CONTENTS

› Extending SIP

› Examples of SIP extensions
  - Reliability of provisional responses
  - Preconditions
  - Caller preferences and user agent capabilities
  - SIP-Specific Event Notification
  - Signaling Compression (SigComp)
  - Content Indirection

› Peer-to-Peer SIP (P2PSIP)
  - Overview
  - Operation
  - Resource Location and Discovery (RELOAD)
EXTENDING SIP

› Global interoperability possible since the core functionality of SIP as specified in **RFC 3261** is present in every implementation
  - A given SIP application can always assume that another SIP application is able to understand the core protocol

› However, many implementations require functionality beyond the core protocol
  - Thus, extensions are required
  - SIP is flexible and easy to extend

› Use of extensions can be negotiated during session establishment
  - Two things are negotiated: the extensions the remote party supports and the extensions that will actually be employed in the session
SIP EXTENSION NEGOTIATION MECHANISM

- Three header fields: Require, Supported and Unsupported
- When a dialog is being established, the UAC lists
  - The names of the extensions it wants to use in a Require header field
  - The names of the extensions it supports in a Supported header field
  - The Unsupported header field is used in error responses
- The UAS can also request extra extensions
- Proxy-Require header field can be used to require support of extensions from proxies
- The extensions that a proxy or another UA supports can be queried by using an OPTIONS method
- The names of extensions are referred to as option tags
1. INVITE
Require: ext1, ext2
Supported: ext3, ext4

2. 200 OK
Require: ext3

3. ACK

Alice wants to use extensions ext1 and ext2. In addition to these, Alice supports ext3 and ext4

Bob wants to use an extra extension, ext3
NEW METHODS

› In a SIP dialog, UAs need to know which methods the other end understands
  - An Allow header field lists all the methods a UA supports
    
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE

› However, the Allow header field cannot be used to express that a particular method is required in a dialog
  - An option tag associated with the method can be used

› Processing of unknown methods and header fields:
  - **Proxies** forward unknown methods and header fields
  - **Redirect servers** ignore unknown header fields, methods and option tags in Require
  - **UASs** ignore unknown header fields and reject unknown methods
EXAMPLES OF SIP EXTENSIONS

› Reliability of provisional responses (RFC 3262)
› SIP-specific Event Notification (RFC 3265)
› User agent capabilities (RFC 3840)
› Caller preferences (RFC 3841)
› Preconditions (RFC 3312, 4032)
› Signaling Compression (RFC 3320, 3486)
› Content Indirection
  › SIP REFER method
    – Refer peers to third parties (RFC 3515)
    – Can be used to implement e.g. call transfer
  › Instant messaging (RFC 3428)
    – The MESSAGE method allows the transfer of instant messages
  › SIP UPDATE method (RFC 3311)
    – Update the parameters of a session
  › Event state publication (RFC 3903)
    – The PUBLISH method to publish e.g. presence information
  › Session timers in SIP (RFC 4028)
    – Periodic refresh of SIP sessions
  › SIP INFO method (RFC 2976)
    – To carry session related control information generated during a session
    – E.g. carrying DTMF digits generated during a SIP session
› And many others…
RELIABILITY OF PROVISIONAL RESPONSES

› Provisional responses are not transmitted reliably in RFC 3261
› However, reliability is important in several cases
› RFC 3262 defines an extension providing **reliable provisional responses**
  - The option tag of the extension is 100rel
  - **PRACK** method is used to acknowledge provisional responses
› The reliability mechanism works by mirroring the current reliability mechanisms for 2xx final responses to INVITE
› Each provisional response is given a sequence number, carried in a **RSeq** header field in the response
› The PRACK message contains an **RAck** header field
RELIABILITY OF PROVISIONAL RESPONSES

Support of reliable provisional responses required

Contains the sequence number of RSeq, sequence number of CSeq and method of CSeq of the reliable provisional response

100 Trying is an exception; it is never sent reliably.

Contains a sequence number in RSeq header field.

Alice

(1) INVITE
   CSeq: 1 INVITE
   Require: 100rel

(2) 100 Trying

(5) 180 Ringing
   CSeq: 1 INVITE
   RSeq: 12345

(6) PRACK
   RACK: 12345 1 INVITE

Proxy

(3) INVITE
   Require: 100rel
   CSeq: 1 INVITE

(4) 180 Ringing
   CSeq: 1 INVITE
   RSeq: 12345

(7) PRACK
   RACK: 12345 1 INVITE

Bob

(8) 200 OK (PRACK)

(9) 200 OK (PRACK)

(11) 200 OK (INVITE)

(12) ACK

(10) 200 OK (INVITE)

(13) ACK
A precondition is a set of constraints about the session which are introduced in the SDP offer.

The recipient of the offer generates an answer, but does not alert the user or proceed with session establishment.

RFC 3312 defines an extension allowing UAs to express preconditions:
- The option tag of the extension is ’precondition’
- A mixture between a SIP extension and a SDP extension

The preconditions are encoded in SDP body.

There are two types of preconditions: access and end-to-end:
- **End-to-end** (e2e) preconditions are useful when end-to-end resource reservation mechanisms are available.
- **Access preconditions** are useful when both UAs perform resource reservations on their respective access networks (local and remote).
EXAMPLE: ACCESS PRECONDITIONS

(1) INVITE
   a=curr:qos local none
   a=curr:qos remote none
   a=des:qos mandatory local sendrecv
   a=des:qos mandatory remote sendrecv

(2) 183 Session Progress
   a=curr:qos local none
   a=curr:qos remote none
   a=des:qos mandatory local sendrecv
   a=des:qos mandatory remote sendrecv

(3) PRACK

(4) 200 OK (to PRACK)

(5) UPDATE
   a=curr:qos local sendrecv
   a=curr:qos remote none
   a=des:qos mandatory local sendrecv
   a=des:qos mandatory remote sendrecv

(6) 200 OK (to UPDATE)
   a=curr:qos local none
   a=curr:qos remote sendrecv
   a=des:qos mandatory local sendrecv
   a=des:qos mandatory remote sendrecv

(7) 180 Ringing
CALLER PREFERENCES AND UA CAPABILITIES

› RFC 3841 describes a set of extensions to SIP which allow a caller to express preferences about request handling in servers
  - Ability to select which URI a request gets routed to
  - Specify request handling directives in proxies and redirect servers
  - Three new request header fields: Accept-Contact, Reject-Contact and Request-Disposition

› RFC 3840 defines mechanisms by which a SIP UA can convey its capabilities and characteristics to other UAs and to register for its domain
  - Contact header field parameters are used

› Example: Alice has multiple UAs: an office phone and a home phone
USER AGENT CAPABILITIES

The REGISTER request below, carries user agent capabilities in its Contact header field:

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bKnashds8
Max-Forwards: 70
From: sip:Alice@example.com;tag=asd98
To: sip:Alice@example.com
Call-ID: hh89as0d-asd88jkk@host.example.com
CSeq: 1 REGISTER
Contact: <sip:alice@host.example.com>;audio;video;
        mobility="fixed";
        class="business";
        language="en,fi";
        methods="INVITE,BYE,OPTIONS,ACK,CANCEL"
Content-Length: 0
```

- The UA supports audio and video communications.
- The UA is fixed as opposed to mobile.
- The UA is used for business communications.
- The UA supports the listed SIP methods.
- Language of the human or automata represented by the UA.
The Request-Disposition header field indicates how servers dealing with the request should handle it.

The Accept-Contact header field contains a description of the destination UAs to which it is OK to send the request.

The Reject-Contact header field contains a description of the UAs to which it is not OK to send the request.

```
INVITE sip:Bob.Jones@domain.com SIP/2.0
Via: SIP/2.0/UDP host1.domain2.com:5060;branch=z9hg4bK74oz98
Max-Forwards: 70
From: Alice <sip:Alice.Smith@domain2.com>;tag=79gy48298h8
To: Bob <sip:Bob.Jones@domain.com>
Call-ID: 56902805845684069@192.0.0.1
CSeq: 1 INVITE
Request-Disposition: proxy, parallel
Accept-Contact: */;mobility="mobile";methods="INVITE,OPTIONS,BYE,CANCEL,ACK,MESSAGE"
Reject-Contact: */;video
Contact: <sip:alice@192.0.0.1>
Content-Type: application/sdp
Content-Length: 180

(Message body not shown)
```
The SIP event notification framework can be used by SIP nodes to request notification from remote nodes.

- These notifications indicate that certain events have occurred.
- Examples:
  - Buddy lists
  - Automatic callback services
  - Message waiting indications

Entities in the network can subscribe:

- To resource state of resources in the network
- To call state of calls in the network

The entities receive notifications when the states of the resources/calls change.

The event notification framework uses two new SIP methods:

- **SUBSCRIBE** is used to subscribe to the status information of a resource
- **NOTIFY** is used to notify of the current status information of the resource and every time the status changes
Example: Bob subscribes to the presence status of Alice

### Subscribers

**Bob**

(1) **SUBSCRIBE** sip:alice@xyz.com
   - Event: presence
   - Expires: 600

(2) **200 OK**
   - Expires: 600

### Notifier

**Alice’s presence server**

(3) **NOTIFY**
   - Event: presence
   - Subscription-State: active;
   - expires=599
   - Content-Type: application/pidf+xml

(4) **200 OK**

- Type of status information is defined by an Event header field. Desired duration of the subscription is 600 seconds.
- The duration of the subscription will be 600 seconds.
- 'active' indicates that subscription has been accepted. The remaining time on the subscription is 599 seconds. The body of the NOTIFY contains a Presence Information Data Format (PIDF) document (XML), which describes Alice’s presence status.
SIP is not an efficient protocol regarding **message size**
- May be problematic e.g., in narrow-band wireless networks

Signaling Compression (SigComp) is a protocol for compressing messages of application protocols

SigComp messages carry **compressed** SIP messages in their payload
- The header contains a decompression algorithm (bytecode)

SigComp defines a Universal Decompressor Virtual Machine (UDVM)
Decompression algorithms are written in UDVM assembly language and compiled to **bytecode** using a UDVM interpreter

The bytecode is run on the UDVM to decompress the payload

A new parameter: **comp=sigcomp**
WHY IS SIGCOMP NEEDED?

INVITE tel:+1-212-555-2222 SIP/2.0
Via: SIP/2.0/UDP [5555::aaa:bbb:ccc:ddd]:1357;comp=sigcomp;branch=z9hG4bKnashds7
Max-Forwards: 70
Route: <sip:pcscf1.visited1.net:7531;lr;comp=sigcomp>, <sip:scscf1.home1.net;lr>
P-Preferred-Identity: "John Doe" <sip:user1_public1@home1.net>
P-Access-Network-Info: 3GPP-UTRAN-TDD; utran-cell-id-3gpp=234151D0FCE11
Privacy: none
From: <sip:user1_public1@home1.net>;tag=171828
To: <tel:+1-212-555-2222>
Call-ID: cb03a0s09a2sdflglj490333
Cseq: 127 INVITE
Require: precondition, sec-agree
Proxy-Require: sec-agree
Supported: 100rel
Security-Verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96; spi-c=98765432; spi-s=87654321; port-c=8642; port-s=7531
Contact: <sip:[5555::aaa.bbb:ccc:ddd]:1357;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Content-Type: application/sdp
Content-Length: (…)

v=0
o=-- 2987933615 2987933615 IN IP6 5555::aaa:bbb:ccc:ddd
s=--
c=IN IP6 5555::aaa:bbb:ccc:ddd
t=0 0
m=video 3400 RTP/AVP 98 99
b=AS:75
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:98 H263
a=fmt:98 profile-level-id=0
a=rtpmap:99 MP4V-ES
m=audio 3456 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:97 AMR
a=fmt:97 mode-set=0,2,5,7; mode-change-period=2
a=rtpmap:96 telephone-event
a=maxptime:20
**SIGNALLING COMPRESSION (2/5)**

- **Local application (SIP)**
  - SIP message
    - Compressor dispatcher
      - Compressor 1
      - Compressor 2
    - State handler
      - State 1
      - State 2
  - Interface from the application. Invokes a compressor
  - Implements a compression algorithm
- **Decompressor (UDVM)**
  - Receives a message, invokes the UDVM
  - Decompresses SIP messages
  - Stores and retrieves state
- **SigComp layer**
  - SigComp message
    - Carries the compressed SIP message and a decompression algorithm
- **Transport layer**
  - Decompressed SIP message
Basic idea: search for repeated patterns in the message

- I.e. exploit the redundancy within a message
- Replace the re-occurrences of a pattern with a pointer to the previous instance of the same pattern
- Some examples of repeated strings are shown in the figure

```plaintext
INVITE sip:Alice@domain.com SIP/2.0
Via: SIP/2.0/UDP p1.domain.com:5060;branch=xyz
Via: SIP/2.0/UDP c1.domain2.com:5060;branch=abc;
    ;received=123.0.100.4
Max-Forwards: 69
From: Bob <sip:Bob@domain2.com>;tag=123
To: Alice <sip:Alice@domain.com>
Call-ID: 123456789@123.0.100.4
Cseq: 1 INVITE
Contact: <sip:Bob@123.0.100.4>
Content-Type: application/sdp
Content-Length: 120

v=0
o=Bob 2890844526 2890844526 IN IP4 c1.domain2.com
s=-
c=IN IP4 123.0.100.4
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```
Often SIP messages belonging to the same dialog contain a lot of information that was also present in earlier messages of the same dialog. This redundant information can be compressed efficiently.

- **Dynamic compression**: compression relative to messages sent prior to the current compressed message.
- **Shared compression**: messages are compressed relative to messages received prior to the current compressed message.

```
INVITE sip:Alice@domain.com SIP/2.0
Via: SIP/2.0/UDP p1.domain.com:5060;branch=xyz
Via: SIP/2.0/UDP c1.domain2.com:5060;branch=abc;
   ;received=123.0.100.4
Max-Forwards: 69
From: Bob <sip:Bob@domain2.com>;tag=123
To: Alice <sip:Alice@domain.com>
Call-ID: 123456789@123.0.100.4
Cseq: 1 INVITE
Contact: <sip:Bob@123.0.100.4>
Content-Type: application/sdp
Content-Length: 120

v=0
o=Bob 2890844526 2890844526 IN IP4 c1.domain2.com
s=-
c=IN IP4 123.0.100.4
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP p1.domain.com:5060;branch=xyz
   ;received=123.1.0.5
Via: SIP/2.0/UDP c1.domain2.com:5060;branch=abc;
   ;received=123.0.100.4
From: Bob <sip:Bob@domain2.com>;tag=123
To: Alice <sip:Alice@domain.com>;tag=987
Call-ID: 123456789@123.0.100.4
Cseq: 1 INVITE
Contact: <sip:Alice@123.0.0.5>
Content-Type: application/sdp
Content-Length: 120

v=0
o=Alice 2890844545 2890844545 IN IP4 c1.domain2.com
s=-
c=IN IP4 123.0.0.5
r=0 0
m=audio 30000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```
Alice wants to receive future requests and responses for this dialog compressed.

Compressed, because Via header field contains comp=sigcomp. comp=sigcomp added to Record-Route; since UAC wishes to receive compressed requests (Contact of INVITE) it is assumed that it would also like to send compressed requests.

1. INVITE Bob *(compressed)*
   Via: Alice;comp=sigcomp
   Route: P1;comp=sigcomp
   Contact: Alice;comp=sigcomp

2. INVITE Bob
   Via: P1
   Via: Alice;comp=sigcomp
   Record-Route: P1
   Contact: Alice;comp=sigcomp

3. 200 OK
   Via: P1
   Via: Alice;comp=sigcomp
   Record-Route: P1
   Contact: Bob

4. 200 OK *(compressed)*
   Via: Alice;comp=sigcomp
   Contact: Bob
   Record-Route: P1;comp=sigcomp

5. ACK Bob *(compressed)*
   Via: Alice;comp=sigcomp
   Route: P1;comp=sigcomp
   Contact: Alice;comp=sigcomp

6. ACK Bob
   Via: P1
   Via: Alice;comp=sigcomp
   Contact: Alice;comp=sigcomp
Content Indirection (1/2)

- Content indirection allows one to replace a Multipurpose Internet Mail Extensions (MIME) body part with an external reference
  - The reference is typically a HTTP URI
- The destination UA fetches the contents of the MIME body part using the references contained in the SIP message

Motivation:
- Sometimes SIP message bodies are too large even after compression
- Reduce the load of proxies
- Content not residing on the endpoint
- Problems associated with IP fragmentation when message is transported over UDP (UDP does not provide transport-layer fragmentation)

Example
- Document sharing during instant messaging
CONTENT INDIRECTION (2/2)

Example: SDP as an external reference

INVITE sip:bob@example.com SIP/2.0
From: <sip:alice@example.net>;tag=347242
To: <sip:bob@example.com>
Call-ID: 3573853342923422@example.net
CSeq: 2131 INVITE
Accept: message/external-body application/sdp
Content-Type: message/external-body; ACCESS-TYPE=URL;
    URL="http://www.example.net/party/10/2008/announcement";
    EXPIRATION="Wed, 1 Oct 2008 12:00:00 GMT"; size=231
Content-Length: 105

Content-Type: application/sdp
Content-Disposition: session
Content-ID: <4e5562cd1214427d@example.net>
APPLICATION AREAS OF SIP

› 3G IP Multimedia Subsystem (IMS)
› SIMPLE (SIP Instant Messaging and Presence Leveraging Extensions)
› SIP VoIP/IM clients (some examples)
  – Pidgin (cross-platform)
  – SIP Communicator (cross-platform)
  – KPhone (Linux)
  – Sipdroid (Android)
  – Linphone (PCs, Android, iPhone)
› SIP-T (SIP for Telephones)
  – Interconnection of PSTN with IP, VoIP calls between gateways
› IP PBXs (Private Branch Exchange)
› Apple FaceTime (iPhone 4)
› Skype
  – Skype Connect (Skype for SIP)
Conventional **client/server SIP** relies on centralized proxy-registrar servers

In Peer-to-Peer SIP (P2PSIP), SIP is used in an environment where the centralized functions are replaced by a **P2P overlay network**

In the overlay network, address-of-record to contact URI mappings are distributed amongst the peers in the overlay

P2PSIP is being standardized in the P2PSIP working group of the **IETF**

"Standardized Skype"
PEER-TO-PEER SIP IN IETF

› Standardized in the P2PSIP Working Group (WG) of the IETF
› The WG is responsible for:
   – Defining concepts, terminology, rationale, and use cases for P2PSIP
   – Standardizing a P2PSIP Peer Protocol
   – Optionally, standardizing a P2PSIP Client Protocol
   – Producing a usage document for P2PSIP
› Topics that are out of the scope of P2PSIP:
   – Issues specific to applications other than locating users and resources for SIP-based communications and presence
   – Research type of questions
   – Locating resources based on something other than URIs
   – Multicast and dynamic DNS based approaches as the core lookup mechanism
**P2PSIP Overlay**

P2PSIP Overlay

- **P2PSIP Bootstrap Peer**: First point of contact for a peer joining the overlay. Can be located:
  - By remembering peers from the last time the peer was in the overlay
  - Through multicast discovery
  - Through manual configuration
  - By contacting a bootstrap server

- **P2PSIP Peer**: Participates in the P2PSIP overlay and provides storage and transport services to other nodes.

- **NAT**: Interacts with the P2PSIP overlay through its associated peer using the Client Protocol. Does not run the distributed database algorithm.

- **P2PSIP Client**: The protocol spoken between clients and peers. RELOAD is also used as the client protocol.

**Joining peer** is a node attempting to become a P2PSIP Peer. Admitting peer helps the joining peer join the network.

**Peer Protocol**: The protocol spoken between P2PSIP Peers to share information and organize the P2PSIP Overlay network. A protocol called Resource Location and Discovery (RELOAD) is used as the peer protocol.

**Client Protocol**: The protocol spoken between clients and peers. RELOAD is also used as the client protocol.

**Participates in the P2PSIP overlay and provides storage and transport services to other nodes.**
P2PSIP OPERATIONS (1/2)

- P2PSIP peers are capable of performing operations such as:
  - Joining and leaving
  - Store and fetch
  - Storing information on behalf of the overlay
  - Transporting messages

- **Joining**: to join a P2PSIP overlay, a joining peer needs to:
  - Contact an *enrollment server*
  - Obtain an overlay configuration document, *certificate* and *Node-ID*
    - Central enrollment process vs. self-generated certificates
  - Contact a *bootstrap peer*
    - The bootstrap peer will refer the joining peer to an *admitting peer*
  - Contact an admitting peer
    - The admitting peer will help the joining peer learn about other peers in the overlay and establish connections to them as appropriate
P2PSIP OPERATIONS (2/2)

› **Storing data**: to perform a user registration (i.e. to insert the user’s contact information into the overlay), a user needs to:
  - Calculate a hash of her user name (e.g. alice@example.com) to produce a **Resource-ID**
  - Locate the peer that is responsible for that Resource-ID
  - Store a Resource-ID to contact address mapping in the **responsible peer**

› **Fetching data**: to initiate a call:
  - Calculate a hash of the callee’s user name to produce a Resource-ID
    - hash(alice@example.com) = 32B4A7F02C
  - Locate the peer that is responsible for that Resource-ID in the P2PSIP overlay
    - A P2PSIP Resource Record with contact information is obtained: alice@example.com → Alice’s Node-ID
  - Establish a direct connection with the callee across NATs
  - Send a SIP INVITE request to the callee
EXAMPLE: ALICE CALLING BOB (1/3)

1. LOOKUP

(1) Calculate hash(bob@p2psip.net) = 4

(2) Fetch Resource Record with Resource-ID 4

(3) Forward Fetch request

(4) Return Resource Record: Bob’s Node-ID = 15

(5) Forward Fetch response

(6) Alice learns that Bob’s Node-ID = 15

Bob
sip:bob@p2psip.net
Node-ID: 15
Resource-ID: hash(bob@p2psip.net)=4

Carol
sip:carol@p2psip.net
Node-ID: 4

Bob’s Resource Record
Resource-ID: 4
Content: Bob’s Node-ID=15

Alice
sip:alice@p2psip.net
Resource-ID: hash(alice@p2psip.net) =2
Node-ID: 11
EXAMPLE: ALICE CALLING BOB (2/3)

2. ATTACH

(1) Establish a connection with Node-ID 15 (Bob's terminal)

(2) Send an Attach request to Bob

(3) Return an Attach response to Alice

(4) A direct connection for SIP between Alice and Bob

Bob
sip:bob@p2psip.net
Node-ID: 15
Resource-ID: 4

Alice
sip:alice@p2psip.net
Resource-ID: 2
Node-ID: 11

Carol
sip:carol@p2psip.net
Node-ID: 4

Bob's Resource Record
Resource-ID: 4
Content:
Bob's Node-ID=15
EXAMPLE: ALICE CALLING BOB (3/3)

Alice
sip:alice@p2psip.net
Resource-ID: 2
Node-ID: 11

Bob
sip:bob@p2psip.net
Node-ID: 15
Resource-ID: 4

3. INVITE

(1) INVITE

(2) 200 OK

(3) ACK
SOME CHALLENGES FOR P2PSIP

› Security and identity assertion
  - No fully distributed system for security exist which would be as robust as a centralized solution
  - Solution: RELOAD uses a centralized entity contacted at enrollment time

› Network Address Translators (NATs)
  - Most peers can be located behind NATs
  - Solution: use of standardized NAT traversal protocols
    › Session Traversal Utilities for NAT (STUN)
    › Traversal Using Relays around NAT (TURN)
    › Interactive Connectivity Establishment (ICE)

› Regulatory issues
  - Lawful intercept, emergency calls
RESOURCE LOCATION AND DISCOVERY (RELOAD)

› A **P2P signaling protocol** specified by the P2PSIP WG
› Used between peers forming an overlay network to provide a self-organizing overlay network service, including
  - Distributed storage
  - Message forwarding
› Allows access from **client nodes** which don’t route traffic or store data
› Provides the following features
  - Security framework
  - Usage model
  - NAT traversal
  - Routing
  - Pluggable overlay algorithms
Each application defines a RELOAD usage.

End-to-end reliability, request state management, dispatches messages and operations.

Provides packet forwarding services. Handles setting up connections across NATs using ICE.

Usages use RELOAD through Messaging API.

Processes messages related to storage and retrieval of data.

Implements an overlay algorithm.

Message Transport

Forwarding and Link Management

Storage

Topology Plugin

Messaging API

Usage layer

SIP Usage

XMPP Usage

Overlay Link API

TLS

DTLS

Each application defines a RELOAD usage.

End-to-end reliability, request state management, dispatches messages and operations.

Provides packet forwarding services. Handles setting up connections across NATs using ICE.
RELOAD FEATURES (1/2)

› Security framework
  – Node-IDs and certificates are assigned by a central enrollment server
  – Also self-signed certificates can be used
  – Security at three levels: connections, messages, stored objects

› Usage model
  – Allows the definition of new application usages
  – RELOAD can be used also by other applications than P2PSIP

› NAT traversal
  – Allows RELOAD to function in environments with NATs and firewalls
  – Interactive Connectivity Establishment (ICE) is used to establish new RELOAD and application protocol connections
Routing
- A lightweight forwarding header to minimize the load of intermediate peers
  - Via list and destination list
- Basic routing mechanism is symmetric recursive

Pluggable overlay algorithms
- RELOAD has an abstract interface to the overlay layer
- Each overlay can select an appropriate overlay algorithm
  - All algorithms rely on the common RELOAD core protocol
- RELOAD defines three methods for overlay maintenance: Join, Leave and Update
- Chord DHT is mandatory to implement
NAT TRAVERSAL

› SIP and RELOAD use Interactive Connectivity Establishment (ICE) to set up connections across NATs
  – ICE is used to discover a working path through NATs
  – (1) Gather candidate addresses
  – (2) Exchange candidates
  – (3) Perform connectivity checks

› ICE makes use of **STUN** and **TURN** protocols

› **STUN** – Session Traversal Utilities for NAT
  – Determine IP address and port allocated by NAT
  – Check connectivity
  – Keep-alives

› **TURN** - Traversal Using Relays Around NAT
  – Control the operation of a relay
NAT TRAVERSAL FOR MEDIA IN SIP

Diagram showing the flow of packets through TURN and NAT relays, with arrows indicating the direction of connectivity checks.