A TCP Service Migration Protocol for Single User Multiple Devices

Chi-Yu Li, Ioannis Pefkianakis, Songwu Lu
Computer Science Department
University of California, Los Angeles
Los Angeles, California, USA
{*lichiyu, pefkian, slu*}*@cs.ucla.edu*

Bojie Li, Chenghui Peng, Wei Zhang
Huawei Technologies co.
Shenzhen, China
{*libojie, pengchenghui*}*@huawei.com*
{*wendy.zhangwei*}*@huawei.com*

***Abstract*—In this paper, we present a new transport protocol TSMP, which seeks to support data transfer for the emerging usage paradigm of ”single user, multiple devices” in a TCP compatible manner. Through its novel naming and proxy-based designs, TSMP is able to retain the current client and server protocol operations of the legacy TCP protocol and TCP-based applications while placing new functions at the proxy. Our evaluation has confirmed its viability.**

***Keywords*-Single-user, multi-device; TCP migration; naming; proxy.**

I. Introduction

In this work, we seek to design protocol solutions for an emerging usage scenario of ”single user, multiple devices.” In recent years, it has become increasingly popular that a user owns several devices with networking capabilities. A survey of the percentage of American adults who own each device [1] shows that several tens percentage of American adults have more than one devices, in which 35% of adults own a smartphone, 59% of adults own a desktop and more than about one in two adults own a laptop. Therefore, an example scenario may become common: a user has a laptop in the office, a desktop at home, while carrying an iPhone or iPad wherever (s)he goes. This emerging single-user, multiple-device setting calls for new innovations in networking protocol design to make it more efficient.

To this end, we describe a novel solution, called TCP Service Migration Protocol (TSMP), that supports ”single-user, multi-device” TCP communications. TCP has been the dominant transport protocol for most Internet applications, and many popular applications such as web-based video streaming, and Instant messaging (e.g., MSN) are based on its operation. There are two main design challenges. First, the protocol operations should support TCP-based data transfer among multiple devices of the same user. TCP sessions should be able to seamlessly migrate among the devices owned by the same user. For example, a user does instant messaging or video streaming on his laptop when he is in his office. When he walks out for lunch, he can proceed the ongoing messaging or video session via his iPhone or iPad. Second, users are able to continue to run legacy TCP and applications with minimal changes at both sides of the client and the server while supporting the notion of single-user, multiple-device in data communications. This will enable reuse of most existing Internet applications. Existing protocols can achieve one of these two goals, but not both.

In this paper, we describe a novel solution, called TCP Service Migration Protocol (TSMP), that supports ”single-user, multi-device” TCP communications. The TCP connection is associated with the user and can seamlessly migrate among the devices belonging to the same user. A key innovation in TSMP is the proxy bridging the client and the server in the existing client-server communication model. The proxy offers two critical services of naming and TCP control/data plane functions. By carefully designing the proxy, TSMP is able to reuse existing TCP and TCP-based applications at both the client and the server without modifications. Our initial evaluation has confirmed the effectiveness of TSMP design.

The rest of the paper is organized as follows. Section II illustrates the usage scenario and identifies the design requirements. Section III describes the related work, and Section IV presents the architecture and TCP control/data-plane solution. Sections V elaborates on the naming management. Section VI evaluates TSMP and Section VII concludes the paper.

II. Single User Multiple Devices

In this section, we first present an example of our intended scenarios and then identify the requirements for our design. We also discuss the applications of our protocol.

A. An Example Scenario

As shown in Figure 1, Bob has three networked devices: PC at home, Laptop in the office, and Smartphone, which he uses while moving. He chats with his friend, Alice, over an instant messaging (IM) application using his smartphone while he is on his way home. In the mean time, Alice wants to share video clips with Bob using HTTP streaming from her web server at her PC. After arriving at home, he switches both the IM session and the HTTP progressive downloading of the remaining videos to his home PC because of its comfortability and larger bandwidth. Then, Bob chats with Alice and watch the latter part of the video on his PC.

Moreover, the service migration among Bob's devices is not perceived by Alice.

Our goal is to design a solution that supports data service migration for one's multiple devices, so that each user can use the most appropriate device for each different situation.

B. System Issues

In our system, we consider data service migration based on the TCP protocol, and thus need to address the issues of migrating a TCP connection among two devices in addition to the single-user, multiple-device naming issues. First, how to keep the intended connection open during migration and prevent the end which is not involved from perceiving the migration? Second, how to transfer TCP connection states from one device to another and make the overhead to cause as less impact as possible on the connection? There may be some transient loss during migration, which may result in shrinking the congestion threshold (CongestionThreshold) or interrupting the connection. The too small value of CongestionThreshold would prevent the congestion window (CongestionWindow) from growing quickly to the appropriate size and a large amount of delay may thus occur. Third, most current naming schemes do not support the feature that one user owns multiple devices. Forth, the IP address has the decoupling roles, particularly as the identifier (ID) and as the locator. The ID of an end device, which is for long-term usage, should not change frequently with the locator which is transient due to mobility.

C. System Requirements

We have several requirements for our system to address the above issues in the following three aspects.

1) Service Migration: To support service migration, the system needs to consider both control-plane and data-plane. The control plane is used to coordinate the operation of service migration, which includes triggering the migration process, discovering the device to which the service is migrated and inform the new device to accept the migration. During service migration, the data plane should be able to cache the transient packets which have been sent by the sender but have not been acknowledged, and make these packets as few as possible to reduce the overhead. After the migration is complete, it will send all the cached packets to the new receiver and then resume the original TCP connection. Its another important task is to avoid retransmission timeout to keep the same value of CongestionThreshold. Service migration may happen from one device with low bandwidth to another with high bandwidth or from the latter to the former. In the former case, we need to make the CongestionThreshold value as large as possible so that the sender's CongestionWindow can grow quickly with the slow start algorithm, to the appropriate size for the larger bandwidth. If the CongestionThreshold value becomes too small, the size of CongestionWindow would increase very slowly because it grows linearly with the additive increase/multiplicative decrease (AIMD) after it exceeds the threshold. However, the CongestionThreshold cannot increase without data transmission so that the best way we can do is to maintain the same value of CongestionThreshold. As for the latter case, the CongestionWindow can shrink quickly to the appropriate size due to the multiplicative decrease.

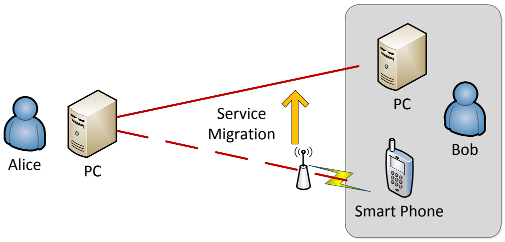

Figure 1. Service is migrated from Bob's phone to his PC.

2) Naming: The namespace should support both the user ID (UID) and the device ID (DID), and be able to map each UID to multiple DIDs. To prevent ID from transiently changing with the locator, the roles of IP address need to be decoupled and a mapping layer should be provided to map each DID to its current IP address which plays the role of only the locator.

3) Backward Compatibility: In order to have good backward compatibility and easy deployment, our solution should be designed with the least modifications possible on the existing systems and applications.

D. Applications

We aim to apply our TSMP protocol to the applications which are based on the TCP protocol. They include two popular types of applications: HTTP video streaming and IM applications. Apple's HTTP Live Streaming [2] is based on the former and Flash Video [3] also supports this video streaming feature. Moreover, many notable users of Flash Video include Youtube, Google, Yahoo and so on. The latter are the applications for real-time text-based communication, such as Skype and Window Messenger. Our solution seeks to bypass the modification of them for easy deployment.

III. Related Work

In this section, we present the solutions which have been proposed to deal with the migration of TCP connections and the signaling protocols which are used for controlling communication sessions. We also introduce other naming schemes, which are based on the Identity/Locator split.

A. TCP Migration

In order to support the migration of TCP connections for satisfying various mobility scenarios, a number of solutions have been proposed. They can be classified into two categories: split connection [4]–[7] and non-split connection

[8]–[11] approaches. Our proxy-based solution falls into the former. All these schemes cannot achieve both of our two goals: no requirements of modifying both applications and the TCP protocol, and enabling a TCP connection migrating between two devices.

The split connection approaches always divide a TCP connection into two sub-connections by inserting a proxy or module into the communication path between two ends. MSOCKS [4] builds its transport layer mobility architecture around a proxy, which enables mobile nodes to migrate data streams between network interfaces or different networks. The way in our solution that the proxy mediates between two sub-connections is similar to this scheme, but both its objective and method of migrating connections are different from ours. A MSOCK socket library sitting between the application layer and the kernel socket is introduced to allow applications to operate on this architecture. Although the applications do not need to be modified, they have to be recompiled with the MSOCK library. I-TCP [6][7] aims to deal with handoff for mobile devices by breaking a TCP connection into two parts, one for the wireless link and the other for the wired link. TCP snoop [5] uses the same way to improve TCP performance in wireless networks. The former employs a new transport protocol with mobility and wireless awareness on the sub-connections in wireless networks, whereas the latter performs local retransmissions based on a few policies dealing with acknowledgements and timeouts.

The schemes in the direction of the non-split connection either modify the existing TCP implementations or introduce a shim layer between the application and the TCP protocol stack. TCP Migrate [8] maintains an established TCP connection while a mobile host's IP address changes by introducing a new Migrate option into the TCP protocol. An open TCP connection can accordingly be temporarily suspended and be reactivated later from another IP address using a special Migrate SYN packet. Migratory TCP [9] supports the migration of a TCP connection, between servers, for service continuity and availability in case of failures. MSL [10] introduces a shim layer, Mobile Socket Layer, which enhances the existing socket implementations to support uninterrupted TCP associations on the devices moving among different networks. It mediates between the application layer and the TCP/IP protocol stack. During mobility, the broken TCP connections are hidden from applications and then reset when mobile hosts move to a new location. SockMi [11] migrates TCP connections by transferring socket states and in-flight data between different devices. A SockMi module placed under the application layer is introduced to coordinate the socket migration, and devices communicate with each other through their SockMid daemons. With these two schemes, applications need to be modified based on their new APIs.

B. Signaling Protocol

The Session Initiation Protocol (SIP) [12] is a widely-adopted signaling protocol for controlling communication sessions, and its applications include video streaming, instant messaging, and file transfer. With its re-INVITE message, users can modify an ongoing session by attaching a new session description. The modification can involve changing addresses or ports, adding or deleting a media stream, and so on. For seamless migration, SIP only works for the sessions which consist of stateless connections, such as UDP-based RTP connections for some video streaming applications, because TCP connections still need to be interrupted and then reestablished after their addresses change if they are migrated among different devices. However, it can be the reference for our SMP signaling design.

C. Naming Support

The naming scheme of our TSMP protocol is based on the Identity/Locator split architecture which a large amount of researchers use to address mobility issues.

A number of protocols [13]–[18] are designed to support mobility using Service ID (SID), UID or DID above the transport layer. C2DM [13] and APNS [14] are currently the two most popular solutions, which are developed by Google and Apple respectively. C2DM identifies users using Google user accounts of which each device should include at least one, whereas APNS identifies devices with opaque device tokens. Both of their purposes are to help the third-party application servers forward small messages or notifications to mobile devices. They cannot associate a user to multiple devices with considering either only UID or only DID. INS [17] and DONA [18] provide service-oriented mobility so that each service is assigned a SID and the location it resides can be resolved over their constructed overlay networks. However, they are unable to bind multiple devices to single user. Haggle [15] and SBone [16] present the concept of identifying both users and devices, which can satisfy the naming requirements of our single-user, multi-device scenario. The former's goal is to deal with mobility issues, whereas the latter's is to provide device sharing among people. They are different from our goal which seeks to enable the migration of ongoing service.

Some schemes [19]–[22] introduce DID to replace the identifier role of IP address and make transport protocols bind to it instead of IP address. An ongoing session of the transport layer would not be interrupted as the IP address of either end changes. Their drawback is that the transport protocols need to be modified. Among these schemes, only UIA's naming scheme [21] can fit in with our intended scenario. It presents a personal namespace, which includes the identities of both users and devices, to organize the user's social network and manage devices.

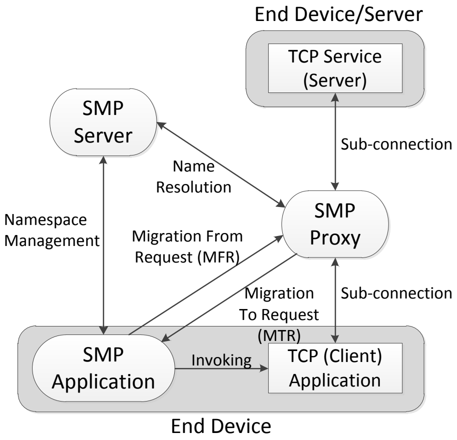

Figure 2. SMP architecture.

IV. TCP Service Migration Protocol (TSMP)

In this section, we first present the SMP architecture. Then, we describe our proxy-based solution for TCP migration and illustrate its complete procedure with an example.

A. Architecture

We employ a proxy-based solution to achieve TCP service migration. As shown in Figure 2, the SMP architecture is composed of three major components: SMP Proxy (SMPP), SMP Server (SMPS) and SMP Application (SMPA). SMPP is interposed between client and server to relay packets from either end to the other and mediate the sub-connections of each TCP connection. In order to bypass the modification of the existing systems and applications, it collaborates with SMPS and SMPA to support the service migration process. SMPA provides an interface for users to make use of the TSMP service and a channel for SMPP to interact with the applications at devices. Each device has an installed SMPA which maintains a namespace group for its owner. The namespace group allows users to manage their own devices and contact others conveniently. SMPS takes care of the global namespace and provides the service of DNS-like name resolution. For namespace consistency and mobility support, there are some functions of namespace management, which are provided for SMPS and SMPA.

B. Proxy-based Solution

SMPP consists of two planes, control plane and data plane. The former coordinates the service migration process, whereas the latter forwards packets between two ends and emulates as a TCP sender to set up a new sub-connection to the new receiver when TCP migration is requested.

1) Control Plane: The control plane coordinates the operation of service migration using two control messages: Migration From Request (MFR) and Migration To Request (MTR). MFR is always sent by SMPA to request SMPP to migrate a TCP connection from the device, where it resides, to another. It should include both the identity of the intended device and the information of the migrated connection so that SMPP can resolve the device's IP address by querying SMPS and identify the connection. The other control message, MTR, is used by SMPP to ask SMPA for invoking its local application to set up a connection to SMPP. Then, SMPP will hook this new sub-connection up to the old sub-connection of the other end, and thus recover the migrated TCP connection.

2) Data Plane: SMPP bridges between the two ends for each TCP connection by forwarding packets from either side to the other, so each connection is divided into two sub-connections which are glued by a mapping table in SMPP. The mapping entry of a connection contains an address pair of each end, IP address and port number. When SMPP receives packets from one end's sub-connection, it replaces the source and destination information with the SMPP address and the other end's address respectively, and then forwards them to the other sub-connection.

For our proxy-based solution, we need to enable the TCP applications to connect to SMPP. Most TCP applications allow users to configure the proxy connection settings, so it can be done by providing the SMPP information to the intended applications. For example, both Windows Live Messenger and Skype Messenger support the SOCKS5 [23] and HTTPS [24] proxies, and most web applications can operate with not only them but the HTTP proxies.

C. TCP Migration

When SMPP receives a MFR request, it will start the migration process of the requested TCP connection. The main concept is that it temporarily pauses the TCP flow until the connection between the new device and SMPP is established, and then resumes it, so the process consists of two phases, transient pause phase and resumption phase. The pause phase aims to freeze the sending process and cache all the outstanding packets which have not be forwarded by SMPP, as well as keep the value of CongestionThreshold unchanged by preventing unnecessary congestion control invocations at the sender. The purpose of the first two actions is to prevent transient loss and keep the connection open, whereas the last action seeks to decrease the overhead of increasing CongestionWindow to the appropriate size of the new sub-connection after the migration finishes. In the resumption phase, SMPP emulates a TCP end to set up a connection to the new device and flush the cached packets to it, and then recovers the sending process. After the connection is resumed, SMPP continues the forwarding process and the old sub-connection is interrupted.

1) Transient Pause Phase: This phase is launched once MFR is received by SMPP, and it does not end until the migration is complete. It is mainly composed of three tasks: advertising the size of the receiver's window to be zero, stopping to forward data packets but caching all of them, and being in response to the zero-window probing.

In the TCP flow control mechanism [25], the receiver can advertise its window with the size zero to stop the sender sending data. The sender does not resume the sending until the advertised window is larger than zero. We employ this feature to stop the sending process by modifying the window size of the TCP headers to be zero in the ACK packets which are forwarded after this phase begins. SMPP should continue to forward the ACK packets which acknowledge the data packets it has forwarded to the old receiver before this phase. The sender thus pauses its sending process, and does the zero-window probing by sending at least one octet of new data periodically. Its purpose is to attempt recovery and guarantee that the re-opening of the window can be reliably reported. During the migration period, SMPP should generate and send an ACK packet, which shows the next expected sequence number and the window size zero, in response to each probe segment. Therefore, the sending TCP would allow the connection to stay open and temporarily freeze the sending process without shrinking the value of CongestionThreshold. We can use the maximum sequence number of the cached packets plus one to be the expected sequence number.

Another task for this phase is to cache the transient packets which have not been forwarded. SMPP starts to cache data packets and stop to forward them once this phase begins. These cached data packets have been sent out by the sender so that the retransmission timeout will be triggered if they are not acknowledged. SMPP accordingly needs to generate and send their ACK packets to the sender in advance for the new receiver. These ACKs should also contain the same information of the expected sequence number and the window size. SMPP needs to make sure that it caches the data segments with all the sequence numbers between the expected sequence number and the acknowledge number of the last ACK packet that the old receiver sends. There may be a case that the old receiver does not acknowledge all its received packets before it tears down the connection. However, these packets would not be cached by SMPP because they have been forwarded. Intuitively, SMPP can just send ACKs to trigger retransmission at the sender and cache them, but the side effect is that CongestionThreshold will be reduced. We can enable SMPP to cache a certain amount of packets to compensate this situation no matter whether it is in the migration state. If there are still some missing packets, relying on the retransmission would not be avoided. We can estimate the cache size with a half of RTT between the sender and the receiver.

2) Resumption Phase: When the new device requests for a new connection due to its SMPA's invocation, the resumption phase begins. SMPP emulates a TCP end to do three-way handshaking with the device and starts to send its cached packets to it. As a TCP sender, SMPP maintains some connection states: CongestionWindow, Congestion-Threshold, and so on. It uses the slow start mechanism when the sending process is initialized or the connection times out, and employs the AIMD algorithm after CongestionWindow reaches CongestionThreshold. SMPP does not forward their ACK packets to the sender. After all the cached packets are acknowledged, SMPP resumes the sender's sending process by forwarding the new receiver's last ACK it receives. The transmission is thus recovered due to the last ACK with a non-zero receive window. SMPP will return to the normal forwarding phase, and discard the emulated TCP states. An issue we need to consider is that the initial sequence number which is chosen at random may result in different sequence number systems between the old sub-connection and the new sub-connection. For this reason, SMPP should add the mapping information of their sequence numbers into the mapping entry of this connection, and modify each packet's sequence number before forwarding it.

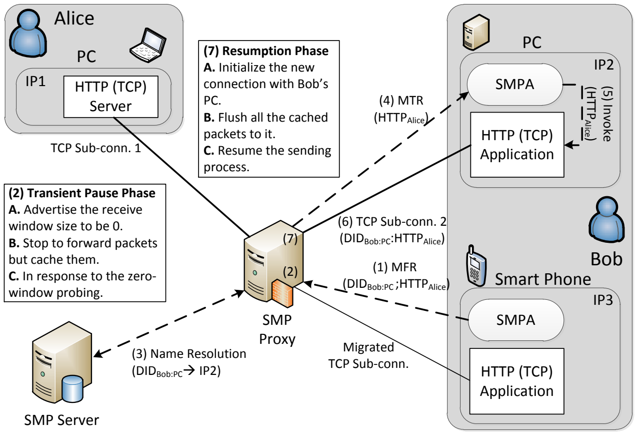
Alice
PC
IP1
HTTP (TCP) Server
TCP Sub-conn. 1
(7) Resumption Phase
A. Initialize the new connection with Bob's PC.
B. Flush all the cached packets to it.
C. Resume the sending process.
(2) Transient Pause Phase
A. Advertise the receive window size to be 0.
B. Stop to forward packets but cache them.
C. In response to the zero-window probing.
(4) MTR (HTTP$_{Alice}$)
PC
IP2
SMPA
HTTP (TCP) Application
(5) Invoke (HTTP$_{Alice}$)
(6) TCP Sub-conn. 2 (DID$_{Bob:PC}$:HTTP$_{Alice}$)
(7)
(2)
(1) MFR (DID$_{Bob:PC}$;HTTP$_{Alice}$)
Bob
Smart Phone
IP3
SMPA
HTTP (TCP) Application
SMP Proxy
Migrated TCP Sub-conn.
(3) Name Resolution (DID$_{Bob:PC}$→ IP2)
SMP Server

Figure 3. TCP Service Migration Procedure: Bob switches Alice's transmission from his Phone to his PC.

D. TCP Service Migration Procedure

In this section, we present the procedure of TCP service migration using an example scenario, as shown in Figure 3. Bob requests a video from Alice's HTTP streaming server using his smart phone. After he arrives at home, he wants to switch the video transfer from his phone to his home PC. In this figure, the dotted lines represents the actions in the control plane whereas those in the data plane are presented by the solid lines. DID represents the device identity, which we will discuss in the next section.

After receiving Bob's migration request, the SMPA at his phone issues a MFR message with the identity of Bob's PC and the information of the migrated connection. The transient pause phase is triggered in SMPP by the MFR and the control plane resolves from SMPS the location of Bob's PC with its DID at the same time. Then, the control plane sends a MTR message to the SMPA at his PC, with the information of the Alice's streaming server. The video application at Bob's PC will be invoked and requested to set up a connection to SMPP. As soon as SMPP receives the connection request, it launches the resumption phase and then returns to the normal forwarding state after this phase ends. The pause phase also ends with it.

V. Naming and Namespace Management

We next introduce the design of namespace in the SMP system and some fundamental management functions.

A. Naming Principles

The namespace is designed based on both the ID/Locator split technology and the requirement of the single-user, multi-device scenario. We organize it into three layers: Name, ID and Locator. They are joined with two-dimensional (2D) mapping: Name to ID to Locator, User ID (UID) to Device IDs (DIDs).

1) Name/ID/Locator: The SMP system maintains a namespace group for each user, which is shown in the SMPAs at his/her devices. Users recognize friends and devices using user name (UN) and device name (DN) respectively in their namepspace groups. In each namespace group, the names, which are changeable and human-readable, are assigned by its owner. The UN of each friend should be unique and the DN of each device needs to be unique in its owner's device set. We introduce two identities, UID and DID, which identify user and device respectively. DID substitutes for the identity role of IP address so that the IP address serves as only the locator. Both of them are globally unique and persistent. The email addresses used to register the SMP system by users are considered as their UIDs. A device's DID in DNS-like dotted notation is generated by combining its owner's UID with the device name which is specified by its owner. For example, Bob registers his UID as *bob@ucla.edu* and the DID of his laptop, named *laptop* at its registration, would be *laptop.bob@ucla.edu*.

2) 2D Name Resolution: Each locally unique UN or DN is associated with a globally unique UID or DID respectively, and each DID can be mapped to its care-of IP address (CoA). The former mapping is maintained in each namespace group, whereas the latter is managed in the global namespace in SMPS. Devices can discover each other with the peer's DID through the SMPS's DNS-like name resolution service. Another dimension of mapping is between a UID and (multi-)DID as a user may own more than one device. It can be done by the identity itself because each DID contains its owner's UID.

B. Namespace Management

Each user has a namespace group in the SMPAs of his/her devices, in which (s)he manages his/her own devices and keeps the information of his/her friends and their devices. SMPS manages the global namespace which includes the information of all the namespace groups, as well as devices' CoA and status. A namespace group is constructed by two functions: service registration, and users and devices introduction. Moreover, the namespace state synchronization is introduced to keep the global namespace to be consistent with each namespace group. Based on the global namespace, SMPS provides name resolution and mobility management.

1) Service Registration: Each user needs to register the SMP system with his/her email through an installed SMPA at any of his/her devices before using TSMP service. His/her namespace group will then be created, which initially contain only the information of the device used for registration.

2) Users and Devices Introduction: In the SMP system, users or devices can introduce with each other using two schemes: Local Rendezvous and Centralized Coordination. The owner(s) of two devices can connect both of them to a common local area network such as WiFi and apply the local rendezvous tool in SMPA, which is similar to Apple's Bonjour [26], to find each other. One end initializes the introduction process and the other needs to acknowledge the request. If both of them belong to the same owner, one of the devices should be newly introduced and will be added into the owner's personal namespace. However, if their owners are different, that is, users introduction, each user will add the other into his/her namespace group , and assign UN and DNs to him/her and his/her devices, respectively. The new state of each namespace group will be updated to SMPS. The medium used for local rendezvous is not limited to WiFi, since Bluetooth, E-mail, SMS message can also be applied.

Two users can also introduce with each other through the SMPS coordination. One user needs to issue a request to SMPS through the SMPA of any his device, and in turn this request will be sent to all of the other's devices. As long as s(he) confirms it on any device, each user's personal namespace will be inserted into the other's namespace group. If a user wants to introduce his/her own new device into his personal namespace, (s)he can issue a request to the SMP system through the device's installed SMPA, and assign a DN to it. The namespace groups with this personal namespace will then be updated.

3) Namespace State Synchronization: Each SMP device needs to maintain its status, because only the on-line devices can be requested to accept TCP service migration. When a device is on-line, its SMPA sends a heartbeat message to SMPS periodically to maintain the status and synchronize its namespace group. SMPS then replies with a message to inform the SMPA of the status of all the devices in its namespace group. When SMPS identifies a lack of heartbeat messages after a time period, the device's status would become off-line. To reduce the overhead of namespace synchronization, only the latest modification timestamp of the namespace group is included in the heartbeat message. If it is different from the corresponding timestamp in SMPS, the SMPA will start to synchronize its namespace group with

SMPS. The fact that many modifications may happen after the latest synchronization may result in conflicts between SMPS and SMPA. We make SMPS and SMPA to keep all the logs of those modifications, so they can get the updated information by reorganizing the updates and applying them in time order. After resolving conflicts, they update the current time to their latest modification timestamp.

4) Name Resolution and Mobility Management: SMPP can resolve the CoA of each DID using the name resolution service provided by SMPS. As for mobility management, each SMPA continually monitors the change of its device's CoA. When it is detected, the new CoA will be immediately updated to SMPS by the SMPA. It also informs SMPS of the information of the ongoing connections in which this device is involved if there is any. Then, SMPS can notify SMPP of changing the IP addresses of these connections' mapping entries.

VI. EVALUATION

The primary goal of TSMP is to allow a user to get data transmission using the most appropriate device for each different situation. We can examine TSMP's performance by evaluating how much overhead it would incur in various settings. The overhead we want to measure is the delay of the TCP migration process which begins at the time that MFT is issued by the receiver and ends at the time that the receiver advertised window is reopen at the sender.

A. Experimental Setup

We evaluate TSMP using NS2 with some measured numbers of the processing delay of SMPS, SMPP and SMPA. We generate TCP traffic with the FTP source and use the module of TCP NewReno. In the topology, except for SMPP and SMPS, there are one server, a pair of the clients which are involved in TCP migration, and multiple pairs of senders and receivers have TCP connections through SMPP. In each experiment, the server sends a 0.8MB file to a client and it always triggers the migration at the fifth second. We conduct three different scenarios of TCP migration: from a WLAN device to another WLAN device, from WLAN to 3G and from 3G to WLAN. We configure the processing delay of SMPS to be 200ms per request based on the statistics of Twitter servers [27] which are 8 Sun X4100s with over 16GB of memcached, and serve over 350000 users, average 600 requests per second. The processing delay of each packet in SMPP is set as 380ns, because it takes about 4000 CPU cycles to send a packet from the driver layer to the application layer based on the settings: Quad-Core Intel Xeon 5355 processor at 2.66GHz and Intel 10Gbps 82598 server NIC adapter [28].

The network latency between SMPP and the SMP device is based on the measured latency between our devices and Google server, because the SMP system may be deployed as a nationwide service in the future. We use the Ping tool to get the approximate round trip times for both 3G and WLAN networks. The round trip time between our iPhone and Google server is average 740ms through 3G network, whereas that between our PC and Google server is average 38ms over WLAN. However, we assume that both SMPP and SMPS are provided by the same provider so that the network latency between them could be very small and the bandwidth is very large. They are thus set as 1ms and 10Gbps, respectively. The bandwidth of SMPP is 1Gbps and that between each device and SMPP is 1Mbps if they are not specified. However, we do not consider the delay of invoking an application by SMPA, because it may vary dramatically with different applications and platforms.

	WLAN	3G	WLAN-WLAN	WLAN-3G	3G-WLAN
Tx Time (s)	7.05	33.04	7.69	25.87	12.42

Table I
THE TRANSMISSION TIME FOR DIFFERENT SCENARIOS.

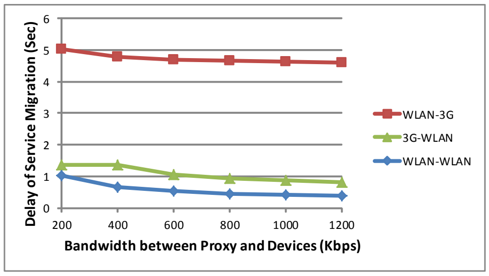

Figure 4. The migration delay varies with different bandwidths and involved networks.

B. The Migration Overhead

We examine the delay of TCP migration by varying the bandwidth between SMPP and the SMP devices which are involved in the migration. The number of concurrent sessions at SMPP is set to 100. As shown in Figure 4, the WLAN-3G scenario results in the higher delay than the others do. It is because the 3G network has longer round trip time, and some packets need to be exchanged upon it to initialize the new TCP subconnection and resume the sending process. Both the TCP three-way handshaking and flushing the cached packets of SMPP to the new receiver incur the major proportion of the overhead. It needs at most about 5 seconds when the bandwidth is higher than 200 Kbps. However, the WLAN-WLAN has the lowest overhead due to its low network latency. We can also understand that the network latency dominates the overhead, compared with the bandwidth. Table I shows the transmission time that is needed for each scenario to send a 0.8MB file. Even if there is some overhead of the migration, it is worth for the 3G-WLAN scenario which saves more than 20 second in the transmission. This is the major scenario of TCP migration, which can benefit people. As for the convenience of mobility, people need to sacrifice some performance with the WLAN-

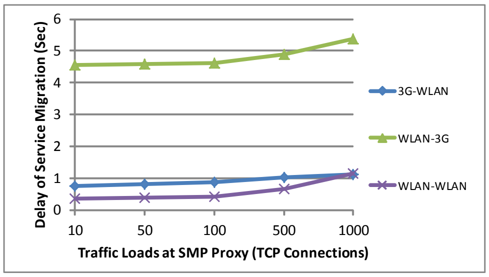

Figure 5. The migration delay varies with the traffic loads at SMPP.

3G scenario.

C. The Migration Overhead in the Scaling Scenarios

We conduct the scaling scenarios by varying the number of concurrent TCP connections from 10 to 1000 at SMPP. Figure 5 shows that the WLAN-3G scenario still gets the longest delay, whereas the WLAN-WLAN has the shortest. There is a minimum migration delay due to the network latency. This delay increases with the traffic loads of SMPP. Therefore, the processing power of SMPP also has an impact on the migration performance. We can adjust it based on the loads of SMPP to guarantee the migration delay to be below a certain number of seconds.

VII. Conclusion

We are entering the post-PC era with the proliferation of various portable devices owned by a user. How to adapt network protocols to such ”single-user, multi-device” scenarios becomes a new challenge. The goal is to allow for users to communicate with others anytime, anywhere, and from any device and reuse existing applications and protocols as much as we can. In this paper, we have described our initial effort along this direction. The main feature of TSMP is to place most new functions at the proxy middlebox, while imposing no changes on both TCP sides of the client and the server. With TSMP, users are able to either save the transmission time of their files or have the convenience of mobility without interrupting ongoing TCP sessions.

References

[1] K. Zickuhr, ”Generations and their gadgets”, Pew Internet & American Life Project, Feb 3, 2011.

[2] R. Pantos, Ed., ”HTTP Live Streaming,” RFC draft-pantos-http-live-streaming-01, 2009.

[3] Adobe Flash Player. http://www.adobe.com/products/flashplayer/. Retrieved: Sep. 2011.

[4] D. A. Maltz et al., ”MSOCKS: An Architecture for Transport Layer Mobility,” IEEE INFOCOM 1998, pp. 1037-1045.

[5] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, ”Improving TCP/IP Performance over Wireless Network,” ACM MOBICOM 1995, pp. 2-11.

[6] A. Fieger and M. Zitterbart, ”Migration Support for Indirect Transport Protocols,” IEEE ICUPC 1997, pp. 898-902.

[7] A. Bakre and B.R. Badrinath, ”I-TCP: Indirect TCP for Mobile Hosts,” IEEE ICDCS 1995, pp. 136-143.

[8] A. Snoeren and H. Balakrishnan, ”An End-to-End Approach to Host Mobility,” ACM MOBICOM 2000, pp. 155-166.

[9] F. Sultan, K. Srinivasan, D. Iyer, and L. Iftode, ”Migratory TCP: Connection Migration for Service Continuity in the Internet,” IEEE ICDCS 2002, pp. 469-470.

[10] X. Qu, J. X. Yu, and R. P. Brent, ”A Mobile TCP Socket,” TR TR-CS-9708, The Austrailian National University, April 1997.

[11] M. Bernaschi et al., ”SockMi: A Solution for Migrating TCP/IP Connections,” PDP 2007, pp. 221-228.

[12] J. Rosenberg et al.,”SIP: Session Initiation Protocol,” RFC 3261, 2002.

[13] C2DM: Google Android Cloud to Device Messaging. http://code.google.com/android/c2dm/. Retrieved: Sep. 2011.

[14] APNS: Apple Push Notification Service. http://developer.apple.com/library/ios/. Retrieved: Sep. 2011.

[15] J. Su, J. Scott, P. Hui, E. Upton, M. H. Lim, C. Diot, J. Crowcroft, A. Goel, and E. de Lara, ”Haggle: Clean-slate Networking for Mobile Devices,” Technical Report, University of Cambridge, Computer Laboratory, Jan. 2007.

[16] P. Shankar, B. Nath, L. Iftode, V. Ananthanarayanan, and L. Han, ”SBone: Personal Device Sharing Using Social Networks,” Technical Report, Rutgers University, Feb. 2010.

[17] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, ”The Design and Implementation of an Intentional Naming System,” SIGOPS Oper. Sys. Rev., 1999, pp. 186-201.

[18] T. Koponen et al., ”A Data-Oriented (and Beyond) Network Architecture,” ACM SIGCOMM 2007, pp. 181-192.

[19] H. Balakrishnan et al., ”A Layered Naming Architecture for the Internet,” ACM SIGCOMM 2004, pp. 343-352.

[20] J. Pan, S. Paul, R. Jain and M. Bowman, ”MILSA: a mobility and multihoming supporting identifier locator split architecture for naming,” IEEE Globecom, December 2008, pp. 2264-2269.

[21] B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Morris, ”Persistent Personal Names for Globally Connected Mobile Devices,” In Proc. of OSDI 2006, pp. 233-248.

[22] R. Moskowitz and P. Nikander, ”Host Identity Protocol (HIP) Architecture,” RFC 4423, 2006.

[23] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, ”SOCKS Protocol Version 5,” RFC-1928, March 1996.

[24] E. Rescorla, ”HTTP Over TLS,” RFC-2818, May 2000.

[25] J.B. Postel, ”Transmission Control Protocol,” RFC-793, September 1981.

[26] Apple Bonjour. http://developer.apple.com/networking/bonjour/. Retrieved: Sep. 2011.

[27] Scaling Twitter: Making Twitter 10000 Percent Faster. http://highscalability.com/scaling-twitter-making-twitter-10000-percent-faster. Retrieved: Sep. 2011.

[28] G. Liao, X. Zhu, and L. Bhuyan, ”A New Server I/O Architecture for High Speed Networks”, IEEE HPCA 2011.