Wireless Pers Commun (2016) 86:1879–1897
DOI 10.1007/s11277-015-3149-9

Design and Implementation of Path Failure Detection and Recovery Mechanism for Multihomed HIMALIS Network

Yusuke Fukushima[1] · Ved P. Kafle[1] · Tomoji Tomuro[1] · Hiroaki Harai[1]

Published online: 12 December 2015

Abstract We have designed the Heterogeneity Inclusion and Mobility Adaptation through Locator ID Separation (HIMALIS) network architecture to natively support sessions transfer from one link to another independently of the network and transport layer protocols. However, lack of a prompt path failure detection mechanism in HIMALIS cannot well utilize multiple links to protect against path failures. In this paper, we propose a complete path recovery mechanism that consists of path failure detection, lively path exploration, and path recovery processes. In particular, the failure detection mechanism is based on monitoring of data sending and receiving instances in end hosts by employing two timers, probe timer and keepalive timer. In case of path failure caused by gateway's upstream link down, the gateway provides the link failure notification to the host for expediting the detection process. We also present an overview of implementation of the mechanism. The implemented functions are verified by using both TCP and UDP applications and evaluated in a 21-node-scale network-emulation environment. The results show that the proposed mechanism provides fast failure detection, about 7 s which is much faster than the failure detection carried out based on TCP retransmission timeout which may take 5 min in the worst case, under both site- and host-multihoming configurations. The results also show that the proposed mechanism takes only about 200 ms to detect gateway's upstream link down.

Keywords ID/locator split · HIMALIS · Multihoming · New generation network · Future internet

✉ Yusuke Fukushima
yfukushima@nict.go.jp

Ved P. Kafle
kafle@nict.go.jp

Tomoji Tomuro
t-tomuro@nict.go.jp

Hiroaki Harai
harai@nict.go.jp

[1] National Institute of Information and Communications Technology, Tokyo, Japan

1 Introduction

Most of the personal communication devices are capable to get simultaneously connected to different types of access networks through multiple interfaces, such as Ethernet, Wi-Fi, 3G and WiMAX. Host-multihoming over such heterogeneous access networks becomes more common in the near future to enhance connectivity, increase communication bandwidth, and improve resiliency against an access network failure. Similarly, for the purpose of performance improvement, load balancing and failure resiliency, a site or access network can also get connected to two or more upper-level transit networks through two or more upstream links, which is called site-multihoming.

Failure resiliency is the most mission-critical requirement to enhance end-to-end connectivity because user applications cannot resume their data communication in case the transport-layer sessions get disconnected due to a link failure in the communication path. There are some router-based approaches [1, 2] that perform a communication path repair function based on local failure identification and packet rerouting techniques. However, router-based solutions are not aware of user applications' requirements and cannot utilize the host's multiple network interfaces for resiliency. Therefore, the multihomed hosts, together with multihomed access networks, can incorporate both better failure resiliency and higher bandwidth.

The traditional host-based solutions are subject to some limitations [3, 4]. For instance, a host fails to retain an ongoing session affected by the link's failure. This is because most transport layer protocols treat the IP address as a host identifier, and applications running on those protocols cannot resume data communication when the IP address become unreachable or unusable due to the link failure. To utilize properly many communication paths available between the communication hosts, ID/locator split-based network architectures, such as Shim6 [5], LISP [6], and HIMALIS [7], have been proposed. Shim6 and LISP focus on addressing the issues of multihoming and routing scalability, respectively, in the current Internet. In contrast, HIMALIS presents a common architectural framework for better support of multihoming and mobility in heterogeneous edge networks that use different network layer protocols, e.g. IPv4 and IPv6.

Although HIMALIS by design allows a multihomed host to transfer a session from one link to another link when a failure occurs in its link, it lacked the capability to promptly detect a communication path failure when failure occurs in a link (e.g. gateway's upstream links) or node (e.g. gateways) not directly attached to the host interface. Here, the communication path includes all links and nodes lying along the path from the source host to the destination host. The communication path fails when any link or node lying along the path fails. With conventional approaches, such as using TCP timeout or application-specific timeout, it may take a longer time to detect the path failure. Therefore, this paper, which is an extended version of papers published in earlier conferences [8, 9], fills this gap by proposing the path failure detection and session recovery mechanism. In the proposed mechanism, newly added two timers controlled by monitoring packets sending and receiving instances provide prompt path failure detection for both unidirectional (e.g. UDP) and bidirectional (e.g. TCP) communication. The ongoing data stream is immediately switched to another lively path. The goal of this work is to provide a common framework of communication path failure detection and recovery that can be applied over various transport and network layer protocols. This paper also presents the implementation overview of the mechanism in a prototype of multihomed HIMALIS networks. This mechanism has been implemented as an extended module of the HIMALIS user space

signaling software. The implemented functions are evaluated in a large-scale network emulation testbed as well.

The remainder of this paper is organized as follows. Section 2 briefly introduces a multihomed HIMALIS network and summarizes the communication path recovery mechanism. The proposed path failure detection mechanism is described in Sect. 3. Section 4 presents an overview of the implementation. Section 5 presents the performance evaluation results. Lastly, Sect. 6 concludes the paper.

2 Overview of HIMALIS Network Architecture

2.1 Multihomed HIMALIS Network

As illustrated in Fig. 1, a HIMALIS network consists of two or more edge or access networks and a single transit network. In each edge network, the network administrator can choose an optimal network layer protocol for better network management. Each edge network is connected to the transit network with one or more HIMALIS gateways (HGs), which provide network protocol or locator translation, to provide global connectivity to hosts residing in the edge networks. In Fig. 1, an edge network on the left side forms site-multihoming, and two edge networks on the right side support host-multihoming. In this example, the mobile host (MH) connects to a site-multihomed edge network, and the correspondent host (CH) makes a host multihomed connections through the two edge networks.

Each HG is assigned with a global locator (GLoc) to its every upstream interface and is reachable at the GLocs from other HGs. The MH and CH have a unique host identifier (HID) and hostname (e.g. *ch#himalis.net* for the CH). The hostname and HID are provided to the host during the initial setup. Since the name registry deployed in the transit network stores the records of hostname, HID and the associated GLocs, the set of HID and GLocs is always discoverable by resolving the hostname from the name registry. In this example, the MH obtains the CH's HID, GLoc3 and GLoc4 by resolving the hostname *ch#himalis.net*.

2.2 Path Establishment and Management

In HIMALIS, hosts and HGs have a new layer, called identity layer, to provide locator-independent functions, such as mobility, multihoming, and data security, by using a pair of the source and destination HIDs. In particular for the communication path management, hosts and HGs maintain sets of the source and destination HIDs and locators (it is also referred to as HID-to-locator mapping) to enable packet-forwarding function in the identity layer. Note that

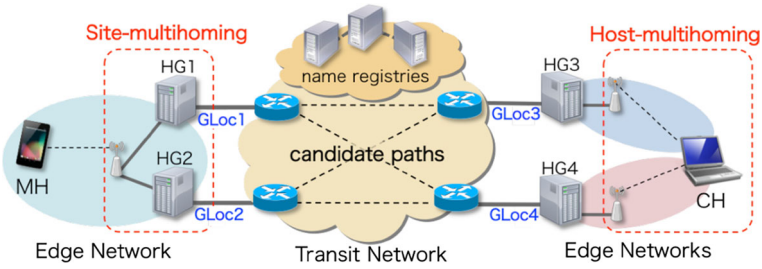


Fig. 1 Two types of multihoming configurations in HIMALIS network

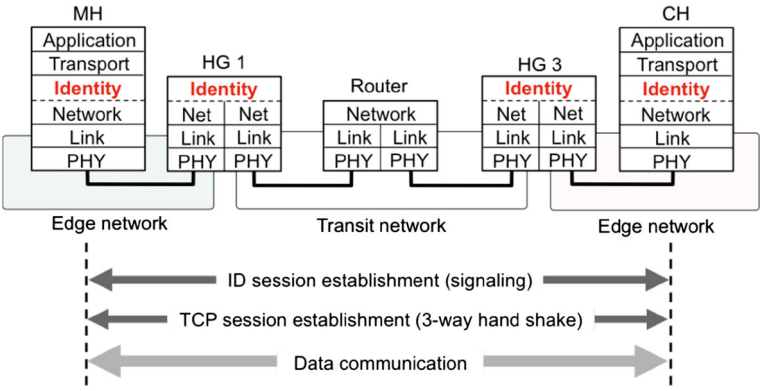


Fig. 2 An example of path establishment and management process between MH and CH through HG 1 and HG 3

the identity layer based packet forwarding is also useful for locator translation while protecting the payload of each packet. From the above reason, when a host wants to communicate with another host, the host establishes a session in the identity layer to provide those locator-independent functions before starting communication in the transport layer.

Figure 2 illustrates an example of data communication from the MH to the CH. As shown in Fig. 1, the MH has only a single interface to connect to the edge network, but there are four distinct communication paths to the CH because both the MH and CH are connected to the transit network with two GLocs. Since a pair of GLocs specifies a communication path, the MH can change communication path anytime in the multihomed HIMALIS network by updating either its own or CH's HID-to-locator mapping. According to the locator selection policy, such as [10], the MH selects a pair of GLocs to and establishes an identity-layer session on the path.

2.3 Basic Failure Detection

As described above, HIMALIS supports to switch a session in the identity layer to other communication paths by updating HID-to-locator mapping in hosts and HGs, but HIMALIS currently does not support path failure detection in the identity layer. Although a host can easily detect own link down event, it may take longer time to detect other cases (e.g. a link failure in the transit network). Moreover, since a session migration among communication paths becomes an essential requirement in multihomed networks, a complete path recovery mechanism including failure detection and path recovery in the identity layer is suitable in the HIMALIS network architecture. In the following section, we propose a prompt path failure detection mechanism to address this issue.

3 Path Failure Detection and Session Recovery Mechanism in Multihomed HIMALIS Network

The proposed mechanism provides a complete host-side active communication path recovery function that consisting of path failure detection, lively path exploration, and path recovery processes. Since the mechanism is designed to operate from the identity layer of

host protocol stack, communications with several transport-layer protocols (i.e. TCP and UDP) as well as various network-layer protocols are supported. In this section, we describe the operation of the proposed mechanism as below.

3.1 Path Failure Detection Process

The proposed path failure detection is simply based on monitoring of packet sending and receiving instances by maintaining two timers: probe timer (PT) and keepalive timer (KT). These two timers are mutually exclusive, that is, only one is running at a time. For a PT (KT) timeout, the host sends a probe (keepalive) packet to the correspondent host to check availability of the ongoing communication path. On receiving the probe (keepalive) packet, the correspondent host immediately sends its response back. The proposed mechanism performs path failure detection based on monitoring packets in the identity layer. The detection mechanism works independently of the transport layer protocols (e.g. TCP and UDP). For instance, even if the TCP function introduces a larger congestion window size and longer delayed acknowledgement, the proposed detection mechanism just exchanges probe packets to check availability of live communication paths.

The proposed detection mechanism consists of two exclusive detection states to support active path failure detection in both the sender and receiver sides (Fig. 3). By supporting failure detection in both sides, this mechanism works even if the correspondent host has disabled the detection process (Fig. 3). For example, a data service provider (i.e. cloud) can disable the detection function to mitigate the processing load. In other word, depending on application requirements or user policy, the detection function can be activated or deactivated. For the bidirectional communication model (e.g. TCP), "State of Data Sender" is enough because the data receiver regularly sends response packets (e.g. ACK packets). However, for a unidirectional communication model (e.g. UDP), this state cannot detect a failure in the receiver side because of lack of regular response packets. To address this issue, the proposed mechanism introduces "State of Data Receiver" to achieve path failure detection in the receiver side as well.

When an identity layer session is established, or when the host receives a packet, the host enters into *Data Receiving* state by starting the KT and stopping the PT. In Data Receiving state, the host does not change the timers when receiving packets. When the host sends a packet, it enters into *Data Sending* state by stopping the KT and starting the PT. In Data Sending state, the host does not change the timers when sending data packets. When

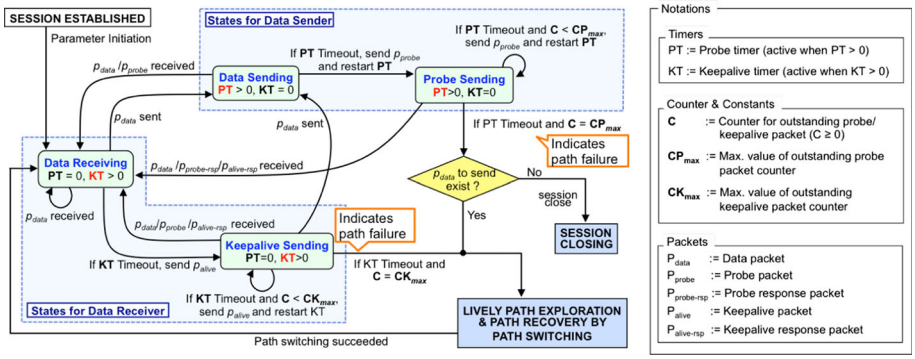


Fig. 3 State machine diagram of the proposed path failure detection

the host receives a data packet in Data Sending state, it transfers into Data Receiving state by stopping the PT and starting the KT. In Data Receiving state, it does not change the timers when receiving data packets.

In Data Receiving state when the KT hits KT timeout seconds (whose value is negotiated during the communication initialization phase), the host sends a keepalive packet to the peer host, restarts the KT, reduces KT timeout seconds to half of the previous KT timeout value, and enters into *Keepalive Sending* state. On receiving the keepalive packet, the peer host is supposed to respond the host immediately with a payload packet (if it has application data to send). If the host receives a data packet, a probe packet or a keepalive response packet in *Keepalive Sending* state, it returns to Data Receiving state and resets its KT timeout to the normal KT timeout and waits until the KT expiration for sending a keepalive packet again. In case the host does not receive any response again in Keepalive Sending state, it will repeatedly send keepalive packets and reduce KT timeout value until the number of outstanding keepalive packets reaches a preset maximum value. This process helps to detect a path failure when the host is receiving datagrams (i.e. UDP packets) from the peer host. Similarly, detecting a path failure for enabling prompt session recovery, closing unused identity sessions is also important to reduce management overheads. When both hosts have no application data to send or receive, and are exchanging only keepalive packets, either may issue a special type of keepalive (i.e. keepalive-close) packet to close the identity layer session.

In Data Sending state when the PT hits PT timeout seconds (whose value is negotiated during the communication initialization phase), the host sends a probe packet, restarts the PT, reduces PT timeout value to half of the previous PT timeout value, and enters into *Probe Sending* state. On receiving the probe packet, the peer host is supposed to respond the host immediately with a probe response packet. If the host receives a probe response packet, it knows that the communication path is still active, stops the PT, and resets PT timeout value to the original value. It then resumes sending application data packets and starts the PT. However, in case the host does not receive a response to its probe packet within the reduced PT timeout, it sends another probe packet and further reduces PT timeout to half of the previous PT timeout. In case the host does not receive any response again in Probe Sending state, it repeatedly sends probe packets and reduces the PT timeout value until the number of outstanding probe packets reaches a preset maximum value. If no response received from the peer host, it stops the PT and enters into the lively path exploration process.

Let $P_{\max}$ and $\mathrm{CP}_{\max}$ denote the maximum values (in millisecond) of PT timeout and outstanding probe packet counter, respectively. When the host is in the sender state, a path failure detection time (T_d) is given by

$$T_{\mathrm{d}} = P_{\max} + \left\lfloor \frac{P_{\max}}{2} \right\rfloor + \left\lfloor \frac{P_{\max}}{2^2} \right\rfloor + \cdots + \left\lfloor \frac{P_{\max}}{2^{CP_{\max}-1}} \right\rfloor < 2P_{\max} \tag{1}$$

$\mathrm{CP}_{\max}$ also affects the detection accuracy since the host sends UDP-based probe packets. These parameters are assumed to be determined based on the requirements of user applications. An example of the failure detection process is shown in Fig. 4. Let us assume that the communication from MH to CH illustrated in Fig. 1 and $P_{\max} = 4000$ and $\mathrm{CP}_{\max} = 3$. Initially, PT is set to $P_{\max}$ value and deactivated. Once a data packet is sent, PT is activated and started with $P_{\max}$ value. On receiving any packets, PT is deactivated. When PT timeout occurs, PT is restarted with half of the previous timeout value, and a probe packet is sent. The timeout value to set PT after C-th continuous timeout is given by floor

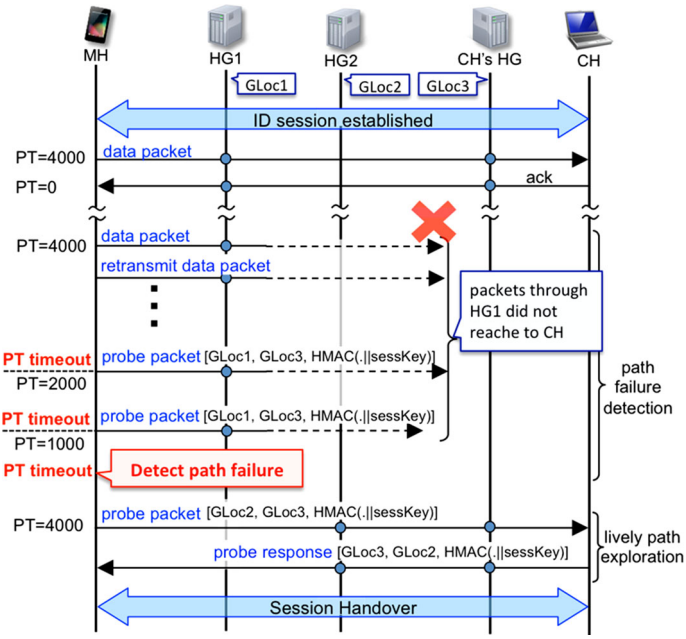


Fig. 4 Path failure detection and lively path exploration, where $P_{\max} = 4000$ ms, and $\mathrm{CP}_{\max} = 3$

($P_{\max}/2^{C}$). Finally, when continuous $\mathrm{CP}_{\max}$-times PT timeout occurs, the detection module identifies the ongoing communication data stream as being on a faulty path. In the detection process, it takes around 7 s to detect path failure, and two probe packets are sent to the CH.

Let $K_{\max}$ and $\mathrm{CK}_{\max}$ denote the maximum values (in millisecond) of KT timeout and outstanding keepalive packet counter, respectively. When the host is in the receiver state, a path failure detection time is as shown below.

$$T_d = K_{\max} + \left\lfloor \frac{K_{\max}}{2} \right\rfloor + \left\lfloor \frac{K_{\max}}{2^2} \right\rfloor + \cdots + \left\lfloor \frac{K_{\max}}{2^{CK_{\max}-1}} \right\rfloor < 2K_{\max} \tag{2}$$

We do not provide additional explanation of these parameters because the overall detection process is quite similar to the detection using the probe packets.

3.2 Lively Path Expolaration Process

Once a path failure has been successfully detected, this module sends probe packets to the peer host through various candidate communication paths, to find one or more lively paths. For each probe packet, PT is restarted with $P_{\max}$ and decremented to detect its timeout. On receiving the probe packet, the peer host sends a response back to the sender, immediately. If the host fails to receive a response from all candidate paths, the host closes the ID session regardless of the transport-layer protocol used.

In the network configuration shown in Fig. 1, there are at least three candidate paths after detecting a path failure in the MH. Since it may not be feasible to make exhaustive path exploration, a simple source and destination addresses selection algorithm [10] is used

in this module. Based on the selection algorithm, once a probe response is received, the path exploration module stops the rest of path exploration process immediately. As illustrated in Fig. 4, each probe packet and the corresponding response packet carry a pair of source and destination GLocs to indicate HGs to pass through. Those probe messages are protected by including their HMAC (i.e. keyed-hash message authentication code) calculated with the session key negotiated during the session initialization phase. In the case of Fig. 4, after the path failure detection process, the MH sends to the CH a probe packet carrying a pair of GLoc2 and GLoc3. On receiving the probe packet in HG2, HG2 checks the destination GLoc in the packet and transfers it to the HG assigned by GLoc3, and then the probe packet is delivered to the CH through CH's HG according to the destination HID. On receiving the probe packet in the CH, the CH sends a probe response packet back to the MH through the same HGs when the probe packet comes in.

The proposed failure detection mechanism supports the initiation of path failure detection from both the sender and receiver sides. However, an undesirable session disconnection can be happened due to execution of the lively path exploration and recovery processes simultaneously from both sides. To avoid this situation, when the ID session initiator (i.e. MH) enters into the path exploration process, the initiator drops any probe packets received from the ID session responder (i.e. CH).

3.3 Path Recovery Process

As soon as a lively path is found, the host transfers the ongoing data communication to the lively path. The path recovery process mainly consists of four sub-processes: switching HGs, packet redirection, location updates for the peer host and location updates for the host's own name registry as depicted in Fig. 5.

In the case that source GLocs in both previous and new paths are different, both switching HGs and packet redirection processes are carried out to reduce packet loss and receive new communication requests. In the switching HGs process, the host sends the peer host registration request to the new HG to send and receive data packets through the new HG. In the packet redirection process, the host sends the handover indication request to the old HG to redirect incoming data packets to the new HG.

In our previous experiments [11], the above two processes were helpful to reduce data packet loss during handover process. Note that they are skipped when the source GLoc does not change on the new path. To maintain the communication path properly, the host sends a handover notification message to the CH. Upon receiving this message, if necessary, the CH sends a peer host registration message to the corresponding HG to send and receive data packets through the HG. Once the HG updates its packet redirection table, the target communication path is recovered.

In particular, when the path failure is caused by either the host's interface down or the HG's upstream link down, the host sends a location update message to the host's name registry to delete the corresponding GLocs. After updating its GLocs in the name registry, newly generated communication requests are delivered to the host properly.

3.4 Gateway-Assisted Prompt Failure Detection

When the HG's upstream link (i.e. the link connected to the transit network) goes down, the host would be unreachable at its GLoc from the peer host, but the HG is still reachable from the host. In this case, if the HG helps the host to know its upstream link down, the

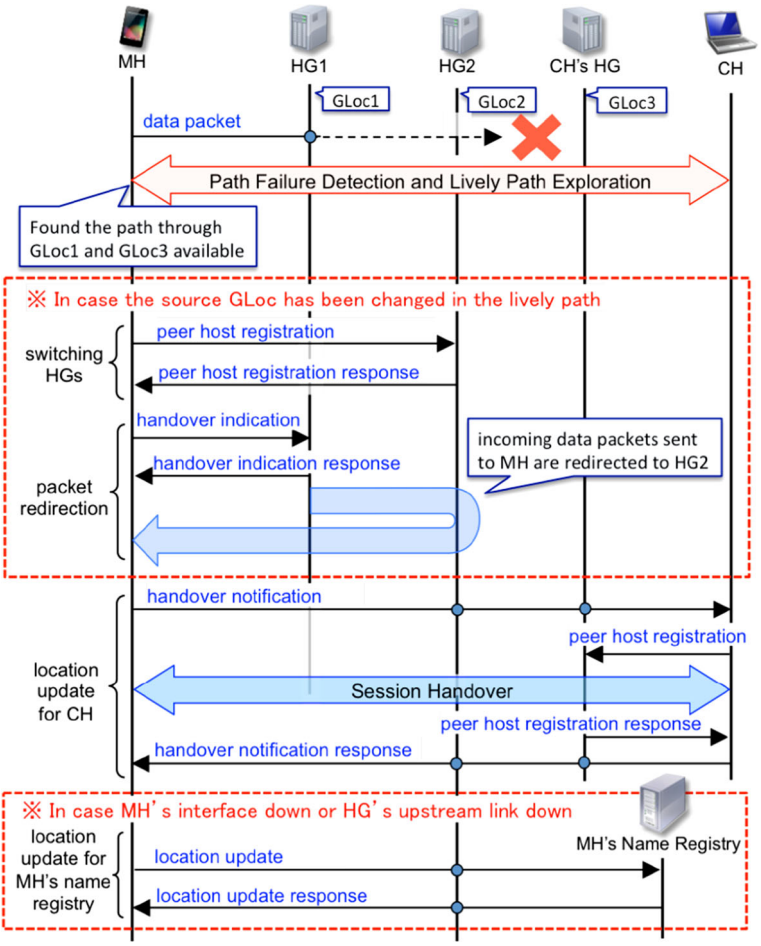


Fig. 5 Path recovery messages sequence

host can quickly switch ongoing communications and does not need to send any probe and keepalive packets for path failure detection. To support gateway-assisted prompt failure detection, we have designed a GLoc unreachable notification mechanism. In this mechanism, when the HG detects its upstream link down by using link-layer trigger, it configures a GLoc unreachable notification message containing GLoc1 as the unreachable GLoc and sends the message to all the hosts connected to the edge network. On receiving this notice, each host transfers the session from the HG to another one if the edge network consists of two or more HGs.

3.5 Session Closing Due to No Data Communication

From the security and resource management points of view, it is important to consider about when to close an identity layer session. In the proposed mechanism, the session is closed when there are no data packets exchanged for a specific time interval.

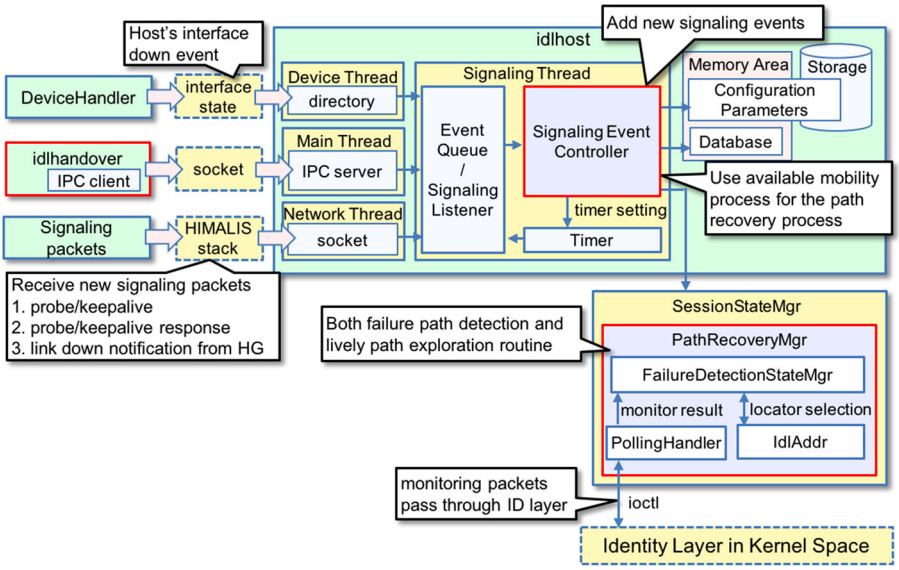


Fig. 6 Schematic block diagram of host signaling software. The modified parts of host signaling software are highlighted with a *red frame* border. (Color figure online)

4 Implementation of Path Recovery Mechanism

We have implemented the mechanism in available HIMALIS software in both hosts and HGs and verify different functions of the proposed communication path recovery mechanism and to evaluate their performances. The latest HIMALIS software is based on Ubuntu 12.04 LTS 64-bit with the Linux kernel version 3.2.40. The core processes in the identity layer (i.e. header processing) are implemented in the kernel space, and other control functions (i.e. signaling state machine shown in Fig. 3) are implemented in the user space. Figure 6 depicts a schematic block diagram of the host signaling mechanism, which is called `idlhost`. The state machine to handle host's ID-based control functions, such as mobility and path security management, are implemented in the signaling thread. When the host starts data communication with another host, the Signaling Event Controller module in the signaling thread launches `SessionStateMgr` to provide a path management function. `DeviceHandler` detects host's interface down.

To realize our path recovery mechanism, we have newly added `PathRecoveryMgr` in `SessionStateMgr`, which consists of `FailureDetectionStateMgr`, `PollingHandler`, and `IdlAddr` modules. `FailureDetectionStateMgr` module controls the entire state transition of the path recovery mechanism. `PollingHandler` module monitors data packet sending and receiving instances in the identity layer of host protocol stack in every 100 μs by employing the HIMALIS-capable `ioctl`. The monitoring results are reported to `FailureDetectionStateMgr`. `IdlAddr` module generates a set of candidate paths with selection priority to `FailureDetectionStateMgr module`. To handle new signaling packets, such as probe and keepalive packets and link down notification messages, we have slightly modified the signaling thread. Moreover, it is also useful to provide a command-based path switching API in the host-multihoming case because users sometimes want to switch communication paths according to the available

bandwidth or security level. Hence, we have also implemented an idlhandover command to provide a command-based path switching function, where this command communicates with the Signaling Event Controller utilizing an inter-process communication (IPC) method.

5 Experimental Evaluation

The following experiments evaluate the overall failure recovery time under both site- and host-multihoming configurations. We have constructed a HIMALIS network environment as shown in Fig. 7 to understand both actual failure detection time and communication recovery delay. In each edge network, an authentication agent/registrant (AAR) and a local name server (LNS) are deployed to provide network access control and name resolution, respectively. In the transit network, a single domain name registry (DNR) and host name registry (HNR) are deployed to provide the functions of name registries. Both MH's and CH's ID-to-locator mappings are stored in the HNR, so that the MH sends a location update message to HNR when the MH changes locators. The AAR, LNS, DNR, and HNR are involved in the signaling plane functions in the HIMALIS network, and they do not affect data plane functions. In the evaluation network of Fig. 7, each edge network has two HGs to form site-multihoming. The four Linux routers are installed to provide a single routing path between each pair of HGs in the transit network. Edge Network 2 is IPv6 network, while the others are IPv4. Total 21 components are installed on physically distinct 21 nodes in the network emulation testbed, StarBED [12, 13]. Each node, Cisco UCS C200

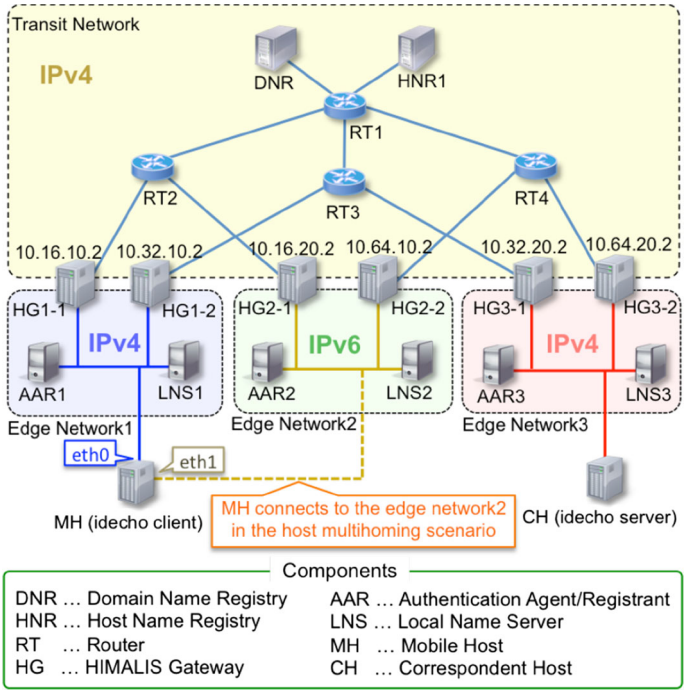


Fig. 7 Experimental network topology

M2, equips two 6-Core Intel Xeon X5670 processors, 48 GB RAM, and six 1000BASE-TX Ethernet adapters, where two Ethernet ports are used for administrative tools, and the rest four Ethernet ports are used to construct an ideal network topology. In this network, the average round trip time (RTT) between 2-hop (4-hop) neighboring HGs is 283 μs (706 μs). Although RTT may vary in the real network, such a short RTT helps to estimate implementation overhead and internal processing time in physical network interfaces.

To measure both failure detection time and path recovery delay, we have developed two applications: *idecho* and *idprobe*. The idecho program is a TCP echo application developed for HIMALIS native socket API and transfers data packets between the MH and the CH. Each application transfers a single packet carrying 256 Byte payload in every sending interval. The data sending process of idecho takes 10 ms sleep at every sending interval. The idprobe program is a UDP-based application developed with HIMALIS native socket API and transfers data packets from the MH to the CH in every 10 ms. Each datagram is provided with a unique packet ID to check packet loss.

In the experiment, we restrict our focus on the sender side path failure detection because the detection process in the receiver side is quite similar to the sender side. Through the entire experiments, P_{max} and CP_{max} are set to 4000 and 3, respectively. The data receiver side parameters (i.e. CK_{max}) are set to large enough values to avoid a case when the receiver side detection process starts sooner than the sender side detection process. Furthermore, TCP_RTO_MAX is set to 5*HZ (shorter than the default value of 120*HZ) to improve the failure recovery response for TCP data transmission, where HZ is defined by an architecture-specific parameter in Linux kernel. The related TCP_RETR2 is set to 60. The value of TCP_RTO_MIN remains the default value (HZ/5). As a result, in the worst-case scenario, the total maximum retransmission time becomes about 5 min (given by the product of TCP_RTO_MAX and TCP_RETR2).

5.1 Failure Recovery in Site-Multihoming Case

In the experimental setup shown in Fig. 7, we have considered that MH connects to Edge Network 1 and starts data communication with CH. According to the source and destination locator selection algorithm [10], the identity-layer session is initially established through HG1-2, RT3, and HG3. With this condition, the following two cases are examined.

5.1.1 Case A-1: Path Failure Due to a Link Failure in Transit Network

Figure 8a, b show both packet-sending and receiving results using the TCP application idecho and UDP application idprobe observed in the MH (using tcpdump command), respectively. Received packets before and after injecting the failure are plotted in red and blue colors, respectively. Dropped packets are plotted in gray. To demonstrate a link failure in the transit network, `iptables` command is used from the MH to drop any data stream passing through the link between RT3 and HG1-2.

In case of data transmission using idecho application, since the MH received acknowledgement immediately for every packet sending, PT was activated and set to P_{max} as soon as the link failure was injected. So, the first PT timeout came after 4 s. At this moment, the MH sent the first probe packet to the CH. Since MH received neither data packets nor the probe response from the CH, MH detected a path failure when the third PT timeout comes. As a result, it took 7 s to detect the path failure. After having detected the failure, the lively path exploration process started. Then, the path recovery process started. As a result, in this experiment, the communication path was recovered within 7.7 s;

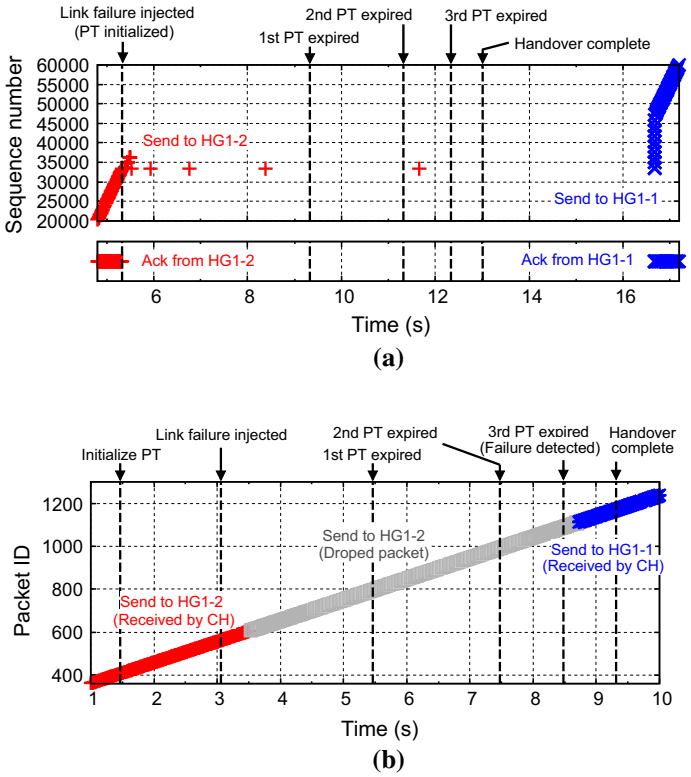


Fig. 8 Failure recovery time when the link between HG1-2 and RT3 fails. **a** Using TCP application idecho. **b** Using UDP application idprobe

however, the total failure recovery time for TCP session resulted in 11.4 s. The additional delay was caused by the fact that RTO value had reached the maximum value (5 s) during the failure detection process. Note that the failure detection logic explained above is also applicable in the CH because the CH becomes a data sender when sending an acknowledgement packet to the MH.

In contrast, in case of data transmission using the UDP application idprobe, the MH did not receive any data packets from the CH. As we can see at Fig. 8b, PT was always activated unless the MH received keepalive packets. In this experiment, the total path recovery delay was about 5.2 s.

5.1.2 Case A-2: Path Failure Due to HG's Upstream Interface Down

The HG1-2's upstream interface or link down event is injected by using *ifconfig* command from the MH. As shown in Fig. 9, in this experiment, it takes a bit longer time to down HG1-2's upstream link, but the MH could successfully received a upstream link down notification message from HG1-2 within 200 ms. The MH then switches the communication path to the next available HG1-1 by sending a signaling packet to register the CH's HID and GLoc in HG1-1. MH sends a location update request message to CH and then can send the data packets before it receives the location update response

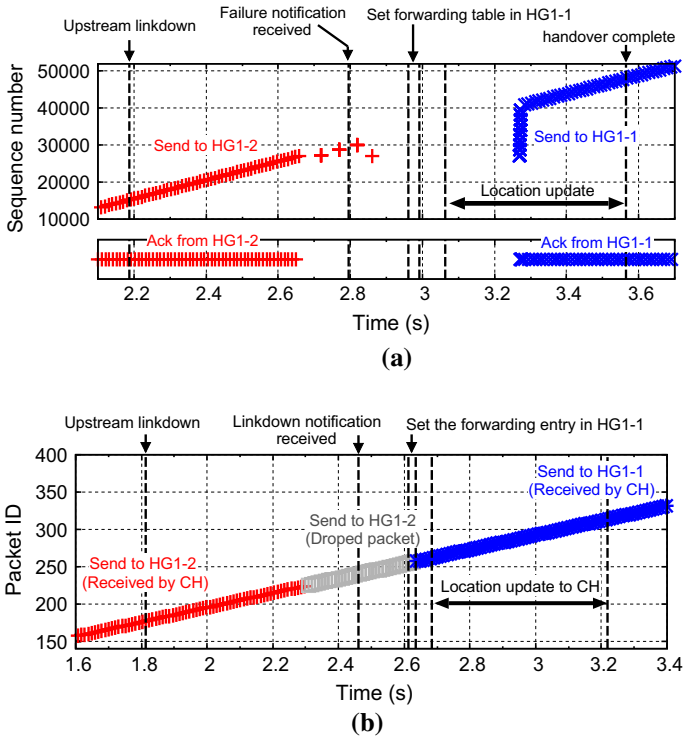


Fig. 9 Failure recovery time when HG1-2's upstream link goes down. **a** Using TCP application idecho. **b** Using UDP application idprobe

message. As a result, the failure recovery time was about 400 ms in case of data transmission using the TCP application idecho (about 340 ms in case of data transmission using the UDP application idprobe). As shown in Fig. 9b, the data communication path can be recovered after setting packet-forwarding entry in the HG1-1 to use the HG1-1 for the data transmission. In the current implementation, although HG1-1 monitors it's link state using a script program, a kernel-based monitoring by using a netlink socket can provide faster detection [9].

5.2 Failure Recovery in Host-Multihoming Case

In contrast with the experiments in the site-multihoming scenario, consider that the MH connects to both Edge Network 1 and 2 through eth0 and eth1, respectively, and starts data communication with the CH using *idecho* or *idprobe* application. Data packets are initially transferred through HG2-1, RT2, RT1, RT3, and HG3-1. In this condition, the following two cases are examined.

5.2.1 Case B-1: Link Failure Occurs in Edge Network 2

Similar to Case A-1, we used `iptables` to drop any data packets passing through the link between the MH and HG2-1 to demonstrate a link failure in Edge Network 2. After the MH

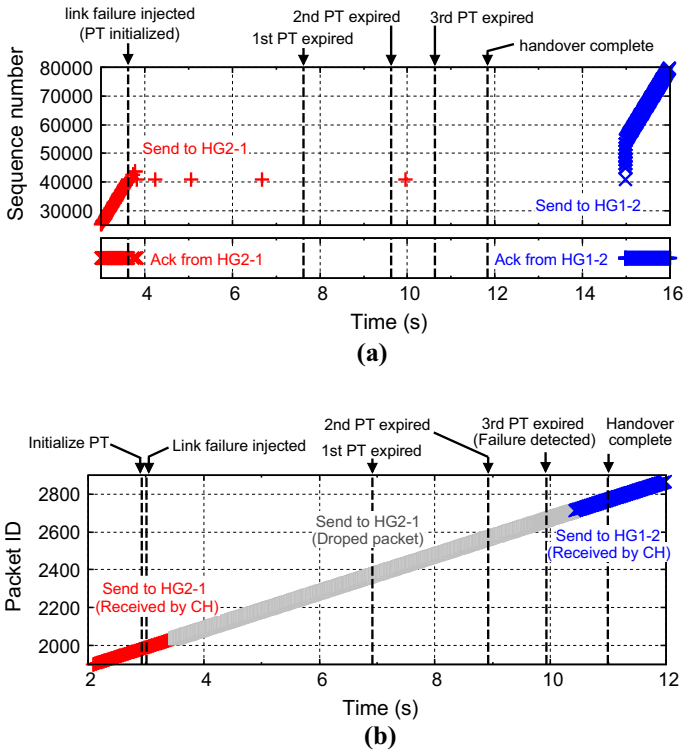


Fig. 10 Failure recovery time after MH's eth1 goes down. **a** Using TCP application idecho. **b** Using UDP application idprobe

detected the path failure event, the MH resumed the ongoing data stream through HG1-2, RT3 and HG3-1 within 11.2 s for the idecho application and 7 s for the idprobe application as shown in Fig. 10a, b, respectively. From the algorithmic point of view, there is only a slight difference in the path failure detection in Case A-1 and Case B-1.

5.2.2 Case B-2: Path Failure Due to MH's Link Down

Figure 11a, b show failure recovery time in case of MH's link down. In this experiment, `ifdown` command was used to down MH's eth1. It took about 200 ms to detect its interface down in the MH. In case of data transmission using idprobe, sending packets were dropped based on the routing table in the MH. In this case, since tcpdump command failed to capture such dropped packets, Fig. 11b does not include dropped packets. After detecting the interface down event, MH took additional 200 ms to switch from HG2-1 to HG1-2 as the active HG for the session. Then, HG2-1 redirected packets destined for MH to HG1-2. Finally, it took about 620 ms for idecho (560 ms for idprobe) to recover communication path.

5.3 Communication Path Switching by External Command

For scheduled path failure due to network maintenance, the command-based path switching function is useful. To verify the implemented function, similar to the host-multihoming

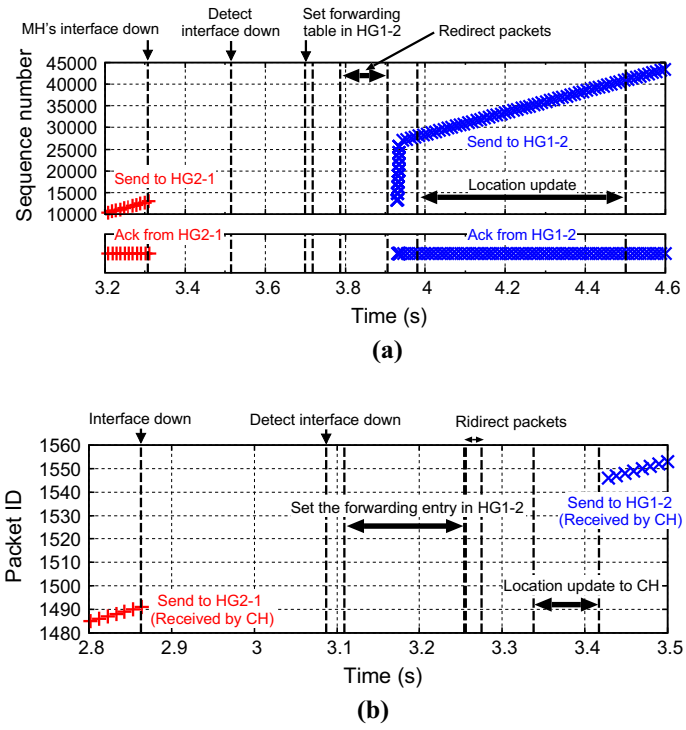


Fig. 11 Failure recovery time after MH's eth1 goes down. **a** Using TCP application idecho. **b** Using UDP application idprobe

scenario, MH was connected to both Edge Network 1 and 2 through eth0 and eth1, respectively, and had started data communication with CH using *idecho* or *idprobe* application. In this case, data packets were initially transferred through HG1-2, RT3, and HG3-1. The experiment observed no packet loss during the session migration from eth1 to eth0, as shown in Fig. 12a, b. Although data packets redirected from HG1-2 had incurred additional delay, the MH received all packets sent from the CH for both idecho and idprobe applications.

From the above experiments and evaluation results, we can conclude that our proposed path failure detection mechanism can detect a path failure at any point with given detection time. In the above experiments, when the maximum probe timeout and the maximum probe attempt times set to 4 s and 3 times, respectively, it resulted in completing the failure detection within 7 s. This is faster than the failure detection carried out by the TCP retransmission timeout that may take 5 min. The evaluation results also show that the proposed mechanism takes only about 200 ms to detect gateway's upstream link down.

6 Summary and Future Work

In this paper, we have proposed a path failure detection and session recovery mechanism for a multihomed HIMALIS network. This mechanism provides a complete path failure recovery, which consists of path failure detection, lively path exploration, and path recovery processes. We have also presented the implementation overview of this

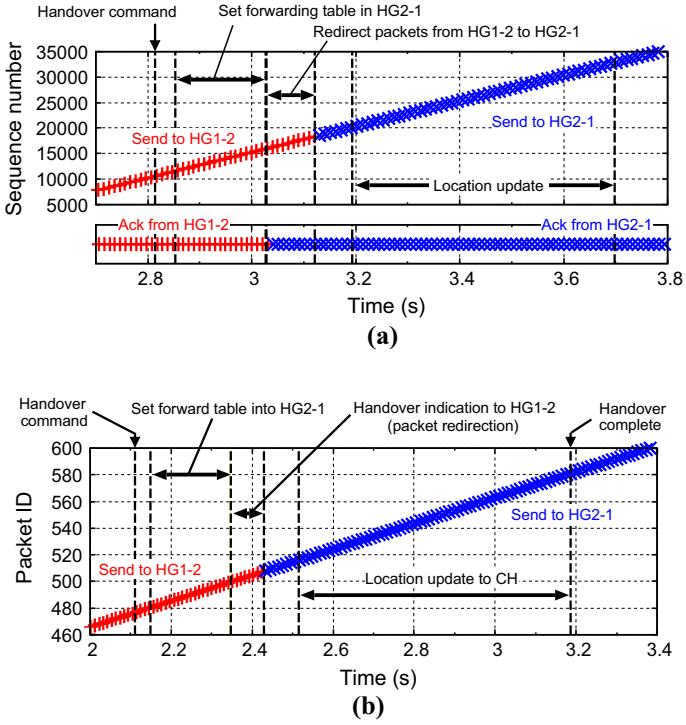


Fig. 12 Data stream migration from one link to another by the external command. **a** Using TCP application idecho. **b** Using UDP application idprobe

mechanism, and each process was implemented in the latest HIMALIS software package. Several experiments were conducted to evaluate overall failure recovery time and the related processing delays in a testbed network environment. Those evaluation results indicate that the proposed path recovery mechanism provides fast path failure recovery under both site- and host-multihoming configurations. The entire mechanism is also verified to be feasible for both IPv4 and IPv6 network protocols. In future work, we will construct a large-scale testbed to evaluate its scalability.

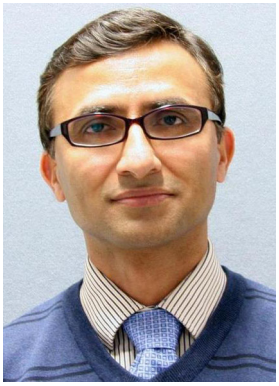
Acknowledgments The authors are grateful to Kenji Fujikawa and Yasunaga Kobari for their valuable input to this research.

References

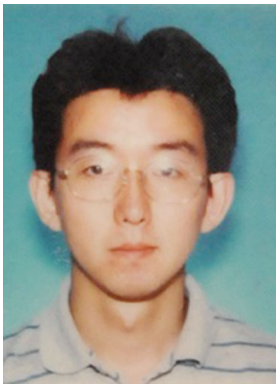
1. Shand, M., & Bryant, S. (2014). IP fast reroute framework, *RFC*, 5714.
2. Yi, C., Afanasyev, A., Wang, L., Zhang, B., & Zhang L. (2012). Adaptive forwarding in named data networking, *ACM SIGCOMM CCR*, *42*(3), 62–67.
3. Katz, D., & Ward, D. (2010). Bidirectional forwarding detection, *RFC*, 5880.
4. Cortés, A., García-Rubio, C., Campo, C., Marín, A., Almenárez, F., & Díaz, D. (2008). Decoupleing path failure detection from congestion control to improve SCTP failovers. *IEEE Communications Letters, 12*(11), 858–860.
5. Nordmark, E., & Bagnulo, M. (2009). Shim6: Level 3 multihoming shim protocol for IPv6, *RFC*, 5533.

6. Farinacci, D., Fuller, V., Meyer, D., & Lewis, D. (2013). Locator/ID separation protocol (LISP), *RFC* 6830.
7. Kafle, V. P., & Inoue, M. (2010). HIMALIS: Heterogeneity inclusion and mobility adaptation through locator ID separation in new generation network. *IEICE Transactions on Communications, E93-B*(3), 478–489.
8. Kafle, V. P., Fukushima, Y., & Harai, H. (2013). Path failure detection and session recovery mechanism in multihomed HIMALIS network. In *The fifth international conference on ubiquitous and future networks* (*ICUFN 2013*) (pp. 558–563). Da Nang, Vietnam.
9. Fukushima, Y., Kafle, V. P., Tomuro, T., & Harai, H. (2014). Implementation of communication path recovery mechanism in a multihomed ID/locator-split network. In *The sixth international conference on ubiquitous and future networks* (*ICUFN 2014*) (pp. 322–327). Shanghai, China.
10. Thaler, D., Draves, R., Matsumoto, A., & Chown, T. (2012). Default address selection for internet protocol version 6 (IPv6), *RFC*, 6724.
11. Kafle, V. P., Fukushima, Y., & Harai, H. (2014). ID/locator split-based distributed mobility management mechanism. *Wireless Personal Communications, 76*(4), 693–712.
12. Miyachi, T., Nakagawa, T., Chinen, K., Miwa, S., & Shinoda, Y. (2011). StarBED and SpringOS architectures and their performance, In *The 7th international ICST conference on testbeds and research infrastructures for the development of networks and communications* (*TridentCom 2011*).
13. StarBED[3] Project. http://starbed.nict.go.jp/en/aboutus/index.html. Accessed on June 18, 2015.

Yusuke Fukushima received Ph.D. degree in Information Science from Tohoku University, Japan in 2009. He is currently a researcher at National Institute of Information and Communications Technology (NICT). Before joining NICT, he shortly worked as an assistant professor at Graduate School of Information Science from Tohoku University and then as a research associate at Faculty of Science and Technology at Sophia University, Japan from 2010 to 2012. His research interests include fault-tolerant routing control, parallel and distributed systems, and designing future network architecture. He is a member of ACM.



Ved P. Kafle received the B.E. degree in Electronics and Electrical Communications from Punjab Engineering College (now PEC University of Technology), Chandigarh, India, the M.S. degree in Computer Science and Engineering from Seoul National University, South Korea, and the Ph.D. in Informatics from the Graduate University for Advanced Studies, Japan. He is now a senior researcher at National Institute of Information and Communications Technology (NICT) and concurrently holding a visiting associate professor's position at the University of Electro-Communications, Tokyo. His research interests include new naming and addressing schemes, ID/locator separation, name resolution architecture, integration of heterogeneous network layer protocols, integration of resource-constrained sensor networks into the Internet for ubiquitous sensing and computing, distributed mobility management, and privacy, security and trust in communication networks. He has been awarded with the ITU Association of Japan Award in 2009 for his contributions to the standardization of Next Generation Network architectures. He also received the best paper award (second prize) at the ITU-T Kaleidoscope event on Innovations for Digital Inclusion, 2009. He is a senior member of IEEE and a member of IEICE.



Tomoji Tomuro received M.S degree in Mathematics from Yamagata University, Japan in 1991, after Faculty of Science and Engineering from WASEDA University. He worked for several software companies, semiconductor equipment manufacturer, he founded a company Tomorrow System Co., Ltd for software and CAE consulting. From 2009, he is a technical expert at National Institute of Information and Communications Technology.

Hiroaki Harai received M.E. and Ph.D. degrees in Information and Computer Sciences from Osaka University, Japan in 1995 and 1998, respectively. He is currently a Director at National Institute of Information and Communications Technology (NICT), Tokyo, Japan, where he is leading Network Architecture Laboratory for Optical and New-Generation Networks. He is concurrently a Visiting Associate Professor of Japan Advanced Institute of Science Technology, Ishikawa, Japan. His current research topic is design and development of new generation network architecture. Dr. Harai was elected to Outstanding Young Researcher in the 3rd IEEE ComSoc Asia-Pacific Young Researcher Award, 2007. He received the 2009 Young Researcher Award from the Ministry of Education, Culture, Sports, Science and Technology. He is a member of IEEE and IEICE.