j}$, over the area indicated in dark grey in Figure 1, when the value the lower left corner of the dark area is fixed to 1. Then the values of the neighbouring points are ρ_i/C and ρ_j/C and we see that $S_{i,j}$ is obtained as the solution of the equation

$$S_{i,j} = 1 + \frac{\rho_i}{C} S_{i,j} + \frac{\rho_j}{C} S_{i,j}.$$

Thus the overall contribution of the set Ω_i to $G_{i,j}(\rho)$ is

$$\frac{\rho_j}{C} \cdot \frac{1}{1 - (\frac{\rho_i}{C} + \frac{\rho_j}{C})} \cdot G_i(\rho).$$

A similar contribution comes from the set Ω_j and the sought for recursion is

$$G_{i,j}(\rho) = \frac{\rho_j G_i(\rho) + \rho_i G_j(\rho)}{C - (\rho_i + \rho_j)}.$$

In general, we have the recursion

$$G_{\mathcal{I}}(\rho) = \frac{\sum_{i \in \mathcal{I}'} \rho_i G_{\mathcal{I} \setminus \{i\}}(\rho)}{C_{\sigma(\mathcal{I})} - \sum_{i \in \mathcal{I}'} \rho_i}, \tag{6}$$

where for any set of classes $\mathcal{I}$, $\sigma(\mathcal{I})$ denotes the link which is saturated in any state $x \in \Omega_{\mathcal{I}}$ and $\mathcal{I}' = \mathcal{I} \cap \mathcal{F}_{\sigma(\mathcal{I})}$ stands for those classes $i \in \mathcal{I}$ that go through link $\sigma(\mathcal{I})$. If $\sigma(\mathcal{I})$ is not unique any of the saturated links can be used as the basis for the recursion.

3.2 Recursion for the throughput

As noted before the throughput can be derived from the normalization constant, eq. (4). One possibility then is to use the recursion (6) to find an explicit expression for the normalization

constant, as in the examples presented later, and to obtain the throughput by derivation. In practice, however, when the number of classes is large the expression for the normalization constant easily becomes too cumbersome to be handled by hand (or even by a symbolic program like Mathematica). For such cases it is desirable to have a more direct way of calculating the throughput numerically. Starting with

$$\gamma_i = \frac{G(\rho)}{\frac{\partial}{\partial \rho_i} G(\rho)},$$

the denominator is decomposed using (5) as

$$\frac{\partial}{\partial \rho_i} G(\rho) = \sum_{k=0}^{N} \sum_{\mathcal{I}_k} \frac{\partial}{\partial \rho_i} G_{\mathcal{I}_k}(\rho) \equiv \sum_{k=0}^{N} \sum_{\mathcal{I}_k} H^{(i)}_{\mathcal{I}_k}(\rho),$$

and each term is obtained by derivation of (6),

$$H^{(i)}_{\mathcal{I}_k}(\rho) \equiv \frac{\partial}{\partial \rho_i} G_{\mathcal{I}_k}(\rho) = \frac{1_{i \in \mathcal{I}'_k}(G_{\mathcal{I}_k}(\rho) + G_{\mathcal{I}_k \setminus \{i\}}(\rho)) + \sum_{j \in \mathcal{I}'_k} \rho_j H^{(i)}_{\mathcal{I}_k \setminus \{j\}}(\rho)}{C_{\sigma(\mathcal{I}_k)} - \sum_{j \in \mathcal{I}'_k} \rho_j},$$

where 1_A is the indicator function having value 1 when event A is true and 0 otherwise. This recursion for the $H^{(i)}_{\mathcal{I}_k}$ can be applied in parallel with the recursion (6) for the $G_{\mathcal{I}_k}$.

4 Application to specific network topologies

We now apply the above algorithm to a number of network topologies for which the saturation property described in section 3.1 holds.

4.1 Line

This configuration consists of $L = n$ links with capacities C_i and $N = n + 1$ classes or routes. Route 0 goes through all links and each of routes $i = 1, \ldots, n$ only goes through link i. When only route 0 is active obviously the link with minimum capacity, denoted by C, is saturated. Whenever a route $i = 1, \ldots, n$ is active link i is saturated (Lemma 1 of [5]).

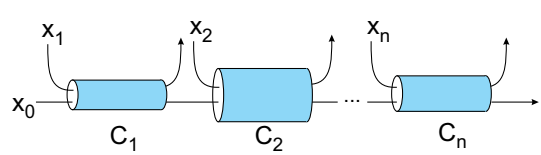


Figure 3: Line configuration.

In this special case it is advantageous to redefine the sets in order to handle even larger groups of states together. We define:

$$\Omega_i = \{x : x_j = 0 \text{ for } j > i\},$$

and correspondingly $G_i(\rho)$ is the state sum over Ω_i. One easily sees that

$$\begin{cases} G_0(\rho) &= \dfrac{C}{C-\rho_0}, \\ G_i(\rho) &= \left(1 + \dfrac{\rho_i}{C_i - \rho_0 - \rho_i}\right) \cdot G_{i-1}(\rho) \quad = \quad \dfrac{C_i - \rho_0}{C_i - \rho_0 - \rho_i} \cdot G_{i-1}(\rho). \end{cases}$$

Thus we have the result

$$G(\rho) = \frac{C}{C-\rho_0} \cdot \frac{C_1 - \rho_0}{C_1 - \rho_0 - \rho_1} \cdots \frac{C_n - \rho_0}{C_n - \rho_0 - \rho_n}.$$

Using (4) the throughput of flow 0 is found to be

$$\gamma_0 = \left(\frac{1}{C-\rho_0} + \sum_{l=1}^{n} \left(\frac{1}{C_l - \rho_l - \rho_0} - \frac{1}{C_l - \rho_0}\right)\right)^{-1},$$

while the throughput of other flows is $\gamma_i = C_i - \rho_i - \rho_0$, for $i = 1, \ldots, n$ (cf. [5]).

4.2 Parking lot

In the parking lot configuration, $L = n$ links with capacities $C_1 \leq C_2 \leq \ldots \leq C_n$ carry $N = n$ classes of flows. Class 1 flows go through all the links, class 2 flows go through links 2 to n, and, in general, class i flows go through links i to n.

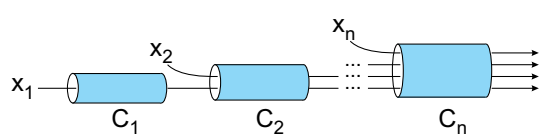


Figure 4: Parking lot configuration.

Whenever $x_i > 0$ but $x_j = 0$ for $j > i$, link $i = 1$ is saturated (this follows again from Lemma 1 of [5]). It is again advantageous to use the aggregate sets Ω_i, $i = 1, \ldots, n$, of the previous section. Now one finds that

$$\begin{cases} G_1(\rho) &= \dfrac{C_1}{C_1 - \rho_1}, \\ G_i(\rho) &= \left(1 + \dfrac{\rho_i}{C_i - (\rho_1 + \cdots + \rho_i)}\right) \cdot G_{i-1}(\rho) \quad = \quad \dfrac{C_i - (\rho_1 + \cdots + \rho_{i-1})}{C_i - (\rho_1 + \cdots + \rho_i)} \cdot G_{i-1}(\rho), \end{cases}$$

from which it follows on denoting the link i load by $R_i = \sum_{j=1}^{i} \rho_j$,

$$G(\rho) = \frac{C_1}{C_1 - R_1} \cdot \frac{C_2 - R_1}{C_2 - R_2} \cdots \frac{C_n - R_{n-1}}{C_n - R_n}. \tag{7}$$

Throughputs are again obtained by eq. (4)

$$\gamma_i = \left(\frac{1}{C_i - R_i} + \sum_{l=i+1}^{n} \left(\frac{1}{C_l - R_l} - \frac{1}{C_l - R_{l-1}} \right) \right)^{-1}, \qquad i = 1, \ldots, n. \tag{8}$$

4.3 Concentration trees

We now consider tree topologies representing access networks. Routes are assumed to attain the network core via a common link constituting the root of the access network. The routes converge progressively over a number of levels. An example of a 4-level concentration tree is depicted in Figure 7. In the sequel, we say that link l_1 is attached to link l_2 if l_2 is next to l_1 on the route from l_1 to the root. It follows from Theorem 1 and property 2 of Pareto efficient allocation that in concentration trees for all $x \in \Omega_{\mathcal{I}}$ a given set of links is saturated thus allowing the use of the recursive method.

2-level trees with three branches

As a first example of concentration trees we consider a 2-level tree which has a common link, the root, with capacity C_0 and three branches with capacities C_1, C_2 and C_3 attached to it, see Figure 5. We assume that $C_i \leq C_0$ for all i and $C_1 + C_2 + C_3 > C_0$; otherwise some of the links would be redundant. Without loss of generality we can further assume that $C_1 \geq C_2 \geq C_3$. Then the pairwise sums have the ordering $C_1 + C_2 \geq C_1 + C_3 \geq C_2 + C_3$.

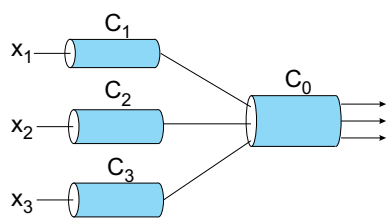
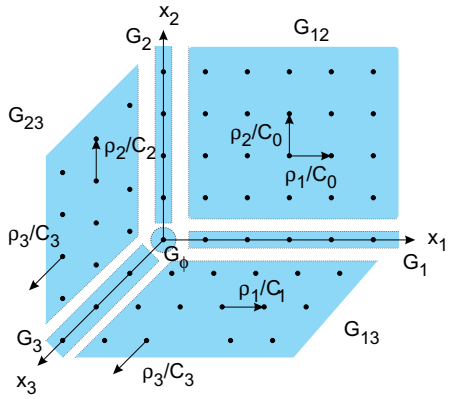


Figure 5: 2-level concentration tree with three branches.

In the analysis, it is necessary to distinguish between different cases according to the ordering of C_0 with respect to these pairwise sums. Below we display the recursion equations for one of the four cases, viz. $C_1 + C_2 \geq C_0 \geq C_1 + C_3$, the analysis in the other cases being similar. In this case the root is saturated if and only if both class 1 and 2 are active (irrespective of the state of class 3). The equations,



$$\begin{cases} G_\emptyset(\rho) &= 1, \\ G_1(\rho) &= \frac{\rho_1 G_\emptyset(\rho)}{C_1-\rho_1}, \\ G_2(\rho) &= \frac{\rho_2 G_\emptyset(\rho)}{C_2-\rho_2}, \\ G_3(\rho) &= \frac{\rho_3 G_\emptyset(\rho)}{C_3-\rho_3}, \\ G_{1,2}(\rho) &= \frac{\rho_1 G_2(\rho)+\rho_2 G_1(\rho)}{C_0-(\rho_1+\rho_2)}, \\ G_{1,3}(\rho) &= \frac{\rho_1 G_3(\rho)}{C_1-\rho_1} = \frac{\rho_3 G_1(\rho)}{C_3-\rho_3}, \\ G_{2,3}(\rho) &= \frac{\rho_2 G_3(\rho)}{C_2-\rho_2} = \frac{\rho_3 G_2(\rho)}{C_3-\rho_3}, \\ G_{1,2,3}(\rho) &= \frac{\rho_1 G_{2,3}(\rho)+\rho_2 G_{1,3}(\rho)+\rho_3 G_{1,2}(\rho)}{C_0-(\rho_1+\rho_2+\rho_3)}, \end{cases}$$

can easily be solved to give the $G_{\mathcal{I}_k}$ and then summed, eq. (5), to give the normalization for case 3 below. In the same way we obtain the normalization constant for the rest of the four cases:

Case 1. $C_2 + C_3 \geq C_0$,

$$G(\rho) = \frac{1}{1-\frac{\rho_1+\rho_2+\rho_3}{C_0}} \left(\frac{1-\frac{\rho_1}{C_0}}{1-\frac{\rho_1}{C_1}} + \frac{1-\frac{\rho_2}{C_0}}{1-\frac{\rho_2}{C_2}} + \frac{1-\frac{\rho_3}{C_0}}{1-\frac{\rho_3}{C_3}} - 2 \right), \tag{9}$$

Case 2. $C_1 + C_3 \geq C_0 \geq C_2 + C_3$,

$$G(\rho) = \frac{1}{1-\frac{\rho_1+\rho_2+\rho_3}{C_0}} \left(\frac{1-\frac{\rho_1}{C_0}}{1-\frac{\rho_1}{C_1}} + \frac{1-\frac{\rho_2+\rho_3}{C_0}}{(1-\frac{\rho_2}{C_2})(1-\frac{\rho_3}{C_3})} - 1 \right),$$

Case 3. $C_1 + C_2 \geq C_0 \geq C_1 + C_3$,

$$G(\rho) = \frac{1}{1-\frac{\rho_1+\rho_2+\rho_3}{C_0}} \cdot \frac{1}{1-\frac{\rho_3}{C_3}} \left(\frac{1-\frac{\rho_1+\rho_3}{C_0}}{1-\frac{\rho_1}{C_1}} + \frac{1-\frac{\rho_2+\rho_3}{C_0}}{1-\frac{\rho_2}{C_2}} - (1-\frac{\rho_3}{C_0}) \right),$$

Case 4. $C_0 \geq C_1 + C_2$,

$$G(\rho) = \frac{1-\frac{\rho_1+\rho_2+\rho_3}{C_0} + \frac{\rho_1}{C_1}\frac{\rho_2}{C_2}\frac{\rho_3}{C_3}(\frac{C_1+C_2+C_3}{C_0}-1)}{(1-\frac{\rho_1+\rho_2+\rho_3}{C_0})(1-\frac{\rho_1}{C_1})(1-\frac{\rho_2}{C_2})(1-\frac{\rho_3}{C_3})}.$$

The two-branch tree ($C_1 \leq C_0$, $C_2 \leq C_0$, $C_1 + C_2 > C_0$) can be obtained as a special case from any of cases 1 through 3 by setting $\rho_3 = 0$, resulting in

$$G(\rho) = \frac{1}{1-\frac{\rho_1+\rho_2}{C_0}} \cdot \left(\frac{1-\frac{\rho_1}{C_0}}{1-\frac{\rho_1}{C_1}} + \frac{1-\frac{\rho_2}{C_0}}{1-\frac{\rho_2}{C_2}} - 1 \right). \tag{10}$$

Together with (9) this suggests the general form of the expression for any number of branches when for any pair of branches i and j it holds $C_i + C_j > C_0$.

As before we obtain from (10) the throughputs of the 2-branch tree. The inverse is given by

$$\gamma_i^{-1} \quad = \quad \frac{1}{C_0-\rho_1-\rho_2} + \frac{\frac{1}{C_i}-\frac{1}{C_0}}{(1-\frac{\rho_i}{C_i})^2}\left(\frac{1-\frac{\rho_1}{C_0}}{1-\frac{\rho_1}{C_1}} + \frac{1-\frac{\rho_2}{C_0}}{1-\frac{\rho_2}{C_2}} - 1\right)^{-1}.$$

Finally, it is instructive to note that by setting $C_2 = C_0$ link 2 becomes redundant and the configuration reduces to a 2-link parking lot. Indeed, in this case (10) is identical to (7).

Homogeneous 2-level trees with any number of branches

When the number of branches in a 2-level tree increases and the link capacities are general, the expressions become more complex. However, if we assume all the branches to have equal capacity, $C_i = C$ for all i, we can handle any number, n, of branches. To start with, we also assume that the loads of all the branches are equal, $\rho_i = \rho$ for all i.

Denote by m the largest integer such that $m \times C \leq C_0$, i.e. $m = \lfloor C_0/C \rfloor$. Applying the recursion to the sets Ω_k where k branches are saturated, $k = 0, \ldots, m$, we get

$$G_k(\rho) = \left(\frac{\rho}{C-\rho}\right)^k, \quad k = 0, \ldots, m.$$

Note that there are $\binom{n}{k}$ sets Ω_k. Applying the recursion to the set $\hat{\Omega}$ where the root is saturated, we deduce

$$\hat{G}(\rho) = \frac{(n-m)\rho}{C_0 - n\rho}\binom{n}{m} G_m.$$

Thus

$$G(\rho) = \sum_{k=0}^{m}\binom{n}{k}\left(\frac{\rho}{C-\rho}\right)^k + \frac{(n-m)\rho}{C_0-n\rho}\binom{n}{m}\left(\frac{\rho}{C-\rho}\right)^m.$$

Figure 6 depicts the throughput, measured in C, obtained with (4) for a system with $n = 20$ branches and C_0 equalling $m = 1, \ldots, n$ times C. The horizontal axis represents the

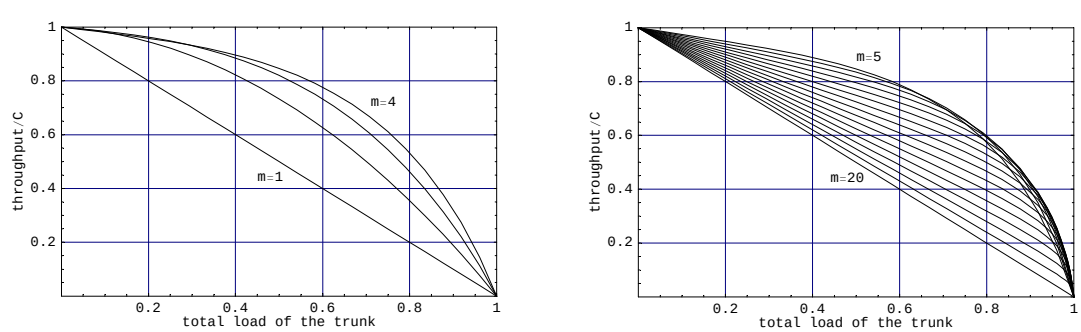


Figure 6: Throughput in a tree with a root and 20 identical branches. Capacity of the root is $m = 1, \ldots, 4$ (left) or $m = 5, \ldots, 20$ (right) times that of a branch.

scaled total load of the root $n\rho/C_0$. Note that the cases $m = 1$ and $m = n$ are identical, both representing effectively a single link system. In the former case, the branch links are redundant. In the latter case, the root link is redundant and the system is equivalent to n independent single link systems.

In case of heterogeneous load distribution, we obtain similarly

$$G(\rho) = \sum_{k=0}^{m} \sum_{i_1,\ldots,i_k} \prod_{p=1}^{k} \frac{\rho_{i_p}}{C - \rho_{i_p}} + \sum_{i_1,\ldots,i_m} \frac{\sum_{i \neq i_1,\ldots i_m} \rho_i}{C_0 - \sum_i \rho_i} \prod_{p=1}^{m} \frac{\rho_{i_p}}{C - \rho_{i_p}}.$$

A 4-level tree

As an example of a larger network consider the 4-level tree of Figure 7 consisting of 10 links with the shown capacities and 9 flow classes numbered by their access links. In particular, we study the throughput of class 10 going through links 10, 7, 3, and 1 as a function of its own load ρ_{10}. The other classes are assumed to have fixed loads as follows: $\rho_1 = \rho_2 = \rho_4 = \rho_6 = \rho_7 = 2$ and $\rho_5 = \rho_8 = \rho_9 = 1$. With these loads, all four links on route 10 have the average residual capacity of 3 units.

As before, one can calculate the normalization constant,

$$G(\rho_{10}) = \frac{6\,(5 - \rho_{10})\,(951 - 411\,\rho_{10} + 46\,{\rho_{10}}^2)}{(3 - \rho_{10})^4},$$

from which one obtains the exact throughput γ_{10} by derivation (4).

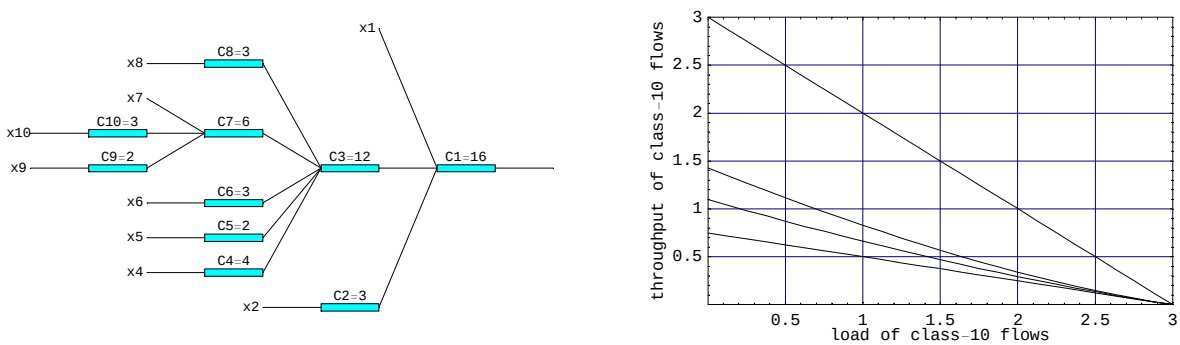


Figure 7: An example of a 4-level tree and comparison of class-10 throughput bounds with the exact result. From bottom up: store-and-forward, parking lot, exact, deterministic.

The throughput is compared with various approximations in Figure 7. The so-called store and forward network[1] was shown in [5] to constitute a lower bound for the throughput for any balanced fair network. This bound amounts to approximating the inverse throughput by the sum of the inverse residual capacities on the route. Another approximation is obtained by neglecting all the capacity constraints outside the considered route, i.e., cross traffic flows

[1]Flows are routed step by step with equal bandwidth sharing on each link.

are made less constrained than they are in reality. This results in a parking lot network with the throughput of eq. (8), which is likely to constitute another (tighter) lower bound. The upper limit is a deterministic approximation derived on constraining the cross traffic classes by links whose capacity equals their offered load. In this stability limit, the cross traffic classes become deterministic and their load can be subtracted from the link capacities on the main route. Throughput is then determined by the bottleneck link; in this example, all four links are bottlenecks with the residual capacity of 3 units.

Since the residual capacities on all the four links are the same, the store and forward bound is a straight line with slope one quarter of that of the deterministic upper bound (always a straight line). When the links outside the considered route are not very heavily loaded the parking lot approximation is not too far from the exact throughput. Clearly, the bounds are tighter for any route where residual capacities are less homogeneous than in the present example.

5 Extension to limited flow rates

So far we have considered a model where flow rates are constrained by network links only. In practice, the rate of a class-i flow may additionally be constrained by a fixed maximum limit a_i representing the user's access line for instance. Balanced fairness is then defined by adding the corresponding constraints in the basic recursion (2) [2].

In [6], the normalization constant was derived for the simple case of a single link fed by several flow classes with different access rate limits. As far as the sum of the access rates of active flows is less than the capacity of the link, bandwidth sharing is inelastic. Only when the sum of the access rates exceeds the link capacity becomes the bandwidth sharing between the flows concerned elastic. The normalization constant can accordingly be divided into two parts. The calculation of the second part, corresponding elastic capacity sharing, is facilitated by the well-known methods for calculating the measure of the boundary between the inelastic and elastic sharing regimes, i.e. what in the literature on multi-bitrate systems is referred to as blocking probability of a given class [11], [12], and by applying the recursion (6). Here we extend these observations to derive an algorithm for calculating the normalization constant of a general concentration tree with access rate limitations.

5.1 The algorithm

The algorithm we present for the calculation of the normalization constant in a concentration tree with limited access rates is a generalization of the algorithm (6). For the present purposes it is, however, advantageous to slightly redefine some of the quantities. We first introduce all the needed definitions and notations.

A tree $\mathcal{T}$ is a set of links l forming a connected network without any loops. Sometimes it is desirable to emphasize the capacities of the links represented by the capacity vector

$C = \{C_l, l \in \mathcal{T}\}$. Then we write $\mathcal{T}(C)$. A link l of a concentration tree is said to be *redundant* if on the route from l to the root of the tree, the root inclusive, there is a link with a capacity less than or equal to C_l, or if the sum of the capacities of the links attached to link l is less than or equal to C_l. If none of the links of a tree is redundant the tree is said to be *irreducible*. From now on we assume that the considered trees are irreducible, i.e., all redundant links have been removed.

A set of links $\mathcal{S}$ of an irreducible tree $\mathcal{T}$ is said to form a *feasible saturation set* if the following hold:

- On the route from any leaf to the root there is at most one link of the set $\mathcal{S}$.
- The capacity C_l of any link l of the tree is greater than the sum of the capacities of those links in the set $\mathcal{S}$ that belong to the subtree $\mathcal{T}_l$ of l, that is $C_l > \sum_{j \in \mathcal{S} \cap \mathcal{T}_l} C_j$.

The latter condition amounts to that in a pruned tree where all the branches not containing any saturated link are removed, all links above a saturated link (closer to the root) are redundant. Note that also the empty set $\emptyset$ is a feasible saturation set.

The set of all feasible saturation sets of tree $\mathcal{T}$ is denoted $\Sigma(\mathcal{T})$. There are several ways to find $\Sigma(\mathcal{T})$ for an irreducible $\mathcal{T}$. One is provided by the recursion:

- The empty set and the set comprising the root solely are feasible saturation sets.
- In addition, feasible saturation sets include all unions of the saturation sets of the (immediate) subtrees, one saturation set from each subtree in the union, such that the sum of the capacities of the links in the union set is less than the capacity of the root.

When this definition is applied recursively, substituting a subtree for the tree, one finally ends up with a tree consisting of a single link only, the saturation sets of which are given by the first rule, and the recursion stops as there are no further subtrees. Below we need also a special notation, $\hat{\Sigma}(\mathcal{T})$, for the set of all feasible saturation sets $\Sigma(\mathcal{T})$ excluding the saturations set comprising the root only.

Now we redefine $\Omega_{\mathcal{S}}$ as follows $\Omega_{\mathcal{S}} = \{x : \sigma(x) = \mathcal{S}\}$, i.e. $\Omega_{\mathcal{S}}$ is the set of all states x for which $\mathcal{S}$ is the saturation set. $G_{\mathcal{S}}$ still means the partial sum over $\Omega_{\mathcal{S}}$: $G_{\mathcal{S}} = \sum_{x \in \Omega_{\mathcal{S}}} \pi(x)$. We also define border set between $\Omega_{\mathcal{S}}$ and $\Omega_{\mathcal{S}'}$ in direction i: $\Omega_{\mathcal{S}|_i\mathcal{S}'} = \Omega_{\mathcal{S}} \cap (\Omega_{\mathcal{S}'} - e_i)$, with element-wise subtraction. The border set $\Omega_{\mathcal{S}|_i\mathcal{S}'}$ comprises those states of $\Omega_{\mathcal{S}}$ where adding one class-i flow changes the saturation set from $\mathcal{S}$ into $\mathcal{S}'$. Similarly we define $G_{\mathcal{S}|_i\mathcal{S}'} = \sum_{x \in \Omega_{\mathcal{S}|_i\mathcal{S}'}} \pi(x)$.

The state sum is decomposed in terms of partial sums $G_{\mathcal{S}}$ for all feasible saturation sets $\mathcal{S}$,

$$G = \sum_{S \in \Sigma(\mathcal{T})} G_{\mathcal{S}}.$$

The algorithm for calculating the $G_{\mathcal{S}}$ consists of three recursive steps. The first step, called reduction, is as follows:

$$G_{\mathcal{S}}[\mathcal{T}] = G_{\emptyset}[\mathcal{T} \mathop{\setminus\!\!\setminus}_{l \in \mathcal{S}} \mathcal{T}_l] \prod_{l \in \mathcal{S}} G_l[\mathcal{T}_l], \tag{11}$$

where $\mathcal{T}_l$ denotes the subtree with link l as the root and $G_l[\mathcal{T}_l]$ is the partial state sum for this subtree for the saturation set consisting solely of the root l of the subtree. $\mathcal{T}\backslash\backslash\mathcal{T}_l$ denotes the tree remaining when subtree $\mathcal{T}_l$ is detached from tree $\mathcal{T}$ and the capacity C_l is subtracted from the capacities of the links on the route from l to the root of $\mathcal{T}$. $\mathcal{T}\backslash\backslash_{l\in\mathcal{S}}\mathcal{T}_l$ denotes the result of repeated application of the subtraction operation over all subtrees $\mathcal{T}_l$ with $l \in \mathcal{S}$. Equation (11) essentially tells that a saturated link l masks the details of the internal state of the subtree $\mathcal{T}_l$ from the other parts of the tree.

The second step, the actual recursion corresponding to (6), gives the partial state sum of a tree with a saturated root,

$$G_l[\mathcal{T}_l] = \frac{\sum\limits_{\mathcal{S}\in\hat{\Sigma}(\mathcal{T}_l)} \prod\limits_{j\in\mathcal{S}} G_j[\mathcal{T}_j] \sum\limits_{i\in\mathcal{F}_l\backslash\cup_{j\in\mathcal{S}}\mathcal{F}_j} \rho_i G_{\emptyset|_i l}[\mathcal{T}_l \mathop{\backslash\backslash}\limits_{j\in\mathcal{S}} \mathcal{T}_j]}{C_l - \sum\limits_{i\in\mathcal{F}_l} \rho_i}.$$

The factor ρ_i in the numerator comes again from the "bridge" from the border states to states where root l is saturated, and the denominator arises from an infinite summation exactly as in the derivation of (6).

Finally, $G_{\emptyset|_i l}[\mathcal{T}_l(C)]$ for any tree $\mathcal{T}_l$ with root l and capacity vector C represents the measure of the border states where the tree has no saturated links but adding one class-i flow brings the root l into saturation. This equals the class-i blocking measure due to the root link l in an equivalent multi-bitrate system,

$$G_{\emptyset|_i l}[\mathcal{T}_l(C)] = G_\emptyset[\mathcal{T}_l(C)] - G_\emptyset[\mathcal{T}_l(C - a_i r_i[\mathcal{T}_l])] - \sum_{j\in\hat{\mathcal{R}}_i[\mathcal{T}_l]} G_\emptyset[\mathcal{T}_l(C)\backslash\backslash\mathcal{T}_j(C)] \cdot G_{\emptyset|_i j}[\mathcal{T}_j(C)], \quad (12)$$

where $\hat{\mathcal{R}}_i[\mathcal{T}_l]$ is the route of class-i flows in tree $\mathcal{T}_l$ up to but excluding the root l and including only non-redundant links, and $r_i[\mathcal{T}_l]$ is a vector of the same form as the capacity vector of tree $\mathcal{T}_l$, with 1 in each entry corresponding to a link used by route $\mathcal{R}_i[\mathcal{T}_l]$ and 0 elsewhere.

The first difference in (12) gives the total blocking measure of class-i flows, considered as inelastic, constant bitrate "calls". The last term subtracts the contribution to the blocking measure by the links on the route to the root. Note that the concept of "blocking probability due link l" as such is not well defined, since there may be states where several links block simultaneously. In such cases we have to attribute the blocking to the lowermost (farthest from the root) of the blocking links. The above formula does precisely this. Equation (12) again provides a recursive definition, expressing $G_{\emptyset|_i l}[\mathcal{T}_l]$ in terms of the corresponding quantity of some of the subtrees of $\mathcal{T}_l$.

In (11) and (12) the state sum over all states of some tree $\mathcal{T}$ with no saturated links, $G_\emptyset[\mathcal{T}(C)]$, corresponds to the state sum of an equivalent multi-bitrate system where each class-i flow (call) has a constant bandwidth requirement a_i and the state space is constrained by the capacities of the links. Calculation of this kind of state sum is a classical problem which allows a nice solution for tree networks by the so-called convolution-truncation algorithm [12]. In this algorithm, each flow class is first associated with a capacity occupancy

distribution corresponding to Poisson distribution of the number of flows. The capacity occupancy distributions of all the classes offered to a given link are convolved and the resulting distribution is then truncated at the capacity of the link. Similarly for each link inside the tree, the truncated capacity distributions of the links attached to this link are convolved and truncated at the capacity of the link. Proceeding from the leaf links to the root, one finally ends up with the truncated capacity occupancy distribution on the root link. The total mass of this distribution gives the desired state sum.

5.2 Example

Here we present some numerical results of the application of the above algorithm for an example network. The example comprises a 2-branch tree with each branch carrying flows of two classes with different access rate limitations, shown by the diagram on the left of Figure 8. The link capacities of the branches are $C_1 = 7$ and $C_2 = 10$, and the common link has the capacity $C_0 = 15$. The access rates of the flows in branch 1 are $a_{1,1} = 1$ and $a_{1,2} = 3$, while for branch 2 the access rates are $a_{2,1} = 2$ and $a_{2,2} = 4$. In the graph of Figure 8 the throughput of each class is given as a function of its own load, with all the other loads being equal to 1. For low loads, the throughput equals the access rate, whereas the throughput goes to zero when the total load of a branch approaches its capacity.

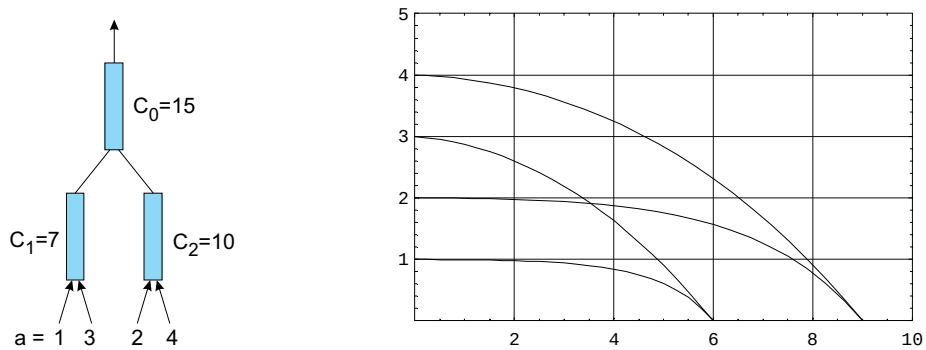


Figure 8: Throughputs of flows in a 2-branch tree with two access rate classes loading each branch.

In Figure 9, we compare the exact throughput of flows in a class with conjectured upper and lower bounds. The system is the same as in the previous example, and the considered class is that with access rate limit $a_{1,2} = 3$ in branch 1. The middle curve of Figure 9 gives the throughput of flows in this class as a function of its own load, when the loads of the other classes equal 1 (i.e., identical to the second uppermost curve in Figure 8).

The conjectured lower bound is again a parking lot bound resulting from relaxing the constraints of the cross traffic streams. In the case of limited access rates, besides the link constraints also the access rate limitations of the cross traffic streams are relaxed. However, for the flow classes with an access rate limitation smaller than that of the considered class, it is sufficient to set these access rates equal to the latter one. In the case of our example,

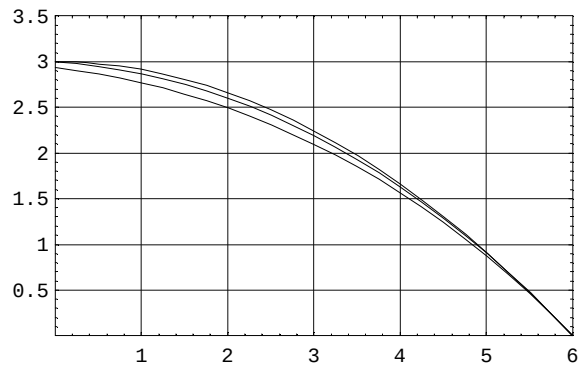


Figure 9: Comparison of conjectured upper and lower bounds with the exact throughput.

this means that we set $a_{1,1} = a_{1,2} = 3$ and $a_{2,1} = a_{2,2} = \infty$. Thus we are left with a simplified system, with branch 1 receiving flows with access rate limit of 3 and load $1+\rho_{2,1}$, and branch 2 receiving non-access-rate-limited flows with load 2.

The conjectured upper bound is obtained by imposing stronger constraints for the cross traffic. For classes with a higher access rate than that of the considered class, the access rates are reduced to that of the considered class. For classes with a lower access rate, the deterministic approach is applied and the load of those classes is subtracted from the link capacities on their route. In the case of our example, the system reduces to one where the capacities of links 0 and 1 are reduced to 14 and 6, respectively, and all flows are access rate limited with rate 3, the load of branch 1 being $\rho_{2,1}$ and that of branch 2 being 2. The corresponding throughput is given by the upper curve in Figure 9. At least in this example the conjectures indeed provide bounds, which in this case are rather tight ones. If deterministic approximation is applied to all the cross traffic streams another, slightly less tight conjectured upper bound is obtained.

6 Finite user population

The insensitivity property of balanced fairness also holds in case of a finite user population. Sessions then do not arrive as a Poisson process but consist of permanent alternating successions of flows and think times. Proceeding exactly as in [2], the corresponding bandwidth sharing network can be identified with a so-called closed Whittle network where each customer represents a flow or a think time. In the general setting considered in [3], the routing is allowed to be reducible, i.e. there may be K classes of customers, with M_k customers in class k, and class-k customers visiting only a subset c_k of the nodes. The subsets $c_1, \ldots, c_K$ form a partition of all the nodes. One of the nodes, i_k, in each class is called the source, and its capacity ψ_k is assumed to depend on the number of customers at this node only. The arrival frequency λ_i at any node i of c_k is defined up to a multiplicative constant per class

by the flow conservation equations,

$$\lambda_i = \sum_{j \in c_k} \lambda_j p_{ji}, \qquad i \in c_k.$$

A unique solution is obtained by requiring for each k, e.g. $\sum_{i \in c_k} \lambda_i = 1$.

As pointed out in [3], the invariant measure of the system is (here given in a slightly different form)

$$\pi(x) = \begin{cases} \Phi(x') \prod_{k=1}^{K} \Psi_k(x_{i_k}) \prod_{i=1}^{N} \rho_i^{x_i}, & \text{for } \sum_{i \in c_k} x_i = M_k, \ \forall k, \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where N is the total number of nodes (including the sources), x' is state vector excluding source nodes, and $\Psi_k(n) = 1/\psi_k(1) \cdots \psi_k(n)$.

Here we show that the normalization constant can be calculated for the closed networks in a similar way as demonstrated for open networks in [6]. To this end denote the normalization constant for fixed populations $M_1, \ldots, M_K$ by $G_{M_1,\ldots,M_K}(\rho)$ and define the K-dimensional generating function $\mathcal{G}(z_1, \ldots, z_K; \rho)$,

$$\mathcal{G}(z_1, \ldots, z_K; \rho) = \sum_{M_1,\ldots,M_K} G_{M_1,\ldots,M_K}(\rho) z_1^{M_1} \cdots z_K^{M_K}.$$

By the summations over the M_k the population constraints in (13) are relaxed, and we end up with unlimited sums over the whole state space, exactly as in the case of an open network,

$$\mathcal{G}(z_1, \ldots, z_K; \rho) = G(\tilde{\rho}') \prod_{k=1}^{K} H_k(z_k \varrho_k),$$

where $G(\rho')$ and $H_k(\varrho_k)$ are the state sums related to $\Phi(x')$ and $\Psi_k(x_{i_k})$, $\varrho_k = \rho_{i_k}$, and $\tilde{\rho}_i = z_{k(i)} \rho_i$ with $k(i)$ denoting the class index k such that $i \in c_k$.

Conversely, from the generating function one can infer the normalization constants $G_{M_1,\ldots,M_K}(\rho)$ and calculate performance metrics in the usual way. In particular, the mean throughput γ_i of node i in the subset c_k, with the populations $M_1, \ldots, M_K$, is

$$\gamma_i = \frac{G_{M_1,\ldots,M_k-1,\ldots,M_K}(\rho)}{\frac{\partial}{\partial \rho_i} G_{M_1,\ldots,M_K}(\rho)},$$

taking into account that the real arrival rate at node i is $\lambda_i G_{M_1,\ldots,M_k-1,\ldots,M_K}(\rho) / G_{M_1,\ldots,M_K}(\rho)$.

Example 1: One PS-link, one thinking stage

This simplest of all possible closed networks provides a non-trivial example of the method. The capacity of the link is denoted by C and the thinking stage is modelled as an infinite system of c-capacity servers.

The generating function is

$$\mathcal{G}(z;\rho,\varrho) = \frac{e^{z\varrho/c}}{1 - z\rho/C}, \tag{14}$$

where ρ and ϱ are the loads of the link and the thinking stage (they are, of course, equal but for calculating the throughput they must be kept separate). Developing (14) in Taylor series in z we can identify the $G_M(\rho,\varrho)$ and using (4) calculate the mean throughput on the link experienced by the flows. For the population sizes $M = 1,\ldots,5$, the results are

$$\gamma_M = \begin{cases} C, & M = 1, \\ \dfrac{c+C}{2c+C}\,C, & M = 2, \\ \dfrac{2c^2+2c\,C+C^2}{6c^2+4c\,C+C^2}\,C, & M = 3, \\ \dfrac{6c^3+6c^2C+3c\,C^2+C^3}{24c^3+18c^2C+6c\,C^2+C^3}\,C, & M = 4, \\ \dfrac{24c^4+24c^3C+12c^2C^2+4c\,C^3+C^4}{120c^4+96c^3C+36c^2C^2+8c\,C^3+C^4}\,C, & M = 5. \end{cases}$$

When $c/C \to 0$, we have $\gamma_M \to C$. In this limit, the flows spend all the time in the thinking stage and the link is always empty. In the opposite limit $c/C \to \infty$, we have $\gamma_M \to C/M$. In this limit, the flows spend all the time on the link equally sharing its capacity.

Example 2: 2-branch tree with separate populations

In the second example we consider a 2-branch tree with separate populations and thinking stages for each branch. Using result (10) for the state sum of a 2-branch tree we can write the generating function

$$\mathcal{G}(z_1,z_2;\rho_1,\rho_2,\varrho_1,\varrho_2) \;=\; e^{\frac{z_1\varrho_1}{c_1}}\, e^{\frac{z_2\varrho_2}{c_2}}\; \frac{\dfrac{1-\frac{z_1\rho_1}{C_0}}{1-\frac{z_1\rho_1}{C_1}} + \dfrac{1-\frac{z_2\rho_2}{C_0}}{1-\frac{z_2\rho_2}{C_2}} - 1}{1 - \frac{z_1\rho_1}{C_0} - \frac{z_2\rho_2}{C_0}},$$

where again the ρ_i stand for the loads of nodes (routes) i and the ϱ_i for the loads of the corresponding thinking stages (and we have $\varrho_i = \rho_i$). Developing this as a Taylor series up to the second power in z_1 and z_2 we can identify the corresponding $G_{M_1M_2}(\rho)$, and calculate the throughput of the tree for the branch-1 route, for $M_1 = 2$ and $M_2 = 2$,

$$\gamma_1 = C_0\,C_1\,\frac{\begin{array}{c} 2\,c_1{c_2}^2C_0C_1 + 2{c_2}^2{C_0}^2C_1 + 2c_1{c_2}^2C_1C_2 + 2c_1c_2C_0C_1C_2 + \\ 2c_2{C_0}^2C_1C_2 + 2c_1{c_2}^2{C_2}^2 + 2c_1c_2C_0{C_2}^2 + c_1{C_0}^2{C_2}^2 + {C_0}^2C_1{C_2}^2 \end{array}}{\begin{array}{c} 4c_1{c_2}^2C_0{C_1}^2 + 2{c_2}^2{C_0}^2{C_1}^2 + 8c_1{c_2}^2{C_1}^2C_2 + 4c_1c_2C_0{C_1}^2C_2 + \\ 2{c_2}^2C_0{C_1}^2C_2 + 2c_2{C_0}^2{C_1}^2C_2 + 4c_1{c_2}^2C_0{C_2}^2 + 4c_1c_2{C_0}^2{C_2}^2 + 2c_1{C_0}^3{C_2}^2 + \\ 8c_1{c_2}^2C_1{C_2}^2 + 4c_1c_2C_0C_1{C_2}^2 + 2{c_2}^2C_0C_1{C_2}^2 + 2c_2{C_0}^2C_1{C_2}^2 + {C_0}^3C_1{C_2}^2 \end{array}}.$$

For $c_1 \to 0$ and $c_2 \to 0$ we get $\gamma_1 = C_1$ as we should. For $c_1 = c_2 \to \infty$ the result is more complicated,

$$\gamma_1 = \frac{1}{2} C_0 C_1 \frac{C_0 C_1 + C_2 (C_1 + C_2)}{2 C_1 C_2 (C_1 + C_2) + C_0 \left({C_1}^2 + {C_2}^2 \right)},$$

which can be identified as the bandwidth share of route-1 flows in a static situation where the resources of the tree are shared under balanced fairness between two flows on route 1 and two flows on route 2. In particular, we note that in the case $C_1 = C_2$ this reduces to $C_0/4$ as expected.

In these two examples we made a power series expansion of the generating function in order to identify the normalization constant $G_{M_1,\ldots,M_K}(\rho)$. As the last example demonstrates, the computations can easily become awkward when the number of classes K and the number of customers in each class grows. There are, however, other techniques to make the inversion such as numerical inversion by path integrals, successfully applied in other contexts [7], [8].

7 Summary

Balanced fairness is a new notion of bandwidth allocation with the very gratifying property that flow level performance metrics are insensitive to detailed traffic characteristics. This is particularly important for data network engineering since performance can be predicted from an estimate of overall traffic volume alone and is independent of changes in the mix of user applications.

The balanced fair allocation for any network is uniquely determined by the basic recursion (2). For larger networks, however, straightforward application of the recursion is hampered by the usual state space explosion problem. Our main contribution is the derivation of a recursive algorithm for directly calculating the normalization constant and flow throughputs for any network for which the set of saturated links depends on the network state through the set of active routes only. While this crucial property is not generally valid, it is provably so for the line, the parking lot and the concentration tree.

An implementation of the algorithm in Mathematica for general concentration trees is publicly available from http://netlab.hut.fi/tutkimus/com2/Qlib/. Several examples of concentration trees including a 4-level tree with 10 links are given. The comparison of exact results with a number of bounds is the prelude to a more thorough future evaluation of approximations useful for practical engineering purposes.

We have extended the algorithm to take account of external limits on flow rate due, for example, to the speed of user access lines. The extended algorithm covers concentration tree topologies. Finally, we demonstrated that the recursive method can be applied for the calculation of the normalization constant and performance metrics also for closed networks with finite flow populations.

Acknowledgments

Part of this work was done during J. Virtamo's visit to France Telecom R&D. He wishes to thank J. Roberts and France Telecom R&D for their kind hospitality. The work was also financially supported by the Academy of Finland.

Appendix A. Proof of Theorem 1

We prove the theorem by induction on x. The assertion is certainly true for $x = 0$ and for $x = e_i$ for all i. Then we have to show that the assertion is true for any x, given that it is true for all $x' < x$.

Reference is made to the four properties of Pareto efficient allocation introduced in subsection 2.2. By property 2, it is easy to see that if there are several links $l \in \sigma^{\mathrm{P}}(x)$, the assertion is decomposed into separate claims for each subtree $\mathcal{T}_l$ with $x^{(l)} < x$, and is true by the induction assumption. Thus, we can assume that $\sigma^{\mathrm{P}}(x) = \{0\}$, where link 0 denotes the root of the tree, and our task is to show that under balanced fairness the root is indeed saturated.

The set of flow classes $\mathcal{F}_0 = I(x)$ can be divided into two disjoint sets, $\mathcal{F}_0 = \mathcal{F}_0' \cup \mathcal{F}_0''$ (non-critical and critical, respectively), such that $\sigma^{\mathrm{P}}(x-e_i) = \{0\}$ for all $i \in \mathcal{F}_0'$ but $0 \notin \sigma^{\mathrm{P}}(x-e_i)$ when $i \in \mathcal{F}_0''$. Then we have,

$$\begin{aligned}
\frac{1}{C_0}\sum_{i\in\mathcal{F}_0}\Phi(x-e_i) &= \frac{1}{C_0}\Big(\sum_{i\in\mathcal{F}_l}\Phi(x-e_i)+\sum_{j\notin\mathcal{F}_l}\Phi(x-e_j)\Big)\\
&\geq \frac{1}{C_0C_l}\Big(C_l\sum_{i\in\mathcal{F}_l}\Phi(x-e_i)+\sum_{i\in\mathcal{F}_l,\,j\notin\mathcal{F}_l}\Phi(x-e_i-e_j)\Big) \qquad (15)\\
= \frac{1}{C_0C_l}\Big(C_l\sum_{i\in\mathcal{F}_0'\cap\mathcal{F}_l}\Phi(x-e_i)&+\sum_{i\in\mathcal{F}_0'\cap\mathcal{F}_l,\,j\notin\mathcal{F}_l}\Phi(x-e_i-e_j)+C_l\sum_{i\in\mathcal{F}_0''\cap\mathcal{F}_l}\Phi(x-e_i)+\sum_{i\in\mathcal{F}_0''\cap\mathcal{F}_l,\,j\notin\mathcal{F}_l}\Phi(x-e_i-e_j)\Big).
\end{aligned}$$

The first two terms are developed as follows

$$\begin{aligned}
C_l\sum_{i\in\mathcal{F}_0'\cap\mathcal{F}_l}\Phi(x-e_i)+\sum_{i\in\mathcal{F}_0'\cap\mathcal{F}_l,\,j\notin\mathcal{F}_l}\Phi(x-e_i-e_j) &\geq \sum_{i\in\mathcal{F}_0'\cap\mathcal{F}_l,\,j\in\mathcal{F}_l}\Phi(x-e_i-e_j)+\sum_{i\in\mathcal{F}_0'\cap\mathcal{F}_l,\,j\notin\mathcal{F}_l}\Phi(x-e_i-e_j)\\
&= \sum_{i\in\mathcal{F}_0'\cap\mathcal{F}_l,\,j\in\mathcal{F}_0}\Phi(x-e_i-e_j) \;=\; C_0\sum_{i\in\mathcal{F}_0'\cap\mathcal{F}_l}\Phi(x-e_i),
\end{aligned}$$

where the last equality is due to the fact that $\sigma(x-e_i) = \sigma^{\mathrm{P}}(x-e_i) = \{0\}$ for all $i \in \mathcal{F}_0'$, i.e. link 0 is saturated under balanced fairness in $x - e_i$. For the last term of (15) we have,

$$\sum_{i\in\mathcal{F}_0''\cap\mathcal{F}_l,\,j\notin\mathcal{F}_l}\Phi(x-e_i-e_j) \;=\; \sum_{i\in\mathcal{F}_0''\cap\mathcal{F}_l}\Big(\sum_{l'\in\mathcal{S}_{\mathcal{F}_0\setminus\mathcal{F}_l}}C_{l'}\Big)\Phi(x-e_i) \;\geq\; (C_0-C_l)\sum_{i\in\mathcal{F}_0''\cap\mathcal{F}_l}\Phi(x-e_i).$$

The first step can be reasoned as follows: By definition, $\sigma^{\mathrm{P}}(x-e_i)$ is different from $\{0\}$ for all $i \in \mathcal{F}_0''$. It consists of a set of links, some possibly within the subtree $\mathcal{T}_l$, some outside it. By property 4, a change in the Pareto efficient saturation set implies a change in the set of active classes. Thus for all $i \in \mathcal{F}_0''$ we must have $x_i = 1$, and in the state $x - e_i$ class i is inactive. None of the links on the route of class i can belong to $\sigma^{\mathrm{P}}(x-e_i)$; otherwise, adding a class-i flow could not bring link 0 into $\sigma^{\mathrm{P}}(x)$. Since all the flow classes $j \in \mathcal{F}_l$ share the same route with class i from link l upwards, it follows that none of these flows goes through a saturated link outside $\mathcal{T}_l$. Consequently, the saturation set $\sigma^{\mathrm{P}}(x-e_i)$, which forms a partition to the flow classes $\mathcal{F}_0 \backslash \{i\}$, does so separately for the flow classes inside $\mathcal{T}_l$ and those outside $\mathcal{T}_l$, $S_{\mathcal{F}_0\backslash\{i\}} = S_{\mathcal{F}_l\backslash\{i\}} \cup S_{\mathcal{F}_0\backslash\mathcal{F}_l}$. The first step is obtained by applying the partition property $\mathcal{F}_0 \backslash \mathcal{F}_l = \cup_{l' \in S_{\mathcal{F}_0\backslash\mathcal{F}_l}} \mathcal{F}_{l'}$, with the $\mathcal{F}_{l'}$ being disjoint,

$$\sum_{j \notin \mathcal{F}_l} \Phi(x - e_i - e_j) \;=\; \sum_{l' \in \mathcal{S}_{\mathcal{F}_0\backslash\mathcal{F}_l}} \sum_{j \in \mathcal{F}_{l'}} \Phi(x - e_i - e_j) \;=\; \sum_{l' \in \mathcal{S}_{\mathcal{F}_0\backslash\mathcal{F}_l}} C_{l'} \Phi(x - e_i),$$

where use has been made of the fact that $l' \in \sigma(x-e_i)$ for all $i \in \mathcal{F}_0'' \cap \mathcal{F}_l$, since $l' \in S_{\mathcal{F}_0\backslash\mathcal{F}_l} \subset S_{\mathcal{F}_0\backslash\{i\}} = \sigma^{\mathrm{P}}(x-e_i)$.
The second step, the inequality, expresses the requirement that for all $i \in \mathcal{F}_0'' \cap \mathcal{F}_l$ adding a class-i flow in state $x - e_i$ has to be able to bring link 0 into $\sigma^{\mathrm{P}}(x)$, taking into account that this can add no more than the amount C_l to the aggregate capacity requirement of the flows outside $\mathcal{T}_l$.

Substituting these inequalities in (15) we finally obtain,

$$\frac{1}{C_0} \sum_{i \in \mathcal{F}_0} \Phi(x - e_i) \;\geq\; \frac{1}{C_l} \sum_{i \in \mathcal{F}_l} \Phi(x - e_i).$$

Thus, link 0 realizes the maximum of

$$\Phi(x) = \max_l \left(\frac{1}{C_l} \sum_{i \in \mathcal{F}_l} \Phi(x - e_i) \right)$$

and is saturated in x completing the proof.

References

[1] D. Bertsekas and R. Gallager, Data Networks (2nd ed.), Prentice Hall, Englewood Cliffs, 1992.

[2] T. Bonald, A. Proutière, Insensitive bandwidth sharing in data networks, Queueing Systems 44 (2003) 69-100.

[3] T. Bonald, A. Proutière, Insensitivity in processor-sharing networks, Performance Evaluation 49 (2002) 193-209.

[4] T. Bonald, A. Proutière, G. Régnié, J. Roberts, Insensitivity results for statistical bandwidth sharing, in: Proceedings of ITC 17, Elsevier, 2001, 125-136.

[5] T. Bonald, A. Proutière, On performance bounds for balanced fairness, Performance Evaluation (2003), to appear.

[6] T. Bonald, A. Proutière, J. Roberts and J. Virtamo, Computational aspects of balanced fairness, in: Proceedings of ITC-18, 2003, to appear.

[7] G.L. Choudhury and W. Whitt, Q2: a new performance analysis tool exploiting numerical transform inversion, in: Proceedings of the Third International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS '95, 18-20 Jan 1995 (1995) 411-415.

[8] G.L. Choudhury, K.K. Leung and W. Whitt, An inversion algorithm to compute blocking probabilities in loss networks with state-dependent rates, IEEE/ACM Transactions on Networking 3 (1995) 585-601.

[9] F.P. Kelly, A. Maulloo and D. Tan, Rate control in communication networks: shadow prices, proportional fairness and stability, Journal of the Operational Research Society 49 (1998) 237-252.

[10] J. Mo and J. Walrand, Fair end-to-end window-based congestion control, IEEE/ACM Transactions on Networking 8 (2000) 556-567.

[11] J. Roberts, U. Mocci, and J. Virtamo (eds.), Broadband Network Teletraffic, Springer-Verlag, Berlin, 1996.

[12] K.W. Ross, Multiservice Loss Models for Broadband Telecommunication Networks, Springer-Verlag, Berlin, 1995.

[13] R. Serfozo, Introduction to Stochastic Networks, Springer-Verlag, Berlin, 1999.