Research Article Journal of Computer Science Applications and Information Technology Open Access

Received: February 23, 2017
Accepted: March 03, 2017
Published: March 13, 2017

# Intelligent Transport Layer for Bandwidth as a Service in Cloud and IoT Applications

Stan McClellan[1]*, Wuxu Peng[2] and Anthony Amaro[2]

[1]Ingram School of Engineering, Texas State University San Marcos, TX, USA
[2]Department of Computer Science, Texas State University, San Marcos, TX, USA

***Corresponding author:** Stan McClellan, Director, Ingram School of Engineering, Texas State University, 601 University Dr, RFM 5200D, San Marcos, TX 78666, USA, Tel: 512-245-1826; Fax: 512-245-7771; E-mail: stan.mcclellan@txstate.edu

## 1 Abstract

Cloud and IoT applications employ a wide range of advanced hardware and software techniques. This paper focuses on utilization of network bandwidth, and proposes the concept of "Bandwidth as a Service" (BaaS) between devices in cloud and IoT environments. Conventional controls for bandwidth are discussed, and a novel algorithm is proposed for dynamically shaping bandwidth into percentage groups containing individual transport-level connections. We show that this approach maximizes bandwidth efficiency while maintaining limitations on network resources. An analysis of bandwidth performance between virtual machines is discussed along with a survey of throughput efficiency and improvements within the hypervisor. The performance of the proposed algorithm using both Xen Bridge and Open vSwitch is also discussed and compared. The proposed algorithm can be used as BAAS modules controlled by individual VM tenants in a cloud datacenter, or it can be embedded/tailored within the algorithms proposed in related research work.

## 2 Keywords:

*Bandwidth Guarantees; Cloud; IoT; Networks; Data Center; Cloud Tenants; Cloud Service; IoT Application;*

## 3 Introduction

Cloud computing has emerged as a dominant model of delivery of computing services, where resources and infor- mation are provided to users over the cloud network [1]. The services offered by cloud computing providers utilize virtualized resources such as storage, service platforms, and computational processing [2]. Cloud hosting companies pro- vide these resources by charging usage-based pricing, where cloud tenants pay for used resources [3]. These resources are used for the various different service capabilities of the Cloud, such as Infrastructure as a Service, where tenants run virtual machines to execute customer business logic; Platform as a Service, where tenants are provided an API (Application Program Interface) to develop cloud-based applications; and Software as a Service, where end-user software is provided to the tenant and executed in the cloud [2,4]. Cloud services are consumed by cloud tenants, and those services are implemented by virtual machines running within physical hosts. Tenants may or may not be aware of the presence of the virtual machines, depending on the cloud platform, however virtual machines are necessary to carry out the provided cloud service on behalf of the tenant. In this context, cloud tenants may either be end users (humans), or virtual machines also within the cloud. There are many physical host machines within a cloud data center, all of which serve multiple tenants and provide many cloud-based services.

### 3.1 IoT vs Cloud Computing

Internet of Things (IoT) is claimed as the third wave of the Internet [5]. It emerged as another wave of Internet enabled applications and is focused on enabling automation and monitoring/controlling of a wide variety of everyday household and industrial devices/embedded systems that have not traditionally been Internet-enabled.

Initially, IoT appeared to have little overlap with cloud computing. However with the increasingly dominant presence of cloud computing and extension of IoT applications, the two are beginning to converge [6]. Fueling this convergence are the virtual networks that underlies cloud computing and large scale IoT applications. There are strong reasons to believe that IoT will utilize the ubiquity of the cloud computing infrastructure. For example, home security monitoring is considered one of the typical IoT applications. Some manufacturers of home monitoring devices also provide cloud-based security video storage and notification services. Naturally, surveillance data are stored in their enterprise cloud and are accessible through cloud Apps.

## 3.2 Bandwidth Management

While virtual machine resources such as memory, pro- cessing power, and storage space may be specified by the business arrangement between tenants and cloud providers, the underlying physical network is typically shared by all tenants and respective virtual machines (see Figure 1). Due to many tenants utilizing the physical network without regulation, network utilization by some tenants may negatively impact available network bandwidth for other tenants within the cloud. As a result, bandwidth utilization in the cloud may be unpredictable [6]. Additionally, the unpredictable nature of tenant virtual machines usage of network resources allows the possibility of a few virtual machines disproportionately affecting network throughput of other tenants [7].

Static bandwidth limits do not take into account the bandwidth requirements by the connections hosted within the cloud, which lead to inefficient utilization of bandwidth resources [8].

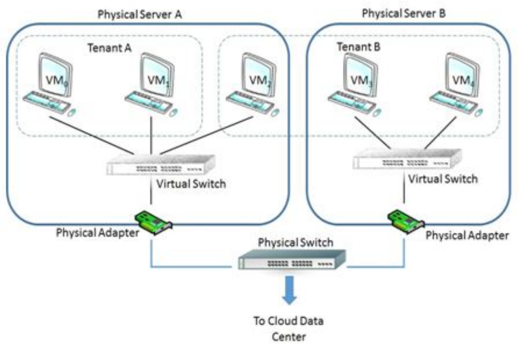
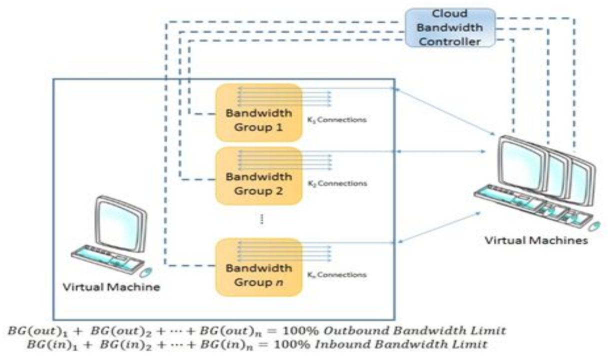


**Figure 1** Tenants and Servers within the Cloud Data Center



Cloud Bandwidth Controller
Bandwidth Group 1
K1 Connections
Bandwidth Group 2
K2 Connections
Bandwidth Group n
Kn Connections
Virtual Machine
Virtual Machines
$BG(out)_1 + BG(out)_2 + \cdots + BG(out)_n = 100\%$ Outbound Bandwidth Limit
$BG(in)_1 + BG(in)_2 + \cdots + BG(in)_n = 100\%$ Inbound Bandwidth Limit

**Figure 2** Dynamic Management of Bandwidth Groups

This represents a need to manage network bandwidth consumed by each virtual machine to prevent contestation and congestion of physical network bandwidth resources.

## 3.3 Bandwidth as a Service

This paper proposes an innovative bandwidth management scheme to occur at the virtual machine connection level of granularity. Groups of network connections are assigned percentages of virtual machine bandwidth, marshaled by the cloud bandwidth controller. A bandwidth service API for manipulation of bandwidth guarantees is provided to programs running on the virtual machine. These features represent the foundation of Bandwidth as a Service. As shown in Figure 2, tenants may specify percentages of bandwidth guaranteed to groups of network connections by the cloud bandwidth controller. Enforcement of bandwidth guarantees occurs at virtual machine operating system level at the direction of the cloud bandwidth controller. This provides the bandwidth con- troller information about bandwidth utilization for each virtual machine, allowing for dynamic bandwidth management.

We propose a solution whereby the cloud bandwidth controller assigns a fixed bandwidth limit per virtual machine for both incoming and outgoing traffic. At the request of tenant programs, the bandwidth controller may divide virtual machine bandwidth into BG(in)n and BG(out)n bandwidth groups. Each bandwidth group represents a minimum assurance best effort bandwidth as a percentage of total bandwidth for incoming and outgoing traffic. The tenant may assign an unbounded number of connections to a bandwidth group for those connections to be assured a minimum percentage of the available bandwidth. Connections within a bandwidth group share the available bandwidth in accordance with the virtual machine operating system network connection algorithm.

Tenant virtual machines under bandwidth management of the proposed cloud bandwidth controller may ensure particular connections are apportioned bandwidth proportional to the total bandwidth allotted to the virtual machine. While the tenant virtual machines are subject to Service Level Agreements (SLA's) that dictate the total amount of bandwidth available, connection grouping allows the virtual machine to explicitly instruct the bandwidth controller to choose which connections packets are dropped in the event that total bandwidth limita- tions are reached, ensuring best-effort bandwidth guarantees. Additionally, the programs running within the tenant virtual machine may access the bandwidth service API to program- matically request in situ modification to the bandwidth groups and network connection assignments. This gives the tenant the ability to control how bandwidth limitations are applied to network connections directly by virtual machine programming, allowing it to respond to demands for bandwidth control by business logic.

## 3.4 TCP vs SCTP

TCP (Transmission Control Protocol) and SCTP (Stream Control Transmission Protocol) are two transport layer con-nection oriented protocols [9-11]. While SCTP is much younger than TCP, their timeout mechanism, which affect the bandwidth, is essentially the same [12]. Although the work in this paper is on SCTP, it carries to TCP with need of minimal revisions.

This paper is organized as follows. In related work section we briefly review some related work. Bandwidth control section discusses the general concept of bandwidth control. In particular it looks into the behavior of connections contained within bandwidth groups and how bandwidth is controlled at the virtual machine level. Bandwidth control algorithm section presents our proposed bandwidth control algorithm. It also discusses the quanta timer mechanism and describes some details of algorithm implementation there. Test description Section describes implementation test setup and results. It reviews and discusses the experimental data collected from the implementation testing. The results show that the proposed algorithm performs well and meet our design goals. Observations section is devoted to discussions of some interesting results and abnormalities observed during the experimentation testing of the algorithm. In particular it discusses as covetous bandwidth effect, and expectations of the SCTP protocol in response to dropped packets and the SCTP streaming concept. Conclusion section concludes with an overall discussion of this work and the direction of current and future work in Bandwidth as a Service research.

## 4 Related Work

Bandwidth management and control in the cloud has received considerable attention in recent years. A lot of work has been done for BAAS in the cloud. We briefly describe some important work in this section. Readers are referred to for more details [13].

Shieh and his colleagues investigated the issue of sharing datacenter networks [7]. The research examined two types of clusters used in production consisting of thousands of servers. The first type of clusters primarily supports public infrastructure cloud services that rent virtual machines along with other shared services. The second type of clusters provides platform cloud services that support map-reduced workloads. They also investigated the number of flows per task, and the task types. Among the tasks that read data across racks, 20% of the tasks use just one flow; another 70% of the tasks vary between 30 and 100 flows; and 2% of the tasks use more than 150 flows. It is observed that this variation is due to the roles of the tasks. In particular, reduced tasks use a large number of flows and hence can severely degrade the performance. The research found some important issues of sharing the datacenter network: a) inadequate of TCP flow control and congestion control scheme; b) TCP can provide high overall network utilization, but cannot support robust performance isolation; c) the de facto way of sharing the network leads to poor schedules. To solve the above problems the research proposed a network bandwidth allocation scheme, named Seawall, which divides network capacity based on an administrative-specific policy. Seawall is an edge based mechanism that lets administrators prescribe how their network is shared, given a network weight for each local entity that serves as a traffic source (VM, process, etc.) Seawall ensures that along all network links, the share of bandwidth obtained by the entity is proportional to its weight. Hence it provides a simple abstraction of network traffic. Seawall can ensure that along all network links, the share of bandwidth obtained by the entity is proportional to its weight (per entity weight). Seawall works irrespective of traffic characteristics such as the number of flows, protocols or participants. Seawall is prototyped and implemented. It is demonstrated that Seawall provides much improved network bandwidth allocation in datacenter networks.

The work in studied the characteristics of multi-tenant datacenters and the supporting network. It looked at the impact of un-predictable application performance in multi- tenant datacenters - revenue loss (of providers and tenants) and expense unpredictability [14]. It proposed and implemented a virtual network abstraction, named Oktopus to capture the performance guarantees offered to tenants, tenants' cost, and the provider's revenue. The abstraction maps tenant virtual networks to the physical network in an online setting, and enforce these mappings. It allows specifications of tenants resource and performance parameters and captures the trade- off between the performance guarantees offered to tenants, their costs and the provider revenue.

In subsequent work, Ballani, et al. studied so called chatty tenants in the cloud and network sharing issue among those tenants [6]. The work defines and differentiates intra-tenant and inter-tenant traffic of datacenter tenants. Informally speak- ing chatty tenants are those tenants which perform significant inter-tenant communications. The proposal tried to provide a viable bandwidth sharing scheme among intra- and inter- tenant traffic and provide tenants with minimum bandwidth guarantees while bounding their maximum network impact. It describes a VM placement algorithm that utilizes max-flow network formulation to satisfy the guarantees. Such guarantees help tenants achieve predictable performance and improve overall datacenter throughput by providing tenants with mini- mum bandwidth while achieving upper-bound proportionality. In addition bounding a tenants maximum impact mitigates potential malicious behavior. The work is backed up with a prototype deployment and large scale simulations.

Lee and et al proposed a new abstraction for specifying bandwidth guarantee in cloud applications [15]. It is observed that in most cases communications of cloud application do not have much similarities with the physical network topology. Hence timing and bandwidth requirements of cloud applica- tions are quite different. A notion named TAG (tenant application graphs) was proposed to describe bandwidth require- ments of cloud applications. An abstraction of TAG, named CloudMirror was described. It was shown that CloudMirror's VM deployment algorithm can optimize the placement of VMs on physical servers while reserving needed bandwidth on physical links and enforcing the reserved bandwidths. The algorithm was tested through simulations based on the work- load derived from Microsoft's bing.com datacenter. Simulation results showed that Cloud Mirror required significantly lower bandwidth than Oktopus for the same application set.

## 5 Bandwidth Control

We propose an assured minimum bandwidth allocation scheme to the tenant virtual machine's network connections. Consider a virtual machine with a fixed bandwidth allocation for incoming traffic. The total number of bandwidth groups can be dynamic or static. For illustration purposes, four bandwidth groups are requested by the virtual machine and granted by the cloud bandwidth controller. The tenant virtual machine has assigned one network connection per group, for four total network connections. In this configuration, network connections one through four are assured a minimum bandwidth of 25% each from the fixed bandwidth allocation for incoming data traffic.

## 5.1 Bandwidth Groups and the Free Pool

Each bandwidth group has an actual limit, and a configured limit. As shown in Figure 3, the bandwidth controller will analyze at each time quanta which groups have not exceeded their actual group bandwidth limit, providing the available bandwidth for

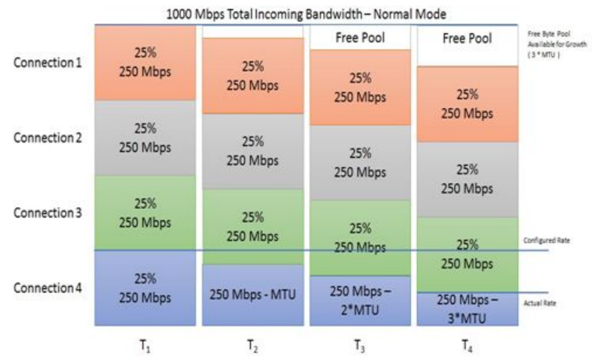


**Figure 3** Example Bandwidth Groups and the Free Pool

other groups to grow. If a bandwidth group has not utilized all of its configured bandwidth, then the actual group limit will be lowered by a (configurable) fixed amount at the end of each quanta. As groups shrink, their unused bandwidth is contributed to the "free pool", allowing groups that wish to grow in the following quanta to draw from the free pool.

## 5.2 Bandwidth Group Depression

In Figure 3, we see connection four was ’depressed’ from its configured rate of 25% to its actual rate of 10% solely because only 10% was being utilized. Afterwards connection one grew from its configured rate of 25% to its expanded rate of 40% of total allocated bandwidth (not shown for brevity), because free bandwidth was available due to the depression of the bandwidth group containing connection four. We now consider the case where connection four wishes to utilize its guaranteed configured bandwidth, in the absence of available bandwidth in the "free pool".

## 5.3 Resetting Bandwidth Group Limits

In the presence of network bandwidth demand, the minimum configured guarantee of 25% to the bandwidth group for connection four must be satisfied. In addition, the band- width group containing connection one was only guaranteed a minimum of 25% even though it grew to use all remaining bandwidth. All bandwidth groups will reset their actual band- width limit to match their configured limit, satisfying both requirements.

Finally, consider the situation where the total bandwidth limitation is exceeded. Identifying the specific bandwidth group which exceeds the total bandwidth limitation is not necessary for the situation to be remedied, since the actual bandwidth limits are simply reset to their configured limits. In this case, reaching more than 100% of the bandwidth maximum limit is not algorithmically allowed since packets would be dropped. In the following quanta, the reset takes place.

## 5.4 Relating Our Bandwidth Control Algorithm to Existing Work

Our proposed bandwidth control scheme can be integrated into existing cloud bandwidth management schemes [7,14,6,15]. For example, intra-virtual machine and intra- datacenter tenants in a datacenter network have shorter prop- agation delays. Therefore they can be allocated a bandwidth group with higher percentage of total bandwidth. Tasks from tenants with one flow or small number of flows usually are caused by real-time applications. They can be allocated a bandwidth group with proper total percentage of bandwidth. Bandwidth groups can be established based on the cost, revenue, and performance QoS.

# 6 Bandwidth Control Algorithm

There are two events where the algorithm logic is triggered, when data packets are sent or received, and when the quanta timer fires. The logic is broken up into two directions, sending and receiving. Other than the direction of data flow, the logic is identical, and order is reversed.

## 6.1 Data is Sent / Received

If the data packet in transit is within the bandwidth limitation, it is passed to the SCTP protocol stack for processing and forwarding to the application layer. If the actual bandwidth for the quanta is beyond the configured limit, but free bandwidth is available from the free pool, then the bandwidth group will grow using bytes from the free pool to accommodate the extra bandwidth and pass the data packet up the stack. If there is no free pool bandwidth available, then the data packet is dropped. Dropping packets at the transport layer has dual effects. First, the endpoint actively drops the packet to prevent processing packets beyond the maximum bandwidth limit. Secondly, the action of dropping the packet on a reliable transport connection (SCTP) is a mechanism to indirectly notify the sender that data loss has occurred, resulting in a Retransmission (RTX). Since, in the receive case, the packet is dropped before the transport layer has a chance to record the packet arrival, no acknowledgment will be sent, causing the sender to hold off sending any more data until the Retransmission Timeout (RTO) has expired. Since retransmissions are not desirable events, steps must be taken to avoid RTX whenever possible.

If the total bandwidth utilized for the current quanta is within (a configurable limit of) 2*MTU (packet maximum transmission unit size) of the total band- width allowed for the virtual machine, explicit congestion notification (ECN) is engaged. For incoming bandwidth traffic, the CE (Congestion Encountered) bit is set by the bandwidth controller for incoming packets, causing the SCTP protocol stack to send an ECNE (Explicit Congestion Notification Encountered) chunk to the peer, thereby reducing incoming bandwidth from the sender. For outgoing bandwidth traffic, the SCTP protocol stack is informed that it received an ECNE chunk (even if it really did not), which triggers the reduction of the outgoing congestion window to slow bandwidth utilization [16]. This help to avoid dropping packets (and associated RTX events) to enforce network bandwidth limitations.

## 6.2 Quanta Timer Expiry

The algorithm depends on a quanta timer (default one second) upon which the bandwidth control mechanism is maintained. The timer expiry logic is executed for all con- figured bandwidth groups. This is the primary mechanism upon which the "free pool" is given bytes while borrowing bandwidth bytes from partitions that are not using it.

## 6.3 Software Implementation

The bandwidth controller algorithm is implemented as part of the SCTP kernel module within the Linux kernel. Requests to transmit data from the application are intercepted by the bandwidth controller algorithm. The decision is made whether to allow the packet to pass on to the transport layer and then to the Ethernet driver, or to be dropped due to bandwidth limitations. Incoming data follows a similar data flow where packets from the Ethernet driver are intercepted by the band- width controller algorithm before reaching the transport layer. Here the decision is made whether to allow the packet to pass on to the transport layer and then to the application or to be dropped due to bandwidth limitations.

The advantage to this approach is that the transport layer is made unaware of any actions taken by the bandwidth controller algorithm allowing the transport layer to commence reliability operations without changes to the transport protocol. This also allows the bandwidth controller algorithm to drop packet data in a bidirectional fashion, relying on the guaranteed delivery feature of the transport layer to compensate.

The following is the pseudo code for the *traffic processing algorithm:*

```
Input: SCTP packet to be transmitted/received
Output: DROP packet or ALLOW processing
  $dropPkt = false$
  for $groupId = 0$ to $MaxBandwidthGroups$ do
    if assocId in Group[groupId] is true then
      if $TotalBytes + pktLen < MaxBandwidth$ then
        if $Group[groupId].byteCount + pktLen >$
        $Group[groupId].actByteLimit$ then
          if $pktLen \leq FreePoolBytes$ then
            $Group[groupId].actByteLimit += pktLen$
            $FreePoolBytes -= pktLen$
            $Group[groupId].byteCount += pktLen$
            $TotalBytes += pktLen$
          else
            if $Group[groupId].actByteLimit <$
            $Group[groupId].conf ByteLimit$ then
              $actByteLimit = confByteLimit$ for all outgoing
              groups...
              if $Group[groupId].byteCount + pktLen >$
              $Group[groupId].actByteLimit$ then
                $dropPkt = true$
              else
                $Group[groupId].byteCount += pktLen$
                $TotalBytes += pktLen$
              end if
            else
              $dropPkt = true$
            end if
          end if
        else
        $Group[groupId].byteCount += pktLen$
        $TotalBytes += pktLen$
        end
      if else
        $dropPkt = true$
        $actByteLimit = confByteLimit$ for all outgoing groups...
      end if
      if $dropPkt = true$ then
        $Group[groupId].bytesDropped += pktLen$
      end if
      if $TotalBytes + ( MTU âĹŮ 2 ) \geq MaxBandwidth$ then
        Engage Explicit Congestion Notification
      end if
    end if
  end for
  return dropPkt
```

Here is the pseudo code for the *timer expiry processing algorithm*. Source code can be found in [17].

```
Input:OS Timer Exipred for Ingress/Egress Group Manage-
  ment
  forgroupId = 0 to MaxBandwidthGroups
    doif assocId in Group[groupId] is true
    then
      if $Group[groupId].byteCount + MTU \leq$
      $Group[groupId].actByteLimit$then
        $Group[groupId].actByteLimit -= MTU$
        $FreePoolBytes += MTU$
      end if
      $Group[groupId].byteCount = 0$
      $Group[groupId].bytesDropped = 0$
    end if
  end for
```

# 7 Test Description

In order to simulate virtual computers communicating within a cloud environment, this research utilizes the Xen Project Hypervisor Server 4.4.1 running with Debian Linux 7.0 (Wheezy) x86-64 as dom0 [18,[19]. The Xen Server host hardware configuration consists of an Intel Core i5-3570K CPU 3.40 GHz with 8192 MB DDR3 RAM 1372 MHz. The Xen Server hosts two virtual machines, both running at default hypervisor configurations and identical software con- figurations. The only differences are the test roles. The guest virtual machines are running Arch Linux 2014.09.03 with Linux Kernel 3.16.3 using default kernel build parameters for Arch Linux [20,21]. Each virtual machine is a uniprocessor with 512 MB of RAM allocated and runs within the Xen Hypervisor as a HVM (Hardware Virtual Machine) client [19]. The algorithm is experimentally verified by using virtual machines with four distinct connections between them, with each connection within its own bandwidth group within both the server and client virtual machine. This data discusses the results of bandwidth limitation on incoming data from the point of view of a "server" which receives data from a client virtual machine. For brevity, only one group is cataloged in this paper, for a server receiving data and a client sending data, encountering bandwidth limiting actions.

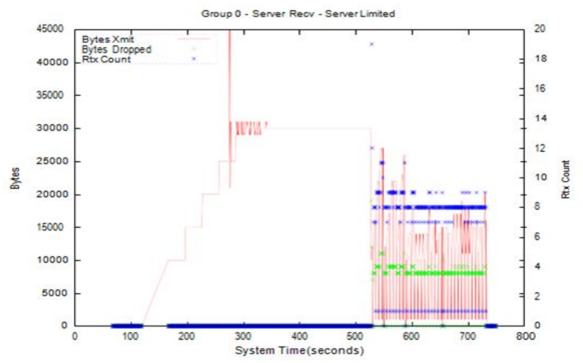


**Figure 4** Incoming Bandwidth Group - 10,000 bytes/sec Minimum Assured

## 7.1 Server Receiving Data

The data reported in this section reflects bandwidth uti- lization from the point of view of the server virtual machine receiving data from a client. There is no bandwidth limitation on the client for this test case so all retransmissions and bandwidth limitations are an effect of the bandwidth limitation actions of the server.

The graph in Figure 4 represents bytes sent every quanta for the 10% bandwidth group. Over time, the connection within the group utilizes more and more bandwidth, holding steady at 30,000 bytes per second. The assured bandwidth for this group is 10,000 bytes per second. At the 527 second mark, the total incoming bandwidth limit is reached, so the bandwidth for this group is now limited by packet drops to its configured 10% minimum assured bandwidth - 10,000 bytes per second. The connection within this bandwidth group is attempting to send more data after each retransmission and is being limited by bandwidth enforcement of the SCTP kernel module. As a result, the recorded bandwidth measurement shows a "zig- zag" pattern due to dropping incoming data packets by the bandwidth enforcement mechanism. At the 720 second mark, the test is terminated.

When packets are dropped, the client sender stops sending data to allow for the packet Retransmission Timeout (RTO) to expire before sending more data. These retransmissions are represented by the blue dots overlaid in Figure 4, and in subsequent figures. Each corresponding drop in bandwidth is associated with a retransmission packet shortly thereafter. The amount of data dropped is represented by the green dots overlaid in Figure 4. This shows the amount of data that the client is attempting to send but was rejected by the kernel module.

## 7.2 Client Sending Data

The data reported in this section reflects bandwidth uti- lization from the point of view of the client virtual machine sending data to a server. There are no dropped packets on the client for

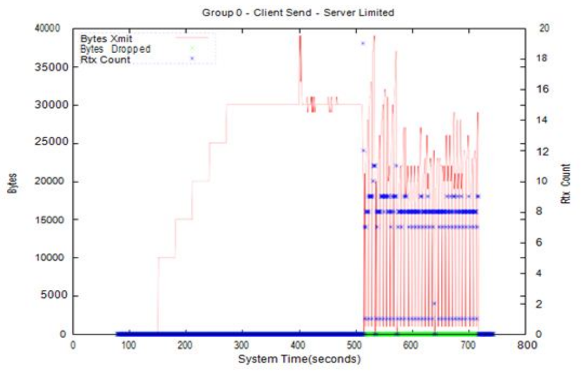


**Figure 5** Outgoing Bandwidth Group - 10,000 bytes/sec Minimum Assured

this test case so all retransmissions and bandwidth limitations are an effect of the bandwidth limitation actions of the server.

The graph in Figure 5 represents bytes sent every second for the 10% bandwidth group. Over time, the connection within the group utilizes more and more bandwidth, holding steady at 30,000 bytes per second. At the 512 second mark, the total incoming bandwidth limit for the server is reached, so the bandwidth for this group is now limited with packet drops by the server, since this group was only configured for 10,000 bytes per second minimum. Notice the increase in retransmissions after this point in time. This demonstrates an effect of the client attempting to send more than is allowed by the server. It is interesting that the SCTP transport protocol did not adjust the congestion window size accordingly; this is discussed in observations section. At the 720 second mark, the test is terminated.

## 7.3 Comparison with Uncategorized Bandwidth Control

The results of our new concept of bandwidth control are compared against the most basic form of bandwidth control; raw (uncategorized) bandwidth limiting at the network inter- face level. The Xen hypervisor comes with a built-in traffic control mechanism which makes use of queuing discipline to provide network bandwidth limitations [22].

To further demonstrate the difference, bandwidth test measurements are provided here with the same test parameters as described in Section V regarding data sent by a client virtual machine with outgoing client bandwidth limited by the receiving server. The bandwidth control mechanism presented as part of this research is disabled to allow only the effects of the Xen bandwidth limitations to be in effect. Xen is configured to limit the bandwidth of the client virtual machine to 100,000 bytes per second, dropping packets that attempt to breach this limit.

We consider only the test results in Figure 6, for brevity, which represents a discrete SCTP connection from client virtual machine to server virtual machine as conducted in Test Description Section . The pattern of traffic reveals a 'bursty' and stochastic behavior

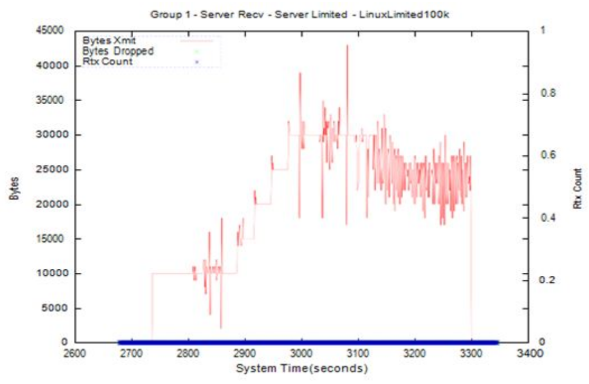


**Figure 6** Outgoing Bandwidth Connection

riddled with retransmissions and periods of time where little to no bandwidth is utilized by the four connections. This behavior is not wholly unexpected and demonstrates the key issue this research addresses.

## 8 Observations

In this section we discuss interesting and anomalous results from the experiments conducted as part of this research.

### 8.1 Covetous Bandwidth Effect

In the test case results we mention that a handful of retransmissions occur even though the bandwidth utilized is fully within the assured minimum bandwidth for the group. This is evidence of an anomaly in the proposed algorithm whereby other connections within bandwidth groups utilizing all available bandwidth could disrupt other bandwidth groups for a single quantum of bandwidth period (one second).
This can occur under the following conditions.

• The virtual machine has met the maximum bandwidth limitation

• Connection groups utilize all available bandwidth

One or more bandwidth groups increase bandwidth uti- lization, still within their configured assured minimum bandwidth Consider a bandwidth group which, in prior quantas (For this research, a quanta period is defined as one second) is not fully utilizing its configured bandwidth, while within the current quanta another bandwidth group attempts to go beyond its configured bandwidth allocation. The bandwidth controller utilizes the unused bandwidth from the first group to allow the second group additional bandwidth. Then, also occurring within the same quanta, the first bandwidth group attempts to utilize the remainder of its configured bandwidth, however there is now none available since the second group consumed all remaining bandwidth. The first group is therefore denied use of the bandwidth, and packets are dropped, causing the retransmission to occur for connections within the first band- width group. Under these conditions, the bandwidth groups are competing for available bandwidth up the total allowed for the virtual machine (coveting bandwidth beyond what was guaranteed) for that specific bandwidth quantum, not giving a chance for other bandwidth groups to utilize bandwidth within its assured minimum limit. This effect lasts for only one quantum and is corrected for in the following quantum. There are some proposed solutions to this issue. The quanta time may be shortened to minimize this effect. Bandwidth overutilization could also be allowed for a maximum of one quanta period to minimize retransmissions.

### 8.2 Transport Congestion Window and RTX

The congestion window as used in SCTP is the amount of unacknowledged transmitted data per connection may be outstanding before the stack delays further packet transmis- sions. SCTP continuously increases the congestion window in attempting to find the maximum throughput capable for the network link utilized using an AIMD approach [23]. The bandwidth controller will drop packets that attempt to go beyond the bandwidth limits (for connections within the limited bandwidth group), causing retransmissions to occur and causing the SCTP protocol stack to significantly decrease the congestion window. However on future transmissions, the amount of data sent across the connection increases up to the bandwidth enforcement limit within a few seconds. This implies that the AIMD increase in the congestion windows by the SCTP stack after an RTX may be too aggressive, causing more RTXs to occur unnecessarily. One solution to this issue is to modify the transport layer to have direct knowledge of the bandwidth limitation, allowing us to consider previous work which modifies transport level parameters (RTO, etc) to improve throughput in similar situations [12,24].

### 8.3 SCTP Streams and Head of Line Blocking

SCTP was chosen as the transport protocol used in these experiments. Early in experiment design, this work focused on individual streams within a single connection being con- tained within different bandwidth groups. The original idea was to focus on how streams are not subject to head-of- line blocking; if a packet for a stream was dropped, only packets for that stream queue up waiting for the retransmission acknowledgment to occur. This is unlike TCP; if a packet was pending acknowledgment, future packets would queue up for transmission until the first acknowledgment came in for the packet at the "head of the line" [25]. With SCTP, other packets assigned to other streams continue to transmit, even as one stream is blocked [23]. If bandwidth limitations are about to be exceeded, packets for a stream trying to utilize all available bandwidth should be dropped without affecting the data transmission of other streams. The goal was to capitalize on streams within bandwidth groups since it as implied they should not affect each other due to the no head-of-line blocking feature.

[6]Additive Increase, Multiplicative Decrease

The data collected showed that if one stream had packets dropped for bandwidth enforcement, all streams would slow or

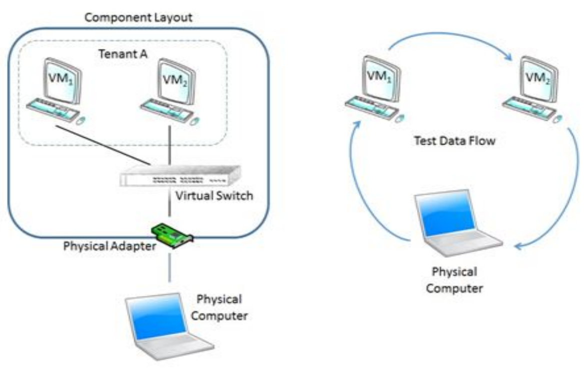


**Figure 7** Test Layout of Open vSwitch vs Xen Bridge

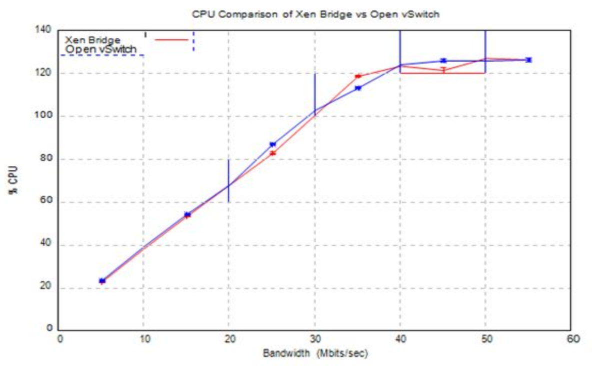


**Figure 8** Dynamic Management of Bandwidth Groups

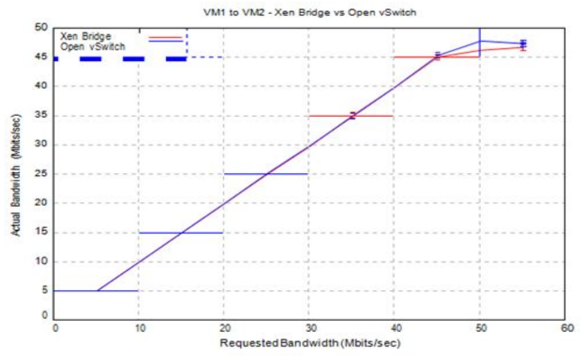


**Figure 9** Dynamic Management of Bandwidth Groups

stop their data bandwidth utilization, even if the other streams within their respective bandwidth groups did not violate their assured minimum bandwidth limits. Even though dropping packets on one stream does not trigger head-of-line blocking on another stream, the congestion window for the entire connection would drop significantly, thus affecting all streams within that connection. As a result, the goal was modified to perform bandwidth control at the transport connection level.

## 9 Xen Bridge Vs Open V-Switch

When analyzing the performance and pertinence of bandwidth control algorithms such the one proposed in this paper, an important question is the impact of network communica- tions of virtual machines on CPU utilization and throughput.

Packets from virtual machines can be routed using dif-ferent techniques. One of most popular techniques is virtual switches. Analyses in prior works appear implying that routing packets through virtual switches has been implemented in an inefficient manner, leading to an increase in CPU utilization as bandwidth load increases [26,27]. More recently two new alternatives, Open vSwitch and Xen Bridge have been proposed and experimented [28]. In particular Open vSwitch was proposed and has been adapted for multi-server cloud environments (openswitch.org).

Naturally we want to compare the performance of our proposed bandwidth control algorithm under both Open vSwitches and Xen Bridges. We catalog the performance in terms of CPU and bandwidth throughput for our test system for the built-in Xen Bridge and the Open vSwitch technolo- gies. Additionally, other throughput performance enhancing techniques are discussed to provide a future direction for improving network I/O communications within the hypervisor. In our bandwidth performance test setup we have two virtual machines (VM1, VM2) hosted within the Xen hypervisor and a physical machine (PM) physically connected to the network port of the host machine. The virtual machines hosted within the hypervisor utilize the virtual switch for communications between each other and communications with the outside physical network. Our test setup utilizes the popular bandwidth measurement tool Iperf3 to perform network connectivity tests in a ring formation such that data flows from VM1 to VM2 to PM and back to VM1, as shown in Figure 8.All increases in bandwidth for test purposes occur identically for all connections between machines. For all performance measurements, the impact imposed on the system by both the built-in Xen Bridge and Open vSwitch (in a mutually exclusive fashion) is cataloged.

Our first test case is CPU utilization for dom0 of the Xen Hypervisor as bandwidth between the test machines increases linearly. The goal of this test is to profile CPU utilization of the overhead introduced by the incremental traffic between the virtual machines and the external physical host. We see in Figure 9 that indeed CPU utilization increases as the bandwidth between machines increases, up to a limit, upon which CPU utilization levels off. Note that since the Xen Hypervisor host is running on a quad-core system, CPU utilization is repre- sented in a scaling fashion from 0 - 400around 40 Mbit per second where increased bandwidth utilization will no longer result in an increase in CPU utilization. This result confirms the findings by Cherkasova and Gardner, with allowances of physical test system differences [27]. This implies that the increased traffic beyond this point does not cause a demand on processing resources, suggesting lost data traffic.

Throughput is another critically important performance parameter of interest. We tested throughput in three different locations in Figure 7.

Our first throughput test case is about the effect on throughput between VM1 to VM2 as bandwidth increases across all machines. We can see from Figure 10 that as the requested bandwidth demand sent by the Iperf3 tool increases, the confirmed bandwidth increases linearly to a operational limit of approximately 45 Mbit

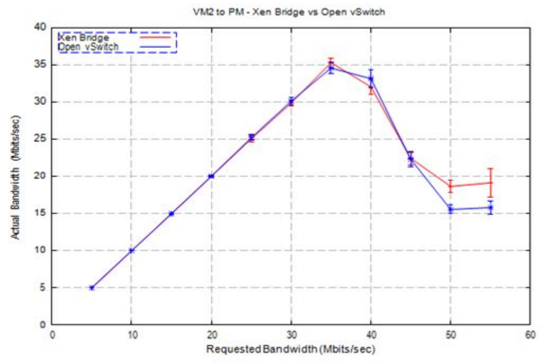


**Figure 10** Dynamic Management of Bandwidth Groups

per second. At this point the throughput between VM1 and VM2 begins to level off. This test shows that Open vSwitch allows for slightly more bandwidth in the 50 Mbit per second case, but both fall short of the requested bandwidth. This appears confirming Bardgett and Zous findings regarding the limited success of Open vSwitch [28].

We then looked at the throughput between VM2 to the PM as bandwidth increases across all machines. The result is shown in Figure 10. We see that the requested bandwidth sent by the Iperf3 tool increases the confirmed bandwidth linearly to an operational limit of approximately 35 Mbit per second. At this point the throughput between VM1 and VM2 begins to drop off significantly for both Xen Bridge and Open vSwitch. This test shows that Xen Bridge allows for slightly more bandwidth in the 50 Mbit per second case, but both fall short of the requested bandwidth. While certainly anomalous, these findings confirm the measurements documented by Menon, et al for throughput between virtual machines in Xen, allowing for differences between test systems [26].

The last throughput testing case looked at the throughput change between the PM to VM1 as bandwidth increases across all machines. As shown in Figure 11 we see as the requested bandwidth sent by the Iperf3 tool increases the confirmed bandwidth increases linearly for the entire test case, up to 55 Mbit per second. It is quite interesting that incoming traffic from the phys-

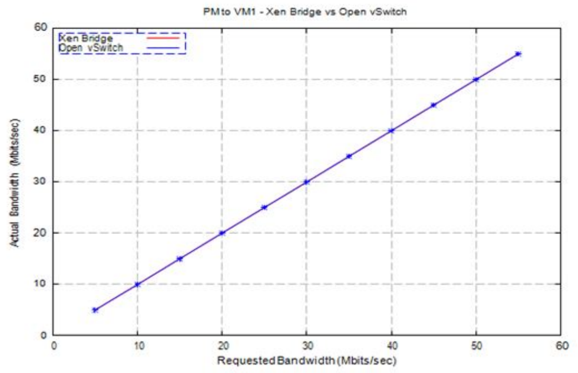


**Figure 11** Dynamic Management of Bandwidth Groups

ical machine to the virtual machine proceeds without any performance degradation. This finding implies that traffic incoming from the physical network proceeds with much higher throughput and efficiency as compared to traffic flowing between virtual machines. This also implies that software switching solutions (such as Open vSwitch and Xen Bridge) still suffer from inefficiency issues, and other approaches such as SR-IOV may provide better alternatives [28].

## 10 Conclusion

We proposed Bandwidth as a Service as a beginning towards providing tenants control over guaranteed minimum bandwidths to virtual machine network connections while the cloud provider maintains overall bandwidth control. We contribute to the BaaS concept through management of bandwidth groups that allow tenant virtual machines to have programmatic control over how bandwidth for groups of network connections are managed, while overall bandwidth limitations are enforced via incoming and outgoing data rates. The cloud bandwidth controller serves as the agent which dictates how much band- width tenant virtual machines may consume, specified by the tenant SLA. The experimental results shown here demonstrate the algorithm to perform this service by a cloud bandwidth controller.

Our work looks at BAAS from an angle different from those in [7,14,6,15]. The proposed algorithm can be used as BAAS modules controlled by individual VM tenants in a cloud datacenter, or it can be embedded/tailored within the algorithms proposed in other research work. Even though in the current research we only implemented groups of bandwidth, it is easy to extend it to hierarchy of bandwidth groups where each bandwidth group can be further divided into subgroups of bandwidth and etc.

Other areas for further research include examining in detail the relationship between the transport protocol congestion window and stream RTX. Examining the "Covetous Band- width Effect" in order to further refine the bandwidth control algorithm, modifications of transport level reliability param- eters for more efficient utilization of bandwidth, leveraging SCTP multi-homing within the cloud environment for more efficient path selection, and investigations into packet processing offload techniques are additional areas of further research [11,29]. Currently we are investigating extension of the work in to BAAS and also looking into green computing issue in BAAS [24].

18. Debian - the universal operating system. 2014; Available from: http://www.debian.org/
19. The xen project, the powerful open source industry standard for virtualization. 2014; Available from:http://www.xen.org/
20. Arch linux.2014; Available from:https://www.archlinux.org/
21. The linux kernel archive. 2014; Available from: https://www.kernel.org/
22. Xiaojing W, Wei Y, Haowei W, Linjie D, Chi Z. Evaluation of traffic control in virtual environment. 2012 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science (DCABES). 2012:332-335. DOI: 10.1109/DCABES.2012.53
23. Stewart RR, Xie Q. Stream Control Transmission Protocol (SCTP): A Reference Guide. Addison-Wesley. 2002.
24. McClellan S, Peng W, Gonzalez E. Improving throughput in sctp via dynamic optimization of retransmission bounds. Network Protocols and Algorithms. 2015;7(3):89-110.
25. Scharf M, Kiesel S. Nxg03-5: Head-of-line blocking in tcp and sctp: Analysis and measurements. Global Telecommunications Conference. 2006:1-5. DOI: 10.1109/GLOCOM.2006.333
26. Menon A, Santos JR, Turner Y, Janakiraman GJ, Zwaenepoel W. Diagnosing performance overheads in the xen virtual machine environment. Proceedings of the 1st ACM/USENIX inter- national conference on Virtual execution environments. 2005:13-23. doi: 10.1145/1064979.1064984
27. Cherkasova L, Gardner R. Measuring cpu overhead for i/o processing in the xen virtual machine monitor. Annual Technical Conference on USENIX Annual Technical Conference. 2005;50:24-24.
28. Bardgett J, Zou C. nswitching: Virtual machine aware relay hardware switching to improve intra-nic virtual machine traffic. IEEE International Conference on Communications (ICC). 2012:2700-2705. DOI: 10.1109/ICC.2012.6363879
29. Dong Y, Yang X, Li J, Liao G, Tian K, Guan H. High performance network virtualization with sr-iov. Journal of Parallel and Distributed Computing, 2012;72(11):1471-1480.