High Availability for IPsec VPN Platforms: ClusterIP Evaluation

Daniel Palomares, Daniel Migault, Wolfgang Velasquez, Maryline Laurent

High Availability for IPsec VPN Platforms: ClusterIP Evaluation

Daniel Palomares*, Daniel Migault*, Wolfgang Velasquez*† , Maryline Laurent†
*France Telecom, {firstname.lastname}@orange.com
†Institut Télécom, Télécom SudParis, CNRS Samovar UMR 5157, maryline.laurent@telecom-sudparis.eu

***Abstract*—To manage the huge demand on traffic, the Internet Service Providers (ISP) are offloading its mobile data from Radio Access Networks (RAN) to Wireless Access Networks (WLAN). While these RANs are considered trusted networks, WLANs need to build a similar trusted zone in order to offer the same security level and Quality of Service (QoS) to End-Users (EU). Although IPsec is widely implemented to create trusted environments through untrusted networks, the industry is increasingly interested in providing IPsec-based services with High Availability (HA) features in order to ensure reliability, QoS and security. Even though IPsec is not originally well suited to provide HA features, some mechanisms like VRRP or ClusterIP can work together with IPsec in order to offer HA capabilities. ClusterIP is actually used by *strongSwan* (an open source IPsec-based VPN solution) to build a cluster of IPsec Security Gateways (SG) offering HA features.**

This paper concentrates on how to build a cluster of IPsec SGs based on ClusterIP. We describe the main issues to overcome HA within IPsec. Then, we measure how HA may affect the EU experience, and provide recommendations on how to deploy ClusterIP. Finally, our tests over an HTTP connection showed that ClusterIP allows fast recovering during a failure.

***Index Terms*—IPsec Clustering, ClusterIP, Security Gateway Handover, Fast IPsec recovering**

I. Introduction

Nowadays, mobile operators are facing the exponential growth of mobile data. Among different solutions, a short term alternative consists in switching mobile data from Radio Access Networks (RAN) to WLAN network. For example, the iWLAN [1] architecture proposes to carry the IP data between the End-User (EU) and the operator's core network through a WiFi access. Once the EU is transferred to the WiFi access, it establishes an IPsec connection with a dedicated Security Gateway (SG) that gives access to some services or simply to the Internet. Furthermore, this IPsec session should maintain the QoS prior to offload and the SG must remain available and reliable to EUs.

The industry is increasingly interested in providing services with High Availability (HA) capacities. The HA clusters aim to increase the availability of a service. In terms of IPsec, an HA IPsec cluster increases the IPsec service's availability. For example, when an IPsec tunnel is established towards a given SG within a cluster, the IPsec parameters could be spread among its cluster members. In the event of a failure during an IPsec session, some other SGs must ensure the VPN service. Actually, the availability and the continuity of service are guaranteed by different mechanisms.

It is important to distinguish a mobility event from an IPsec context transfer due to a failover. Mobility means that an EU changes the IP address of the tunnel but the IPsec parameters of the communication remains in the same nodes. On the other hand, a failover mechanism requires a new SG to go on with an existing VPN session and the IPsec parameters must be set.Thus, all these parameters should be transferred towards another SG, and the affected EUs attaches the new SG.

Throughout this article, we concentrate on the methods that ensure IPsec availability as well as the continuity of an IPsec service. This mainly involves the transfer of a tunnel between different SGs. We also evaluate how an EU is affected when this happens.

The scenario we consider is an EU that wishes to reach an HTTP server placed within a trusted network protected by a SG. The EU first establishes a VPN towards a SG in the trusted network. Hence, the SG authenticates and protects the traffic between the EU and the HTTP server. We use strongSwan [2] to establish these tunnels. In order to provide HA capacities, we use ClusterIP, allowing to build a cluster of SGs that share a common IP address without having a physical machine to perform this task. Thus, each SG determines from the incoming packets whether it is responsible for it or not.

Our testbed consists in a cluster of two SGs configured with a common IP address. When a EU establishes a VPN towards the cluster, one of the SGs is considered the active SG, whereas the other member is considered the passive (also known as stand-by SG). The active SG is the one that takes responsibility of the tunnel, whereas the passive SG is waiting to become the newly active SG if a failure occurs to the active SG. Both SGs synchronize all their IPsec tunnels so that they keep track of every single tunnel established with any EU. Our experiments are mainly focused in causing a failure to the active SG during a VPN session and evaluating the High Availability performances of the platform. At this point, the passive SG (not affected by the failure) detects no activity through the Synch Channel between them and consequently applies a new ClusterIP policy and becomes the newly active SG for a given IPsec tunnel. This ensures availability of the VPN services and avoids renegotiation from scratch of each tunnel of concerned EUs.

This paper considers a HA solution for IPsec using *strongSwan*. First, section II positions our work and related subjects. Then, section III describes the challenge to overcome VPN counters synchronization during an IPsec context transfer. Following section IV explains how to set up the platform with ClusterIP. Section V explains how *strongSwan* implements ClusterIP in order to create IPsec SG clusters. Section VI shows the experiments performed as well as the results obtained. And section VII gives our conclusions.

II. Position of our Work & Related Work

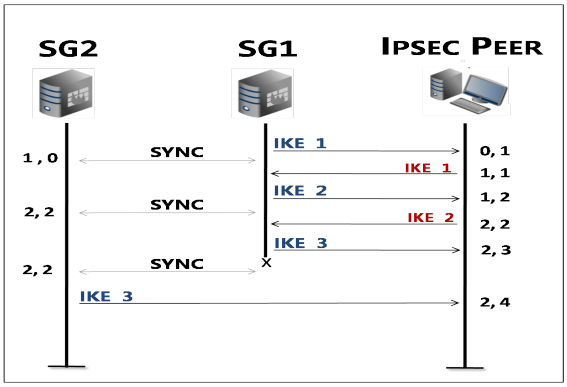
Concerning High Availability within IPsec, there exist similar works and approaches:

- **Yu [3]** proposes to solve availability issues on IPsec by simulating a cluster mechanism for IPsec gateways. As far as we know, this is the only paper that addresses similar issues. Its *seamless switching* mechanism aims to spread SAs among both active and passive SGs. The author does not recommend a High-Reliable link between SGs in order to communicate the SAs, but the article mentions that the members of the cluster could be deployed in different network segments. The *seamless switching* process consists of adding a notify payload during the IKE_AUTH exchange in order to establish two tunnels (one VPN towards the active SG and a second stand-by VPN towards the passive SG of the cluster). The passive SG receives the IPsec information from the responsible active SG and installs a passive VPN towards the EU. Finally, there is also a mechanism of *seamless switching* to transparently change from the active-to-passive SG and to synchronize ESP replay sequence number. On the other hand, the case of a heavily loaded SG is not considered. The results are based on simulations and not in real implementation. By contrast, our ClusterIP-based mechanism do not need additional IKEv2 payloads, so it is transparent for EUs.
- **RFC3768: The Virtual Router Redundancy Protocol (VRRP) [4]** aims to solve the failure of a single SG in a network. It adds redundancy to the network by creating a group of SGs with a common virtual IP address. This is similar to our mechanism based on ClusterIP. As such, those routers which belong to the same VRRP group will use the same virtual IP. Every interface that is configured with VRRP owns a virtual IP address that is common to all routers being part of the redundant topology. In the case where two or more routers are configured with VRRP, the responsibility is determined with the parameter **vrrp-priority**. It defines which router has the biggest priority in order to take responsibility as a SG among all the group. The goal is to create a default SG with a unique IP address so that any host accesing through one of the SGs, does not know how many routers are composing the group. The responsible SG acting as default SG is called *master virtual router*, whereas all the other routers are called *backup virtual routers*, which are ready to get the role of *master virtual router* and forward packets if the *master virtual router* fails. Note that VRRP does not aim to perform load-balancing as it does not distribute the load of VPNs among different routers, but it adds fault tolerance (redundancy) to the network. By contrast, ClusterIP is able to identify IPsec traffic and spread the load among different SGs. This permits VPN distribution among different SGs within a cluster. Additionally, the existing Open Source implementation of ClusterIP on *strongSwan*, supports hot-standby clustering between different SGs during an active VPN session. For all these reasons, ClusterIP was selected instead of VRRp over IPsec.
- **RFC6311: Protocol Support for High Availability of IKEv2/IPsec [5]**. This RFC proposes an extension to the IKEv2 protocol. It aims to solve the refreshing of both IKEv2 (IKE_SA) and IPsec (CHILD_SA or IPsec_SA) counters due to a mismatch caused by a failure take-over process. The scenario addressed in the document is oriented to solve hot-standby IPsec-Clusters failures. A hot-standby IPsec-Cluster consists of a group of IPsec SGs in which there is only one member active at a time. All the IKEv2/IPsec contexts **are distributed among all the members of the cluster**. When a failure occurs on the active SG during an IPsec/IKEv2 communication, the End-User (EU) continues sending IKEv2/IPsec traffic towards the cluster, leading to unsynchronized counters and packets loss. In order to reestablish the synchronization of counters, the new SG sends an IKEv2 message request of type INFORMATIONAL in order to negotiate counters. Concerning our work, this protocol solves the main troubles of synchronization of both IKEv2 messages (IKE_SAs) and IPsec counters (CHILD_SAs). Although it involves changes to the IKEv2 protocol, this extension handles the renegotiation of the IKEv2/IPsec counters in an efficient manner and should be considered if any IPsec/IKEv2 counters mis-synchronization occurs.

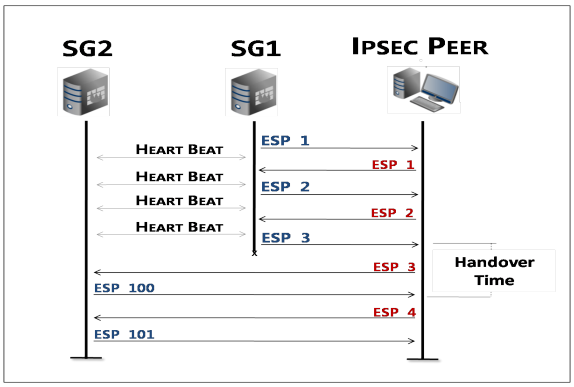
III. IPsec/IKEv2 and High Availability Constraints

IKEv2 and IPsec were primarily designed for static configurations. IPsec/IKEv2 states has accordingly been designed to remain installed in the same device during a session or VPN Tunnel establishment. However, todays requirements demand facilities to profit from mobility, handover, offload or even VPN session migration between SGs. This situation leads often to new extensions. For example: MOBIKE extension for mobility [6], MOBIKE-X draft for mobility and multihoming [7] [8] [9], CXTP and mechanisms to transfer security contexts [10] [11] [12] or a High Availability protocol to synchronize counters [5].

The huge demand on mobile data have increased the demand on system's availability. This is the goal of ClusterIP. Clustering a VPN service at the network layer level (e.g. IPsec) seems to have positive impact when dealing with reliability of IPsec-based communications. Because IPsec imposes a strict



(a) Stale Messages ID values



(b) Stale Sequence Number value

Fig. 1. IKE/IPsec counters desynchronization

check of its counters, the main issue is to synchronize both SG's IPsec parameters and counters. When an EU establishes an IPsec protected communication with a server, it first needs to configure how to protect the communication with the SG. The protocol used to negotiate the security parameters of the communication is IKEv2. Further exchanges allow to agree on a secure channel that serves to negotiate the IPsec parameters (to actually protect the IP traffic). IKEv2 and IPsec are **different** protocols with their **own** counters. The *message ID* counters provide anti-replay protection and keep track of every IKEv2 exchange, whereas the *sequence numbers* provide anti-replay protection and strict control of IPsec traffic. Even though the amounts of IKE messages do not represent as much traffic as the IPsec traffic, the synchronization of *messages ID* is critical due to the very strict control of the IKE_SAs when setting up a VPN tunnel and a very narrow window size.

A. IKEv2/IPsec Counter Synchronization

The IKEv2/IPsec suite aims to protect IP traffic. However, IKEv2 and IPsec act at different layers. IKEv2 is an application layer protocol which queries and responds to port 500 and 4500 in order to negotiate a secure channel between two endpoints sharing an IKE Security Association (IKE_SA). On the other hand, IPsec is a protocol that takes place at the IP layer and protects the traffic according to IPsec Security Associations (IPsec_SAs) policies, which are previously negotiated through the IKE_SA secure channel. All IKEv2 exchanges consist of a request-response pair of messages. It is mandatory to retransmit a request until it has been acknowledged. The IKEv2 window mechanism allows to send some IKEv2 requests without receiving a response, but once the window's limit is achieved, the oldest request must be acknowledged resulting in a window increase by one. This window is managed through the IKEv2 counters called: *Message ID*. Each IKEv2 message header includes its corresponding *message ID*, so that IKEv2 can strictly control the window as well as all unacknowledged messages. When the *message ID* is n and the window size is w, only IKEv2 messages between $n+1 < Message\ \ ID < n+w$ can be processed. If no acknowledgment is received after a long period of time, then the IKE_SA is deleted. Figure 1a shows an example of the mis-synchronization during IKEv2 exchanges between two endpoints with a window size $w = 1$.

Challenges concerning IKEv2 when implementing cluster of IKEv2/IPsec SGs are:

- *Stale Value of Message ID:* when takeover takes place, it is possible that the newly active SG is not aware of the last IKEv2 response made by the previously active SG. If this happens when taking responsibility, then the *message ID* used by the new responsible SG will be stalled. Figure 1a illustrates this case.
- *Unacknowledged Request:* when a passive SG takes responsibility during a fail-over, it may happen that it is unaware of the last request sent, and thus the counter is stale. Receiving an unexpected *message ID* response would result in discarding the packet. This leads to IKE_SA destruction.

The anti-replay parameter of the IPsec Security Association is called *sequence number* counter. When the Anti-Replay service is enabled, it controls every incomming/outgoing IP packet protected with IPsec. Note that IPsec allows a sender to skip forward by sending a higher *sequence number*. Remember also that a duplicated usage of a *sequence number* is forbidden. Thus, when the *sequence number* counter is n and the window size is w, any message with $sequence\ \ number < n - w + 1$ will be discarded. A big window size (E.g. 1024) means that a node is capable to handle a bigger range of packets that arrive out of order. On the other hand,when $sequence\ \ number = n$ and the window size is very little (E.g. $w = 1$), the node is capable to remember only the last *sequence number*. The following packet must have a value $sequence\ \ number > n-1$, otherwise the packet is dropped.

The biggest challenge concerning IPsec when implementing cluster of IKEv2/IPsec SGs is:

- *Stale Sequence Number value:* when a passive SG takes responsibility and becomes the newly active SG, it may happen that the *sequence numbers* are out of date. Figureřefesp illustrate this situation. This occurs when the newly active SG starts sending IPsec protected packets with staled *sequence numbers*, implying that the EU rejects all the duplicated packets due to anti-replay

protection of IPsec. Instead, note that IPsec allows to increase these *sequence numbers* without preventing the EUs, and the communication would not be interrupted.

There are three ways to avoid sequence number values. First, in the case that the newly active SG keeps an IPsec session from another SG, it may send an IKEv2 message to the EU in order to update both the Message ID and the sequence number value. However, this solution requires to create a new notify payload and thus to modify the IKEv2 protocol itself. Indeed, [5] proposes this method of resynchronization.

The second approach is a client unaware method. It consists in creating a cluster of IPsec SGs which maintain synchronized Message ID values as well as sequence number values. These latter can be synchronized by implementing ClusterIP (see sectionřefsec:clusterip).

The third approach concerns the sequence number values only. When a newly active SG takes responsibility of a tunnel, it may skip a big number of sequence number values and the IPsec session will still not be interrupted. Note that the IPsec protocol allows both endpoints to skip some sequence number values without prior agreement.

IV. Clustering Methods for IPsec

This section positions Load-Balancing facing High Availability, details how takeover may impact the client and finally describes ClusterIP configuration and operation with IPsec.

A. Load-Balancing Versus High-Availability Clusters

A Load-balancing cluster is a set of nodes where more than one of the members may be active at the same time. Load-balanced clusters are implemented by sharing the workload between cluster nodes and offering better performance. HA-clusters operate by having redundant nodes intended to provide a service when other node fails. For Linux, the latter has been implemented using a free software package developed by the Linux-HA project, having the heartbeat software as the main product. Heartbeat automatically monitors resources so that they can be restarted or moved to another node on failure.

B. ClusterIP Implementation

When using a ClusterIP approach, no special hardware is required to benefit from Load-Balancing. Indeed, ClusterIP is intended to provide load-balancing features without having a load-balancer. The configured members of the cluster does share a multicast MAC address and thus receive the same packets. Then, a lower-layer mechanism (netfilter code) on each node, filters packets by calculating responsibility through an algorithm (E.g hashing the IP source of each packet). When applied, ClusterIP acts as a parameter for the `iptables` command.

The nodes in a cluster usually have two Network Interface Cards (NIC). One of the NICs MAC address is replaced by the shared cluster MAC address and then a common virtual IP address is mapped onto it. The other NIC, being completely independent, can be used for any other purpose, as for example inter-nodes communications (E.g. Hearbeat mechanism). Given the case where a machine counts with only one NIC, it is also possible to install a second virtual IP address on the same interface.

1) ClusterIP & IPsec : Originally, ClusterIP does NOT handle IPsec traffic. In fact, a given IPsec communication is associated to counters, and thus must be treated by a single s. Two SGs cannot handle a given IPsec communication unless the two SGs share a common IPsec context for the same communication. However, if an IKE daemon (E.g. strongSwan) handles to synchronize IKE_SAs and IPsec_SAs states, a modified version of ClusterIP that handles IPsec traffic could solve synchronization troubles; but the overhead for synchronizing ESP sequence numbers could be very high. Thus, deploying IKEv2 and IPsec in a cluster requires the synchronization of a large amount of information among all the cluster members. On the other hand, if less information is synchronized, fail-over would take longer to perform. As stated in III-A, synchronizing counters might be the major barrier to overcome when it comes to setting up a cluster with IPsec. Some states involved in an IKEv2/IPsec session establishment are long lasting:

- *IKE Security Associations*: a SG may establish hundreds or thousands of IKE_SAs. Also, they may live for several minutes, hours, or days. They contain keys, selectors and other information concerning IKE traffic.
- *IPsec Security Associations*: a SG may establish hundreds or thousands of IPsec_SAs. Also, they may live for several minutes, hours, or days. They contain keys, selectors and other information concerning IPsec traffic.
- *Security Policy Database SPD*: they may live as long as an IKE_SA but they also tend to live longer in some operative systems.

IKE Counters (Message ID Counters) are the longest living states but at the same time are required to synchronize less often since synchronization might only occur whenever an IKE_SA is created or some INFORMATIONAL or REKEY exchange occurs. However, IKE needs to update the Message ID Counter immediately, as processing a message having a higher ID is not allowed (see III-A). This is achieved by synchronizing IKE message counters after every single IKE exchange.

Concerning the anti-replay counters, every ESP/AH protected packet carries a *sequence number* that cannot be reused since the anti-replay feature would consider it as an attack, leading to drop all the packets and issuing attack warnings. Synchronizing anti-replay IPsec counters is not reasonable neither, due to the high load introduced (for each packet emitted). As a result, the designed solution synchronizes the counters every n-th packets (10.000 packets). This choice is justified since skipping *sequence numbers* is allowed in IPsec, and highly reliable delivery service (as in IKEv2) is not provided.

V. StrongSwan's HA Plugin

StrongSwan is a complete OpenSource IPsec-based VPN Solution for Linux operating systems. Its High-Availability plugin implements a ClusterIP-based mechanism that is able to maintain IKE_SAs and IPsec_SAs in case of failover. The current release strongSwan 5.x supports clusters of two nodes maximum. This section explains how the HA plug-in achieves active/active High Availability and Load Sharing capabilities.

The IKE daemon of strongSwan synchronizes the IKE state and the basic IPsec SA state without *Sequence Numbers*. The remaining tasks are carried out by a modified ClusterIP plug-in called High Availability (HA). The HA plug-in requires two patches against the kernel in order to allow ClusterIP to work with IPsec. These patches modify the ClusterIP netfilter module, more specifically, the PREROUTING hook that marks received packets for forwarding before the decryption/encryption process. A third patch is required to modify the Linux firewall (iptables) in order to work over the patched kernel.

A. IKE Daemon Implementation

- *Daemon Hooks*: a hook is a function that is in charge of collecting information (in this case Synchronization data) for later use in preparing messages that are going to be sent to other members in the cluster in order to notify SA state changes or pushing information towards the plugin. Hooks are created and registered by the plugin at the daemon bus. Table I shows the hooks used by the HA plugin.
- *Synchronization messages*: table II shows the different synchronization messages types that can be exchanged between nodes in a cluster according to the implemented HA plugin of strongSwan. Messages are sent with no encryption by the hook functions using UDP datagrams on port 4150. However, an IPsec tunnel could be established in order to transmit this information.
- *State synchronization*: state changes are executed by Synchronization messages exclusively. They carry all the information required to create a duplicate of the active node IKE_SAs and IPsec_SAs. Duplicated IKE_SAs do not handle traffic and are installed in a PASSIVE state while duplicated IPsec_SAs are installed in the Kernel and subjected to ClusterIP algorithms.
- *Control messages* : table III shows the different control messages implemented by the HA plugin. These messages are sent along with the synchronization messages with the purpose of notifying segment responsibility changes.
- *Failover*: the state of a segment in a cluster is set by the HA plugin to either ESTABLISHED or PASSIVE. This is decided on each node using the same ClusterIP hashing function based on the source IP address. By using the same hashing function it guarantees that the cluster responsibility will not be assigned to both nodes at the same time. The activation/deactivation of a segment is performed over all the IKE_SAs that are found on that actual segment. There is no impact on their IPsec_SAs, they are always active.
- *Node reintegration*: reintegration is meant to take the failed node after its recovery and reinserting it into the cluster as a backup node again. The recently reincorporated node needs to fully synchronize the state information; this is achieved by pushing all the active IKE_SAs messages, cached in the active node, onto the newly arrived node. Synchronizing IPsec_SAs is not possible using the cache, as the messages do not contain *Sequence Number* information managed in the kernel. To reintegrate a node, the active node initiates rekeying on all IPsec_SAs.

Hook	Description
ike_keys( )	Receives IKE key material (DH, nonce, proposals)
ike_updown( )	Monitors state changes of IKE_SAs
message( )	Used to update IKE *Message IDs*
child_keys( )	Receives CHILD key material
child_state_change( )	Monitors state changes of IPsec_SAs

TABLE I
HOOKS USED BY THE STRONGSWAN'S HA PLUGIN

Synch Message	Description
IKE_ADD	Message used when a new IKE_SA is established. It contains all information to derive keys
IKE_UPDATE	Message used to update information of a concerned IKE_SA, for example, when authentication is done
IKE_MID_I	Updates the *Message ID* of the initiator
IKE_MID_R	Updates the *Message ID* of the responder
IKE_DELETE	It is used to delete a corresponding IKE_SA
CHILD_ADD	Message used when adding a new IPsec_SA
CHILD_DELETE	Message used to delete an IPsec_SA

TABLE II
SYNCHRONIZATION MESSAGES OF THE HA PLUGIN

VI. Performance Tests & Results

A. Testbed description

Our performance tests are conducted in two different topologies, one with HA-plugin enabled (i.e. based on ClusterIP) and the second one with no HA features (i.e. no ClusterIP) to compare how ClusterIP improves the performances. The first scenario, shown in figure 2a, counts with an HTTP server, an IPsec peer and two VPN SGs; strongSwan is configured to provide High-Availability cluster

Control Message	Description
SEG_DROP	Message to drop responsibility of segments
SEG_TAKE	Message to take responsibility of segments
STATUS	Heartbeat mechanism to prove liveness and segment responsibility
RESYNC	Used to resynchronize a list of segments

TABLE III
CONTROL MESSAGES FOR SEGMENT CHANGES NOTIFICATION

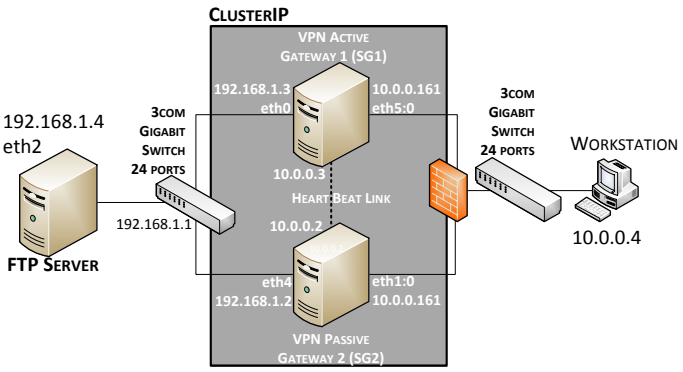
between the SGs and its members keep in synch through the Heart Beat link (Synch Channel). In the second topology, illustrated in figure 2b, the whole traffic goes through a single active SG, meaning that strongSwan is used as a VPN solution but with no fail-over node, thus the HA plugin is not loaded.

The results are represented in graphs with box-and-whiskers style. This kind of representation is mostly used to plot statistical data. For every measurement made (>1000 samples per measure), the box-and-whisker plot indicates: (i) the smallest observation, (ii) the lower quartile, (iii) the upper quartile, (iv) the largest observation and (v) the median. The space between the lower and upper quartile represents 50% of the samples. For more clarifications, refer to [13].

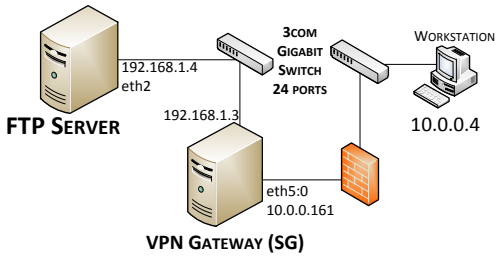
During the tests, we used two different implementations (`time` and `top`) in order to measure the IPsec performance under different circumstances. The command `time`, launches the specified program command with given arguments. When the command has finished to run, `time` writes a message to the standard output giving timing statistics about this program run. The outputs of `time` are (i) the *elapsed real time* between invocation and termination of the command, (ii) the ***user CPU time*** or ***cpu-us*** spent executing instructions of the calling process and (iii) the ***system CPU time*** or ***cpu-sys*** spent in the system while executing tasks on behalf of the calling process.

The command `top`, provides the real-time CPU activity. It shows a list of the ongoing system tasks . We identified the IPsec related task ID within this list and we collected all the information concerning the CPU consumption during the tests. All tests were done using different number of CPUs in order to compare the impact of having several CPUs sharing the workload (1,2,3 or 4 CPUs).

1) First Test - ClusterIP overhead Measurements Test: The purpose of this test is to evaluate how much CPU resources and time the modified ClusterIP adds to the whole VPN service. Note that when the HA plugin of strongSwan is activated (i.e. 2a), for each incoming packet the modified ClusterIP hashes the IP header to check whether or not the SG is the



(a) Scenario using ClusterIP



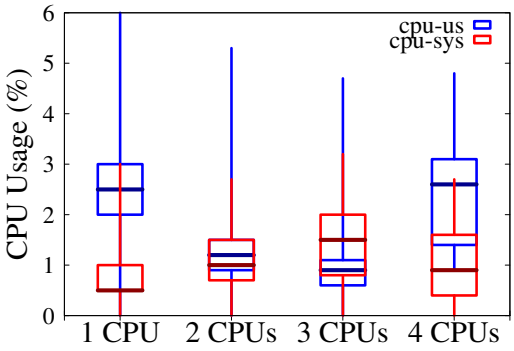
(b) Scenario NOT using ClusterIP

Fig. 2. Scenarios

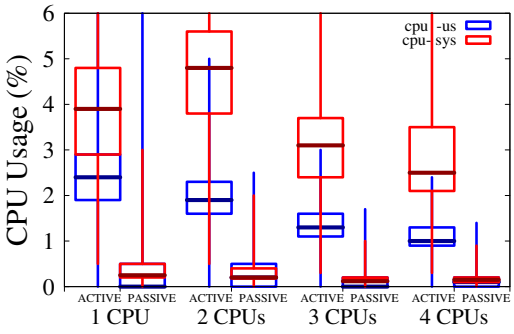
responsible node to handle that packet.

This test is performed using both topologies described in figures 2a and 2b, thus comparing the impact of ClusterIP. The test consists of downloading a file of $1GB$ from an HTTP server (placed behind the SG) towards the peer by using the command `wget` on the peer's side. During the tests based on figure 2a, the HA plugin is enabled whereas during tests as in figure 2b the HA plugin is deactivated. We measured the CPU impact and the time spent by the system to complete an HTTP download (using the `wget` command). The IKEv2 exchanges are always protected with AES128-SHA1, whereas two different algorithms for ESP encryption were also analyzed throughout the test: AES128-SHA1 and NULL-SHA1 (where NULL means no encryption, note that strongSwan does not support no integrity check and it is always SHA1). We also varied from one (1) to four (4), the number of CPUs available on each SG; this allows analyzing the evolution of the CPU consumption of both types of encryption. As mentioned in VI-A, the CPU consumption and the download time (user and system time) are obtained through the `top` and `time` command respectively.

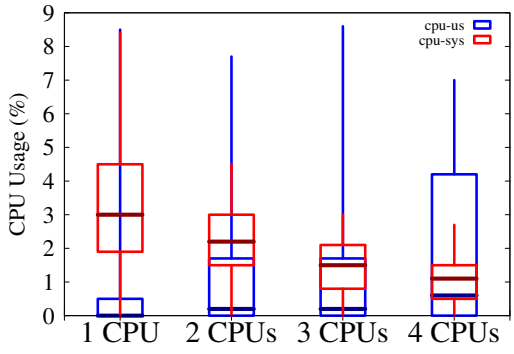
CPU Consumption: figures 3a and 3c represent the CPU usage with no HA features. They implement different encryption algorithms. Figure 3c uses NULL encryption whereas figure 3a uses AES128. When NULL encryption is used, packet treatment is improved and the CPU consumption decreases around 30% . The CPU consumption of the userland is practically the same for both cases. So, NULL encryption might improve CPU performance but it also might



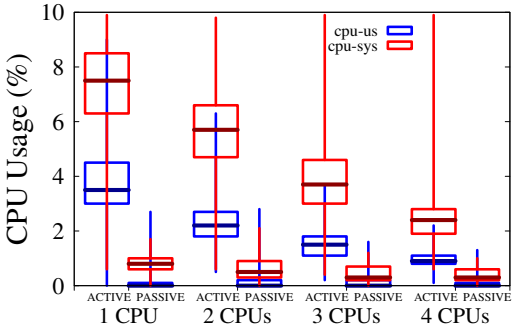
(a) Enc:AES128 - No HA-plugin



(b) Enc:AES128 + HA-plugin



(c) Enc:NULL + NO HA-plugin



(d) Enc:NULL + HA-plugin

Fig. 3. First Test: Download Performance Test (CPU Usage)

downgrade the security level as the flow is integrity protected but not encrypted.

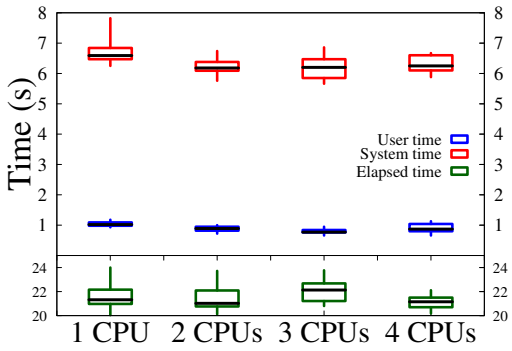
Considering AES128 and NULL encryption but with HA-plugin activated, figures 3b and 3d illustrate the CPU consumption and show with different active CPUs a 3 to 8% CPU Usage. AES128 registers more activity (3% to 8% of CPU Usage) than a node using NULL encryption (2.5%to 5% of CPU Usage). Once again, the CPU consumptions of the userland have the same behavior for both scenarios. As in AES128 with no HA-plugin, a peak is observed in the case where 2CPUs are used. We observe that the ClusterIP-based plugin is not well suited when 2CPUs are being used.

Download Time: Results concerning the HTTP download time are shown in figures 4a, 4b, 4c and 4d. Two main scenarios are compared: the impact of HA-plugin on the download time and the impact of the encryption method used during these downloads. When no HA-plugin is used, AES-128 encryption introduces more System Time than when NULL encryption method is used. The System Time and the Elapsed Time (total time to complete the download) increase by 11% using AES128. However, no variation is observed in the User Time, which means that the operating system always performs the same user mode tasks. Note that the modified ClusterIP requires the kernel to be patched in order to allow IPsec packet filtering. No variation of the User Time was observed when using the HA-plugin. It always stayed around $1s$ in all scenarios of the first test. Finally, when comparing the impact of the encryption methods showed that a cluster with AES128 takes around 30% more time to download than a download that uses NULL encryption.

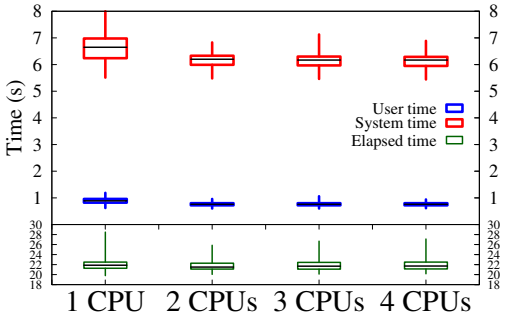
2) Second Test - QoS on an HTTP connection: The second test is evaluates the quality of service (QoS) ensured by the HA plugin in terms of upper-layers (E.g. HTTP-based downloads) reactivity. The test consists of downloading a file of size 50MB from the HTTP server towards the VPN peer. On the client side, we measured the Elapsed Time (obtained via the command `time`), which represents the time to complete the download. We do the same for both topologies (with & without HA-plugin, figures 2a and 2b correspondingly), thus comparing the impact of the HA-plugin and added overhead during the download.

Both scenarios consider the encryption and integrity protection with AES128 and SHA1 algorithms respectively. With the ClusterIP-based plugin activated, both figures illustrate the time to download a file of size $50MB$. Figures 5a, 5b show the download time through Ethernet connections. The User Time stays invariable.

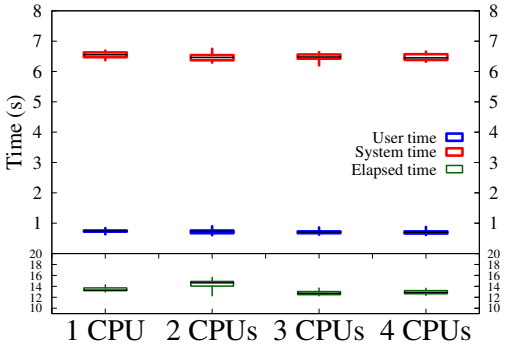
On the other hand, differences are observed when the HA-plugin is activated. The fact to decide either to treat or not a packet (by filtering at the IP layer), increases the System Time by 35%. Also the Elapsed Time increases by 25%, which introduce some overhead.



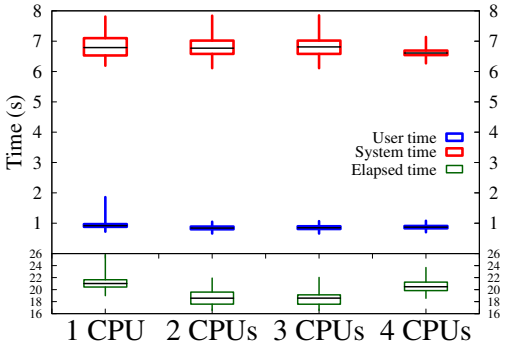
(a) Enc:AES128 - No HA-plugin



(b) Enc:AES128 + HA-plugin

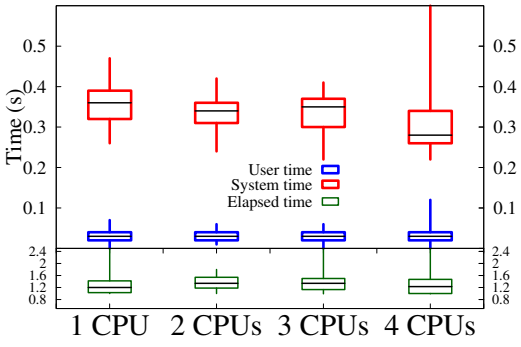


(c) Enc:NULL + NO HA-plugin

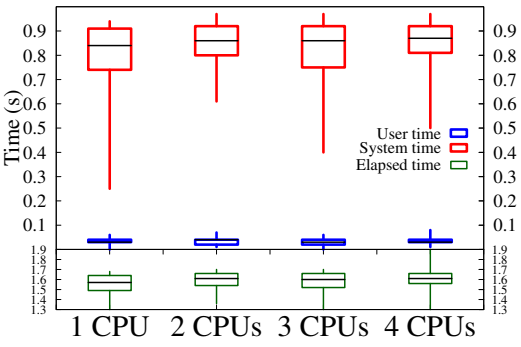


(d) Enc:NULL + HA-plugin

Fig. 4. First Test: Download Performance Test (time)



(a) Enc:AES128 + NO HA-plugin



(b) Enc:AES128 + HA-plugin

Fig. 5. Second Test - QoS on an HTTP Connection

Throughout this test, the fact of using 1,2,3 or 4CPUs did not impact noticeably the download time. For example, figures 5a and 5b show that the behavior of the download time is very similar for each CPU scenario. Finally, as we can see, upper-layers might be impacted when this HA feature is activated. Administrators wanting to implement a cluster of VPN SGs should justify the need for this solution. A typical example is an environment where thousands of tunnels should be established, so letting the cluster to spread responsibility over different nodes, and thus spreading the load on each SG. Even if the load for a single connection would be slightly higher by using the HA-plugin, a real positive impact would be noticeable only when a large number of VPN connections are spread among the cluster members.

3) Third Test - Interruption Time of an HTTP communication: When a failure event occurs, the ClusterIP module used by strongSwan allows the cluster to switch transparently from the active node to the passive SG node. We concentrate on evaluating the handover network time (see figure 1b) during fail-over. The scenario is illustrated in figure 2a. The test consists of downloading a file of size $500MB$ from the HTTP server towards the VPN peer. After five 5 seconds of download, we interrupt the outgoing network interface of the active VPN SG, causing a failure event. The passive SG of the

cluster stops receiving responses through the Synch Channel, and thus takes responsibility of the VPN tunnel. We captured the traffic between the VPN peer and the cluster during the test. Figure 1 represents some protected ESP traffic at the IP layer. The period of time between the last ESP packet emitted by the active SG (SG1) and the first ESP packet emitted by the passive SG (SG2) is considered as the *Handover Time*. This time corresponds to the network delay of takeover from the active node to the passive SG.

Figure 6 represents the results obtained when measuring the *Handover Time*. The quartiles illustrate the download time. The black colored quartiles shows the download time during a fail-over event, taking $3-4$ seconds more than a download without interruption (colored in red).

Note that the handover time (at the right-bottom and right-upper side) are illustrated with three vertical lines that represent the lower quartile, the median and the upper quartile. The handover time with no traffic control varies from $1.63s$ to $1.66s$ (right-bottom figure). This time is considered as network-friendly time to perform a fail-over. However, the difference between the download time on both scenarios (with and without fail-over) stays over $3s$, $1.7s$, $3.2s$, and $3.1s$ for 1,2,3 and 4CPUs respectively. This overhead or difference is due to updating the IPsec databases. An update action is blocking the database and thus blocking the communication. This delay is expected to increase as the number of tunnel increases. Upper-layers treatment add more delay to the communication as well the system also takes some time in order to accomplish all tasks. The results are compliant to the expected $1s$ to $3s$ performances as required in strongSwan's specifications. We also tested an additional *Handover Time* where the bandwidth limit is imposed to $2MBps$, emulating the use case of a RAN (Radio Access Network). Once again, the network fail-over time remains between $1.9s$ and $2.3s$. The reason is that with reduced rates, the platform is not overloaded. The impact of the number of CPUs cannot be measured because, the time between heartbeat exchanges are dominant. Nevertheless, the *No Failover* case (colored in red), shows that the ClusterIP module has better performance when 1CPU or 4CPUs are being used. In terms of QoS, it should be considered only to use 1CPU or 4CPUs , instead of 2CPUs or 3CPUs.

VII. Conclusions

Throughout this article, we measured the impact and performance of using a modified ClusterIP module in order to clusterize SGs over IPsec. The availability of the IPsec service is improved thanks to the hot-standby clustering ensured by ClusterIP. Also, *strongSwan* guarantees the continuity of an IPsec session thanks to its HA plug-in allowing to spread all the IPsec tunnels among its cluster members. Results showed that an active SG spends 5% to 8% of CPU more than a passif SG when clustering IPsec tunnels. We also observed that there is an additional load when using the HA-plugin of strongSwan among the cluster members. Downloading a $500MB$ file takes 25% more time when using

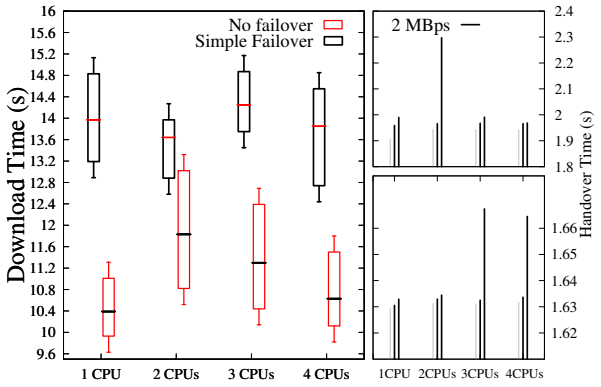


Fig. 6. Third Test: Handover Time

ClusterIP and the HA plug-in. As such, an administrator willing to implement this solution should take into account the performance costs of adding HA features to its VPN service.

Furthermore, one main drawback of ClusterIP is its limitation to be deployed within a same network segment. Further investigation will consider transferring an IPsec/IKEv2 context between two SGs owning different IP addresses. Finally, we will also consider stress-tests by using a new *load-tester* plug-in developed by *strongSwan*. Future work will concentrate on using our own VPN tunnel management tool that allows IPsec transfer between SGs with different IP addresses.

References

[1] 3GPP-LTE, "3gpp system to wireless local area network (wlan) interworking; system description, ts 23.234, release 10," Standard, Mar. 2011.

[2] A. Steffen., "the OpenSource IPsec-based VPN Solution: *StrongSwan*." [Online]. Available: http://www.strongswan.org

[3] L. Yu, S. Jia, C. Xu, J. Guan, and D. Gao, "An ipsec seamless switching mechanism with high availability and scalability by extending ikev2 protocol," *IET Conference Publications*, vol. 2011, no. CP588, 2011.

[4] R. Hinden, "Virtual Router Redundancy Protocol (VRRP)," RFC 3768 (Draft Standard), Internet Engineering Task Force, Apr. 2004, obsoleted by RFC 5798. [Online]. Available: http://www.ietf.org/rfc/rfc3768.txt

[5] R. Singh, G. Kalyani, Y. Nir, Y. Sheffer, and D. Zhang, "Protocol Support for High Availability of IKEv2/IPsec," RFC 6311 (Proposed Standard), Internet Engineering Task Force, Jul. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6311.txt

[6] P. Eronen, "IKEv2 Mobility and Multihoming Protocol (MOBIKE)," RFC 4555 (Proposed Standard), Internet Engineering Task Force, Jun. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4555.txt

[7] D. Migault, "MOBIKE eXtension (MOBIKE-X) for Transport Mobility and Multihomed IKE_SA," (Work in Progress), Internet Engineering Task Force, Sep. 2009. [Online]. Available: http://www.ietf.org/html/draft-mglt-ipsec-mm-mobikex-00

[8] ——, "IPsec mobility and multihoming requirements : Problem statement," (Work in Progress), Internet Engineering Task Force, Sep. 2009. [Online]. Available: http://www.ietf.org/html/draft-mglt-ipsec-mm-requirements-00

[9] D. Migault and C. Williams, "Multiple Interfaces IPsec Security Requirements," (Work in Progress), Internet Engineering Task Force, Mar. 2012. [Online]. Available: https://datatracker.ietf.org/doc/draft-mglt-mif-security-requirements/

[10] J. Loughney, M. Nakhjiri, C. Perkins, and R. Koodli, "Context Transfer Protocol (CXTP)," RFC 4067 (Experimental), Internet Engineering Task Force, Jul. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4067.txt

[11] F. Allard, J.-M. Bonnin, J.-M. Combes, and J. Bournelle, "IKE Context Transfer in an IPv6 Mobility Environment," in *MobiArch'08, 22 août, Seattle (WA), USA*. FT - France Télécom, Division R&D, Issy Les Moulineaux (France Télécom), RSM - Dépt. Réseaux, Sécurité et Multimédia (Institut Mines-Télécom-Télécom Bretagne-UEB), 2008.

[12] D. Palomares, "Mechanisms to Ensure Continuity of Service for *IPsec/IKEv2* Based Communications," in *ICSNA-2011: International Conference on Secure Networking and Applications ,24-25 October, Paris, France*. FT - France Télécom, Division R&D, Issy Les Moulineaux (France Télécom), 2011.

[13] Wikipedia, "Boxplot — Wikipedia The Free Encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Box_plot