Copyright ⓒ2016 KSII

High-Availability Virtual Communication for Cloud Access

Suthee Sirisutthidecha and Kiattisak Maichalernnukul
College of Information and Communication Technology
Rangsit University, Pathumthani, Thailand
[e-mail: suthee.si@rsu.ac.th]
*Corresponding author: Suthee Sirisutthidecha

Received December 1, 2015; revised April 29, 2016; revised June 17, 2016; accepted June 29, 2016; published August 31, 2016

Abstract

Cloud computing is a paradigm in which information is permanently stored in servers on the Internet and cached temporarily on clients. Virtual private network (VPN) is the most widely used technology for secure cloud access. Unfortunately, VPN-based cloud services become unavailable when a VPN failure occurs. In this paper, we propose a new scheme to improve the availability of VPN connections against such failures, called high-availability virtual communication (HAVC). Unlike most of the multipath transmission schemes in the literature, the proposed scheme is implemented by using a virtualization technique, and its protocol functions are independent of existing networks - potential clients are not required to modify their applications or operating systems. Simulation results show that the HAVC can not only tolerate VPN failures but also achieve high transmission performance.

Keywords: Network virtualization, overlay networks, cloud computing

A preliminary version of this paper appeared in the Eighth International C* Conference on Computer Science and Software Engineering, Yokohama, Japan, 13-15 July, 2015 [20]. In this TIIS submission, the number of considered VPN gateways is extended from 2 to *N*, and new topics/results including degraded services of real-time applications, HAVC-NIC implemention, HAVC connection, practical usage, high availability and failover, and comparison with a resilient VN are also presented.

http://dx.doi.org/10.3837/tiis.2016.08.002 **ISSN : 1976-7277**

1. Introduction

Cloud computing and virtualization are the novel deployments of large-scale computing systems over the Internet. Indeed, virtualization technology is a tool which facilitates cloud-computing operation. It can also improve the availability and reliability of cloud platform. With this technology, cloud providers can guarantee service availability in terms of service level agreement (SLA) [1]. All services and applications based on cloud computing are normally provided over the public Internet. Most cloud providers then use virtual private network (VPN) to deliver secure cloud network access. VPN can be simply deployed on an existing shared physical network infrastructure like the Internet.

Nevertheless, there are many causative factors in VPN failures [2]-[3], e.g., poor network and software/hardware problems of VPN gateways. Such failures can lead to service interruption, poor SLA, and cloud providers' fines. When this happens and users try to re-access the Internet or make a new tunnel through another VPN gateway in order to connect the original cloud server, their Internet protocol (IP) addresses will be changed and the remaining segments of the original data stream become useless. Applications and services on cloud, such as group collaboration and voice over Internet protocol (VoIP), do not tolerate link outages even for milliseconds [4]-[8]. Meanwhile, transmission control protocol (TCP) applications, e.g., storage and file sharing, require a robust transportation in order to avoid interruption of the existing establishment [3], [9]. Therefore, high availability and fault tolerance of VPN communications must be considered in cloud access. There is a lot of research work taking advantage of resource redundancy in order to provide reliable execution of services and communication when the system goes down. Related work can be described as follows.

The work in [9]-[12] dealt with network failover by leveraging the redundancy of an underlying network, while in [13], VPN performance over multiple access links was improved by using inverse multiplexing with a modified TCP congestion control. In [14], a VPN failure was recovered by establishing VPN gateway connections in a virtualization layer. In [15]-[17], a TCP server failure was tolerated by using a backup server that takes over the TCP connection whenever the primary fails. In [18], a redundant end-system and its network coupling with the master were introduced to fulfill the requirements for time-critical VoIP applications. The work in [19] presented a fast failure detection and failover scheme that uses multiple session initiation protocol (SIP) dispatchers and multiple SIP proxy servers to achieve high availability. Still, backup devices, e.g., VPN gateways, servers, and redundant links, have not been explicitly utilized for increasing end-to-end throughput.

This paper aims at improving the availability of VPN communications for cloud access, especially on the client side, in a transparent manner. Specifically, we propose high-availability virtual communication (HAVC) which can maintain a cloud connection over VPN. While multipath transmission schemes [21]-[24] have recently attracted much attention for their high-availability performance, there are also several difficulties in their practical implementation. One of them is that users are required to modify their applications or operating systems. In contrast, the proposed HAVC is implemented by using a virtualization technique, and its protocol, i.e., logical communication, actually works independently of existing networks including VPN and cloud server. Another difficulty of implementing multipath transmission schemes is the emergence of reordering among packets of the same flow due to usage of different paths with diverse delays. To handle the out-of-order problem

incurred by multipath transmission, we present a method of distributing HAVC frames among all available HAVC paths, called HAVC-distribution (HAVC-D) method. The concept of this method is similar to that of [22], where transmission delays of each path are calculated and a data segment is sent through the TCP connection with the lowest delay. However, in [22], multipath transmission was deliberately established between two pre-specific gateways and thus crucially relies on them, whereas the HAVC-D method has no such restriction.

The main contributions of this paper are threefold.

1) The HAVC has the following superiority over other kinds of virtual networks (VN) that enable high availability (i.e., resilient VNs [25]-[26], software defined networks (SDNs) [27]-[28], and overlay networks [10], [12]): First, the HAVC services can be created on demand, regardless of the corresponding underlying network. On the contrary, the resilient-VN services rely on pre-reserved resources of the corresponding substrate network, and also an essential prerequisite for SDN services is that network elements (e.g., routers) must support SDN protocol. Second, a large number of overlay nodes are needed to operate overlay networks effectively, but the HAVC requires only two key components, i.e., HAVC network interface card (HAVC-NIC) and HAVC switch (HAVC-S), to function (See **Fig. 3**).

2) The HAVC can preserve a cloud connection in case of VPN failures. Due to its virtualization-based implementation, existing users and cloud servers are not required to modify their applications or operating systems (see Sections 4 and 5).

3) The HAVC can improve transmission performance in spite of its extra processing overhead, which makes it promising for real-time applications, whose transmission performance is critical (see Section 6).

The rest of the paper is organized as follows. In Section 2, a network environment is introduced, and related problems are discussed in Section 3. The HAVC system and protocol are described in Sections 4 and 5, respectively. Section 6 presents simulation results for the proposed system. Finally, Section 7 concludes the paper.

2. Network Environment

We consider a public-Internet based environment in which a number of VPN gateways are deployed on geographically distributed places in order to make sure that users are reachable (see **Fig. 1**). Indeed, we can deploy as many VPN gateways as possible. Moreover, VPN system can be implemented by any VPN technology, e.g., OpenVPN, IP security (IPSec), or point-to-point tunnelling protocol (PPTP). A group of cloud servers (called cloud site) are also deployed somewhere on the Internet. The VPN gateways are connected to the cloud site with different links. For such connections, lease lines are used in order to make reliable communication. The cloud servers provide services, Internet applications, or Internet services over VPN channels. In order to tolerate the failures of VPN gateways and improve the speed of transmission, users need to establish multiple VPN connections to these VPN gateways.

3. Related Problems

In this section, we will describe two different types of failures in the above network environment, degraded services of real-time applications, and re-establishment of TCP applications.

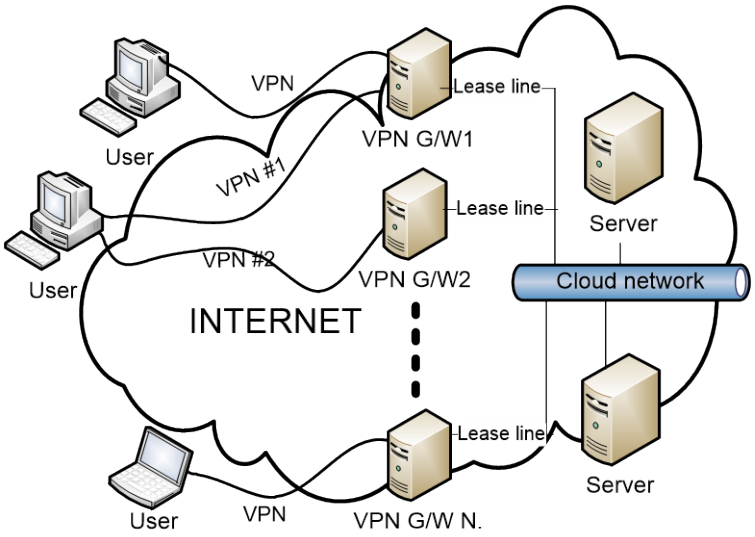


Fig. 1. Network environment

3.1 Network Failure

In the public Internet, a network failure could occur at various points including those on the client side or VPN gateway side [2]-[3], [15]. Examples of this failure include the failure of a link between a VPN gateway and the Internet, the failure of a link between a user and the Internet, and the network path failure or congestion in a backbone part of the Internet.

3.2 VPN Gateway Failure

A VPN gateway failure could occur from hardware error or software error, e.g., memory buffer overflow, central processing unit (CPU) overload, software bug or malfunction of some software processes [3], [8]. Furthermore, a VPN gateway server might be attacked from denial of service, malicious software and so on.

3.3 Degraded Services of Real-Time Applications

The services of real-time applications on cloud, e.g., group collaboration and VoIP, are considered as time-sensitive data transmission which operate under the real-time transport protocol (RTP) and RTP control protocol (RTCP) [4]-[7]. Consequently, jitter (i.e., maximum acceptable delay variation) becomes a major issue. For example, when VoIP connectivity on the client side has broken, a disconnected user must reconnect the cloud server within a range of few hundred milliseconds. Otherwise, the corresponding VoIP service will degrade significantly [29]-[31].

3.4 Re-Establishment of TCP Applications

According to its characteristics, TCP does not prevent itself from fault tolerance [32]-[33]. When a TCP server fails or loss its connection, TCP streaming will be broken. On the other hand, when a user's connection is lost, the TCP connection can no longer be transparently restored [2], [15]-[17]. Even if there has been a backup TCP server, the user is required to make a new TCP connection to it. In other words, the backup server cannot send the remaining TCP segments nor acknowledge the state of primary TCP connection as the user's IP address has been changed [15]-[17].

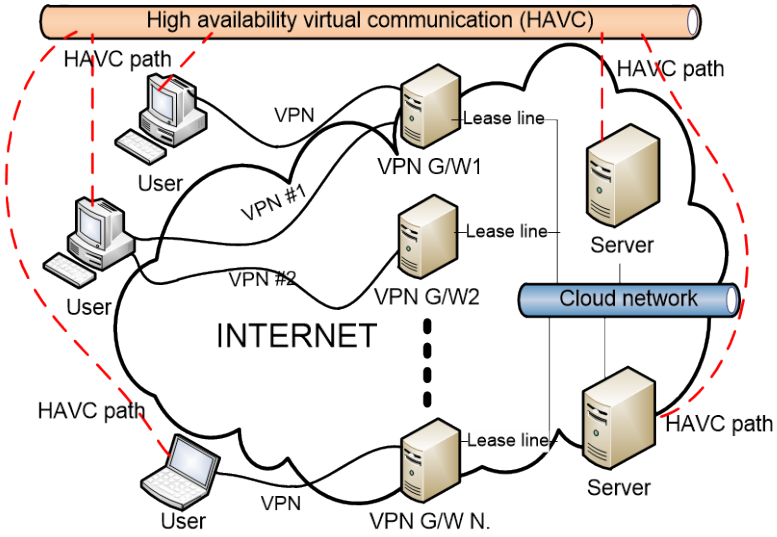


Fig. 2. Idea of HAVC

4. HAVC System

The idea of HAVC is illustrated in **Fig. 2**. The VPN gateways which provide VPN communication for users are connected to a cloud site with different links. To access the cloud, users first create VPN channels via any available Internet access technology, e.g., Wi-Fi or digital subscriber line (xDSL). Then, they make VPN connections to those VPN gateways.
In HAVC, a high-availability virtual network is created over its existing counterpart without modification of the TCP/IP suite. Users and cloud servers can work on this virtual network, like a virtual workgroup, and for an original IP packet, the source and destination are preserved. Thus, if a user's IP address is changed, the remaining segments or user datagrams for the original connectivity will not fail. Moreover, it is not necessary for users and cloud servers to modify their applications or their operating systems. In an HAVC network, users can indeed get large aggregated bandwidth from their parallel data transmissions.

As shown in **Fig. 3**, the HAVC system consists of two key components: HAVC-NIC and HAVC-S. To utilize the HAVC, a user and cloud server first install HAVC-NICs. Then, the user establishes VPN connections to all available VPN gateways, and plugs the installed HAVC-NIC into an HAVC-S over the VPN channels. Meanwhile, the HAVC-NIC installed on the cloud server is plugged into that HAVC-S through TUN/TAP tunneling (see **Fig. 5**). The virtual channel between any of the HAVC-NICs and the HAVC-S is called HAVC path. The number of HAVC paths that one can create depends on one's existing VPN connections. The HAVC-S is actually deployed on the cloud site to provide two main services: switching and addressing. Details will be discussed later.

Fig. 4 illustrates the HAVC multipath transmission. Typically, a user establishes a TCP connection to the cloud server when employing TCP applications. On the other hand, user datagram protocol (UDP) applications do not require such establishment. The user's TCP segments or datagrams are first encapsulated in an IP packet and Ethernet frame, respectively. Then, the Ethernet frame is intercepted by HAVC-NIC and encapsulated in an HAVC frame through adding a header that specifies the sequence number of payload (see **Fig. 7**).

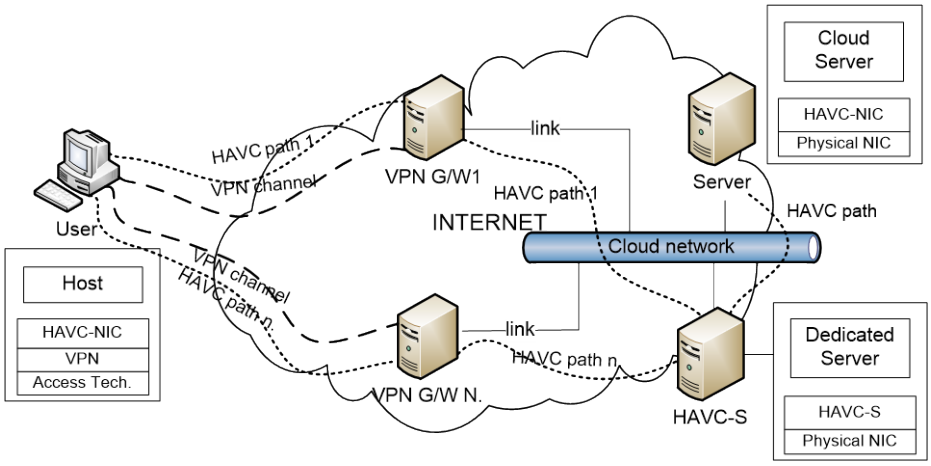


Fig. 3. HAVC system and its components

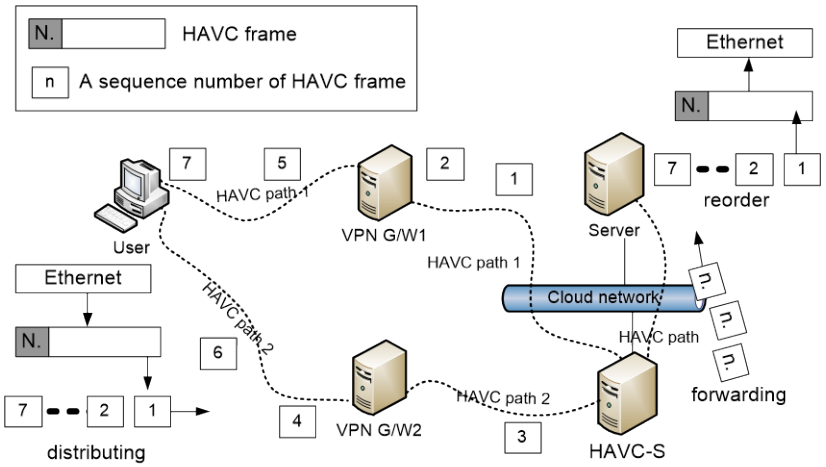


Fig. 4. HAVC multipath transmission

As seen in **Fig. 4**, the user has a couple of VPN connections and creates two HAVC paths. Therefore, this user can transfer data over both paths. More precisely, sequenced HAVC frames can be distributed to them. Once such frames arrive at the HAVC-S, they are immediately forwarded to the cloud server. At this server, the HAVC frames are reordered according to the corresponding sequence numbers, and are decapsulated to get the original Ethernet frames.

4.1 HAVC Network Interface Card (HAVC-NIC)

HAVC-NIC is a software required to be installed in end-host users. Its functions are classified according to two sub-layers: overlay packet and HAVC. **Fig. 5** shows the HAVC-NIC layer along with other layers in the network model. Since an HAVC-NIC is implemented based on the virtualization concept, it is not necessary for users to modify their existing applications and operating systems. This software can emulate a standard Ethernet interface card in its operating system. With HAVC-NIC, an Ethernet frame can be intercepted and encapsulated as HAVC frame. The HAVC frame transmission is therefore similar to the native Ethernet transmission.

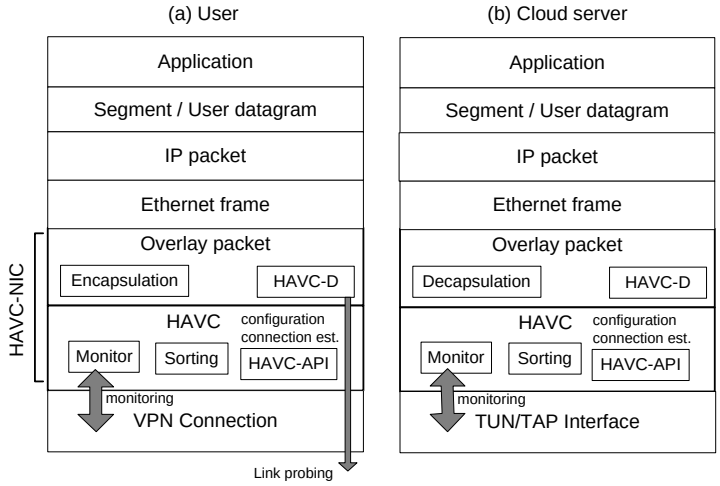


Fig. 5. HAVC network interface card (HAVC-NIC)

As shown in **Fig. 5**, either encapsulation or decapsulation module belongs to the overlay packet sub-layer. Another module in this sub-layer is called HAVC-D module, and its function is to distribute HAVC frames among all available HAVC paths. This module periodically probes the physical link status through HAVC connectivity, and the result will be used later in the HAVC-D method. Details of this method will be given in Section 5.

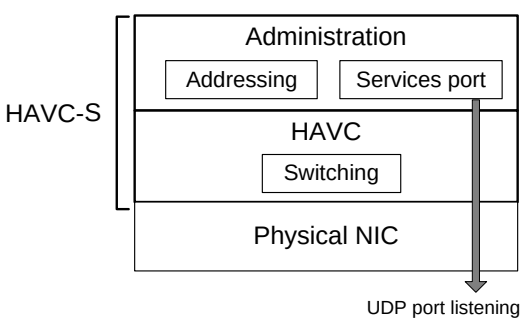
In the HAVC sub-layer, HAVC application programming interface (HAVC-API) module manages HAVC connection and host configuration. Details will be discussed in Section 5. Besides, the monitor module for a user and that for a cloud server will function differently. The former monitor module periodically detects the status of VPN connection. Both failure detection and failure recovery will be performed by this module. Typically, user space that has been created by VPN connection is returned to HAVC-API for managing an HAVC path. Note that if there is only one VPN channel, this channel should be a default output interface. Furthermore, since VPN connection is not defined as a function of HAVC-NIC, a VPN channel must be created by manual or automatic VPN implementation. Meanwhile, the latter monitor module detects the user space of TUN/TAP interface. Lastly, a sorting module is used to reorder HAVC frames arriving at the receiver end before sending these frames to be processed in the upper layer.

4.2 HAVC Switch (HAVC-S)

HAVC-S is an overlay node that installs the HAVC switch server software to support specified functions. **Fig. 6** shows the HAVC-S structure. In the administration sub-layer, the addressing module manages HAVC MAC addresses. For instance, it will assign such an address to a user who desires to join the system. In addition, the service port is a UDP port that waits for connecting from such users.

In the HAVC sub-layer, when a new HAVC environment starts, the switching module will create a switching table in order to maintain the physical connection. In this table, the “user IP address” column keeps the connecting IP address (VPN address) of a user. Because one can create multiple HAVC paths by constructing multiple VPN connections, there could be several entries for one’s specific destination MAC (DMAC) address. The HAVC-S uses the “DMAC” column for finding the corresponding destination. For example, if a cloud server desires to

transmit data to a user, then the HAVC-S will use the first row of the user's record to initiate their connection.



Switching table

DMAC	Port	User IP address	Status
HAVC MAC 1	1	10.10.1.1	Active
HAVC MAC 1	2	20.20.1.1	Active
HAVC MAC 2	3	10.10.1.20	Active
HAVC MAC 3	4	20.20.1.32	Inactive
...	...	...	...
HAVC MAC N	n	-	-
HAVC MAC N	n+1	-	-

Fig. 6. HAVC switch (HAVC-S)

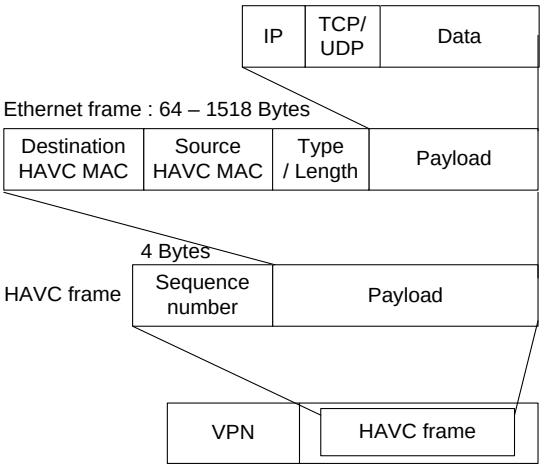


Fig. 7. HAVC frame encapsulation

5. HAVC Protocol

In this section, we will present HAVC protocol, which consists of addressing, encapsulation, connection, and multiple transmission.

5.1 Addressing and Encapsulation

The address of HAVC (i.e., HAVC MAC address) is defined to be compatible with the standard Ethernet application programming interface (i.e., the standard MAC-48 or extended unique identifier (EUI)-48 [33]). **Fig. 7** shows the HAVC frame structure. The HAVC frame encapsulates the Ethernet frame that consists of four fields: destination HAVC MAC address, source HAVC MAC address, type of Ethernet standard or length of IEEE802.3 standard, and payload. The payload depends on TCP maximum segment size or user datagram codec size. Thus, the size of the Ethernet frame can be 64–1518 bytes. The "sequence number" header of the HAVC frame specifies the sequence number for the payload of HAVC frame body.

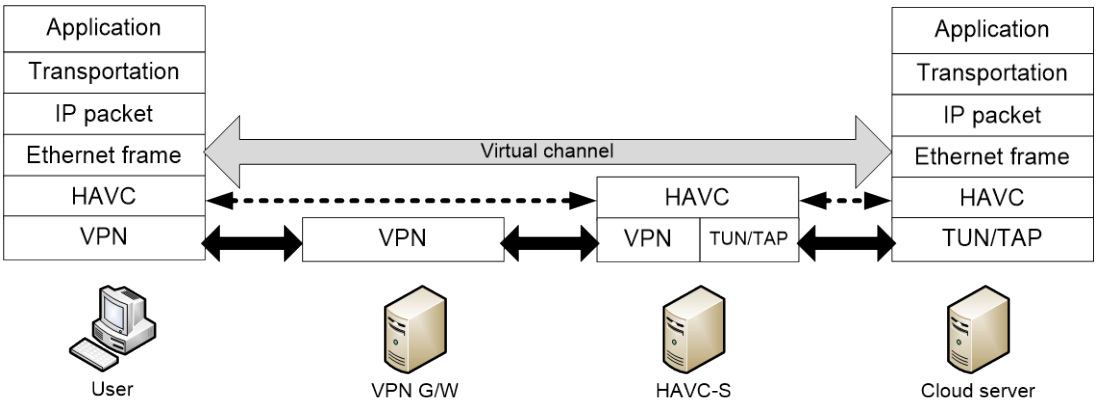


Fig. 8. HAVC connection

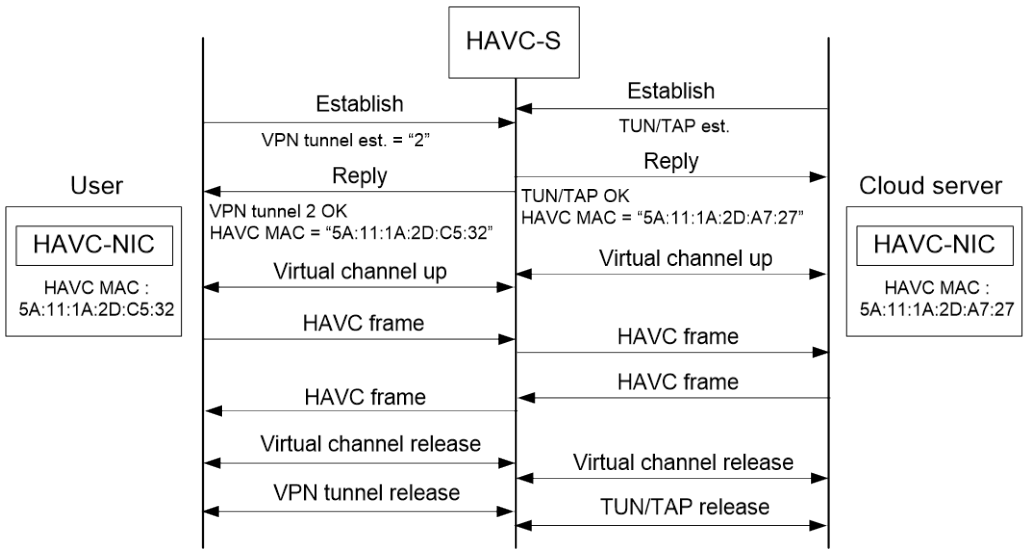


Fig. 9. HAVC timing diagram

5.2 Connection

Figs. 8 and **9** illustrate HAVC connection and the corresponding timing diagram, respectively. First, an end-host user is required to install an HAVC-NIC in order to emulate a standard Ethernet interface in its operating system. The HAVC-API in this HAVC-NIC starts connecting to an HAVC-S through an initialization request with the number of tunnel options that depends on available VPN connections. Then, the HAVC-S replies with the allocated resources: HAVC MAC address and available tunnel options. After getting this reply, the HAVC-API configures itself with the assigned HAVC MAC address. The end-host user is now ready to send/receive HAVC frames and utilize the HAVC-NIC as a standard Ethernet interface. It should be noted from **Fig. 8** that a physical connection between the HAVC-S and cloud server is simply established by TUN/TAP. In addition, the HAVC-S acts as a network bridge which sends frames across different network segments (i.e., VPN and TUN/TAP).

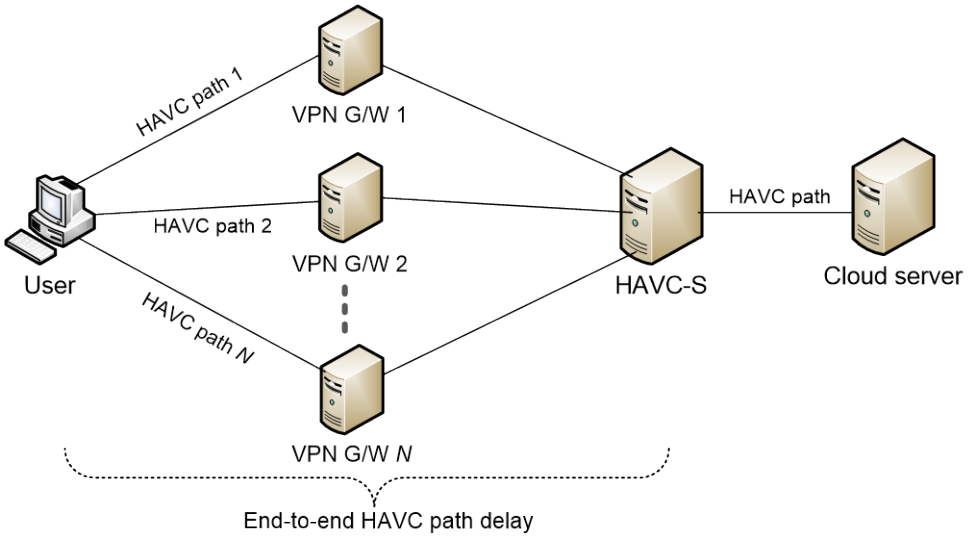


Fig. 10. HAVC-D method

5.3 Multipath Transmission (HAVC-D)

In a multipath communication, segments that are transmitted through different paths spend different transmission times [34]-[36]. This can have two adverse effects on TCP congestion control mechanism: 1) Premature timeout of the segment on the longest path; and 2) non-sequential arrivals of segments at the destination. These results lead to fast retransmit/fast recovery [32], [36]. Thus, corresponding applications would get poor throughput from unwanted events. In real-time applications, jitter which results from a variety of packet arrivals at the receiver would strongly affect quality of conversation and streaming playback [6]-[7]. However, there is an application-based mechanism that is used to compensate for jitter and out-of-order packet delivery, called play-out buffering. Note that the longer the buffer, the lower the quality of speech and playback. On the contrary, a shorter buffer increases the number of dropped packets.

HAVC-D is a method of distributing HAVC frames among all available HAVC paths (for example, N paths as shown in **Fig. 10**. The score of the i-th HAVC path is first calculated by

$$Score_i = Tr_i + Td_i,\ i = 1, 2, 3, \ldots, N \tag{1}$$

where $Tr_i \coloneqq \frac{\text{Maximum HAVC frame size}}{\text{Bandwidth of the } i\text{−th HAVC path}}$ is the corresponding maximum transmission time, and

$Td \coloneqq \frac{\text{Round trip time of the } i\text{−th HAVC path}}{2}$ is the corresponding end-to-end delay (between the user

and HAVC-S), and then HAVC frames are sent through the HAVC path having the lowest score. Note that the HAVC path with the highest bandwidth will get the lowest score. However, the score of an HAVC path can increase when the corresponding network delay increases. Therefore, the HAVC path through which HAVC frames are sent would change dynamically. The HAVC-D can enable in-order data delivery and reduce the data-sort cost at a receiver.

Nevertheless, there can be out-of-order data caused by delay fluctuation on the Internet.

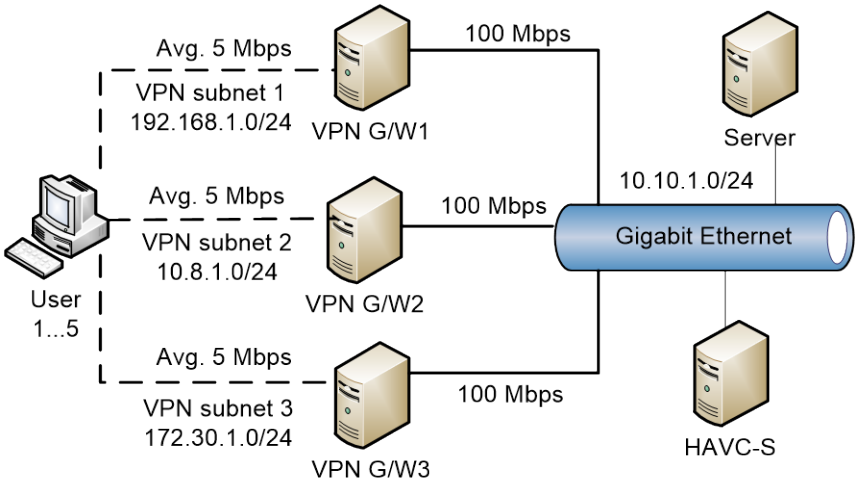


Fig. 11. Simulation model

6. Simulation Results

We simulate a network environment as shown in **Fig. 11**. Six personal computers (PCs) with Intel Core i7 and 8-GB memory are used as a client, VPN gateways, and servers. In the client PC, the Oracle VM Virtual Box 4.3 is used for building five guest OS with Ubuntu platform. The VPN gateways, namely G/W1, G/W2, and G/W3, are implemented using IPSec on Ubuntu platform. One of the server PCs operates on Windows Server 2012 platform as a cloud server, and another operates on Ubuntu Linux as an HAVC-S. The cloud server and HAVC-S are on the Gigabit Ethernet network, and the three VPN gateways are connected to the cloud network. Note that we have deployed three different VPN subnets: 192.168.1.0/24; 10.8.1.0/24; and 172.30.1.0/24. In order to access the cloud server, each user will make tunnels through those separate VPN gateways. The connection bandwidth for such tunnels is set to be on average 5 Mbps unless stated otherwise. The average delay between a user and VPN gateway depends on the corresponding intranet condition.

6.1 Overhead Effect

In this subsection, we consider the case where five users make two tunnels over G/W1 and G/W2 to access the cloud server, and apply a continuous PING test [37]-[38]. The result has shown that latency of HAVC is about 2–4 ms larger than that of IPSec (which is considered as a baseline) and its alternative, namely secure sockets layer (SSL) [32, Chapter 27]. The specifications of IPSec and SSL are shown in **Table 1**. This penalty is due to the overhead of HAVC processing, which includes adding the sequence-number header of an HAVC frame, forwarding such frames at the HAVC-S, and re-ordering them at a receiver (i.e., either a user or the cloud server). However, the latency of HAVC can be bettered by the users and HAVC-S when their physical resources, e.g., CPU and RAM, are upgraded.

Now, we test the average file transfer throughput for HAVC. The sizes of data in this test are 1 MB, 5 MB, 25 MB, 50 MB, and 100 MB. Examples of such data include documents,

Table 1. Specifications of IPSec and SSL

	IPSec	SSL
Open-source software	OpenSwan	OpenVPN
Encryption	256-bit advanced encryption standard (AES)	256-bit AES
Digest	Secure hash algorithm 1 (SHA1)	SHA1
Encapsulation protocol	Encapsulating security payload (ESP)	SSL
Client connection	Layer 2 tunneling protocol (L2TP)/IPSec	TUN/TAP

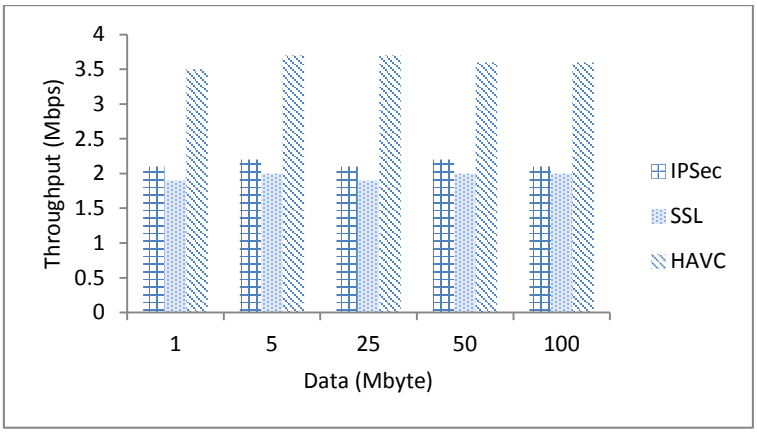


Fig. 12. Throughput measurement

pictures, and videos [39]. **Fig. 12** shows that the throughput of HAVC achieves 3.2–3.5 Mbps while the throughput of IPSec and SSL is around 2.2 and 2 Mbps, respectively. Therefore, it is clear that the HAVC can utilize transmission over two links. As mentioned in [29], [33], [39], however, some applications such as hypertext transfer protocol (HTTP) and VoIP would get low throughput because they require a significant amount of small-packet encapsulation in transmission and entail significant overhead. Such applications consume more bandwidth and resources when compared to the file transfer protocol (FTP) applications.

Further, it is worthwhile to evaluate how well the proposed HAVC performs in dynamic networking conditions. To this end, we vary the maximum delay between a user and VPN gateway from 5 ms to 100 ms and the loss rate from 0.001% to 1%, and measure the throughput of IPSec, SSL, and HAVC. The error bars based on 100 measurement trials are plotted in **Figs. 13** and **14**. We can see that the performance gains of the HAVC over the IPSec and SSL can be achieved even in the high-delay-variation or high-loss-rate regime.

6.2 Practical Usage

In this subsection, we conduct two HAVC experiments: one on TCP applications and the other on UDP applications. The details of the first experiment are as follows: 1) Load 2-MB webpages through HTTP; 2) Open e-mails and 5-MB attached file; and 3) Browse 10-MB database over a web application. The response time of these activities is then plotted in **Fig. 15**. We see from this figure that the response time obtained by using HAVC is less than that obtained by using IPSec and SSL. Therefore, the HAVC is promising for use on a cloud environment.

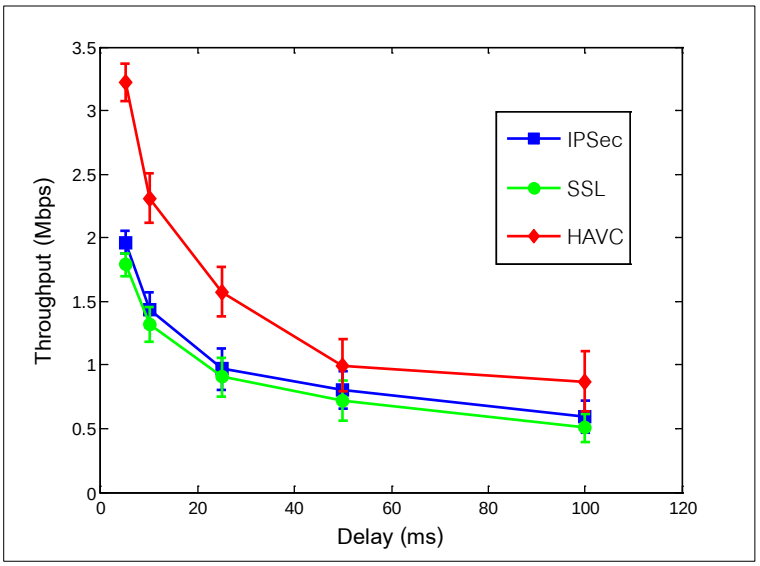


Fig. 13. Throughput versus delay

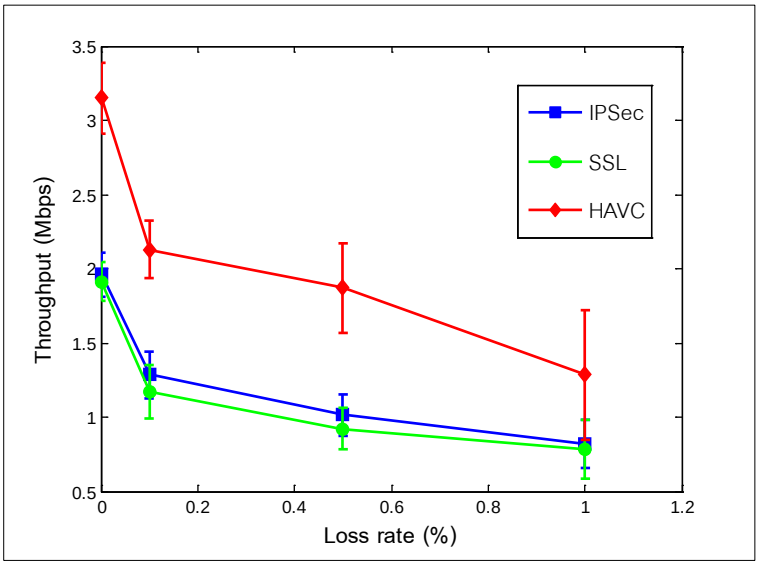


Fig. 14. Throughput versus loss rate

The second experiment consists of the following steps: 1) Create VoIP packets of 50-800 bytes (corresponding to different current codec techniques [40]-[41]); 2) Transmit these packets between users within 300 s; and 3) Measure the latency. This measurement indicates that the latency of HAVC is 15-20% more than that of IPSec. In **Fig. 16**, we plot the latency of HAVC with G.729 codec, which is mostly used in VoIP applications where bandwidth must be conserved. This figure shows that, for average end-to-end network delays of 5-100 ms (i.e., in the countrywide delay range), the HAVC can satisfy the quality-of-service (QoS) requirement [40]-[41].

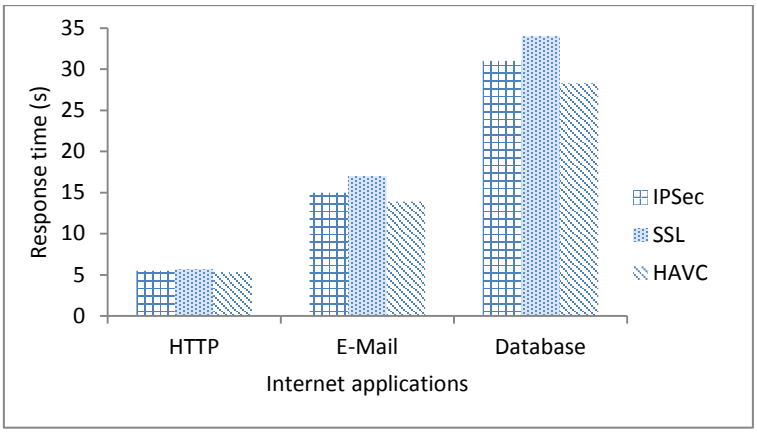


Fig. 15. Response time measurement

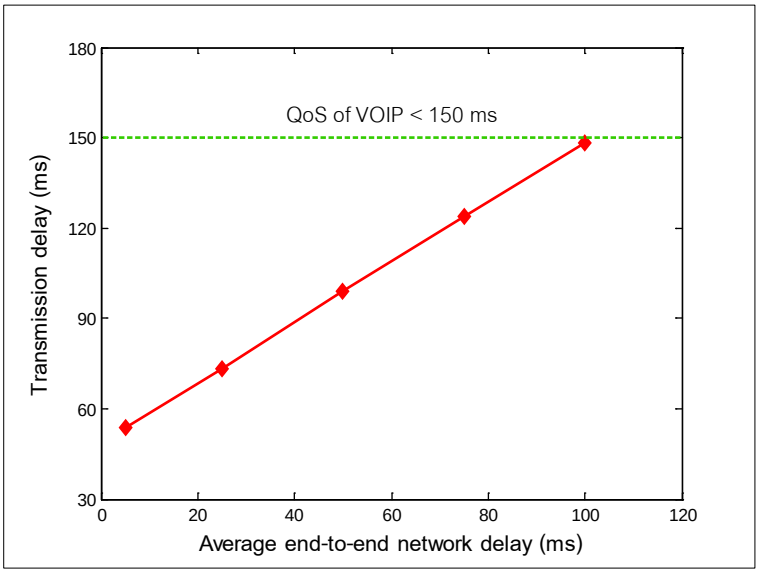


Fig. 16. VOIP experiment

6.3 High Availability and Failover

In this subsection, we will investigate the ability of HAVC to enable a user to continuously access the Internet in case of VPN gateway and network failures. Therefore, we consider the following tests.

- two-path HAVC: 1) Connect a user to G/W1 and G/W2 for cloud access; 2) Drop either of these VPN gateways at 30 s and recover it at 70 s; and 3) Drop either of them at 100 s and recover it at 130 s.

- three-path HAVC (Case 1): 1) Connect a user to G/W1, G/W2, and G/W3 for cloud access; 2) Drop one of these VPN gateways randomly at 30 s and recover it at 70 s; and 3) Drop one of the three active gateways randomly at 100 s and recover it at 120 s.

- three-path HAVC (Case 2): 1) Connect a user to G/W1, G/W2, and G/W3 for cloud

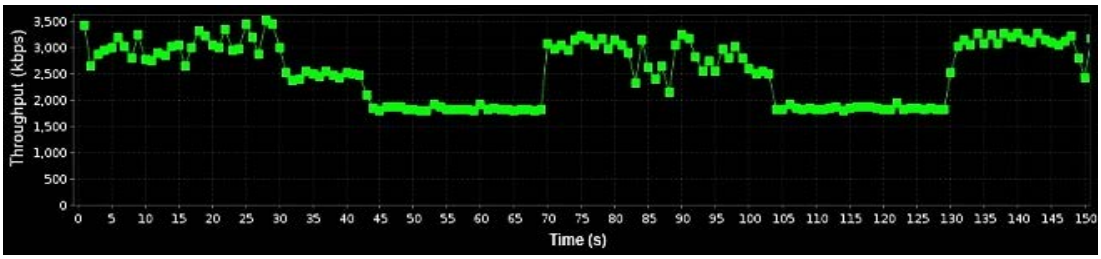


Fig. 17. Throughput of two-path HAVC

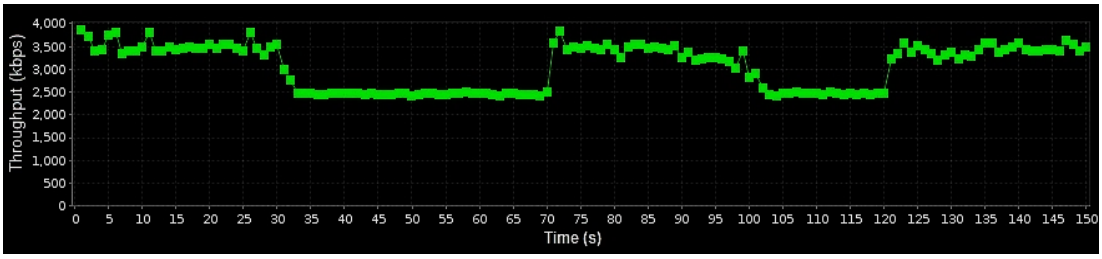


Fig. 18. Throughput of three-path HAVC (Case 1)

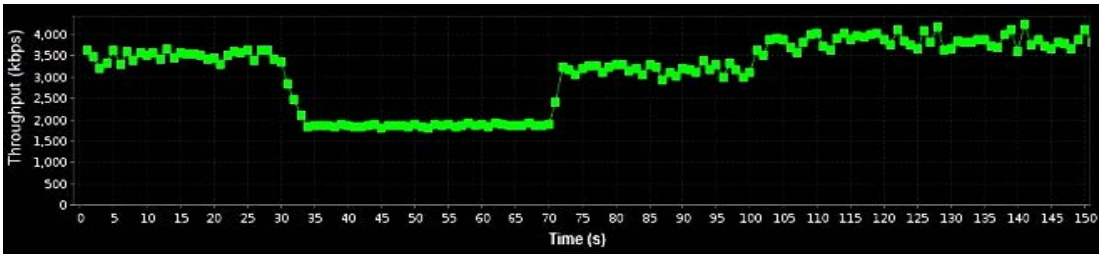


Fig. 19. Throughput of three-path HAVC (Case 2)

access; 2) Drop two of these VPN gateways randomly at 30 s, and recover one of them at 70 s and the other at 100 s.

Fig. 17 shows the throughput for the two-path HAVC test. It can be seen that, at the beginning of such connections, the throughput is 3 Mbps on average. When VPN gateway and network failures occur (i.e., at 30 or 100 s), the throughput decreases to approximately 2 Mbps. Still, the user can maintain a VPN connection to the cloud server.

Figs. 18 and **19** show the throughput for the different three-path HAVC tests (i.e., Cases 1 and 2, respectively). We can see from both figures that once the network has more active HAVC paths, it can achieve a higher throughput. This demonstrates the high-availability benefit of the proposed HAVC. Finally, it is noteworthy that increasing the number of active HAVC paths from two to three will give less gain than going from one to two. The main reason is that the HAVC-D method is actually not designed for the purpose of load balancing, but just facilitates concurrent multipath transfer.

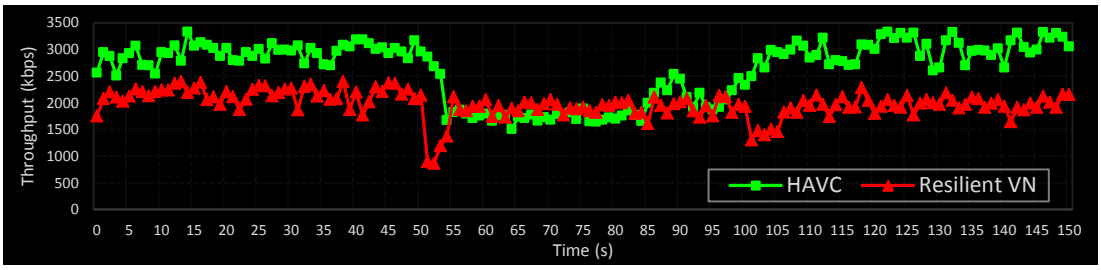


Fig. 20. Throughput of HAVC network versus resilient VN

6.4 Comparison with a Resilient VN

In this subsection, we will compare the two-path HAVC network with a resilient VN, which is adopted from [25]. Specifically, the latter network is built on a shared substrate network, and a bandwidth of 5 Mbps is reserved for each substrate link on the primary paths. We assume that one of the substrate links fails after 50 s a user has connected the virtual network service provider, and this failure is restored at 100 s. Meanwhile, in the HAVC network, either of the two VPN gateways that a user has connected is dropped and recovered at 50 s and 100 s, respectively. **Fig. 20** shows the resultant throughput. We observe that there is a sudden large decrease in the throughput of resilient VN (i.e., at 50-55 s). This is because this kind of network requires several seconds to fix the link failure (by enabling the restoration paths) [25]. Overall, the HAVC network performs better than the resilient VN due to the benefit of using all possible active HAVC paths.

7. Conclusion

We have presented HAVC that can preserve a cloud connection in case of VPN gateway and network failures. Due to its virtualization-based implementation, existing users and cloud servers are not required to modify their applications or operating systems. Moreover, the HAVC can improve transmission performance in spite of its extra processing overhead. The simulation results have verified that the resultant cloud connectivity is not breakdown when the above failures occur.

In evaluating our scheme, it is assumed that there is only one HAVC-S that serves the whole HAVC system, and the failure of this switch server has not yet been considered. If the HAVC-S goes down, then the HAVC system will become temporarily out of service. This issue remains as future work to be addressed.

References

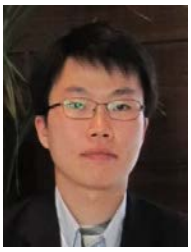
[1] V. C. Emeakaroha et al., "Towards autonomic detection of SLA violations in cloud infrastructures," *Future Generation Computer Systems*, vol. 28, no. 7, pp. 1017-1029, July, 2012. Article (CrossRef Link)

[2] N. Ayari et al., "Fault tolerance for highly available internet services: concepts, approaches and issues," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 2, pp. 34-46, July, 2008. Article (CrossRef Link)

[3] J. Han, G. Malan and F. Jahanian, "Fault-tolerant virtual private networks within an autonomous system," in *Proc. of Int. Conf. on Reliable Distributed Systems*, pp. 41-50, October 13-16, 2002. Article (CrossRef Link)
[4] J. Davidson et al., *Voice over IP fundamentals*, Cisco Press, 2006.
[5] M. Decina and V. Trecordi, "Voice over Internet protocol and human-assisted e-commerce," *IEEE Communications Magazine*, vol. 37, no. 9, pp. 64-67, September, 1999. Article (CrossRef Link)
[6] H. P. Singh et al., "VoIP: state of art for global connectivity-a critical review," *Journal of Network and Computer Applications*, vol. 37, pp. 365-379, January, 2014. Article (CrossRef Link)
[7] S. Karapantazis and F. Pavlidou, "VoIP: A comprehensive survey on a promising technology," *Computer Networks*, vol. 53, no. 12, pp. 2050-2090, August, 2009. Article (CrossRef Link)
[8] J. C. Brustoloni, "Automatic VPN client recovery from IPSec pass-through failures," in *Proc. of IEEE Int. Conf. on Local Computer Networks*, pp. 756-763, November 15-17, 2005. Article (CrossRef Link)
[9] M. Zhang et al., "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *Proc. of USENIX Annual Technical Conf.*, pp. 8-16, June, 2004. Article (CrossRef Link)
[10] Y. Chen et al., "The service overlay network design problem for interactive internet applications," *Computers and Operations Research*, vol. 57, pp. 73-82, May, 2015. Article (CrossRef Link)
[11] Y. Jia and C. Phillips, "Fast and reliable IP recovery for overlay routing in mission critical message oriented middleware," in *Proc. of Int. Conf. on Computational Science and Engineering*, pp. 1577-1584, December 19-21, 2014. Article (CrossRef Link)
[12] T. Murase, H. Shimonishi and M. Murata, "Overlay network technology QoS control," *IEICE Transcations on Communications*, vol. e89-b, no. 9, pp. 2280-2291, September, 2006. Article (CrossRef Link)
[13] J. Brassil et al., "Improving VPN performance over multiple access links," in *Proc. of Int. Conf. on Broadband Communications, Networks and Systems*, pp. 649-656, September 8-11, 2008. Article (CrossRef Link)
[14] Y. Matsuhashi et al., "Transparent VPN failure recovery with virtualization," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 78-84, January, 2012. Article (CrossRef Link)
[15] J. Kato, H. Fujita and Y. Ishikawa, "Design and implementation of a fault tolerant single IP address cluster," in *Proc. of Int. Conf. on Dependable Computing*, pp. 175-183, December 13-15, 2010. Article (CrossRef Link)
[16] M. Marwah, S. Mishra and C. Fetzer, "TCP server fault tolerance using connection migration to a backup server," in *Proc. of Int.Conf. on Dependable Systems and Networks*, pp. 373-382, June, 2003. Article (CrossRef Link)
[17] R. R. Koch et al., "Transparent TCP connection failover," in *Proc. of Int. Conf. on Dependable Systems and Networks*, pp. 383-392, June 22-25, 2003. Article (CrossRef Link)
[18] B. Sevcik et al., "Network layer based redundancy for time-critical VoIP applications," in *Proc. of IEEE AFRICON*, pp. 1-5, September 23-25, 2009. Article (CrossRef Link)
[19] W. Wu et al., "A fast failure detection and failover scheme for SIP high availability networks," in *Proc. of Int. Syms. on Dependable Computing*, pp. 187-190, December 17-19, 2007. Article (CrossRef Link)
[20] S. Sirisutthidecha and K. Maichalernnukul, "Reliable virtual channels over VPN for cloud," in *Proc. of the Eighth Int. C* Conf. on Computer Science and Software Engineering*, pp. 133-137, July 13-15, 2015. Article (CrossRef Link)
[21] M. Yabandeh, S. Zarifzadeh and N. Yazdani, "Improving performance of transport protocols in multipath transferring schemes," *Computer Communications*, vol. 30, no. 7, pp. 3270-3284, November, 2007. Article (CrossRef Link)
[22] Y. Hasegawa et al., "Deployable multipath communication scheme with sufficient performance data distribution method," *Computer Communications*, vol. 30, no. 17, pp. 3285-3292, November, 2007. Article (CrossRef Link)

[23] M. Li, A. Lukyanenko and Y. Cui, "Network coding based multipath TCP," in *Proc. of Global Internet Symp.*, pp. 25-30, March 25-30, 2012. Article (CrossRef Link)
[24] M. Shon, M. Jo and H. Choo, "An interactive cluster-based MDS localization scheme for multimedia information in wireless sensor networks," *Computer Communications*, vol. 35, no. 15, pp. 1921-1929, September 2012. Article (CrossRef Link)
[25] Y. Chen et al., "Resilient virtual network service provision in network virtualization environments," in *Proc. of Int. Conf. on Parallel and Distributed Systems*, pp. 51-58, December 8-10, 2010. Article (CrossRef Link)
[26] I. B. Barla, D. A. Schupke and G. Carle, "Resilient virtual network design for end-to-end cloud services," in *Proc. of Int. IFIP TC6 Networking Conf.*, pp. 161-174, May, 2012. Article (CrossRef Link)
[27] J. T. Araújo et al., "Software-defined network support for transport resilience," in *Proc. of IEEE Network Operations and Management Symp.*, pp. 1-8, May 5-9, 2014. Article (CrossRef Link)
[28] K. Nguyen, Q. T. Minh and S. Yamada, "A software-defined networking approach for disaster-resilient WANs," in *Proc. of Int. Conf. on Computer Communications and Networks*, pp. 1-5, July 30 - August 3, 2013. Article (CrossRef Link)
[29] E. Rodrigues, F. Cavalcanti and S. Wanstedt, "QoS driven adaptive congestion control for voice over IP in multiservice wireless cellular networks," *IEEE Communications Magazine*, vol. 46, no. 1, pp. 100-107, January, 2008. Article (CrossRef Link)
[30] H. Ying et al., "An adaptive algorithm for real-time data transmission in multi-hop overlay networks," in *Proc. of Int. Conf. on Communications and Networking in China*, pp. 1-5, August 25 -27, 2010. Article (CrossRef Link)
[31] X. Zhang et al., "Using P2P overlay to improve VoIP quality in SIP+P2P system," in *Proc. of Int. Conf. on Information Engineering*, pp. 255-259, July 10-11, 2009. Article (CrossRef Link)
[32] W. Stallings, *Data and Computer Communications*, 10th ed., Pearson, NY, 2013.
[33] D. Jielin, *Network Protocol Handbook*, 2nd ed., Javvin Technology Inc, Saratoga, CA, 2005.
[34] G. Chong, Z. Ling and Y. Yuan, "Packet reordering analysis for concurrent multipath transfer," *International Journal of Comuunication Systems*, vol. 27, no. 12, pp. 4510-4526, January, 2014. Article (CrossRef Link)
[35] K. C. Leung, V. O. K. Li and D. Yang, "An overview of packet reordering in transmission control protocol: problems, solutions and challenges," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 522-535, April, 2007. Article (CrossRef Link)
[36] J. Xiaomin et al., "A throughput improved path selection method based on throughput prediction model and available bandwidth for MPTCP," *International Journal of Future Generation Communication and Networking*, vol. 8, no. 2, pp. 105-114, 2015. Article (CrossRef Link)
[37] C. Labovitz, A. Ahuja and F. Jahanian, "Experimental study of Internet stability and wide-area backbone failures," in *Proc. of Int. Symp. on Fault-Tolerant Computing*, pp. 143-147, 1998. Article (CrossRef Link)
[38] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proc. of Int. Conf. ACM SIGCOMM*, pp. 124-151, August 31 - September 3, 1999. Article (CrossRef Link)
[39] RFC 6349, *Framework for TCP Throughput Testing*, 2011. Article (CrossRef Link)
[40] RFC 6374, *Packet Loss and Delay Measurement for MPLS Networks*, 2011. Article (CrossRef Link)
[41] RFC 5481, *Packet Delay Variation Applicability Statement*, 2009. Article (CrossRef Link)

Suthee Sirisutthidecha received the B.S. degree in physics and the M.S. degree in information technology from Kasetsart University, Thailand, in 1998 and 2005, respectively. He has more than ten years of experience in the area of network engineering. Since 2012, he has been a Lecturer with Rangsit University, Thailand.



Kiattisak Maichalernnukul received the B.E. and M.E. degrees from Chulalongkorn University, Thailand, in 2002 and 2004, respectively, and the Dr.-Ing. degree from University of Hannover, Germany, in 2010, all in electrical engineering. From 2005 to 2006, he was a Research Assistant with the National Electronics and Computer Technology Center, Thailand. From 2007 to 2011, he was a Scientific Assistant with the Institute of Communications Technology, University of Hannover. Since July 2011, he has been a Lecturer with Rangsit University, Thailand. His research interests are in the areas of communication theory and signal processing for wireless communications.