Architectural Design of Unified Multiplex Communications for One-Time Use of IP Addresses

Shingo Ata*, Hiroshi Kitamura†, and Masayuki Murata‡
*Graduate School of Engineering Osaka City University, Japan
Email: ata@info.eng.osaka-cu.ac.jp
†NEC Corporation / University of Electro-Communications, Japan
‡Graduate School of Information Science and Technology, Osaka University, Japan

In the conventional IP network, all servers have a risk against any implicit/explicit attacks from unknown devices because the IP address of the server is fixed during the server is active. In this paper, we try to solve the problem by proposing a new communication architecture where a node has multiple IP addresses. Each IP address is used "one-time," and valid only for a single session (e.g., TCP connection). We first analyze the architecture by itself from the technical viewpoint, and describe the detailed design including new concepts needed to realize. We then describe the technical descriptions of kernel/userland behavior in the proposed architecture.

I. Introduction

We reconsider fundamentally the current communication architecture of "IP address + port number" used for the identification of the endpoint widely, and propose the *Unified Multiplex Communication Architecture* which we utilize the wide IPv6 addressing to discontinue the notion of port numbers [1].

The main thing we have to emphasize is that, even if the address of the server has been known to a third party, the address cannot be used anymore to connect the server in future because the address is only valid for a single session. We call it as *One-time* use of IP addresses.

The motivation behind the use of one-time IP address is to let end users be enable to run (deploy) their private servers (i.e., for internal/private use) easier. The main problem under such a personalized communication is how to stand by communication requests from other nodes. In the conventional IP network, all servers have a risk against any implicit/explicit attacks from unknown devices because the IP address of the server is fixed during the server is active, though the end user intends to access to the server remotely only for him/herself.

This work is partally supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE) by Ministry of Internal Affairs and Communications of Japan.

In this paper, we design the basic technologies to be necessary to realize IP addresses for the exclusive use of each session in Unified Multiplex Communication Architecture. In particular, we focus on TCP (Transmission Control Protocol) which is used widely, and consider what kind of technologies are necessary from the difference of the communication procedure in the current and Unified architectures.

The rest of this paper is organized following. We first describe briefly the architecture of Unified Multiplex in Section II. We then describe the new technologies needed for Unified Multiplex in Section III. We next explain about implementation in Section IV. Summary and future topics are presented in Section V.

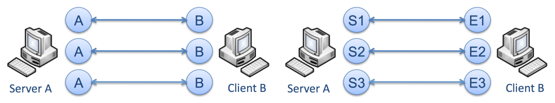
II. Overview of Unified Multiplex Architecture

We explain briefly about the Unified Multiplex communication architecture, Here, we refer the existing communication architecture as *Legacy Architecture* to distinguish from Unified Architecture.

A. Type of Addresses in Unified Multiplex

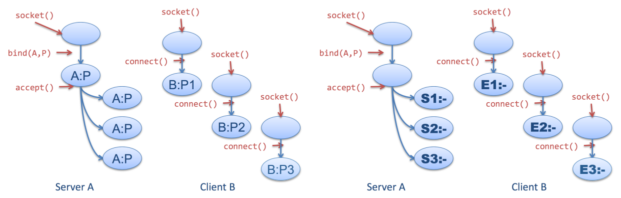
We define two kinds of following addresses to realize an address for exclusive use of the service in Unified Multiplex.

Ephemeral Address (EA) [2] is an address including the equivalent information of the ephemeral port number. EA is assigned at initiating a session (e.g., before TCP connection establishment), and is used throughout the session. The assigned address is ONLY used to the associate session. After the session expires, it is released and disposed (not used anymore) at the termination of the session.



(a) Legacy Communication (b) Unified Communication

Fig. 1. Legacy/Unified Communication



(a) Legacy (b) Unified

Fig. 2. Socket State and Address Assignment

Specific Service Address (SSA) is a kind of one-time addresses and only valid for a single session assigned to server-side. Before the communication, an SSA is generated for a client to communicate with the server and notified to the client. The client establishes the connection to the server by using the notified SSA. After the session expires the address is also released and no longer used in future. The main issue on SSA is how to tell the generated SSA to the client prior to establishing the communication. There are several ways to tell the SSA, one is to handle a DNS query and embed the generated SSA in the DNS reply message. How to solve this problem is not the scope of this paper (See DNSO [3] for detail).

Figure 1 compares communication styles in both Unified and Legacy architectures. In the Legacy architecture, three TCP connections use the same IP addresses (A and B), while the port numbers at Client B are different. These connections are distinguished at both nodes by the port numbers of connections. On the other hand, in the Unified architecture, different ephemeral addresses (E1, E2, and E3) for Client B are assigned to every connection, and different specific service addresses for Server A are also assigned.

B. *Requirements of Unified Multiplex Communication*

1) Communication Procedure in Unified Architecture: The procedure of the communication is almost the same in the Unified architecture, however, the time when the address is associated to the socket is clearly different. In the Legacy architecture, the IP address has already been assigned when the network interface is active, and therefore for network applications the IP address is already known at the phase of socket creation (i.e., `socket()` API). However in the Unified architecture, EA or SSA is not clearly allocated at the time when the socket is being created or the socket is ready to accept connections.

Figure 2 shows the address/port setting to socket in Legacy and Unified environments. In the Legacy architecture, both Server A and Client B have IP addresses (A and B) prior to running the program. That is, the server program easily binds the IP address and port number to the socket by calling `bind()` with parameters (A:P) [1]. After `accept()` is called, a new socket is created for every established connection but address/port information are preserved from the original socket. As the result, sockets for three connections have same property (A:P). For the client program, at the time of calling `connect()`, IP address is already known and ephemeral port number is assigned. After calling `connect()` three sockets in this figure are assigned (B:P1), (B:P2), and (B:P3) respectively.

On the other hand, in the Unified architecture, the SSA is not necessary to be assigned when `bind()` is called, but in the latest case the SSA may be assigned when the new socket has been created for the established connection (S1, S2, S3 in this figure). At the client the EA (E1, E2, E3) is automatically assigned to the socket when `connect()` is called. In both cases, port number is no longer necessary to be assigned to the socket because the EA and SSA already include the equivalent information about port numbers.

Because we assign different IP address to every connection from the client, the IP address for the socket is not determined yet when `listen()` is called to wait connection requests from clients. IP address is actually settled to the socket after `accept()` has been called and a new socket has been created. Also, to accept multiple connections from different clients, different IP address should be set to different newly created sockets for every `accept()` call.

2) Connection Identification in Unified Architecture: In the Legacy architecture, the end node distinguishes the session by using a transmit and receive port numbers as well as transmit and receive IP addresses. These information for the socket are stored into PCB (Protocol Control Block) [4] in the kernel. Because we do not refer to a port number in the transmit and receive of the packet in the Unified architecture, a process to ignore the

[1] A:P means that IP address is A and port number is P

port number when it processes transmitted and received packets is necessary.

3) Address Disposal at Termination of Communication in Unified Architecture: In the Unified architecture, the lifetime of address (EA or SSA) is completely the same as the duration of the session. Therefore, the assigned address should be released and disposed immediately at the end of the session. To achieve it the process to inactivate the IP address when the session has been finalized is necessary.

4) Requirements for Unified Communication: Based on above, requirements to realize TCP in the Unified architecture are as follows.

1) Delay Address Setting at session establishment
   - Assignment of SSA at creating socket
   - SSA assignment at session initiation
   - EA assignment at session initiation
2) Ignoring port numbers
   - At searching PCB
   - In port number fields of sending packets
3) Address release at end of session
   - Client: EA release at closing session
   - Server: SSA release at closing session

III. New Technologies for Unified Multiplex Communication

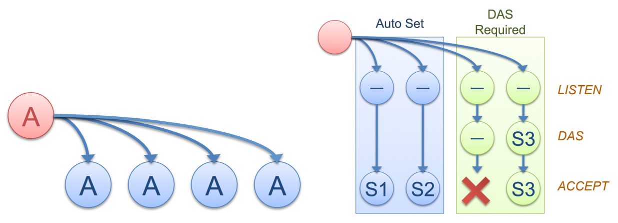
In this section, we describe new functions/capabilities needed to realize the Unified architecture separately. Functions can be divided into three categories: address settings, state transition of TCP, and interoperability.

A. Address Settings

1) DAS (Delayed Address Setting): DAS is a process to set an address in socket after `bind()` is called at the server. This is commonly used to assign a Specific Service Address when `accept()` is called. The detailed procedure of DAS is described in [1].

2) Uncertain State of Addresses: The current DAD (Duplicated Address Detection) [5] is not compatible with Unified architecture. Therefore we define the new state of the address in addition to states defined in RFC 4862 called `UNCERTAIN`. Detailed description about the `UNCERTAIN` state is presented in [6].

3) AUTO_SET (Auto Address Setting): In the Unified architecture, SSA/EA must be assigned to every connection when the connection has been established. To get connected to the server with the SSA, the explicit value of SSA should be determined and notified to the client which is used to call `connect()`. The server



(a) Legacy (b) Unified

Fig. 3. Addresses assigned to socket in Legacy/Unified architectures

then waits the connection from the client for the notified SSA. To resolve this situation, we modify the procedure of `listen()` in following. We set explicit value of SSA for every element of the backlog queue of connection requests, whose length is specified in `listen()`. Each socket in the backlog can set two following attribute as principle of the address assignment.

- **Auto Set:** Address is automatically set by the kernel after `listen()` has completed. The address that the client will use is obtained from the set of addresses assigned by `AUTO_SET`.
- **DAS Required:** Kernel does not assign any addresses to sockets. Address should be assigned manually by `DAS` before `accept()` is called.

Figure 3 compares addresses assigned to sockets between the Legacy and Unified architectures. Red circles are original sockets and blue and green sockets are sockets derived (newly created) sockets. In the Legacy architecture all sockets in the backlog queue have the same IP address A, which is also the same as the address of the original socket. On the other hand, when `listen()` has been called, addresses of sockets are not determined in both “Auto Set” and “DAS required” cases.

4) Address Generation Methods: The procedure of the address generation is also an important role in the Unified architecture. As simple approaches, sequential, or random value generation can be applied. However, *well-considered* address generation would be able to add a new capability or feature in the communication. For example, random address generation can improve the privacy that a third-party cannot identify the node easily by capturing communication packets. A discussion on the address generation is presented in [7].

B. State Transition of TCP

As described before, in the Unified architecture the SSA is assigned to the socket after `bind()` is called.

TABLE I
INTEROPERABILITY WITH LEGACY ARCHITECTURE NODES

Mode Level	Use of port number for distinguishing sessions		Destination port number in packets	Available Communication Style Server(Addr, Port) – Client (Addr, Port)
	Server	Client		
Port ignore L0	No	No	Any	U(SSA, *) — U(EA, *)

Port ignore L1	No	No	Source port number of received packet	U(SSA, *) — U(EA, *)
				U(SSA, *) — L(LA, EP)

Port ignore L2	No	Yes	Source port number of received packet	U(SSA, *) — U(EA, *)
				U(*, *) — L(LA, EP)

Port aware	Yes	Yes	Source port number of received packet	U(*, WP) — L(LA, EP)
				U(*, RP) — L(LA, EP)
				U(*, WP) — L(LA, RP)
				U(*, RP) — L(LA, RP)

Node	U: Unified, L: Legacy
Address	SSA: Service Specific Address, EA: Ephemeral Address, LA: Legacy Address
Port	EP: Ephemeral Port, RP: Reduced Port (LEGACY_COMPAT), WP: Well-known Port

There is a problem that the node cannot accept a connection request immediately after `bind()` has been called, because there is no valid SSA prepared for the socket. To solve this problem, we re-define the state transition of TCP slightly. Specifically, we subdivide the state `LISTEN` into two sub-states `LISTEN0` and `LISTEN`. `LISTEN0` indicates that the socket is ready for waiting to accept connections, but not ready to actually accept the connection due to lack of SSA. To transit `LISTEN0` to `LISTEN` the address assignment to the socket is required by DAS. Note that if the attribute of the socket is "Auto Set", the address is assigned automatically and the state is changed from `LISTEN0` to `LISTEN` immediately.

C. *Interoperability*

When we consider about transition to the Unified Multiplex communication architecture, it is unrealistic that all nodes cope with a Unified architecture at the same time, and it is necessary to consider about coexistence with the Legacy architecture nodes. For Legacy nodes, interoperability with Unified nodes is important because the Unified architecture particularly discontinues to use the concept of port numbers, while the Legacy architecture port numbers are used to identify the session.

According to the combination of operations, we consider what kind of nodes can communicate with each other. The summary is show in Table I.

IV. IMPLEMENTATION

Currently, the proposed Unified Multiplex Architecture has been implemented in FreeBSD 6.2R which includes:

- Modification of major socket APIs to support EAs, SSAs and UNCERTAIN state
- Userland program for Delay Address Setting (DAS)
- `sysctl` parameters for interoperability
- Modification of userland commands `ifconfig`, `sockstat`, `netstat` to handle EAs/SSAs

Following is an example that 2001::1 is assigned as an uncertain pool address to be used for EA/SSA.

```
# ifconfig lnc0 inet6 2001::1 pool
# ifconfig lnc0 inet6
  lnc0: flags=108843<UP,BROADCAST,RUNNING,
  SIMPLEX,MULTICAST,NEEDSGIANT> mtu 1500
    inet6 fe80::20c:29ff:feb8:9d7c%lnc0
      prefixlen 64 scopeid 0x1
    inet6 2001::1 prefixlen 64 uncertain pool
```

To use the pool address for SSA, simply run `das` command to associate the pool address with the PID of the server which intends to use.

```
# sockstat -6
  USER COMMAND PID FD PROTO   LOCAL FOREIGN
  root sshd    719 3  tcp6    *:22  *:*
# das 719 lnc0 2001::1
```

V. CONCLUDING REMARKS

In this paper, we have described the design of the basic technologies to be necessary to realize the Unified Multiplex communication architecture. Through the analysis of behavior in Unified architecture, we have created some new concepts that are applicable to conventional architecture. We have already completed implementation in kernel and userland command on FreeBSD 6.2R based on this design. As future topic, we need to improve the design based on the feedback from experimental evaluation.

REFERENCES

[1] H. Kitamura, S. Ata, and M. Murata, "Communication architecture evolution enabled by introducing specific IP address for each session - unified multiplex communication architecture -," *submitted for publication*, Sept. 2010.

[2] H. Kitamura, S. Ata, and M. Murata, "IPv6 ephemeral addresses," *Internet Draft, draft-kitamura-ipv6-ephemeral-address-01*, July 2009.

[3] K. Shima and H. Kitamura, "IPv6 global communication architecture build on a mechanism for conveying service dedicated address information (DNSO)," *IEICE Technical Report (IN2008-29)*, Mar. 2008.

[4] W. R. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation.* Reading, Massachusetts: Addison-Wesley, 1995.

[5] S. Thomson, T. Narten, and T. Jinmei, "IPv6 stateless address autoconfiguration," *RFC 4862*, 2007.

[6] H. Kitamura, S. Ata, and M. Murata, "Harmless IPv6 address state extension (uncertain state)," *Internet Draft, draft-kitamura-ipv6-uncertain-address-state-01*, July 2009.

[7] K. Sakakima, S. Ata, and H. Kitamura, "Anonymous but traceable ip address-based communication system," *Proceedings of NetCom 2009*, Dec. 2009.