An Illustrative Discussion of Different Perspectives in Network Engineering

V.Grout, S.Cunningham and R.Hebblewhite

Centre for Applied Internet Research (CAIR), University of Wales, NEWI, Wrexham, UK
e-mail: v.grout@newi.ac.uk

Abstract

This paper presents three case studies in network design and management from the different, but often overlapping and blurred, perspectives of the theorist and practitioner. It addresses the gulf between the comfortable models of graph theory, statistics and suchlike and the harsh realities of real-world networks. As, through these case studies, it becomes evident that network techniques, algorithms, etc. at large in the Internet bear little resemblance to their counterparts in the textbooks, the question is posed (and partially answered) as to whether the original theory is really any use!

Keywords

Mathematics, Network engineering, Theory and practice, Traffic shaping, Internet routing, Network design.

1. Introduction: Theory and Practice

In principle, the relationship between the mathematical theorist and the real-world practitioner is a sound one – tried and tested in fact. This happy alliance builds bridges from force diagrams, sends astronauts into space with matrix algebra and finds hitherto unknown particles using group theory (Sternberg, 1995). Usually, there are two forms the relationship may take …

In the first, the theorist, in their abstract potterings, studies an academic problem, formulates and formalises it, constructs a model and solves it. Very often they have no particular idea as to *why* they are giving it their time; the problem is just *there*, and that is enough for the theorist! At some later stage, along comes the practitioner with a particular real-world problem to overcome and stumbles across the work of the theorist. They pick up the theorist's model, realise its potential to them, apply the methods and their problem too is solved. Everyone is happy!

Alternatively, the practitioner may arrive at the problem first and *consult* the theorist. The theorist then takes the real-world problem, has a good look at it, strips it of its baggage – *abstracts* it, formulates and formalises it as before, constructs a model and solves it before handing the gift-wrapped solution back to the theorist. The theorist, newly armed, confronts the original real-world problem, which duly capitulates. The alliance works once more!

Unfortunately, however comfortable this relationship might seem, we know that it does not always work this way. Bridges fall down from time to time, space missions have been known to go wrong with catastrophic results and, occasionally, the theory predicts things that no-one can prove or find. (So where *is* all that dark matter then?!) It often turns out that neither party is exactly *wrong*; merely that they have not entirely understood each other. They see things in a different light. They have slightly different interpretations of the problem that lead to different conclusions. There is tension in the relationship.

It can hardly be avoided that this is particularly true in Internet engineering and the mathematical theory that supposedly supports it; it may even be that the problem is a lot worse. It is difficult to say why. Perhaps it is because modern networks are so *new* and mathematics (most of it anyway - comparatively) so *old*. Perhaps it is a difference in outlook; maybe the nerds and the geeks are a lot different after all? Whatever the reason, the divide is there.

This paper considers this divide through three real examples, drawn from well-known fields in network theory and engineering: traffic analysis, network design and routing. Each begins with the classical theory and contrasts this with the behaviour of the real world, sometimes with both theory and practice evolving over time. In each case, we ask whether the mathematics actually helps or hinders the network engineer. Each offers lessons to be learned. The results are interesting. We begin by looking at *traffic*.

2. Traffic Analysis and Shaping

There was once a golden age of harmony between the theorist and practitioner when the telephone was the primary means of electronic communication. Once a call was established, it laid claim to the channel that carried it and held on to it until the end of the call. No-one paid any attention to what went on between call set-up and tear-down – there was no need. The only data of interest was the frequency of the calls and, possibly, their *distribution*.

Statisticians and network engineers alike handled such traffic with the *Poisson* model, applied either to the call durations or the gaps between them. Suppose calls are independent events, in which case the arrival (say) of a call is effectively random. Then the time between two successive calls (the *inter-arrival time*) is modelled as *negative exponential* with *probability density function* (*pdf*), $f(t) = \theta e^{-\theta t}$. (This seems reasonable since the probabilities of long inter-arrival times t get exponentially *less* likely as t increases.) Then the *number of calls*, n, in a given period of time can be shown (Kingman, 1993) to have pdf, $P(n) = \theta^n e^{\theta} / n!$, which is the Poisson distribution. Very neat!

Whether the telephone traffic ever really did follow the Poisson model hardly matters now. (It generally seemed to but there was never really enough of it to be sure.) The problems started when circuit-switching began to give way to packet-switching. For many years the distribution of packets was cheerfully assumed to be Poisson, conveniently ignoring the fact that these were not really independent entities in their own right any more (a necessary requirement for Poisson) but parts of larger streams, connections and dialogues. Although multiple packet streams could now share the same channel, the existence of logical structures (albeit possibly many of them) at higher levels made true randomness unlikely.

Sure enough, as soon as anyone took the trouble to look (Paxson and Floyd, 1995), it became clear that packet (i.e. Internet) traffic was anything but Poisson. Observed data simply did not fit the equations. What was wrong? Had the mathematics failed?

No, not really. As we have already noted, the packets are not random in the way calls were (assumed to be). The inter-arrival times, therefore, are not negative exponential and the Poisson model for the number of packets in unit time no longer follows. The model is invalid so should not be used. Forget it!

However, this is hardly the end of the story. On consideration, a different, deeper question now presents itself. If Internet traffic is not Poisson then it is not random. If it is not random then it must have some *pattern*. *What* pattern? How can we describe it? How can we *define* it? How can we *quantify* or even *measure* it? Generally, in the real-world, if something has a pattern then it has a shape; but how does this apply to packets? What *shape* is Internet traffic?

Incredibly, the answer predates the Internet by fifty years or so – and arguably hundreds more – and comes from a completely unrelated field! In the early part of the twentieth century, the English hydrologist, Harold Hurst was studying the water levels of the River Nile. (He was lucky really: he had over 800 years' worth of Egyptian records to work with and there are few areas in which any sort of data goes back that far.) He noticed that the levels, although fluctuating apparently randomly over short periods, appeared to follow cycles of low and high over longer ones. Looking closer, he could see cycles within cycles, and so on. (His aim was to build a reservoir 'just big enough' that would never run dry.)

Hurst called the phenomenon *Long Range Dependence* (*LRD*) and introduced a parameter, *K*, to define it (Hurst, 1951). *K* is effectively a correlation coefficient - but a measure of the data's correlation with *itself* over different time frames. Consequently, a value of $K = ½$, indicates absolutely no LRD, i.e. random traffic, whereas $K = 1$ indicates total LRD. LRD is closely linked to the mathematical concept of *self-similarity*. In later years, Mandelbrot, in his work on the fluctuations in stock markets, Mandelbrot and Hudson (2005) called the phenomenon *fractal* and, on discovering that Hurst had effectively beaten him to it by several decades, renamed *K* as *H* and called it the *Hurst Parameter* in recognition (and embarrassed admiration no doubt).

The actual definition of LRD is based on the *autocorrelation function* (*ACF*) of a data stream. For any given *lag k* (the window over which similarity is considered), the ACF is given as a function of the mean and variance (Rezaul et al., 2006a). Now, adding all these ACFs for $-\infty \leq k \leq \infty$, gives a working definition of LRD, namely that this sum is finite for non-LRD data but infinite for LRD data.

Very interesting to be sure, but what has this to do with Internet traffic? Well, quite simply, in 1993, LRD was discovered on an Ethernet LAN (Leyland et al., 1994)! There was some scepticism at the time but in the decade that followed, the evidence grew to unavoidable proportions. LRD traffic has now been shown to be present (sometimes) in all parts of the Internet, often characterised simply as *bursty*. (So what *shape* is Internet traffic? *Fractal*!) Worse still, it was then demonstrated (Park et al., 1997) that LRD traffic *caused problems* within a network, domain or internet by making TCP buffers more prone to overflow, etc. LRD is not necessarily *always* present but, when it is, it does bad things.

Fortunately, provided we can detect LRD, there are simple algorithms to either lower it (by reordering packets, etc.) (Rezaul and Grout, 2007a) or, through traffic shaping, reduce the burstiness (Rezaul and Grout, 2007b) (which can be shown to be much the same thing). All that is needed is the Hurst Parameter, H, of the traffic in question. Ah!

Now, the problem with this is that the Hurst Parameter was never really defined in this way! The nature of the formulae concerned (Hurst, 1951),(Rezaul et al., 2006a) works the other way around. That is, given a value of (say) $H = ¾$, it is fairly simple to construct a data series with the right level of LRD but, for a given data series, there is no exact formula for H; the best we can do is *approximate* it. No problem; Hurst himself had a method for this; it was called *Rescaled-Range* (*R/S*) and it worked nicely for Nile river levels (and Mandelbrot's stock markets too for that matter).

So it should be a simple enough process to reproduce the calculations for our packet streams – and it is. Take some typical Internet traffic (ITA, 2007), apply R/S to it and out comes the estimate of H. The only problem is that we often get values of $H > 1$, which is clearly impossible, or $H < ½$ for known LRD traffic. True, there are estimates of LRD other than R/S available (see (Rezaul et al., 2006a) for a comparison) but none fare any better for Internet traffic. (This can also be demonstrated by circular means: chose a value of H, generate an appropriate packet sequence, then try to estimate H using R/S, etc. The result can be way off.)

We are left with the simple conclusion that R/S and the other estimates are not particularly suited for calculating the LRD of Internet traffic. Fortunately, in 2005, a new estimator was developed (Rezaul et al., 2006b) that produced much better results. This new approach used a particular variant of the ACF of the packet sequence data, with *lag 2*, to estimate H, and appeared to make a much better job of it. The extreme values disappear and the traffic shaping algorithms become properly effective.

So where does that leave us? Have we simply been waiting all this time for a better estimator? No, we have been waiting for the *right* estimator. Internet traffic has its own characteristics, beyond LRD, that are not necessarily the same as we find in rivers and stock markets. The new estimator works nicely for Internet traffic smoothing; there is no guarantee it will be as good for building reservoirs or investment portfolios!

Lesson One: Find the right tool for the job!

3. Spanning Trees and Network Design

Time for a change of scene. Let us take a step back from the intricacies of traffic measurement and imagine that the network has not even been built yet! Our second case study concerns possibly the simplest - as in the easiest to state – *network optimisation problem* there is. Given a number of locations, what is the best way to connect them together? Well, according to the theorist and their textbook, it goes like this ...

To optimise anything, we need an *objective function*. In this case, cost seems the likely candidate. Fig. 1 lays out the model: n nodes with an *$n \times n$ cost matrix $C = (c_{ij})$* where c_{ij} is the cost of connecting node i to node j ($c_{ij} = \infty$ if there is no link). We could insist that costs be *Euclidean* (i.e. proportional to the straight-line distance between two points) or that the cost matrix be *symmetric* ($c_{ij} = c_{ji}$ for all i,j) but neither is essential. Non-Euclidean costs allow local factors to be considered whilst an asymmetric cost matrix permits unbalanced loads so our model, at first sight, looks flexible enough to keep everyone happy. And in fact it gets better because our objective would now appear to be to find a connecting network that minimises total cost and some very old algorithms (Kruskal, 1956),(Prim, 1957) will do this very efficiently. The result will be a minimal spanning tree solution something like Fig. 2.

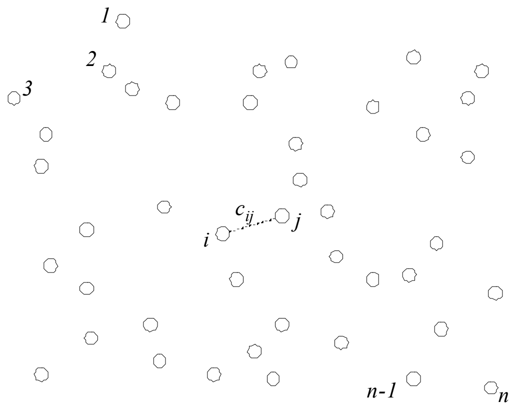

Figure 1: Nodes and costs

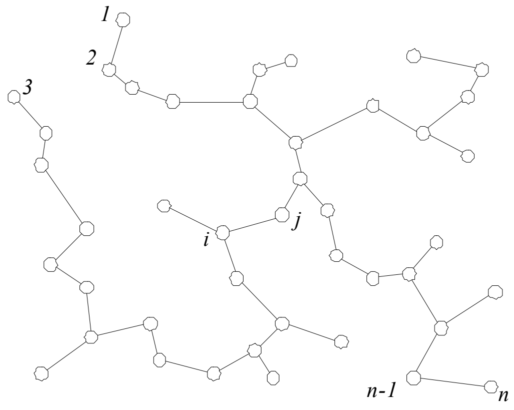

Figure 2: A 'Minimal Spanning Tree' (MST) solution

Unfortunately, give this solution back to the practitioner and they will laugh (or worse), and it is hardly difficult to see why. This is not a 'real' network. The MST solution is entirely impractical for a number of reasons – far too many to deal with one-by-one – but two in particular scream out. Firstly, the network is hopelessly vulnerable. A single failure, of node *i* say or of the link *(i,j)* is catastrophic. There is no redundancy, no alternate routes, so one half of the network is cut off from the other. Secondly, some of the paths between nodes are ridiculously long, particularly between apparently near neighbours. Subscribers at nodes 2 and 3 for example are hardly likely to be impressed that data between them passes half way round the network.

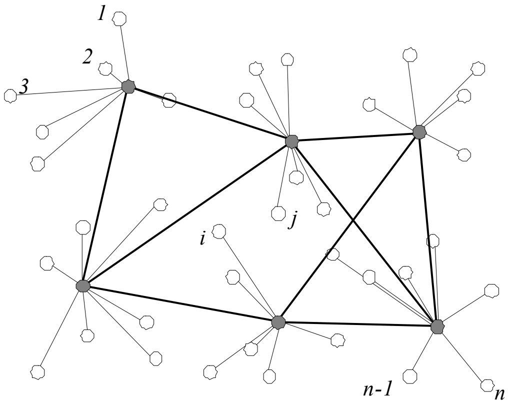

Figure 3: A more robust network

OK, so the MST solution is not such a good one after all. The practitioner is more likely to be impressed with a network something along the lines of Fig. 3. This is a good compromise. A more strongly-connected core network carries most of the traffic. Although peripheral nodes may still lose contact through failure, the rest of the network is unaffected. If a core node or link fails there are alternate paths; and no paths are particularly long.

No need for the theorist to give up yet though. It is still no bad thing to be minimising cost but the solution must be *constrained*. We can seek to find an optimal set of core nodes, whilst constraining the structured solution. We can insist that each peripheral node connects to its nearest core node. We might constrain the path length between core (and hence peripheral) nodes or insist upon a minimum number of (node or link) independent paths between nodes but this, in practice, proves to be difficult (Garey and Johnson, 1979). Better in fact to specify a minimum *degree* (number of connected neighbours) for each core node. In fact, if necessary, peripheral nodes can even connect to two or more core nodes for robustness. All these things can be built into the constraints. The old simple MST algorithms no longer work of course. (In fact, it is now difficult to find perfectly optimal solutions at all for larger problems.) However, a combination of *greedy* algorithms and *local searches* will usually see us home (or somewhere close anyway). *Heuristics* may have replaced *exact* methods but we seem to still be doing a decent, if imperfect, job.

Unfortunately the final blow comes when we take a closer look at these costs – these costs that we have cheerfully assigned statically to each potential link. Where do these costs come from? How do we determine the cost of connecting *i* to *j*? In fact, in the real world, the length of the link, Euclidean or otherwise, has very little to do, in practice, with its cost. Much more relevant is its *capacity* - the amount of traffic it can carry. A link must be able to carry not just the traffic between its own end-points but all traffic routed over that link. Fig. 4, for example, shows three different connection configurations for the same node set. In 4(a) *(p,q)* is a core link and carries traffic between several node pairs. The 'same' peripheral link in 4(b) will cost less and in 4(c) there is no link at all.

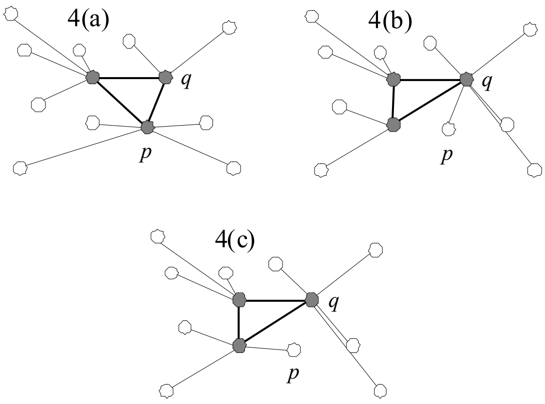

Figure 4: Costs depend upon topology

The conclusion that follows is an uncomfortable one. To determine the cost of a link we need to know its capacity. But this can only be calculated if we know how much traffic it has to carry, which implies a knowledge of the topology (shape) of the network, which in turn is the solution we seek! We might well have a *traffic matrix* $T = (t_{ij})$ (where t_{ij} is the traffic originating at *i*, destined for *j*) - in fact it would be hard to design a network without it - but it hardly helps. *T* is a requirement, not a plan: we still have no idea where the traffic goes in the network or how much traffic each link has to carry unless we know the topology. To paraphrase: to specify the cost matrix *C*, requires a knowledge of the solution network - our input needs the output. Hardly a well-formed optimisation problem!

So, with our theoretical model in ruins, what do we do? In truth it is usually left to the practitioner. Something has to work somehow. The following is typical … If all we can do is determine the cost of a *known* network then let us start from there. Put *all* nodes in the core network and directly connect every pair. Knowing (or maybe even guessing) the traffic *T*, it is easy to calculate (or approximate) the traffic on each link (because all traffic is carried directly), its cost and therefore the cost of the complete solution. Of course, it will be a very expensive solution but that makes it easy to find a better one. Consider, in turn, dropping each node from the core network, decide where the affected traffic goes (it will be re-routed via other nodes/links, possibly making good use of spare capacity), recalculate the cost, and choose the drop that maximises the saving. At an appropriate point, and subject to the constraints, consider dropping links from the core network as well. Stop when there do not appear to be any further drops that will improve the cost.

And this ‘double-drop’ algorithm actually *works*. At least, that is, it finds a viable solution. It will not be optimal of course. It will be some local cost minimum, possibly considerably above the true optimum cost, which may well come from a completely different topology, and the worst of it is that we have no way of knowing exactly *how* bad it is. But it works nonetheless. We are hardly in a position to criticise the practitioner for such a clumsy effort - something is better than nothing. MSTs look great on paper but don’t translate well into the real world (although see section 5). The textbook has failed … or, at least, we have been looking at the wrong textbook!

Lesson Two: Find the right model for the situation!

4. End-to-Path Costs and Routing

Our final example considers Internet *routing*. Here in contrast to the previous sections, we consider the case initially from the point of view of the practitioner, albeit one willing to borrow an idea or two from the theorist in the first place. (Why? Because this how it is actually done on real-world, functioning Internet routers!)

We take, as our starting point a network whose physical topology is already established - although possibly not 100% reliable. Fig. 5 is a good example with which to work, concentrating on the core rather than the peripheral network. The thick/thin, solid/dotted lines simply give a general impression that some links may be larger (have a greater capacity) or more reliable than others.

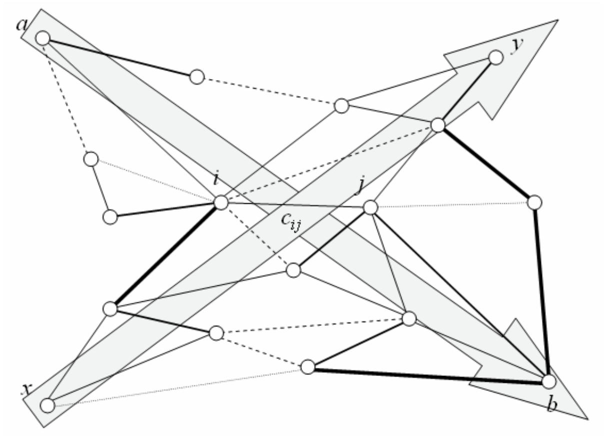

Figure 5: A routing problem

We also appear to be giving each link a *cost*! This may appear contradictory, considering the conclusions of the previous example. But this is different, surely? The network topology is now known, the links are in place: of course we know how much each one costs!

And yet, in fact, it is not as simple as that; not if we are now to consider how the traffic is to be directed, routed, over the network. Routing is a key process within the Internet - one in which little, if anything can be assumed. Even if the network has been designed with the routing implicit in the double-drop process in the first place (and it probably has not - it has more likely just ‘evolved’), traffic characteristics *change*. They change in both the short and long term (Leyland et al., 1994). Also, things go wrong: links and nodes fail – then sometimes start working again. Any routing process in a large network must be flexible, *dynamic*. In this environment, the notion of cost needs to be treated extremely carefully. The initial installation cost of each link is hardly likely to be relevant now.

What we are really after is a measure of how *efficient* a routing strategy is (so that we can try to maximise it). The cost of a link is therefore its tendency to restrict this efficiency. In a simple sense, it may be a fixed measure of the link’s capacity but it may have a dynamic interpretation - such as instantaneous load, delay or failure. Whatever the practitioner decides the essential *metric* to be, that is what the routing should seek to minimise. In fact, this opens up an entirely new line of enquiry. What exactly *should* we be trying to measure, maximise or minimise? Throughput? Delay? Customer satisfaction? Bank balance? We simply do not have the room to pursue this particular question here but it turns out to be a fascinating study: a start is made in (Grout et al., 2004).

To make any progress at all, we will have to narrow the field - take an example. *Open Shortest Path First* (*OSPF*) (Moy, 1998) uses link capacity as its default measure. In fact it calculates link cost as $c = 10^8 / b$ where b is the speed, or *bandwidth* of the link. Minimising cost means finding routes using links with higher bandwidth - which makes sense, surely?

Well, yes. No argument so far. The problem comes with the implementation. Both OSPF and the newer *Intermediate System to Intermediate System* (*IS-IS*) (Gredler and Goralski, 2004), use *Dijkstra’s Shortest Path Algorithm* or *DSPA* (Dijkstra, 1959) to find these minimum routes. Actually the practitioner is inclined to be a little smug about this: DPSA is both optimal and efficient and comes straight from the theorist’s textbook. So it looks like the job is done.

But up jumps the theorist! DSPA is optimal, yes, but it only finds the shortest (least cost) path from one node to another - *in isolation*. If there were only these two nodes in the network, with no other traffic around, it would work wonderfully - but that is not the case. *All node pairs* are (potentially) sending traffic to each other. The independent, pair-wise nature of DSPA cannot take into account how routes *compete* for use of these links.

We need to be clear what the problem is here because of course there *are* algorithms that will calculate multiple paths in a network – but that misses the point. Consider the routes from nodes a to b and x to y using (or not using) the link (i,j) in Fig. 5. Either route using the link will reasonably accrue a cost c_{ij} but if *both* use it then, surely, this cost itself has to be reconsidered? Again, using capacity as our example of a cost metric, the cost c_{ij} will be based on the ability of the link to provide a certain level of throughput for a stream of traffic. If it has to deal with two such streams then clearly this ability is reduced; we could argue - possibly simplistically - that it is halved. However we measure it, the cost of the link really should increase by some factor. In OSPF terms, it may be doubled.

Again, in case this seems too obvious, consider Fig. 5. If DSPA, working on the route from a to b, includes the link (i,j) then this adds c_{ij} to its cost. However, if another route shares the link then c_{ij} is doubled so another c_{ij} is added to the same route. But the same is also true of the second route. The contribution of the link to the cost of each route has doubled and to the network routing as a whole increased by a factor of four. The real implication is that, had we known this when we started, we might have chosen different - and better - individual routes. DSPA, working independently for each route, has let us down. And, of course, the problem increases in complexity when we consider *all* routes competing for *all* links. Individually 'optimised' routes do not produce optimal routes for the good of the network as a whole. (Consider road traffic jammed on the main trunk route; it will be better for all if a percentage of drivers take side roads – everyone will get there quicker. However, the tricky bit is to find the right percentage so that neither is disadvantaged.) Unfortunately, OSPF and IS-IS are about as sophisticated as current Internet routing protocols get. (Cisco, of course, have an alternative approach through their *Enhanced Interior Gateway Routing Protocol* (*EIGRP*) but that opens up an entirely different can of worms (Houlden et al., 2006). and sadly there is no space for that here!) There may also be a few better ones on the drawing board (Ji and Yeung, 2005) but they have yet to leave it!

Actually, as an aside, it is a complete nonsense to assume that costs are always added in this way anyway (Houlden et al., 2006). Fair enough, if we are considering something like network delay, then adding delays through individual links should give the overall delay across the network. That makes sense. But *capacity* measures the link's ability to carry traffic (rather than discard it). It effectively identifies potential *bottlenecks*. Would not a *minimum* or *maximum* be better than a *summation*? How about cost calculated as *probability* of failure? This needs a *product* surely? Obvious says the theorist? Try telling the practitioner using the existing protocols!

(Once again, at this point, it could be argued that EIGRP fulfils some of these ideals. That this is not really the case is discussed in (Houlden et al., 2006).)

Well anyway, taking a conscious decision not to turn off into such treacherous ground, we turn a blind eye to these DSPA/OSPF/IS-IS/EIGRP shortcomings and deal with the multiple route problem. How can we derive (or even approximate) the best overall routing when the interaction of these routes interferes with the very costs by which we calculate them? Now, when you put it like that, it starts to sound familiar. *Our input needs the output*. Is this not similar to the example from the previous section? We dealt with that (at least the practitioner did) using the double-drop method, using what the theorist called an *initial solution* and *local search*. Can we do the same here?

Sounds reasonable. What would be our starting point? Well, *Dijkstra* of course. Start with the routes found by applying DSPA individually but then look at the effect of shared links. Now recalculate the costs for the affected links (with as much or as little sophistication as you like) and find the 'true' cost of the routing for the entire network. Now apply the local search - little tweaks or perturbations to this solution. Consider varying each of the routes, in turn, and in their own various ways. Recalculate costs as before for the affected links and implement the change that maximises the improvement for the network as a whole. Stop when there are no more improvements. No, it will not be *optimal* – these greedy searches rarely are - but it will be a lot *better*! Well done. What a shame it will not work!

The stumbling block this time is not *how* to calculate costs, but *where*! The process we have described above takes a *global* view of the network. (We might well ask how it could be otherwise.) If a network-optimal routing is to be applied then surely it has to be determined *centrally*, in its entirety, then passed to individual network devices to be implemented. Unfortunately, this is not how Internet routing protocols work. Routing is a *distributed* process running *independently* on each router. Although routers exchange routing *information* (connectivity, usage, etc.) to learn the current *shape* and *state* of the network, they do not share routing *intent*. There is no interchange that would allow the global link loads to be determined under a future routing plan. Hence the effect on link attributes and costs can not be determined as suggested and the necessary global view of a routing strategy cannot be taken. Distributed routing protocols share *past* and *present* information but not their view of the *future*. (Perhaps they should but they don't!) Is there anything that can be done about it?

Well, there are some suggestions: algorithms such as *Widest Shortest Path* (*WSP*) routing (Ji and Yeung, 2005) and the *Minimum Interference Routing Algorithm* (*MIRA*) (Salvadori and Battiti, 2003) can be made to work cooperatively in constrained environments (De Ghein, 2006). More generally, *Ant Colony Optimisation* (*ACO*) (Johnson and Perez, 2005), for example, appears to have promise. ACO mimics the way individual ants share information to cooperate to find the best strategy for the good of the colony – in finding food for example. (The link between ants working together by apparently working separately and network routers behaving similarly turns out to be a good one.) In fact, with the newer ideas like this, we see for the first time the theorist and the practitioner beginning to work together – understanding and sharing a common view of the problem and the requirements of a successful solution (Dorigo and Stützle, 2004). The sad fact is that initiatives like this have emerged before and come to nothing. We are still some way away from seeing ACO routing (or anything like it) implemented on a production Internet router.

Lesson Three: Understand the limitations of both models and tools!

5. Conclusions

There is certainly no intention to take sides here. Hopefully the examples given balance the arguments - or maybe apportion blame equally? The theorists are doing good mathematics and the practitioners are solving real problems so no-one is exactly at fault. What is true is that there is a gulf, a reality gap, between the two that probably should be closed, or at least reduced. But, how is this to be done?

Well, to an extent, the process has already begun - and has been going for some time. Of course not all graph algorithms are useless in the real world; if it were not for Dijkstra's algorithm, network routing would be very crude indeed. Some networking solutions are extremely well designed and even elegant - the principles of Internet multicasting, and public-key cryptography (Forouzan, 2003), for example.

And yet, in other areas, the misunderstanding is still there - and needs to be addressed. The theorists are still over-simplifying certain problems. (It might not be obvious that an MST solution suits any real-world application, let alone a networking one.) The practitioners are still using words like ‘optimal’ when they really mean ‘a bit better’; (Internet routing, for example, is still some way from optimal and, in fact can be shown to be effectively insoluble (Wang and Crowcroft, 1996).) At the heart of it all is probably a lack of communication. At the very least, we need to get the theorist and practitioner together; better still, to consider why, after all this time, they still misunderstand each other. Better again, start some dialogue and look for solutions. There are so many good algorithms out there and so many problems crying out for them. (In fact, MST algorithms *are* useful in some real world applications - eliminating loops from bridged networks for example (Norton, 2007).) ACO or some such advance may provide a coherent routing strategy from distributed routing processes - but the ideas have to be implemented. What we need is for the solutions to match the problems - and this is the real challenge.

However, for now, we achieve a lot if we just *understand* the problems and *learn* the lessons:

Lesson One (Algorithms): Find the right tool for the job!

Lesson Two (Models): Find the right model for the situation!

Lesson Three (Scope): Understand the limitations of both models and tools!

Simple enough? Well, let us see …

There will never be sufficient space in a paper of this length to deal with all the complexities of network modelling and optimisation and it is true that we have left a lot unsaid. However, before we finish, just reconsider the question we raised in passing in the routing section; namely, *what should we be trying to optimise?* We offered various suggestions there as to what the objective might be but all were different ways of representing *cost* … cost as measured by traffic carried, the cost of customer dissatisfaction, etc.

But now consider a real-world ‘project’ … say to build a new network … or redimension an existing one … or find a better routing. This project will have a *budget*. The budget will be fixed. *Cost* will be fixed ... *Cost is a constraint, not the objective function*! The objective (subject to a fixed cost constraint) may be to maximise redundancy, minimise delay, etc. but this is a whole new problem! No room for that here though!

Finally, if any of this has made any sense, a challenge … What of *wireless* networks? What new opportunities (or threats) do they present the relationship between the theorist and the practitioner? How do we optimise these? (Not much point in minimising link - i.e. wire - costs in wireless networks is there?) No? So what is the problem then? (Grout, 2005) How is it to be formulated? How is it to be solved? (Morgan and Grout, 2007)

6. References

De Ghein, L. (2006), *MPLS Fundamentals*, Cisco Press.

Dijkstra, E.W. (1959), “A Note on Two Problems in Connexion with Graphs”, *Numerische Mathematik*, vol. 1, pp. 269-271.

Dorigo, M. and Stützle, T. (2004), *Ant Colony Optimization*, MIT Press.

Forouzan, B.A. (2003), *Data Communications and Networking*, McGraw-Hill.

Garey, M.R. and Johnson, D.S. (1979), *Computers and Intractability: A guide to the theory of NP-completeness*, Freeman.

Gredler, H. and Goralski, W. (2004), *The Complete IS-IS Routing Protocol*, Springer-Verlag UK.

Grout, V., Houlden, N., Davies, J., McGinn, J. and Cunningham, S. (2004), "A Unified Framework for Optimal Routing", *System Integration for an Integrated Europe*, J. Ehleman, ed.,K. Antlová, pp. 177-186, Preciosa.

Grout, V. (2005), "Principles of Cost Minimisation in Wireless Networks", *Jnl. Heuristics*, vol. 11, no. 2, pp. 115-133.

Houlden, N., Grout, V., McGinn, J. and Davies, J. (2006), "Extended End-to-End Cost Metrics for Improved Dynamic Route Calculation", *Proc. 6th Int. Network Conf.*, pp89-96.

Hurst, H.E. (1951), "Long-term Storage Capacity of Reserviors", *Trans. American Soc. Engrs.*, vol. 116, pp. 770-808.

ITA. (2007), The Internet Traffic Archive, available at http://ita.ee.lbl.gov/html/traces.html (accessed 21st September 2007).

Ji, L. and Yeung, K.L. (2005), "A Novel Two-step Approach to Restorable Dynamic QoS Routing", *IEEE Jnl. Lightwave Ttechnology*, vol. 23, no. 11, pp. 3663-3670.

Johnson, C.M. and Perez, E. (2005), "An Ant Colony Optimization Algorithm for Dynamic, Multi-Objective Network Routing", *Proc. 1st Int. Conf. Internet Technologies & Applications*, pp. 293-300.

Kingman, J.F.C. (1993), *Poisson Processes*, Oxford University Press.

Kruskal, J.B. (1956), "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem", *Proc. American Math. Soc.*, vol. 7, pp. 48-50.

Leyland, W.E., Taqqu, M.S., Willinger, W. and Wilson, D.V. (1994), "On the Self-similar Nature of Ethernet Traffic (extended version)", *IEEE/ACM Trans. Networking*, vol. 2, no. 1, pp. 1-15.

Mandelbrot, B. and Hudson, R.L. (2005), *The (Mis)behaviour of Markets: A fractal view of risk, ruin and reward*, Profile Books.

Morgan, M. and Grout, V. (2007), "Metaheuristics for Wireless Network Optimisation", *Proc. 3rd Adv. Int. Conf. Telecommunications*, p15.

Moy, J.T. (1998), *OSPF: Anatomy of a Routing Protocol*, AddisonWesley.

Norton, M. (2007), "Understanding Spanning Tree Protocol: The fundamental bridging algorithm", http://www.oreillynet.com/pub/a/network/2001/03/30/net_2nd_lang.html (accessed 21st September 2007).

Park, K., Kim, G. and Crovella, M. (1997), "On the Effect of Self-similarity on Network Performance", *Proc. SPIE Int. Conf. Performance and Control of Network Systems*, pp. 296-310.

Paxson, V. and Floyd, S. (1995), "Wide Area Traffic: The failure of Poisson modelling", *IEEE/ACM Trans. Networking,* vol. 3, issue 3, pp. 226-244.

Prim, R.C. (1957), "Shortest Connection Networks and Some Generalisations", *Bell Sys. Tech. Jnl.*, vol. 36, pp.1389-1401.

Rezaul, K.M., Pakštas, A. and Gilchrist, R. (2006a), "An Investigation of the Properties of the HEAF Estimator using Simulation Experiments and MPEG Encoded Video Traces", *Proc. 10th IEEE Int. Conf. Intelligent Engineering Systems*, pp. 276-281.

Rezaul, K.M., Pakštas, A., Gilchrist, R. and Chen, T.M. (2006b), "HEAF: A Novel Estimator for Long-range Dependent Self-similar Network Traffic", *Next Generation Teletraffic and Wired/Wireless Advanced Networking*, Y. Koucheryavy, J. Harju and V.B Iversen, eds., Springer LNCS 4003, pp. 34-45.

Rezaul, K.M. and Grout, V. (2007a), "CoLoRaDe: A Novel Algorithm for Controlling Long-Range Dependent Network Traffic", *Proc. 6th Int. Conf. Networking*, pp. 57-67.

Rezaul, K.M. and Grout, V. (2007b), "BPTraSha: A Novel Algorithm for Shaping the Bursty Nature of Internet Traffic", *Proc. 3rd Advanced Int. Conf. Telecommunications*, pp. 8-18.

Salvadori, E. and Battiti, R. (2003), "A Load-balancing Scheme for Congestion Control in MPLS Networks", *Proc. 8th IEEE Int. Symp. Computers and Communications*, pp. 951-956.

Sternberg, S. (1995), *Group Theory and Physics*, Cambridge University Press.

Wang, Z. and Crowcroft, J. (1996), "Quality of Service Routing for Supporting Multimedia Applications", *IEEE Jnl. Selected Areas in Communications*, vol. 14, no. 7, pp. 1228-1234.