Virtual Link: An Enabler of Enterprise Utility Computing

Krishna Kant
Intel Corporation

***Abstract*— Dynamically provisioned virtual clusters provide a means of consolidating servers in a data center and for supporting utility computing. In this paper we propose the notion of *virtual link* as an interconnection abstraction for virtual clusters which provides a mechanism for implementing QoS and congestion management at layer 2. The paper examines a variety of issues associated with virtual links including signaling, interfacing with layer 3 mechanisms, congestion feedback, congestion control, and reconfiguration. These mechanisms are essential to make Ethernet the unified fabric of choice in emerging enterprise computing environments.**

Keywords: virtual clusters, virtual links, unified fabric, data center, QoS, diffserv, autonomic control, MPLS.

I. Introduction

Modern data centers have grown to be large networked environments with thousands of servers and tens to hundreds of different applications. Most current data centers use SMPs (symmetric multiprocessors) for large applications like databases, centralized SAN (system area network) storage (usually Fiber Channel based), and, if needed, specialized fabrics for low-latency inter-process communication (IPC). However, there is a strong emerging trend from this traditional picture towards a cluster computing environment built around Ethernet as a *unified fabric* that carries all traffic types. (See [3] for more complete discussion on motivations for this trend.) Recent advancements such as 10 Gb/sec Ethernet data rates, HW offload and user mode implementation of TCP/IP, iSCSI (SCSI protocol over TCP/IP), etc. are the main drivers for this.

A large physical cluster running multiple clustered applications can be divided up into a number of dynamically reconfigurable *virtual clusters* (VC's), one for each application. The nodes in a virtual cluster could be either physical nodes or virtual nodes (VN's). A virtual node is defined via the virtual machine (VM) running on the physical node. The virtual nodes of a VC are connected via some virtual communication channels, which we explore in this paper. Such a virtual cluster model can yield several advantages over a static subdivision of the physical cluster or direct sharing of all physical resources, as discussed in section II. The notion of *utility computing* [6] essentially requires virtual cluster type of ideas, but at a larger and wider scale. The growing interest in deploying Ethernet at the MAN level (*Metro Ethernet*) further advances the case for utility computing [2]. Conversely, the mechanisms discussed in this paper are directly relevant to metro ethernet as well.

In this paper, we concentrate on the virtual communication channels and call them as either *virtual paths* (end to end) or *virtual links* (within a layer 2 domain). The paper is primarily concerned with what form the virtual paths and links should take and explore the corresponding QoS, congestion control and reconfiguration issues.

The outline of the paper is as follows. Section II motivates the need for dynamically reconfigurable virtual clusters. Section III defines the concepts of virtual links and paths, discusses required features and signaling for them, and addresses scalability and reconfiguration issues. Section IV discusses the QoS, congestion control and reconfiguration issues related to VL. Section V then compares virtual link based congestion control against endpoint control. Finally, section VI concludes the paper.

A. 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 VLAN flows can be differentiated based on the 3-bit CoS (class of service) field. However, VLANs are inadequate for providing virtual cluster abstraction since (a) VLANs don't comprehend virtualization, (b) VLANs are supposed to be static and manually configured entities, and (c) VLANs don't provide any means of BW provisioning and congestion control.

The label switched paths (LSPs) defined for MPLS provide virtual communication channels that can pack a high degree of sophistication in terms of traffic engineering. For example, an extension of RSVP, called RSVP-TE, has been standardized for reserving resources on LSPs. This helps to automate the setup. However, a direct implementation of layer 3 features such as MPLS, diff-serv, RSVP, etc. in layer 2 switches would significantly add to cost, latencies, management complexities, etc.

The extension of ethernet to metro distances needs to deal with some of the same issues as data centers, namely QoS and congestion control. However, the proposed schemes in this regard use traditional mechanisms such as DSCP, CoS, ethernet over traffic engineered MPLS paths (martini encapsulation), etc. The main emphasis here appears to be sophistication rather than simplicity & autonomic setup.

II. Why Dynamic Virtual Clusters?

Surveys of commercial data centers invariably reveal that the average utilization of servers, storage devices and network

infrastructure is typically in 5-10% range. There are a variety of reasons for this, including:

1) Peak demand based capacity allocation. This is often driven by highly unpredictable load and huge costs of lost business during peaks seasons (e.g., Christmas rush).
2) "Other" workloads. Servers in a data center don't just run live workload; in fact, many more servers are often dedicated to supporting various development and testing activities. The utilization of these servers is understandably low and highly bursty and helps bring down the average utilizations substantially.
3) Need for isolation. Many business functions (e.g., customer support, account management, etc.) need comparatively little resources, but are still installed on separate servers to avoid multiple services being down due to a single failure. Machine crashes/hangs are routine with development/testing activities and those servers must be isolated from one another and from the live workload.

The notion of dynamic virtual clusters can significantly reduce the installed equipment (and hence increase its utilization) while still catering to unpredictible workload and isolation requirements. It is important to note here that the motivation for reduced equipment is not just equipment cost, less equipment also implies lower costs of management, upgrade, patching, repair, space, power consumption and cooling. Naturally, automation is the key to realizing any savings since any kind of manual intervention to setup & reconfigure virtual clusters will significantly negate its advantages.

Although the idea of dynamically expanding and contracting the nodes on which an application runs is very old, the virtual cluster notion has several new elements. First of all, a virtual cluster is a customer level entity and may be used for a dynamically changing set of applications. Second, the notion of virtual nodes (whereby virtual clusters share the physical nodes and links) is a new twist. Even here, the level of sharing could vary greatly. For example, given the emerging multiple core machines along with HW support for virtualization, certain HW resources (e.g., cores, functional units, MACs, etc.) may be physically partitioned (perhaps dynamically) while others are shared. Third, with power and cooling issues becoming ever larger areas of concern in large data centers, shutting down unused or poorly used VN's, physical nodes and even the network infrastructure can pay huge dividends in terms of power/cooling related costs. Of course, a good management should even significantly reduce the number of servers needed and hence the numerous headaches (patching, upgrade, etc.).

One other point to note here is that the physical (and hence virtual) nodes are not necessarily general purpose compute nodes, they could well provide specialized services such as access to iSCSI based or object based storage, XML or SSL accelerators, etc. Such nodes may involve a high degree of sharing (e.g., many virtual clusters using specialized VN's carved out of one or a few real specialized nodes).

A deeper look into the virtual cluster usage model reveals a natural hierarchy. At the first level, a large physical cluster may be divided up into VC's "owned" by different customers. In this case, the customer may be provided certain SLA's in terms of both computing capacity and network bandwidth. The virtual cluster belonging to a customer could further host multiple virtual sub-clusters to support the usual data-center tiering (front-end, mid-tier and backend). Each tier could then be further subdivided to isolate individual applications.

Although such an hierarchy is interesting, it requires a detailed knowledge of resource requirements at each level and therefore impractical. In particular, while it is reasonable to provision some fixed inter-node BW in a customer-level VC (based on what the customer wants or is willing to pay) or for a VC used for a specific purpose (live workload, development, etc.), having such a knowledge about individual applications of a customer is not. Consequently, we consider only one level of virtual clusters along with the implicit assumption that VC BW requirements are known (or can be reasonably estimated).

III. Virtual Paths and Links

A *virtual path* can be regarded as an abstract communication channel that connects any two nodes of a virtual cluster. A virtual path consists of sequence of one or more *virtual links*, each of which spans a *layer 2 domain*. We define L2 domain as the set of layer 2 devices (switches) delimited by layer 3/4 devices (routers & servers).[1] In the following subsections, we motivate virtual links, discuss layer2 queuing & scheduling support for them, define a signaling protocol and consider scalability issues.

A. Why Virtual Links?

One major characteristic of data center networks is that *most of the interconnect devices are (layer 2) switches, rather than (layer 3) routers*. In fact, small data centers may not even use any routers except at the periphery. The main reasons for this are low cost, lower latency, and almost zero configuration effort, for switches. In contrast, routers can be expensive and pose a management nightmare due to access control, diff-serv, filtering, and related setups. The deployment of metro ethernet will extend the reign of layer 2 switches even over MAN distances.

Given such an environment, a virtual link, as defined above, is a fundamental concept. When a VN to VN path involves routers, a sequence of virtual links can be stitched together at the routers to form virtual paths. Such a stitching should automatically adapt to any routing table changes without L3 being even aware of presence of virtual links.

In order to support the virtual link concept, the link layer in the switches needs to provide some additional features including:

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.

[1]For clarity, we henceforth consider switches as strictly layer2 devices.

4) Layer 2 congestion feedback to the L2 boundary, which may require additional L2 messages and/or feedback bits in regular packets.
5) In order to be scalable, it should be possible to use the same queue for multiple virtual links.

Provision of these features is discussed in next few subsections. Although many of these features can be provided using extensions of label distribution protocol (LDP) of MPLS, we shall deliberately avoid a close tie-in with MPLS since (a) many paths have no router at all, and (b) routers may not be MPLS capable.

B. 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 three CoS (class of service) bits for traffic prioritization. These bits are defined to be compatible with the IP layer ToS (type of service) bits thereby allowing them to be a direct copy of ToS bits; however, there is no requirement that things be done this way. In any case, this provides for at most 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.

The number of VQ's required is directly determined by the number of virtual links that share a single physical Ethernet segment. In most cases, a few VL's might suffice, however, larger numbers may make sense for shared storage access or special services (e.g., XML or SSL accelerators). Also, in the typical blade server environment, we might want to put every L2→L2 communication on a separate VL, which may need 16 VL's per real link. Thus, with an assumption of 16-32 VQ's, 4-5 bits are required for queue index specification.

The signaling mechanism required to setup the queues can use arbitrary amount of information since signaling will be done via special frames; however, the bigger concern is the number of bits needed in regular packets to specify VL specific queuing. In this regard, we can consider the following 2 mechanisms:

1) Packets simply carry VQ id (4-5 bits), however, since the queue id assigned to a given VL can change at each intermediate device, the VQ bits will have to be changed at intermediate switch. Such a scheme requires more signaling work but fewer bits in regular packets.
2) Packets carry VL and VC id's which are mapped to appropriate VQ's id's at each switch.[2] Such a scheme makes it easy to assign multiple virtual links to the same queue; however, the price is paid in terms of more bits per regular packet.

Let us now consider the required scheduling algorithms for the queues. To support CBR-like traffic, it is necessary to provide constant rate scheduling. For all other traffic types, a weighted round-robin (WRR) or a similar scheme may be attractive since it will automatically divide available BW in proportion to the specified VL BW requirements. In fact, by combining the two policies, one could enforce some minimum committed rate for premium traffic and proportional division for others. It is surely possible to implement lot more sophisticated schemes, but their practical utility may be questionable.

[2]For generality, we assume explicit VL ids distinguish between various VL's of a given VC. This may not be necessary.

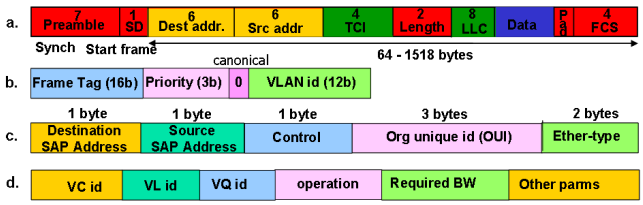

Fig. 1. a) 802.3 frame format, b) 802.1 header, c) 802.2 SNAP header, d) Signaling msg format

C. Signaling Support for VL's

The signaling protocol is essential to ensure that VC's can be provisioned and dynamically reconfigured w/o any manual intervention. The first issue in signaling is to define a new ethernet frame type that does not perturb any existing messaging. Fig 1(a) shows the format of 802.3 ethernet frame including 802.1, 802.2 and data components. The 802.1 component is represented by TCI (tag control info), which is shown in detail in Fig 1(b). It includes frame type (2 bytes), 3 COS bits, and 12 VLAN bits. The 802.2 part is shown in Fig 1(c) and includes the so-called SNAP (subnetwork access protocol) header. SNAP is introduced primarily for enforcing compatibility with older ethernet formats; for regular ethernet frames source & destination SAP addresses are 0xAA, Control=1, and OUI=0. There are plenty of available Ethertype values for new frame types; one such value can be designated for frames carrying signaling messages.

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 for setting up queue thresholds, as discussed in section IV-B.

As stated in the last section, regular messages must carry either the VQ id (4-5 bits) or VL/VC id (many more bits). In addition, at least 1-2 bits are required for congestion feedback. Finding space for these is challenging without perturbing current formats. For this reason, we henceforth concentrate only on VQ based approach. One low-impact possibility is to use 802.1 frame type, which currently has only one value 0x8100. Repurposing priority bits for VQ indication is undesirable since they are needed within each VC. Another approach 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 space for congestion indication use. This along with canonical bit (unused currently) can satisfy congestion indication requirements.

Virtual links can now be setup by sending a VL_initf message with the VQ at the L2 endpoint as the starting VQ parameter value. The message wiggles its way through and sets up the queue translation table (QTT) at each intermediate L2

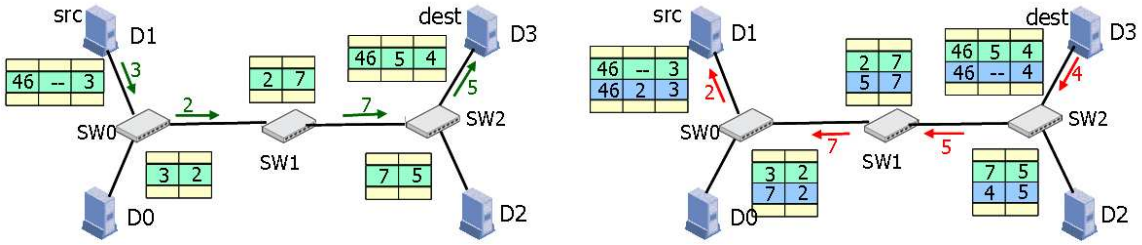

Fig. 2. IIlustration of VL setup in a L2 domain: a) Forward, b) Backward

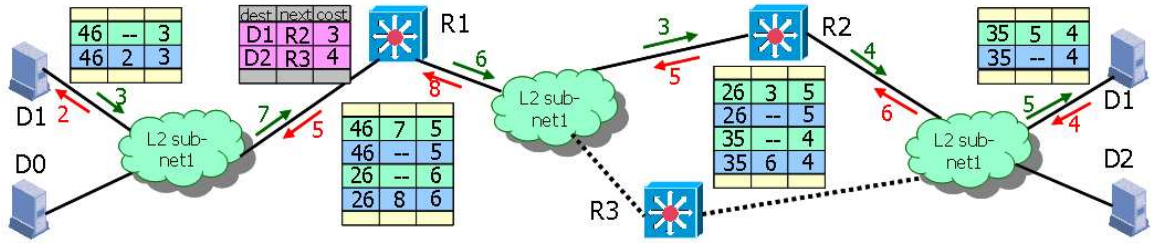

Fig. 3. IIlustration of VL setup across routers: a) Forward, b) Backward

device. 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. Finally, to ensure that the remote end knows of successful VL setup, D1 needs to send a confirmation message, say VL_ack to D3. As usual, this message can also be combined with data transfer. Since messages may be lost, both ends need to run timers and repeat messages if the timer expires (with the understanding that no duplicate entries are made in the table). Another issue to handle is that a setup message encounters a switch with inadequate available BW to continue setup. In this case, a VL_reject message needs to be propagated along with necessary cleanup.

The above setup procedure is 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.

Fig. 3 shows a more complex scenario with multiple routers in the path from D1 to D3. Here, we have three L2 domains, and a separate VL needs to be setup in each domain. When the VL_initf message reaches router R1, the local layer 2 not only sends a VL_initb backwards, but also continues the setup forwards by choosing a new VL id (26) *assuming that a usable VL does not exist already*. For this, the L2 needs to know the next hop chosen by L3 (via the current routing table shown next to the router). The rest of the setup will proceed routinely. If the routing table changes later on, the L2 can trigger setup of a new VLs (e.g., R1→R3 and R3→D3). The old VL's (e.g., R1→R2, R2→D3) can be either torn down at this point, or retained as "inactive" to allow for a quick switch-over in case the routing path reverts back to the original. In such a case, the eventual teardown procedure must be able to do "batch teardown" of VL's. Again, MPLS can be used between routers to handle these issues.

D. Scalability Issues for Virtual Links

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. That is, all VL's of a given VC that happen to pass through a switch can (and should) use the same VQ – distinct queues are required only on a per-VC basis. This can drastically reduce the need for switch queues and expensive setup procedures. One way to think about this is that we still establish VL's as usual, however, the chosen queue at each L2 is merely a function of VC id. That is, if a VQ already exists for a VC, a new one is not created. This works since the switches do store the VC id associated with a queue.

Although we can continue to think of all VL's being established one by one, all we want is to establish a VQ for each switch port used by a VC. This can be done trivially

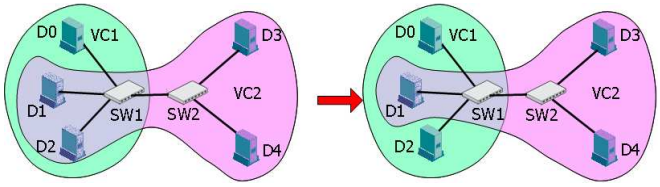

Fig. 4. Virtual Cluster Reconfiguration

by reaching all nodes from any given node. However, as discussed in section IV-B, the setting up of congestion thresholds autonomically requires a more elaborate procedure.

Let us now reconsider the VP setup procedures illustrated in Fig. 3. With new VQ's being allocated only on a per VC basis, the number of VQ's allocated at routers are limited by the number of VC's that cross router boundaries. Given a topology aware VC configuration procedure and the fact that routers are typically significantly outnumbered by switches in a data center, the VL tables at routers should stay fairly small even when inactive VL entries are retained. Thus scalability should not be an issue even in large clusters.

Nevertheless, there might be a case for foregoing VL setups to routers completely. For example, if the application is configured carefully, the inter-router traffic may be smaller and less latency sensitive than traffic within a L2 domain. Another possibility is the use of MPLS and diff-serv features available at router with setups done via RSTP-TE or similar protocols. In these cases, any communications where the next hop is a router can simply pass through a default VQ at the switches, say VQ 0. The main attraction of this approach is that it completely eliminates any need for consulting or tracking routing tables.

E. Dynamic Reconfiguration Issues

Dynamic reconfiguration of virtual clusters brings in a host of issues such as what parameters to monitor, when to trigger reconfiguration, how to select the nodes to be included or excluded, and how to adjust the virtual links. In this paper, we consider only the last issue.

We assume that any traffic flow that doesn't use VL concept is routed via VQ 0. Besides this, the number of VQ's is maintained equal to number of VC's that have traffic flowing through the device in question. 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. Fig. 4 shows a reconfiguration scenario where originally, nodes D0-D2 belong to VC1 and nodes D1-D4 to VC2. Now, when node D2 is removed from VC2, it initiates a VL teardown message towards all other nodes. However, since VC2 queues at SW1 & SW2 are still required for access to node D1, no queue deletion should actually happen. A simple way to handle this problem is to maintain a *reference count* for each VQ at a switch. The reference count gives the number of virtual nodes belonging to the corresponding VC that are connected to this switch. A teardown could then simply decrement this number and remove the queue when the count reaches 0.

IV. Layer 2 Congestion Control

Given the abundance of (layer 2) switches in the data center, layer 2 congestion control is essential in the data centers. Yet, the 802.3 standard does not provide any decent congestion control mechanism. The only available mechanism is Xon/Xoff flow control, which applies to the entire link and thus cannot control individual flows (including virtual links).

The only other mechanism for dealing with congestion is packet drop if the switch buffer space runs low. 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 in data centers because of high data rates, bursty traffic, and long latencies suffered by retransmitted packets. Furthermore, losses may force the transport to go through the software "slow path" (instead of the hardware accelerated "fast path"). It follows that we need some mechanism in switches similar to ECN (explicit congestion notification) at routers, so that losses can be prevented.

A workable layer 2 congestion control scheme in switches must support 3 distinct aspects: (a) *congestion detection*, (b) *congestion feedback* to layer 2 edge, and (c) *congestion control* at or above layer2 edge. In the following, we discuss these aspects briefly and then consider how virtual links can help.

A. Congestion Feedback

Congestion detection is typically done via queue thresholds which need to be set judiciously (discussed later). The threshold crossing at a switch can be carried to the L2 edge in one of the following ways: (a) Explicit, (b) Implicit, and (c) Mixed. The *explicit feedback* refers to a switch preparing a special ethernet frame with the necessary congestion information and sending it upstream to the L2 edge. *Implicit feedback*, in contrast, refers to the marking of regular packets by the congested switch. (This is like ECN marking at the routers). Finally, the *mixed feedback* refers to the case where the downstream L2 edge periodically sends feedback packets upstream and these packets are marked by the congested switches. (This is kind of like the RM cells in ATM.)

The explicit feedback has the distinct advantage in terms of being able to convey whatever feedback information is desired (e.g., address of the congested switch, congestion severity, offending source/destination IP address, etc.). However, feedback packets necessarily require better QoS treatment at intermediate switches (via CoS bits), could make the congestion worse, and may overwhelm the L2 edge that has to examine them and take action. For this reason, generally, explicit feedback is not a good idea.

Implicit feedback is very efficient; furthermore, it automatically summarizes congestion information along the path to the L2 edge. However, implicit feedback can reasonably be expected to convey only a few bits worth of information. Implicit feedback can be forward or backward. The *forward marking* is like the ECN mechanism [8] where the downstream packets are marked and the L2 edge receiving them is responsible for turning the marking around for packet going in the opposite direction. Forward marking requires 2 bits in regular

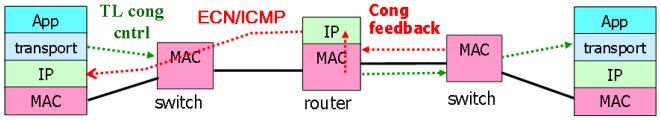

Fig. 5. Congestion Feedback and control

packets and could be problematic if the traffic is primarily unidirectional. In *backward marking*, the switch directly marks the upstream packets. Backward marking requires more intelligence in switches, but reduces feedback delay and needs only one bit in regular packets. In the interests of keeping the switches simple, we believe that forward marking is the better approach.

The mixed feedback attempts to achieve advantages of both explicit and implicit schemes. Although the issue of additional traffic remains, it is possible to control the frequency of feedback very well.

B. Congestion Detection

In general, the threshold setting, and hence congestion detection, can happen at three levels: entire switch, per port, and per VQ of a port. Switch level threshold refers to threshold for the total available buffer space in the switch. In the interests of efficient utilization of available memory, switches often do not dedicate fixed amounts of buffer space for each port; instead a centralized buffer pool is shared among all ports. Let L denote the switch level threshold, L_i the threshold for port i, and L_{ij} the threshold for VQ j at port i. Of course, L must set less than total available memory M; however, $\sum_{\forall i} L_i$ exceeds L and may even exceed M. Similarly, $\sum_{\forall j} L_{ij} > L_i$ for any i. The idea, obviously, is to allow some queues to temporarily build up while others remain lean.

Although the general principles for setting thresholds are well known, extensive experience with SS7, ATM, and IP contexts indicates that they can be notoriously difficult to set properly. Instead, we use the following simple procedure: the signaling messages used in VL setup provide an estimate of the delay at each switch from the time of forward marking until the traffic can be throttled. This estimate multiplied by the VC BW provides an estimate of how far to back up the threshold from the buffer size. It is possible to adjust the threshold later if necessary (via an update message).

C. Congestion Response

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. 5. Here the congestion at SW2 is reported to the router layer 2, whereas the congested path extends further back via SW1 into the endpoint D1. The congestion control can be done here either at the L2 edge, or at the endpoint D1. The two types of control have different requirements and characteristics and are discussed below.

L2 edge flow control is at layer 2 and therefore appropriate in the VC context. The VL concept with a committed BW allows us to throttle the flow to the extent of provisioned BW until no congestion feedback arrives for some period (say,

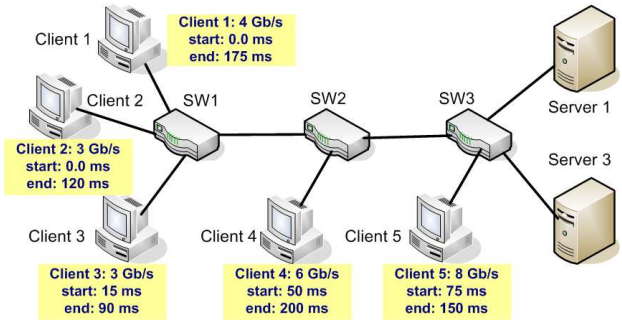

Fig. 6. Full Test Network

a few seconds). The consequence of layer 2 flow control is obviously the backpressure that percolates to the higher layers. Thus, at a router, IP level congestion will eventually occur and can be propagated via ECN to the endpoint in case of TCP. Unfortunately, UDP flows will still be restrained (if at all) by application level mechanisms triggered due to packet drops by the router.

Fig.5 hints at two other congestion control schemes as well. First, it is possible to forego L2 flow control and instead propagate the congestion feedback up the stack to IP (at a router) or to transport (at an endpoint). However, in this case, ECN is not the right way to propagate the congestion to endpoint even for TCP, since the congestion isn't at the router, but somewhere further downstream! This distinction along with some indication of the congested flows out of the router must be indicated to the transport endpoint so that it throttles only the relevant flows (TCP or UDP). This can be accomplished by sending an ICMP source quench message as an explicit L3 feedback. As before, explicit feedback carries its risks in terms of additional traffic during congestion periods, however, it can provide more intelligent control.

A detailed comparison of all these possibilities is beyond the scope of this paper. Instead, here we consider networks with just one L2 domain and address the need for switch level traffic differentiation.

V. VL vs. Endpoint Control

The proposals in this paper will require switches to support the following features:

(a) Congestion threshold setting & detection at various levels.
(b) Congestion feedback to the L2 edge from switches.
(c) Congestion control at L2 edges (or at a higher layer).
(d) Multiple queues per port to support virtual links, and
(e) Layer 2 signaling for VL setup, teardown and updates.

While (a)-(c) are unavoidable for congestion control, it is important to critically examine the need for (d)-(e). In particular, an alternative approach is to do endpoint control only and avoid any traffic differentiation at the switches. Reference [7] proposes such an endpoint control scheme. The main attraction of this approach is less support at switches and consequently no need to allocate VQ bits in regular packets. In the following, we discuss the pros and cons of the two approaches and compare their performance.

For brevity, we will not discuss the details of the endpoint control technique in [7]. The basic idea is as follows: Initially,

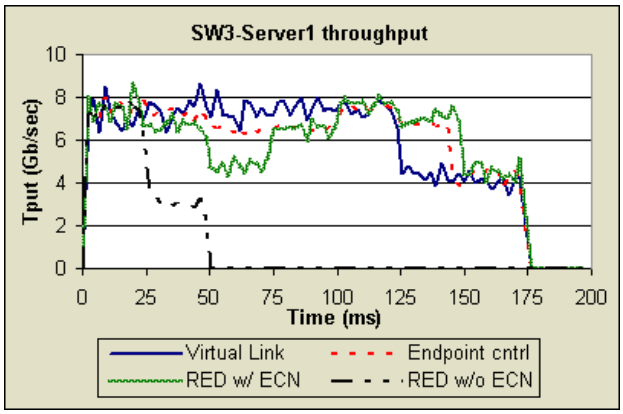

Fig. 7. Throughput of VC1 client 1

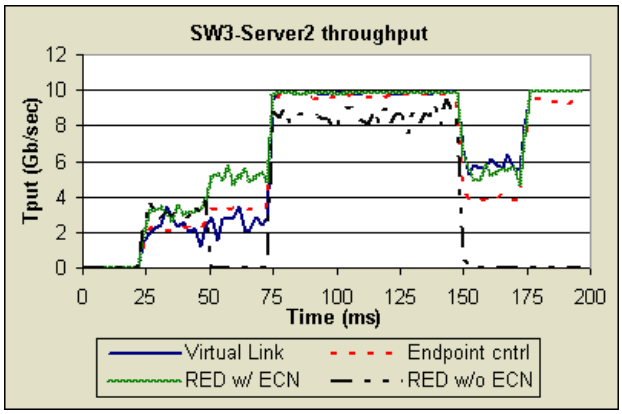

Fig. 8. Throughput of VC1 client 2

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 via feedback messages that are marked by intermediate switches by their maximum congestion level. The measured congestion level is then used to constrain the flow along the path just enough so that the congestion level drops to a predefined nominal value.

It is not the intent of this paper to delve into the details of the endpoint flow control procedure. Here we simply want to show performance of such a control against the virtual link based control.

Fig. 6 illustrates the test cluster used for the results reported here. All links in this network were chosen to be 10 Gb/s Ethernet in order to emphasize the emerging 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 each of which again generated 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. The update sizes are assumed to be exponentially distributed with a mean of 8KB. The interarrival times of clients are always 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. 6. The start and stop times are staggered so that we can have a number of overload scenarios.

The simulations were done by using OPNET which provides a comprehensive implementation of all network layers plus a few application layers (e.g., database). Yet substantial modifications were required in order to implement a) switch level traffic differentiation, b) Endpoint layer 2 flow control, and c) application level flow control. It is important to note here that the network in Fig. 6 attempts to make a good use of available features in the OPNET; there is no suggestion that actual end-clients (outside the the Enterprise) are the ones actually being models. In fact, the “clients” here should be regarded as merely other servers in the Enterprise.

Figs 7 and 8 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. The results are shown for 4 different situations:

A Virtual links (blue curve). The switches are set up with WFQ queuing with 2:1 weights for VC1 and VC2.

B Endpoint Control (red curve). Again the weights are set up for 2:1 BW division between VC1 and VC2.

C No differentiation at switches, however, the switches do provide an ECN-like packet marking mechanism.

D Similar to case C, except that there is no ECN-like marking at the switches.

The main point of including (C) and (D) is primarily to show their inadequacy since one cannot expect proper BW subdivision w/o any differentiation. We shall first follow the traffic evolution for VC1 by referring to Fig 7. Note that we expect 6.7 Gb/s throughput for VC1 under stress conditions. 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 and the total traffic driven over SW1-SW2 link is 10 Gb/s. Without the ECN (case D) 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. ECN is still capable of controlling the backlog effectively (case C), though not quite as well as cases A & B.

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 (D) now experiences a connection reset due to too many retransmission timer expiries. Case (C) 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 A and B 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 A (virtual link) tends to favor VC1 a bit whereas case B (endpoint control) favors VC2 somewhat. Note that the control in case (A) is more variable because it is just TCP driven as opposed to case (B) 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 (C) shows an increase in VC1 throughput! To understand this, notice that until time 75ms, SW3-Server2 link 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 (A) but both cases (B) and (C) 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. Case A immediately reduces VC1 throughput down to 4 Gb/s but case (B) continues at higher throughput for some time. This difference is due to slightly different treatment of VC1 and VC2 traffics by virtual link and endpoint control cases. 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 8. The behavior here is in some ways complementary to that in Fig 7, 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.

The main conclusion from this and several other studies is that both virtual link and endpoint control can work reasonably well. The virtual link concept requires additional queue setup and traffic differentiation in the switches, but the ultimate flow control is still driven by TCP. In the endpoint control, we have an additional layer of flow control including the probing all paths from the endpoint and managing flows on them intelligently. In large switched networks, the endpoint control may have scalability issues, whereas virtual link provides a more distributed approach to managing traffic. The virtual link based approach is also more tolerant of mixed environments where the flow control capable devices are mixed with legacy devices. In particular, 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, so long as non-differentiated switches are not the sources of severe congestion, the whole system will continue to perform well.

VI. Conclusions

In this paper we studied the concept of virtual links as a means of supporting virtual clusters in the enterprise and possibly spread across MAN distances (assuming metro ethernet based implementation). The virtual link concept requires traffic differentiation, congestion detection and propagation support in the switches, which has been largely missing from the Ethernet standard. It is shown that the virtual link concept can maintain scalability, simple implementation, and assist in virtual cluster reconfiguration decisions. It was also shown that the virtual link based congestion management works effectively, simply and can be introduced gradually. Such a support is essential to make Ethernet a true unified data center fabric for the emerging enterprise utility computing environments.

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] K. Kant and N. Jani, “SCTP performance in data center environments”, Proc. of SPECTS, Philadelphia, PA, July 2005.

[4] K. Kant, “Coordinated BW control in Data Centers”, to appear in IEEE/ACM. workshop on Grid Computing, Seattle, WA, Nov 2005.

[5] T. Shanley, *Infiniband Network Architecture*, Mindshare Inc., 2002.

[6] M. Kallahalla, M. Uysal, et. al., “SoftUDC: A software based data center for utility computing”, Special issue of IEEE Computer on Internet data centers, Nov 2004 (Eds. K. Kant & P. Mohapatra).

[7] G. Mcalpine, ”Congestion Management for Switched Ethernet”, submitted for publication.

[8] http://www.icir.org/floyd/ecn.html (collection of annotated references on ECN).