Mobility Support for Next-Generation Internet Signaling Protocols

Xiaoming Fu
University of Goettingen
Institute for Informatics
Lotzestrasse 16-18
37083 Goettingen, Germany
Email: fu@cs.uni-goettingen.de

Henning Schulzrinne
Columbia University
Dept. of Computer Science
1214 Amsterdam Avenue, M.C. 0401
New York 10027, USA
Email: hgs@cs.columbia.edu

Hannes Tschofenig
Siemens AG
Corporation Technology
CT IC 3 Otto-Hahn-Ring 6
81739 Munich, Germany
Email: hannes.tschofenig@siemens.com

***Abstract*— Internet signaling protocols establish, maintain and remove state along the data path. Next-generation signaling protocols design must meet the scaling requirements imposed by the various tasks of the Internet signaling applications, such as resource reservation and middlebox configuration, and to meet the demand for general functionality in signaling protocols, including strong security, reliability, congestion control, support for various signaling purposes and message sizes, and efficient support for mobility. This paper presents a generic signaling architecture, the Cross-Application Signaling Protocol (CASP) and describes how it supports efficient and secure signaling in IP mobility scenarios. In this approach, the signaling functionality is splitted into two layers: a generic messaging layer which provides the generic functionality for message delivery, and a client layer consisting of a next-hop discovery client and any number of client protocols which perform the actual signaling tasks. The essential mechanisms required to support mobility are: (1) a session identifier uniquely selected by the initiator and effective discovery of the cross-over node; (2) a branch identifier incrementally assigned for the new branch and efficient release of state in the abandoned branch; (3) ensuring discovery messages are delivered exactly following the path that mobile IP packets are encapsulated; (4) effective hop-by-hop authentication and re-authorization provided by the messaging layer, non hop-by-hop security for signaling clients and denial-of-service protection in the discovery client.**

I. Introduction

Internet signaling protocols establish, modify and remove state along the data path and thus form the foundation for a number of diverse network tasks, such as reserving resources, configuring middleboxes such as NATs and firewalls, diagnosing network behavior and depositing active network code in routers. The path of the signaling message can traverse a subset of the routers seen by the data packets, or can be more loosely aligned with the data flow.

The Resource ReSerVation Protocol (RSVP) [1], [2], [3] has been designed to support QoS resource reservation. It introduced a number of features for Internet signaling, such as soft-state, two-pass signaling message exchange, separating signaling from underlying routing protocols, and modularity through the use of opaque objects. RSVP tightly couples application semantics such as resource reservation to the delivery of signaling messages. Since it was optimized for multicast resource reservation as part of the IntServ framework, including receiver diversity, its implementation complexity is significant. RSVP was also designed when IP mobility was in its infancy and network address translations (NATs) were uncommon. Initially, RSVP only supported reliability by retransmission of messages after each timeout interval; later, a simple retransmission mechanism was added [3].

Particularly, since RSVP identifies signaling sessions by IP addresses, it becomes difficult to address host mobility in IP networks [4], [5]. When hosts move, the state established along the previous route remains until it times out after multiples of the soft-state interval, typically after more than a minute. With even modest mobility, large amounts of "abandoned" states may cause inefficient resource utilization. In addition, IP-in-IP encapsulation (tunnel) in mobile IP causes RSVP messages sent to a mobile host also encapsulated as messages with a new protocol number and a new (source or destination) address in its outer header, thus concealed to the RSVP nodes along the path and unable to perform expected signal tasks.

Furthermore, signaling protocols offer opportunities for attackers to perform replay attacks, denial of service attacks, eavesdropping or traffic analysis. Thus, authentication and authorization are particularly important. Here, RSVP offers only rudimentary services that largely rely on shared secrets.

A number of efforts on mobility in Internet signaling (e.g., [6], [7]) have been made recently. However, most of them were only meant for QoS signaling, and few were designed to be able to meet other requirements for next-generation signaling protocols.

To address these challenges, we present a new generic signaling architecture, the Cross-Application Signaling Protocol (CASP) [8]. Following the two-level RSVP model proposed by Braden and Lindell [9], which provides extensibility for signaling client protocols by separating them from the generic signaling protocol, the CASP architecture further addresses issues of mobility support, security and congestion-controlled delivery. Under our design, we extend the RSVP model by allowing reliable transport mechanisms (such as TCP and SCTP), separating discovery from signaling message delivery, and supporting host mobility intrinsically, without introducing RSVP-in-RSVP tunnels [10].

The rest of the paper is organized as follows. The CASP

architecture is presented in Section II. Mobility support in the CASP architecture is described in Section III. Related security considerations are presented in Section IV.

II. CASP – A Generic Internet Signaling Architecture

A. Overview

As shown in Fig. 1, the CASP architecture consists of a generic *messaging* layer, which transports signaling messages between the initiator of the signaling session and the responder and a *client* layer, which consists of a *next-hop discovery* client and any number of specific signaling client protocols. The client protocols perform the actual signaling operations, such as QoS resource reservation, NAT/firewall configuration, or network diagnosis, where client data are carried in opaque objects. Typically, the initiator is the data sender and the responder is the data receiver, but CASP supports both sender-initiated actions, such as reserving resources, as well as receiver-initiated ones.

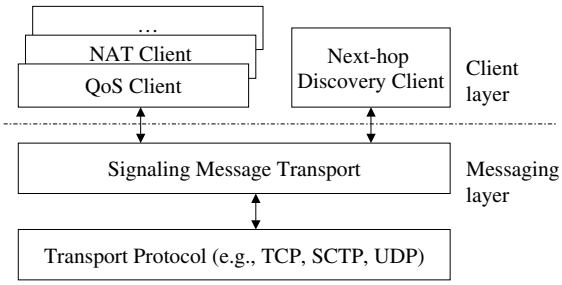

Fig. 1. CASP – a generic Internet signaling architecture

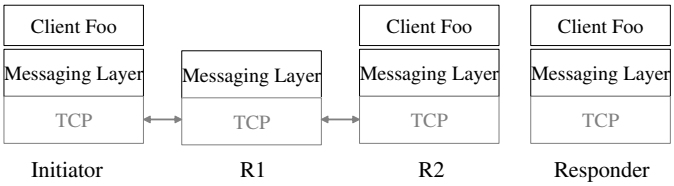

Fig. 2. CASP signaling example

While delivering generic messaging layer signaling messages, the messaging layer establishes, refreshes or releases *states* for signaling *sessions*; it also remembers the path traversed by installing state at individual routers (stateful approach) or records a route (stateless approach). A session is identified by the initiator with a cryptographically random session identifier. Additionally, a *flow identifier* describes the data flow the signaling message pertains to.

Fig. 2 illustrates an example of CASP signaling where TCP is used as the underlying transport mechanism. A signaling client ("*foo*") requests the CASP messaging layer for delivery its service from the initiator along the path to the responder, whereas it is possible that some intermediate CASP nodes (in this example, R1) does not support the requested client layer. Then the following operations take place in order:

1) The initiator creates a messaging layer session identifier, and determines that the next CASP node is R1 and there is an existing TCP connection between the initiator and R1. It then generates a CASP message with supplied signaling client payload and delivers it to R1.
2) Upon receipt of the CASP message sent by the initiator, R1 determines whether it supports the requested client type, then simply performs the similar procedure as in the initiator; additionally, it also remembers the previous hop. R2 differs from R1 in that it passes its client payload on to the correspondent client protocol. After that, it establishes a TCP connection between R2 and the responder, as there is no previous TCP connection.
3) In the responder, after receiving the client data, the client layer may decide to send a response message to the initiator, following the reverse chain of CASP nodes.

B. Signaling Message Delivery

The CASP messaging layer is built on existing reliable or unreliable transport protocols, such as TCP, SCTP or UDP, depending on the needs of the application. Small, "one-shot" signaling messages can be embedded into the UDP or raw-IP discovery message for efficiency, while larger messages and reliable responses then make use of a chain of reliable transport connections (TCP, SCTP). Naturally, the end-to-end transport behavior may be determined by the weakest link. In many cases, signaling peer nodes will communicate with each other repeatedly and thus maintain long-lasting connections, avoiding the connection set up latency. As a result, messaging layer session setup latency is, on average, low.

Modern reliable transport protocol offer flow control, congestion control, message fragmentation and fast loss recovery, which are important characteristics for a generic signaling protocol. For example, public-key-based session setup messages or active network code may well be large. Such large messages may need fragmentation and congestion control which are their functionalities of TCP and SCTP, but not of unreliable transport protocols like UDP. The use of a reliable transport also makes it possible to use TLS [11] for channel confidentiality and integrity.

A CASP messaging layer session is established between an initiator and a responder, along a chain of CASP nodes, with a session identifier chosen by the initiator. At each node, the CASP messaging layer remembers its previous next CASP node, if not being the initiator; it also determines the next node along the data path, checks if there is an existing transport connection to that node, or establishes one if not, and then forwards the message downstream. The node then remembers the upstream node and associates it with the session identifier, a state refresh timer and a state expiration timer. This ensures all messages for a session traverse the same set of CASP nodes, in both directions.

CASP messages can be generated in any intermediate node, either due to client state refreshes or a route change notification made to the messaging layer. In either case, they may traverse the remainder of the CASP nodes, until their TTL reaches zero or the signaling reaches the intended target address.

Multiple next and previous hops may be maintained for a single CASP messaging state, differentiated using *next-hop branch identifiers* (see Section III for more details).

C. Next-hop Discovery

In RSVP, a PATH message (using UDP or raw IP with a router alert option) allows any RSVP nodes along the path towards its destination to intercept and process the message. In the CASP architecture, to allow extensibility of protocols, next-hop discovery is separated from signaling protocols. *Scout protocol*, a common discovery mechanism using RSVP PATH-like message with Router Alert option, is introduced to actively determine next CASP hop along the path without bothering application functionalities. However, each node can choose its own next-node discovery mechanism, relying on manual configuration, router advertisements, link state routing protocols, scout, or, for loosely-path-coupled operation, server discovery solutions such as DNS or SLP.

D. Signaling Client Protocols

Client protocols perform actual signaling tasks. In the QoS resource reservation client [12], we define five message types: reserve, commit, reserve, query, response and release. Reserve and commit messages can create a reserve state and a commit state, identified by their client session identifiers, respectively. Both types of QoS client states are soft-state; to maintain the client session they need to be refreshed by sending reserve and commit messages before the corresponding state expires. QoS resources being reserved will not be able to used until being committed; a release message release associated reserve and commit states.

The signaling client protocols are independent from each other. In addition to QoS resource reservation client, the CASP architecture can also support any other signaling client protocols. Multiple clients may be supported simultaneously and possibly grouped within a messaging layer session (when sharing a same flow identifier), however, the messaging layer only delivers a single client message at one time, to meet different timing requirements of delivering different client messages.

III. Mobility Support in the CASP Architecture

In addition to direct routing between a mobile node (MN) and the corresponding node (CN), Mobile IP (MIP) also may introduce one or more IP-in-IP encapsulation tunnel(s) as part of or the full route between the home agent (HA) and the corresponding node (CN). In the following subsections, first we describe our approach for the direct routing (i.e., no tunnel) case, then extend it to the mobile IP tunnel case.

A. Basic Approach

When trying to support direct routing in MIP for CASP signaling, three problems arise. First, it becomes a problem if state information stored at routers are indexed with the Care of Address (CoA), as the CoA is subject to change due to handoffs. Second, a same state should not be established twice along the path (in case of QoS resource reservation, this is referred as "double reservations" problem). Third, the existing state on the "*abandoned*" path (i.e. the path between the cross-over node and the old access router) should be released after a handoff.

Through the session identifier uniquely selected by the initiator, and the flow identifier reflecting the sender and receiver information, the first two problems are resolved. When the MN moves, the CASP messaging layer only changes the flow identifier of the session, without changing the session identifier, where it detects the introduction or release of MIP tunnels (see Section III-B for details) or simply a route change in the MN or the CN. Then it triggers the next-hop discovery client to determine the new next signaling node and updates its information in the messaging-layer state according to the unique session identifier.

Releasing of existing state in the old path, by default, can be done by the state timer expiration. However, as the state timer is relatively long, keeping state in these nodes may be inefficient. Alternatively, we can use local explicit teardown messages. A reasonable place for initiating such a teardown message is the cross-over node.

We introduce a next-hop branch identifier to help the release of the abandoned states and determine the behavior of the cross-over node, e.g., the node where the old and new paths merge after a route has changed. Once a new next signaling node is determined, the messaging layer state in the current associates it with a new next-hop branch identifier (to represent the new branch). Then a refresh message is sent through the new branch to establish the necessary states, until the cross-over node with a same session state is reached. After that a teardown message assigned with the old branch identifier can be sent resersely towards another end point and terminated by finding a different branch identifier for the same session state. We call this procedure "*local repair*" and it is also applicable for normal route change cases other than mobile IP.

In the example in Fig. 3, an MN acting as signaling initiator communicates with a CN. When it changes its AR to that of a new network, it is associated with a new CoA (nCoA) and the flow identifier in the signaling message is changed. The session identifier, however, remains intact so that only a single state is held in the CASP nodes along the path from the cross-over node to the CN. After the cross-over node is determined, it can issue a teardown message to release states towards the MN along the reverse path that former signaling messages traverse.

There are two issues with local repair. First, local repair may result in session states to be unnecessarily released in nodes; this has been resolved by comparing the branch identifier

in the release message with the session state. Second, local repair in fast movement may cause errors. For example, if the MN continues to move after it sends a refresh message to the discovered new branch, it might happen that this refresh message may arrive at the cross-over node later than the latest-sent refresh message. This will cause an error in signaling (see steps 1-8 in Fig. 4, where steps 7 and 8 are handled incorrectly due to the last-come refresh from branch 2), as states in the latest path will be released upon the recept of the last refresh message in the cross-over node, but the MN might be not connected to the previous branch any more. Another instance of the same problem is ping-pong, where an MN moves between the same two ARs rapidly. To overcome this, we assign the branch identifiers in an incremental way and define only signaling messages with a branch identifier larger than the branch identifier of the same session can update or initiate the release of session states.

Note the refresh messages arriving at the cross-over node should be forwarded on towards the signaling target, to refresh existing states to reflect the change in flow identifier. Additionally, creation, update or removal of messaging-layer state also triggers associated local signaling client(s) to create, update or release related client state accordingly.

B. Operation over Mobile IP Tunnels

CASP enables the delivery of signaling messages in MIP tunnels by hop-by-hop addressing for signaling messages and separating next-hop discovery from message delivery. However, there are still a few issues with mobile IP tunnels. First, if signaling into tunnels is necessary, the tunnel end points should support CASP. As mobile IP tunnels can be long — possibly even crossing the core — even though the two tunnel end points may be physically adjacent, a general assumption can be made that MIP tunnel end points should support CASP signaling.

Furthermore, at tunnel entry points or CASP nodes inside a tunnel, unlike normal IP routing cases, it is infeasible to discover next CASP hop by the target address. The solution is to use the tunnel exit address as the destination to the discovery request message in this case. Discovery requests initiated at the tunnel exit or outside a tunnel are destined to the normal destination, e.g., the MN or the CN.

A pseudo-code for mobility support in CASP summarizing discussions above is shown in Algorithm 1.

IV. Security Considerations

Due to the separation of the discovery mechanism from signaling messages and the change from end-to-end addressing to hop-by-hop addressing in CASP, existing security protocols such as TLS [11] and IPsec [13] can be used without modification. These protocols support a number of different authentication and key exchange protocols (e.g. IKE[14]) with different properties. Security for individual client data objects in a non hop-by-hop way is provided with the help of CMS [15] at the level of client protocols when selective protection for the semantics of these objects is required.

Algorithm 1 Pseudocode of mobility support for CASP signaling

```
/* State initialization */
s_id = choose_session_id();
branch_id = 0;
creat_m_state(&m_state, branch_id, s_id, ...);

/* for MN, CN, HA or FA/MAP/GFA/... */
if (a node gets a mobility route change notification) then
  flow_id = (nCoA, CN, ...);
  next_hop = discovery(dest); /* if the node is a tunnel entry
  or inside a tunnel, dest is the tunnel exit; otherwise dest
  is the normal dest */
  if (next_hop != old_next_hop) then
    /* this is a new branch */
    branch_id = (branch_id + 1) % MAX_VAL;
    creat_m_state(&m_state,branch_id,flow_id,s_id,...);
  else {update the existing state with the new flow_id}
    update(&m_state (s_id), flow_id);
  end if
  /* send a refresh, local repair if it is a new branch */
  mesg = new_msg (t_flag = "off", branch_id, flow_id,
  s_id,...); /* t_flag=off means to refresh, not teardown */
  send_msg(&mesg);
end if

/* Determine the cross-over node */
if (a node gets a msg with t_flag == 0) then
  if (s_id(msg) == s_id(m_state)) then
    /* decide whether to update branch_id */
    if (branch_id>=(branch_id(m_state)+1) % MAX_VAL)
    then
      branch_oid = branch_id(m_state);
      update(&m_state(s_id), branch_id(&msg));
    end if
    /* then send a teardown mesg in the old path */
    mesg=new_msg(t_flag=1, branch_oid, flow_id, s_id,...);
    send_msg(&mesg); /* send the new mesg backward */
    forward_msg(&msg); /* forward refresh mesg on */
  else {it is not the cross-over node}
    if (s_id(msg) does not exit in local state) then
      create(&m_state(s_id, flow_id(&msg));
    else
      update(&m_state(s_id), flow_id(&msg));
    end if
    forward_msg(&msg);
  end if
end if

/* Determine the end of teardown msg */
if (a node gets a msg with teardown_flag==1) then
  if (branch_id ¿ branch_id(&m_state) then
    remove(&m_state); remove(&client_state);
    forward_msg(&msg);
  end if/*else stop here, don't forward on the teardown
  msg*/
end if
```

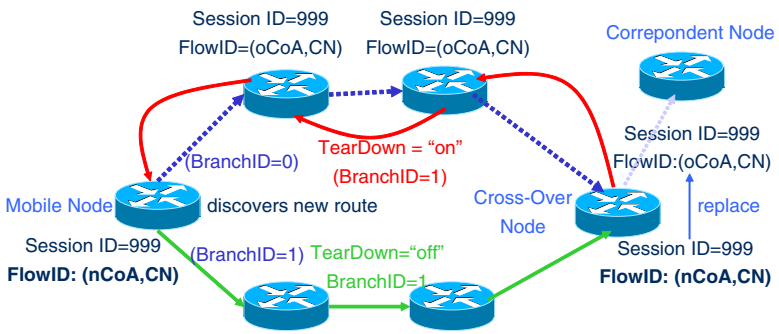

Fig. 3. Mobility support in the CASP architecture: basic approach

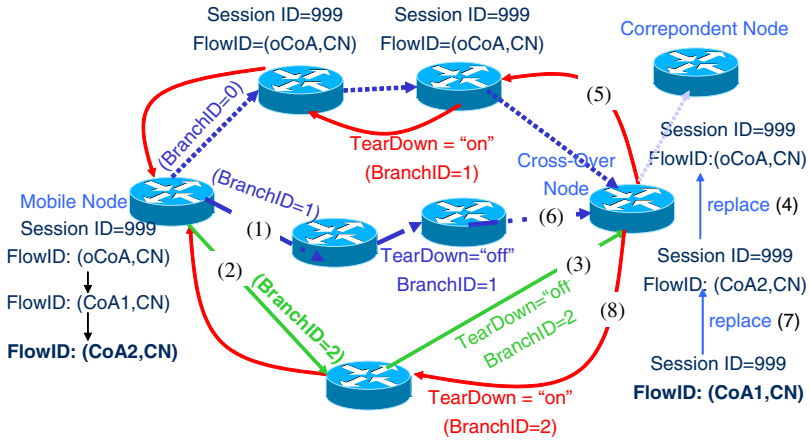

Fig. 4. Mobility support in the CASP architecture: possible error due to fst mobility

V. Summary

To summarize, our approach based on a layered signaling architecture presented in this paper overcomes the major challenges faced by existing techniques for providing generic signaling services over IP-based networks, including reliable, secure and congestion-controlled delivery of signaling messages of arbitrary size. Through the use of session and branch identifiers, old state can be torn down efficiently. In general, mobility support mechanisms introduced in this paper can be extended to more general route change and tunnel support.

Nevertheless, a number of open issues still exist. Of particular importance is the tradeoffs using different transport mechanisms and routing interfaces with mobile IP. We are currently investigating these issues through analytical modeling and a prototype implementation of general CASP components (messaging layer and discovery client) and certain critical signaling client protocols.

References

[1] L. Zhang, S. Deering, D. Estrin, S. Shen, and D. Zappala, "RSVP: A New Resource ReserVation Protocol," *IEEE Network*, vol. 7, no. 5, pp. 8–18, Sept. 1993.

[2] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource Reservation Protocol (RSVP) – Version 1 Functional Specification," RFC 2205, Sept. 1997.

[3] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, and S. Molendini, "RSVP Refresh Overhead Reduction Extensions," RFC 2961, Apr. 2001.

[4] C. Perkins, "IP Mobility Support for IPv4," Internet Engineering Task Force, RFC 3344, Aug. 2002.

[5] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6," Internet Engineering Task Force," Internet draft, work in progress, Mar. 2003.

[6] A. Talukdar, B. Badrinath, and A. Acharya, "MRSVP: a Resource Reservation Protocol for an Integrated Services Network with Mobile Hosts," *Wireless Networks*, vol. 7, no. 1, pp. 5–19, 2001.

[7] S.-B. Lee and A. Campbell, "Insignia: In-band Signaling Support for QoS in Mobile Ad Hoc Networks," in *Proc of 5th International Workshop on Mobile Multimedia Communications (MoMuC'98)*, Berlin, Germany, Oct. 1998.

[8] H. Schuzrinne, H. Tschofenig, X. Fu, J. Eisl, and R. Hancock, "CASP – Cross-Application Signaling Protocol," Internet draft, work in progress, Sept. 2002.

[9] B. Braden and B. Lindell, "A Two-level Architecture for Internet Signaling," Internet draft, work in progress, Oct. 2002.

[10] A. Terzis, J. Krawczyk, J. Wroclawski, and L. Zhang, "RSVP Operation over IP Tunnels," RFC 2746, Jan. 2000.

[11] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," Internet Engineering Task Force, RFC 2246, Jan. 1999.

[12] H. Schuzrinne, H. Tschofenig, X. Fu, and J. Eisl, "A Quality-of-Service Resource Allocation Client for CASP," Internet draft, work in progress, Mar. 2003.

[13] R. Atkinson and S. Kent, "Security Architecture for the Internet Protocol," Internet Engineering Task Force, RFC 2401, Nov. 1998.

[14] D. Carrel and D. Harkins, "The Internet Key Exchange (IKE)," Internet Engineering Task Force, RFC 2409, Nov. 1998.

[15] R. Housley, "Cryptographic Message Syntax," Internet Engineering Task Force, RFC 2630, June 1999.