A new Scheme for IP-based Internet-Mobility

Thomas Dreibholz, Andreas Jungmaier
Computer Networking Technology Group
University Duisburg-Essen, Germany
{dreibh,ajung}@exp-math.uni-essen.de

Michael Tüxen
Fachhochschule Münster, Germany
University of Applied Sciences
tuexen@fh-muenster.de

Abstract

In this contribution we present a new type of mobility management for IP-based networks that, contrary to conventional approaches, does not focus on the network layer, but on the transport and session layers.

At the heart of this new mobility concept is the reliable transport protocol SCTP, with an enhancement for dynamic address reconfiguration. This is described in this paper. A session layer based on the reliable server pooling (RSerPool) protocol suite provides for sessionmonitoring and control. The suggested solution is transparent for applications,requires no changes in the network infrastructure, and is evaluated with a real-world implementation.

Finally, we present first results from the application of this mobility concept to different mobility scenarios. These were obtained from working SCTP and RSerPool implementations that have been developed within our group.

1. Introduction

With the integration of data services (e.g. short and multimedia message services, e-mail and web access) into new mobile devices, cellular network providers areimminently required to move away from their traditional provisioning of twonetwork services (voice and IP-based data services) in their core and accessnetworks, into providing IP-based mobile access for customers.

Mobility handling has always been a deficiency of IP-based networks, as these were planned for a fixed, hierarchical infrastructure. Until now, proposed solutions for this problem have mainly been proof-of-concept implementations with experimental character [10, 11]. Moreover, since support for these mechanisms requires architectural support at thenetwork layer, as with Mobile-IP, or substantial changes to established protocolsat the transport layer, acceptance has been low.

In our paper we present a new scheme for mobility handling, that is based mainly on transport layer mobility. This is provided by the reliable Stream Control Transmission Protocol (SCTP, cf. [13, 15]) and a protocol extension for SCTP dynamic address reconfiguration [12].

Transport layer mobility provides persistent connections as long as only one participant of a connection changes its point of attachment to the network (i.e. its network layer address) at a time. Only in the case where both participants of a connection change their addresses simultaneously, the transport layer mobility may fail, and the connection may break. In such a situation, a session layer solution for mobility must arrange handovers which are transparent for applications. The session layer solution must provide for efficient network-wide registration and lookup of peers. It is based on the so-called RSerPool protocol suite defined by the Reliable Server Pooling Working Group of the IETF [16, 14, 18, 2]. In the following section we describe the transport protocol SCTP, together with the extensions that allow use of this protocol for a transparent transport layer mobility (which we call Mobile-SCTP). Section 3 descibes Reliable Server Pooling, while section 4 outlines different mobility concepts, including Mobile-IP and Mobile-SCTP. Section 5 will explain the issues occurring with simultaneous handovers and present solutions for these. Finally, in section 6, we will evaluate the behaviour of the proposed scheme for mobility based on experiments with a real implementation that has been developed by our group. [1] Moreover, we present solutions which show how, in many cases, the protocol behaviour during a handover can be improved substantially.

2. SCTP

The Stream Control Transmission Protocol (SCTP) [13] is a relatively new transport protocol that has been defined by the IETF Signaling Transport working group for the transport of signaling data [8]. It is also a general pur-

[1] Acknowledgement: This work was partially funded by the Bundesministerium für Bildung und Forschung (BMBF) of the Federal Republic of Germany (Förderkennzeichen 01AK045). The authors alone are responsible for the content of the paper.

pose transport protocol which provides a more flexible data delivery than TCP, by separating reliable delivery from reordering through use of the SCTP message stream layer, and an increased fault tolerance by using network level redundancy (support of multi-homed endpoints).

2.1. Protocol Overview

SCTP connections (named *associations* in SCTP parlance) are established after a 4-way handshake between two *SCTP endpoints* (i.e. protocol instances), usually a client and a server. The server, after having received the client's association setup request, returns an acknowledgement of the setup request containing a data structure called *cookie*, which is protocted by a secure message authentication code (MAC), and does not change state.

Only when this cookie is returned by the client unchanged does the server allocate resources and establish a new association. This ensures that a server will not blindly commit resources on invalid connection setup requests.

The association setup requests contain lists of valid client/server transport addresses (i.e. combinations of a number of IP addresses and one port number). The set of all possible combinations of the server's IP addresses and the server port, together with the client's IP addresses and the client port gives all possible identifiers of one association. Thus, SCTP explicitly supports multi-homed endpoints, i.e. endpoints which support more than one IP address (be it IPv4 or IPv6).

SCTP is a message-oriented, reliable transport protocol. Unlike TCP, the SCTP send- and receive-primitives preserve message boundaries. The protocol may multiplex several short messages into one SCTP packet (which may subsequently be transmitted as the payload of one IP packet). By using MTU discovery, SCTP avoids fragmentation at the IP layer.

2.2. SCTP Packet Format

SCTP packets (cf. Figure 1) consist of a common header, followed by a variable number of *chunks*, of which there are two types: control and data chunks. The common header contains source and destination port numbers (similar to TCP and UDP), as well as a 32 bit value named *tag* and a 32 bit CRC32C checksum [15]. The tag is a randomly chosen value exchanged with the peer endpoint at association startup, which protects associations from attacks, when attackers try to blindly insert forged SCTP packets into an existing association.

Upon receiving an SCTP packet, the checksum is verified, and the packet discarded if the checksum is invalid. Then endpoints use the source and destination IP addresses and ports to identify to which association the packet belongs. Now, if the tag does not match the initially received value, the packet is generally discarded.

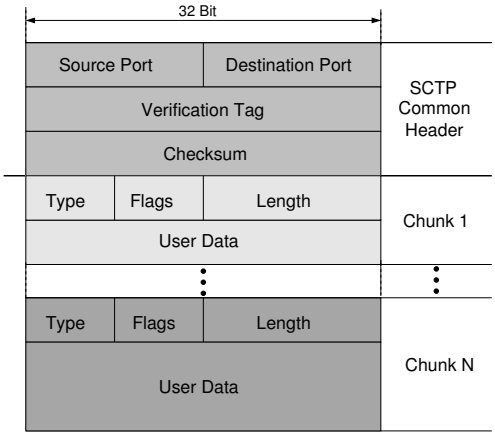

Figure 1. SCTP packet format with common header and chunks

The chunks contain either the actual user data (data chunks) or control data (control chunks) used by the association for control purposes, acknowledgements, monitoring of peer reachability with heartbeats, termination and setup of associations, error messages and optionally, protocol extensions.

2.3. Message Streams

SCTP provides its user with a flexible method of data delivery by separating the reliable transfer of messages between endpoints (ensured by proper use of 32 bit transmission sequence numbers (TSN), acknowledgements and retransmission timers) from the actual delivery to the user process.

This is achieved at the cost of introducing an internal multiplexing mechanism, called *streams*, with 16 bit stream identifiers and 16 bit stream sequence numbers.

When establishing an association, two endpoints negotiate the number of incoming and outgoing streams they are willing to support. There may be a different number of outgoing and incoming streams. Therefore, SCTP streams are effectively unidirectional channels, within which messages are usually transported in sequence, unless the user requests a message to be delivered by an unordered service. These unordered messages may bypass any other messages during the delivery process [13].

The stream mechanism may reduce the effects of head-of-line blocking, especially in the case of a large number of small messages and a small number of streams, since the reordering mechanism of one stream is not affected by that

of another stream waiting for retransmission of a message that was lost.

2.4. Multi-Homing

The SCTP supports multi-homed endpoints with more than one IP address. For an endpoint, the definiton of a *path* is one destination address of its peer endpoint. This assumes that an endpoint does not use source routing, and may not influence its routing table by any other means. Clearly, each multi-homed endpoint may reach its peer by a number of different paths. The endpoint computes an own set of congestion parameters unique for each destination path of an association, and thus guarantees a behaviour compatible with existing transport protocols (especially TCP, cf. [6]).

Generally, multi-homing does not guarantee the reachability of an endpoint nor does it guarantee a higher availabilty in case of network failures. That is due to the complex effects of multi-homing and routing (consider two dual-homed endpoints that have two interfaces each, but are residing in the same IP subnet; they would use only one - the default - interface unless the routing table is changed after one interface or path fails). In a carefully engineered network setup however, or in an Internet setting with connectivity to more than one internet service provider (ISP), a protocol supporting multi-homing may greatly improve network level fault tolerance.

2.4.1 SCTP Address Management

Upon intialization the client sends its association setup request, containing a set of its source addresses to one of the known addresses of the server. The server conveys in turn a set of its addresses to the client in the acknowledgement of the association setup request. Once the association is fully established, both can elect one of the peer addresses as the *primary path*, which should subsequently receive the main traffic load. Additionally, the user may explicitly request use of a path different from the *primary*.

2.4.2 Path Monitoring

By default, SCTP endpoints monitor the reachability of their peers by regularly sending heartbeat control chunks to all idle destination addresses of the peer endpoint. Upon reception of a heartbeat control chunk, an endpoint replies with a heartbeat acknowledgment control chunk.

The endpoint will keep an error counter for each path that is incremented, should the endpoint not receive an acknowledgement within a certain time. If the error counter exceeds a protocol variable, the state of the considered path will be set to *unreachable*. The endpoint will then continue to send heartbeats to this address, allowing the reinstatement of the path status to *reachable* at a later stage.

Since endpoints should send their acknowledgements of data and heartbeat control chunks back to the originating peer destination address, paths that are actively used for data transmission need not be monitored by heartbeat chunks.

2.4.3 Path Selection

The primary path carries the main load of user data. Thus growth of the congestion window usually only occurs for one path, a desirable state for general internet deployment, because of fairness to TCP. Other paths are only used for data retransmissions and heartbeat control chunks.

Sending retransmissions on otherwise idle, uncongested paths will have an advantageous effect on recovery from packet loss, if it is due to link congestion. In such a situation, SCTP may perform much better than protocols like TCP that do not support multi-homing [5].

Should the primary path become unreachable, an endpoint may send data to another, active address and report the failure to its user which can subsequently choose a new primary path. Furthermore, the SCTP user can – at any time – request transmission to a destination address that is different from the primary path. Should that address be unreachable, the protocol may choose another, active path.

2.5. Endpoint Failure Management

An SCTP endpoint keeps track of the number of consecutive retransmissions of data or heartbeat chunks sent to the peer endpoint (as opposed to one path). Each time a chunk is acknowledged, the counter is cleared. Once the counter exceeds the association error limit, the peer endpoint is considered unreachable, and the association is closed.

2.6. Dynamic Address Reconfiguration

SCTP is extensible through the use of new control chunks (cf. section 2.2). A proposed SCTP extension may be used to dynamically add or remove addresses from an ongoing transport layer association [12]. Moreover, this extension can be used to signal to the peer endpoint which IP address is preferrable as the primary address. Thus, in a handover situation where connectivity to two networks may be given, a mobile device can signal to its peer which network is the preferred destination to send data to and thus improve data throughput.

This is achieved by the use of Address Configuration Change (ASCONF) control chunks, which contain a variable number of request parameters for the peer. These either signal:

- addition of address requests,
- removal of address requests, or

- set primary address requests.

This mechanism can also be used in cases where the originating source IP address of the ASCONF request does not match any known SCTP association (when addresses have changed before this could be signaled to the peer endpoint), since the ASCONF contains an additional address parameter that must have been known to belong to the concerned association beforehand.

3. RSerPool

While the SCTP protocol provides improved network level fault tolerance, it does not improve a node's reliability. If a node fails, the service it provides is interrupted. An obvious solution to cope with this problem is to have redundant nodes, called server pools. If one server fails, its clients can arrange an application-layer failover (see [1] for more information) to another serverwhich continues the service. To create an open standard for serverpooling at the session layer, the IETF Reliable Server Pooling (RSerPool)working group has been founded.

The RSerPool protocol suite focuses on providing server redundancy using server pools. In combination with the network fault tolerant transport protocol SCTP, it is possible to build systems without single points of failure. The RSerPool architecture uses three classes of elements:

Pool Elements (PEs) These are a pool of servers, all providing the same service within the pool.

Pool Users (PUs) These are clients being served by one Pool Element.

Name Servers (NSs) These nodes provide a translation service and supervise the PEs.

An example RSerPool scenario is shown in Figure 2. A pool (a set of servers providing the same service) is identified by a pool handle, i.e. a byte vector of arbitrary length (e.g. an ASCII string "Video Pool"). If a server wants to become a PE for a specific pool, it registers itself at a name server with the corresponding pool handle. The name server which does the registration is called the PE's home-NS. This home-NS also monitors the PE's availability using session layer keep-alives. If the PE does not answer these keep-alives or PUs report that the PE is not reachable, the NS may decide to remove it from the namespace. If the home-NS itself becomes unavailable, the PE simply selects another one as home-NS and executes a reregistration.

The protocol used between the PEs and the NSs is called the Aggregate Server Accesss Protocol (ASAP), currently being defined in [14]. It provides registration, reregistration and deregistration of PEs, supervision of PEs using session

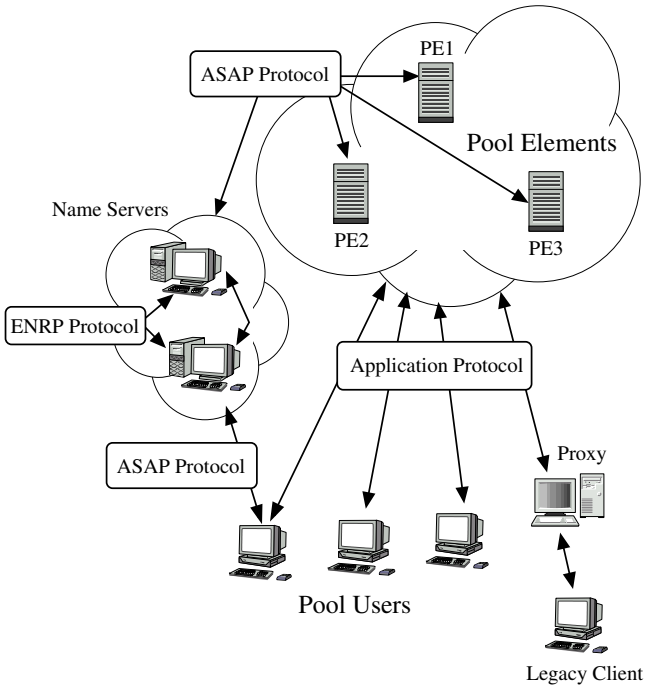

Figure 2. RSerPool Example Scenario

layer keep-alive messages and detection of NSs using server announcements via multicast (if available).

Pool handles are only valid within an operational scope. All NSs within an operational scope have information about all PEs within this operational scope. This means that the namespace used by RSerPool is flat, as opposed to, e.g. DNS. The protocol used by NSs within an operational scope to synchronize their name spaces is called the Endpoint Name Resolution Protocol (ENRP) [18].

Like a PE, the PUs can also listen to server announcements via ASAP to detect usable NSs. If a client wants to be served by a PE belonging to a certain pool, it sends a name resolution request containing the corresponding pool handle to one of the NSs. The NS will respond with a subset of all transport addresses which can be used to access (different) PEs. This communication also uses ASAP.

The selection of the final PE is realized in two steps: in the first step, the NS can select a subset of all known PEs and their transport addresses in the pool. This selection is based on the pool policy. Examples for pool policies are "round robin"or "least used". In the second step, the PU has to select one of the PEs in the given subset. This is also based on the pool policy.

Whenever a PU detects that a PE cannot be reached, one of the NSs is informed. This information, combined with the ongoing supervision by keep-alive messages, is used to remove PEs from the pool if they are out of service. In

contrast to the Domain Name System (DNS), RSerPool:

1. uses a flat namespace,
2. allows arbitrary pool handles,
3. provides a high probability that only PEs are announced which are in service, and
4. allows dynamic registration and deregistration of PEs.

In case of a failure of a PE, the PU can fail-over to a different PE of the same pool. The grade of intervention of the RSerPool user depends on the required service. If a message loss during fail-over is not acceptable, the upper layer has to use application-level acknowledgements and its own buffering. Otherwise, no intervention is necessary if the upper layer allows for some messages to be dropped during failover and always uses pool handles for sending messages.

4. Mobility Concepts

4.1. Mobile-IP and Mobile-IPv6

Mobile-IP [10] and Mobile-IPv6 [4] require initial registration of nodes with entities named Home-Agents (HAs). When nodes are located in their initial home networks, they are reachable by their corresponding home address(es).

In the case of Mobile-IP, there must be a corresponding agent entity in the foreign network, named Foreign Agent (FA), that accepts registrations from nodes that have changed their point of attachment and are located in the foreign network. A tunnel is established between the HA and the FA, and all IP-packets directed to the home address of the node are tunneled via the HA to the FA that in turn forwards them directly to the node.

In the case of Mobile-IPv6, the FA is not needed anymore, as a node can directly tell its peer and its HA to set a source route via its current gateway router. In that case, packets to the home address are still tunneled by the HA via the current gateway router in the foreign network, all other packets may be directly sent to the current gateway router that forwards them to the mobile node.

Essentially, with Mobile-IP or Mobile-IPv6, mobility is becoming transparent to the transport layer and all higher layers. A node is always reachable by at least its home address. This comes at the cost of triangular routing (between peer, HA, and mobile node in a foreign network) in the case of Mobile-IP. Mobile-IPv6 takes a better approach, but deployment of this protocol is still in very early stages.

4.2. Mobile-SCTP

The SCTP, together with the extension described in section 2.6, supports persistent transport layer connections with mobile, IP-based endpoints. These need to support IP multi-homing, and dynamic IP address assignment (e.g. by DHCP [3]). A mobile client can then communicate with a non-mobile server as shown in Figure 3.

A mobile node in Area 1 is reachable by one address, say A1. When the mobile node moves into area 2 which is covered by two access points, its network adapter discovers the new physical access, and configures a new address, say A2, and route. This is conveyed to the peer of the mobile node by means of an ASCONF add address request (make-before-break). The node is then reachable by the destination transport addresses A1 and A2.

When it moves on to area 3, it loses connectivity to the first base station, and shuts down the interface belonging to A1. This is, again, conveyed to the peer by sending an ASCONF delete address request. The peer then knows that the mobile node is only reachable by the transport destination address A2. This mechanism also works in situations where the node cannot send the ASCONF add address request from the already known address/interface combination A1, as specified in section 2.6, so a break-before-make is also possible.

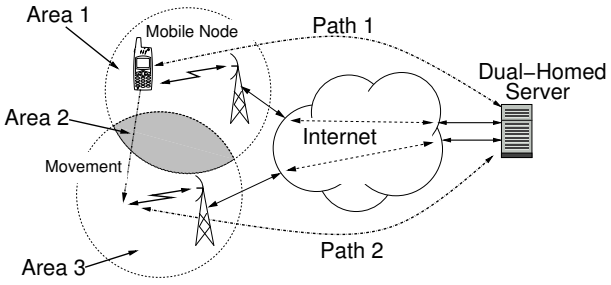

Figure 3. Node Mobility with Mobile-SCTP

5. Simultaneous Handovers

Having two communicating nodes, only one of which is mobile, Mobile-SCTP is always able to keep the association established when the mobile node moves. Even if it temporarily loses the connection and hands over to a new network, it is able to send its new address(es) to the non-mobile node. But if both nodes are mobile, the situation may occur that both nodes move to new networks simultaneously, and therefore change their addresses at the same time. Then each node is unable to inform the other about its address change, since all known addresses of the peer

node have become invalid. Therefore, the SCTP association breaks. To cope with this challenge, solutions described in this sections are possible.

5.1. SCTP and Mobile-IP

Using mobile IP, each (mobile) peer is always reachable via its home address. Packets sent to this address are tunneled via HA and FA to the mobile device. In cases where both clients move simultaneously, the respective ASCONF requests should be sent to a peer's home address, and will thus reach the peer in any case. All user data can then be sent on a direct path to the last known peer transport address (in a foreign network). To enforce a sensible policy for this, the initial association setup should always be done using the Mobile IP home address, and the current foreign address should then be added to the association. *Any* new ASCONF request should then be sent to the home address.

5.2. Dynamic DNS

Another possiblity to deal with simultaneous mobility of peers are dynamic DNS updates (see [17]). When the addresses of the mobile devices change, their DNS entries must be updated. Subsequent to the address changes, the transport connection will fail, and needs to be newly established. For this, the SCTP peers obtain the new address by querying their DNS. Then the application needs to restart the SCTP association. Should the DNS registration of the peer not have succeeded (as it is based on UDP), the new setup of the SCTP connection will fail.

The general problem of DNS is the fact that it has been designed under the assumption that its entries change infrequently. Using dynamic DNS to support mobile devices, the lifetimes have to be set sufficiently low (e.g. at most a fewseconds) to cope with frequent changes. This makes caching useless and therefore leads to increased network overhead for name resolution queries.

5.3. Mobile-SCTP and RSerPool

RSerPool provides a simple and efficient framework to solve the simultaneous mobility problem: between the transport layer (Mobile SCTP) and the application layer, a session layer is inserted (see Figure 4). When the transport connection breaks due to simultaneous mobility, the session layer ensures establishment of a new transport association and triggers an application-specific failover procedure.

In the RSerPool framework, (at least) one node registers as a pool element of a server pool having a unique handle. An example is shown in Figure 5. The called video phone system registers under a unique pool handle. This handle may be compared to a telephone number. Now, the caller

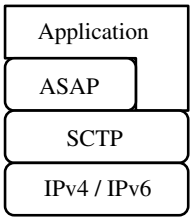

Figure 4. Protocol Stack for Mobile-SCTP with RSerPool

can establish the association by first doing a RSerPool name resolution with the help of a NS and then connecting to the resolved transport address of the called node.

As long as a handover to a new network is only made by one node at a time, the SCTP protocol in conjunction with the Add-IP extension is always able to cope with this change and update the transport addresses of the association transparently for the application. But in the case of simultaneous mobility, the association breaks. Now, the called node reregisters with its new transport address but under the same previous pool handle. The ENRP protocol ensures that all NSs of the operational scope get the updated pool data. Therefore, the NS functionality can be compared to a location register in mobile networks. Using an appropriate pool policy (e.g. "the newest element"), the caller is now able to let the RSerPool name server resolve the pool handle to the new transport address. Then, it can establish a new association and execute an application-specific failover procedure. After that, the application can continue the communication session.

6. Experimental Results

6.1. Setup

We investigate the handover behaviour in a lab environment where two user applications run as PU and PE on two dual-homed hosts. A RSerPool NS runs on a third host which also acts as router and network emulator (see Figure 6). Note that the hosts were not actually mobile, but their address configuration changed as is expected from moving mobile devices.

The applications send bidirectional, CBR (constant bit rate) traffic between the PE and PU, at a rate of 64 Kbit/s. For some experiments (see e.g. section 6.3 below), a higher rate of approximately 1 MBits/s was chosen. Delays were configured so that the packets that were sent between addresses A1 and B1, A2 and B2, respectively, were delayed by 50, 100 or 200 ms. These settings reflect values that may

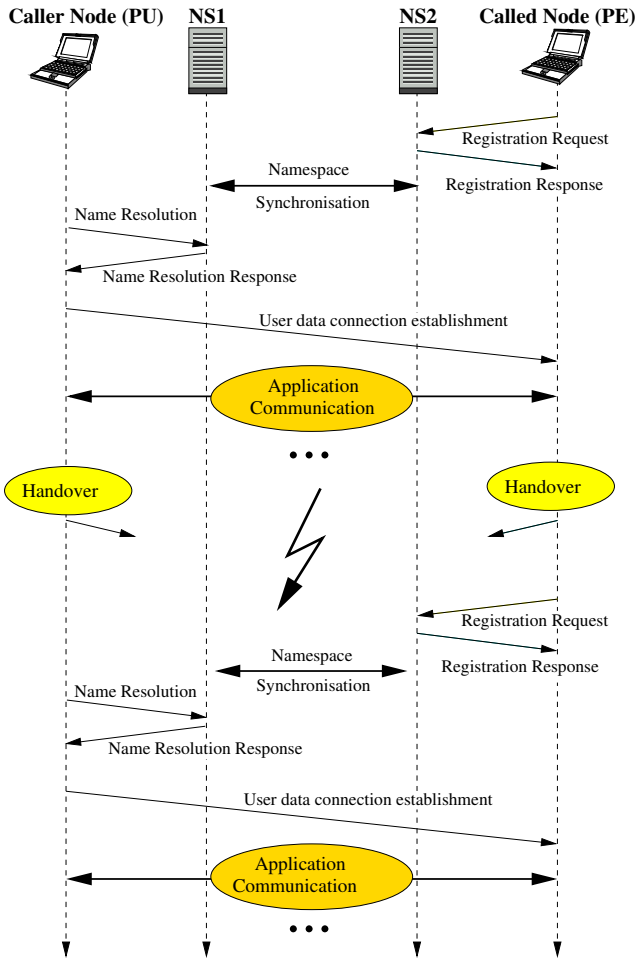

Figure 5. Mobility Example Scenario

be reached by current wireless technologies such as WaveLan or UMTS [7].
With the given elements, the following scenarios are possible:

1. Mobile PU, performing a handover, and a fixed PE:
   (a) make-before-break;
   (b) break-before-make.
2. Fixed PU and a mobile PE, performing a handover:
   (a) make-before-break;
   (b) break-before-make.
3. Mobile PU and a mobile PE, both performing a handover at the same time: (simultaneous handover)
   (a) make-before-break;
   (b) break-before-make.

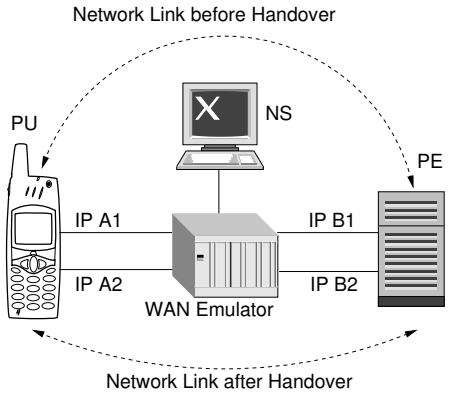

Figure 6. Experimental network setup

Scenarios, 1 and 2 are similar, except for the fact that a mobile PE will reregister with the NS after changing its addresses whereas a mobile PU will not. However, this does not affect the ongoing transport layer connection between the PE and PU. In fact, the registration is necessary only because it is required by the current ASAP draft [14]. The NS, which is also engaged in a Mobile SCTP connection with the mobile PE, is aware of any PE address changes instantaneously. However, for security reasons, it only updates its database entry after the PE has done a proper reregistration.

We will further investigate only 1(b), 3(a) and 3(b), since all interesting situations are covered in these scenarios. The presented results are based on prototype SCTP and Reliable Server Pooling implementations that were created in cooperation between the University of Duisburg-Essen and Siemens AG, ICN, Munich, and are freely available under GNU Public Licenses from `http://www.sctp.de`. For further explanation of the RSerPool implementation see [2].

6.2. Normal Handover: Break-before-Make

A first set of experiments was conducted in a setting where the mobile PU loses its initial network connectivity, and cannot reach the fixed peer PE for some time (call this time T_{break}). Referring to Figure 3, this reflects the movement of a mobile client from Area 1 outside of Areas 1,2 or 3. After T_{break}, the mobile PU enters Area 3, and continues the ongoing connection. At this time the mobile PU receives a new address, that is unknown to its peer PE. It sends an SCTP address add request (cf. section 2.6), and continues the ongoing connection.

The data that is sent towards the PU over the deactivated initial path needs to be retransmitted over the newly added SCTP path. After this, the normal connection is resumed using the new path. Figure 7 presents the data rate that is received by the PU during the time of handover, for different values of T_{break}. The data that was queued when the initial

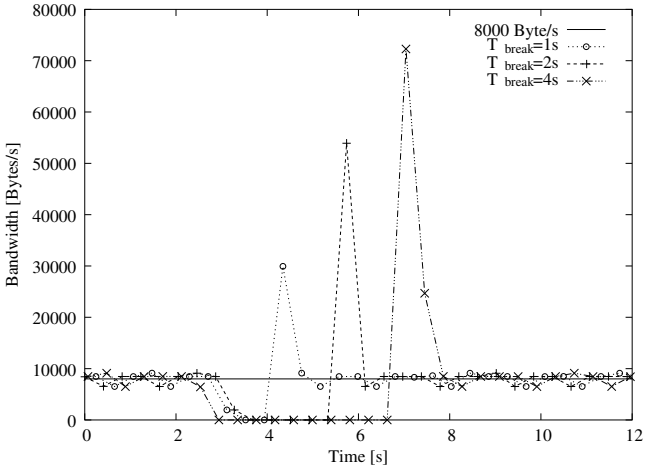

Figure 7. Bandwidth received by the PU (Rate 64 KBit/s, RTT=50 ms)

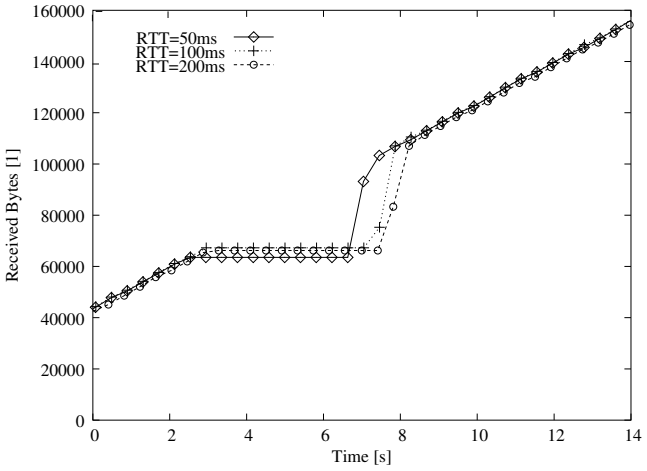

Figure 8. Bytes received during break-before-make handover (Rate 64 KBit/s, T_{break}=4s)

path went down is being retransmitted in a burst, after the new path becomes available, and therefore, the bandwidth used shows a short spike after the handover. All in all, the normal transmission is quickly resumed after network connectivity is regained.

Figure 8 displays the effect of differing round trip times on the continuation of a connection. It shows the number of bytes that have been received at a certain time. Not surprinsigly, for shorter RTTs, the SCTP add address request reaches the peer PE faster, and thus the transmission rate is back to normal more quickly.

6.3. Simultaneous Handovers: Make-before-Break

The next set of experiments was conducted in a setting where both the PU and PE gain connectivity to a new network before the connectivity to the old network is lost after some time T_{over}. This reflects a mobile PU that moves from Area 1 via Area 2 to Area 3 in Figure 3, and a mobile PE with a similar behavior in its own access network.

The ongoing connection can be maintained seamlessly over the new network. For low bandwidth traffic, the result looks similar to Figure 9, i.e. the connection is not affected at all by the handover.

The case of a higher bandwidth connection gives some more interesting insights: when a new address is added to a connection, it is made new primary path, since the PU sends an SCTP set remote primary request along with the add address request. At that time, the old path has a large congestion window, whereas the new path has a small congestion window (i.e. twice the size of the path MTU). Therefore the data rate over the new path is initially significantly lower than that from the old path. The received bandwidth drops and recovers only after a while. For the recovery process the bandwidth used exceeds the normal rate because the CBR data queues up when the congestion window has not fully openend up. This effect prolongs the duration of handovers when such a high bandwidth is not available.

6.4. Simultaneous Handovers: Break-before-Make

Finally, we investigated the scenario where a PU and PE change their point of attachment to the network at the same time, and the transport connection breaks. When SCTP notifies the RSerPool layer of the failure, this layer triggers an application specific failover procedure. This procedure consists of a new lookup of the PE at the NS, and establishment of a new SCTP connection. The test program we used did not retrieve previously unsent data from the transport layer (this would be possible), in order to send this data over the new connection. It simply continued to send new data, as this is the expected behavior in the case of voice data transmission. Figure 11 shows the bandwidth obtained by traffic towards the PU.

The length of the period where no data is received does not depend on the length of time when the client has no network connection, but on the time the underlying SCTP transport takes to detect that the connection has failed. For a better behaviour in this case, the time for recognizing the SCTP association failure must be minimized, which can be achieved by setting the error thresholds to lower than the values recommended in [13].

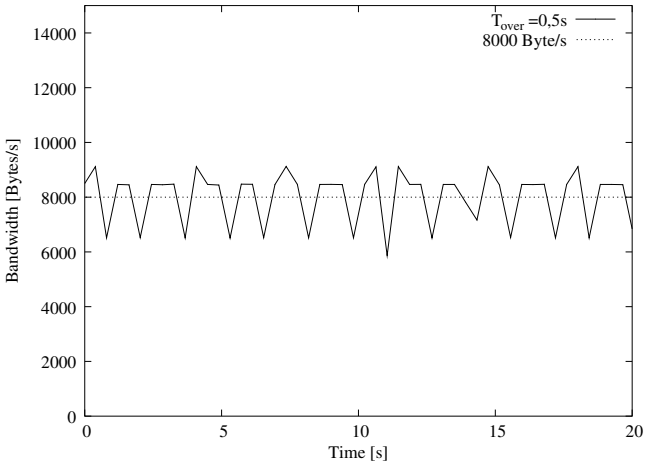

Figure 9. Bandwidth received during make-before-break handover (Rate 64 KBit/s, T_{over}=0.5s, RTT=50ms)

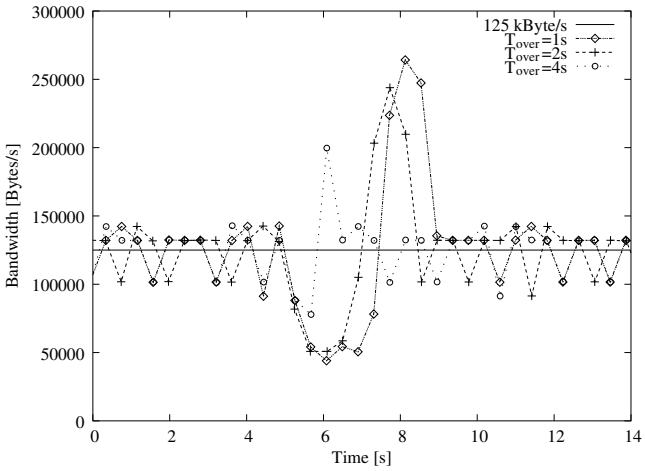

Figure 10. Bandwidth received during make-before-break handover (Rate 1 MBit/s, RTT=50ms)

6.5. Remarks

Due to time constraints, only basic parameter settings have been evaluated, and the results presented in this section only represent a snapshot of ongoing work. We will prepare a more detailed review of these results along with proper statistical analysis of their dependability in the final version of this paper.

7. Conclusions and Further Work

In this paper, a new scheme for mobility management is presented that relies on the transport protocol SCTP with the extension for dynamic address reconfiguration, and the reliable server pooling protocol suite. We evaluated the usability of these protocols in an experimental setup and presented promising first results. Mobile-SCTP already provides a working solution for a wide number of mobility setups (e.g. fixed streaming server and mobile clients).Together with the reliable server pooling protocol suite, it may serve as anIP based solution for all mobility scenarios. Contrary to mobility solutions at the network layer, such as Mobile-IP, the presented solution requiresabsolutely no changes of the network layer.

Optimal handling of break-before-make situations is essential for protocols that are to cope with supporting mobility of users. The solution we presented here is able to cope with this special problem, albeit more optimal settings for fast peerfailure recognition will need to be investigated. Our open source implementations of the described protocols will serve as a basis for continuing these investigations. To systematically investigate a larger set of parameters, simulation studies of SCTP and Mobile-SCTP are being prepared using OPNET simulation tools [9]. This will allow a more detailed analysis of optimal parameters for the failure discoveryin diverse network and mobile environments.

References

[1] T. Dreibholz. An efficient approach for state sharing in server pools. *Proceedings of the 27th Local Computer Networks Conference (LCN 2002)*, November 2002.

[2] T. Dreibolz and M. Tüxen. High availability using reliable server pooling. *Linux Conference Australia 2003*, January 2003.

[3] R. Droms. *Dynamic Host Configuration Protocol*, March 1997. RFC 2131.

[4] D. Johnson, C. Perkins, and J. Arkko. *Mobility Support in IPv6*. IETF, Network Working Group, February 2002. draft-ietf-mobileip-ipv6-21.txt, work in progress.

[5] A. Jungmaier, E. P. Rathgeb, M. Schopp, and M. Tüxen. Sctp - a multi-link end-to-end protocol for ip-based networks. *AEÜ International Journal of Electronics and Communication*, 55 (2001)(1):46–54, September 2000.

[6] A. Jungmaier, M. Schopp, and M. Tüxen. Performance evaluation of the stream control transmission protocol. *Proceedings of the Joint IEEE ATM Workshop 2000 (ATM 2000 Conference)*, pages 141–148, June 2000.

[7] A. Mercier, P. Minet, L. George, and G. Mercier. Adequacy between multimedia application requirements and wireless protocols features. *IEEE Wireless Communications*, pages 26–34, December 2002.

[8] L. Ong, I. Rytina, M. Garcia, H. Schwarzbauer, L. Coene, H. Lin, I. Juhasz, M. Holdrege, and C. Sharp. *Framework*

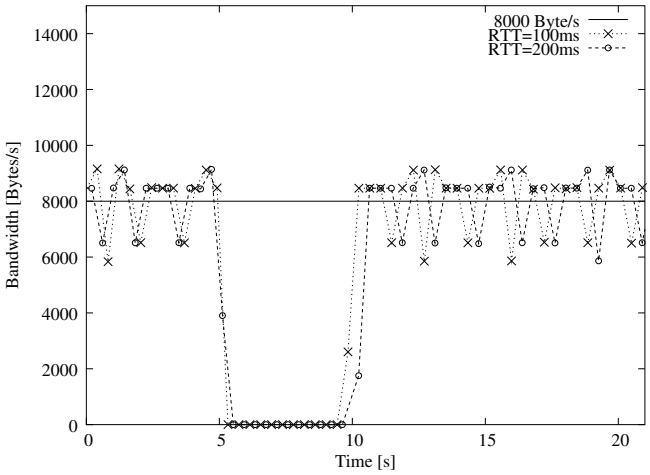

Figure 11. Recovery after RSerPool application level failover

Architecture for Signaling Transport, October 1999. RFC 2719.

[9] OPNET-Technologies. *The OPNET Modeler*, 2003. http://www.opnet.com/products/modeler/home.html.

[10] C. Perkins and Ed. *IP Mobility Support for IPv4*, August 2002. RFC 3344.

[11] A. C. Snoeren and H. Balakrishnan. *TCP Connection Migration*. Network and Mobile Systems Group at the MIT Laboratory for Computer Science, November 2000. http://www.sds.lcs.mit.edu/papers/draft-snoeren-tcp-migrate-00.txt.

[12] R. Stewart, M. Ramalho, Q. Xie, M. Tüxen, I. Rytina, M. Belinchon, and P. Conrad. *Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration*. IETF, Transport Area Working Group, February 2003. draft-ietf-tsvwg-addip-sctp-07.txt, work in progress.

[13] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. *Stream Control Transmission Protocol*, October 2000. RFC 2960.

[14] R. Stewart, Q. Xie, M. Stillman, and M. Tüxen. *Aggregate Server Access Protcol (ASAP)*. IETF, Reliable Server Pooling Working Group, February 2002. draft-ietf-rserpool-asap-06.txt, work in progress.

[15] J. Stone, R. Stewart, and D. Otis. *Stream Control Transmission Protocol (SCTP) Checksum Change*, September 2002. RFC 3309.

[16] M. Tüxen, Q. Xie, R. Stewart, M. Shore, L. Ong, J. Loughney, and M. Stillman. *Requirements for Reliable Server Pooling*, January 2002. RFC 3237.

[17] P. Vixie, Ed., S. Thomson, Y. Rekhter, and J. Bound. *Dynamic Updates in the Domain Name System (DNS UPDATE)*, Apr. 1997. RFC 2136.

[18] Q. Xie, R. Stewart, and M. Stillman. *Endpoint Name Resolution Protcol (ENRP)*. IETF, Reliable Server Pooling Working Group, March 2003. draft-ietf-rserpool-enrp-05.txt, work in progress.