ACHIEVING SECURE SERVICE SHARING OVER IP NETWORKS

David Lai  Zhongwei Zhang  Chong Shen
lai@usq.edu.au  zhongwei@usq.edu.au  shen@usq.edu.au

University of Southern Queensland
Toowoomba, Queensland 4300, Australia

Abstract: No matter how many and how comprehensive the services a network can provide, in order to satisfy the diverse requirement of services, networks should share services among themselves. For secure service sharing on IP networks, the authenticity of users and the scalability of participating networks are always two major issues among others. Service Network Graph (SNG) was proposed to address the problems of cross network authentication and scalability, which usually occur in a dynamic aggregations of heterogeneous networks.

Our SNG approach is based on Authentication Propagation and Service Paths. Authentication Propagation is a process of relaying authentication results from the authenticating network to the service providing network. Within an SNG, networks delegate authentication duties to some other networks which gather all authentication and service information and return the authentication result to the user. A Service Path is designed to hold all the authentication delegation information from the user’s home network to the service providing network. An example of Service Path in a network, $N_x$, looks like: $< F : /N_x/N_y/N_z/S_z/Service_z > < 4 >$ where the second field, $/N_x/N_y/N_z/S_z/Service_z$, stands for the NetworkPath of a service, $Service_z$, which is provided by a server $S_z$, in a network $N_z$. We can work out the routes for the authentication and service information from the NetworkPath as (1) from $N_x$ to $N_y$ if it does not end at $N_z$; (2) from $N_y$ to $N_z$ if it does not originate from $N_z$; (3) from $N_z$ to $N_y$ if it does not end at $N_x$; and (4) from $N_y$ to $N_x$ if it does not originate from $N_z$.

These routes can be represented in the 4-tuples form: $(< Net_{ori} >, < Net_{from} >, < Net_{dst} >)$. To differentiate route (2) and route (4), $< Net_{ori} >$ is used. Route (1) can be expressed as $(Net_x, Net_x, Net_y, Net_z)$ using the 4-tuple notation. Obviously, it is not efficient to extract the routes from incoming Service Paths each and every time. Besides, the NetworkPath field may contain a substantial number of networks. Hence reusing the routes could improve the efficiency. The 4-tuple notation facilitates the reuse of routing information.

In this paper, we devise a 4-tuple (ATR tuple) representation of authentication and service information routes. The ATR tuple representation is shown to be an alternative representation of SNG other than the graphical representation. We also explore how the ATR tuple representation can be applied to facilitate the authentication propagation process. A set of experiments on network simulator, OMNeT++, have been carried out to illustrate the application of SNG with ATR tuples to IP networks. The preliminary simulation results show that the ATR tuple representation greatly simplifies the implementation of the SNG authentication routing algorithm, and secure service sharing can be achieved as well.

Key words: Service Network Graph, service sharing, authentication delegation, authentication propagation, service path, routing tuple.
1 Introduction

A network has limited resources and can provide only a limited number of services. Administrative constraints also limit the categories of service available to users. Users may request services that are beyond the domain of service for the network and can only be honored with shared services provided by other networks. To share a service, authenticity of users and scalability of participating networks are two major issues related to security.

Service sharing involves cross network authentication. For security and privacy reasons, authentication servers keep their authentication information repository private. Furthermore, different networks always use different authentication schemes. One set of authentication information can hardly fulfill the requirements of all authentication schemes.

To enable cross network authentication, an AS needs to have access to the rightful authentication information or entrust other AS to perform the authentication task. Administration and maintenance of a global repository of authentication information render such repository not practical. Besides sharing of the authentication information may breach the privacy of users.

Alternatives to a global repository of authentication information include the use of X.509 [1] digital certificates and the establishment of trust [2, 3, 4]. Both approaches require all participants involved in the authentication process to accept a single or a group of third parties. The use of X.509 digital certificates in cross network authentication requires a commonly trusted Certification Authority which binds the user identity with the public key in an X.509 digital certificate and all authentication servers must trust the certificates issued by the Certificate Authority. In other words, all of them have to adopt the use of X.509 digital certificates as their authentication scheme and trust the same Certificate Authority in performing the duty of binding user identities and the corresponding public keys before cross network authentication is possible. In addition, they have to agree on a common set of policies regarding the validity and revocation of X.509 digital certificates [5, 6]. In particular, an authentication server prescribes a list of trust agents which are authorized to issue trust tokens [7, 8, 9, 10]. The aggregated result of a number of trust tokens is used to determine the authenticity and possible authorization of the user [11]. The authentication server then distributes the authentication duty to the prescribed trust agents while making its own final authentication decision. A common set of overlapped trust agents is required for cross network authentication.

Service Network Graph (SNG) [12, 13, 14] is a new way to handle cross network authentication, which enables service sharing among dynamic aggregation of autonomous networks. The authentication duty of the authentication server, AS\textsubscript{S}, in the service providing network, N\textsubscript{S}, is delegated to the authentication server, AS\textsubscript{H}, of the home network, N\textsubscript{H}, of an individual user. N\textsubscript{H} and AS\textsubscript{H} are referred to as the delegatee network and delegatee AS while N\textsubscript{S} and AS\textsubscript{S} are the delegator network and delegator AS. The delegator AS\textsubscript{H} performs authentication, bundles the result, a session key and the Service Path into an authentication token (AToken), and passes the AToken to the server via AS\textsubscript{S}. The server accepts the authentication result from the delegatee AS\textsubscript{H} the same way as that from the delegator AS\textsubscript{S} and appends the necessary service information to the AToken which is handed back to the user via AS\textsubscript{S} and AS\textsubscript{H}. With authentication delegation, a server can authenticate any user from a delegatee network and users from a delegatee network can access shared services provided by delegator networks. Participating networks of an SNG can maintain their own authentication schemes, keep their own private authentication information repository private and still can perform cross network authentication. They can join or detach from SNG while maintaining their autonomy.

When AToken is routed from a home network to a service providing network, routing information for the AToken can be extracted from the NetworkPath information in the
Service Path which is part of the AToken. In practice, the token routing information can be reused and more readily available if each AS can keep its own set of AToken routing information. To facilitate routing of ATokens, we propose to use a 4-tuple representation for storing the token routing information (ATR tuple). We formulated the ATR tuples in such a way that they can be used to hold routes for forward or backward ATR routing. Each AS is required to hold only the ATR tuples related to itself.

The rest of the paper is structured as follows: Section 2 is an overview of SNG. In particular, Authentication Delegation, Service Paths and Authentication Propagation are briefly explained. In Section 3 we investigate the routing of ATokens and the ATR tuple representation. An example of deriving the ATR tuples from an SNG is also given. Lastly, we apply the ATR tuple representation in Section 4 in an implementation of SNG over IP networks. A set of experiments was carried out using ATR tuple implementation of the SNG and network simulator OMNeT++. The paper is ended with a conclusion and future works.

2 Overview of Service Network Graph

SNG is based on Service Paths, Authentication Delegation and Authentication Propagation. They are summarized in the sections below.

2.1 Authentication Delegation

In SNG, network \( N_x \) can extend its services by attaching to another network, say \( N_y \). On the other hand, network \( N_y \) shares its services by delegating its authentication duty to an attached network say \( N_x \). Hence a network can extend its service by attaching to another network as an authentication delegatee network and share its services as an authentication delegator network. A delegatee network can also be a delegator network as shown in Figure 1. \( N_4 \) is the delegator network of \( N_1 \) and is also the delegatee network of \( N_2 \). \( N_1 \) and \( N_3 \) are both delegator and delegatee network of each other.

A network delegates its authentication authority to another network when its authentication server (AS) generates and shares an Authentication Token key (ATK) with the delegatee network AS. It also informs the delegateee network AS of the services it provides, locally or shared from other networks in the form of Service Paths detailed in section 2.2. Local services are directly shared with the delegatee network and tagged with free or restricted to control further delegation. Shared services with free authentication delegation can further be shared indirectly with other delegatee networks.

When \( U_x \) requests a service provided by \( N_y \), \( AS_x \) in home network \( (N_x) \) of \( U_x \) authenticates \( U_x \) on behalf of \( N_y \). If the authentication is successful, \( AS_x \) generates a session key for the service request. The Service Path, session key and authentication result are bundled together to form an AToken.

2.2 Service Paths

When referring to a service, we have to specify from where we can access the service. As a service can be provided by different servers in more than one network, we have to specify the network path for the service. Other important attributes for services are the service cost and whether the service can be further shared or not. If a shared service can be shared with other networks, it is a free authentication delegation and a restricted authentication delegation means that the service should not be shared with other networks. So a typical Service Path looks like:
\[
< F : S_1 / S_{v4} > : < 4 >
\]
means service \( S_{v4} \) provided by local server \( S_1 \) with cost 4 units
is a service shared with free authentication delegation
and can be shared with other networks.
Authentication Propagation path is local authentication only.

\[
< R : N_1 / N_4 / N_2 / S_2 / S_{v123} > : < 25 >
\]
means service \( S_{v123} \) provided by server \( S_2 \) in \( N_2 \) with cost 25 units
is a service shared with restricted authentication delegation
and should not share with other networks.
Authentication Propagation path is \( N_1 \implies N_4 \implies N_2 \).

The Service Listing server (SLS) of a network provides a complete and updated list of services available including local and shared services.

### 2.3 Authentication Propagation

In order to explain Authentication Propagation, we will refer to Figure 2 assuming each network has an AS, a server \( S \), a user \( U \) and a Service Listing server SLS with the appropriate subscript to denote the network they belong to. We use \( P_{xy} \) to denote a service path in which service \( y \) is provided by network \( x \). The SNG depicted in Figure 2 has four participating networks. \( N_1 \) is attached to \( N_4 \) as indicated by the single headed arrow while \( N_1 \) and \( N_3 \) are attached to one another as indicated by the double headed arrow.

The Authentication Token propagation path is represented by dotted lines as shown in Figure 2.

We will use:

- **Send** \((< \text{from} >, < \text{to} >, < \text{msg} >)\)
  - to mean \(< \text{from} >\) send a message \(< \text{msg} >\) to \(< \text{to} >\)

- **Encrypt** \((< \text{msg} >, < \text{key} >)\)
  - to mean message \(< \text{msg} >\) is encrypted with \(< \text{key} >\)

A service request starts with a user, say \( U_1 \) asks for a Service List from \( SLS_1 \).

\[
\begin{align*}
\text{Send}(< U_1 >, < SLS_1 >, < \text{"Request for ServiceList"} >) \\
\text{Send}(< SLS_1 >, < U_1 >, < P_{1a}, P_{1b}, P_{2a}, P_{2b} >)
\end{align*}
\]

\( U_1 \) makes his choice, say \( P_{2a} \) and sends his authentication information along with \( P_{2a} \) to the AS1.

\[
\text{Send}(< U_1 >, < AS_1 >, < P_{2a}, "AuthenInfo" >)
\]

When \( U_1 \) is authenticated, \( AS_1 \) generates a session key \( K_{1u} \) for the service request. The session key is bundled with other authentication information and the Service Path as an authentication token. The authentication token is propagated to \( AS_4 \) and finally reached \( AS_2 \). Note that \( ATK_{4a} \) is the ATK between \( AS_1 \) and \( AS_4 \), and \( ATK_{2a} \) is the ATK between \( AS_1 \) and \( AS_2 \).

\[
\begin{align*}
\text{Send}(< AS_1 >, < AS_4 >, < P_{2a}, Encrypt(< K_{1u} >/, < ATK_{4a} >/) >) \\
\text{Send}(< AS_4 >, < AS_2 >, < P_{2a}, Encrypt(< K_{1u} >/, < ATK_{2a} >/) >)
\end{align*}
\]

\( AS_2 \) then passes the session key to the service providing server which returns the service information to \( AS_2 \). The messages are encrypted with the server key \( K_{2a} \).
Send($AS_2$, $S_2$, $P_{2a}$, encrypt($K_{1u}$, $K_{2s}$))
Send($S_2$, $AS_2$, $P_{2a}$, encrypt($"ServiceInfo"$, $K_{2s}$))

The service information follows a reversed path specified in $P_{2a}$ and arrives at $AS_1$.
Send($AS_2$, $AS_1$, $P_{2a}$, encrypt($"ServiceInfo"$, $ATK_{2a}$))
Send($AS_1$, $AS_1$, $P_{2a}$, encrypt($"ServiceInfo"$, $ATK_{4a}$))

The authentication propagation process is completed when $AS_1$ passes the service information and the session key $K_{1u}$ to $U_1$.
Send($AS_1$, $U_1$, $P_{2a}$, $K_{1u}$, "ServiceInfo")

$U_1$ can now communicate with $S_2$ and can access service directly. Note that the security of passing authentication information and session key from $AS_1$ to $U_1$ depends on the authentication and session key generation schemes adopted. One appropriate candidate is Dynamic Password and the associated key exchange scheme [15] which performs authentication and session key generation at the same time.

$AS$ of a network in SNG holds a service list from which it will derive the forward and backward routes for the AToken. In the next section, we will discuss how to represent the AToken routes in an SNG to optimized the performance for Authentication Token routing.

3 Service Network Graph Representation

The Service Network Graph is a logical representation of the authentication delegation relationships. A graphical representation is best suited for a human administrator. In practice, we must represent the service network graph in a form which best suits our purpose - routing of the authentication tokens and optimizing the Service paths.

To understand how an SNG can be represented using AToken routes, we will start with the routing mechanism of Authentication Tokens.

3.1 Authentication Token Routing

A Service Path itself provides enough information to route the Authentication Token to the destination. As an example, in Figure 2, consider a service path:

$$< F : /N_3/N_1/N_4/S_4/Sv_4 > : < 2 > \quad (1)$$

When $AS_1$ gets this service path, it will parse the NetworkPath field of the service path to get the routing information. As $AS_1$ is located in $N_1$, it will forward the AToken to $N_4$ if the AToken comes from $N_3$ or to $N_3$ if the AToken comes from $N_4$.

Since the AToken routing information can be reused for other service requests with the same service path, extracting the forward and backward AToken routing information with each and every AToken is not efficient. At the same time, the NetworkPath field of a service path may contain substantial number of networks. Hence storing the AToken routing information for reuse improves performance.

3.2 Tuple Representation

To reuse any AToken routing information, we must store the necessary information to determine the appropriate route. Referring to the SNG shown in Figure 2, consider the pair of Service Paths:

$$< F : /N_2/N_1/N_4/S_4/Sv_4 > : < 4 > \quad (2)$$
Service path 2 tells AS₁ to forward the AToken to N₄ while service path 3 instructs AS₁ to forward the AToken to N₅. The two AToken routes are different and the choice depends on the destination network which are N₃ and N₅. Note that the destination network may not be the same as the network an AToken is routed to. For instance, \(< F : /N₂/N₁/N₃/S₃/Sv₃ >:< 4 >\) informs AS₁ to forward the AToken to N₃ and the destination network is N₅. Both service path uses the same backward AToken route: pass AToken from N₁ to N₂.

Consider another pair of Service Paths:

\(< F : /N₂/N₁/N₄/S₄/Sv₄ >:< 4 >\) \hspace{1cm} (4)
\(< F : /N₃/N₁/N₄/S₄/Sv₄ >:< 4 >\) \hspace{1cm} (5)

Both Service Paths informs AS₁ to forward the ATokens to N₄. Service path 4 tells AS₁ to route the AToken to N₂ while service path 5 instructs AS₁ to route the AToken to N₃ when the ATokens return from N₄. Again, the two AToken routes are different and can be distinguished one from another by looking at the origin network.

Hence to hold the necessary information for AToken routing, we use 4-tuples (ATR tuples) to hold not only the network an AToken comes from \((N_{from})\) and goes to \((N_{to})\), but also the origin network \((N_{ori})\) and destination network \((N_{dest})\):

\(< N_{ori} >, < N_{from} >, < N_{to} >, < N_{dest} >\)

We now derive ATR tuples when from an SNG and explain how an SNG follows from an ATR representation. The graphical representation and the ATR tuple representation are shown to be equivalent.

### 3.3 Deriving ATR Tuples from an SNG

As discussed in Section 2, the following rules are used for building up an SNG:

**SNG Building Rule 1** When network N₁ attaches to network N₂ in an SNG, N₂ will pass its service list to N₁, assuming all services have free authentication delegation.

**SNG Building Rule 2** When network N₁ attaches to network N₂ in an SNG, it will pass the service list it gets from N₂ to all its delegatee networks. And in so doing, all delegatee networks can access all local or shared services provided by N₂.

**SNG Building Rule 3** When a network N₁ receives a Service Path from its delegator network N₂ in an SNG, N₁ will pass the AToken routes (ATR tuples) it worked out from the Service Path back to the appropriate delegator networks.

We will build an SNG such as the one shown in Figure 2. Suppose the SNG was built in the following five stages of attachment:

1. N₄ attaches to N₂.
2. N₂ attaches to N₁.
3. N₁ attaches to N₄.
4. N₃ attaches to N₁.
5. N₁ attaches to N₃.

For the sake of clarity, we assume a uniform cost of 2 for all services.
Stage 1  \(N_4\) attaches to \(N_2\).

At the very start, \(N_4\) attaches to \(N_2\). After the attachment process, it has acquired the Service Path: \(< F : N_4/N_2/S_2/S_v 2 >\). The forward and backward ATR tuples are worked out: \((N_4,N_1,N_2,N_2), (N_4,N_2,N_4,N_2)\) respectively. The backward ATR tuple is sent back to \(N_2\). The ATR tuples derived are shown in Table 1.

Stage 2  \(N_2\) attaches to \(N_1\). When \(N_2\) attaches to \(N_1\), it will acquire the Service Path: \(< F : N_2/N_1/S_1/S_v 1 >\). \(N_2\) extracts the forward ATR tuple \((N_2,N_2,N_1,N_1)\) and backward ATR tuple \((N_2,N_1,N_2,N_1)\) which is sent to \(N_1\).

At the same time, \(N_2\) has to inform \(N_4\) of the newly acquired Service Path and \(N_4\) will add a corresponding Service Path to its service list: \(< F : N_4/N_2/N_1/S_1/S_v 1 >\). The forward and backward ATR tuples extracted by \(N_4\) from the newly acquired Service Path are listed in Table 2. Note that a network keeps only those ATR tuples that are useful to them. An ATR tuple is useful only to the network which has the same name as the attribute \(< Net_{from} >\) in the tuple. Hence \(N_1\) will keep the second and last ATR tuples while \(N_2\) will keep the first, fourth and fifth ATR tuples in Table 2.

Stage 3  \(N_1\) attaches to \(N_4\). \(N_1\) acquires service paths \(< F : N_1/N_4/S_4/S_v 4 >\), \(< F : N_1/N_4/N_2/S_2/S_v 2 >\) from \(N_4\). \(N_1\) will pass the newly acquired service paths to \(N_2\) but \(N_2\) will have only one additional service paths \(< F : N_2/N_1/N_4/S_4/S_v 4 >\), \(< F : N_2/N_1/N_4/N_2/S_2/S_v 2 >\) is referring to a local service. No new service path is acquired by \(N_4\) as the services will then have a looping Network Path like \(N_4/N_2/N_1/N_4/N_2/S_2/S_v 2 >\). \(N_1\) and \(N_2\) will follow the same steps as discussed in Stage 2 to build the ATR tuples as shown in Table 3.

Stage 4  \(N_3\) attaches to \(N_1\). \(N_3\) acquires, from \(N_1\) service paths \(< F : N_3/N_1/S_1/S_v 1 >\), \(< F : N_3/N_1/N_4/S_4/S_v 4 >\), \(< F : N_3/N_1/N_4/N_2/S_2/S_v 2 >\). ATR tuples are generated similar to the process in previous stages and listed in Table 4.

Stage 5  \(N_1\) attaches to \(N_3\). This time, \(N_1\), \(N_2\) and \(N_4\) will all have new Service Paths: \(< F : N_1/N_3/S_3/S_v 3 >\), \(< F : N_3/N_1/N_3/S_3/S_v 3 >\) respectively. The ATR tuples are listed in Table 5.

We can easily verify that the ATR tuples indeed represent all the necessary AToken routing information. Our next step is to demonstrate how ATR tuples can be used to build up a graphical representation of an SNG.

### 3.4 Graphical Representation of SNG from ATR Tuples

We notice that in any ATR tuple, if the \(< Net_{from} >, < Net_{to} >\) attribute pair matches with the attribute pair \(< Net_{ori} >, < Net_{dest} >\) in a forward route, network \(Net_{ori}\) is attached to network \(Net_{dest}\). To build up a graphical representation of an SNG, we need to have all the direct authentication delegation relationships. Other indirect authentication delegation relationships can be used to verify the result.

From the set of ATR tuples derived in Section 3.3, we collect all distinct ATR tuples that have the match attribute pairs in Table 6. Note that the last action ends up with a double headed arrow. The final SNG may have a different look from the SNG in Figure 1, but they have the same authentication delegation relationships and in which case, we consider them as two valid graphical representations of the same SNG. In next section, we will explore how to implement SNG over IP networks using network simulator OMNeT++.
4 Application of SNG over IP networks

To illustrate how SNG can be applied to the secure service sharing over IP networks, we design a few IP networks and simulate the SNG shown in Figure 2 using network simulator OMNeT++. The simulation network was set up with the following assumptions:

- Each network has one AuthenticationServer (AS), one ServiceListingServer (SLS), one Server (S) and one User (U).
- Each server (network) provides one type of services.
- A service request originates from user $U_1$ of network $N_1$.
- SNG configuration is shown in Figure 2.

In the simulations, the messages used carry five parameters: server, start, stop, src, dest. These five parameters are used to hold the information required for ATToken forward and backward routing.

The ATR tuples from Section 3.3 are hard coded in the OMNeT++ module for AS. We code all tuples in one module used for all AS. In practice, only the tuples required for each AS need to included in the implementation of each AS.

The simulated events are

1. A local service request for $Sv_1$ provided by $S_1$ in $N_1$ from $U_1$.
2. A directly shared service request for $Sv_4$ provided by $S_4$ in $N_4$ from $U_1$. The Authentication Token has to propagate from $AS_1$ to $AS_4$.
3. An indirectly shared service request for $Sv_2$ provided by $S_2$ in $N_2$ from $U_1$. The service is provided by $S_2$ in $N_2$. $N_2$ shares the service with $N_4$. On the other hand, $N_4$ shares all services it provides, including shared service $Sv_2$, with $N_1$. Hence users from $N_1$ can access indirectly shared service $Sv_2$. The Authentication Token now has to propagate from $AS_1$ to $AS_2$ via $AS_4$.

4.1 Local Service Request

User requesting a local service is the simplest case in SNG. It does not matter if the network is part of an SNG or not as it does not involve any Authentication Token routing. Figure 3(a) shows the basic operation of a local service request. A local service request starts with a request for a service list from the Service Listing Server $SLS_1$. The user $U_1$ then choose a service, sends the corresponding Service Path $< F : ./S_1/Sv_1 >$ and authentication information to the Authentication Server $AS_1$. $AS_1$ generates a session key and send it to Server $S_1$. $S_1$ replies $AS_1$ with the authentication for $Sv_1$. $AS_1$ will relay the information for $Sv_1$ and the session key to user $U_1$. Note that $AS_1$ does not need any ATR tuple for authentication Token routing.

A screen capture of simulation output window is shown in Figure 3(b).

4.2 Shared Service Request - Direct Sharing

For shared services, there are two cases - direct sharing and indirect sharing. Direct sharing is the case when you access local services of a delegator network. Indirect sharing is the case when you access shared services of a delegator network. Indirect sharing is possible when the authentication delegation is not restricted. For simplicity of discussion, we assume that
all delegations are free and all shared services can be further shared with other networks when they attach to a network in an SNG. Figure 4(a) shows an SNG with \( N_1 \) attached to \( N_4 \). \( U_1 \) initiated a service list request to \( SLS_1 \). This time, he chooses a service offered by \( S_4 \). A screen capture of simulation output window is shown in Figure 4(b).

### 4.3 Shared Service Request - Indirect Sharing

When a network \( N_1 \) shares its services with another network \( N_2 \), \( N_2 \) may further share the services to other networks attached to it. These type of sharing is an example of indirect sharing. Figure 5(a) shows an SNG with \( N_4 \) attached to \( N_2 \). When \( N_1 \) attaches to \( N_4 \), all shared services from \( N_2 \) is indirectly shared with \( N_1 \) also. A screen capture of simulation output window is shown in Figure 5(b). Note that the main difference in the three cases is the Authentication Propagation process. In local service access, there is no Authentication Propagation. For direct sharing of services, Authentication Propagation is just single hop traffic and indirect sharing of services involves multi-hops Authentication Propagation.

Although extra message passing are involved in the Authentication Propagation process, after authentication, the user communicates directly with the server. The traffic overhead is minimized this way.

### 5 Conclusion

Service sharing is one of the desired features of networks. Cross network authentication is the key to the success of service sharing. The dynamic nature of the networks participating in service sharing adds more constraints to plausible solutions. In this paper, we proposed the ATR tuples to represent the routing information of ATokens. We have demonstrated how to derive those Tuples from an SNG and have established a graphical representation of an SNG from the ATR tuples. We also applied the Tuple representation in the implementation of SNG over IP networks and illustrated the case with an OMNeT++ simulation. In the future, we will focus on optimizing the Service paths and establishing an optimal local view of an SNG. Authorization for SNG is another important research topic in the future.

### References


Figure 1: A Service Graph

Figure 2: User requesting a shared service
<table>
<thead>
<tr>
<th>Service Path</th>
<th>In</th>
<th>ATR tuple</th>
<th>Direction</th>
<th>Sent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; F: N_4/N_2/S_2/S_2v_2 &gt;:&lt; 2 &gt;)</td>
<td>(N_4)</td>
<td>((N_4, N_4, N_2, N_2))</td>
<td>forward</td>
<td>(N_4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((N_4, N_2, N_4, N_2))</td>
<td>backward</td>
<td>(N_2)</td>
</tr>
</tbody>
</table>

Table 1: ATR tuples derived in Stage 1

<table>
<thead>
<tr>
<th>Service Path</th>
<th>in</th>
<th>ATR tuple</th>
<th>Direction</th>
<th>Sent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; F: N_4/N_1/S_1/S_1v_1 &gt;:&lt; 2 &gt;)</td>
<td>(N_2)</td>
<td>((N_2, N_4, N_1, N_4))</td>
<td>forward</td>
<td>(N_2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((N_2, N_1, N_2, N_1))</td>
<td>backward</td>
<td>(N_1)</td>
</tr>
</tbody>
</table>

Table 2: ATR tuples derived in Stage 2

<table>
<thead>
<tr>
<th>Service Path</th>
<th>In</th>
<th>ATR tuple</th>
<th>Direction</th>
<th>Sent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; F: N_1/N_4/S_4/S_4v_4 &gt;:&lt; 2 &gt;)</td>
<td>(N_1)</td>
<td>((N_1, N_4, N_4, N_1))</td>
<td>forward</td>
<td>(N_1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((N_1, N_4, N_1, N_4))</td>
<td>backward</td>
<td>(N_4)</td>
</tr>
</tbody>
</table>

Table 3: ATR tuples derived in Stage 3

<table>
<thead>
<tr>
<th>Service Path</th>
<th>In</th>
<th>ATR tuple</th>
<th>Direction</th>
<th>Sent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; F: N_3/N_1/N_4/S_4/S_4v_4 &gt;:&lt; 2 &gt;)</td>
<td>(N_3)</td>
<td>((N_3, N_3, N_1, N_4))</td>
<td>forward</td>
<td>(N_3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((N_3, N_1, N_3, N_4))</td>
<td>backward</td>
<td>(N_1)</td>
</tr>
</tbody>
</table>

Table 4: ATR tuples derived in Stage 4
<table>
<thead>
<tr>
<th>Service Path</th>
<th>in ATR tuple</th>
<th>Direction</th>
<th>Sent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; F : N1/N3/S3/Sv3 &gt; :&lt; 2 &gt;</td>
<td>(N1, N1, N3, N3)</td>
<td>forward</td>
<td>N1</td>
</tr>
<tr>
<td></td>
<td>(N1, N3, N1, N3)</td>
<td>backward</td>
<td>N3</td>
</tr>
<tr>
<td>&lt; F : N2/N1/N3/S3/Sv3 &gt; :&lt; 2 &gt;</td>
<td>(N2, N1, N2, N3)</td>
<td>forward</td>
<td>N2</td>
</tr>
<tr>
<td></td>
<td>(N2, N1, N3, N3)</td>
<td>forward</td>
<td>N1</td>
</tr>
<tr>
<td></td>
<td>(N2, N3, N1, N3)</td>
<td>backward</td>
<td>N3</td>
</tr>
<tr>
<td>&lt; F : N4/N2/N1/N3/S3/Sv3 &gt; :&lt; 2 &gt;</td>
<td>(N4, N1, N2, N3)</td>
<td>forward</td>
<td>N4</td>
</tr>
<tr>
<td></td>
<td>(N4, N2, N4, N3)</td>
<td>backward</td>
<td>N2</td>
</tr>
<tr>
<td></td>
<td>(N4, N2, N1, N3)</td>
<td>forward</td>
<td>N2</td>
</tr>
<tr>
<td></td>
<td>(N4, N1, N2, N3)</td>
<td>forward</td>
<td>N1</td>
</tr>
<tr>
<td></td>
<td>(N4, N1, N3, N3)</td>
<td>backward</td>
<td>N3</td>
</tr>
</tbody>
</table>

Table 5: ATR tuples derived in Stage 5

<table>
<thead>
<tr>
<th>ATR Tuple</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N4, N4, N2, N2)</td>
<td>Draw N4. Draw N2. Draw an arrow from N4 to N2</td>
</tr>
<tr>
<td>(N2, N2, N1, N1)</td>
<td>Draw N1. Draw an arrow from N2 to N1</td>
</tr>
<tr>
<td>(N1, N1, N4, N4)</td>
<td>Draw an arrow from N1 to N4</td>
</tr>
<tr>
<td>(N3, N3, N1, N1)</td>
<td>Draw N3. Draw an arrow from N3 to N1</td>
</tr>
<tr>
<td>(N1, N1, N3, N3)</td>
<td>Draw an arrow from N3 to N1</td>
</tr>
</tbody>
</table>

Table 6: ATR tuples and the corresponding graphical representation

(a) Simulation network configuration  (b) Simulation output

Figure 3: Local service request
Figure 4: Directly shared service request

Figure 5: Directly shared service request