Implementation and Evaluation of a NAT-Gateway for the General Internet Signaling Transport Protocol

Roland Bless and Martin Röhricht

Institute of Telematics, Department of Computer Science

Motivation

- Signaling protocols useful set of tools
  - Dynamically install, maintain, and manipulate state in network nodes
  - Create messaging associations between signaling peers

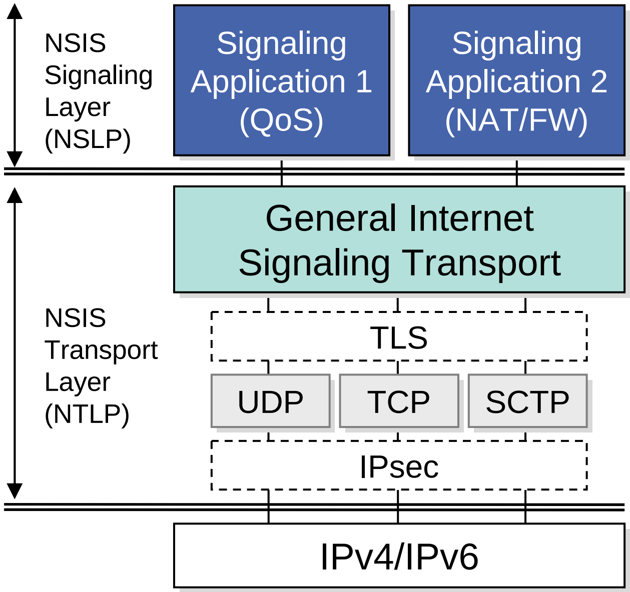
- Network Address Translation (NAT) gateways
  - Mitigate potential shortage of IPv4 addresses
  - Translate IP address and UDP/TCP port information

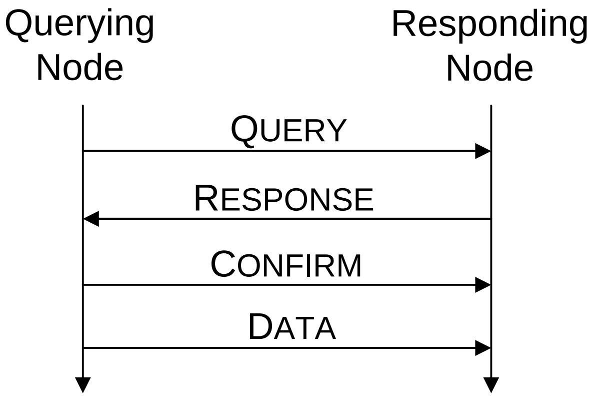
- Signaling messages carry IP address information in their payload
  - NAT gateway must be GIST-aware
    - Rewrite addressing information in signaling message’s payload

→ Create an application level gateway for the General Internet Signaling Transport (GIST) protocol

Next Steps in Signaling Framework

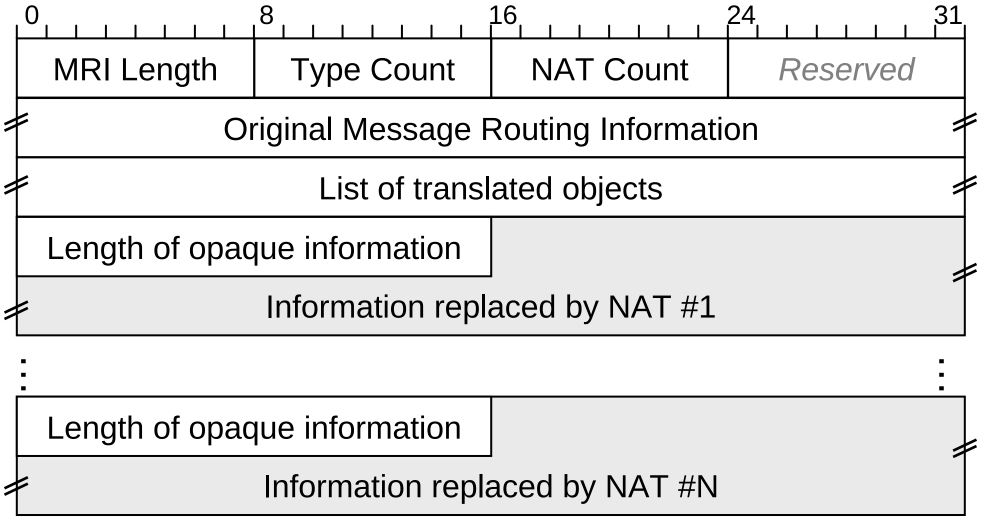
- IP-based signaling framework
- Two-layered approach
- General Internet Signaling Transport Protocol (GIST)
    - Routing and transport of signaling messages
        - Message Routing Information (MRI)
        - Network Layer Information (NLI)
        - Messaging Associations
    - 3-way handshake (QUERY, RESPONSE, CONFIRM) plus DATA
    - Supports delayed-state installation
        - Installation of routing state at Responding Node delayed until final CONFIRM arrives


NSIS Signaling Layer (NSLP)
Signaling Application 1 (QoS)
Signaling Application 2 (NAT/FW)
General Internet Signaling Transport
NSIS Transport Layer (NTLP)
TLS
UDP
TCP
SCTP
IPsec
IPv4/IPv6



Signaling Message’s Address Translation

- Transparent translation
  - Translate GIST header fields as is done with Layer 3 and 4
  - → Not applicable if cryptographic protection is used
- Non-transparent translation
  - Use special NAT Traversal Object (NTO)
    - Must be included by NAT gateway into initial QUERY message
    - Echoed back by Responding Node



0 8 16 24 31

MRI Length | Type Count | NAT Count | *Reserved*

Original Message Routing Information

List of translated objects

Length of opaque information

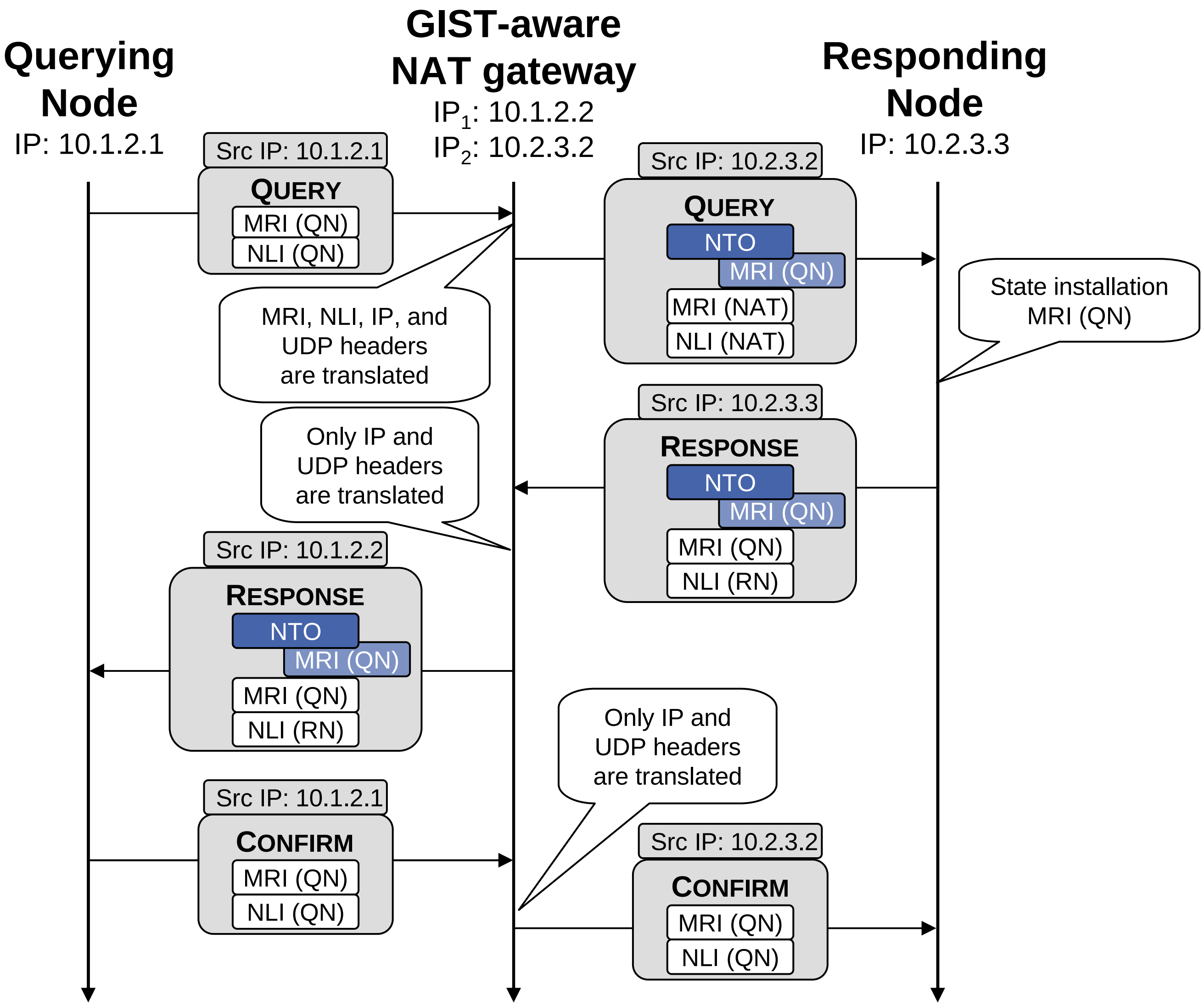
Information replaced by NAT #1

⋮

Length of opaque information

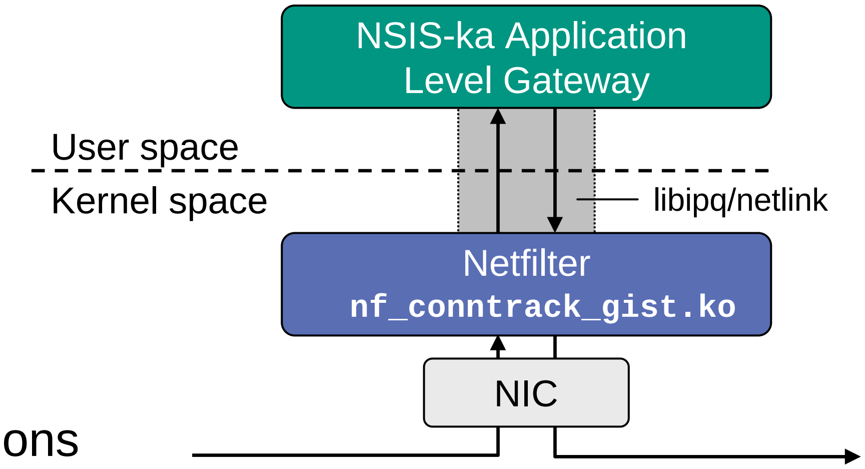
Information replaced by NAT #N

GIST handshake with GIST-aware NAT-gateway



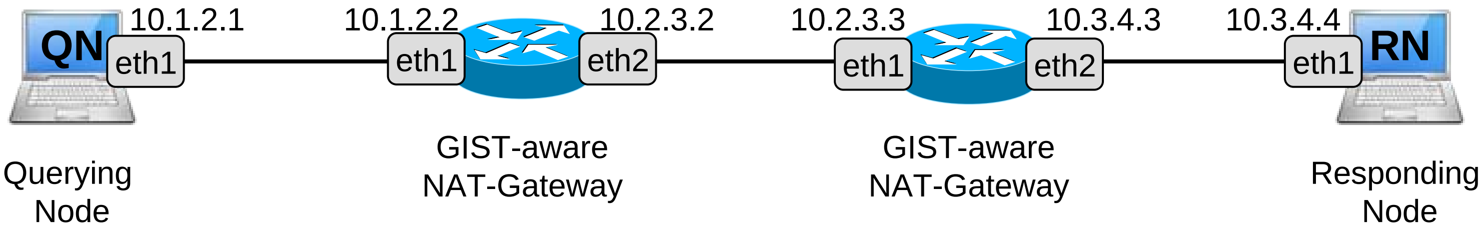
Implementation

- Kernel part
  - Intercept and filter GIST packets
  - Use Linux netfilter framework
  - Communication to user-space via Linux netlink messaging system
- User-space part
  - Performs remaining packet translations
    - Translate IP and UDP header
    - Translate address information in MRI and NLI
    - Insert NAT Traversal Object
    - Serialize GIST PDU, re-calculate IP and UDP checksums
  - Based on existing NSIS-ka implementation (http://nsis-ka.org)
    - Not entire NSIS-ka suite (~40,692 lines of code) required
- Kernel module – 420 lines of C code
- GIST-aware NAT gateway – 680 lines of C++ code



Evaluation

- Evaluation in a real testbed environment
  - Four standard PCs (Pentium IV 2.8 GHz, 4 GB RAM, Gbit Ethernet)
  - Ubuntu 10.04 with Linux kernel 2.6.32



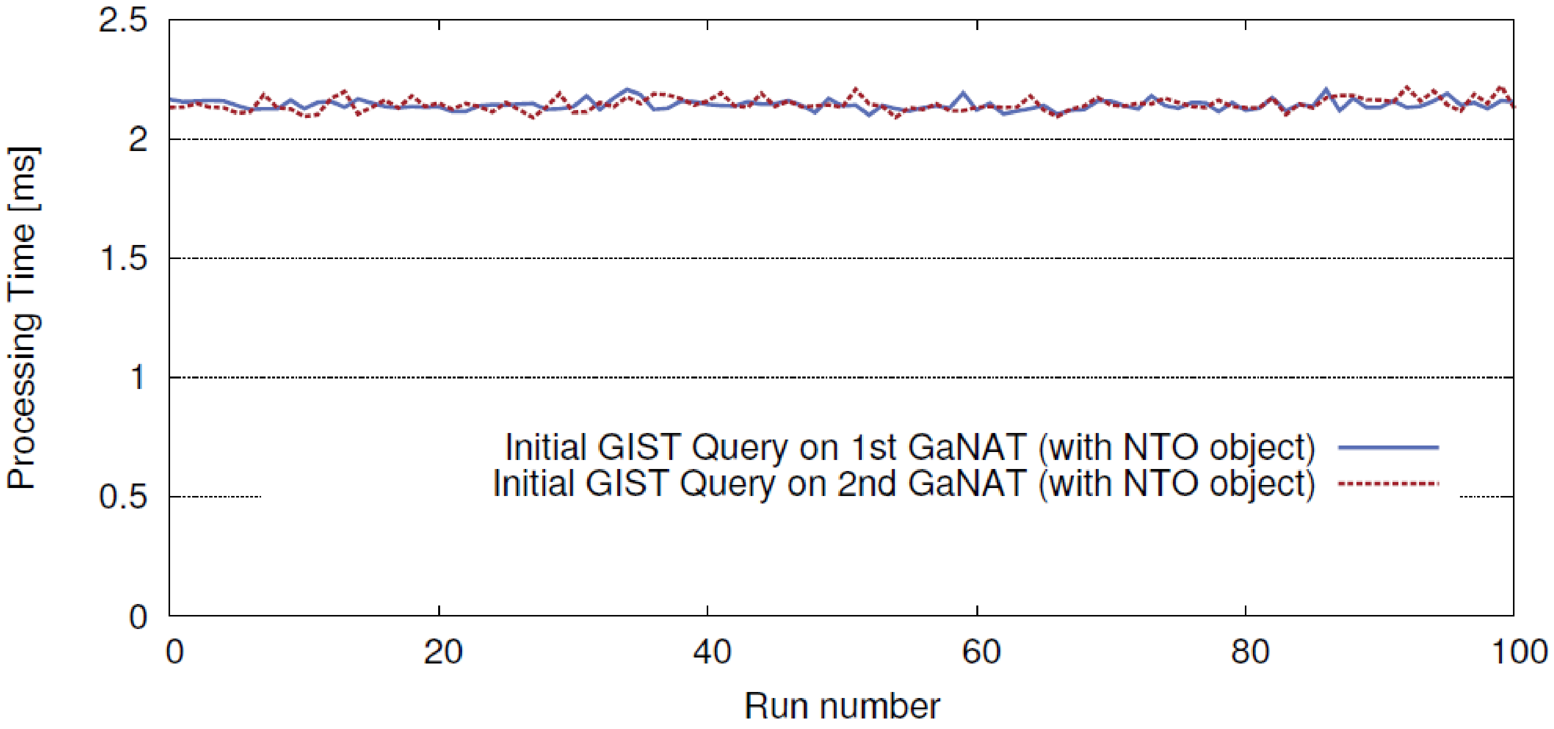
  - Latency intentionally kept small (~0.165 ms)

- Processing time of different GIST PDUs on first GIST-aware NAT gateway

- Processing time for complete GIST handshake and one subsequently sent DATA message

Evaluation – Processing time for initial QUERY message

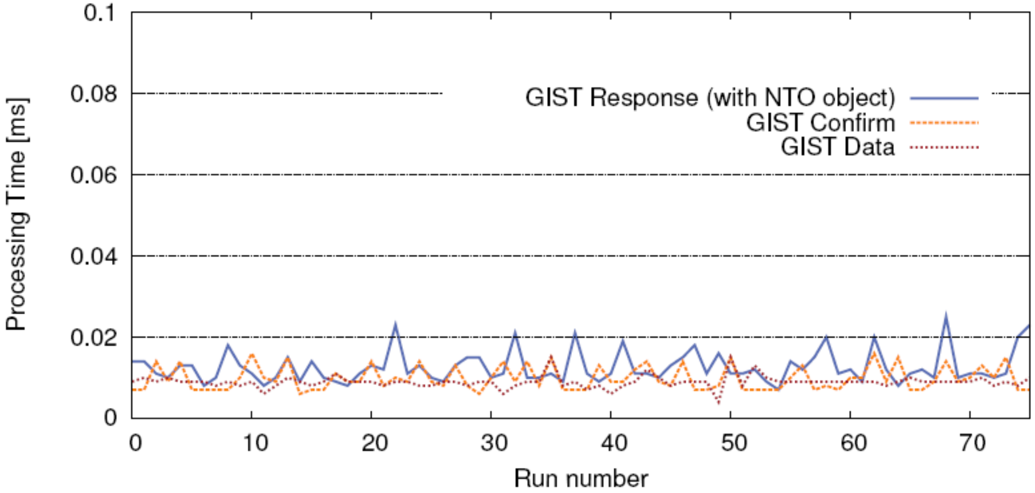
- Processing time for initial QUERY messages
  - NAT traversal objects are included



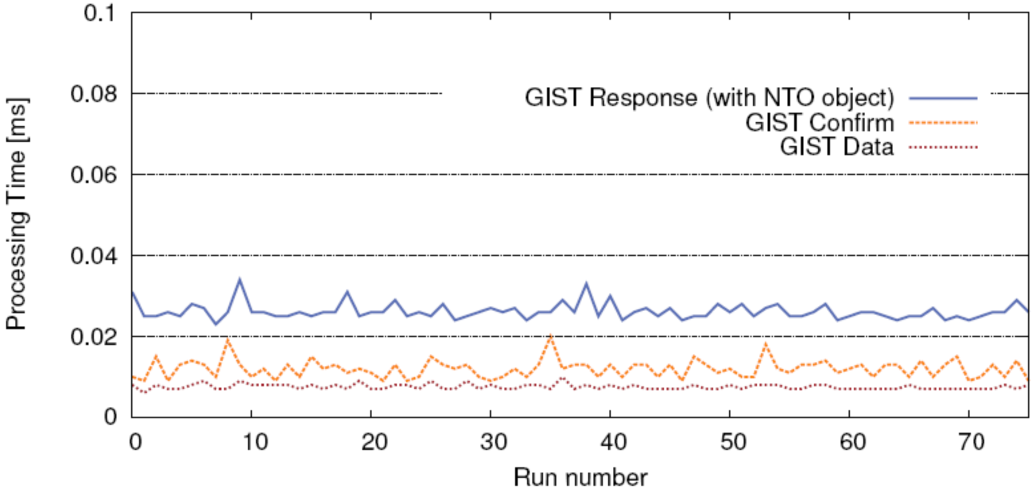
Evaluation – Processing Time of different GIST PDUs

- Measured on first GIST-aware NAT gateway

- Over TCP



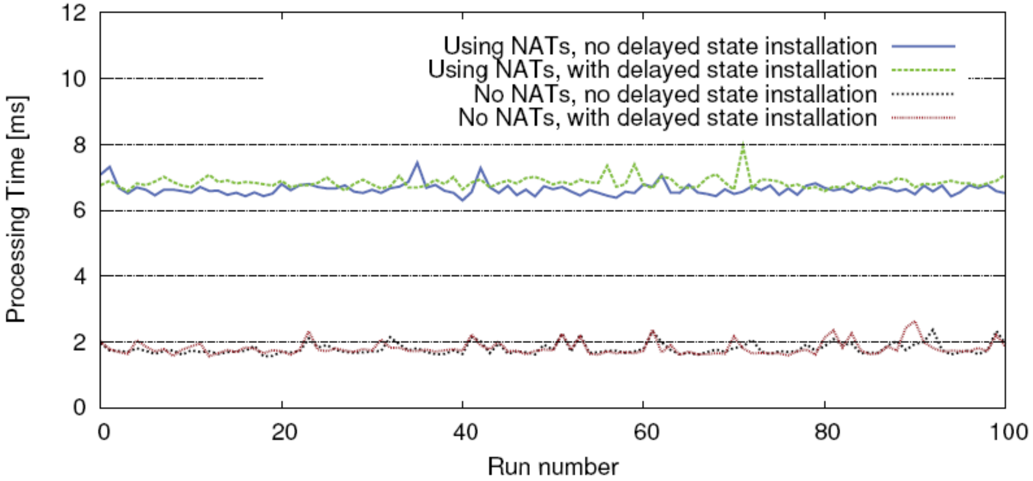
- Over UDP



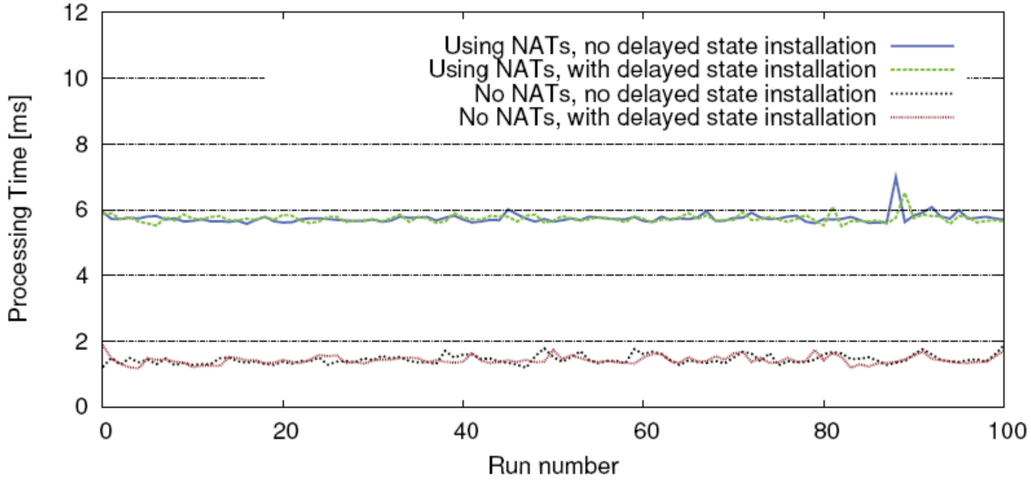
Evaluation – Complete GIST handshake

- Complete GIST handshake with one subsequently sent DATA message

- Measured on Querying Node using TCP



- Measured on Querying Node using UDP



Conclusion

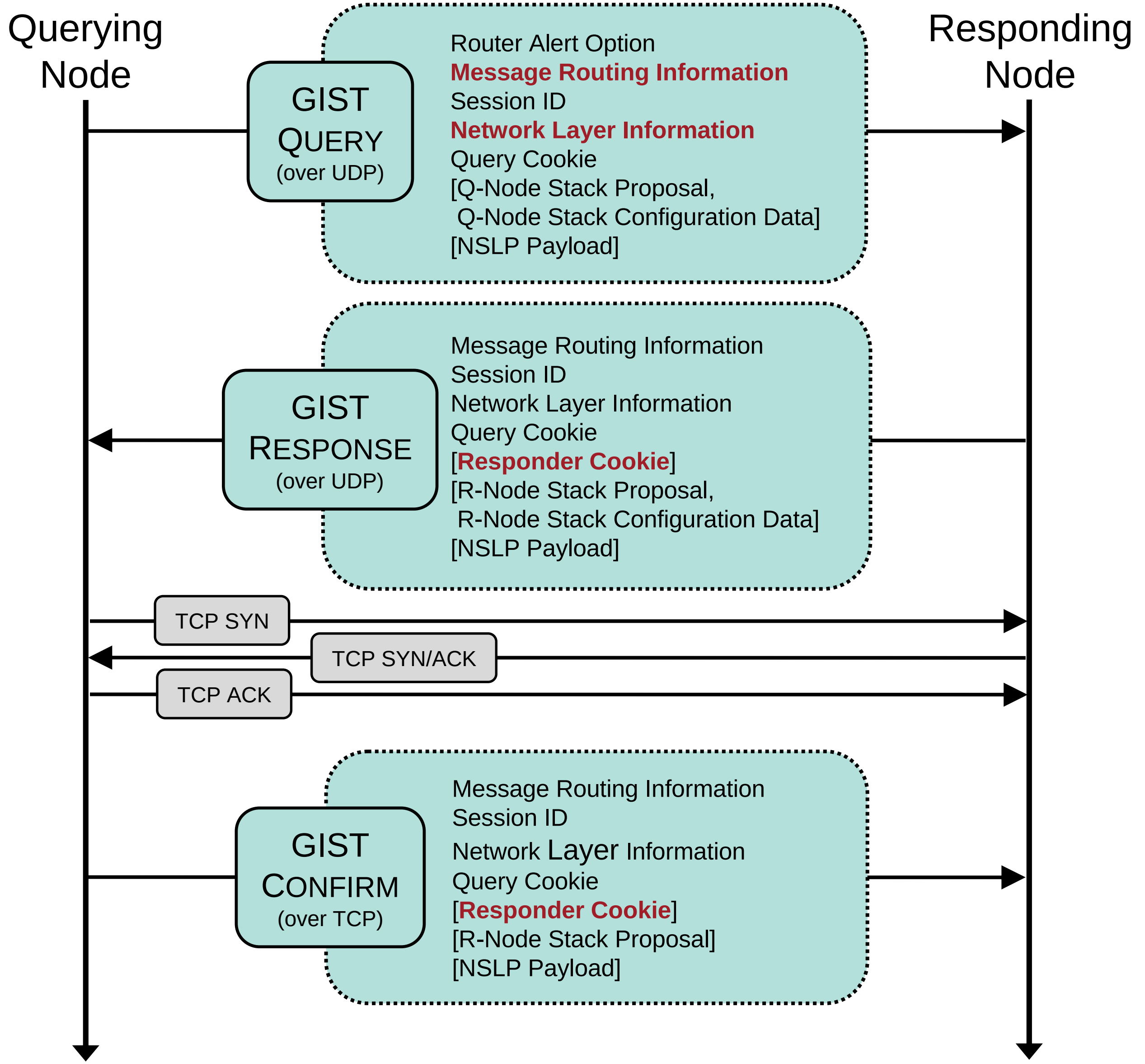
- Design of a NAT application level gateway for the General Internet Signaling Transport protocol
- Implementation of a NAT Traversal Object as being specified
  - Works as expected
  - Use GIST Responder Cookie for delayed-state installation

→ Allows NSIS signaling messages to safely traverse such NAT gateways

- Evaluations show
  - Slight overhead for initial GIST QUERY messages
  - Almost no overhead for subsequent GIST messages
  - Only small impact on duration of complete GIST handshake
  - Delayed-state installation with no notable performance overhead

Thank you for your attention

Questions?



Querying Node

Responding Node

GIST QUERY
(over UDP)

Router Alert Option
Message Routing Information
Session ID
Network Layer Information
Query Cookie
[Q-Node Stack Proposal,
Q-Node Stack Configuration Data]
[NSLP Payload]

GIST RESPONSE
(over UDP)

Message Routing Information
Session ID
Network Layer Information
Query Cookie
[**Responder Cookie**]
[R-Node Stack Proposal,
R-Node Stack Configuration Data]
[NSLP Payload]

TCP SYN

TCP SYN/ACK

TCP ACK

GIST CONFIRM
(over TCP)

Message Routing Information
Session ID
Network Layer Information
Query Cookie
[**Responder Cookie**]
[R-Node Stack Proposal]
[NSLP Payload]

Evaluation Results – Different PDUs

- Processing time of different GIST PDUs on the first GIST-aware NAT gateway

Processing time on the first GIST-aware NAT gateway

	Avg [ms]	Median [ms]	StdDev [ms]
UDP Query (with NTO)	2.153	2.161	0.152
TCP Response (with NTO)	0.012	0.011	0.004
UDP Response (with NTO)	0.026	0.026	0.002
TCP Confirm	0.010	0.009	0.003
UDP Confirm	0.013	0.012	0.002
TCP Data	0.009	0.009	0.001
UDP Data	0.008	0.007	0.001

Evaluation Results – Complete handshakes

- Complete GIST handshake with one subsequently sent DATA message

GIST handshake duration using TCP

	Avg [ms]	Median [ms]	StdDev [ms]
Using NATs, with DSI	6.843	6.820	0.178
Using NATs, without DSI	6.659	6.630	0.182
No NATs, with DSI	1.816	1.746	0.210
No NATs, without DSI	1.797	1.732	0.176

GIST handshake duration using UDP

	Avg [ms]	Median [ms]	StdDev [ms]
Using NATs, with DSI	5.737	5.722	0.127
Using NATs, without DSI	5.744	5.720	0.154
No NATs, with DSI	1.432	1.413	0.124
No NATs, without DSI	1.449	1.407	0.136