ELSEVIER

COmputer NETworks

Beyond QoS signaling: A new generic IP signaling framework

Xiaoming Fu [a,*], Hannes Tschofenig [b], Dieter Hogrefe [a]

[a] *Institute for Informatics, University of Göttingen, Lotzestr. 16–18, 37083 Göttingen, Germany*
[b] *Siemens Corporate Technology, Otto-Hahn-Ring 6, 81739 Munich, Germany*

Abstract

This paper describes the design principles and an introduction of a framework and protocols for generic IP signaling, namely the Cross-Application Signaling Protocol (CASP) and its signaling applications. While reusing certain features of the existing RSVP protocol, CASP overcomes its shortcomings and may be deployed as a replacement technology to provide simpler, mobility-supported, more extensible and more secure signaling services in IP based networks. This paper discusses challenges of today's IP signaling protocols and addresses fundamentals and key aspects of CASP and its current signaling applications. In addition, a comparison with previous signaling protocol proposals and an outlook of future work in this area are also given.
© 2006 Elsevier B.V. All rights reserved.

1. Introduction

Signaling protocols are necessary to install and manipulate states in network nodes. Examples of typical signaling protocols include Signaling System 7 (SS7) [1] in telephony networks and ITU Q.2931 for ATM networks. In the Internet, which was designed as a connectionless, packet-switched network, IP datagrams are multiplexed in network nodes, and processed and forwarded in a "best-effort" way. As a result, the current Internet does not maintain flow-based state in the routers, except for some middleboxes, such as Network Address Translators (NATs) and stateful packet filtering firewalls (FWs). However, with its rapid development, the Internet is being used for an increasing number of different applications. Some of these applications demand services similar to that of a circuit-switched network, for example, real time applications would require certain Quality-of-Service (QoS) support from the network, while network administrators may wish to allow the traversal of NATs and firewalls, or to collect the network performance data from an end-to-end path. To deliver such services

---
* Corresponding author. Tel.: +49 551 39 14411; fax: +49 551 39 14403.

E-mail addresses: fu@cs.uni-goettingen.de (X. Fu), hannes. tschofenig@siemens.com (H. Tschofenig), hogrefe@cs.uni-goettingen.de (D. Hogrefe).

for a wide range of Internet applications, signaling protocols which manage state in Internet nodes along the data path are necessary.

Since QoS is regarded as a vital feature for the next generation Internet, QoS signaling has been one of the hottest topics in the area of Internet research. In early years, when it was believed that multicast was going to be a popular fashion in communications, the Internet Engineering Task Force (IETF) designed a multicast-oriented experimental protocol for resource reservation, which is called STream protocol Version 2 (ST-II) [2]. ST-II supports point-to-multipoint multicast communication; however it encounters complexity and scalability problems in terms of the number of receivers and size of reservation states. The Resource ReSerVation Protocol (RSVP) [3,4] was then designed to provide support for multipoint-to-multipoint reservation setup in a more efficient way.

RSVP introduces a number of important features for Internet signaling, such as soft state, two-pass signaling message exchange, and modularity through the use of opaque objects. It therefore quickly emerged at the forefront of academic and industrial interests and was regarded as a good candidate for a basic signaling protocol for IP-based networks.

However, because RSVP was originally designed for optimal support of QoS resource reservation in an early QoS model – the Integrated Services model – and not for general-purpose signaling, there have been debates regarding RSVP's complexity, security, scalability and modularity in order to meet new requirements over the years. To address these problems, the IETF Next Steps In Signaling (NSIS) working group [5] was formed to investigate the new current signaling deficiencies and to collect the new requirements. As an outcome the working group started their work on a generic signaling protocol suite. These protocols, still in beta-revision, could reinvigorate the drive for standards to provide various signaling services via a universal means, such as in QoS reservations and configuring NATs or firewalls. This would allow network providers to build value-added services that are more secure and extensible than existing ones. For end users, this would mean that they are not tied to one vendor's equipment.

The rest of the paper is organized as follows. Section 2 discusses the general principles of Internet signaling and challenges that current protocols face. Section 3 presents an overview of the Cross-Application Signaling Protocol (CASP), an IP signaling framework which is designed to meet requirements for next generation signaling. Section 4 explains details about CASP components and how CASP masters challenges. A comparison of properties (such as protocol overhead and security) of CASP against other approaches is presented in Section 5. Finally, Section 6 summarizes the paper and outlines future works.

2. Principles and challenges of IP signaling

For the purpose of this discussion, a signaling protocol establishes, maintains and deletes state in a number of nodes which end-to-end flows traverse. State manipulation can either be done in a "soft" or in a "hard" way. SS7 and ST-II are example signaling protocols using hard state. Hard state refers to a state created in the network which can only be removed upon receiving an explicit tear-down message for deletion, while soft state refers to a non-permanent state which will expire unless refreshed. In comparison to hard state signaling, soft state signaling is preferred as it provides better robustness in case of network situation changes, allowing the end-to-end communication to recover to a desired configuration without the need to send explicit messages.

The soft state concept was introduced with the RSVP protocol originally designed for Internet QoS signaling, particularly for the IntServ QoS model [6]. The basic operation of RSVP can be illustrated as follows.

As shown in Fig. 1, the host transmits the *PATH* message to the unicast or multicast group address(es) of the receiver(s), which conveys the sender information and its data flow characteristics (*TSpec*). The *PATH* message, which is encapsulated in a raw IP or UDP datagram with a Router Alert Option, is sent through all the routers along the path of the data flow, and at each RSVP router the RSVP process intercepts it, and creates a *PATH* state. This state allows response messages to travel backwards along the previously established path. Upon receipt of the *PATH* message, a *RESV* message is sent back toward the sender following the same path in the reverse direction, requesting the necessary resources (such as guaranteed bandwidth) for the data flow (*FlowSpec*). The data receiver authorizes the QoS reservation in the *RESV* message. Each RSVP router attempts to establish and maintain a *RESV* state to provide the requested service upon the receipt of the *RESV* message. Both *PATH* and *RESV* states are soft: *PATH* and *RESV*
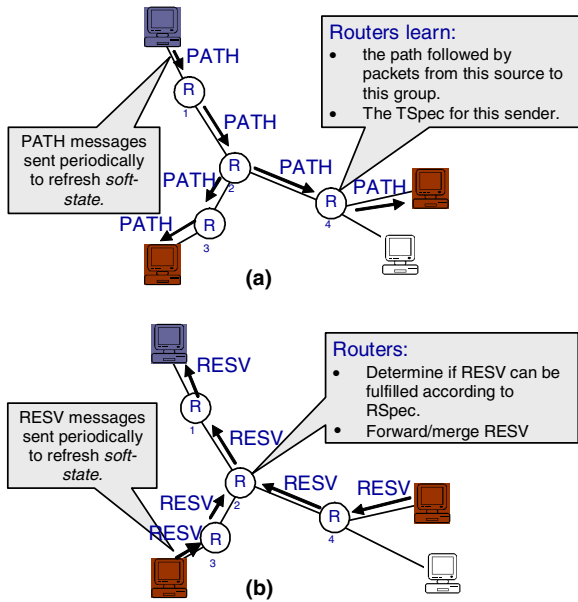
Fig. 1. RSVP basic operation: (a) RSVP PATH messages sent by sender(s) and (b) RSVP RESV messages sent by receiver(s).

messages are sent periodically by hosts and routers to maintain these states, otherwise they will time out. The established state can be removed when the end host sends an explicit tear-down message after the application has finished sending the data. The end host may also rely on the soft-state time-out. RSVP does not rely on specific routing protocols and adapts to multicast group membership changes.

It is possible to reuse RSVP for general-purpose Internet signaling. In fact it has been extended for many signaling purposes other than Internet QoS, such as label distribution for traffic engineering with RSVP-TE [7]. It has to be mentioned that the usage of RSVP in the domain of MPLS modifies some of the basic RSVP design principles and is as such not a classical application of path-coupled signaling. However, many intrinsic design limitations in RSVP have hindered its acceptance as a general-purpose signaling protocol, which motivated the introduction of a new framework. Some critical challenges are:

Protocol complexity: First, the original RSVP makes per-flow reservations, requiring a pair of PATH and RESV state in RSVP routers that are indispensable part of the protocol operation, which is intended for the receiver-originated resource reservation purpose and optimized for

multicast flows. This in turn requires a number of unnecessary objects and error handling for unicast data traffic. For the majority of signaling services which do not require multicast,[1] this only introduces overhead in protocol processing. Considering state overhead in each router is proportional to the number of data flows,[2] which requires non-trivial processing overhead. Although hop-by-hop reliability has been added later [9], in the basic RSVP specification [4], signaling messages are refreshed without distinguishing between message losses due to network congestion, route change or other message corruption. Thus, the reliability of signaling process has to rely on relatively long-term periodic refreshes, which makes it difficult to meet the requirements in all scenarios.

Security: RSVP does provide basic security mechanisms with the RSVP Integrity Object [10] and with extensions for Identity Representation [11]. However, it does not allow standard security mechanisms, such as Transport Layer Security (TLS) [35] or IP Security (IPsec) [36], to be used because of the end-to-end addressing of a number of messages (e.g., PATH messages). Out-of-band key distribution is assumed and the selection of security association for end-to-end addressed messages is based on routing table lookups which is only useful in certain environments where the discovery nature of the end-to-end addressed messages is not meaningful. Some deficiencies can also be identified in the area of user authentication and authorization in roaming environments, primarily due to the interaction with COPS [12]. To the best of our knowledge, COPS has never experienced the same degree of deployment as Radius [13] or Diameter [14] and is therefore not regarded as an inter-domain AAA solution. For accounting and credit control COPS does not provide the adequate support and Radius or Diameter has to be used. Furthermore, the missing support for sender-initiated QoS reservations is not aligned with the common practice, where charging and pricing for the Internet service are often made for data senders.

---

[1] According to http://www.multicasttech.com/multicast_faq.html, the penetration of multicast in the Internet only accounts for a minority of less than 5% after nearly two decades development in multicast.

[2] Aggregated RSVP reservations [8] allows coarser-grained state but the basic RSVP operations are the same.

Mobility: RSVP was designed when node mobility had not yet matured. Unfortunately, even modest mobility introduces complications for RSVP [19,20]. For example, route changes and change of the end host's IP address, are natural subsequences upon a mobile node's movement. For downlink reservations (i.e., those with targeted data flow sending from the corresponding node to the mobile node), one has to rely on a next PATH refresh message, since a re-establishment of the RSVP routing path states can only be triggered by end-to-end addressed PATH messages. Furthermore, the flow identifier (flow ID) in RSVP, which is used to uniquely identify state in RSVP routers, consists of IP addresses and port numbers of sender(s) and receiver(s). According to Mobile IP [15,16], a mobile node will obtain a new IP address (care-of address) at the new point of attachment. Thus, a flow ID will be changed after a mobile node moves to a new location during an ongoing application session. Moreover, most IP mobility schemes introduce IP tunnels of some kind to allow the data traffic to reach the current location of the mobile node. To support these scenarios in RSVP is very challenging: even though some proposals attempted to extend RSVP for use in mobile environment (e.g., [18,19,17], to be shortly discussed in Section 3), there are still a number of open issues [25].

Extensibility: First, RSVP was designed to carry objects encoded in "Type-Length-Value" (TLV) opaque to QoS models. However, RSVP X-Specs (*TSpec*, *FlowSpec*, etc.) are mostly QoS-specific. There is no clear distinction between application-specific functions and general-purpose signaling functions. Second, signaling objects may vary in size – they can be fairly large, such as certificates, authorization tokens or active networking code. If a signaling message is too large and exceeds the link MTU, IP fragmentation is needed. This can lead to problems, e.g., in IPv6 where routers do not perform fragmentation. Thus, in RSVP, the messages size is limited by the MTU. Third, the overall design of RSVP was flat: all functionalities (reliability, soft-state, routing of signaling messages, multicast, etc) are integrated in single PATH/RESV messages. It was well-suited for Internet QoS signaling, but does not fit for general purpose signaling services.

It is questionable whether a revised RSVP (with all the desired features included and others removed) should be called RSVP especially if it is not backward compatible anymore. We think that the changes are substantial enough to give the protocol (and the framework) a new name.

3. CASP – A framework for next generation IP signaling

In order to meet some of the requirements listed in the previous section, e.g., mobility and extensibility, some attempts to adapt RSVP have been made. Some proposals, such as MRSVP (proposed by Talukdar et al. [17]), RSVP Flow Transparency (proposed by Shen et al. [18]), and Localized RSVP (proposed by Manner et al. [19]), attempted to resolve mobility issues with RSVP. For the convenience of the discussion in this paper we will refer to these types of approaches as "M-RSVP". The basic idea is to introduce a mobility-independent flow ID and thus after mobility takes place, a subsequent RSVP signaling procedure with the same flow ID will update established state information. Based on RSVP's approach for receiver-initiated reservations authorization problems arise and also result in a long latency in re-establishing QoS states (see for example [20]). Furthermore, it is difficult to remove established states along the old path (most likely, only by relatively long-term state timeouts), whereas this is important for frequent path changes due to mobility.

These approaches mentioned above address mobility but none of them demonstrates a greater extensibility in supporting various signaling services. Therefore, recent research by Braden and Lindell [21] has attempted to separate its general signaling functionality from application-specific signaling functionalities, known as "Two-Level Architecture for Internet Signaling" (referred as "Two-level RSVP" hereafter). This approach makes a distinction between application-independent signaling services (referred as NSIS Transport Layer Protocol or NTLP in the NSIS terminology) and application-specific signaling services (provided by the NSIS Signaling Layer Protocols or NSLPs) to extend RSVP for general signaling. Work by Shore [22], Brunner and Greco [23], Westberg et al. [24] have developed some plausible system designs that demonstrate advantages based on Two-level RSVP signaling.

In principle, all approaches described above can be categorized as filter-based approaches, which aim to perform signaling while discovering the next hop. However, due to the complexity and difficulty

in supporting large signaling messages, mobility, congestion control and security, these approaches – which were directly derived from RSVP – failed to achieve wide acceptance, as pointed out during the NSIS analysis phase [25].

Therefore, we developed a new approach as an alternative to these approaches, the so-called Cross-Application Signaling Protocol (CASP) [26,27]. In comparison to filter-based approaches, CASP introduces the separation of discovery from signaling (herein referred as "discovery-based approach"), and a new control plane concept of "session identifier" to uniquely identify signaling sessions, unlike the flow ID used in RSVP, while maintaining some merits of filter-based approaches (e.g, "router-alert" way for next-hop discovery). The framework and protocols based on CASP concepts were developed and turned into a standardization track after CASP's initial documents came out in September 2002.

3.1. CASP overview

As shown in Fig. 2, the CASP architecture consists of (1) a generic messaging layer, which transports signaling messages between the sender of the signaling session and the responder, as well as maintaining messaging layer state, and (2) 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 signaling and resource reservation, firewall/NAT 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
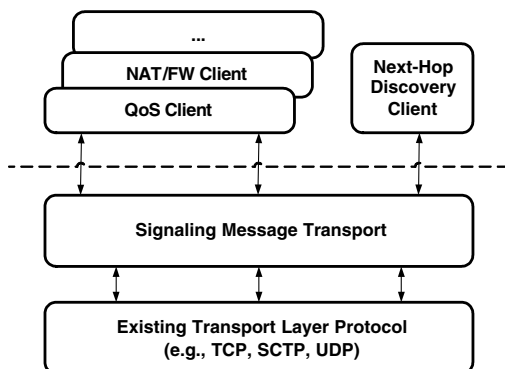
reserving resources, as well as receiver-initiated ones. The *Scout* protocol, a common discovery mechanism using RSVP *PATH*-like message with Router Alert Option, is introduced to actively determine next CASP hop along the path without bothering application functionalities. However, each node can choose its own next-hop 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, DHCP or SLP [28] when available, which in most cases could be more efficient than the *Scout* protocol. The process of discovering the next-hop NSIS aware node introduces security vulnerabilities that need to be addressed for each individual protocol. The security threats and the respective countermeasures of *Scout* are analyzed and discussed in [26].

Modern reliable transport protocols offer flow control, congestion control, fragmentation and reliability, which are important characteristics for a generic signaling protocol. For example, digitally signed messages or active networking code can be fairly large. Such large messages may need fragmentation and congestion control, which are the functionalities of TCP and SCTP [29], but not of unreliable transport protocols like UDP. Therefore, 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 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, depending on the number of imposed signaling sessions and the client layer refresh intervals, non-edge signaling peer nodes may communicate with each other with their client message exchanges before a transport connection expires and thus keeping their transport connections active for a long time, and avoiding the connection set up latency; if necessary (e.g., between non-edge neighboring CASP nodes, especially where route changes are more likely), the messaging layer can also have "keep-alive" soft state refreshes, without involving client layer. As a result, the average session setup latency is low. Furthermore, a messaging layer state teardown does not necessarily teardown the underlying transport connection, since it may be needed for a later session.

Fig. 2. The CASP architecture.

Unlike normal CASP messages, CASP *Scout Request* messages utilize UDP with a Router Alert Option [30,31] for its transport (like RSVP PATH messages). Although the destination of a *Scout Request* is the data receiver, the first CASP node with the desired client layer functionality along the path will respond (with a *Scout Response*) without forwarding it further on. *Scout Requests* have their own reliability mechanism. They are retransmitted periodically, with an exponentially increasing retransmission interval, which is a relatively small value.

3.2. CASP operation

A CASP messaging layer session is established between an initiator and a responder, along a chain of CASP nodes, with a cryptographically generated random session identifier (session ID) chosen by the initiator [53]. Additionally, a flow ID describes the data flow the signaling message pertains to. At each node, the CASP messaging layer remembers its previous CASP node, aside from the initiator. It also determines the next node along the data path, checks if there is an existing transport connection to that node; if not, establishes one, 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 that all messages for a session traverse the same set of CASP nodes, in both directions. The importance of the session identifier is described in Section 4.3. While delivering generic messaging layer signaling messages, the messaging layer establishes, refreshes, or releases states for signaling sessions; it also remembers the traversed path by installing state at individual routers (stateful approach) or records a route (stateless approach). Without requiring a CASP message to be delivered to the target, the CASP client layer decides whether a received CASP message needs to be forwarded on, or simply processed and/or responded with a feedback. Moreover, CASP utilizes a session identifier concept, which is a (probabilistically) unique and independent of flow sender or receiver. The session identifier is part of the signaling message and also the primary key to the messaging layer state, allowing proper operation in mobile environments (as it will be described in Section 4.3).

Fig. 3 illustrates an example of CASP signaling where TCP is used as the underlying transport mechanism (the SCTP case is similar; when UDP
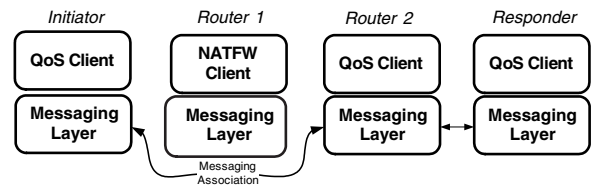
Fig. 3. An example of CASP signaling.

is used, a CASP message will be sent to a discovered next hop without checking or reusing any existing connection). The QoS signaling client of the initiator requests the CASP messaging layer to deliver its services from the initiator towards the responder. It is possible that some intermediate CASP nodes (in this example, Router 1) do not support the requested client layer functionality and the following operations take place:

(1) The initiator creates a messaging layer session identifier, and determines that the next CASP node supporting QoS client is Router 2. The initiator triggers the Scout discovery process periodically by sending a Scout request towards the destination address. Router 2, in our example, will respond with a Scout response message. If there is an existing TCP connection between the initiator and Router 2 then a CASP message is generated and delivered to Router 2.
(2) Upon receipt of the CASP message sent by the initiator, R2 passes its client payload on to the correspondent QoS client before forwarding the CASP message further. Additionally, it also remembers the previous hop. Router 2 might need to perform the same discovery procedure again. After determining that the Responder is the next hop to send CASP message, Router 2 establishes a TCP connection between itself and the responder, if no such TCP connection exists.
(3) After the responder receives the CASP message and processes the QoS client data, it may need to return an acknowledgment to the initiator.

4. CASP design details

4.1. Application-specific signaling and extensibility under the CASP framework

Theoretically, CASP can support any number of application-specific signaling protocols (and discovery mechanisms). We developed two demanding

application-specific signaling protocols, namely the QoS resource reservation client for CASP (CASP-QoS [32]) and the firewall/NAT signaling client CASP (CASP-FW/NAT [33]). Based on services provided by CASP general signaling support, CASP-QoS intends to support QoS signaling functionalities provided by RSVP, while CASP-FW/NAT installs firewall pinholes and creates NAT bindings along the path. These clients have their own soft state but are not concerned with message transmission (such as dealing with packet loss, fragmentation and retransmission; these are handled by CASP messaging layer through the reuse of existing transport protocols). Before presenting these current signaling applications of CASP, we discuss the interactions between these client protocols and the general signaling transport.

Any signaling client can request general signaling transport services through a few primitives: *Client-Req*, *ClientResp*, and *ClientNotification*. In CASP, separation of a generic messaging layer from an application-specific client layer allows for any other signaling client protocols to be easily added. Each client only relies on common CASP signaling transport services and can be changed without affecting other clients. Furthermore, separation of a next hop discovery functionality from the messaging layer allows easier security protection of signaling procedures, and avoids complexity in messaging layer. This also helps to remove the restriction on the application signaling protocol, such as message size to be limited to MTU or introducing lower-layer overhead. Each client signaling application or discovery mechanism is identified by a type to be registered with IANA, and then used; this is much like RSVP object definition but allows a better extensibility due to its more modular design.

As the first signaling application supported by the CASP framework, CASP-QoS supports both sender-initiated and receiver-initiated reservations, mostly designed for unicast communications. It defines five message types: *Query*, *Commit*, *Reserve*, *Release*, *Response*, each of which may contain several objects in TLV format, including bandwidth, partial reservation, IntServ Flowspec, etc. The combination of "Reserve–Commit" makes a typical reservation: after a Reserve message determines whether the resources along the data path are available for reservation on behalf of a flow, a Commit message will make the actual reservation. However, the Commit message can either immediately be initiated by the responder and follow the reverse chain

of nodes that the Reserve message traverses, or initiated by the initiator (after it receives a Response). Once created, QoS reservations are maintained as soft state: they will expire unless receiving periodical Reserve messages for refresh. In general, a Response message can be an answer to a Reserve or Commit message, and reports the result of these operations. In addition, the initiator can use a Query message to find out whether resources are available at any time along the CASP-QoS node chain, from the received Response message. An example message flow for CASP-QoS operation is shown in Fig. 4.

The second signaling application after CASP-QoS is CASP-FW/NAT [33]. It allows nodes to signal information to firewalls, or to establish a NAT binding, as well as to provide the signaling initiator with the NAT information, and supports both sender-initiated and receiver-initiated operations. It defines six message types: *Path*, *Create*, *Release*, *Response*, *Query*, and *Trigger*. A Create message allows to establish or update FW/NAT state (e.g., flow ID, such as source/destination address, port number, transport protocol, SPI, etc.) upon a successful authorization and verification of the security policy. If the verification fails, a Response message with error indication is returned to the requesting entity. In the receiver-initiated operation, a Path message is used as a trigger for the responder to issue a Create message, whereas a Create message can be issued alone without a path trigger in a sender-initiated operation. A FW/NAT device can send an asynchronous event notification – the trigger message – to the end node. In addition, a Query message provides diagnostics functionality for the initiator to look up FW/NAT state along the path
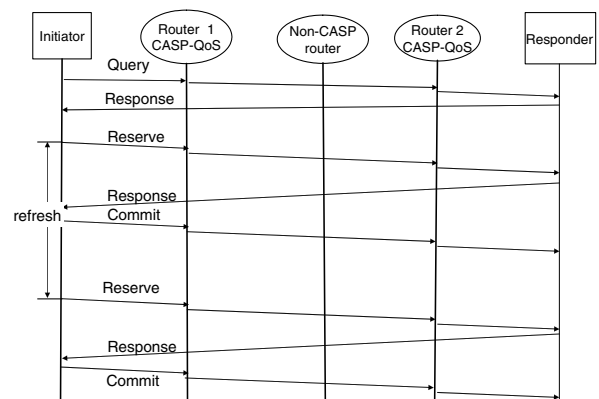

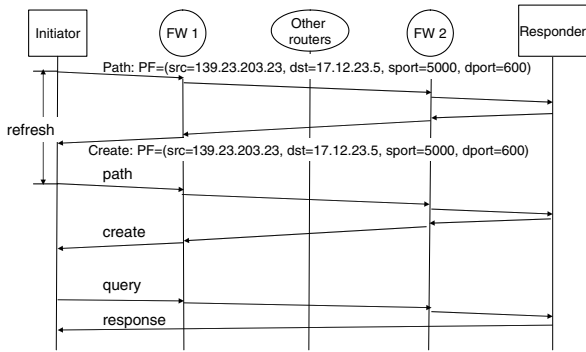Fig. 4. An example of CASP-QoS operation.

Fig. 5. An example of CASP-FW/NAT operation.

with the received response message. A typical example of CASP-FW/NAT operation with regard to receiver-initiated firewall signaling is illustrated in Fig. 5.

Message format for the CASP protocol suite can be found in Appendix A.

4.2. General signaling transport

As depicted in Section 3.1, we can view the signaling architecture in CASP as three distinct parts: CASP messaging layer, the CASP client layer and the specialized client responsible for next hop discovery. In this section we present the CASP messaging layer and the Scout protocol for next hop discovery, which comprise the key functionality of general signaling transport and use of the following three types of messages:

Scout request (*ScoutReq*), which contains a router alert option and requested application type and will be processed (and by replying a scout response) if a CASP node receiving this message supports the given application type.
Scout response (*ScoutResp*), which indicates the information about discovered next hop (IP address, etc).
CASP signaling message (*CaspMesg*), which carries a session ID and other information (such as flow ID) for routing messages towards the responder, as well as an ADD/DEL flag indicating to install or remove state;

Furthermore, there are several internal messages within the CASP general signaling transport mechanism, namely Discovery request (*DiscReq*), and Discovery response (*DiscFail* and *DiscSucc*).

In order to provide enough robustness, the general signaling transport mechanism in CASP supports soft state in reacting to changes of network situations (for example, see Section 4.3 for discussions on route change and mobility cases). The state in messaging layer indicates the delivery path of a given session, uniquely identified by a session ID. Additionally, associated with a soft state there are two types of timers: a refresh timer in signaling initiator whose expiration triggers a refresh of soft state along the path; and a state timer in all participating CASP nodes which controls the expiration of the soft state.

Figs. 6 and 7 depict the finite state machines for the general signaling transport mechanism provided in CASP. The scout protocol has only two general states: Idle and WaitResp (waiting for scout response), whereas the messaging layer keeps track of much more complex states: Idle, WaitDisc (waiting for discovery success), NHopDiscd (next hop is discovered but state not established), and Established. Their transitions with different event triggers show how the CASP signaling transport defined in [26] may be implemented. However, implementers can build their own CASP signaling systems and choose their own way of internal processing of CASP messages and interfacing with applications using CASP.

4.3. Mobility support in CASP

By its intrinsic design, CASP signaling supports route change and mobility scenarios, due to the introduction of a unique session ID that is
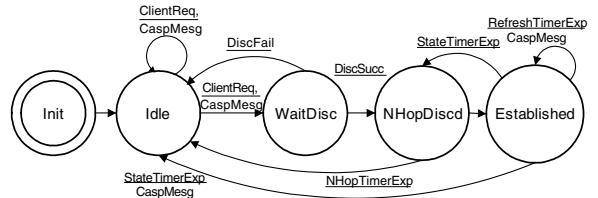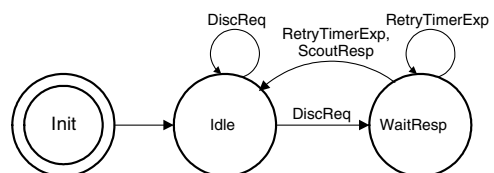

Fig. 6. CASP messaging layer state machine.

Fig. 7. CASP Scout protocol state machine.

independent of source and destination IP addresses, the extensible discovery component, and additional a Branch_ID which identifies the changed signaling path segment. A CASP node use a counter for the Branch_ID, incrementing it by one when a new branch occurs. A node needs to be able to determine which branch is the "most up-to-date" one. Branch_IDs typically are just "locally" meaningful, not necessarily end-to-end.

In a typical route change example in Fig. 8, a CASP session is initiated and travels along the upper dotted path through a branching node towards its responder. At the branching node the path segmented is initially identified as Branch_ID (B) = 1. In case of end host mobility, the leftmost B = 1 is not necessary, because the branching node (node 1) is the mobile node itself. When a route change or mobility event is detected (e.g., through some signal about the completion of the new mobility registration procedure, or a notification from a local route/binding table change via a CASP/routing interface), a new discovery procedure will be performed and the new path is identified with an incremental number (B = 2 in this example). A CASP signaling message is sent along the new path to establish the necessary state (lower dotted lines in Fig. 8); creation of a new CASP messaging state can also trigger the creation of a new CASP signaling client state if necessary. When a converging node (node 6 in Fig. 8) is reached, which is identified by finding a node with the same session ID, this node can initiate a delete operation to remove the existing state along the reverse direction towards the initiator. This process will be stopped when a node with the same Branch_ID is met (see Fig. 8).

In a mobility scenario, a mobile node can have a new care-of address (CoA) after a handoff in addition to a route change. As CASP signaling messages carry a flow ID, while the session ID remains constant for a same signaling session, state recovery (in the new path segment) and removal (in the old segment) can be easily handled in a similar way as normal route change. Considerations about slightly more complex movement behaviors and details about CASP interworking with mobile IP can be found in [34].

Mobility support for a protocol like CASP also raises security concerns particularly regarding the right for a particular signaling entity to modify or delete state allocated along a number of CASP nodes. The problem statement for this particular authorization problem (known as *reservation or session ownership*) is documented in Section 4.10 of RFC 4081 [55] and solutions have been outlined in [53].

4.4. Security in CASP

Security needs to be addressed in a number of places in the CASP protocol suite:

- Security for the discovery mechanisms. In case of the Scout protocol basic security protection is provided using cookies and nonces to deal with off-path adversaries. Scout itself cannot provide protection against man-in-the-middle adversaries.
- Communication security between neighboring CASP nodes that implement the same signaling application. Authentication and key exchange protocol ensure that signaling messages can subsequently be integrity, replay and confidentiality protected.
- Signaling applications need to perform authorization based on the need for their application. The QoS signaling application might, for example, need to interact with an AAA infrastructure (as we illustrated in [54]).

Instead of inventing new security mechanisms, CASP tries to reuse existing ones as much as possible. The messaging layer provides security protection between neighboring CASP nodes by using TLS [35] and IPsec [36]. RSVP cannot use standard security protocols as described in detail in [37] due to the strong coupling of discovery and signaling message delivery. A number of authentication and key exchange protocols are available to secure the use of CASP, ranging from secret key-based to public key based authentication, for example, Kerberos, SRP or TLS public key based authentication. This flexibility is important, since there are a variety of
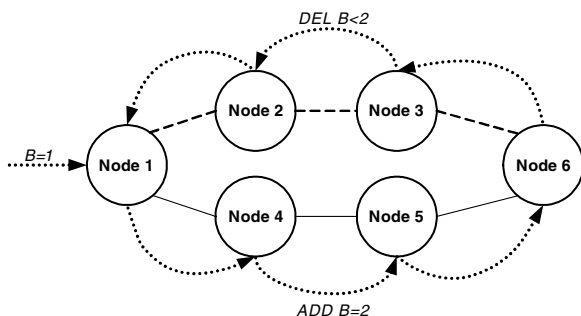

Fig. 8. An example of route change and mobility support in CASP.

environments using IP signaling, which can further affect different network segments in an end-to-end communication which have diverse security properties. In CASP, the protection between neighboring CASP nodes can provide integrity, replay, confidentiality protection and data origin authentication [26]. Additionally, IKE [38], IKEv2 [39] and the TLS Handshake protocol provide some protection against denial of service attacks and allow the two peers to authenticate each and to ensure that only authorized nodes can send signaling messages.

For additional security protection between non-neighboring CASP nodes, security mechanisms can be introduced for specific client layer applications. It turned out that QoS and FW/NAT signaling applications require different security protection due to their different authorization behavior. Since the authorization decision is highly signaling application specific it is necessary to deal with these aspects separately for each application, as shown in [40,33]. Offering these security services at the client layer is necessary since they are not common for all applications and a richer semantic is required. For example, an authorization decision for a QoS request is often associated with a monetary compensation (interested reader may see [46] for a detailed discussion on QoS authorization aspects).

Most security threats for the CASP-FW/NAT can be addressed by the lower layer based on the security protection offered between neighboring entities. Certain security functions can only be performed at a CASP-FW/NAT signaling application itself where policies based on roles or traits are applied. Furthermore, certain authorization decisions might need to be performed by intermediate nodes based on the initiator of the FW/NAT signaling exchange. Section 4 of [33] illustrates the different deployment and authorization scenarios.

The security mechanisms envisioned for the client layer have also focused on reusing existing state-of-the-art security protocols such as the Extensible Authentication Protocol (EAP) [41], Cryptographic Message Syntax (CMS) [42] and authorization tokens, for example using the Security Assertion Markup Language (SAML) [51,52], OSP tokens [49] or tokens defined in the context of SIP and RSVP [50].

5. Discussion

In this section, we discuss protocol properties of CASP in comparison with its precedence ST-II,

RSVP and RSVP variants, such as some of its standard extensions, as well as proposed mobility extensions (denoted as M-RSVP) and the Two-level RSVP. Our analysis considers protocol complexity, extensibility and security features in QoS signaling (since it is the only common function that all the protocols support) and the generic IP signaling aspect. We follow the work of Mitzel et al. [45], who analyzed the protocol complexity of RSVP in comparison with ST-II, with an emphasis on multicast group dynamics, but our focus will be on signaling for unicast communications.[3] Mobility and extensibility aspects have been discussed in earlier sections and are therefore not repeated in this section.

5.1. Protocol overhead

We consider three types of protocol overhead. First, there is a bandwidth overhead, i.e., each signaling-aware node may periodically send refresh messages to its active neighbor; second, the memory/storage overhead, corresponding to the requirements of the node to maintain "active neighbor" state as well as its own reservation state per end-to-end flow (memory overhead), and the third, processing overhead corresponding to the requirement of message processing and state handling such as recovery. Processing overhead can also be related to extensibility but will be discussed in Section 5.1.3. The refresh timers have a great effect on the protocol overhead and recovery period, and no explicit timer values are mandated by the protocol standards. Instead, we compare the design philosophies behind the runtime behaviors of the protocols.

5.1.1. Memory overhead

ST-II and basic RSVP [4] require a linear state repository and refreshing corresponding to served flow numbers in each signaling-aware node; however, RSVP extensions on reservation aggregation [8] and refresh overhead reduction [9] can reduce this overhead: aggregation allows reservation state to be stored in a coarser granularity. Thus, RSVP's memory overhead scales better than ST-II with the number of reservations. M-RSVP follows similar overhead to RSVP, with an exception that M-RSVP may add an additional requirement on memory

---

[3] Although it would be interesting to study IP multicast support in CASP but multicast does not seem to be a major deployment scenario, as discussed in Section 2.

consumption, namely the state may be reflected by additional parameters (such as permanent home address or HoA) other than that related to a mobile host's IP address (the topologically correct address, i.e., CoA). Two-level RSVP does not introduce changes in the RSVP state representation but mainly modifies message formats and semantics, therefore it follows similar memory overhead as RSVP. In comparison, the CASP messaging layer only requires maintaining transport connection state (including its neighboring CASP node information) which is shared by and opaque to all related reservation states (as well as other types of client states), thus the required memory overhead for message transport is lower than ST-II, RSVP, M-RSVP and Two-level RSVP. Our recent work [43] show that messaging layer session repository requires just between 128 and 256 bytes of per session state and tens of bytes for per-peer transport connection state, in addition to the overhead in normal messaging layer handler repository (e.g., 4K bytes per thread implementation which can be shared by hundreds of sessions), whereas an RSVP implementation [47] may require about 2K bytes for about 1000 signaling sessions. CASP client layer includes fraction of the memory that an equivalent RSVP needs, and can also utilize aggregations, thus, the message overhead in client layer is clearly lower than the other protocols.

5.1.2. Bandwidth overhead

Bandwidth overhead is mainly concerned with signaling message frequency and size, where frequency is a more important issue (as signaling messages only add small amount of header information to the signaling payload of possibly fairly large size). ST-II uses hard state with no refreshes, so it requires sophisticated state synchronization operations (thus message numbers) in case the network condition changes. RSVP and its variants use soft state refreshes to keep their reservation state alive during the lifetime of the signaling session (by default, every 30 s) [4,48]; the bundling mechanism [9] ensures that only a single reservation message is propagated over a link per refresh period, and the summary message can further reduce the signaling message size by just including a list of message_IDs. These refreshes serve for several purposes: (1) reliability of the signaling messages, (2) detecting (and state recovery) of route changes, (3) detecting (and state recovery) node failures, and (4) detecting aliveness of the signaling entities. As a result, RSVP

introduces a bandwidth overhead that is about the multiplication of refresh timer and the state number, except in the refresh reduction extension where bandwidth overhead is reduced.

In comparison, pure CASP messaging layer refreshes (without client data) are just for keeping transport connections alive. Given the use of reliable transport, this refresh is rarely sent – only when a node anticipates that it still needs to be there for a next signaling session; otherwise, if there is already any active session, delivery of a next client layer refresh message will automatically refresh the messaging layer state. Therefore, CASP results in a very low bandwidth overhead in messaging layer. For the CASP (QoS) client layer, its state is similar to normal RSVP reservation state, but only covers part of the RESV state concerning resource parameters without the need of having the PATH state. Furthermore, CASP can also use refresh overhead reduction, such as mechanisms of message bundling and refresh session IDs, and aggregations when necessary. Thus, overall protocol bandwidth overhead in client layer is lower than RSVP.

5.1.3. Processing overhead

Processing overhead (and the robustness of the protocol) is related to the protocol design aspect. As a hard state protocol, ST-II has chosen to utilize a new IP protocol for message transport and an active failure detection mechanism for state reliability based on Hello, Status, and Notify messages, followed up with complex recovery mechanisms, all of which add considerable complexity to the protocol. Basic RSVP relies on datagram protocols (IP or UDP) for message transport and pure soft state refreshes to automatically adapt to the link or node failures without additional protocol complexity. However, its inefficiency in recovery from failures should also be noted; thus RFC 2961 adds a per-hop notification to previous signaling hop for message reliability, which adds complexity in the RSVP protocol processing. Furthermore, RSVP targets at primarily multicast session (with unicast as a degenerate case of multicast), which in turn requires a number of additional complexity in determination or avoidance of message loops, reservation failure and killer problems [4], as well as a number of objects (e.g., styles) and related state machine operations.

In contrast, via the reuse of existing reliable transport protocols for message delivery, and focusing on signaling applications that are mostly unicast, CASP avoids protocol complexity inside a

general signaling protocol. Note, this nevertheless would likely result in a negative effect, namely the performance may be limited by the interaction between CASP and the underlying transport protocols, especially transport-layer connection setup, removal and congestion control properties. The interaction details will have to be studied and evaluated in-depth. On the other hand, the design of CASP messaging layer tries to minimize the effect of connection setup delay by reusing existing connections between neighboring peers. Individual CASP discovery clients can vary from each other on their discovery performance, but the basic discovery client scout processing has been preliminarily measured as hundreds of μs with marginal overhead in cookie generation and cookie comparison (with enhanced security) [43].

5.2. Extensibility

ST-II was designed specifically for QoS signaling in coupled with routing, whereas RSVP separates signaling from routing, and allows using new opaque objects for extensions, for example RSVP-TE and RSVP operation over tunneling and ATM networks [25]. However, the composition of RSVP messages still contains a number of objects specific to IntServ, multicast, or the two-way signaling message exchange procedure, such as Sender_Template, SCOPE, STYLE, RSVP_HOP. M-RSVP targets at a limited scope of QoS signaling using RSVP in mobility scenarios. Two-level RSVP, on the other hand, allows more general signaling purposes but still relies on the flow ID as in standard RSVP to manage state information.

In contrast, CASP introduces a session ID concept in messaging layer for identifying signaling session state but minimizes non-generic objects (flow ID is remained just for relating to data plane use of the signaling application state), provides mobility support (which is increasingly common), and extensibility to any discovery mechanism, allowing maximal extensibility for any signaling purposes. In the client layer, upon receipt of a CASP-QoS message, CASP-QoS can decide any type of subsequent protocol operation (without limiting to two-way reservation setup and receiver-orientation).

5.3. Security

ST-II did not consider security. Later stages of RSVP added some security features to RSVP

[10,11], however, the encapsulation of PATH and PathTear messages using the router alert option and destined to flow destination fundamentally introduce a security hole for any operational RSVP nodes. Most proposals for mobility support for RSVP, as well as Two-level RSVP, do not consider security. In contrast, by the use of Session ID in the control plane and distinct discovery client, CASP provides a better security framework for protocol extension from the very beginning [26,53].

6. Summary and outlook

To summarize, CASP distinguishes itself as a new Internet signaling framework from previous approaches by its key design principles. This includes the separation of general signaling transport from application-specific signaling, the separation of discovery from signaling, as well as reusing existing transport protocols, thereby addressing the major challenges for next generation Internet signaling. A detailed comparison of a few relevant signaling approaches discussed in this paper is provided in Table 1. It can be concluded that the CASP framework appears to be a promising approach for work on the next generation of Internet signaling due to its apparent advantages. It resolves the major challenges faced by the current approaches, including complexity (by getting rid of multicast and QoS-caused overhead), security (by stronger security for signaling due to a separation of discovery from signaling messages), mobility (by introducing a unique session ID, Branch_ID and effective general route change handling) and extensibility (the modularity of CASP allows easier extending to other signaling applications and discovery mechanisms). The general approach in CASP makes a large number of potential signaling applications possible, including QoS resource reservation, label distribution for MPLS networks, measurement gathering, code distribution for active networks, etc.

However, it should also be noted that CASP is still not a perfect fit to meet all needs, especially with regard to the trade-off between efficiency and extensibility/flexibility. Additionally, the details about security, route change and mobility support in CASP still need to undergo further investigations. These activities are currently ongoing in the IETF NSIS working group. Meanwhile we are studying performance and issues pertaining to complexity and robustness with these protocols through

Table 1
A comparison of different IP signaling approaches

Feature	ST-II [2]	RSVP [3,4]	M-RSVP [17–19]	Two-level RSVP [21–23]	CASP [26,32,33]
State maintenance	Hard state	Soft state	Soft state	Soft state	Soft state
Signaling approach	Filter-based	Filter-based	Filter-based	Filter-based	Discovery-based
Signaling scenarios	Focused on end-to-end (e2e) signaling	Focused on e2e signaling; proxy mode proposed in [44]	Focused on e2e signaling; proxy mode proposed in [19]	Not specified	Support large number of signaling scenarios (e.g., end-to-middle, middle-to-middle, e2e, middle-to-end)
Initiation	Sender-initiated	Receiver-initiated	Receiver-initiated	Receiver-initiated	Sender- and receiver-initiated
Way of next hop discovery	Message interception based on a specialized network layer protocol (ST-II)	Message interception based on a Router Alert Option	Same as RSVP	Same as RSVP	Various discovery mechanisms possible (e.g., Scout, DNS- or DHCP-based, or extending routing protocols)
Reliability for signaling traffic	Yes, by hop-by-hop acknowledgment with retransmission	No; [9] adds hop-by-hop acknowledgment with retransmission	No	No	Yes, provided by the underlying TCP or SCTP on a hop-by-hop basis
Signaling message size	MTU-limited	MTU-limited	MTU-limited	Requires fragmentation when exceeding MTU	No limitation due to fragmentation support of TCP and SCTP
Complexity	Support 1:n multicast (built on unicast routing); hard state: complex	Support m:n multicast, reservation failure handling: complex	Inherited from RSVP, plus mobility handling	Unicast, additional complexity in handling fragmentation	Mostly for unicast; some complexity in handling of different transport protocols
Security	Not specified	Mostly for (esp. IntServ) QoS signaling; dynamic key management not considered; other limitations analyzed in [37]	Not specified	Not specified	Use of IPsec/TLS for stronger messaging layer security; additional security mechanisms at client layer
Mobility	Not supported	Not supported	Supported	Not supported	Supported
Extensibility	Designed for QoS signaling	Mostly designed for QoS signaling, not well-suited for other signaling applications	Mostly for QoS signaling	Flexible in adding new signaling applications	Flexible in adding new signaling applications and discovery mechanisms

rigorous analysis of the protocol behaviors and prototype implementations. First open implementation of CASP has been released and its initial design and evaluation has been reported in [43].

Nevertheless, it is very likely that the needs for Internet signaling will continue to move on after a set of new Internet protocols are standardized (related proposed standards are expected to be published in late 2006). However, we believe the fundamental principles and experiences gained from CASP development will extend beyond the protocols themselves into the new protocol designs of the future.

Acknowledgements

Appendix A. Message Format for CASP

This Appendix provides an overview of the CASP, Scout protocol, CASP-QoS and CASP-FW/NAT message format. These messages and their constituent objects are defined in the same way as RSVP [4].

A.1. CASP message

A CASP message consists of a common header, followed by a body consisting of any number of TLV-encoded objects. The CASP header is defined as shown in Fig. 9.
Where,

- *Length* covers the length in bytes of the whole CASP message. This field does not exist for the SCTP as transport case.
- *Flags*: currently 4 bits are defined:

- *Reverse (R) bit* indicates that a node should route in the opposite direction to the data flow.
- *Tear-down (T) bit* indicates that this message tears down all CASP messaging and associated client state. If not set, the message establishes or refreshes messaging state.
- *Discovery (D) bit* requests that the node perform a new discovery operation. If not set, the old next-hop should be used if possible.
- *Unsecure (U) bit* indicates that the message has traversed a hop without channel security.
- *TTL* is decremented by each CASP hop. If reaching zero the message should be discarded.
- *Hop count* is incremented by each CASP hop.

Each CASP object consists of a CASP object header and its object content (aligned to a length of multiple of 4 bytes). The CASP object header is defined in Fig. 10, where the field *Class_Num* can be one of the following: *Source_address* (Class = 1), *Destination_ADDR* (Class=2), *SESSION_ID* (3), *Flow_ID* (4), and *Client_Protocol* (6). The field *C-Type* further specifies the class type for a particular object. For example, *C-Type* values 1 and 2 are used to distinguish IPv4 and IPv6 versions of an object such as *Flow_ID*, respectively. The default value is 1. For client protocols, the field *C-Type* identifies which exact client, e.g., Scout(C-Type = 0), CASP-QoS (C-Type = 1), CASP-FW/NAT (C-Type = 2).

The mandatory object *FLOW_ID* is currently defined as 5-tuple: sender and receiver IP addresses and ports, and protocol ID. This information is carried in messaging layer to properly route CASP messages when messaging state is not available or new discovery is needed. In addition, two optional objects are currently defined:

- *Branch_ID* object: to detect and recover signaling state in mobility scenarios. See Section 4.3.
- *CASP_TIMER* object: to set the m-layer state timer, default to 30s if such an object is not included.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|                      Length (bytes)                           |
+---------------+---------------+---------------+---------------+
|R T D U / / / /|     TTL       |   Hop count   |   Reserved    |
+---------------+---------------+---------------+---------------+
```

Fig. 9. CASP Message Header (TCP as transport protocol case).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|          Length (bytes)       |  Class-Num    |    C-Type     |
+---------------+---------------+---------------+---------------+
```

Fig. 10. CASP Object Header.

A.2. Scout protocol

Scout protocol messages are defined as follows:

```
<Scout Request Message>
  ::= <Common Header><Type='ScoutReq'>
     <CASP_SOURCE_ADDR>
     <SCOUT_COOKIE_NI>
<Scout Response Message>
  ::= <Common Header><Type='ScoutResp'>
     <SCOUT_COOKIE_NI>
     <SCOUT_COOKIE_ NR>
```

where, in CASP Common Header part, field *Class* = 6, *C-Type* = 0.

A.3. CASP-QoS protocol

A CASP-QoS message consists of a common header, followed by a body consisting of a variable number of TLV-encoded objects. The common header is defined in Fig. 11.

The fields in the common header are defined as follows:

- *Length* indicates the length of the whole CASP-QoS message in bytes;
- *Flags*: currently 3 flags are defined:
  – The Commit (C) bit indicates that the message receiver should send back a Commit message in the opposite direction,
  – The Removal (R) bit indicates that this message removes all CASP-QoS state (Reserve and Commit states, if any) for the CASP-QoS session. If not set, the message establishes or refreshes CASP-QoS state,
  – The Partial Reservation (P) bit requests that a partial reservation is acceptable. If not set, the reservation from the sender to the receiver should be tried if possible,

- *Type*: the CASP-QoS message type. Current valid types are:
  – Type 1: CASP-QoS Query Message,
  – Type 2: CASP-QoS Reserve Message,
  – Type 3: CASP-QoS Commit Message,
  – Type 4: CASP-QoS Release Message,
  – Type 5: CASP-QoS Response Message;
- *Sequence number* indicates the reservation sequence information for its communicating CASP-QoS neighbor. It is of local relevance and used to refer to a reservation state together with Session_ID, for the purpose of avoiding possible message misinterpretation e.g., due to disordering.

Currently the following CASP-QoS objects are defined:

- *Flow_ID*: CASP-QoS can carry its own Flow_ID which contain necessary information about which flow which should receive a particular QoS treatment;
- *ERROR_NODE* indicates the IP address where a reservation failed;
- *VERSION* indicates which type of flowspec is used. This is used to easily detect a new flow-specs, rather than including the flowspec itself. Currently Version = 0 indicates IntServ Controlled-Load flowspec;
- *NEXT* indicates what is the desired next message responding to current one;
- *FLOWSPEC* indicates the flowspec (see Fig. 12) that the application would accept/desire. There can be one or two flowspec(s).

A.4. CASP-FW/NAT protocol

Currently, CASP-FW/NAT defines the following message types:

- Type 1: CASP-FW/NAT Path Message,
- Type 2: CASP-FW/NAT Query Message,
- Type 2: CASP-FW/NAT Create Message,
- Type 3: CASP-FW/NAT Release Message,
- Type 4: CASP-FW/NAT Response Message,
- Type 5: CASP-FW/NAT Trigger Message.

Each CASP-FW/NAT message consists of a common header (as shown in Fig. 13, similar to CASP-QoS common header, where the *flags* field

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|         Length (bytes)        |   Class = 6   |   C-Type = 1  |
+---------------+---------------+---------------+---------------+
|    Flags      |     Type      |        sequence number        |
+---------------+---------------+---------------+---------------+
```

Fig. 11. CASP-QoS Common Header.

```
   31           24 23          16 15           8 7            0
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 1 | 0 (a) |    reserved       |             7 (b)             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 2 |   5 (c)    |0| reserved    |             6 (d)             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 3 |   127 (e)    |    0 (f)    |             5 (g)             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 4 | Token Bucket Rate [r] (32-bit IEEE floating point number) |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 5 | Token Bucket Size [b] (32-bit IEEE floating point number) |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 6 | Peak Data Rate [p] (32-bit IEEE floating point number)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 7 | Minimum Policed Unit [m] (32-bit integer)                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 8 | Maximum Packet Size [M]  (32-bit integer)                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   (a) - Message format version number (0)
   (b) - Overall length (7 words not including header)
   (c) - Service header, service number 5 (Controlled-Load)
   (d) - Length of controlled-load data, 6 words not including
         per-service header
   (e) - Parameter ID, parameter 127 (Token Bucket TSpec)
   (f) - Parameter 127 flags (none set)
   (g) - Parameter 127 length, 5 words not including per-service
         header
```

Fig. 12. Current supported flowspec: IntServ CLS.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|         Length (bytes)        |  class = 6    | C-Type = 2    |
+---------------+---------------+---------------+---------------+
|   flags       |    Type       |        sequence number        |
+---------------+---------------+---------------+---------------+
```
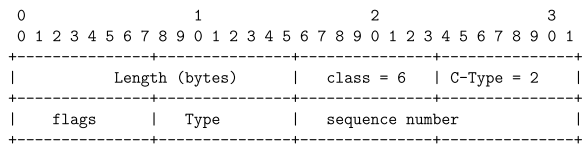
Fig. 13. CASP-FW/NAT Common Header.

is reserved) and a variable number of CASP-FW/NAT objects.

The following CASP-FW/NAT objects are currently being investigated:

- *Logging Action* indicates which packet filter(s) want to have logging defined.
- *ApplicationID* contains an identifier to provide more information about the data for which the policy rule is installed. Application-level firewalls and firewalls with stateful inspection are able to use this information.
- *NEXT* indicates the next request that the signaling message receiver should generate if the incoming message was successfully processed.
- *Authorization Token* to be specified in detail.
- *CMS Credential Object* allows user specific cryptographic credentials to be transmitted to specific CASP peers (or networks) along the path.
- *Age* object is used to quickly determine whether any of the NSLP object has changed (for example packet filter), to avoid a bit-by-bit comparison. The Age object might be useful for messages which refresh established state information only.

References

[1] T. Russell, Signaling System #7, second ed., McGraw-Hill, 1998.

[2] L. Delgrossi, L. Berger, Internet Stream Protocol Version 2 (ST2) Protocol Specification – Version ST2+, RFC 1819 (August 1995).

[3] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, RSVP: A New Resource ReSerVation Protocol, IEEE Network 7 (5) (1993) 8–18.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification, RFC 2205 (September 1997).

[5] IETF Next Steps In Signaling (NSIS) Working Group [Online]. Available from: <http://www.ietf.org/html.charters/nsis-charter.html>.

[6] J. Wroclawski, The Use of RSVP with IETF Integrated Services, RFC 2210 (September 1997).

[7] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, RSVP-TE: Extensions to RSVP for LSP Tunnels, RFC 3209 (December 2001).

[8] F. Baker, C. Iturralde, F.L. Faucheur, B. Davie, Aggregation of RSVP for IPv4 and IPv6 Reservations, RFC 3175 (September 2001).

[9] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, S. Molendini, RSVP Refresh Overhead Reduction Extensions, RFC 2961 (April 2001).

[10] F. Baker, B. Lindell, M. Talwar, RSVP Cryptographic Authentication, RFC 2747 (January 2000).

[11] S. Yadav, R. Yavatkar, R. Pabbati, P. Ford, T.E. Moore, S. Herzog, R. Hess, Identity Representation for RSVP, RFC 3182 (October 2001).

[12] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry, The COPS (Common Open Policy Service) Protocol, RFC 2748 (January 2000).

[13] C. Rigney, S. Willens, A. Rubens, W. Simpson, Remote Authentication Dial In User Service (RADIUS), RFC 2865 (June 2000).

[14] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, Diameter Base Protocol, RFC 2588 (September 2003).

[15] C. Perkins, IP Mobility Support for IPv4, RFC 3344 (August 2002).

[16] D. Johnson, C. Perkins, J. Arkko, Mobility Support in IPv6, RFC 3775 (June 2004).

[17] A. Talukdar, B. Badrinath, A. Acharya, MRSVP: a Resource Reservation Protocol for an Integrated Services Network with Mobile Hosts, Wireless Networks 7 (1) (2001) 5–19.

[18] C. Shen, W. Seah, et al., Mobility Extensions to RSVP in an RSVP-Mobile IPv6 Framework, Internet draft (Jule 2002) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-shen-nsis-rsvp-mobileipv6/>.

[19] J. Manner, K. Raatikainen, Localized QoS Management for Multimedia Applications in Wireless Access Networks, in: Proc. IASTED IMSA 2003 (January 2003).

[20] M. Thomas, Analysis of Mobile IP and RSVP Interactions, Internet draft, (Oct. 2002) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-thomas-nsis-rsvp-analysis/>.

[21] B. Braden, B. Lindell, A Two-level Architecture for Internet Signaling, Internet draft (Nov. 2001) [Online]. Available from: <http://www.isi.edu/rsvp/DOCUMENTS/draft-braden-2level-signaling-01.txt>.

[22] M. Shore, The NSIS Transport Layer Protocol (NTLP), Internet draft (May 2003) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-shore-ntlp/>.

[23] M. Brunner, R. Greco, Towards RSVP Version 2, Internet draft (Oct. 2002) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-brunner-nsis-rsvp2/>.

[24] L. Westberg, G. Karagiannis, et al., A Proposal for RSVPv2, Internet draft (Oct. 2002) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-westberg-proposal-for-rsvpv2/>.

[25] J. Manner, X. Fu, Analysis of Existing Quality-of-Service Signaling Protocols, RFC 4094 (May 2005).

[26] H. Schulzrinne, H. Tschofenig, X. Fu, A. McDonald, CASP – Cross-Application Signaling Protocol, Internet draft (March 2003). Also as Technical Report No. IFI-TB-2003-01, Institute for Informatics, University of Göttingen, Germany, March 2003.

[27] H. Schulzrinne, X. Fu, C. Pampu, C. Kappler, Design of CASP – a Technology Independent Lightweight Signaling Protocol, in: Proc. of IPS'03, Salzburg, Austria (February 2003).

[28] E. Guttman, C.E. Perkins, J. Veizades, M. Day, Service Location Protocol, Version 2, RFC 2608 (June 1999).

[29] R.J. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, Stream Control Transmission Protocol, RFC 2960 (October 2000).

[30] C. Partridge, A. Jackson, IP Router Alert Option, RFC 2113 (February 1997).

[31] C. Partridge, A. Jackson, IPv6 Router Alert Option, RFC 2711 (October 1999).

[32] H. Schulzrinne, H. Tschofenig, X. Fu, J. Eisl, A Quality-of-Service Resource Allocation Client for CASP, Internet draft (March 2003). Also as Technical Report No. IFI-TB-2005-07, Institute for Informatics, University of Göttingen, Germany, November 2005.

[33] H. Tschofenig, H. Schulzrinne, C. Aoun, A Firewall/NAT Traversal Client for CASP, Internet draft (Mar. 2003) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-tschofenig-nsis-casp-midcom/>.

[34] X. Fu, H. Schulzrinne, H. Tschofenig, Mobility Support for Next-Generation Internet Signaling Protocols, in: Proc. of VTC'03-Fall, Orlando, Florida (October 2003).

[35] T. Dierks, C. Allen, The TLS Protocol Version 1.0, RFC 2246 (January 1999).

[36] S.A. Kent, R. Atkinson, Security Architecture for the Internet Protocol, RFC 2401 (November 1998).

[37] H. Tschofenig, R. Graveman, RSVP Security Properties, Internet draft (May 2005) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-ietf-nsis-rsvp-sec-properties/>.

[38] D. Harkins, D. Carrel, The Internet Key Exchange (IKE), RFC 2409 (November 1998).

[39] C. Kaufman, Internet Key Exchange (IKEv2) Protocol, Internet draft (Feb. 2003) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-ietf-ipsec-ikev2/>.

[40] H. Tschofenig, Path-Coupled NAT/Firewall Signaling Security Problems, Internet draft (Jul. 2004) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-tschofenig-nsis-natfw-security-problems/>.

[41] B. Aboba, L. Blunk, et al., Extensible Authentication Protocol (EAP), RFC 3748 (June 2004).

[42] R. Housley, Cryptographic Message Syntax, RFC 2630 (June 1999).

[43] X. Fu, D. Hogrefe, S. Willert, Implementation and Evaluation of the Cross-Application Signaling Protocol (CASP), in: Proc. of ICNP 2004, Berlin, Germany (October 2004).

[44] Y. Bernet, N. Elfassy, S. Gai, D. Dutt, RSVP Proxy, Internet draft (Mar. 2002) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-ietf-rsvp-proxy/>.

[45] D. Mitzel, D. Estrin, S. Shenker, L. Zhang, An Architectural Comparison of ST-II and RSVP, in: Proc. of INFOCOM 1994 (June 1994).

[46] H. Tschofenig, M. Buechli, S. Van den Bosch, H. Schulzrinne, NSIS Authentication, Authorization and Accounting Issues, Internet Draft (March 2003) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-tschofenig-nsis-aaa-issues/>.

[47] M. Karsten, J. Schmitt, R. Steinmetz, Implementation and Evaluation of the KOM RSVP Engine, in: Proc. of INFOCOM 2001 (April 2001).

[48] P. Pan, H. Schulzrinne, Processing Overhead Studies in Resource reservation Protocols, in: Proc. of 17th International Teletraffic Congress (ITC), Salvador, Brazil (September 2001).

[49] European Telecommunications Standards Institute, Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) – Open Settlement Protocol (OSP) for Inter-domain Pricing, Authorization, and Usage Exchange, Technical Specification 101 321, Version 2.1.0 (May 2000).

[50] L. Hamer, B. Gage, H. Shieh, Framework for Session Set-up with Media Authorization, RFC 3521 (April 2003).

[51] E. Maler, R. Philpott, P. Mishra, Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1 (September 2003).

[52] E. Maler, R. Philpott, P. Mishra, Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1 (September 2003).

[53] H. Tschofenig, H. Schulzrinne, R. Hancock, A. McDonald, X. Fu, Security Implications of the Session Identifier, Internet Draft (June 2003). Also as Technical Report No. IFI-TB-2005-08, Institute for Informatics, University of Göttingen, Germany, November 2005.

[54] F. Alfano, P. McCann, H. Tschofenig H. Tschofenig, Diameter Quality of Service Application, Internet Draft (Oct. 2005) [Online]. Available from: <http://ietfreport.isoc.org/idref/draft-alfano-aaa-qosprot/>.

[55] H. Tschofenig, D. Kroeselberg, Security Threats for Next Steps in Signaling (NSIS), RFC 4081 (June 2005).

Xiaoming Fu received a Ph.D. degree in Computer Science from Tsinghua University, China in 2000. During 2000–2002, He was member of research staff at Telecommunication Networks Group, Technical University Berlin, Germany. Since September 2002, he is an assistant professor at the Institute for Informatics, University of Göttingen, Germany, leading a research team working on network architectures, protocol design, validation and performance evaluation, Internet QoS and signaling, security, mobile networks beyond 3G, overlay and multimedia networking. In these aspects he has actively contributed to EU projects ENABLE, DIADALOS II, VIDIOS and IPT, as

well as in other international collaborations. He is an Expert for ETSI STFs on IPv6 Interoperability Testing, a member of the IEEE and the ACM, and has served on the technical program committees of IEEE ICDCS, GLOBECOM and ICC.

including NSIS, GEOPRIV, EAP, MIP6, TLS, IPsec and PANA, and has contributed to EU funded projects, such as SHAMAN, Ambient Networks and ENABLE.

Hannes Tschofenig received a Diploma degree from the University of Klagenfurt, Austria. He joined Siemens Corporate Technology in 2001 where he is currently a research scientist. His research focuses on security for mobile communications. He is chair of the IETF Emergency Context Resolution with Internet Technologies (ECRIT) working group and Secretary of the NSIS working group. He is a co-author of RFC 3726 "Requirements for Signaling Protocols", RFC 4081 "Security Threats for Next Steps in Signaling (NSIS)", RFC 4279 "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", "RSVP Security Properties" and a number of papers and Internet drafts. He is active in various IETF working groups,

Dieter Hogrefe received his Diploma degree and Ph.D. from the University of Hannover, Germany. His research activities are directed towards Computer Networks and Protocol Engineering. In these fields he has published several books and numerous papers on Internet technology, analysis, simulation and testing of formally specified communication systems. After years of research positions in Siemens, he held professorships at the Universities of Dortmund, Berne and Lübeck. Since 2002, He is Professor for Telematics at the University of Göttingen. He represents the IITB (Fraunhofer Institute for Information and Data Processing) in the European Telecommunication Standards Institute, ETSI, where he is chairman of the Technical Committee Methods for Testing and Specification.