Coursework 2

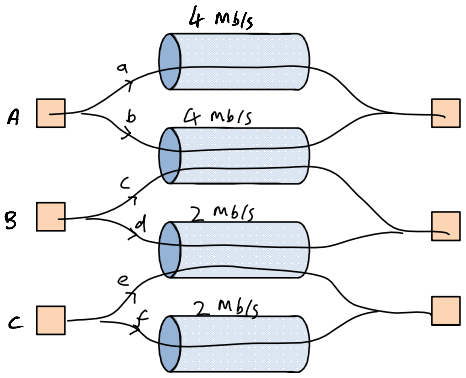
Simulation, Job models

Network Performance—DJW—2009/10

This coursework is worth 3.5% of your final grade. You should hand it in to Computer Science reception by noon on the due date. You may use any computer language you wish.

The networks group at UCL is developing a multipath version of TCP. The idea is that if there are several paths between a source and a destination, then TCP ought to be able to send traffic on each of the paths simultaneously; this should ensure that all the network's capacity is fully used, and it should also make flows more resilient to traffic surges or link failures that affect some but not all of their paths.

In this coursework your task is to write a simulator of multipath TCP, along the lines of the Processor Sharing simulator described in lecture 9, and to demonstrate that you can use your simulator to make sensible measurements. In Coursework 4 you will use your simulator to investigate the behaviour of multipath TCP under different scenarios, and compare it to theoretical predictions.



Consider the simple network shown above, consisting of four links, three source-destination pairs, and six paths. New flows arise at each of the sources, and the interarrival times for source i are Exp(λ_i) random variables, so mean interarrival time is $1/\lambda_i$ seconds. Flow sizes are random; they are all Exp(1) Mb. We wish to compare three different multipath algorithms:

(i) Under the `unipath` algorithm, each flow picks one of its possible paths at random and stays with that path, each path being equally likely; the capacity of each link is divided equally between all flows that are using that link. For example, if there are currently 2 flows using path a, 3 flows using path b, 0 flows using path c, 4 flows using path d, 1 flow using path e and 2 flows using path f, and a new flow arrives at source C, then with probability 50% it will choose path e and get throughput 2/6 Mb/s, and with probability 50% it will choose path f and get throughput 2/3 Mb/s.

(ii) Under the `uncoupled` algorithm, each flow uses both of its possible paths simultaneously; the capacity of a link is divided equally between all flows that are using that link. For example, if there are currently 5 flows at source A, 4 flows at source B and 4 flows at source C, then each of the flows from source C is getting throughput 2/8 Mb/s on path e and 2/4 Mb/s on path f.

(iii) The `coupled` algorithm, like the `uncoupled` algorithm, lets each flow use all its possible paths simultaneously; however it shares capacity differently. There is no explicit formula for how it shares capacity. Instead, here is a Python function which approximates the throughput that results on each of the six paths:

```
def thr_coupled(c,n):
```

```
    c1,c2,c3,c4 = [float(cc) for cc in c]
    nA,nB,nC = n
    za,zb,zc,zd,ze,zf = c1,c2/2,c2/2,c3/2,c3/2,c4
    for i in range(10):
        if nA+nB>0:
            zb = trunc( (nA*c2+nA*zd-nB*c1)/(nA+nB), (0,c2) )
            zc = c2-zb
        if nB+nC>0:
            ze = trunc( (nC*c3+nC*zc-nB*c4)/(nB+nC), (0,c3) )
            zd = c3-ze
    mA,mB,mC = [(float('Inf')) if nn==0 else nn) for nn in n]
    return (za/mA, zb/mA,  zc/mB, zd/mB,  ze/mC, zf/mC)
def trunc(x,(min_,max_)): return min(max(x,min_),max_)
```

For example, if there are currently 5 flows at source A, 4 flows at source B and 4 flows at source C, then each of the flows at source C is getting throughput 0.423 Mb/s on path e and 0.5 Mb/s on path f.

Program a simulator of this system. Your written report should include the source code. It should also include a brief description of how your simulator works—imagine you are giving instructions to a colleague who knows how to program but who knows nothing about simulation or about networks.

Your report should also include verification that the program works. For example, you might work through an example by hand, with pen and paper, and check that your program gives the same answer. Make sure that you verify every part of your code.

We wish to understand how flow completion time is affected by choice of algorithm and by the arrival rates. Specifically, for this coursework, we will investigate arrival rates $(\lambda_A, \lambda_B, \lambda_C) = (0.3x, 0.3x, 0.4x)$ for x in the range 5–15.

Plot a graph showing the average flow completion time for flows from source A, as a function of x. Your graph should have three lines, one for each of the three algorithms. Run your simulator at least 10 times for each value of x, and use this to calculate error bars for your plot. Repeat for flows from source B, and for flows from source C.

When you run simulations, there are two issues to bear in mind. (i) If you start your simulator from some atypical initial state, then there is some 'run-in' time until it produces useful output. For example, if you start it with no active flows from any of the sources, then the first few flows are likely to complete more quickly than average. Therefore you should run your simulator for a while and discard the first few readings. (ii) If you simulate an overloaded system, for example if x is very very high, then flows arrive more quickly than they can be served, so the number of active flows will build up, so the flow completion time will increase steadily over the course of your simulation. Such a system is called 'unstable'. If the system is unstable then it does not make any sense to talk about 'average flow completion time'. Therefore you should only plot average completion time in situations where the system is stable, and you should find at what value of x it becomes unstable.

Explain how you have dealt with the problems of run-in and instability.

Further reading:
`http://www.cs.ucl.ac.uk/staff/d.wischik/Research/respool.html`,
`http://research.microsoft.com/apps/pubs/default.aspx?id=67961`