Mathematical Modelling for Computer Networks- Part I **Spring 2013 (Period 4)**

Congestion Control 2: Utility, Fairness and Optimization in Resource Allocation

Lecturers: Laila Daniel and Krishnan Narayanan *Date:11th March 2013*

Abstract

In this lesson we consider how network resources (bandwidth of links) should be allocated to the different flows in the network to satisfy a given fairness criterion. A generic fairness notion called α-*fairness* is introduced and various well-known fairness criteria such as proportional fairness, minimum potential delay fairness and max-min fairness are seen to be special cases of this general notion corresponding to different values of α. The resource allocation problem is formulated as a *convex optimization* problem and the role of *Lagrange multipliers* in obtaining its solution is described. The concept of utility function is introduced. The optimization framework described here plays a crucial role in devising congestion control algorithms that are described in the next lesson.

2.1 Bandwidth sharing in a network

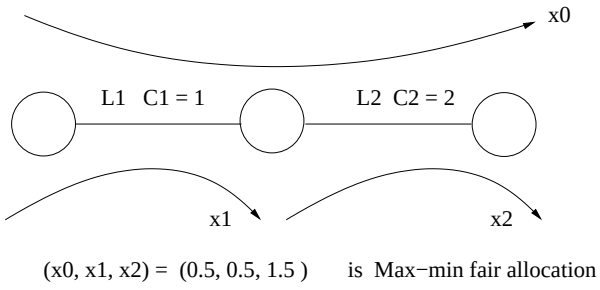

Figure 2.1: Resource allocation: Max-min fairness

Figure 2.1 shows the scenario of three sources sharing a linear network of two links, L1 and L2 of capacities C1 = 1 and C2 = 2. Let x_1 be the rate of the source that uses only L1, x_2 be the rate of the source that uses only L2 and x_0 the rate of the source that uses both the links for its flow.

2.1.1 Fair sharing is not equal sharing always

If *equal sharing* is the interpretation of fairness, then all the sources in Figure 2.1 get the same rate of 0.5 because with this rate the link capacity of L1 is saturated. This allocation of rates leaves the link L2 half-empty. However, without any source being worse off, this idle capacity can be allocated to x_2 to make its rate 1.5. As this will ensure that a given volume of data is transferred sooner than in the case of equal sharing, this allocation is better for the network.

A basic requirement that any efficient resource sharing principle has to satisfy is called *Pareto efficiency*. Note that the equal allocation scheme in the above network does not satisfy this requirement but the latter scheme does.

Definition 2.1 *Pareto Efficiency. An allocation of resources in a system is Pareto efficient if no other allocation can make some user better off and simultaneously none of the other users worse off. In other words, an allocation of resources in a system is Pareto efficient, if any one's allocation cannot be improved without making someone else's allocation smaller.*

2.1.2 Fairness vs Efficiency

An allocation can be regarded as a vector $\mathbf{x}$ whose components $x_0, x_1, \ldots, x_n$ are the rates allocated to various flows in the network. Consider an allocation of $\mathbf{x} = (x_0, x_1, x_2) := (0.5, 0.5, 1.5)$ for the system given in Figure 2.1. The total amount of user traffic carried by the network is $\sum x_i = 0.5 + 1.5 + 0.5 = 2.5$. Consider another allocation $\mathbf{x}' = (x_0, x_1, x_2) := (0, 1, 2)$ which neglects x_0. Then the total traffic carried by the system is 3, which is 0.5 greater than in the first case. However, this improved throughput is achieved by setting x_0 to zero which may be unacceptable. Hence there is a trade-off between fairness and aggregate throughput. Here for each unit of traffic for the flow x_0, we have to accept a reduction of a unit of traffic from both x_1 and x_2. So some level of balance is needed between the user rates that combines fairness with efficiency.

2.2 Different kinds of Fairness

We next illustrate resource allocation in the example network given in Figure 2.1 under the three common fairness criteria namely, Max-min fairness, Min-potential delay fairness and Proportional fairness. These criteria themselves are explained subsequently. To this end, we describe an optimization framework which allocates the rates of the sources according to the given fairness criterion which is captured by the notion of utility function.

The allocation $\mathbf{x} = (x_0, x_1, x_2) := (0.5, 0.5, 1.5)$ is an example of *max-min fair* allocation. An allocation is *max-min fare* if it attempts to maximize the minimum rates allocated in the network.

If we consider the allocation $\mathbf{x} = (x_0, x_1, x_2) := (0.25, 0.75, 1.75)$, and ask the question if it is a fair allocation, it could still be 'yes', if it happens that x_0 does not require allocation beyond 0.25.

So, it is desirable to make allocations to a source by taking into account its perceived worth for the source - this is captured by the notion of *utility* of a given rate for the source. Utility is a term used in economics to capture an individual's valuation of the worth of goods that he/she would like to purchase using his/her own money.

2.3 Resource Allocation - an optimization framework

Let $\mathcal{U}_r(x_r)$ be the utility of rate x_r to user r. $\mathcal{U}_r(x_r)$ is assumed to be a smooth (continuous differentiable) concave function. It captures the notion of law of diminishing returns in economics - a certain addition of resources to what one already has increases the total worth, but it contributes less and less to the increase if one has more of the resource already. The logarithm function is an example of this behaviour.

The goal of resource allocation in the network is to maximize the the aggregate utility of all the users in the network subject to the capacity constraints of the links in the network. We assume that all the links have fixed capacity.

So the network resource allocation problem can be formulated as follows.

Allocate resources subject to

$$\text{Max} \quad \sum_{\{x_r\} \in \mathcal{S}} \mathcal{U}_r(x_r)$$

$$\text{Subject to} \quad \sum_{r:l \in r} x_r \leq c_l \quad l \in \mathcal{L}$$

$$x_r \geq 0, \, r \in \mathcal{S}$$

Here $\mathcal{U}_r(x_r)$ is the utility we associate with the rate x_r of user r. $\mathcal{S}$ is the set of users in the network. $\mathcal{L}$ denotes the set of all links in the network. A user r is often identified with its flow rate x_r and also its route given by the links used by its flow. so the first set of constraints says that the aggregate rate of all the flows through any link $l \in \mathcal{L}$ does not exceed the capacity of the link c_l. The second set of constraints says that the rate x_r of any flow is non-negative.

As we seek to maximize a concave objective function subject to linear constraints, this resource allocation problem amounts to maximizing a *concave* function over a convex set. From optimization theory, this is a convex optimization problem and maximizing a *concave* function (or equally minimizing a *convex* function) over a convex set has a unique solution. So the given problem has a unique solution.

2.3.1 Proportional Fair resource allocation

Proportional Fairness:

The utility function $\mathcal{U}_r(x_r) = log\, x_r$ captures resource allocation according to the criterion of proportional fairness. By assigning a weight w_r to this function, we get weighted proportional fairness given by $\mathcal{U}_r(x_r) = w_r \, log\, x_r$.

Consider resource allocation in a network using the proportional fairness criterion. Let $\{\hat{x_r}\}$ be the allocation vector according to proportional fairness. Then for any other allocation vector $\{x_r\}$, we obtain the inequality

$$\sum_r \frac{x_r - \hat{x_r}}{\hat{x_r}} \leq 0$$

So this condition states that if we deviate from the optimal rates $\{\hat{x_r}\}$ of proportional fairness to some other feasible allocation $\{x_r\}$, then the sum of the proportional changes in each user's rates is less than or equal to 0.

Similarly, if the allocation is made according to weighted proportional fairness the corresponding inequality becomes

$$\sum_r w_r \frac{x_r - \hat{x_r}}{\hat{x_r}} \leq 0$$

Hence the name (weighed) proportional fairness for resource allocation using the utility function $\mathcal{U}_r(x_r) = w_r \, log\, x_r$.

A formal justification of the above inequality is a simple consequence of the behaviour of convex functions.

The optimization problem for proportional fair resource allocation for our example network becomes

$$\begin{aligned} \text{Max} \quad & log\, x_0 + log\, x_1 + log\, x_2 \\ \text{Subject to} \quad & x_0 + x_1 \leq 1 \\ & x_0 + x_2 \leq 2 \\ & x_0, x_1, x_2 \geq 0 \end{aligned}$$

To solve this optimization problem, we use the standard technique based on Lagrange multipliers. The idea behind this approach can be easily understood from this example. Here we introduce a variable called *Lagrange multiplier* corresponding to each of the inequality constraints. In our example, it is easy to see that both the capacity constraints can be satisfied with equality. Furthermore $log\, x \to -\infty$ as $x \downarrow 0$ implies that the optimal solution will allocate non-zero rate to all users. So corresponding to the optimal solution, the variables x_0, x_1 and x_2 are strictly greater than 0. The theory of convex optimization tells us that the *complementary slackness* condition has to be satisfied by the optimal solution. This requires that the Lagrange multipliers can be positive precisely for those constraints that are satisfied under equality and they are zero otherwise. So for our example problem, only the Lagrange multipliers λ_1 and λ_2 corresponding to the capacity constraints of the links L1 and L2 can be non-zero (indeed positive)

The Lagrangian for our problem is given by

$$L(\mathbf{x}, \boldsymbol{\lambda}) = log\, x_0 + log\, x_1 + log\, x_2 + \lambda_1(1 - x_0 - x_1) + \lambda_2(2 - x_0 - x_2)$$

where
$\mathbf{x} = (x_0, x_1, x_2)$ is the rate allocation vector and
$\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ is vector of (positive reals) Lagrange multipliers

The optimal point corresponds to the stationary point of the Lagrangian function. So by setting $\frac{\partial L}{\partial x_r} = 0 \ \forall r$ we get

$$\frac{1}{x_0} = \lambda_1 + \lambda_2, \quad \frac{1}{x_1} = \lambda_1, \quad \frac{1}{x_2} = \lambda_2$$

So we obtain the allocation vector $\mathbf{x}$ as a function of the Lagrange multipliers.

$$x_0 = \frac{1}{\lambda_1 + \lambda_1}, \quad x_1 = \frac{1}{\lambda_1}, \quad x_2 = \frac{1}{\lambda_2}$$

We can see here that x_1 depends only on λ_1, x_2 depends only on λ_2 and x_0 depends both λ_1 and λ_2. So the optimum rate for a source depends only on the Lagrange multipliers of its route. We can interpret the Lagrange multiplier of a link as the *congestion price* of the link which is positive precisely when the link is fully utilized and zero when it is underutilized. This indicates that the Lagrange multiplier is a natural measure of the congestion experienced by a link.

There is a splendid isolation principle that we see here. The flow rate of a source depends only on the congestion of the links in its own route and not on the congestion of links that are not on its path.

So using the interpretation of the Lagrange multiplier of a link as the congestion price of the link, we can state that for our linear network example the flow rates of the sources under proportional fairness are inversely proportional to the *sum* of the congestion prices of their respective routes.

The interpretation of Lagrange multipliers as congestion prices of links and the flow rates of the various sources as being inversely proportional to the sum of the congestion prices of their respective routes is a crucial insight that we make use of later in describing the Kelly model of congestion control.

Using the constraints $x_0 + x_1 = 1$ and $x_0 + x_2 = 2$, we get

$$\lambda_1 = \sqrt{3} \quad \text{and} \quad \lambda_2 = \frac{\sqrt{3}}{\sqrt{3}+1}$$

Thus the optimum values for the user rates are

$$x_0 = \frac{\sqrt{3}+1}{3+2\sqrt{3}}, \quad x_1 = \frac{1}{\sqrt{3}}, \quad x_2 = \frac{\sqrt{3}+1}{\sqrt{3}}$$

2.3.2 Resource Allocation under Minimum Potential Delay Fairness

We formulate the minimum potential delay fairness criterion by specifying the corresponding utility function and the rest of the analysis for optimal resource allocation under this fairness criterion follows the same pattern as before.

Consider the case where a user r is trying to send a file of size w_r. Then $\frac{w_r}{x_r}$ gives the time taken to transfer a file when the rate allocated for user r is x_r. In this case we can formulate the utility function as $\mathcal{U}_r(x_r) = -\frac{w_r}{x_r}$

So if minimizing the total transfer time of all the sources in the network is the objective, then the network seeks to maximize the above utility function (Here we have taken w_r to be 1).

$$\text{Max} \quad \sum_{r \in \mathcal{S}} \left(-\frac{1}{x_r} \right)$$

The Lagrangian for this problem is given by

$$L(x, \lambda) = -\frac{1}{x_0} - \frac{1}{x_1} - \frac{1}{x_2} + \lambda_1(1 - x_0 - x_1) + \lambda_2(2 - x_0 - x_2)$$

Setting $\frac{\partial L}{\partial x_r} = 0 \ \forall r$, we get

$$\frac{1}{x_0^2} = \lambda_1 + \lambda_2$$

$$\frac{1}{x_1^2} = \lambda_1$$

$$\frac{1}{x_2^2} = \lambda_2$$

Using the equality link constraints corresponding to optimal allocation rates

$$x_0 + x_1 = 1$$

$$x_0 + x_2 = 2$$

We can solve for x_0, x_1, x_2 in terms of λ_1 and λ_2 to get

$$x_0 = \sqrt{\frac{1}{\lambda_1 + \lambda_2}}, \quad x_1 = \sqrt{\frac{1}{\lambda_1}}, \quad x_2 = \sqrt{\frac{1}{\lambda_2}}$$

Now by comparing the allocation vector under minimum potential delay fairness with that under proportional fairness we see that for our example network, the different rates here depend on the Lagrange multipliers of their respective routes as before but note the new square root factor involved.

2.3.3 Resource Allocation under Max-min fairness

The goal of *max-min* allocation is to maximize the minimum allocation. So max-min fairness scheme attempts to give maximum protection to the user who gets the least amount of resources. So for an allocation that has been made under max-min fairness, the allocation of anyone cannot be increased further without reducing the allocation of someone who has a smaller allocation. So max-min fairness can be regarded as giving maximum protection to the *weak* and in that sense it is regarded as an egalitarian notion of fairness (staunch socialist). Now we can give a formal definition of max-min fair allocation.

Definition 2.2 *Max-min fair allocation: A vector of rates x_r is max-min fair if for any set of rates y_r that satisfies the capacity constraints the following is true:*

if $y_s > x_s$ for some $s \in S$ then there exists a $p \in S$ such that $x_p \leq x_s$ and $y_p < x_p$

Yet another way of describing max-min fair allocation is that no one's allocation can be further increased without making someone who is already poor (compared to him) poorer.

A link l is a *bottleneck* link for a source r if the link is fully utilized and r has the largest flow rate among all sources using link l.

Definition 2.3 *Bottleneck-link for a source: A link l is a bottleneck link for source r if it has the following properties.*

$$\sum_{s \in S_l} x_s = c_l \quad \textit{and} \quad x_r \geq x_s \quad \forall s \quad \textit{such that} \quad x_s \in S_l$$

Here S_l identifies all the flows using the link l.

There is a lemma which relates max-min fairness and a bottleneck link of a source. It states that a set of rates x_r is max-min fair if and only if every source r has a bottleneck link.

2.4 α fairness and general class of utility functions

We can define $U_r(x_r)$ as a general class of utility functions that captures different fairness criteria such as proportional fairness, minimum potential delay fairness and max-min fairness. It also captures many other fairness criteria that lie between them a suitable choice of the parameter α which takes values in the interval $(0, \infty)$.

$$U_r(x_r) = w_r \frac{x_r^{1-\alpha}}{1-\alpha} \quad \alpha > 0, \quad \alpha \neq 1, \tag{2.1}$$

These utility functions are called α-fair utility functions and the rate allocations are called α-fair allocations as different values of α_r yield different fairness criteria.

Case 1: Weighted proportional fairness $(\alpha \to 1)$

Weighted proportional fairness can be captured by the utility function in the limiting case when $\alpha \to 1$. Maximizing the sum of $\frac{x_r^{1-\alpha}}{1-\alpha}$ gives the same optimum value as maximizing the sum of $\frac{x_r^{1-\alpha}-1}{1-\alpha}$

Applying L'Hôpital's rule we get

$$\lim_{\alpha \to 1} \frac{x_r^{1-\alpha} - 1}{1-\alpha} = log\, x_r$$

$$U_r(x_r) = w_r\, log\, x_r$$

Case 2: Minimum delay potential fairness

The notion of minimum delay potential fairness corresponds to $\alpha = 2$ in the general utility function.

$$U_r(x_r) = \frac{w_r}{x_r}$$

The resource allocation using this utility function is weighted minimum potential delay fair.

Case 3: Max-min fairness

Max-min fairness corresponds to the case where $\alpha \to \infty$

So in conclusion we may state that the α-fairness is captured by the utility functions given by

$$U_r(x_r) = \begin{cases} w_r \frac{x_r^{1-\alpha}}{1-\alpha} & \alpha > 0,\ \alpha \neq 1 \\ w_r\, log\, x_r & \alpha = 1 \end{cases} \tag{2.2}$$

Note: Another aspect of fairness is that it facilitates 'fluency of sharing'. A strong/rigid notion of fairness as in max-min fairness (MMF) can result in inefficient utilitization of resources. Weighted proportional fairness (WPF) is considerate to the weak in the spirit of MMF, but is not as rigid. So WPF is a 'mellowed' form of MMF. Note that both WPF and MMF are special cases of α-fair utilities corresponding to $\alpha \to 1$ and $\alpha \to \infty$ respectively. And so, intuitively fairness is regarded as improving with increasing α.

Our notes has used material from [S04, SS07].

2.4.1 MATLAB Experiments using the Convex optimization package CVX

Example 1- Proportional fairness

```
% Convex optimization: Example 1- Proportional fairness
clc;
clear all;
close all;

cvx_begin
variables x y z
maximize (log(x)+log(y)+log(z))
subject to
x >= 0; y >= 0; z >= 0;
x+y <= 2; x+z <= 1;

cvx_end
x
y
z
CVX Warning:
   Models involving "log" or other functions in the log, exp, and entropy
   family are solved using an experimental successive approximation method.
   The method requires multiple calls to the solver, so it can be slow; and
   in certain cases it fails to converge. See Appendix D of the the user's
   guide for more information about this method, and for instructions on how
   to suppress this warning in the future.

Successive approximation method to be employed.
   SDPT3 will be called several times to refine the solution.
   Original size: 14 variables, 8 equality constraints
   3 exponentials add 24 variables, 15 equality constraints
-----------------------------------------------------------------
 Cones  |             Errors              |
Mov/Act | Centering  Exp cone   Poly cone | Status
--------+---------------------------------+---------
  3/  3 | 7.987e-01  5.042e-02  0.000e+00 | Solved
  3/  3 | 8.352e-02  5.211e-04  0.000e+00 | Solved
  2/  3 | 7.044e-03  3.687e-06  0.000e+00 | Solved
  2/  2 | 6.067e-04  2.695e-08  0.000e+00 | Solved
  0/  0 | 4.970e-05  0.000e+00  0.000e+00 | Solved
-----------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): -0.954771


x =

    0.4226


y =

    1.5774
```

```
z =

    0.5774

>>
```

Example 1- Minimum Potential Delay fairness

```
% Convex optimization: Example 1- Minimum potential delay fairness
clc;
clear all;
close all;
cvx_begin
variables x y z


maximize -(inv_pos(x) + inv_pos(y) + inv_pos(z))
subject to
x >= 0; y >= 0; z >= 0;
x+y <= 2; x+z <= 1;

cvx_end
x
y
z

Calling SDPT3: 14 variables, 8 equality constraints
------------------------------------------------------------

 num. of constraints =  8
 dim. of sdp    var  =  6,   num. of sdp  blk  =  3
 dim. of linear var  =  5
*****************************************************************
   SDPT3: Infeasible path-following algorithms
*****************************************************************
 version  predcorr  gam  expon  scale_data
   HKM      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|1.1e+01|1.2e+01|1.1e+03| 3.000000e+01  0.000000e+00| 0:0:00| chol  1  1
 1|0.942|1.000|6.1e-01|1.0e-01|9.7e+01| 2.495101e+01 -2.960945e+01| 0:0:00| chol  1  1
 2|0.964|1.000|2.2e-02|1.0e-02|1.1e+01| 6.279254e+00 -4.000009e+00| 0:0:00| chol  1  1
 3|1.000|0.952|6.1e-08|5.9e-03|1.8e+00| 5.272820e+00  3.532245e+00| 0:0:00| chol  1  1
 4|1.000|1.000|2.9e-08|1.0e-04|3.5e-01| 4.800235e+00  4.455093e+00| 0:0:00| chol  1  1
 5|0.967|0.967|4.8e-09|1.3e-05|1.2e-02| 4.668588e+00  4.656890e+00| 0:0:00| chol  1  1
 6|0.963|0.973|6.9e-09|1.3e-06|3.7e-04| 4.663814e+00  4.663455e+00| 0:0:00| chol  1  1
 7|0.961|0.964|3.4e-09|4.9e-08|1.4e-05| 4.663645e+00  4.663631e+00| 0:0:00| chol  1  1
 8|0.951|0.938|3.8e-10|3.7e-09|7.9e-07| 4.663638e+00  4.663638e+00| 0:0:00| chol  2  1
 9|1.000|1.000|2.9e-11|7.6e-11|1.1e-07| 4.663638e+00  4.663638e+00| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   =  9
```

```
 primal objective value =  4.66363813e+00
 dual   objective value =  4.66363802e+00
 gap := trace(XZ)       = 1.12e-07
 relative gap           = 1.09e-08
 actual relative gap    = 1.08e-08
 rel. primal infeas     = 2.89e-11
 rel. dual   infeas     = 7.64e-11
 norm(X), norm(y), norm(Z) = 4.5e+00, 7.0e+00, 8.2e+00
 norm(A), norm(b), norm(C) = 4.7e+00, 3.8e+00, 2.7e+00
 Total CPU time (secs)  = 0.32
 CPU time per iteration = 0.04
 termination code       =  0
 DIMACS: 3.7e-11  0.0e+00  1.0e-10  0.0e+00  1.1e-08  1.1e-08
------------------------------------------------------------------
------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): -4.66364

x =

    0.4864

y =

    1.5136

z =

    0.5136
>>

P.S. Error using ==> cvx.times at 173
Disciplined convex programming error:
    Cannot perform the operation: {positive constant} ./ {real affine}

Error in ==> cvx.rdivide at 19
z = times( x, y, './' );

Error in ==>
                D = 1./(c-S);

The expression 1./(c-S) is neither convex nor concave. CVX is meant solely for convex problems.
Now, if you intend for c-S to be strictly positive, then you could perhaps use inv_pos(c-S),
which is a convex expression.
```

References

[S04] R. Srikant, "The Mathematics of Internet Congestion Control". *Birkhauser*, 2004.

[SS07] S. Shakkottai and R. Srikant, "Network Optimization and Control". *Now publishers*, 2007 (available online).