Virtual Link: An Enabler of Enterprise Utility Computing

Krishna Kant

Intel Corporation, krishna.kant@intel.com

Abstract. Dynamically provisioned virtual clusters provide a means of consolidating servers in a data center and for supporting utility computing. Data centers typically sport a large number of (layer 2) switches and very few routers, yet, the existing layer-2 QoS are not well developed. This paper proposes the notion of *virtual link* as an interconnection abstraction to provide granular QoS. The paper also presents an experimental study comparing virtual link based congestion control against other alternatives for emerging 10 Gb/sec Ethernet links. It is shown that virtual links can provide desired capabilities with a small perturbation to existing standards and can work well in mixed legacy environments.

1 Introduction

Utility computing refers to the notion of treating the entire data center as a pool of resources (computes, storage, special functions, etc.) which can be assigned dynamically to various applications as needed. Utility computing almost demands cluster capable applications and is greatly helped by a single unified fabric over which resources of various sorts can be accessed efficiently. We shall call the set of nodes allocated to a given application as a "virtual cluster". The obvious advantage of the consolidation is the increase in server utilization, which is often found to be in 5-10% range in current data centers. The ability to grow or shrink individual virtual clusters can be used to adapt physical resources to dynamically changing application needs and to minimize power consumption.

A unique feature of commercial data centers is that *most of the interconnect devices in a data center are (layer 2) switches, rather than (layer 3) routers.* In fact, smaller networks may not even have any routers except at the edges. The main reasons for this are low cost, lower latency, and almost zero configuration effort, for switches. Given such an environment, *layer3 QoS features (e.g., diffserv, intserv, etc.) are inadequate within a data center.* Furthermore, a blind implementation of these features at layer 2 is undesirable as it would take away the advantages of switches over routers. For example, setting up DSCP parameters is known to be very tricky [6] and requires detailed knowledge of the flows. Instead, our goal here is to define simpler mechanisms that can be largely automated.

A *virtual cluster* can be thought of as the realization of a clustered application and includes the *virtual nodes* (VN's) on which the application runs, and the

virtual paths (VP's) over which these VN's communicate. We call the node as virtual since it could well be implemented via a virtual machine running on a physical node. A virtual path between VN's can be further viewed as a sequence of one or more *virtual links* (VL), where a VL is defined to span only a *layer 2 domain*. [An L2 domain is the set of layer 2 devices (switches) delimited by layer 3/4 devices (routers & servers)]. Thus, a VP results by stitching together VL's at the intervening routers, if any. Note that the communication between VN's that belong to the same physical node may use some efficient local mechanism and is not addressed here.

The outline of the paper is as follows. Section 2 discusses related work in the field. Section 3 discusses the support required for establishing, tearing down and using virtual links. Section 4 discusses the QoS, congestion control and reconfiguration issues related to virtual link. Finally, section 5 compares virtual link based congestion control against endpoint control.

2 Related Work

Virtual LAN (VLAN) is a standardized mechanism (802.1q/p) for segmenting an Ethernet network into "islands" such that traffic from of VLAN is not accessible to another VLAN. The flows in each VLAN can be differentiated based on the 3-bit CoS (*class of service*) field in the extended Ethernet header. VLANs are inadequate for providing virtual cluster abstraction since they are intended to be static and do not provide any congestion control mechanisms.

The IEEE task force on Ethernet congestion management, known as 802.1ar, is currently examining ways of improving congestion notification and management [4]. The main objectives of this effort are to enable switches to mark packets and allow endpoint layer-2 to do 802.1x type link flow control at the level of individual "virtual pipes" or CoS classes. Adding an ECN like feature [7] at layer2 that TCP can exploit has also been considered here. These capabilities are certainly helpful in supporting the virtual link concept addressed in this paper.

The label switched paths (LSPs) defined for the well known MPLS (multi-protocol label switching) scheme provide virtual communication channels that can pack a high degree of sophistication in terms of traffic engineering [1]. For example, an extension of RSVP, called RSVP-TE, can be used for reserving resources on LSPs (RFC3209). This helps to automate the setup. However, a direct implementation of these layer-3 features would make the switches too complex and expensive, and is not desirable.

The extension of Ethernet to metro distances needs to deal with several issues including inadequacy of 4096 VLANs, scalability of broadcast procedures, and, QoS and congestion control [2]. One method to provide QoS over MAN areas is to make use traffic engineered MPLS paths and then run Ethernet protocol on top of this. An alternative approach is to extend Ethernet frame format slightly via *VLAN stacking*, also known as *Q-in-Q* mechanism. The former scheme is unsuitable for data centers; the latter scheme can be used, but will require some further extensions as discussed in section 3.1.

3 Supporting Virtual Links

Supporting virtual links requires some additional features which are listed below and discussed in subsequent subsections.

1. A pair of send & receive queues to which one or more virtual links can be mapped. We call these as *virtual queue pairs* (VQP) or simply VQ's.
2. Every packet needs to carry additional information to use the appropriate queues.
3. A signaling mechanism to setup, teardown and update virtual links.
4. A mechanism to convey switch congestion to the L2 boundary and local propagation and handling at higher (i.e., L3 or L4) level.

3.1 Virtual Link Queues

Virtual link queues are necessary to provide isolation between VL's and to control scheduling policies. Currently, the only queuing differentiation available in the Ethernet is via the IEEE 802.1p/q standard, which adds a 4-byte TCI (tag control info) field to Ethernet frames. Fig.1(a) shows the Ethernet-II frame format and Fig.1(b) shows the details of TCI field. As stated earlier, the CoS bits are intended for traffic prioritization and are similar to IP layer ToS (type of service) bits. CoS bits provide for only 8 queues, which is inadequate in the VL context. Also, the purpose of CoS bits is prioritization (e.g., giving control messages – such as the signaling messages that we will introduce – higher priority over others), and not really appropriate for VL application.

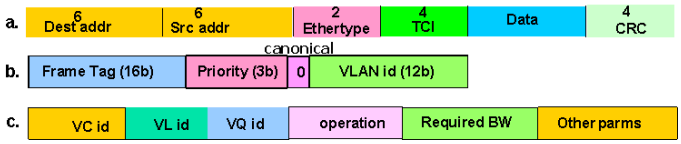

Fig. 1. a) Ethernet-II frame format, b) 802.1 header, c) Signaling msg format

The L2 Ethernet frames associated with IP and other higher layer protocols should be able to convey the queue id to the switches on the path. This can be done either by carrying the VQ id directly (say, 4-5 bits) or by carrying both VL and VC id's which can be mapped to VQ id's locally at each switch. Although the latter scheme is lot more flexible, it requires many more bits in the Ethernet frame. Finding extra bits in Ethernet frames is quite challenging w/o perturbing the standards substantially; therefore, we henceforth assume that VQ id's are carried directly.

Several approaches are possible to convey VQ id in Ethernet frames, but all of them have some impact on standards. One simple idea is to designate, say, 64 high end VLAN bit patterns (out of a total of 4096) for 32 VQ id's leaving 1 bit worth of information for congestion indication use. This along with canonical bit (unused currently) can satisfy congestion indication requirements. Within a data

center, the number of VLANs is generally quite limited and thus the reduction in the number of possible VLANs is not an issue. The more serious issue is the potential use of high end VLAN bit patterns in real data centers. A somewhat different idea is to exploit the Q-in-Q type of encoding used in metro Ethernet, which basically adds additional Ethertype and TCI fields in the frame [2] (See Fig 1(a)). As stated earlier, these fields are used for transportation of frames between LAN segments. This encoding does require minor firmware updates to the switches. However, if we want to use the outer TCI bits for VQ-id and congestion indication, we will have to create yet another outer Ethertype so that the scheme does not conflict with metro Ethernet usage. The main advantage of doing so is that we now gain 4 bytes in each frame, which can be used to carry VL or VC ids, if we so desire. The down side is higher processing overhead and reduction of maximum data length by another 6 bytes.

The VQ id can be used by switches to implement a variety of scheduling mechanisms, including weighted round robin (WRR) for dividing available BW between competing flows. This is straightforward and not discussed any further.

3.2 Signaling Support for VL's

The signaling protocol is needed to provision VC's automatically since a manual provisioning is a recipe for non-use. The signaling requires a new Ethernet frame type similar to the one in Fig.1(a) but with a different Ethertype value.

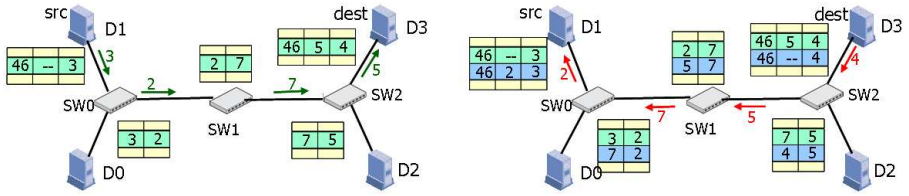

Fig. 2. Illustration of VL setup in a L2 domain: a) Forward, b) Backward

Fig 1(c) shows the generic format for signaling messages using this new ethertype. Here, the operation field indicates VL operations, including VL_initf, VL_initb and VL_ack for VL setup and a few others for VL teardown and parameter update. The field "other parms" include additional information that may be necessary for setting up queue thresholds.

Virtual links can now be setup by sending a special message, say *VL_initf*. This message starts with the local VQ's number and wiggles its way through the switches on the path to the L2 end. At each switch, it sets up the queue translation table (QTT). Fig. 2(a) illustrates this process for setup of a VL (with id 46) from server D1 to D3 via switches 0-2. To start with, layer 2 of D1 creates the table entry (46, –, 3) indicating that VQ 3 is used for VL 46 at this node. It then sends a VL_initf message to SW0. SW0 locally allocates a VQ id (say, 2) and creates a table entry (3,2) indicating that what comes with VQ 3 must go into VQ 2. The table will also store the VC id, requested BW, and other QoS

parameters (if any), but these are not shown for simplicity. SW0 then changes the VQ id in the VL_initf message to 2 and forwards it to SW1. This process continues until the message reaches D3, which too sets up its table entry. This table entry (46, 5, 4) says that a packet for VL 46 coming with VQ id 5, will be placed in VQ 4. Note that *the dynamic allocation of VQ ids makes it easy to allocate only as many queues as are really required.*

The operations in Fig. 2(a) only take care of forward VL setup. When D3 receives this message, it needs to echo a *VL_initb* message towards D1, which effectively does the same thing in the other direction. Fig. 2(b) shows the tables at the end of complete setup. For simplicity, we have numbered send & receive VQ's identically, but this is not essential. Since messages can be lost, we also need some handshake mechanism (e.g., like the one in TCP or SCTP) to recover. The details of this are straightforward and will be omitted. Finally, we also need signaling messages to tear down virtual links and to adjust their parameters.

The above setup procedures are not claimed to be unique – ATM, Frame Relay, and most significantly MPLS LDP all use a similar scheme (with minor variations of the theme). In fact, it is possible to do VL setup by extending MPLS LDP protocol so that the switches examine the LDP messages [1]. We do not follow this approach to avoid the need for MPLS capabilities.

The setup scheme can be extended to establish virtual paths, by successively establishing virtual links across routers. The main difficulty here is the need to reestablish VL's as the routing table entries change. For lack of space we do not discuss the details here.

3.3 Scalability and Reconfiguration Issues

Conceptually, it is nice to use a distinct virtual path for every VN to VN communication; however, this can quickly become unscalable. In this section, we discuss issues related to limited usage of VL's.

To start with, we note that the main motivation for VL's is to isolate flows corresponding to *different* virtual clusters. Thus, if two or more VL's of a given VC happen to pass through a switch, they should all use the same virtual queue at this switch. Although this can be enforced easily in the VL setup procedure given in the last section, it is possible to simplify things even further. Note that the end result of setting up all VL's of a VC is to reserve a queue at each switch port encompassed by this VC. This can be done trivially by reaching all VN's of the VC from any given node.

The establishment of virtual paths across routers could get rather complex and may impact layer3. Fortunately, if the application is configured properly, the inter-router traffic should be smaller and less latency sensitive than the traffic within L2 domains. In this case, we can forego keeping any VL distinctions for paths that cross router boundaries. Or, such paths can be aggregated into a small set of "pipes" that exploit diff-serv and other IP level QoS features. For example, all IPC traffic may go through one pipe, all storage through another, etc. The main attraction of this approach is that it limits the scope of VL's and thus enhances scalability w/o and substantial performance implications.

The idea of aggregating multiple VL's into a smaller number of "pipes" or classes can be taken further to enhance scalability. In particular, the queue pairs are established based on the characteristics of applications running in the VC's; and thus the number of simultaneous queues is limited by the number of such characteristics identified.

Let us now briefly address the issue of dynamic reconfiguration of virtual clusters. We assume that any traffic flow that doesn't use VL concept is routed via VQ 0. The addition of a node to a VC is straightforward and will allocate new queues only at new switches/routers that are used by this VC. However, deletion of nodes from a VC must ensure that VQ's are not deleted until they become completely unused. This can be addressed easily via a reference count type of scheme.

4 Layer 2 Congestion Control

Any discussion of QoS is incomplete w/o examining congestion control issues since, for the most part, QoS is relevant only during congestion scenarios. The Ethernet standard only provides the 802.1x (so called Xon/Xoff) flow control. Unfortunately, this scheme applies to the entire link and does not provide flow control at the level of individual flows. Thus, packets may be dropped for individual flows. A reliable transport protocol such as TCP or SCTP will react to packet drops and reduce flows; however, *dropping packets in a data center environment is highly undesirable* because of high data rates, bursty traffic, and long latencies suffered by retransmitted packets. It follows that we need some mechanism in switches to explicitly convey congestion situation to the endpoints.

A workable layer 2 congestion control scheme in switches must support two basic functions: (a) Congestion detection and feedback to layer 2 edge, and (b) Congestion control at or above layer2 edge. In the following, we discuss these aspects briefly and then show that the congestion control can be simplified by using virtual links.

Congestion detection is best done via queue thresholds which need to be set judiciously. The VL signaling mechanism can be used for setting the thresholds. The threshold crossing at a switch can be carried to the L2 edge in many ways, as discussed in [4]. The basic schemes include implicit feedback (i.e., forward or backward packet marking), explicit feedback (sending feedback packets from endpoints), or mixed. Implicit schemes are generally preferred since they do not increase traffic during congestion; however, they require additional bits in the packets.

The congestion feedback mechanism discussed above brings the feedback only to the L2 edge. How this feedback is used depends on various congestion control options as illustrated in Fig. 3. In particular, bringing the feedback from a router to the endpoint may either be via ECN (as a result of backpressure on the router), or explicitly via ICMP message. The consequences of latter mechanism are not considered here due to lack of space.

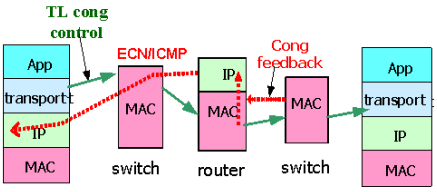

Fig. 3. Congestion Feedback and control

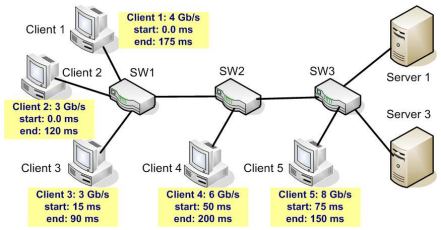

Fig. 4. Full Test Network

5 Experimental Study of L2 Congestion Control

In this section, we compare virtual link based congestion control against other alternatives in order to exhibit the pros and cons of virtual link based congestion QoS and congestion management.

5.1 Congestion Control Mechanisms

For simplicity, let's limit discussion to applications that use primarily a reliable connection oriented communication mechanism (e.g., TCP or SCTP). Now, if no VL support is available, the congestion will eventually manifest itself as packet loss followed by the transport reaction to it. (We used TCP-Reno in these experiments.) This is the baseline scenario studied here and is designated as L4-loss.

A slightly enhanced scheme is to enable the switches to report congestion via the ECN mechanism, except that it is implemented at layer 2. We further assume that TCP will examine these layer-2 ECN bits as well and take the same action as with layer-3 ECN bits. We call this the L4-ECN method.

Climbing up the feature ladder, we assume switches can mark packets for congestion but don't do any traffic differentiation. We still assume an appropriate signaling procedure to set congestion thresholds at switches. The idea now is to do an intelligent flow control at the endpoint NIC and thereby enforce proper BW allocation to various flows. We call this scheme as L2-FC. A concrete example of such and scheme is explored in [3, 5].

Here we describe the scheme in [5] only briefly. Initially, each L2 endpoint sends out probes (or special signaling messages) to discover paths to all other endpoints of this L2 domain. All these paths are then constantly monitored for congestion at the endpoint. The feedback scheme is the *mixed feedback* discussed earlier. In particular, a switch on the path updates the congestion indicator if its current congestion level is higher than the one in the probe. Via this mechanism, each L2 endpoint is able to maintain the maximum congestion level along each path. This congestion level along with the desired weight (or relative BW) for various flows is used to do a bang-bang control for forcing the congestion along the path down to a predefined nominal value.

The final scheme studied is the VL based congestion control, which we call L4-VL. Although this scheme requires features (a)-(f), the congestion control is

still at the endpoint TCP level (hence the name L4-VL). This scheme requires very little support from the endpoint NIC and does not require a sophisticated congestion control to be built into the NIC. On the other hand, the scheme provides no support for throttling UDP flows.

5.2 Congestion Control Performance

In this section we provide a detailed experimental comparison of the congestion performance the schemes L4-Loss, L4-ECN, L2-FC and L4-VL. For the comparisons, we used the OPNET simulation package which provides comprehensive implementations of all relevant networking layers (MAC, IP, TCP, ...) and prebuilt models of many commercial switches and routers. It also provides a few application layers (e.g., database, FTP, VOIP, etc.). Yet substantial development work was required in order to implement the following features: a) switch level traffic differentiation, b) Endpoint layer 2 flow control, and c) Application level flow control. The simulated network is shown in Fig. 4. It is important to note here that the "clients" in Fig. 4 are not really the end-clients (usually outside the Enterprise), but rather other servers (e.g., mid-tier servers making DB requests) residing within the Enterprise.

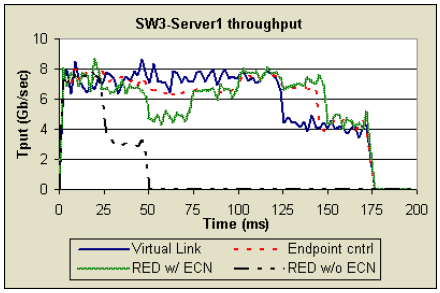

Fig. 5. Throughput of VC1 client 1

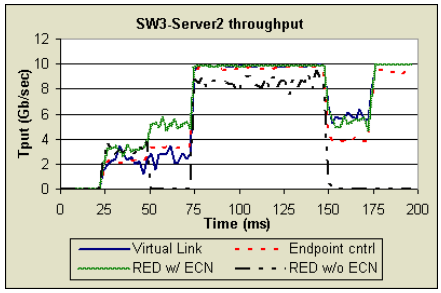

Fig. 6. Throughput of VC1 client 2

All links in Fig. 4 follow the IEEE 10 Gb/s standard. This is done to emphasize the emerging high speed data center environment. The physical cluster here is divided up into 2 virtual clusters:

VC1: This includes clients 1 & 2 and server1. Here both clients 1 and 2 generate database traffic over TCP.

VC2: This includes clients 3, 4 & 5 and server2. Each of these clients also generates database traffic over TCP.

In both cases, the traffic is 100% database updates which means that the congested flow direction is from client to server. (This scenario was chosen for simplicity; the mechanisms do work well irrespective of the direction of congestion.) The update sizes are assumed to be exponentially distributed with a mean of 8KB. The interarrival times of clients are uniformly distributed with maximum value twice that of the minimum value. The mean traffic driven by each client plus the start and stop times of each client are shown in Fig. 4. The

start and stop times are staggered so that we can have a number of overload scenarios.

Figs 5 and 6 show, respectively, the achieved throughput for VC1 and VC2 respectively. Here the intent is to give 2/3rd of the BW to VC1 and 1/3rd to VC2. In other words, we expect VC1 to receive 6.67 Gb/s throughput under stress conditions.

We start with the traffic evolution for VC1 by referring to Fig 5. For the first 15 ms, only clients 1&2 are on, and together drive 7 Gb/s. Not surprisingly, this traffic is carried properly in all cases. At 15 ms, client 3 comes on and the total traffic driven over SW1-SW2 link is 10 Gb/s. Without the ECN (case L4-Loss) the highest traffic source (Client 1) experiences heavier losses than others and effectively shuts down. As a result, the VC1 throughput drops down to 3 Gb/s (Client 2 rate) for this case. This type of "shut-down" scenario was observed consistently in many situations and points to the inadequacy of depending on just the packet losses. In contrast, ECN is still capable of controlling the backlog effectively (case L4-ECN), though not quite as well as cases L4-VL & L2-FC.

At time 50ms, client 4 comes on. The total BW driven through SW2-SW3 link is now 16 Gb/s and we are under severe congestion. Case L4-Loss now experiences a connection reset due to too many retransmission timer expiries. Case L4-ECN still survives but now shows its deficiency – w/o any differentiation, TCP will simply tend to equalize BW of all the congested sources. As a result, both VC1 and VC2 will achieve 5 Gb/s BW. Both cases L4-VL and L2-FC maintain close to 2:1 throughput ratio between the two VCs in this case, as required, however, there is some difference in their performance. In particular, case L4-VL (virtual link) tends to favor VC1 a bit whereas case L2-FC (endpoint control) favors VC2 somewhat. Note that the control in case (L4-VL) is more variable because it is just TCP driven as opposed to case L2-FC which does additional layer 2 flow-control.

At time 75ms, client 5 also comes on. Since this simply adds to the existing overload, no change is expected. Surprisingly, however, case L4-ECN shows an increase in VC1 throughput! To understand this, notice that until time 75ms, the link from SW3 to Server2 was not saturated, but now it does get overloaded. Consequently, TCP connections at clients 3-5 all back off hard (more so at client 3) and this allows for higher VC1 throughput.

At time 90 ms, client 3 goes off. This has no impact on case L4-VL but both cases L2-FC and L4-ECN have a throughput increase because of less VC2 traffic. In fact, almost the entire VC1 traffic is able to get through in all three cases primarily because Client 4 continues to remain mostly shutout due to congestion on SW3-Server2 link. At time 120 ms, Client 2 also goes off, thereby reducing VC1 rate down to 4 Gb/s. Client 5 then goes off at time 150 ms, but it does not affect VC1 traffic (because client 5 traffic has little interference with it). Finally, at time 175 ms, VC1 traffic turns off completely.

Let us now briefly examine VC2 throughput in Fig 6. The behavior here is in some ways complementary to that in Fig 5, since a favoring of VC1 implies a disfavoring of VC2 and vice versa. The only point worth noting is that at time

175 ms, when VC1 traffic turns off completely, VC2 traffic actually surges to fill the link because of the earlier backlog.

6 Conclusions

The main conclusion from the above and several other studies is that switch level congestion detection and marking are essential for an acceptable performance. A TCP level congestion control driven by this marking (L4-ECN) can control the congestion but is unable to provide the desired QoS. Finally, both L2-FC and L4-VL schemes can provide decent congestion control and QoS. However, the two have somewhat different characteristics. The L4-VL scheme requires more perturbation to the switch infrastructure, but does require embedding sophisticated flow control in the NICs. The L2-FC scheme can also suffer from scalability issues in large L2 networks.

A crucial consideration in proposing new features for existing networks is compatibility with legacy implementations. With the L2-FC scheme, a legacy NIC will (a) ignore any congestion feedback and (b) will not even participate in the probing done by the newer NICs. The consequence of (a) is eventual TCP-level action based on packet losses. The consequence of (b) is imprecise accounting of paths and path flow control at the newer NICs and hence inaccurate control. In the L4-VL scheme, a non-VL capable switch will not understand signaling messages and would simply pass them on along the path. The normal packets also will not get any differentiation at these switches but will be silently forwarded along. Thus, if the legacy switches are not the sources of severe congestion, the whole system will continue to perform well. If the legacy switches do experience congestion, the flows through them will be governed by the default TCP behavior (i.e., TCP's tendency to equalize flows).

Acknowledgements: The author would like to thank Gary Mcalpine for assistance in detailed implementation and experimentation. Thanks are also due to Raj Ramanujan for discussions relating to some of the ideas in the paper.

References

1. "QoS Support in MPLS networks", MPLS/Frame Relay alliance whitepaper, May 2003.
2. G. Chiruvolu, An Ge, et. al., "Issues and approaches on extending Ethernet beyond LANs", IEEE Computer, March 2004, pp 79-86.
3. G. Mcalpine, M. Wadekar, et. Al., "An architecture for congestion management in Ethernet clusters", IEEE IPDPS Workshop 9, April 2005, Denver, CO.
4. H. Barraas, M. Wadekar, et. al., "Problem Space for Ethernet Congestion Management", IEEE 802.1ar Congestion Management Group Presentation, sept 2004.
5. G. Mcalpine, "Congestion Management for Switched Ethernet", Proc. of high perf. interconnects for distributed computing, July 2005.
6. S.H. Low, F. Paganini, et. al., "Linear stability of TCP/RED and a scalable control", Computer Networks, vol 43, no 5, pp633-647, 2003.
7. http://www.icir.org/floyd/ecn.html (collection of annotated references on ECN).