MUSeS: Mobile User Secured Session

Daouda Ahmat, Damien Magoni

MUSeS: Mobile User Secured Session

Daouda Ahmat
University of Bordeaux – LaBRI
Talence, France
adaouda@labri.fr

Damien Magoni
University of Bordeaux – LaBRI
Talence, France
magoni@labri.fr

***Abstract*—Mobility and security are very important services for both current and future network infrastructures. However, the integration of mobility in traditional virtual private networks is difficult due to the costs of reestablishing broken secure tunnels and restarting broken application connections. In order to address this issue, we propose a new communication system called Mobile User Secured Session. Based upon a peer-to-peer overlay network, it provides security services to the application layer connections of mobile users. The secure and resilient sessions allow user connections to survive network failures as opposed to regular transport layer secured connections. We have implemented a prototype and have assessed its proper functioning by running experimentations upon a simple virtual dynamic network.**

I. Introduction

Mobile equipments and wireless networks have progressively provided a high connectivity for user connections. However, such connectivity often comes at the expense of vulnerabilities to attacks such as eavesdropping. Malicious users can infiltrate untrusted dynamic networks and attack communication between legitimate users. Virtual Private Networks (VPNs) are a technology which offers a high security to the traffic [1]. Traditionally, these systems allow the user to securely and remotely communicate with its Intranet through insecure networks such as the Internet. Security services provided by these infrastructures are very robust against malicious users' attacks. However, traditional VPNs fail to support the mobility of users. Indeed, network failures automatically break-up secure tunnels and involve a subsequent negotiation that is then necessary to reestablish broken tunnels. This negotiation requires expensive computational operations in order to restore tunnels as well as transport and application layers' connections. This phase of negotiation causes not only significant latency but also presents risks of Man-In-The-Middle attacks. Traditional VPNs can not therefore effectively operate in dynamic environments.

In this paper, we propose a new mobile VPN system called Mobile User Secured Session (*MUSeS*). This system is designed to support both the mobility and the traffic security of the user. *MUSeS* allows user connections to survive disruptions caused by network failures or location changes.In addition, our infrastructure has the ability to hide these disruptions from the user. In this paper, our main contributions are structured as follows:

- We first describe the design principles and architecture of our system (Section II).
- We then experiment our secure session resilience mechanism in a virtualized dynamic environment (Section III).
- We finally provide a background summary of prior and related work (Section IV).

II. System Design

A. Overview

As shown in Figure 1, *MUSeS* is a communication system directly used by the applications. *MUSeS* is using another communication system called CLOAK which enables the creation and management of a P2P overlay network above any IP network [2]. *MUSeS* is designed to provide traffic security and session continuity. Applications produce data and send it to the *MUSeS* middleware. Data is then sliced in packets that are encrypted and authenticated. Unlike IPsec or TLS which rely on IP addresses to define the identity of the communication entities, *MUSeS* relies on permanent device identifiers called *names* provided and managed by CLOAK. The *MUSeS* system has the ability to support several simultaneous applications. In addition, any application setting up a *MUSeS* session is able to survive to network interruptions of any duration. It can keep a user session active after a connection disruption and can resume an interrupted session when the failed connection is reestablished. Session resumption is performed without any session key renegotiation.

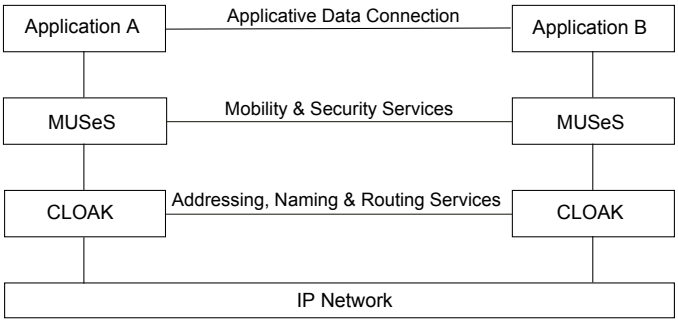

Fig. 1. Design overview.

B. CLOAK Overlay

CLOAK is an architecture for building P2P overlay networks which are autonomous and dynamic. A network node can join any peer already inside the overlay usually by setting up a transport layer connection such as TCP with this peer.

The new peer is then offered an address which is unique and dependent on its location in the overlay. These addresses are used by a greedy routing algorithm for forwarding the data inside the overlay. In order to cope with network dynamics, CLOAK provides a distributed algorithm for maintaining the consistency of the assignment of the addresses to preserve the greedy routing property. Because an address depend on a peer's location, it changes when the peer disconnects from its neighbors and reconnects to new neighbors. To ease communication between peers, CLOAK assigns to each peer a name which remains permanent during the lifetime of the overlay. In order to be autonomous, CLOAK provides a DHT for mapping the names of the peers to their addresses. These names are used by *MUSeS* to setup a secure connection between two peers.

CLOAK was originally presented in our paper [3] which contained an extensive amount of background and related work as well as some preliminary simulation results upon static networks concerning path length. Improving upon this foundation, our paper [2] presented the protocols and modules of the architecture with greater details and reported simulation results upon dynamic networks concerning routing success ratio, path length and stretch, as well as DHT performances. The addressing and routing system based on hyperbolic geometry which is used by CLOAK was presented in our paper [4]. Both the distributed addressing algorithm and the greedy routing algorithm are detailed in this previous paper and we have not included them here for the sake of brevity. The DHT scheme defined by CLOAK over this hyperbolic system is fully explained in our paper [5].

The Figure 2 shows how a packet is forwarded between the source and the destination. When a secure *MUSeS* node *A* generates a packet to send to a distant node *B*, it locally forwards the packet to the associated underlying CLOAK node *A'* via a loopback TCP connection. Next, the underlying node *A'* routes the packet to the destination through the P2P overlay network consisting in chained TCP connections. The CLOAK node *B'*, associated to the final destination *MUSeS* node *B*, intercepts the packet and locally forwards it to *B*. The P2P overlay network ensures therefore the proper routing of *MUSeS* secured packets over the network.

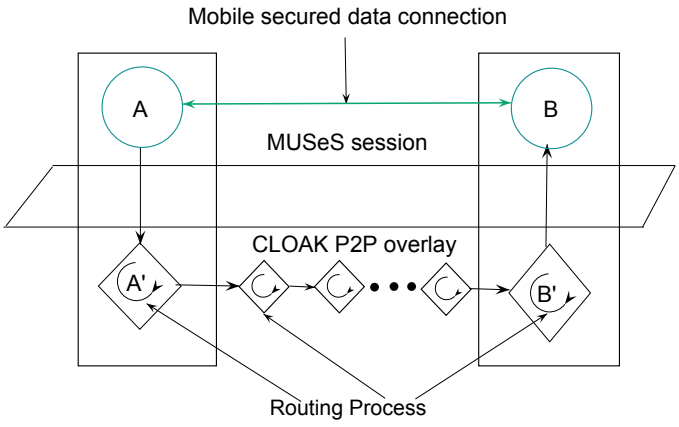

Fig. 2. Routing through the P2P overlay network.

C. Architecture

The *MUSeS* system is based on three main components:

- The Session Security Module (SSM) manages security services such as confidentiality, integrity and protection against replay attacks.
- The Session Reliability Module (SRM) is designed to provide fault-tolerant user sessions. In other words, SRM offers user session continuity service that survives despite connection disruptions caused by lower layers.
- The Local Connection Manager (LCM) handles multiple local TCP connections in order to support several simultaneous user applications.

1) Session Security Module (SSM): The *MUSeS* security module consists mainly in encrypting and decrypting traffic payload as well as checking the integrity of the persistent fields (sequence number, source and destination identifiers, source and destination ports, ...) of each packet. All send/receive operations between two users are performed through a secure end-to-end tunnel. The SSM component protects the user's traffic against eavesdropping, modification attacks and replay attacks. The goal of these protections is to prevent malicious users from attacking legitimate users' traffic.

Precisely, when Alice and Bob communicate, the data exchanged by the two correspondents is encrypted and authenticated. In other words, when Alice wants to send data to Bob, for each packet that she generates, Alice encrypts the payload and authenticates the encrypted payload and persistent fields (SrcID, DstID, SessID, SeqNBR, ConTYPE, ...) of the packet header. The authentication consists in computing a digest on both the invariant fields and the encrypted payload. The Integrity Check Value (ICV) thus obtained is added to this packet.Before decrypting a received packet, Bob computes an ICV and compares it with the ICV fixed on this packet by Alice in order to check both authenticity and integrity. If the check is successful, then Bob decrypts the payload of the packet. Otherwise, the packet is assumed to be corrupted and it is automatically destroyed. If the decryption process fails, the packet is also destroyed.

The Routing Header (RH) used by the CLOAK overlay is neither authenticated nor encrypted by *MUSeS*. Thus, each intermediary overlay peer can update these values so as to route packets towards the destination user. In order to avoid replay attacks, SSM separately identifies each packet by a sequence number (*SeqNBR*) and a session identifier (*SessID*). The combination *SeqNBR* • *SessID* is unique to a *MUSeS* connection. These values are authenticated because they belong to the persistent fields. *SeqNBR* and *SessID* are therefore free of modification attacks.

2) Session Reliability Module (SRM): The Session Reliability Module or SRM is a *MUSeS* module which manages the session continuity mechanism. SRM is able to detect network failures and to suspend a communication for a long time. It can keep interrupted sessions active and can reestablish them subsequently. In addition, SRM permits also to hide disruptions caused by lower layers to the application layer.

Therefore, SRM has the ability to make the failures occurred at lower layers transparent to the users.

a) Interruption Detection: When a connection fails, SRM has the ability to detect it. The destination unreachability detection is made possible by an acknowledgment mechanism that is based on the exchange of *request-reply* messages. When there is a packet loss, detected by a lack of acknowledgment or a waiting timeout, the communication will be temporarily suspended.

Each peer concerned by an interrupted session attempts to reestablish the connection with the corresponding peer. Its sends *Re-Hello* packets at regular intervals of time to attempt to restart the disrupted session as shown in Figure 3. In order to check remote peer connectivity, *Re-Hello* messages contain, among other fields, the sequence number of the last packet correctly received or sent. The first peer that receives *Re-Hello* request can reply to this message. When a peer replies to a *Re-Hello* message, disrupted session is therefore resumed.

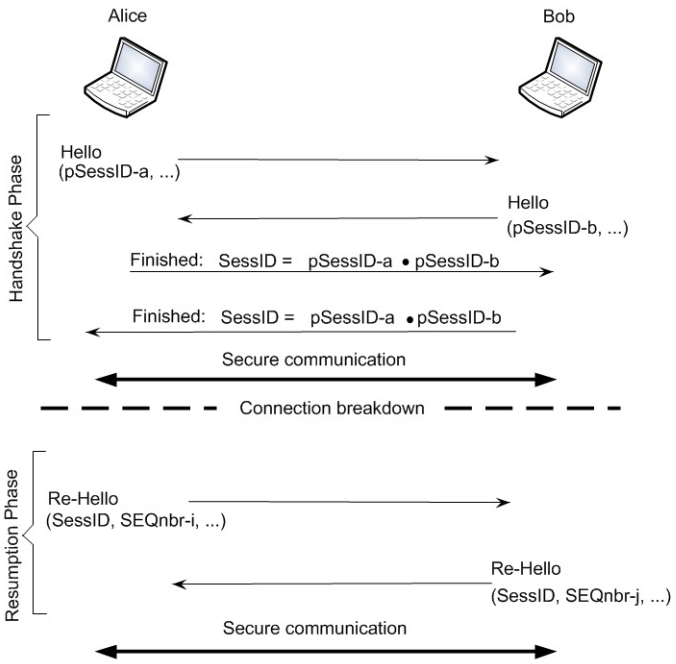

Fig. 3. Messages exchanged in the handshake and resumption phases.

b) Session Identifier: When a mobile *MUSeS* entity wishes to communicate with a remote corresponding node, these two peers compute together, in the *handshake* phase, a unique value in order to singly identify each session following the same principle as with a socket connection. When *Alice* contacts *Bob* to communicate (*handshake phase*), *Alice* generates a partial session ID (*pSessID-a*) and sends it to *Bob* as shown in Figure 3. After receiving *pSessID-a*, *Bob* generates also a partial session ID (*pSessID-b*) and sends it to *Alice*. Each corresponding determines the final session ID (*SessID*) by concatenating the partial session IDs: $SessID = pSessID\text{-}a \bullet pSessID\text{-}b$. To close this phase, *Alice* sends a *Finished* message to *Bob*, and he sends the same message to *Alice*. The communication, identified by a *SessID*, can be therefore started between the two actors. Precisely, a partial session ID (*pSessID*) is essentially computed by a combination between a source port number (*psN*) and a peer ID (*pID*): $pSessID = psN \bullet pID$.

c) Acknowledgment Scheme: Based on a cumulative acknowledgment scheme such as in the TCP protocol, the *MUSeS* system sends one reply message when *n* packets have been sent in order to optimize network bandwidth. When the sender node sends *n* successive packets but it does not receive an acknowledgment message from the receiver node before the expiration of timeout, the corresponding node assumes that the underlying CLOAK connection has failed and pauses the traffic sending for the time necessary. The established *MUSeS* session is kept active by the two correspondents until the disrupted CLOAK path is restored. The acknowledgement scheme is presented in Algorithm 1.

Algorithm 1 Acknowledgement Scheme.

Require: $Max\_i \geq 0 \wedge Max\_j \geq 0$
  repeat
    $i \Leftarrow i + 1 \quad j \Leftarrow 1$
    pkt = packet.next_packet()
    send_packet(pkt)
    if $i = Max\_i$ **or** pkt.flag_last = **true then**
      $r \Leftarrow$ recv_ack_packet()
      while $r \in \{timeout, \neg ok\}$ **do**
        if $j = Max\_j$ **then**
          session_close()
          exit()
        else
          pause()
          r $\Leftarrow$ recv_ack_packet()
          $j \Leftarrow j + 1$
        end if
      end while
      $i \Leftarrow 0$
    end if
  until pkt.flag_last = **true**

d) Secure Mobility: In our model, the concept of secure mobility is a way to allow mobile users to keep an established secure session active when the underlying system connection fails until a subsequent new connection is started. In other words, when a user changes its location as shown in Figure 4 or when its connections are disrupted by network failures, the user session survives as long as necessary to resume the communication. This ability prevents *MUSeS* connections from breaking when the underlying topology changes.Therefore, the session reestablishment is free from renegotiating security parameters. To reestablish an interrupted communication, the corresponding peers just need to exchange some session resumption messages shown in Figure 3 in order to restart

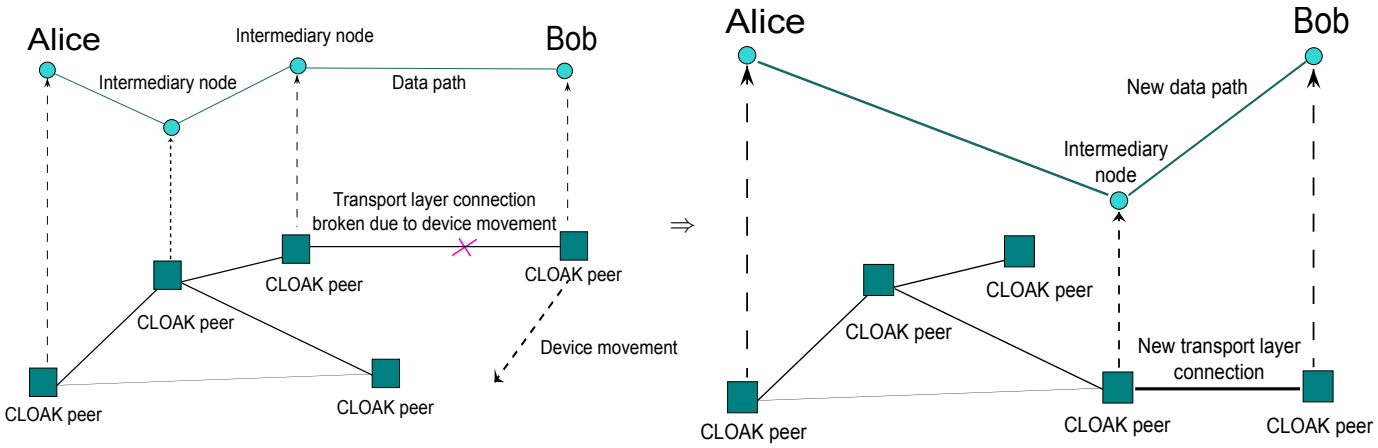

Fig. 4. User mobility with ongoing secure session.

the interrupted session at the point where the session was suspended.

3) Local Connection Manager (LCM): The *MUSeS* system is able to simultaneously support several user applications. The *MUSeS* system and the user applications communicate through local TCP sockets. Local TCP connections are managed by the Local Connection Manager (LCM) as shown in Figure 5. Each user application is separately identified by its source port. When data is received, the LCM module locally forwards the data to the expected user application. The LCM builds a hash table that associates each socket descriptor (value) with a socket source port number (key). Socket source port numbers are generated in a unique way by the OS. Source port numbers contained in the hash table allow the LCM to identify the local TCP socket descriptors corresponding to the proper destination applications. When the SRM detects a connection disruption or fails to reach the remote peer, it automatically notifies the LCM. Then, the LCM suspends data reception from user applications until the connection reestablishment. The local socket is never closed by *MUSeS* (and it normally never fails) unless the user wants to close it explicitly.

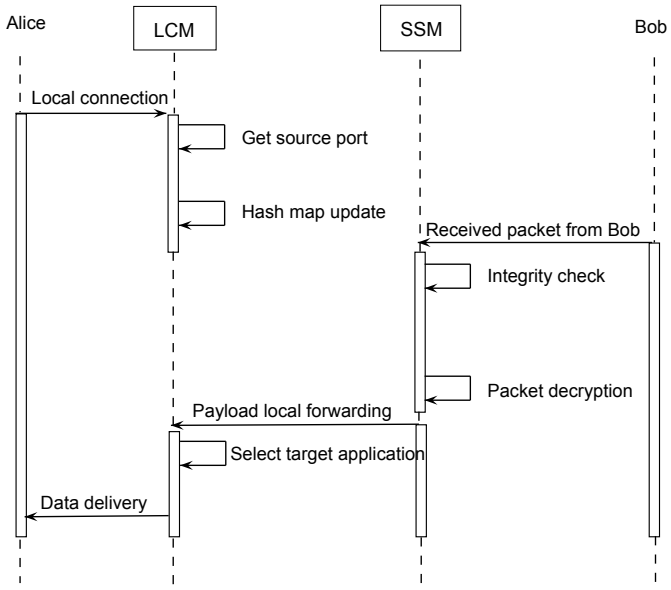

Fig. 5. Interaction between user applications and *MUSeS*.

D. Security Analysis

Based on the use of standard cryptographic algorithms, our system protects user communications from common traffic attacks. The authentication of user applications is done through public key cryptography where the CLOAK DHT is used as a PKI. Finally, communications between the *MUSeS* middleware and the local applications on a given machine are not secured. These unprotected local communications may create security issues. For instance, a local communication channel may be intercepted by a malicious user in order to spy a connection between two *MUSeS* users. We assume for the moment that the risks of attacks against local connections are avoided by the proper security of the machine itself.

E. Implementation

MUSeS is implemented as a middleware inserted between user applications and our CLOAK P2P overlay middleware. Currently, applications must use the *MUSeS* API to setup secure and mobile sessions and thus must be recompiled. Devices running these applications must run the CLOAK middleware and belong to a CLOAK overlay. *MUSeS* is implemented in the C language and consists in roughly 3000 lines of code. The code can be viewed at [6]. It is only a preliminary prototype implementation. The OpenSSL library is used by our implementation in order to provide security services such as protection against eavesdropping and protection against modification attacks. Traffic integrity is ensured by the HMAC-SHA1 cryptographic function which seals each traffic packet which implies that the two correspondents share a secret key. Traffic confidentiality is ensured by the AES algorithm which is used to encrypt the traffic payload. We currently assume that the needed secret keys have been previously exchanged securely by using public/private key signatures and encryption. For testing and evaluating *MUSeS*, we have written a very basic client-server application that is able to send a data

file over *MUSeS*. Furthermore, because CLOAK has only be implemented inside simulators so far, there is no usable CLOAK prototype and thus a static form of routing has been implemented as a helper in the *MUSeS* prototype in order to emulate the CLOAK layer.

III. Experimentation

An experimentation has been carried out with the above implementation in a dynamic environment composed of one mobile node. The emulation of the dynamic environment is done with our emulator *nemu* [7], a dynamic network emulator using QEMU. A mobile node inside this environment has the ability to leave one network (one virtual router) in order to join another one (another virtual router). This event causes a network failure during the move until a possible subsequent reconnection. This disruption is transparent for the application and it does not prevent the *MUSeS* system from continuing to run despite the fact that the mobile node is disconnected for a moment. Technically, in our experimentation shown in Figure 6, the node's mobility consists in causing an artificial failure on a virtual network interface. We disconnect a virtual wire from a virtual switch and reconnect it on another virtual switch. The *MUSeS* system hides this network change not only to the user's application but also to the remote corresponding node. The Figure 6 illustrates this *network change* scenario. In this Figure, *Bob* communicates with *Alice* when he decides to change its network location. He leaves the network on the left side of router *R5* and he joins the network on the left side of router *R4*. This changing location causes a network failure. However, this network disruption is transparent to both *Alice* and *Bob* applications. Subsequently, *Bob* re-contacts *Alice* and their session is therefore transparently resumed.

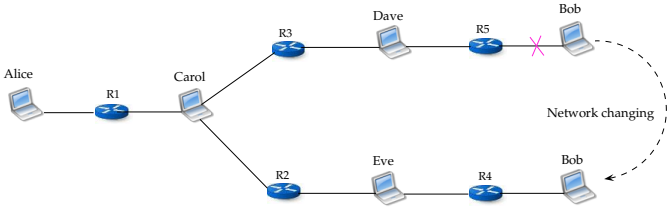

Fig. 6. Location changing scenario

Figure 7 shows the evolution of the throughput between the two corresponding applications over time. The network interruption happens at 59 seconds after the start of the experiment and the throughput instantly drops to zero. The connectivity is reestablished at the network and CLOAK level at 73 seconds and 82 hundredth. Because the CLOAK layer is currently just an emulation, this time interval may be greater when using the real implementation. After that, control messages are exchanged at the *MUSeS* level in order to resume the data transfer which finally happens at 76 seconds and 81 hundredth. Thus it takes 3 seconds to resume the transfer at the *MUSeS* level.

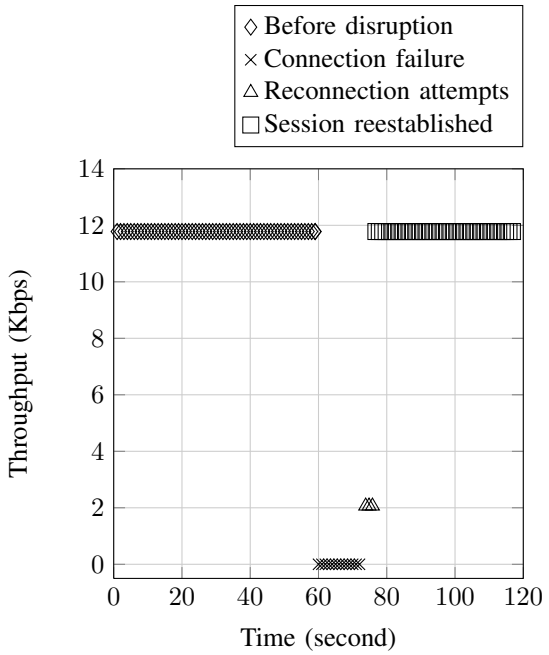

Fig. 7. Reconnection throughput *vs* time.

IV. Related Work

Mobile VPN systems based on both IPsec [8] and Mobile IP [9] have been proposed in [10], in order to attempt to overcome traditional VPN inherent mobility drawbacks. Nevertheless, this combination causes inconsistency problems [11] between IPsec and Mobile IP. In order to overcome these problems, another model has been proposed which is based on the use of two HAs (internal HA and external HA) and two FAs (internal FA and external FA) [12]. However, this model imposes the use of three imbricated tunnels({x-MIP{GW{i-MIP{original packet}}}}). In addition, an IPsec-based mobile VPN requires n tunnels (n security layers) when there are n IPsec hops between the source and destination entities. Therefore, VPN systems based on this model suffer network overhead.

Based on a topology organized in communities, P2P mobile VPN systems have also been proposed such as ELA [13] or N2N [14]. In opposition to most dynamic VPN systems, these systems have the advantage to be scalable and to have ability to communicate across NAT. Decentralized P2P VPN infrastructures are a flexible and self-organized way that allow users to create through untrusted network their own secure network. N2N and ELA topologies are very similar despite the fact that N2N is based on layer two whereas ELA is based on layer three.

SIP-based mobile VPN systems have also been proposed in the literature in [15] and [16]. However, the centralized model (client/server architecture) of the session continuity protocol (SIP) implies scalability issues. In addition, proposed SIP-based mobile VPN are only adapted to real-time applications.

FAST VPN [17] is a mobile VPN infrastructure designed for domestic networks. This VPN system creates a virtual

Internet Service Provider (vISP) inside a private network. This system allows to establish a secure VPN between users and a vISP. This technical concept capitalizes not only on the high security and privacy given by the VPN but also on the mobility provided by the flexibility of wireless networks. However, FAST VPN is not scalable and provides a restricted service to a private network: it permits only to secure Internet access through an untrusted private network.

The TLS protocol provides a way to resume a TLS session without requiring session-specific state at the TLS server [18]. In contrast to this technique that stops a session and restarts it later, our system has the ability to always maintain an established session active despite connection failures or network disruptions. *MUSeS* also makes connection disruptions transparent to the application as opposed to the TLS protocol.

Mobile VPN architectures based on SSH and TLS protocols are been proposed in [19] and [20]. These extensions of SSH and TLS protocol allow session to survive network failures. Designed to support resilient connections, mobile VPN presented in [20] can span several sequential TCP connections in order to resist connection failure and IP address change. Mobile VPN presented in [19] introduces a session resumption concept in order so as to resume sessions without having to renegotiate new session keys.

In opposition to aforesaid schemes, another interesting mobile VPN architecture has been designed [21]. Based on SOCKSv5 proxy, this mobile VPN model allows to authenticate clients and to build a secure SSL tunnel between users and a SSL gateway that protects their Intranet. However, this system is not scalable and when the SSL gateway is compromised then the entire network will be affected.

There are many other solutions that provide mobility services such as *session-based mobility* [22], the *persistent connection* model [23] and *session layer mobility* (SLM) [24]. Although these systems do not offer security services, they allow session continuity despite connection failures.

V. Conclusion and Future Work

In this paper, we have presented a new security and mobility system called *MUSeS* based on a P2P overlay system called CLOAK. *MUSeS* provides secure and resilient sessions to the applications run by mobile users. In other words, user sessions can be paused for any period of time and can be restarted without the cost of renegotiating security parameters. In addition, the *MUSeS* system makes lower layers disruptions transparent to user applications and to remote corresponding peers. *MUSeS* has also the ability to handle multiple local TCP connections in order to support simultaneously several user applications. We have designed and implemented a prototype of our system and we have assessed it in an emulated dynamic network environment.

Our future work will consist in completing and improving the current implementation as well as integrating a public key management scheme suitable for dynamic networks to provide the users with a means to exchange secret keys in-band. We plan to use certificate-less public key management techniques to solve traditional key management issues.

References

[1] T. Berger, "Analysis of current VPN technologies," in *Proc. of the 1st Int'l Conf. on Availability, Reliability and Security*, 2006.

[2] T. Tiendrebeogo, D. Magoni, and O. Sie, "Virtual Internet Connections Over Dynamic Peer-to-Peer Overlay Networks," in *Proc. of the 3rd Int'l Conf. on Evolving Internet*, 2011, pp. 58–65.

[3] C. Cassagnes, D. Bromberg, and D. Magoni, "An overlay architecture for achieving total flexibility in internet communications," in *Proc. of the 8th Int'l Conf. on Advanced Information Technologies for Management*, 2010, pp. 39–60.

[4] C. Cassagnes, T. Tiendrebeogo, D. Bromberg, and D. Magoni, "Overlay Addressing and Routing System Based on Hyperbolic Geometry," in *Proc. of the 16th IEEE Int'l Symp. on Computers and Communications*, 2011, pp. 294–301.

[5] T. Tiendrebeogo, D. Ahmat, and D. Magoni, "Reliable and Scalable Distributed Hash Tables Harnessing Hyperbolic Coordinates," in *Proc. of the 5th Int'l Conf. on New Technologies, Mobility and Security*, 2012.

[6] D. Ahmat and D. Magoni, *MUSeS*. [Online]. Available: http://www.labri.fr/perso/magoni/cape/

[7] V. Autefage and D. Magoni, "Network emulator: a network virtualization testbed for overlay experimentations," in *Proc. of the 17th IEEE Int'l Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks*, 2012.

[8] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," http://www.ietf.org/rfc/rfc4301.txt, 2005.

[9] C. Perkins, "IP Mobility Support for IPv4," http://tools.ietf.org/html/rfc3344/, 2002.

[10] R. Ruppelt, A. Pelinescu, C. Constantin, J. Floroiu, D. Sisalem, and B. Butscher, "Building ALL-IP Based Virtual Private Networks in Mobile Environment," in *Int. Works. on Informatik and Mobile communication over wireless LAN: Research and applications*, 2001.

[11] F. Adrangi and H. Levkowetz, "Problem Statement: Mobile IPv4 Traversal of Virtual Private Network (VPN) Gateways," http://www.ietf.org/rfc/rfc4093.txt, 2005.

[12] S. Vaarala and E. Klovning, "Mobile IPv4 Traversal across IPsec-Based VPN Gateways," http://www.ietf.org/rfc/rfc5265.txt, 2008.

[13] S. Aoyagi, M. Takizawa, M. Saito, H. Aida, and H. Tokuda, "ELA: A Fully Distributed VPN System over Peer-to-Peer Network," in *Proc. of the Symp. on Applications and the Internet*, 2005, pp. 89–92.

[14] L. Deri and R. Andrews, "N2N: A Layer Two Peer-to-Peer VPN," in *Proc. of the 2nd Int'l Conf. on Autonomous Infrastructure, Management and Security*, 2008.

[15] S. Huang, Z. Liu, and J. Chen, "SIP-based mobile VPN for real-time applications," in *IEEE Wireless Communications and Networking Conf.*, 2005, pp. 2318–2323.

[16] T.-C. Chen, J.-C. Chen, and Z.-H. Liu, "Secure Network Mobility for Real-Time Applications," *IEEE Transactions on Mobile Computing*, vol. 10, pp. 1113–1130, 2011.

[17] A. Zúquete and C. Frade, "Fast VPN Mobility Across Wi-Fi Hotspots," in *2nd IEEE Int'l Workshop on Security and Comm. Networks*, 2010.

[18] J. Salowey, H. Zhou, P. Eronen, and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State," *IETF RFC 5077*, 2008.

[19] J. Schönwälder, G. Chulkov, E. Asgarov, and M. Cretu, "Session Resumption for the Secure Shell Protocol," in *IFIP/IEEE Int'l Symp. on Integrated Network Management*, 2009, pp. 157–163.

[20] T. Koponen and P. Eronen, "Resilient Connections for SSH and TLS," in *Proc. of USENIX Annual Technical Conf.*, 2006.

[21] S. Minglei, T. Chengxiang, and W. Haihang, "Mobile VPN Scheme Based on SOCKS V5," in *Int'l Conf. on Machine Vision and Human-Machine Interface*, 2010, pp. 792–795.

[22] A. Snoeren, "A Session-Based Approach to Internet Mobility," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.

[23] Y. Zhang and S. Dao, "A "Persistent Connection" Model for Mobile and Distributed Systems," in *Proc. of the 4th Int'l Conf. on Comp. Comm. and Networks*, 1995.

[24] B. Landfeldt, T. Larsson, Y. Ismailov, and A. Seneviratne, "SLM, a framework for session layer mobility management," in *Proc. of the 18th Int'l Conf. on Comp. Comm. and Networks*, 1999, pp. 452–456.