Internet Engineering Task Force
Internet Draft
Expires in May, 2003
draft-ietf-ospf-scalability-02.txt

Gagan L. Choudhury
Vera D. Sapozhnikova
AT&T

Anurag S. Maunder
Sanera Systems

Vishwas Manral
Netplane Systems

November, 2002

Explicit Marking and Prioritized Treatment of Specific OSPF Packets for Faster Convergence and Improved Network Scalability and Stability

Status of this Memo

Abstract

In this draft we propose the following mechanisms to improve the scalability and stability of OSPF-based network:

(1) Process the Hello packets at a higher priority compared to other OSPF packets. In order to facilitate this, explicitly mark the Hello packets, to differentiate them from other OSPF packets. One way of special marking is to use a different Diffserv

codepoint for Hello packets compared to other OSPF packets.

(2) In the absence of special marking, or in addition to it, use other mechanisms in order not to miss Hello packets. One example is to treat any packet received over a link as a surrogate for a Hello packet (an implicit Hello) for the purpose of keeping the link alive.

(3) The same type of explicit marking and prioritized treatment may be beneficial to other OSPF packets as well. One important example is LSA acknowledgment packet that can reduce retransmissions during periods of congestion. Other examples include (a) Database description (DBD) packet from a slave that is used as an acknowledgement, and (b) LSAs carrying intra-area topology change information.

It is possible that some implementations are already using one or more of the above mechanisms in order not to miss the processing of critical packets during periods of congestion. However, we suggest the above mechanisms to be included as part of the standard so that all implementations can benefit from them.

Table of Contents

1. Introduction

Due to world-wide increased traffic demand, data networks are ever increasing in size in terms of number of nodes, number of links, adjacencies per node and Link State Database size. Our motivation is to improve the ability of large networks to withstand the simultaneous or near-simultaneous update of a large number of link-state-advertisement messages, or LSAs. We call this event, an LSA storm. An LSA storm may be initiated due to many reasons. Here are some examples:

(a) one or more link failures due to fiber cuts,

(b) one or more node failures for some reason, e.g., software crash or some type of disaster in an office complex hosting many nodes,

(c) requirement of taking down and later bringing back many nodes during a software/hardware upgrade,

(d) near-synchronization of the once-in-30-minutes refresh instants of some types of LSAs,

(e) refresh of all LSAs in the system during a change in software version.

In addition to the LSAs generated as a direct result of link/node failures, there may be other indirect LSAs as well. One example in MPLS networks is traffic engineering LSAs generated at other links as a result of significant change in reserved bandwidth resulting from rerouting of Label Switched Paths (LSPs) that went down during the link/node failure.

The LSA storm causes high CPU and memory utilization at the node processors causing incoming packets to be delayed or dropped. Delayed acknowledgements (beyond the retransmission timer value) results in retransmissions, and delayed Hello packets (beyond the Router-Dead interval) results in links being declared down. A trunk-down event causes Router LSA generation by its end-point nodes. If traffic engineering LSAs are used for each link then that type of LSAs would also be generated by the end-point nodes and potentially elsewhere as well due to significant changes in reserved bandwidths at other links caused by the failure and reroute of LSPs originally using the failed trunk. Eventually, when the link recovers that would also trigger additional Router and traffic engineering LSAs.

The retransmissions and additional LSA generations result in further CPU and memory usage, essentially causing a positive feedback loop. We define the LSA storm size as the number of LSAs in the original storm and not counting any additional LSAs resulting from the feedback loop described above. If the LSA storm is too large then the positive feedback loop mentioned above may be large enough to indefinitely sustain a large CPU and memory utilization at many network nodes, thereby driving the network to an unstable state.

In the past, network outage events have been reported in IP and ATM networks using link-state protocols such as OSPF, IS-IS, PNNI or some proprietary variants. See, for example [Ref1-Ref4]. In many of these examples, large scale flooding of LSAs or other similar control messages (either naturally or triggered by some bug or inappropriate procedure) have been partly or fully responsible for network instability and outage.

It has been suggested [Ref5] to reduce the Hello interval and Router-Dead interval significantly in order for OSPF to detect link failures and recoveries faster. Reduction of Router-Dead interval would make it even more likely for links to be declared down due to missed Hellos.

We use a simulation model to show that there is a certain LSA storm size threshold above which the network may show unstable behavior caused by large number of retransmissions, link failures due to missed Hello packets and subsequent link recoveries. We also show that the LSA storm size causing instability may be substantially increased by providing prioritized treatment to Hello and LSA Acknowledgment packets. Furthermore, if we prioritize Hello packets then even when the network operates somewhat above the stability threshold, links are not declared down due to missed Hellos. This implies that even though there is control plane congestion due to many retransmissions, the data plane stays up and no new LSAs are generated (besides the ones in the original storm and the refreshes). Based on these observations we propose prioritized treatment of Hello, LSA acknowledgment and other critical OSPF packets and a special marking to facilitate that.

One might argue that the scalability issue of large networks should be solved solely by dividing the network hierarchically into multiple areas so that flooding of LSAs remains localized within areas. However, this approach increases the network management and design complexity and may result in less optimal routing between areas. Also, ASE LSAs are flooded throughout the AS and it may be a problem if there are large numbers of them. Furthermore, a large number of summary LSAs may need to be flooded across Areas and their numbers would increase significantly if multiple Area Border Routers are employed for the purpose of reliability. Thus it is important to allow the network to grow towards as large a size as possible under a single area.

Our proposal here is synergistic with a broader set of scalability and stability improvement proposals. [Ref6, Ref7] proposes flooding overhead reduction in case more than one interface goes to the same neighbor. [Ref8] proposes a mechanism for greatly reducing LSA refreshes in stable topologies. [Ref9] compares several restricted flooding algorithms in terms of their ability to withstand large LSA storms and robustness to failure conditions. [Ref10] proposes a wide range of congestion control and failure recovery mechanisms.

Section 2 describes the network under simulation and Section 3 provides the simulation results. Section 4 gives the basic observations based on the simulation results. Section 5 explains the need for prioritized treatment of certain critical OSPF packets and special marking to facilitate that. Section 6 gives the summary.

2. The Network Under Simulation

We generate a random network over a rectangular grid using a modified version of Waxman's algorithm [Ref11] that ensures that the network is connected and has a pre-specified number of nodes, links, maximum number of neighbors per node, and maximum number of adjacencies per node. The rectangular grid resembles the continental U.S.A. with maximum one-way propagation delay of 30 ms in the East-West direction and maximum one-way propagation delay of 15 ms in the North-South direction. We consider two different network sizes as explained in Section 3.

The network has a flat, single-area topology.

Each node is a Router and each link is a point-to-point link connecting two routers.

We assume that node CPU and memory (not the link bandwidth) is the main bottleneck in the LSA flooding process. This will typically be true for high speed links (e.g., OC3 or above) and/or links where OSPF traffic gets an adequate Quality of Service (QoS) compared to other traffic.

Different Timers:
  LSA refresh interval = 1800 seconds,
  Hello refresh interval = 10 Seconds,
  Router-Dead interval = 40 seconds,
  LSA retransmission interval: two values are considered, 10 seconds
    and 5 Seconds (note that a retransmission is disabled on the
    receipt of either an explicit acknowledgment or a duplicate LSA
    over the same interface that acts as an implicit acknowledgment)
  Minimum time between successive generation of the same LSA = 5
    seconds,
  Minimum time between successive Dijkstra SPF calculations
    is 1 second.

Packing of LSAs: It is assumed that for any given node, the LSAs generated over a 1-second period are packed together to form an LSU but no more than 3 LSAs are packed in one LSU.

LSU/Ack/Hello Processing Times: All processing times are expressed in terms of the parameter T. Two values of T are considered, 1 ms

and 0.5 ms.

In the case of a dedicated processor for processing OSPF packets the processing time reported represents the true processing time. If the processor does other work and only a fraction of its capacity can be dedicated to OSPF processing then we have to inflate the processing time appropriately to get the effective processing time and in that case it is assumed that the inflation factor is already taken into account as part of the reported processing time.

The fixed time to send or receive any LSU, Ack or Hello packet is T. In addition, a variable processing time is used for LSU and Ack depending on the number and types of LSAs packed. No variable processing time is used for Hello.
Variable processing time per Router LSA is (0.5 + 0.17L)T where L is the number of adjacencies advertised by the Router LSA. For other LSA types (e.g., ASE LSA or a "Link" LSA carrying traffic engineering information about a link), the variable processing time per LSA is 0.5T.

Variable processing time for an Ack is 25% that of the corresponding LSA.

It is to be noted that if multiple LSAs are packed in a single LSU packet then the fixed processing time is needed only once but the variable processing time is needed for every component of the packet.

The processing time values we use are roughly in the same range of what has been observed in an operational network.

LSU/Ack/Hello Priority: Two non-preemptive priority levels and three priority scenarios are considered. Within each priority level processing is FIFO with new packets of lower priority being dropped when the lower priority queue is full. The higher priority packets are never dropped.

In Priority scenario 1, all LSUs/Acks/Hellos received at a node are queued at the lower priority.

In Priority scenario 2, Hellos received at a node are queued at the higher priority but LSUs/Acks are queued at lower priority.

In Priority scenario 3, Hellos and Acks received at a node are queued at the higher priority but LSUs are queued at lower priority.

All packets generated internally to a node (usually triggered by a timer) are processed at the higher priority. This includes the initial LSA storm, LSA refresh, Hello refresh, LSA retransmission and new LSA generation after detection of a failure or recovery.

Buffer Size for Incoming LSUs/Acks/Hellos (lower priority): Buffer size is assumed to be 2000 packets where a packet is either an Ack, LSU, or Hello.

LSA Refresh: Each LSA is refreshed once in 1800 seconds and the refresh instants of various LSAs in the LSDB are assumed to be uniformly distributed over the 1800 seconds period, i.e., they are completely unsynchronized. If however, an LSA is generated as part of the initial LSA storm then it goes on a new refresh schedule of once in 1800 seconds starting from its generation time.

LSA Storm Generation: As defined earlier, "LSA storm" is the simultaneous or near simultaneous generation of a large number of LSAs. In the case of only Router and ASE LSAs we normally assume that the number of ASE LSAs in the storm is about 4 times that of the Router LSAs, but the ratio is allowed to change if either the Router or the ASE LSAs have reached their maximum possible value. In the case of only Router and Link LSAs (carrying traffic engineering information) we normally assume that the number of Link LSAs in the storm is about 4 times that of the Router LSAs, but the ratio is allowed to change if either the Router or the Link LSAs have reached their maximum possible value. For any given LSA storm we keep generating LSAs starting from Node index 1 and moving upwards and stop until the correct number of LSAs of each type have been generated. The LSAs generated at any given node is assumed to start at an instant uniformly distributed between 20 and 30 seconds from the start of the simulation. Successive LSA generations at a node are assumed to be spaced apart by 400 ms. It is to be noted that during the period of observation there are other LSAs generated besides the ones in the storm. These include refresh of LSAs that are not part of the storm and LSAs generated due to possible link failures and subsequent possible link recoveries.

Failure/Recovery of Links: If no Hello is received over a link (due to CPU/memory congestion) for longer than Router-Dead Interval then the link is declared down. At a later time, if Hellos are received then the link would be declared up. Whenever a link is declared up or down, one Router LSA is generated by each Router on the two sides of the point-to-point link. If "Link LSAs" carrying traffic engineering information is used then it is assumed that each Router would also generate a Link LSA. In this case it is also assumed that due to rerouting of LSPs, three other links in the network (selected randomly in the simulation) would have significant change in reserved bandwidth which would result in one Link LSA being generated by the routers on the two ends of each such link.

3. Simulation Results

In this section we study the relative performance of the three

Priority scenarios defined earlier (no priority to Hello or Ack, priority to Hello only, and priority to both Hello and Ack) with a range of Network sizes, LSA retransmission timer values, LSA types, processing time values and Hello/Router-Dead-Interval values:

Network size: Two networks are considered. Network 1 has 100 nodes, 1200 links, maximum number of neighbors per node is 30 and maximum number of adjacencies per node is 50 (same neighbor may have more than one adjacencies). Network 2 has 50 nodes, 600 links, maximum number of neighbors per node is 25 and maximum number of adjacencies per node is 48. Dijkstra SPF calculation time for Network 1 is assumed to be 100 ms and that for Network 2 is assumed to be 70 ms.

LSA Type: Each node has 1 Router LSA (Total of 100 for Network 1 and 50 for Network 2). There are no Network LSAs since all links are point-to-point links and no Summary LSAs since the network has only one area. Regarding other LSA types we consider two situations. In Situation 1 we assume that there are no ASE LSAs and each link has one "Link" LSA carrying traffic engineering information (Total of 2400 for Network 1 and 1200 for Network 2). In Situation 2 we assume that there are no "Link" LSAs and half of the nodes are ASA-Border nodes and each border node has 10 ASE LSAs (Total of 500 for Network 1 and 250 for Network 2). We identify Situation 1 as "Link LSAs" and Situation 2 as "ASE LSAs".

LSA retransmission timer value: Two values are considered, 10 seconds and 5 seconds (default value).

Processing time values: Processing times for LSUs, Acks and Hello packets have been previously expressed in terms of a common parameter T. Two values are considered for T, which are 1 ms and 0.5 ms respectively.

Hello/Router-Dead-Interval: It is assumed that Router-Dead interval is four times the Hello interval. In one case it is assumed that Hello interval is 10 seconds and Router-Dead-Interval is 40 seconds (default values), and in the other case it is assumed that Hello interval is 2 seconds and Router-Dead-Interval is 8 seconds.

Based on Network size, LSA type and processing time values we develop 6 Test cases as follows:

Case 1: Network 1, Link LSAs, retransmission timer = 10 sec.,
        T = 1 ms, Hello/Router-Dead-Interval = 10/40 sec.

Case 2: Network 1, ASE LSAs, retransmission timer = 10 sec.,
        T = 1 ms, Hello/Router-Dead-Interval = 10/40 sec.

Case 3: Network 1, Link LSAs, retransmission timer = 5 sec.,

T = 1 ms, Hello/Router-Dead-Interval = 10/40 sec.

Case 4: Network 1, Link LSAs, retransmission timer = 10 sec., T = 0.5 ms, Hello/Router-Dead-Interval = 10/40 sec.

Case 5: Network 1, Link LSAs, retransmission timer = 10 sec., T = 1 ms, Hello/Router-Dead-Interval = 2/8 sec.

Case 6: Network 2, Link LSAs, retransmission timer = 10 sec., T = 1 ms, Hello/Router-Dead-Interval = 10/40 sec.

For each case and for each Priority scenario we study the network stability as a function of the size of the LSA storm. The stability is determined by looking at the number of non-converged LSUs as a function of time. An example is shown in Table 1 for Case 1 and Priority scenario 1 (No priority to Hellos or Acks).

LSA STORM SIZE	Number of Non-Converged LSUs in the Network at Time(in sec)								
	10s	20s	30s	35s	40s	50s	60s	80s	100s
100 (Stable)	0	0	24	29	24	1	0	1	1
140 (Stable)	0	0	35	48	46	27	14	1	1
160 (Unstable)	0	0	38	57	55	40	26	65	203

Table 1: Network Stability Vs. LSA Storm
(Case 1, No priority to Hello/Ack)

The LSA storm starts a little after 20 seconds and so for some period of time after that the number of non-converged LSUs should stay high and then come down for a stable network.
This happens for LSA storms of sizes 100 and 140. With an LSA storm of size 160, the number of non-converged LSUs stay high indefinitely due to repeated retransmissions, link failures due to missed Hellos for more than the Router-Dead interval which generates additional LSAs and also due to subsequent link recoveries which again generate additional LSAs. We define network stability threshold as the maximum allowable LSA storm size for which the number of

non-converged LSUs come down to a low level after some time. It turns out that for this example the stability threshold is 150.

The network behavior as a function of the LSA storm size can be categorized as follows:

(1) If the LSA storm is well below the stability threshold then the CPU/memory congestion lasts only for a short period and during this period there are very few retransmissions, very few dropped OSPF packets and no link failures due to missed Hellos. This type of LSA storms are observed routinely in operational networks and networks recover from them easily.

(2) If the LSA storm is just below the stability threshold then the CPU/memory congestion lasts for a longer period and during this period there may be considerable amount of retransmissions and dropped OSPF packets. If Hello packets are not given priority then there may also be some link failures due to missed Hellos. However, the network does go back to a stable state eventually. This type of LSA storm may happen rarely in operational networks and they recover from it with some difficulty.

(3) If the LSA storm is above the stability threshold then the CPU/memory congestion may last indefinitely unless some special procedure for relieving congestion is followed. During this period there are considerable amount of retransmissions and dropped OSPF packets. If Hello packets are not given priority then there would also be link failures due to missed Hellos. This type of LSA storm may happen very rarely in operational networks and usually some manual procedure such as taking down adjacencies in heavily congested nodes is needed.

(4) If Hello packets are given priority then the network stability threshold increases, i.e., the network can withstand a larger LSA storm. Furthermore, even if the network operates at or somewhat above this higher stability threshold, Hellos are still not missed and so there are no link failures. So even if there is congestion in the control plane due to increased retransmissions requiring some special procedures for congestion reduction, the data plane remains unaffected.

(5) If both Hello and Acknowledgement packets are given priority then the stability threshold increases even further.

In Table 2 we show the network stability threshold for the five different cases and for the three different priority scenarios defined earlier.

Case Number	Maximum Allowable LSA Storm Size For: No Priority to Hello or Ack	Priority to Hello Only	Priority to Hello and Ack
Case 1	150	190	250
Case 2	185	215	285
Case 3	115	127	170
Case 4	320	375	580
Case 5	120	175	225
Case 6	185	224	285

Table 2: Maximum Allowable LSA Storm for a Stable Network

4. Observations on Simulation Results

Table 2 shows that in all cases prioritizing Hello packets increases the network stability threshold, and in addition, prioritization of LSA Acknowledgment packets increases the stability threshold even further. The reasons for the above observations are as follows. The main sources of sustained CPU/memory congestion (or positive feedback loop) following an LSA storm are (1) LSA retransmissions and (2) links being declared down due to missed Hellos which in turn causes further LSA generation and future recovery of the link causing even more LSA generation.

Prioritizing Hello packets avoids and practically eliminates the second source of congestion. Prioritizing Acknowledgements significantly reduces the first source of congestion, i.e., LSA retransmissions. It is to be noted that retransmissions can not be completely eliminated due to the following reasons. Firstly, only the explicit Acknowledgments are prioritized but duplicate LSAs carrying implicit Acknowledgments are still served at the lower priority. Secondly, LSAs may get greatly delayed or dropped at the input queue of receivers and therefore Acknowledgments may not even get generated in which case prioritizing Acks would not help. Another factor to keep in mind is that since Hellos and Acks are prioritized, the LSAs see bigger delay and potential for dropping. However, the simulation results show that on the whole prioritizing Hello and LSA Acks are always beneficial and significantly improve the network stability threshold.

Our simulation study also showed that in each of the cases, instead of prioritizing Hello packets if we treat any packet received over a link as a surrogate for a Hello packet (an implicit Hello) then we get about the same stability threshold as obtained with prioritizing Hello packets.

If we prioritize Hello packets then even when the network operates somewhat above the stability threshold, links are not declared down due to missed Hellos. This implies that even though there is control plane congestion due to many retransmissions, the data plane stays up and no new LSAs are generated (besides the ones in the original storm and the refreshes)

5. Need for Prioritized Treatment of Critical OSPF Packets and Special Marking to Facilitate That

The observations in the previous section clearly show that prioritizing Hello and LSA Acknowledgment packets are greatly beneficial in improving the scalability and stability of large networks. In addition to these packets it may be beneficial to treat certain other OSPF packets at the higher priority as well. One example (during the database exchange process between neighbors following a link recovery) is the Database Description packet from a slave that is used as an acknowledgment for the previous Database Description packet sent from the master. Another example is an LSA carrying a change information which may trigger SPF calculation and rerouting of Label Switched Paths. It is preferable to transmit this information faster than other LSAs in the network that are just once-in-30-minutes refreshes and typically would not trigger any route computation or route change.

Given that there is a need for providing prioritized treatment to certain OSPF packets, the next natural question is how to facilitate this prioritization.

If it is possible to
examine the packet header (for the purpose of prioritization) much faster than processing the whole packet then prioritized treatment is possible without any protocol changes.

However, we also propose that a special marking be used for categorizing all OSPF packets into one of two priority classes. It is also important to separately mark OSPF packets from other IP packets. One way to do this is to reserve two diffserv

codepoints, one for higher priority OSPF packets and another one for lower priority OSPF packets. With this special marking it would be easy for OSPF implementers to treat Hello, LSA acknowledgment, and other critical OSPF packets at a higher priority and thereby significantly improve the scalability and stability of networks using OSPF.

6. Summary

In this draft we point out that the node processors of a large network may be subjected to a sustained CPU/Memory congestion as a result of a large LSA storm caused by some type of failure/recovery of nodes/links or synchronization among refreshes. There is a certain LSA storm size threshold above which the network may show unstable behavior caused by large number of retransmissions, link failures due to missed Hello packets and subsequent link recoveries. Using a simulation study we show that the LSA storm size causing instability may be substantially increased by providing prioritized treatment to Hello and LSA Acknowledgment packets. Furthermore, if we prioritize Hello packets then even when the network operates somewhat above the stability threshold, links are not declared down due to missed Hellos. This implies that even though there is control plane congestion due to many retransmissions, the data plane stays up and no new LSAs are generated (besides the ones in the original storm and the refreshes).

Based on the above observations we propose the following:

(1) Process the Hello packets at a higher priority compared to other OSPF packets. In order to facilitate this, explicitly mark the Hello packets, to differentiate them from other OSPF packets. One way of special marking is to use a different Diffserv codepoint for Hello packets compared to other OSPF packets.

(2) In the absence of special marking, or in addition to it, use other mechanisms in order not to miss Hello packets. One example is to treat any packet received over a link as a surrogate for a Hello packet (an implicit Hello) for the purpose of keeping the link alive. Our simulation study shows that this mechanism is just as effective as explicitly prioritizing Hello packets.

(3) The same type of explicit marking and prioritized treatment may be beneficial to other OSPF packets as well. One important example is LSA acknowledgment packet that can reduce

retransmissions during periods of congestion. Our simulation study shows that prioritization of both Hello and LSA Acknowledgment packets is considerably more effective than just prioritizing Hello packets. Other examples include (a) Database description (DBD) packet from a slave that is used as an acknowledgement, and (b) LSAs carrying intra-area topology change information.

It is possible that some implementations are already using one or more of the above mechanisms in order not to miss the processing of critical packets during periods of congestion. However, we suggest the above mechanisms to be included as part of the standard so that all implementations can benefit from them.

7. Acknowledgments

We would like to acknowledge Jerry Ash, Margaret Chiosi, Elie Francis, Jeff Han, Beth Munson, Roshan Rao, Moshe Segal, Mike Wardlow, and Pat Wirth for collaboration and encouragement in our scalability improvement efforts for Link-State-Protocol based networks.

8. References

[Ref1] Pappalardo, D., "AT&T, customers grapple with ATM net outage," Network World, February 26, 2001.

[Ref2] "AT&T announces cause of frame-relay network outage," AT&T Press Release, April 22, 1998.

[Ref3] Cholewka, K., "MCI Outage Has Domino Effect," Inter@ctive Week, August 20, 1999.

[Ref4] Jander, M., "In Qwest Outage, ATM Takes Some Heat," Light Reading, April 6, 2001.

[Ref5] C. Alaettinoglu, V. Jacobson and H. Yu, "Towards Milli-second IGP Convergence," Work in Progress.

[Ref6] A. Zinin and M. Shand, "Flooding Optimizations in Link-State Routing Protocols," Work in Progress.

[Ref7] J. Moy, "Flooding over Parallel Point-to-Point Links," Work in progress.

[Ref8] P. Pillay-Esnault, "OSPF Refresh and flooding reduction in

stable topologies," Work in progress.

[Ref9] G. Choudhury, V. Manral, "LSA Flooding Optimization Algorithms and Their Simulation Study," Work in progress.

[Ref10] J. Ash, G. Choudhury, V. Sapozhnikova, M. Sherif, A. Maunder, V. Manral, "Congestion Avoidance & Control for OSPF Networks", Work in Progress.

[Ref11] B. M. Waxman, "Routing of Multipoint Connections," IEEE Journal on Selected Areas in Communications, 6(9):1617-1622, 1988.

9. Authors' Addresses

Gagan L. Choudhury
AT&T
Room D5-3C21
200 Laurel Avenue
Middletown, NJ, 07748
USA
Phone: (732)420-3721
email: gchoudhury@att.com

Vera D. Sapozhnikova
AT&T
Room C5-2C29
200 Laurel Avenue
Middletown, NJ, 07748
USA
Phone: (732)420-2653
email: sapozhnikova@att.com

Anurag S. Maunder
Sanera Systems
370 San Aleso Ave.
Second Floor
Sunnyvale, CA 94085
Phone: (408)734-6123
email: amaunder@sanera.net

Vishwas Manral
NetPlane
189, Prashasan Nagar,
Road Number 72
Jubilee Hills, Hyderabad
India

email: Vishwasm@netplane.com