This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Analysis of RSVP-TE Graceful Restart

O. Komolafe and J. Sventek
Dept. of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK.
femi@dcs.gla.ac.uk, joe@dcs.gla.ac.uk

***Abstract*—GMPLS is viewed as an attractive intelligent control plane for different network technologies and graceful restart is a key technique to ensure this control plane is resilient and able to recover adequately from faults. This paper analyses the graceful restart mechanism proposed for a key GMPLS protocol, RSVP-TE. A novel analytical model, which may be readily adapted to study other protocols, is developed. This model allows the efficacy of graceful restart to be evaluated in a number of scenarios. It is found that, unsurprisingly, increasing control message loss and increasing the number of data plane connections both increased the time to complete recovery. It was also discovered that a threshold exists beyond which a relatively small change in the control message loss probability causes a disproportionately large increase in the time to complete recovery. The interesting findings in this work suggest that the performance of graceful restart is worthy of further investigation, with emphasis being placed on exploring procedures to optimise the performance of graceful restart.**

I. Introduction

It is widely accepted that Generalized Multi-Protocol Label Switching (GMPLS) is an attractive automated, intelligent control plane for different network technologies. GMPLS comprises of IP-based protocols to support routing, signalling and link management that, when properly orchestrated, will simplify network operation and offer the possibility of potentially lucrative, novel, on-demand services. Consequently, there have been significant developments in the standardisation of GMPLS and the salient parts of the GMPLS architecture have been well-defined [12], [4]. GMPLS typically comprises of the use of Open Shortest Path First - Traffic Engineering extensions (OSPF-TE) [9] for routing, the Resource Reservation Protocol - Traffic Engineering extensions (RSVP-TE) [1], [3] for signalling and the Link Management Protocol (LMP) [10] for link management.

As with any network architecture, resiliency is a key requirement of GMPLS controlled networks. Since GMPLS may be used for a range of different network technologies, most of the work on data plane resiliency is applicable. However, the fact that GMPLS typically necessitates a separation of the data plane and control plane means it is often necessary to consider control plane resilience independently. Consequently, control plane reliability and resilience is a topic gaining increasing attention [11] . While approaches that seek to minimise the possibility of control plane failures have been proposed [7], the consensus within the community is that mechanisms to ensure the GMPLS control plane recovers adequately from failure are more likely to be deployed and, hence, such mechanisms are attracting ever-increasing attention [3], [14], [6].

Approaches to ensure that the control plane recovers adequately from faults are mostly based on "graceful restart". Graceful restart mechanisms have been defined for many routing protocols, exploiting the fact that most modern routers separate the routing and forwarding processes. Therefore, since it is possible for the routing process to fail independently of the forwarding process, it is desirable for traffic forwarding to continue in the presence of a routing process fault and for the routing process to be restored as quickly and efficiently as possible. Graceful restart techniques define the procedures that allows these goals to be met. Naturally, the exact graceful restart procedures are protocol-dependent but have many common aspects, including the requirements for the restarting router to inform its neighbours whether it supports graceful restart, for the neighbours to detect when it has failed and restarted, for the neighbours to attempt to minimise data plane disruption and for the restarting router to resynchronise its database with those of the neighbours after the restart.

Since RSVP-TE is a key GMPLS protocol and its graceful restart techniques are as yet not completely standardised, this paper studies the graceful restart mechanisms which have been proposed for RSVP-TE to date, using an analytic method that may be readily adapted to other protocols. Section II describes the graceful restart mechanisms proposed for RSVP-TE and a novel analytical model for evaluating its performance is developed in Section III. Numerical results are presented in Section IV and Section V concludes the paper and suggests avenues for further work.

II. RSVP-TE Graceful Restart

The RSVP-TE hello extension, defined in RFC 3209 [1], forms the basis of RSVP-TE graceful restart since it provides a means for an RSVP-TE node to detect when a neighbour is unreachable or when it has restarted. The hello extension requires *Hello* messages, shown in Figure 1, to be exchanged at regular intervals (5ms suggested [1], although 3s is a more practical default) and the failure to receive a *Hello* message within a certain interval (default is $3\frac{1}{2}$ times the hello interval [1]) means a node presumes it can no longer communicate with its neighbour. In order to detect when a neighbour has restarted, *Hello* messages contain a *Src_Instance* field and a *Dst_Instance* field, as shown in Figure 1. Each node fills the *Src_Instance* field with a value representing its per neighbour instance and fills the *Dst_Instance* field with the *Src_Instance* value most recently received from the neighbour. Since whenever a node restarts it changes its *Src_Instance* value, restarts are easily detectable.

RFC 3473 [3] enhances the RSVP-TE hello extension to specify procedures for detecting and handling control channel faults and nodal faults. A node that supports RSVP-TE

1-4244-0353-7/07/$25.00 ©2007 IEEE

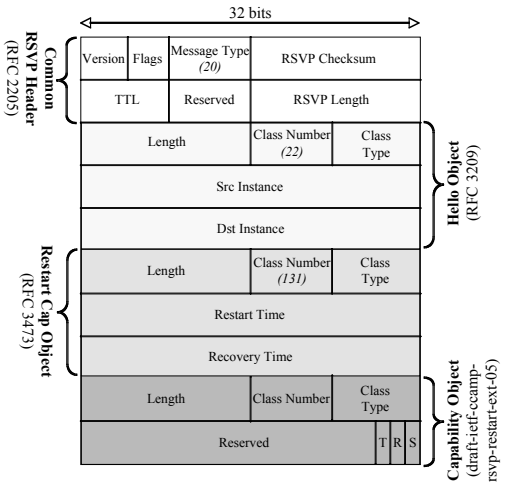

Fig. 1. Format of RSVP-TE Hello Message

graceful restart adds a new *Restart_Cap* object to its *Hello* messages. This object includes a *Restart Time* field and a *Recovery Time* field. The *Restart Time* is the time the sender takes to restart. If the data plane is unaffected by control plane failure so that the restart may occur over an indeterminate time, the sender indicates an infinite *Restart Time* with the value 0xffffffff. However, if a finite restart time is specified, after detecting that communication with the sender is lost, a neighbour waits for at least the duration specified by the *Restart Time*, behaving as if it were receiving the relevant RSVP-TE refresh messages from the sender for established data plane connections and sending only *Hello* messages with the *Dst_Instance* set to 0 to the sender, as shown in Figure 2. Eventually, the neighbour receives a *Hello* message from the sender. If the *Src_Instance* value is unchanged, then there has been a control channel failure and the nodes refresh all shared state. If, on the other hand, the *Src_Instance* value has changed, the sender must have failed and restarted. The *Recovery Time* field in the received *Hello* message also indicates whether the restarting node was able to preserve its forwarding state and, if so, the duration for which it is willing to participate in the recovery process. The behaviour of a neighbour thereafter is dependent upon whether it is upstream or downstream of the restarting node.

An upstream neighbour refreshes all *Path* state that it shares with the restarting node. Each *Path* message contains a *Recovery Label* object, corresponding to the label value in the most recently received *Resv* message from the restarting node. Since there may be a large number of *Path* messages produced and it is essential not to overwhelm the restarting node with a deluge of messages in a short time interval, it is recommended that the *Path* messages be distributed evenly over half the recovery time. Upon receipt of a *Path* message containing a *Recovery Label* object from its upstream neighbour, the restarting node searches its forwarding table for the corresponding entry. If found, the appropriate RSVP state is created and the entry bound to the associated data plane connection [3].

Procedures for a downstream neighbour have been more recently defined and are currently being standardised [14], allowing the restarting node to recover all the necessary state regardless of its position. These extensions mean the restarting node can obtain all the information it previously transmitted in *Path* messages from its downstream neighbour. A newly defined *RecoveryPath* message conveys the information contained in the most recently received *Path* message to the restarting node. An ability to send and receive *RecoveryPath* messages is indicated by appropriate fields in the newly defined *Capability* object included in the *Hello* messages. The downstream neighbour sends a *RecoveryPath* message for each data plane connection associated with the restarting node for which it had previously sent a *Resv* message, spacing the messages over half of the recovery period to avoid inundating the restarting node. Upon the receipt of a *RecoveryPath* message, the restarting node checks its forwarding table for the corresponding entry and responds with a *Path* message, as shown in Figure 2. The downstream neighbour responds to this *Path* message with a *Resv* message which the restarting node processes and duly forwards to the upstream neighbour.

The receipt of a *Path* message containing the *Recovery Label* object from the upstream neighbour and the *RecoveryPath* and *Resv* messages from the downstream neighbour means the restarting node is able to reconstruct its RSVP state. It is desirable to complete graceful restart as quickly as possible. Consequently, the total time to reconstruct and resynchronise the appropriate RSVP states for all the relevant connections, allowing normal control plane operations to resume, is pivotal since any state that is not resynchronised is typically cleared at the end of the recovery period and the corresponding data plane connections removed. An analytical model which may be used to estimate the time to complete graceful restart, as a function of the number of connections, durations of the different constituent stages and message loss probability is developed in Section III.

III. Analytical Model

The key stages in RSVP-TE graceful restart, described in Section II, may be modelled using an absorbing Markov chain. The analytical model used in this paper exploits known results about absorbing Markov chains [5] and is derived from a model previously used to study the stability of routing protocols [15]. The model focuses on exchanges between the restarting node and its downstream neighbour and allows the time to complete recovery to be calculated in different circumstances.

The probabilities of a packet being lost on links *t*owards the failed node and on links *f*rom the failed node are p_t and p_f respectively. These different values are used because of the likelihood that the system will behave as if there is a greater likelihood of message loss on links towards the restarting node due, for example, to the restarting node being highly loaded because of a deluge of messages being sent to it. Figure 3 shows a state diagram focusing on the exchanges between the restarting node and the downstream neighbour during the recovery period. In addition to the transition probabilities,

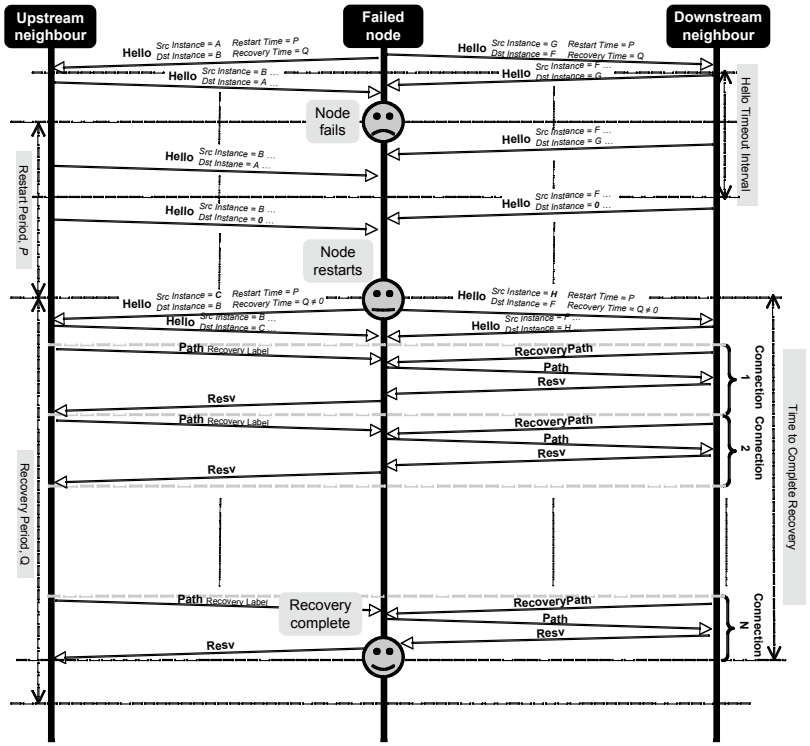

Fig. 2. Key steps in RSVP-TE graceful restart

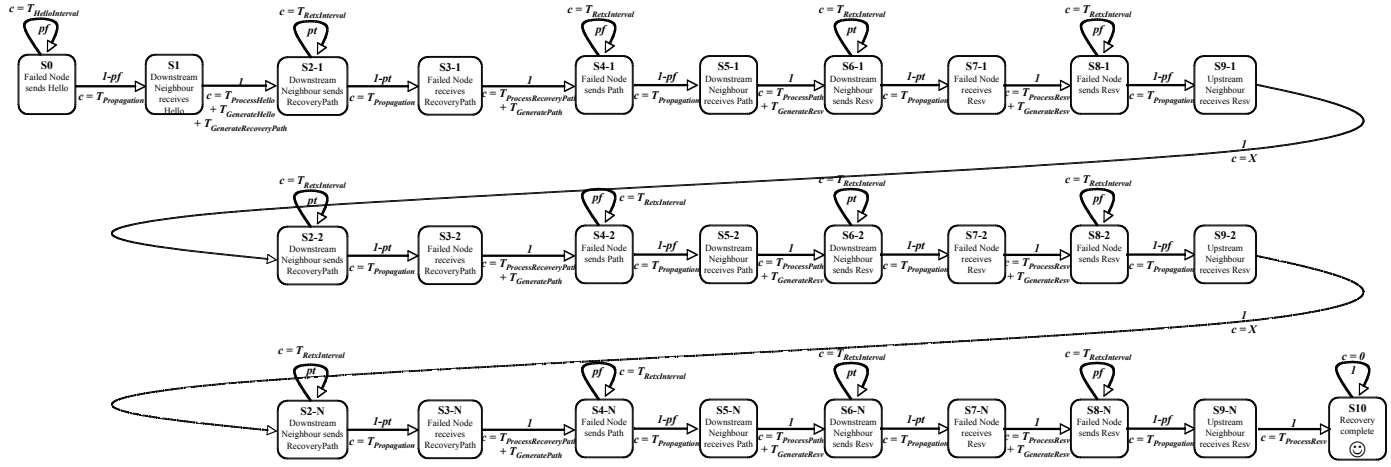

Fig. 3. Markov chain modelling RSVP-TE graceful restart

Figure 3 shows the cost, c, associated with each transition. This cost is an estimate of the time taken for the corresponding transition.

The system begins in State *S0*, where the restarting node sends an *Hello* message to the downstream neighbour. Given that the message loss probability on links from the restarting node is p_f, this *Hello* message reaches the downstream neighbour with a probability of $1 - p_f$, the transition probability from State *S0* to *S1*. The cost of this transition is the corresponding time, the propagation delay, $T_{Propagation}$. On the other hand, if this *Hello* message is lost, another *Hello* message will be produced after the hello interval, $T_{HelloInterval}$. Hence, there is a transition from State *S0* to itself with a transition probability of p_f and a cost of $T_{HelloInterval}$. In State *S1*, the downstream neighbour has received the *Hello* message from the restarting node. The downstream neighbour subsequently processes the *Hello* message, taking a time of $T_{ProcessHello}$ to do so, generates and sends a response *Hello* message, taking $T_{GenerateHello}$, and creates the *RecoveryPath* message for the first connection, taking $T_{GenerateRecoveryPath}$. The

summation of these times is the cost of the transition from State *S1* to State *S2-1* and the transition probability is 1 since it is assumed that, after receiving the *Hello* message, the downstream neighbour always eventually sends a *RecoveryPath* message.

The transitions between State *S2-1* and State *S9-1* are repeated once for each data plane connection; i.e. States *2-2* to *9-2* refer to the second connection and so on, up to States *2-N* to *9-N* which correspond to the N^{th} connection. In State *S2-1*, a hello adjacency has been formed between the downstream neighbour and restarting node and so the downstream neighbour sends a *RecoveryPath* message for the first connection. This message may be lost with a probability of p_t, the loss probability on links towards the restarting node. If lost, a retransmission occurs after an appropriate interval, $T_{RetxInterval}$, and so there is a corresponding transition from State *S2-1* to itself. On the other hand, the *RecoveryPath* message reaches the restarting node with a probability of $1-p_t$. Therefore, the transition probability from State *S2-1* to State *S3-1* is $1-p_t$ and the associated cost is $T_{Propagation}$, the propagation delay. Upon receiving the *RecoveryPath* message, the restarting node processes the message and generates an appropriate *Path* message, taking $T_{ProcessRecoveryPath}$ and $T_{GeneratePath}$ respectively. Summing these values gives the costs associated with the transition from State *S3-1* to State *S4-1*; the transition probability is 1 since it is assumed the restarting node always processes the received *RecoveryPath* message and generates a *Path* message. The remaining transitions up to State *S9-1* may be readily understood from the message exchanges depicted in Figure 2.

In State *S9-1*, the upstream neighbour has received the *Resv* sent by the restarting node and so the recovery is complete for the first connection. Recovery for the second connection commences with the transition from State *S9-1* to State *S2-2* with a probability of 1. The cost of this transition, the interval between the end of recovery of the first connection and start of recovery of the second connection, is *X*. In the "worst case" scenario, the message exchanges pertaining to each connection are carried out serially, hence *X* is the time taken for the upstream neighbour to process the *Resv* it received from the restarting node for the last connection and for the downstream neighbour to generate the *RecoveryPath* message for the next connection, i.e. $X = T_{ProcessResv} + T_{GenerateRecoveryPath}$. On the other hand, it is likely that some pipelining will occur. In an attempt to expedite the recovery process, neighbours may send recovery messages for subsequent connections without waiting for the previous connections to be completely recovered. If it is assumed that the time to recover each connection is independent of the total number of connections being simultaneously recovered, pipelining may be modelled by having $X < 0$. Since *X* refers to the time taken for the transition, it may appear strange for *X* to have a a negative value, however, $X < 0$ simply means an attempt is made to recover the subsequent connection before the previous connection is completely recovered. The Markov property means the value of *X* will not affect the behaviour of the rest of the system. Hence, by setting *X* appropriately, the impact of pipelining the recovery of connections can be investigated.

Two of the main approximations necessary to reduce complexity and ensure the model remains tractable are:

- RFC 2961 [2], which defines the *Ack* message and the retransmission algorithm for RSVP messages, suggests the use of an exponential back off algorithm for unacknowledged trigger messages. The suggested defaults were an initial default retransmission interval of 0.5s (or the round-trip time, if known), doubling this interval between successive retransmissions and limiting the number of retransmissions to three. In the system depicted in Figure 3, it is evident that the retransmission interval, $T_{RetxInterval}$ remains fixed and the number of retransmissions is unrestricted. Furthermore, the impact of sending or losing *Ack* messages is not incorporated in the model. These approximations, essential to make the model tractable, mean that the model will slightly outperform an implementation adhering to the suggested default values.
- Pipelining the recovery of the connections will incur some additional overhead (e.g. due to resource contention and sharing) which may result in an increase in the recovery time as the number of connection being simultaneously recovered rises. However, for simplicity, the model assumes that the time to recover each connection is independent of the total number of connections being simultaneously recovered. This assumption means the model will likely outperform a real-life implementation.

Given the Markov chain illustrated in Figure 3, known properties of absorbing Markov chains [5] may be used to compute the average time to absorption in State *S10*. This time, T, is given in Equation 1.

$$\begin{aligned} T &= T_{GenerateHello} + T_{ProcessHello} \\ &\quad + T_{GenerateRecoveryPath} \\ &\quad + (N+1)T_{ProcessResv} + (4N+1)T_{Propagation} \\ &\quad + (N-1)X + \frac{p_f T_{HelloIntveral}}{1-p_f} \\ &\quad + N[\frac{2p_f T_{RetxIntveral}}{1-p_f} + \frac{2p_t T_{RetxIntveral}}{1-p_t} \\ &\qquad + T_{ProcessRecoveryPath} + T_{GeneratePath} \\ &\qquad + T_{ProcessPath} + 2T_{GenerateResv}] \end{aligned} \quad (1)$$

Equation 1 means that the time for graceful restart to be completed may be readily computed for a given number of connections and packet loss probabilities, provided the duration of the key constituent processes are known. Obtaining Equation 1 is a key contribution of this work since it suggests that an appropriate analytical model may be applied to investigating the performance of graceful restart. This model may be readily adapted to different implementations of RSVP-TE graceful restart or to other protocols. Section IV assigns reasonable exemplar values to parameters in Equation 1, allowing the performance of RSVP-TE graceful restart to be evaluated in a number of scenarios.

IV. Exemplar Numerical Results

Some of the parameters in Equation 1 have default values suggested which are initially used in this section; the hello

interval is set to 5ms [1] and the retransmission interval is set to 500ms [2]. Some previous work has measured the time taken by an exemplar RSVP-capable router to process different RSVP messages in a number of scenarios, finding typical durations being a few milliseconds [13]. Hence, it is assumed that it takes 2ms to generate and process *Hello* messages. Since *Path*, *RecoveryPath* and *Resv* messages are larger and contain more RSVP objects than *Hello* messages, it is assumed that the time taken to generate these messages is 10ms . Upon receipt of a *Path*, *RecoveryPath* or *Resv* message, the restarting node must search its forwarding plane, create corresponding RSVP state and so on, hence the time to process these messages, is assumed to be a relatively large 40ms. Lastly, the propagation delay is set to a moderate value of 0.1ms. It should be noted that the values assigned to the different processes in this section are merely exemplar values; the form of Equation 1 means alternative values may be easily entered and results obtained.

When these values are entered into Equation 1 with the probability of message loss being equal in either direction (i.e. $p_t = p_f$) and with the connections being recovered serially (i.e. $X = T_{ProcessResv} + T_{GenerateRecoveryPath}$), the results obtained for a range of packet loss probabilities and different number of connections is shown in Figure 4. Figure 4 shows that, unsurprisingly, a rise in the packet loss probability leads to an increase in the time to complete recovery, an increase attributable to more retransmissions due to the higher packet loss. It is also evident that increasing the number of connections leads to a rise in the time taken to complete recovery, for any given packet loss probability, due to the fact that exchanging the control messages and creating the appropriate RSVP state for the connections are carried out sequentially.

The retransmission interval is a configurable parameter in Equation 1 and the impact of reducing it from 500ms to 50ms is shown in Figure 5. Figure 5 shows that decreasing the retransmission interval typically leads to a significant decrease in the time to complete recovery for any given packet loss probability and, furthermore, the impact of the number of connections is diminished. The hello interval is another configurable parameter and it was increased from the 5ms default value in RFC 3209 [1] to the more realistic value of 3s. The results, omitted for brevity, were similar to Figure 4. Taken together, these two observations suggest that the retransmission interval is more consequential than the hello interval. This finding may be explained by noting that the number of critical *Hello* messages in the graceful restart process is small in comparison to the number of critical *RecoveryPath*, *Path* and *Resv* messages (a discrepancy that rises with increasing number of connections), hence, unsurprisingly, the configurable parameter that affects these three messages is of greater significance. This observation suggests that, during the recovery period, more emphasis should be placed on reducing the time taken for events which are affected by the number connections. So, for example, increasing the hello interval while decreasing the retransmission interval by a commensurate amount is likely to improve the overall performance, without affecting the message processing overhead excessively.

Figure 6 shows that, if the loss probability of messages sent to the restarting node is a 1000 times that of messages it sends (i.e. $p_t = 1000p_f$), there is a drop in the time to complete recovery when compared to the results in Figure 4. However, it is somewhat surprising that the decrease is relatively small, suggesting that the "request and response" nature of the exchanges during graceful restart means that the performance of the worse party is often the determining factor in the overall performance. An example of a scenario in which this asymmetric behaviour may arise is when the restarting node is highly loaded (due to performing recovery with multiple nodes simultaneously) and so fails to receive and process the messages sent by the downstream neighbour yet the downstream neighbour is able to receive and process any messages it receives. This observation suggests that the message processing capacity of the restarting node is likely to be a performance bottleneck during graceful restart.

Thus far, the worst case scenario in which the connections are recovered sequentially has been considered. In order to investigate the impact of pipelining the recovery of the connections, the value of *X* was chosen to model the case when an attempt is made to begin recovery of the next connection 20ms after the start of the attempt to recover the previous connection. The results are presented in Figure 7. As would be expected, there is a significant drop in the time to complete recovery, when Figure 7 is compared to Figure 4. Furthermore, the impact of the the number of connections on the recovery time is diminished. Hence, pipelining is likely to significantly improve the overall performance. Nevertheless, it is pivotal that a judicious spacing the of the recovery process for the connections is undertaken, to minimise any resource sharing and contention which will degrade the performance.

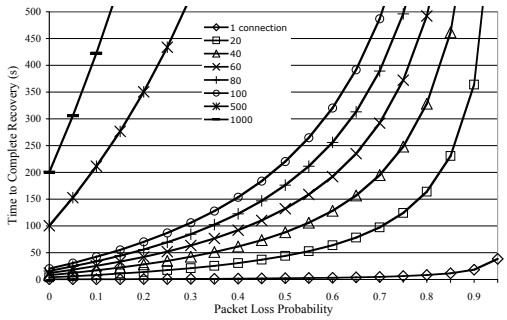

Fig. 4. Impact of packet loss on time to complete recovery when connections recovered serially ($T_{RetxInterval} = 500ms$, $T_{HelloInterval} = 5ms$ & $p_t = p_f$)

Arguably the most striking feature of Figures 4 to 7 is that, in most cases, the time to complete recovery rises exponentially with the packet loss probability. Consequently, at some threshold, there is a great increase in the time to complete recovery for a relatively small change in the packet loss probability. Admittedly, it is unlikely that the packet loss probabilities in the control plane will be as large as these threshold values for a prolonged duration. Nevertheless, the

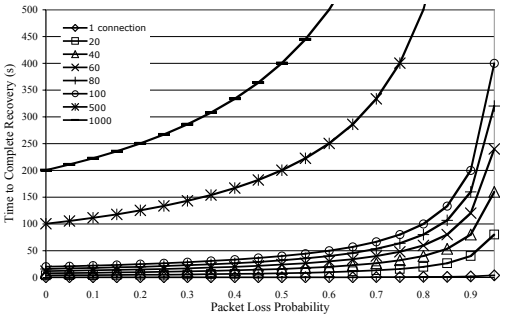

Fig. 5. Impact of packet loss on time to complete recovery when connections recovered serially ($T_{RetxInterval} = 50ms$, $T_{HelloInterval} = 5ms$ & $p_t = p_f$)

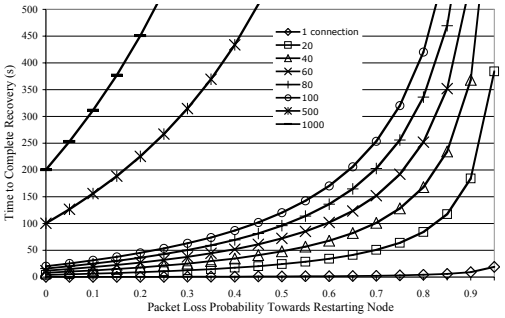

Fig. 6. Impact of packet loss on time to complete recovery when connections recovered serially ($T_{RetxInterval} = 500ms$, $T_{HelloInterval} = 5ms$ & $p_t = 1000p_f$)

existence of these trends suggest that studying the effect of non-ideal behaviour during graceful restart is worthwhile since they suggest that the effect of additional errors on the already degraded control plane may be severe.

V. Conclusions and Future Work

This paper has studied the graceful restart mechanism proposed for RSVP-TE, a key GMPLS protocol. A novel analytical model, which may be readily adapted to study other protocols, has been developed. This model allows the impact of the duration of the key constituent stages, the loss of control messages and the number of data plane connections on the efficacy of graceful restart to be evaluated. It was found that, unsurprisingly, increasing control message loss and increasing the number of connections increased the time to complete recovery. It was also discovered that a threshold exists beyond which a relatively small change in the message loss probability causes a disproportionately large increase in the time to complete recovery.

The analytical model presented in this paper is viewed as a first step in the evaluation of RSVP-TE graceful restart. Naturally, simplifications had to made when developing the model, implying the results presented in this paper are likely to be somewhat idealistic. Hence, an interesting avenue of future work is to enhance the analytical model, or to employ other performance evaluation techniques (e.g. simulation or

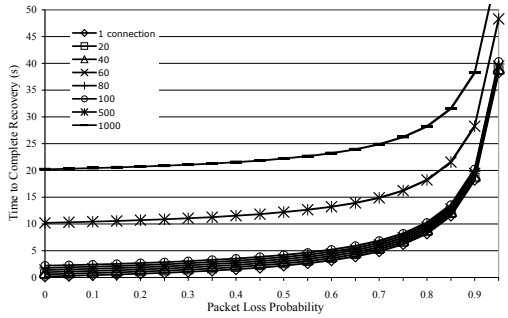

Fig. 7. Impact of packet loss on time to complete recovery when connections recovery pipelined ($T_{RetxInterval} = 500ms$, $T_{HelloInterval} = 5ms$ & $p_t = p_f$)

measurement), to study graceful restart thoroughly. Interesting open issues include quantifying the impact of the heavy load on the restarting node, identifying the optimal spacing of the recovery process for different connections, determining the most expedient recovery period in different circumstances and studying the impact of multiple nodal/link faults on the graceful restart.

VI. Acknowledgments

The authors wish to thank the UK Engineering and Physical Sciences Research Council for their support of this research through grant EP/C004442/1. The authors also thank Adrian Farrel for helpful comments on the paper.

References

[1] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels', RFC 3209, Dec. 2001.
[2] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, S. Molendini, "RSVP Refresh Overhead Reduction Extensions" RFC 2961, April 2001.
[3] L. Berger (Ed.), "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions" RFC 3473, Jan. 2003.
[4] A. Farrel, I. Bryskin, "GMPLS Architecture and Applications", Morgan Kaufmann, 2006.
[5] C. Grinstead, J. Snell, "Introduction to Probability", AMS, 1997.
[6] A. Jajszczyk and P. Rozycki, "Recovery of the Control Plane after Failures in ASON/GMPLS Networks", IEEE Network, pp.4-10, Jan/Feb 2006.
[7] Y. Kim, "Requirements for the Resilience of Control Plane", draft-kim-ccamp-cpr-reqts-01.txt, Oct. 2005.
[8] K. Kompella (Ed.), Y. Rekhter (Ed.), "Routing Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)" RFC 4202, Oct. 2005.
[9] K. Kompella (Ed.), Y. Rekhter (Ed.), "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)" RFC 4203, Oct. 2005.
[10] J. Lang (Ed.) "Link Management Protocol (LMP)" RFC 4204, Oct. 2005.
[11] G. Li, J. Yates, D. Wang, C. Kalmanek, "Control Plane Design for Reliable Optical Networks", IEEE Communications Magazine, pp. 90-96, February 2002.
[12] E. Mannie (Ed.), "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", RFC 3945, Oct. 2004.
[13] A. Neogi, T. Chiueh, P. Stirpe, "Performance Analysis of an RSVP-Capable Router", IEEE Network, pp.56-63, Sept./Oct. 1999.
[14] A. Satyanarayana (Ed.), R. Rahman (Ed.), "Extensions to GMPLS RSVP Graceful Restart", draft-ietf-ccamp-rsvp-restart-ext-05.txt, Oct. 2005.
[15] A. Shaikh, A. Varma, L. Kalampoukas, R. Dube, "Routing Stability in Congested Networks: Experimentation and Analysis", Proc. ACM SIGCOMM 2000.