Network Stability with Delay Minimisation in a QoS based OSPF Network

Chee Yeew Yong, Theodoris Michalareas, Dr. Lionel Sacks

University College London

Abstract: I am investigating a form of adaptive IP routing, which uses a Quality of Service (QoS) metric in order to compute routes. The algorithm for performing this routing is implemented in the Network Simulator-2 simulation engine. Using this simulation engine, results from experiments performed on a simple network topology are analysed and possible conclusions are drawn. The algorithm described in this paper uses link delay as its QoS cost metric, and the routing algorithm uses thresholds and incremental factors in order to determine the relevant cost metric to return to the routing *computation algorithm*. It will be shown that the threshold and incremental factor can be tuned to improve the stability of a single multiplexed flow of traffic with an 'acceptable' trade off in delay.

1 Introduction

Routing in the Internet even up to today is very much an issue, judging from the research still being performed. However, there has been more interest in Quality of Service (QoS) based routing in recent years, [1,2,3] is but to name a few research and publications. QoS based routing is, in its basic form, route selection based on some QoS constraints e.g. bandwidth availability or delay requirements. The problem that QoS routing is trying to solve is the provision of an improved user service level in order to support new multimedia requirements of the Internet today [4]. However, the routing algorithm that I am considering takes into account that the routing protocol, OSPF (Open Shortest Path First), is currently the routing protocol deployed in the Internet. OSPF is a form of *Link State* routing protocol.

In the Network Simulator-2 (NS-2), there is already an implementation of the OSPF protocol called *Link State Routing*. By extending the capability of the Link State (or more accurately, OSPF protocol) protocol in NS-2, an algorithm that takes into account the link delay as its cost metric together with a triggering process that can reduce the number of routing computation is implemented. Reducing route computation satisfies the criterion of *stability* for a routing algorithm that takes this dynamic cost metric. By tuning the threshold and incremental factors, the *QoS criterion* can be achieved.

2 OSPF protocol and Link States

The main feature of OSPF is the link states [5]. Link states (literally, state of the links) are the building blocks to creating a complete 'map' of the topology that is kept by each of the node in the network. This 'map' is represented by the Link State Database. To populate this Link State database, each node essentially 'measures' the cost of each of its links and sends this information to all its neighbours. The cost metric that the OSPF protocol specifies can be of any value, but usually it is the weight given to a particular link administratively reflecting the monetary cost etc. More importantly, it is usually a static value. Conversely, an adaptive, say delay based, routing algorithm would use the delay of the link as its cost metric.

The sending of this information to its neighbours is via *Link State Advertisement (LSA)* packets and it uses the *flooding protocol*. The receiving node for each LSA will replicate the LSA packet and sent it out on its every output link except the one that it had received the LSA from. Currently the LSAs are sent periodically (usually a long period) or if there is change in topology i.e. a link going down (instantaneous).

The OSPF protocol also specifies that the Dijkstra [5] be used as to compute the routes or shortest path tree based on the Link State Database that each node has. Therefore, every node is essentially the sink for a spanning tree joining all the other nodes in the network, after the route computation.

3 Delay based routing algorithm

The delay-based algorithm extends the OSPF protocol, keeping in mind that route computation still uses the Dijkstra algorithm. Dijkstra algorithm has the property of minimising the *cost the routes* that it has computed, and since cost of the routes of a delay-based algorithm is the *sum of individual links' delay* along the routes, this Dijkstra property is advantageous. Take note that the link delay here is the sum of the link *propagation delay* and its *mean queuing delay* over the sampling interval of the link.

Since link state protocol supports very precise metrics, there is no difficulty in keeping precise delay values in the Link State Database [5]. Given the precision of the link delay measured, there is a small possibility of finding alternative routes that have *the same individual link delays,* except at initialisation.

The delay-based algorithm, if implemented in its most primitive form, has the inherent problem of route *fluttering*. Given the precision of the link delay sampled values, it is essentially trivial to find another lower cost route, as opposed to the current route to a destination. Elaboration of this fact is that, by using a particular route, the traffic

being routed there will increase the links' delays, and hence not the shortest (cheapest) cost route anymore. In order to de-sensitise this precision, the delay-based algorithm uses 2 heuristics in order to change the *returned delay value* to the routing protocol as opposed to the *measured delay value*. One heuristic is a threshold based triggering mechanism:

$$\begin{aligned} &if \quad \{abs(\frac{d(m_t)-d(s_{t-1})}{d(s_{t-1})}) \geq \frac{thr}{100}\} \quad \{ \\ &\qquad return \quad d(m_t) \qquad\qquad \text{---------- equation 1} \\ &\qquad d(s_t) = d(m_t) \quad \} \\ &else \quad \{ \quad return \quad d(s_{t-1}) \quad \} \end{aligned}$$

Equation 1: Threshold-delay based algorithm: *abs* stands for absolute values, *d(m)* is the delay measured, *d(s)* is the delay saved, *t* is the current sampling interval ($0 \leq t \leq$ simulation time), *thr* is the threshold.

Using algorithm defined in *equation 1,* a new measured delay value is sent back to the routing protocol only if it exceeds the previous value by a percentage set by *thr* (threshold). Notice that values are still being returned to the routing protocol for updating the Link State Database, but paths are not recomputed unless there is a change in the Link State Database since its last update.

Using an incremental factor to introduce damping into return of the measured delay value can extend the threshold delay-based algorithm even further. The algorithm looks like this:

$$\begin{aligned} &if \quad \{abs(\frac{d(m_t)-d(s_{t-1})}{d(s_{t-1})}) \geq \frac{thr}{100}\} \quad \{ \\ &\qquad return \quad (\frac{incrf}{100}) * d(m_t) \qquad\qquad \text{---------- equation 2} \\ &\qquad d(s_t) = (\frac{incrf}{100}) * d(m_t) \quad \} \\ &\qquad else \quad \{ \quad return \quad d(s_{t-1}) \quad \} \end{aligned}$$

Equation 2: Incremental-threshold-delay based algorithm: Essentially the same as *equation 1*, except for the *incrf* which is the incremental factor.

The incremental factor is essentially restricting the amount to which the *returned delay value* can change. This in turn allows areas of the network to actually 'catch up' with each other since link delay can vary and fluctuate widely among different areas of the network. Note that both *thr* and *incrf* (incremental factor) are set *a priori* as percentages. For *thr*, 0% is the 'normal' OSPF protocol for getting cost measurement from the links; whilst 100% is *double* the previous measured cost (delay or otherwise) of the link and not an upper limit. For *incrf*, 100% is a ceiling since it will then be the same as the threshold-delay based algorithm.

Other 'static' constraints of the delay-based algorithm (generically), are the sampling interval, time to propagate LSAs, size of network topology, and the randomisation of LSAs. Sampling interval affects the number of route changes for a given period of time since for a delay based algorithm (without *thr* or *incrf*), would likely to change route every sampling interval. Size of network and therefore time to propagate the LSAs to other nodes in the network affects the convergence time of routes computed by different nodes in the network. Randomisation of the LSAs is part of the OSPF protocol to avoid the problem of all the nodes in the network, at every sampling interval, flooding its output links with LSAs. This mechanism affects the number of route changes and the convergence period.

4 Simulation setup

NS-2 uses a combination of Tcl scripting language and C++ language to implement and configure its simulation engine. The implementation of the incremental-threshold-delay based algorithm is entirely in Tcl code.

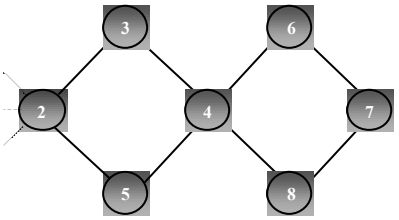

Figure 1: Network topology for simulation in NS-2. Note that the sources are not shown here.
The network topology used for all simulations are shown in *Figure 1*.

The important static settings for the simulations are:
(a) sampling interval for measuring a new link cost = 1 second, (b) randomisation = uniform 0.1 - 1.0 seconds
(c) link bandwidth in network (not including sources) 1.0Mb/s, link bandwidth for sources 2.0Mbit/s

5 Results

Measurements for analysing the traffic in the network are sampled from all the nodes in the network topology shown in *Figure 1*. Traffic is always sent from *Node 2* to *Node 7*. It is measured that, using a script in Tcl, that all 4 possible routes are used by the routing computation algorithm to route traffic. Each simulation is run for 200 seconds with a sampling rate (for producing results) of 0.5 seconds. Simulations will vary either one or both *thr* and *incrf*. Only selected results are shown.

Traffic sources used in simulations:
NeX (negative exponential) sources setting: 4 NeX sources each set to a peak rate of 0.35 Mbit/s
CBR (constant bit rate) setting: 1 CBR source sent at 1Mbit/s
Pareto setting: 4 Pareto sources, each set to a peak rate of 0.35 Mbit/s, with parameter $\propto = 1.5$

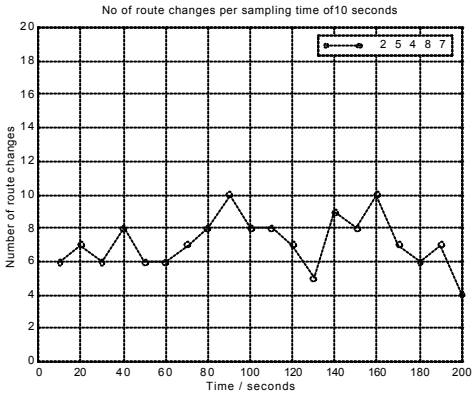

Figure 2: Plot of the number of route changes every 10s over 200s for one possible route. *thr* = 10%, *incrf* =100%. NeX traffic sources used.

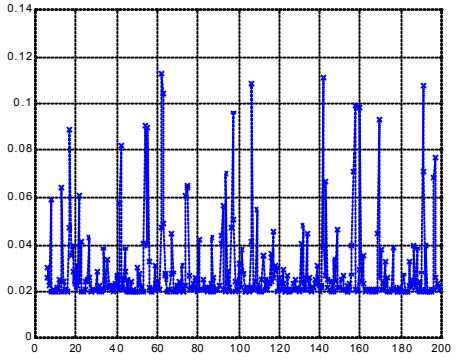

Figure 3: Plot of the delay of the selected route at sampling time. *thr*= 10%, *incrf*= 100%. NeX traffic sources used.

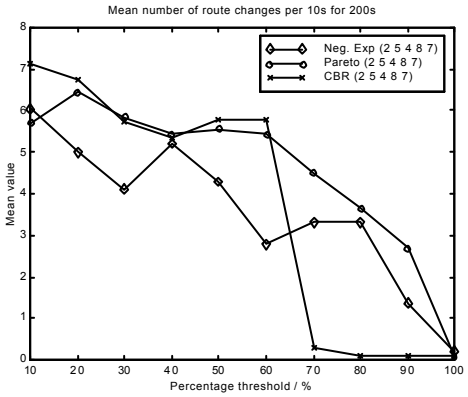

Figure 3: Shows the mean number of route changes every 10s averaged over 200s, with *thr* varying from 10% to 100%, and *incrf* at 100%. Plots for NeX, CBR and Pareto traffic sources and only one particular route shown.

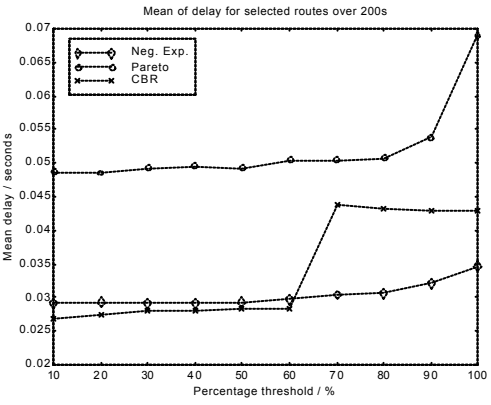

Figure 4: Shows the mean delay of selected route at each sampling (for results) time, with *thr* varying from 10% to 100%, and *incrf* at 100%. Plots for NeX, CBR and Pareto traffic sources are shown.

Figure 2 and *3* are plots of the raw data collected for each simulation. *Figure 2* shows the number of changes for one possible route, which if divided by half and rounded to nearest integer, would give exactly how many times the particular route is used every 10 seconds. *Figure 3* plots the delay of the route (remember that the route delay here is the sum of the individual link delays) used at every sampling (for results) interval. The peaks show the emptying of buffers for the previous route, after a new route was computed.

Figure 3 and *4* are summaries of the plots in *Figure 1* and *2* respectively. They also show plots from different types of traffic sources. It can be observed from *Figure 3* that the mean number of route changes every 10 seconds averaged over 200 seconds decreases as the *thr* value increases. This show that the stability criterion is can be better matched at higher *thr*. Conversely, for the mean delay of the selected routes at each sampling (for results) time, the delay increases as *thr* increases. As the route does not change so frequently as *thr* increases, the nodes' output buffers will start to fill depending on the statistical variation of the traffic sources. Observing both *Figure 3* and *5* (a zoomed version of NeX traffic in Figure 4), a trade off between increasing delay of routes (QoS criterion) and decreasing the number of route changes (stability criterion) can be found at *thr* from 70% to 90% if NeX traffic sources are used. This *operating region* is also similar to that of Pareto traffic sources. The only difference between Pareto and NeX traffic is that more packets are buffered due to statistical variation of the traffic sources leading to higher delays but the general trend between the two are similar. The operating region for CBR is lower because of the small statistical variation nature of CBR traffic source but the changes are more starkly observed. Point to note is that in a real network with real traffic, we would expect the traffic to be a mix of Pareto and Poisson (multiplexing NeX traffic sources approximates Poisson). Since their operating regions are similar, this would probably allow the same incremental-threshold-delay based algorithm to run on their aggregated traffic flow.

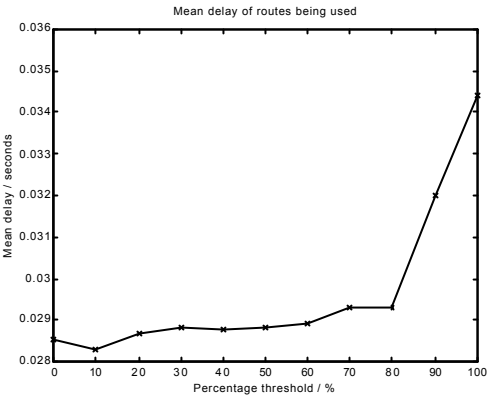

Figure 5: Plot of mean delay of selected route at each sampling (for results) time, with *thr* varying from 10% to 100%, and *incrf* at 100%. NeX traffic sources only.

6 Conclusion

The operating region of a single traffic flow can be found by just tuning the *thr* value and setting a constant *incrf*, even though the criterion for stability and QoS requirement are antithesis. For Pareto and negative exponential traffic sources, the operating regions are similar for a single multiplexed flow of traffic, but for CBR, it is lower. These are however, coarse-grained granularity measurements that approximate the operating region sufficiently well to produce an 'acceptable' trade off between the criterions. It is hypothesised by tuning the *incrf*, more fine-grained conclusions can be made of the operating region.

7 Future work

Future work will be looking at tuning the *incrf* and also mixing appropriate background traffic in order to evaluate its effect on the operating regions.

8 Acknowledgements

I would like to thank Dr. Lionel Sacks, my supervisor, for his invaluable ideas and comments, Theodore Michalereas, my Ph.D. collaborator for his guidance and help in this project, and Chiang Kang Tan, my colleague, for providing some useful insights.

9 References

[1] R.Guérin, D. Williams, A. Orda, QoS Routing Mechanisms and OSPF Extensions, proceeding of GLOBECOMM 1997

[2] G.Apostolopoulos, R.Guérin, S.Kamat, S.K. Tripathi, QoS Routing: A Performance Perspective, proceedings of ACM SIGCOMM 1998

[3] Shigang Chen, Klara Nahrstedt, An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions, IEEE Network Magazine, Special Issue on Transmission and Distribution of Digital Video, Vol. 12, No. 6, November-December 1998, pp. 64-79.

[4] Z.Wang, J.Crowcroft, Quality of Service Routing for Supporting Multimedia Applications, IEEE Journal Selected Areas in Communications, 1996

[5] C.Huitema, Routing in the Internet, Prentice Hall Inc., New Jersey 1995