Socket API Extensions for On-Demand Mobility Management

draft-ietf-dmm-ondemand-mobility-11

Danny Moses

Updates since IETF-98

- Added a new Socket function for requesting IP session continuity type
- Added a new session continuity type – Graceful-Replacement to support 3GPP SSC mode 3

The ‘blocking’ problem

- In Chicago, the WG requested to provide alternatives for the use of setsockopt() due to potential ‘blocking’ issue

- ‘Blocking’ is caused by an invocation that triggers an exchange of packets with the network, in order to request a certain type of session continuity service which may not immediately return

Alternatives for resolution

Three possible alternatives:

1. Add a new blocking function after the call to setsockopt()
2. Replace setsockopt() with a new function that may block, and bind() to bind the generated IP address to the socket
3. Use the new blocking function with a implicit ‘bind’

1. Add a new blocking function

This alternative was suggested in Chicago:

- Call setsockopt() to request a certain IP session continuity service.
- The function triggers a request but returns before the IP prefix was provided by the network
- Call a new function – waitforscservice() – which will return after the service (IP prefix) is provided by the network

Code sequence

- Call socket() // open a socket
- Call setsockopt() // set the required session continuity service:
  // Fixed, session-lasting or non-persistent.
  // This call may trigger the TCP stack in the
  // mobile node to request the address.
- Call waitforscservice() // block the thread until the address with the
  // desired session-continuity service is provided
- Call connect() // start the TCP 3-way handshake
- Start receiving and transmitting bytes

This works but why use setsockopt() at all? We can call the new function an pass the required service…

2. Replace setsockopt() with a new function

This alternative uses a new defined function – setsc() which the app uses to request a session continuity type.

Setsc() will return after the desired IP prefix is provided by the network. It provides the generated IP address structure.

With the returned IP address structure, the app calls bind() and connect()

Code sequence

- Call socket() // open a socket
- Call setsc() // indicate the required session continuity service
  // and block until the address is provided. Return
  // the address that was generated as a result of
  // this request.
- Call bind() // With the provided IP address – bind the socket
  // with the provided source IP address
- Call connect() // start the TCP 3-way handshake
- Start receiving and transmitting bytes

This is better. It also uses bind() to associate the generated IP address with the socket – a natural usage of bind()

3. Use setsc() with an implicit bind

We can avoid the need to bind altogether, by having setsc() implicitly bind the generated IP address to the socket.

With this alternative, setsc() does not return the IP address structure. When it returns, the address is associated with the socket

Code sequence

- Call socket() // open a socket
- Call setsc() // indicate the required session continuity service
  // and block until the address is provided
- Call bind() // With the provided IP address – bind the socket
  // with the provided source IP address
- Call connect() // start the TCP 3-way handshake
- Start receiving and transmitting bytes

Selected the 2nd alternative

The new draft version uses the 2nd alternative. It seemed to us that using an explicit call to bind() fits best with the spirit of the Socket interface.

Support future On-Demand types

Added a new On-Demand type: Graceful-replacement (3GPP's SSC mode 3)

Graceful-replacement means that the IP prefix is not guaranteed to last throughout the lifetime of the IP session, but before it disappears, the network provides a new IP prefix an some time to gracefully switch to the new prefix.

A 3 bit flags is defined for On-Demand types:

0 – reserved

1 – Fixed

2 – Session-Lasting

3 – Non-Persistent

4 – Graceful-replacement

5-7 – Vendor-specific

Any comments?

Next steps

- Resume WGLC