

Design of a generic services signaling framework

Jaiwant Mulik*
jmulik@temple.edu

Olivier Marcé†
Olivier.Marce@alcatel.fr

1 Introduction

In this document we outline the design of the *Services Signaling Framework* (SSF), which deals with the signaling component of network services. SSF provides a framework that can be used to implement functionality such as installation and maintenance of services and accounting. For purposes of this document, a network service is a value-added feature provided to the user by instantiation of a service at a host in the network. Due to space constraints this abstract focuses on the installation operation.

There are several existing attempts that are based on the signaling requirements for services within a *fixed* domain. The MIDCOM signaling protocol requirements [6] defines the requirements for signaling between a service manager and NAT and Firewall services located at middleboxes (inter-node signaling). The ForCES signaling protocol requirements [3] lists the requirements for signaling between the forwarding and control elements of routers (intra-node signaling). Beagle is the signaling protocol used as part of the Darwin [2] resource management system. Beagle can be used to install *delegates* at network nodes to perform sophisticated resource management.

2 Requirements for a generic signaling framework

The requirements of SSF are designed to provide the signaling protocol implementer with commonly required functionality while providing enough flexibility to implement service specific behavior for any domain. The following lists the considered requirements. Non-functional requirements such as security, extensibility, scalability and footprint size, are not considered in this talk.

1. **Query:** SSF should provide the ability to query the service initiation capability of a node. For example, SSF should be able to provide information about a hardware router that is only extensible by adding new hardware. In case a general purpose device is being used as a router, SSF should provide information about both hardware and software capabilities of

that node. SSF should provide information about the execution environment of the node being signaled. This information is useful when multiple versions of the service code are available to support the service on more than one execution environment. A signaling protocol should be able to query if SSF can provide *liveness* status of service instances (see requirement 3 for an exception).

2. **Install:** The SSF should be able to install object code on a network node. The signaling protocol using SSF could provide either the object code or a reference (e.g. a URL) to the object code. SSF should be capable of installing multiple instances of the same service on a node and to distinguish between them. SSF should provide an interface that allows the signaling protocol to define packet flows based on layer 3 or layer 7 data. Finally, SSF must provide a mechanism for the signaling protocol to associate packet flows with service instances. Since it is possible that a service might alter the contents of a packet leading to the packet now matching other SSF filters, SSF should allow for the signaling protocol to decide if the packet should be retried for matching filters or not. If the signaled host allows it, a way to specify the order the filters are tested for matching should also be provided by SSF.
3. **Maintenance:** SSF should be able to report the *liveness* of a service instance at a network node. In case of hardware only network nodes though, *liveness* reports might not be possible.
4. **Message passing:** It is expected that the signaling protocol will use this functionality to pass configuration messages to service instances and to carry the response from service instances. SSF should provide the ability to asynchronously transfer data chunks (e.g. key-value pairs) provided by the signaling protocol between the signaling host and the signaled host.
5. **Accounting:** SSF should provide basic accounting functionality related to each service instance. At the minimum, accounting should include the number of packets delivered to each service instance.

3 Architecture of SSF

Only the components used for service installation are described here. For each service instance installed at a node and for each packet flow definition, SSF provides a unique

*Computer and Information Sciences Department, Temple University, Philadelphia, USA

†Alcatel R&I Dept, Marcoussis, France

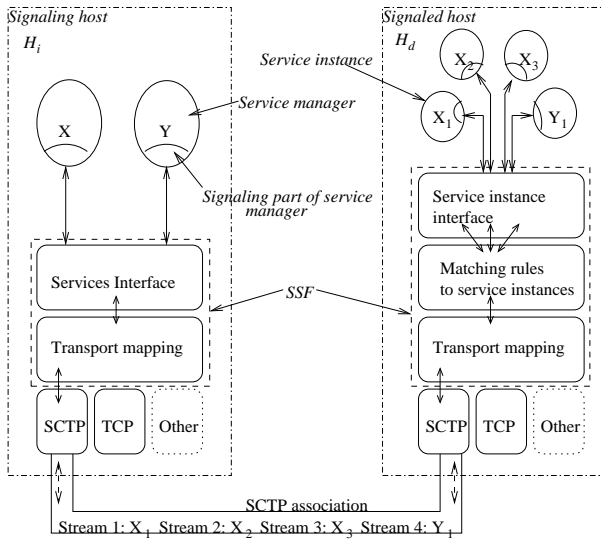


Figure 1: Components of SSF

identifier to the signaling protocol. Using these identifiers the signaling protocol can associate packets flows with service instances. The unique identifiers can also be used for maintenance and message passing. At the network host that implements services, SSF maintains a list of filters. Each filter defines a packet flow and is associated with service instances that have been registered to handle packets matching that flow. Some services need to have one instance per flow in a way that cannot be defined at the initial phase since the set of flows is not known *a priori*. For example, when the service provides transcoding functionality to a flow, each users' flow needs to be associated with a unique instance. In this case since the value of the users source port is not known before the flows start, it is not possible to statically instantiate services. One possible solution could be to have a gatekeeper mechanism to which the user declares the flow characteristics, then a new instance of the service is instantiated and associated to the filter corresponding to this flow, by the signaling host. The other solution, which is preferred in the context of SSF, is to have a *meta-service* that is associated to a filter matching all the users' flows to deal with, and which has the ability to instantiate a new service instance and a new filter when it identifies a new flow coming up. This meta-service is a kind of delegate of the signaling host, installing the instances and filters when needed.

By providing a common interface to define filter rules, SSF removes the burden of having each signaling protocol define its own filtering mechanism and then having multiple filtering mechanism on a given network node. Figure 1 illustrates the major components of SSF.

Since, architecturally SSF sits between the service signaling protocol and the transport layer, SSF can transparently utilize innovative transport protocols such as SCTP [5]. SSF can use the multiple streams feature of SCTP to reduce communication cost compared to each signaling protocol having its own transport layer connection. The SSF endpoints can communicate using a single SCTP association.

4 Use case

The following describes a typical use case of SSF. Consider the case where a host must provide multimedia flow coding adaptation: depending on the capacity of the link toward the receiver, the flow is transcoded if necessary to the best suited codec. SSF is first used by the signaling host H_i to query the signaled host H_d . After check of the compatibility between the needed and available resources, H_i can install on H_d , the transcoding service X . H_i then sends using SSF, a request for three instances of X , X_1 , X_2 and X_3 . SSF returns an identifier for each of the instance to H_i allowing it to send the correct parameters to the three instances. In this case, the parameter is the bit rate of the codec. The identifier can also be used for maintenance and accounting purposes. For each of the instances, H_i defines the filters allowing H_d to distinguish the flows toward receivers R_1 , R_2 , and R_3 and it associates the flows to X_1 , X_2 and X_3 .

5 Conclusion

The design of SSF aims to define a single framework for different signaling protocols in different domains. The goal of SSF is to ease the design of these protocols by providing common functionalities. The focus of SSF is on the installation, instantiation and association with flows of the services. It does not aim to provide a generic all-in-one signaling protocol, but rather to provide a common building block used by domain specific protocols designers. Hence, it should not be compared to other solutions for services signaling such as SNMP [1] and COPS [4]. SSF aims to fulfill functional and flow related requirements expressed for protocol like MIDCOM or ForCES. The complete set of features and the expected benefits of SSF, as well as its applicability to these and other protocols, will be further discussed during the talk.

References

- [1] M. Barnes. Middlebox Communications (MID-COM) Protocol Evaluation, June 2002. Internet-draft, <draft-ietf-midcom-protocol-eval-02.txt>.
- [2] P. Chandra et al. Darwin: Customizable Resource Management for Value-Added Network Services. *IEEE Network*, 15(1):22–35., 2001.
- [3] H. Khosravi and T. Anderson. Requirements for Separation of IP Control and Forwarding, July 2002. Internet-draft, <draft-ietf-forces-requirements-06.txt>.
- [4] Yacine El Mghazli and Olivier Marce. COPS client-type for Active Networks, July 2002. Internet-draft, <draft-yacine-cops-an-01.txt>.
- [5] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol, October 2002. IETF RFC 2960.
- [6] R P Swale, P A Mart, P Sijben, Scott Brim, and Melinda Shore. Middlebox Communications (midcom) Protocol Requirements, November 2001. Internet-draft, <draft-ietf-midcom-requirements-05.txt>.