DMM Working Group
Internet-Draft
Intended status: Informational
Expires: January 27, 2019

A. Yegin
Actility
D. Moses
Intel
K. Kweon
J. Lee
J. Park
Samsung
S. Jeon
Sungkyunkwan University

July 26, 2018

On Demand Mobility Management
draft-ietf-dmm-ondemand-mobility-15

Abstract

Applications differ with respect to whether they need session continuity and/or IP address reachability. The network providing the same type of service to any mobile host and any application running on the host yields inefficiencies. This document describes a solution for taking the application needs into account by selectively providing session continuity and IP address reachability on a per-socket basis.

Table of Contents

1. Introduction

In the context of Mobile IP [RFC5563][RFC6275][RFC5213][RFC5944], the following two attributes are defined for IP service provided to mobile hosts:

Session continuity: The ability to maintain an ongoing transport interaction by keeping the same local end-point IP address throughout the life-time of the IP socket despite the mobile host changing its

point of attachment within the IP network topology. The IP address of the host may change after closing the IP socket and before opening a new one, but that does not jeopardize the ability of applications using these IP sockets to work flawlessly. Session continuity is essential for mobile hosts to maintain ongoing flows without any interruption.

IP address reachability: The ability to maintain the same IP address for an extended period of time. The IP address stays the same across independent sessions, and even in the absence of any session. The IP address may be published in a long-term registry (e.g., DNS), and is made available for serving incoming (e.g., TCP) connections. IP address reachability is essential for mobile hosts to use specific/ published IP addresses.

Mobile IP is designed to provide both session continuity and IP address reachability to mobile hosts. Architectures utilizing these protocols (e.g., 3GPP, 3GPP2, WIMAX) ensure that any mobile host attached to the compliant networks can enjoy these benefits. Any application running on these mobile hosts is subjected to the same treatment with respect to session continuity and IP address reachability.

It should be noted that in reality not every application may need these benefits. IP address reachability is required for applications running as servers (e.g., a web server running on the mobile host). But, a typical client application (e.g., web browser) does not necessarily require IP address reachability. Similarly, session continuity is not required for all types of applications either. Applications performing brief communication (e.g., ping) can survive without having session continuity support.

Achieving session continuity and IP address reachability with Mobile IP incurs some cost. Mobile IP protocol forces the mobile host's IP traffic to traverse a centrally-located router (Home Agent, HA), which incurs additional transmission latency and use of additional network resources, adds to the network CAPEX and OPEX, and decreases the reliability of the network due to the introduction of a single point of failure [RFC7333]. Therefore, session continuity and IP address reachability SHOULD be provided only when necessary.

Furthermore, when an application needs session continuity, it may be able to satisfy that need by using a solution above the IP layer, such as MPTCP [RFC6824], SIP mobility [RFC3261], or an application-layer mobility solution. These higher-layer solutions are not subject to the same issues that arise with the use of Mobile IP since they can utilize the most direct data path between the end-points. But, if Mobile IP is being applied to the mobile host, the higher-

layer protocols are rendered useless because their operation is inhibited by Mobile IP. Since Mobile IP ensures that the IP address of the mobile host remains fixed (despite the location and movement of the mobile host), the higher-layer protocols never detect the IP-layer change and never engage in mobility management.

This document proposes a solution for applications running on mobile hosts to indicate whether they need session continuity or IP address reachability. The network protocol stack on the mobile host, in conjunction with the network infrastructure, provides the required type of service. It is for the benefit of both the users and the network operators not to engage an extra level of service unless it is absolutely necessary. It is expected that applications and networks compliant with this specification will utilize this solution to use network resources more efficiently.

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Solution

3.1. Types of IP Addresses

Four types of IP addresses are defined with respect to mobility management.

- Fixed IP Address

A Fixed IP address is an address with a guarantee to be valid for a very long time, regardless of whether it is being used in any packet to/from the mobile host, or whether or not the mobile host is connected to the network, or whether it moves from one point-of-attachment to another (with a different IP prefix) while it is connected.

Fixed IP addresses are required by applications that need both session continuity and IP address reachability.

- Session-lasting IP Address

A session-lasting IP address is an address with a guarantee to be valid throughout the life-time of the socket(s) for which it was requested. It is guaranteed to be valid even after the mobile host had moved from one point-of-attachment to another (with a different IP prefix).

Session-lasting IP addresses are required by applications that need session continuity but do not need IP address reachability.

- Non-persistent IP Address

This type of IP address has no guarantee to exist after a mobile host moves from one point-of-attachment to another, and therefore, no session continuity nor IP address reachability are provided. The IP address is created from an IP prefix that is obtained from the serving IP gateway and is not maintained across gateway changes. In other words, the IP prefix may be released and replaced by a new one when the IP gateway changes due to the movement of the mobile host forcing the creation of a new source IP address with the updated allocated IP prefix.

- Graceful Replacement IP Address

In some cases, the network cannot guarantee the validity of the provided IP prefix throughout the duration of the opened socket, but can provide a limited graceful period of time in which both the original IP prefix and a new one are valid. This enables the application some flexibility in the transition from the existing source IP address to the new one.

This gracefulness is still better than the non-persistence type of address for applications that can handle a change in their source IP address but require that extra flexibility.

Applications running as servers at a published IP address require a Fixed IP Address. Long-standing applications (e.g., an SSH session) may also require this type of address. Enterprise applications that connect to an enterprise network via virtual LAN require a Fixed IP Address.

Applications with short-lived transient sessions can use Session-lasting IP Addresses. For example: Web browsers.

Applications with very short sessions, such as DNS clients and instant messengers, can utilize Non-persistent IP Addresses. Even though they could very well use Fixed or Session-lasting IP Addresses, the transmission latency would be minimized when a Non-persistent IP Addresses are used.

Applications that can tolerate a short interruption in connectivity can use the Graceful-replacement IP addresses. For example, a streaming client that has buffering capabilities.

3.2. Granularity of Selection

IP address type selection is made on a per-socket granularity. Different parts of the same application may have different needs. For example, the control-plane of an application may require a Fixed IP Address in order to stay reachable, whereas the data-plane of the same application may be satisfied with a Session-lasting IP Address.

3.3. On Demand Nature

At any point in time, a mobile host may have a combination of IP addresses configured. Zero or more Non-persistent, zero or more Session-lasting, zero or more Fixed and zero or more Graceful-Replacement IP addresses may be configured by the IP stack of the host. The combination may be as a result of the host policy, application demand, or a mix of the two.

When an application requires a specific type of IP address and such an address is not already configured on the host, the IP stack SHALL attempt to configure one. For example, a host may not always have a Session-lasting IP address available. When an application requests one, the IP stack SHALL make an attempt to configure one by issuing a request to the network (see Section 3.4 below for more details). If the operation fails, the IP stack SHALL fail the associated socket request and return an error. If successful, a Session-lasting IP Address gets configured on the mobile host. If another socket requests a Session-lasting IP address at a later time, the same IP address may be served to that socket as well. When the last socket using the same configured IP address is closed, the IP address may be released or kept for future applications that may be launched and require a Session-lasting IP address.

In some cases it might be preferable for the mobile host to request a new Session-lasting IP address for a new opening of an IP socket (even though one was already assigned to the mobile host by the network and might be in use in a different, already active IP sockets). It is outside the scope of this specification to define criteria for choosing to use available addresses or choosing to request new ones. It supports both alternatives (and any combination).

It is outside the scope of this specification to define how the host requests a specific type of prefix and how the network indicates the type of prefix in its advertisement or in its reply to a request).

The following are matters of policy, which may be dictated by the host itself, the network operator, or the system architecture standard:

- The initial set of IP addresses configured on the host at boot time.

- Permission to grant various types of IP addresses to a requesting application.

- Determination of a default address type when an application does not make any explicit indication, whether it already supports the required API or it is just a legacy application.

3.4. Conveying the Desired Address Type

[RFC5014] introduced the ability of applications to influence the source address selection with the IPV6_ADDR_PREFERENCE option at the IPPROTO_IPV6 level. This option is used with setsockopt() and getsockopt() calls to set/get address selection preferences.

Extending this further by adding more flags does not work when a request for an address of a certain type results in requiring the IP stack to wait for the network to provide the desired source IP prefix and hence causing the setsockopt() call to block until the prefix is allocated (or an error indication from the network is received).

Alternatively a new socket API is defined - getsc() which allows applications to express their desired type of session continuity service. The new getsc() API will return an IPv6 address that is associated with the desired session continuity service and with status information indicating whether or not the desired service was provided.

An application that wishes to secure a desired service will call getsc() with the service type definition and a place to contain the provided IP address, and call bind() to associate that IP address with the socket (See pseudo-code example in Section 4 below).

When the IP stack is required to use a source IP address of a specified type, it can use an existing address, or request a new IP prefix (of the same type) from the network and create a new one. If the host does not already have an IPv6 prefix of that specific type, it MUST request one from the network.

Using an existing address from an existing prefix is faster but might yield a less optimal route (if a hand-off event occurred after its configuration). On the other hand, acquiring a new IP prefix from the network may be slower due to signaling exchange with the network.

Applications can control the stack's operation by setting a new flag - ON_NET flag - which directs the IP stack whether to use a

preconfigured source IP address (if exists) or to request a new IPv6 prefix from the current serving network and configure a new IP address.

This new flag is added to the set of flags in the IPV6_ADDR_PREFERENCES option at the IPPROTO_IPV6 level. It is used in setsockopt() to set the desired behavior.

4. Usage example

4.1. Pseudo-code example

The following example shows pseudo-code for creating a Stream socket (TCP) with a Session-Lasting source IP address:

```
#include <sys/socket.h>
#include <netinnet/in.h>

  // Socket information
int              s ;            // socket id

  // Source information (for secsc() and bind())
sockaddr_in6     sourceInfo     // my address and port for bind()
in6_addr         sourceAddress  // will contain the provisioned
                                // source IP address
uint8_t          sc_type = IPV6_REQUIRE_SESSION_LASTING_IP ;
                                // For requesting a Session-Lasting
                                // source IP address

  // Destination information (for connect())
sockaddr_in6     serverInfo ;   // server info for connect()

  // Create an IPv6 TCP socket
s = socket(AF_INET6, SOCK_STREAM, 0) ;
if (s!=0) {
      // Handle socket creation error
      // ...
} // if socket creation failed
else {
       // Socket creation is successful
       // The application cannot connect yet, since it wants to use
       // a Session-Lasting source IP address It needs to request
       // the Session-Lasting source IP before connecting
     if (setsc(s, &sourceAddress, &sc_type)) == 0){
          // setting session continuity to Session Lasting is
          // Successful. sourceAddress now contains the Session-
          // LAsting source IP address
```

```
            // Bind to that source IP address
          sourceInfo.sin6_family = AF_INET6 ;
          sourceInfo.sin6_port = 0  // let the stack choose the port
          sourceInfo.sin6_address = sourceAddress ;
                                  // Use the source address that was
                                  // generated by the setsc() call
          if (bind(s, &sourceInfo, sizeof(sourceInfo))==0){
               // Set the desired server's information for connect()
             serverInfo.sin6_family = AF_INET6 ;
             serverInfo.sin6_port = SERVER_PORT_NUM ;
             serverAddress.sin6_addr = SERVER_IPV6_ADDRESS ;

               // Connect to the server
             if (connect(s, &serverInfo, sizeof(serverInfo))==0) {
                 // connect successful (3-way handshake has been
                 // completed with Session-Lasting source address.
                 // Continue application functionality
                 // ...
             }  // if connect() is successful
             else {
                 // connect failed
                 // ...
                 // Application code that handles connect failure and
                 // closes the socket
                 // ...
             } // if connect() failed
          } // if bind() successful
          else {
                 // bind() failed
                 // ...
                 // Application code that handles bind failure and
                 // closes the socket
                 // ...
          } // if bind() failed
       }  // if setsc() was successful and of a Session-Lasting
          // source IP address was provided
       else {
            // application code that does not use Session-lasting IP
            // address. The application may either connect without
            // the desired Session-lasting service, or close the
            // socket...
       } // if setsc() failed
   }  // if socket was created successfully

     // The rest of the application's code
     // ...
```

4.2. Message Flow example

The following message flow illustrates a possible interaction for achieving OnDemand functionality. It is an example of one scenario and should not be regarded as the only scenario or the preferred one.

This flow describes the interaction between the following entities:

- Applications requiring different types of OnDemand service.

- The mobile host's IP stack.

- The network infrastructure providing the services.

In this example, the network infrastructure provides 2 IPv6 prefixes upon attachment of the mobile host to the network: A Session-lasting IPv6 prefix and a Non-persistent IPv6 prefix. Whenever the mobile host moves to a different point-of-attachment, the network infrastructure provides a new Non-persistent IPv6 address.

In this example, the network infrastructure does not support Fixed IP addresses nor Graceful-replacement IP addresses.

Whenever an application opens an IP socket and requests a specific IPv6 address type, the IP stack will provide one from its available IPv6 prefixes or return an error message if the request cannot be fulfilled.

Message Flow:

- The mobile device attaches to the network.

- The Network provides two IPv6 prefixes: PREFsl1 - a Session-lasting IPv6 prefix and PREFnp1 - a Non-persistent IP v6 prefix.

- An application on the mobile host is launched. It opens an IP socket and requests a Non-persistent IPv6 address.

- The IP stack provides IPnp1 which is generated from PREFnp1.

- Another application is launched, requesting a Non-persistent IPv6 address.

- The IP stack provides IPnp1 again.

- A third application is launched. This time, it requires a Session-lasting IPv6 address.

- The IP stack provides IPsl1 which is generated from PREFsl1.

- The mobile hosts moves to a new point-of-attachment.

- The network provides a new Non-persistent IPv6 prefix - PREFnp2. PREFnp1 is no longer valid.

- The applications that were given IPnp1 re-establish the socket and receive a new IPv6 address - IPnp2 which is generated from PREFnp2

- The application that is using IPsl1 can still use it since the network guaranteed that PREFsl1 will be valid even after moving to a new point-of-attachment.

- A new application is launched, this time requiring a Graceful-replacement IPv6 address.

- The IP stack returns setsc() with an error since the network does not support this service.

- The application re-attempts to open a socket, this time requesting a Session-lasting IPv6 address.

- The IP stack provides IPsl1.

5. Backwards Compatibility Considerations

Backwards compatibility support is REQUIRED by the following 3 types of entities:

- The Applications on the mobile host

- The IP stack in the mobile host

- The network infrastructure

5.1. Applications

Legacy applications that do not support the OnDemand functionality will use the legacy API and will not be able to take advantage of the On-Demand Mobility feature.

Applications using the new OnDemand functionality MUST be aware that they may be executed in legacy environments that do not support it. Such environments may include a legacy IP stack on the mobile host, legacy network infrastructure, or both. In either case, the API will return an error code and the invoking applications may just give up and use legacy calls.

5.2. IP Stack in the Mobile Host

New IP stacks MUST continue to support all legacy operations. If an application does not use On-Demand functionality, the IP stack MUST respond in a legacy manner.

If the network infrastructure supports On-Demand functionality, the IP stack SHOULD follow the application request: If the application requests a specific address type, the stack SHOULD forward this request to the network. If the application does not request an address type, the IP stack MUST NOT request an address type and leave it to the network's default behavior to choose the type of the allocated IP prefix. If an IP prefix was already allocated to the host, the IP stack uses it and may not request a new one from the network.

5.3. Network Infrastructure

The network infrastructure may or may not support the On-Demand functionality. How the IP stack on the host and the network infrastructure behave in case of a compatibility issue is outside the scope of this API specification.

5.4. Merging this work with RFC5014

[RFC5014] defines new flags that may be used with setsockopt() to influence source IP address selection for a socket. The list of flags include: source home address, care-of address, temporary address, public address CGA (Cryptographically Created Address) and non-CGA. When applications require session continuity service and use setsc() and bind(), they SHOULD NOT set the flags specified in [RFC5014].

However, if an application sets a specific option using setsockopt() with one of the flags specified in [RFC5014] and also selects a source IP address using setsc() and bind() the IP address that was generated by setsc() and bound using bind() will be the one used by traffic generated using that socket and options set by setsockopt() will be ignored.

If bind() was not invoked after setsc() by the application, the IP address generated by setsc() will not be used and traffic generated by the socket will use a source IP address that complies with the options selected by setsockopt().

6. Summary of New Definitions

6.1. New APIs

setsc() enables applications to request a specific type of source IP address in terms of session continuity. Its definition is:

int setsc(int sockfd, in6_addr *sourceAddress, sc_type addressType);

Where:

- sockfd - is the socket descriptor of the socket with which a specific address type is associated
- sourceAddress - is a pointer to an area allocated for setsc() to place the generated source IP address of the desired session continuity type
- addressType - Is the desired type of session continuity service. It is a 3-bit field containing one of the following values:
  0 - Reserved
  1 - FIXED_IPV6_ADDRESS
  2 - SESSION_LASTING_IPV6_ADDRESS
  3 - NON_PERSISTENT_IPV6_ADDRESS
  4 - GRACEFUL_REPLACEMENT_IPV6_ADDRESS
  5-7 - Reserved

setsc() returns the status of the operation:

- 0 - Address was successfully generated
- EAI_REQUIREDIPNOTSUPPORTED - the required service type is not supported
- EAI_REQUIREDIPFAILED - the network could not fulfill the desired request

setsc() MAY block the invoking thread if it triggers the TCP/IP stack to request a new IP prefix from the network to construct the desired source IP address. If an IP prefix with the desired session continuity features already exists (was previously allocated to the mobile host) and the stack is not required to request a new one as a result of setting the IPV6_REQUIRE_SRC_ON_NET flag (defined below), setsc() MAY return immediately with the constructed IP address and will not block the thread.

6.2. New Flags

The following flag is added to the list of flags in the IPV6_ADDR_PREFERENCE option at the IPPROTO6 level:

IPV6_REQUIRE_SRC_ON_NET - set IP stack address allocation behavior

If set, the IP stack will request a new IPv6 prefix of the desired type from the current serving network and configure a new source IP address. If reset, the IP stack will use a preconfigured one if it exists. If there is no preconfigured IP address of the desired type, a new prefix will be requested and used for creating the IP address.

7. Security Considerations

The setting of certain IP address type on a given socket may be restricted to privileged applications. For example, a Fixed IP Address may be provided as a premium service and only certain applications may be allowed to use them. Setting and enforcement of such privileges are outside the scope of this document.

8. IANA Considerations

This document has no IANA considerations.

9. Contributors

This document was merged with [I-D.sijeon-dmm-use-cases-api-source]. We would like to acknowledge the contribution of the following people to that document as well:

Sergio Figueiredo
Altran Research, France
Email: sergio.figueiredo@altran.com

Younghan Kim
Soongsil University, Korea
Email: younghak@ssu.ac.kr

John Kaippallimalil
Huawei, USA
Email: john.kaippallimalil@huawei.com

10. Acknowledgements

We would like to thank Wu-chi Feng, Alexandru Petrescu, Jouni Korhonen, Sri Gundavelli, Dave Dolson and Lorenzo Colitti for their valuable comments and suggestions on this work.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, DOI 10.17487/RFC5014, September 2007, <https://www.rfc-editor.org/info/rfc5014>.

11.2. Informative References

[I-D.sijeon-dmm-use-cases-api-source] Jeon, S., Figueiredo, S., Kim, Y., and J. Kaippallimalil, "Use Cases and API Extension for Source IP Address Selection", draft-sijeon-dmm-use-cases-api-source-07 (work in progress), September 2017.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, DOI 10.17487/RFC5213, August 2008, <https://www.rfc-editor.org/info/rfc5213>.

[RFC5563] Leung, K., Dommety, G., Yegani, P., and K. Chowdhury, "WiMAX Forum / 3GPP2 Proxy Mobile IPv4", RFC 5563, DOI 10.17487/RFC5563, February 2010, <https://www.rfc-editor.org/info/rfc5563>.

[RFC5944] Perkins, C., Ed., "IP Mobility Support for IPv4, Revised", RFC 5944, DOI 10.17487/RFC5944, November 2010, <https://www.rfc-editor.org/info/rfc5944>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <https://www.rfc-editor.org/info/rfc6275>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <https://www.rfc-editor.org/info/rfc6824>.

[RFC7333] Chan, H., Ed., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", RFC 7333, DOI 10.17487/RFC7333, August 2014, <https://www.rfc-editor.org/info/rfc7333>.

Authors' Addresses

Alper Yegin
Actility
Istanbul
Turkey

Email: alper.yegin@actility.com

Danny Moses
Intel Corporation
Petah Tikva
Israel

Email: danny.moses@intel.com

Kisuk Kweon
Samsung
Suwon
South Korea

Email: kisuk.kweon@samsung.com

Jinsung Lee
Samsung
Suwon
South Korea

Email: js81.lee@samsung.com

Jungshin Park
Samsung
Suwon
South Korea

Email: shin02.park@samsung.com

Seil Jeon
Sungkyunkwan University
Suwon
South Korea

Email: seiljeon@skku.edu