NSIS for NS-2

NSIS (Next Steps in Signalling) is a signalling framework being developed by the IETF, based on various signalling protocols, of which the Resource Reservation Protocol (RSVP) is the corner stone. This framework is used for application signalling, in order to install and maintain flow states in the network, similar to other protocols such as the aforementioned RSVP.

GIST

Currently, the NSIS implementation for NS-2 is based on the [1] specification and is intended to simulate the messaging transport and routing mechanisms of the GIST layer. Although the NSLP layer is represented by some specific objects it does not play any important role on the simulations. It is just a stub intended to make the GIST simulations closer to a real implementation.

The GIST module provides all objects specified in [1]. However, some TLVs (type-length-value) objects included in the GIST message, are not used to take any decision.

Simulation Model

To transport the messages between peers, the GIST layer can operate at datagram or connection mode. Although on the NSIS specification the use of one or another mode depends on the local policies of the node (e.g. the router) and the requirements of the application sending the message, on the ns-2 the selection is based just on a configuration parameter set at *Tcl* script level. Although a number of existing protocols (e.g. SCTP, DCCP, etc.) can be used in both cases, the simulated scenarios are configured to use UDP for datagram mode and TCP for connection mode.

When employing the connection mode, the TCP connections are reused whenever possible. To decide whether a new connection is to be established or if an existing one should be reused, it is taken into account just the path the message flows through. In this way, if a TCP connection already exists in the link between two adjacent GIST nodes, it will be used. In contrary, a new connection will be created. **Erro! A origem da referência não foi encontrada.** illustrates a case where just one TCP connection is used by all the applications. Nodes N0, N3, and N4 need to contact (by TCP connection) nodes N5, N6, and N7, respectively. As all NSIS messages need to cross a common link, just a single TCP connection is established between N1 and N2.

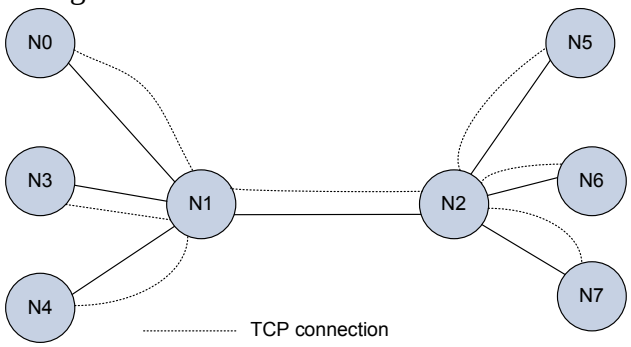

Figure 1: TCP connection reuse

If a TCP connection remains idle for a certain period of time it will automatically be closed. Case the connection is needed after that, another TCP connection establishment process is initiated. Although a special kind of refresh message can periodically be sent by the node to maintain the connection alive, we do not simulate it.

In the simulations all GIST nodes implement a three-way handshake to set up the necessary routing states between adjacent peers, as illustrated in Figure 2. When required by the NSLP application, the sender starts the discovery process of the next GIST peer by sending a Query Message. The

message is intercepted by the peer that stores some sender's information and sends back a Response Message. Finally, the sender feeds its routing state table with information of the peer and generates a Confirm Message.

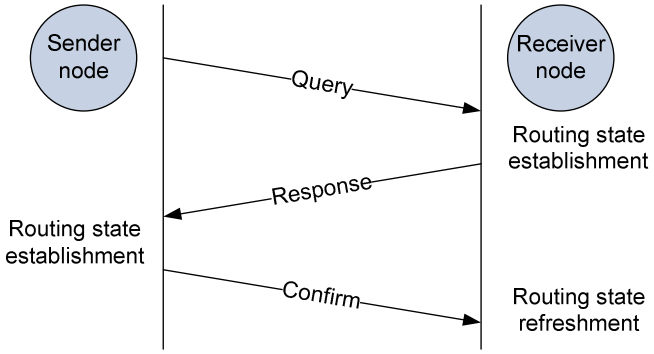

Figure 2: Three-way handshake

The GIST refresh messages also simulated. They are needed to ensure that the routing states remain valid along the time (i.e. there were no route changes). If after a specified time *x* a refresh message does not come, the corresponding routing state will be removed.

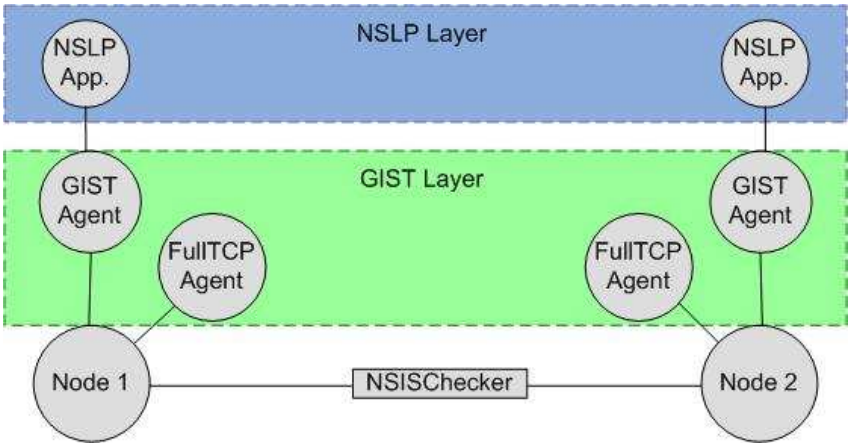

Figure 3 NSIS components

The high-level components of a NSIS simulation scenario are depicted in Figure 3 and described below:

- NSLP application - responsible for starting a NSIS communication by calling the *sendMessage* GIST method. When the NSIS agent sends a path message, each node along the data path must intercept and process it, create a new message, and send it towards the receiver specified in the MRI;
- GIST agent – It inherits the Agent class and is responsible for the datagram mode GIST communications (using the UDP transport protocol). The GIST agent is also responsible for the background maintenance, removing from the routing state table the flows that have not been refreshed. When receiving a message, the agent can process and forward it even if the message is not matched with local application signalling (NSLP application). In this case, the GIMPS object decreases the GIMPS hop in the header and puts the message again in the link, using the *NSISChecker*;
- FullTCP agent – a TCP agent implementation that can transmit data in both directions (no TCP Sink agent is used). This feature is needed since the GIST peers needs to perform a handshake in order to establish a TCP connection. The agent is called by the GIST agent whenever a TCP connection is required. In this way, the GIST agent also intermediate the communication between the NSLP application and the FullTCP agent;
- NSISChecker object – As stated before when the NSIS agent sends a path-message to a given destination, every node along the data path must intercept it to discover which peer is

its previous hop. To accomplish this task avoiding changes in NS-2 router modules, an object have been added to the link module. The *NSISchecker* object intercepts all packets on the link and, in the presence of NSIS messages, it handles them to the appropriated GIST agent. This interaction is shown in Figure 4. If the message is not a NSIS type, the *NSISchecker* returns the packet back to the link without any changing on it.

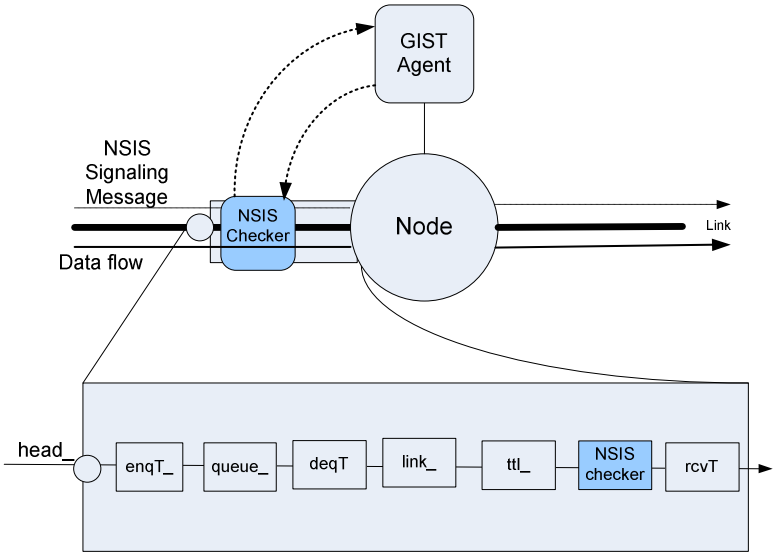

Figure 4 Interception of a NSIS message

TCL Interface

The major *Tcl* procedures and parameters needed to run a NSIS scenario are listed below:

To include a *NSISChecker* agent in the link to intercept the NSIS messages:
`<Simulator> config-nsis-link <src node> <dst node>`
Example:
```
$ns duplex-link $n0 $n2 1.2Mb 10ms DropTail
$ns config-nsis-link $n0 $n2
```
The lines above setup a link between nodes `n0` and `n2.`

To attach a new NSIS agent to a node:
`<Node> add-gimps-agent`
Example:
```
set nsis0 [$n0 add-gimps-agent]
```
The line above creates and attaches a GIST agent to the node n0 and stores a reference to it in `nsis0 variable`.

To establish a session between the NSIS agents:
`<GIST-agent> session <source node> <dst node> <flowID>`
Example:
```
set ses1 [$nsis0 session $n0 $n2 1]
```
The command creates a session between nodes n0 and n2 for the flow ID 1. At moment, the flow ID doesn't have any functionality but it can be used by others NSLP implementations. The reference of this session is stored in `ses1`, since GIST agent uses it later to send data.

To send data between NSIS agents:
`<GIST-agent> sender <session> "command"`
Example:

```
$ns at 1.3 "$nsis0 sender $ses1 \"xxx\""
```
The command above sends a path-message as a string "xxx", using the $ses1. As the NSLP layer implementation is just a stub, the actual string value does not matter. Once the packet arrives at the final destination, the packet is transferred from GIST to the NSLP layer and there it is disposed.

To connect two NSIS agents:
```
<Simulator> connect-gimps-agents <src_id> <dst_id>
```
Example:
```
$ns connect-gimps-agents [$n0 id] [$n2 id]
```

To select the transmission mode:
```
<Simulator> nsis-transmission-mode <mode>
```
The valid modes are: “dmode” for UDP connections (datagram) and “cmode” for TCP (connection mode).
Example:
```
$ns nsis-transmission-mode "dmode"
```

To configure the life time of the routing states stored at the GIST nodes:
```
Agent/GIMPS set lifetime <time>
```
Example:
```
Agent/GIMPS set lifetime_ 30
```

To create and attach a NSLP application to a GIST agent:
```
set app1 [new Application/NSIS]
$app1 attach-agent $lgimps
```

Full example

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set f1 [open out.ns w]
$ns trace-all $f1
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$ns duplex-link $n0 $n1 10Mb 1ms DropTail
$ns duplex-link $n1 $n2 10Mb 1ms DropTail
$ns config-nsis-link $n0 $n1
$ns config-nsis-link $n1 $n2
$ns nsis-transmission-mode "dmode" ;#datagram mode
Agent/GIMPS set lifetime_ 30
Agent/GIMPS set poolingTime_ 10
Agent/GIMPS set debug_ 0
set g0 [$n0 add-gimps-agent]
set g1 [$n1 add-gimps-agent]
set g2 [$n2 add-gimps-agent]
$ns connect-gimps-agents [$n0 id] [$n1 id]
$ns connect-gimps-agents [$n1 id] [$n2 id]
set app0 [new Application/NSIS]
set app1 [new Application/NSIS]
set app2 [new Application/NSIS]
$app0 attach-agent $g0
$app1 attach-agent $g1
$app2 attach-agent $g2
$n0 set class_ 1
set sesid0 [$app0 session [$n0 id] [$n2 id] 1]
$ns at 0.1 "$app0 sender $sesid0 7"
$ns at 60 "finish"
```

```
proc finish {} {
        global ns f1 nf
        $ns flush-trace
        close $nf
        close $f1
        exit 0
}
$ns run
```

References

1. H. Schulzrinne, R. Hancock, “GIST: General Internet Signaling Transport draft ietf-nsis-ntlp-09”; Internet Draft; February 2006.