Extensions to Session Initiation Protocol (SIP) and Peer-to-Peer SIP

T-110.7100
Applications and Services in Internet
3.11.2009

Jouni Mäenpää
NomadicLab, Ericsson

ERICSSON
TAKING YOU FORWARD

Contents

- 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

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

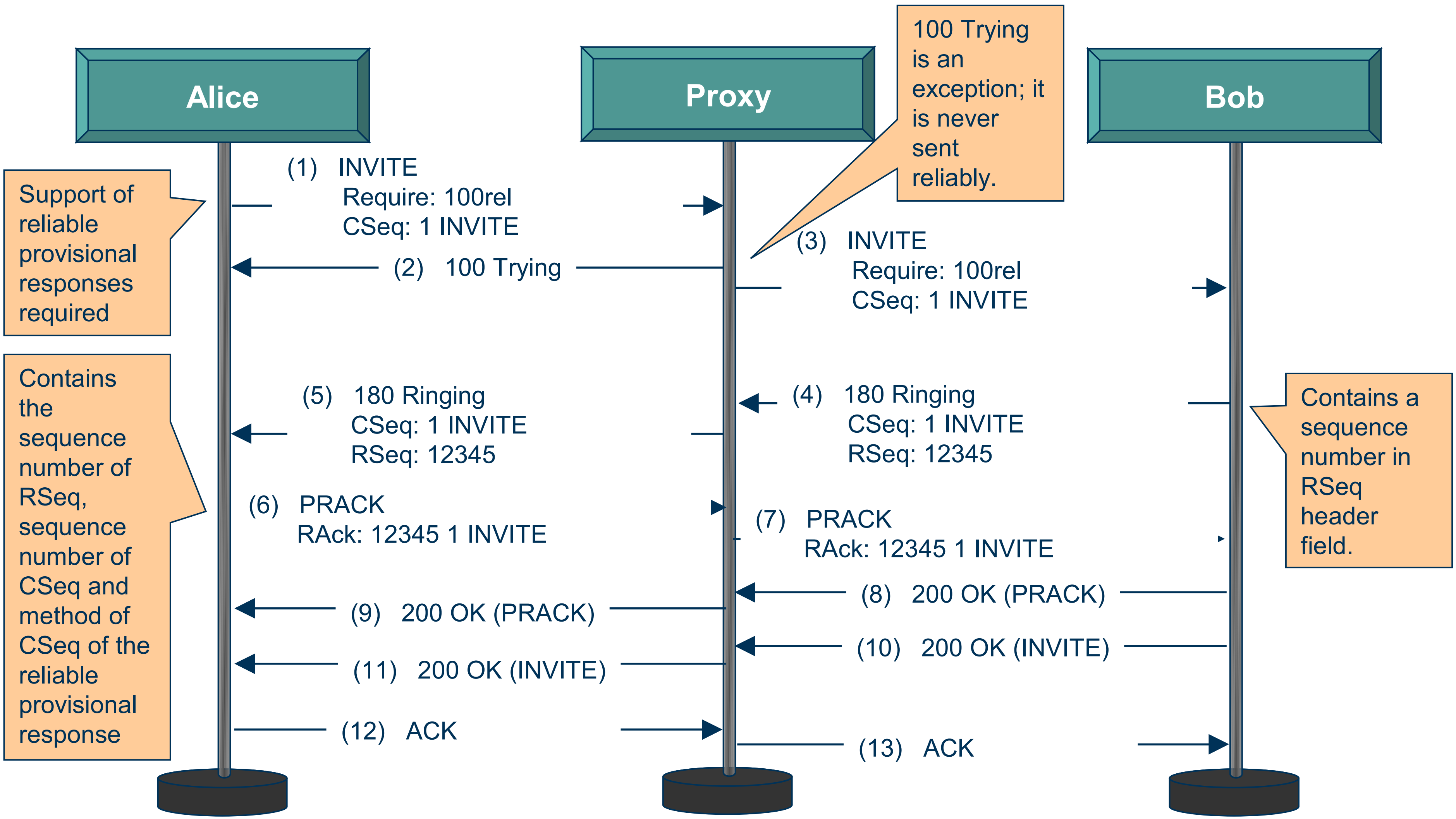
Examples of SIP Extensions

- 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 the core SIP protocol (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

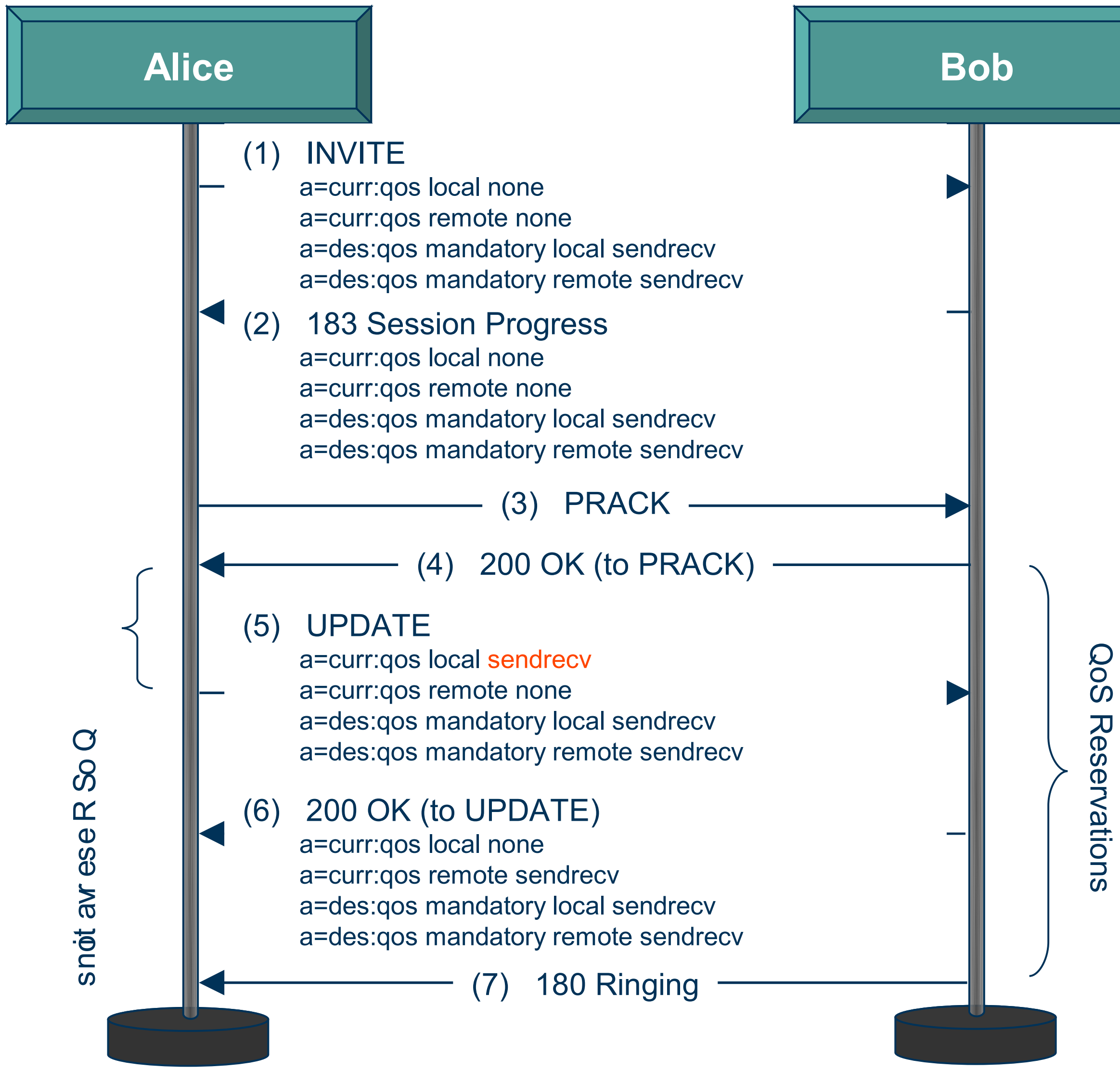
Reliability of Provisional Responses



Preconditions

- 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



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

Language of the human or automata represented by the UA

The UA supports the listed SIP methods

Caller Preferences

- 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

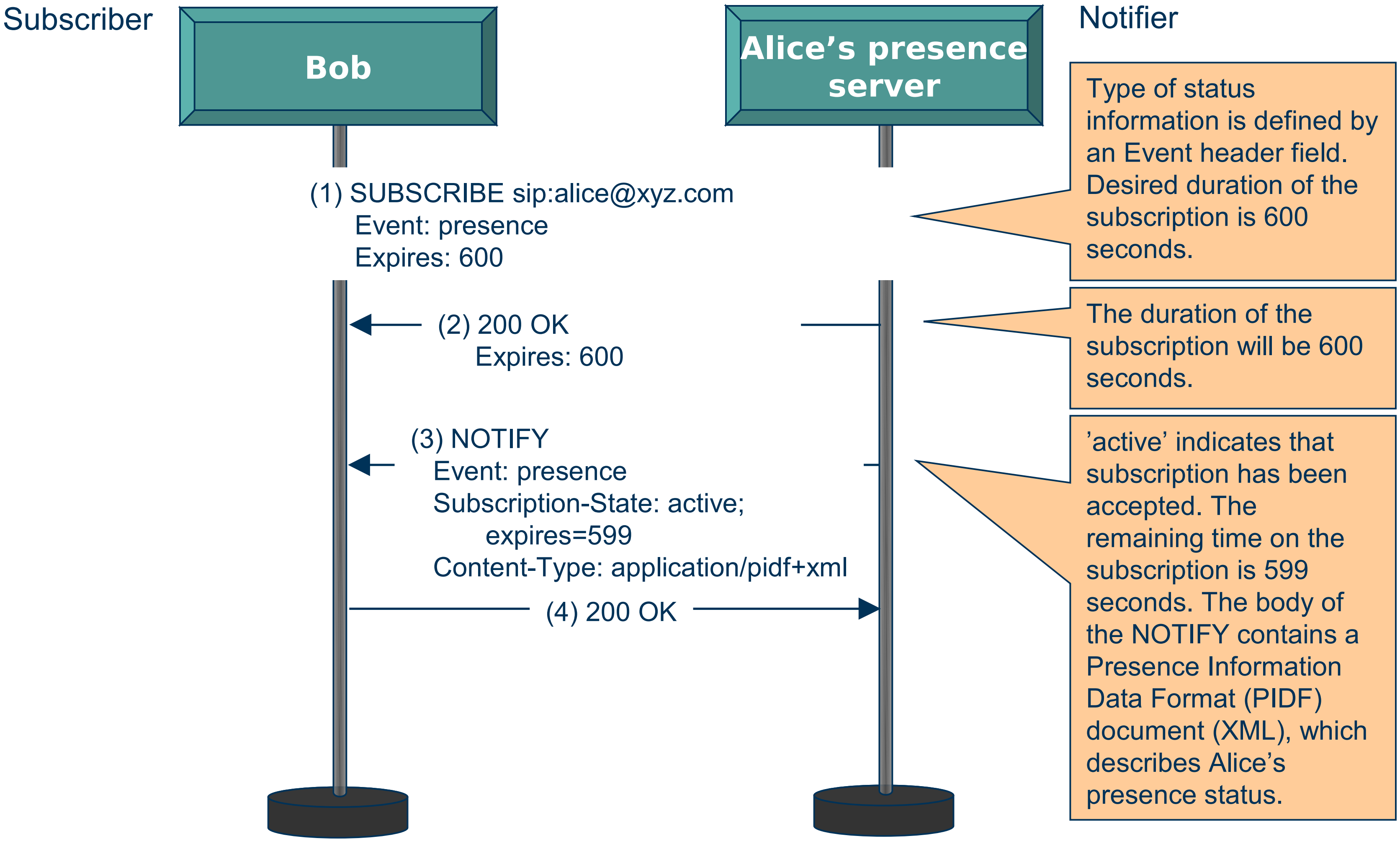
```
INVITE sip:Bob.Jones@domain.com SIP/2.0
Via: SIP/2.0/UDP host1.domain2.com:5060;branch=z9hg4bK74oz98
Max-Forwards: 70
From: Alice <sip:Alice.Smith@domain2.com>;tag=79gy48298h8
To: Bob <sip:Bob.Jones@domain.com>
Call-ID: 569028058456840690192.0.0.1
CSeq: 1 INVITE
Request-Disposition: proxy, parallel
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)
```

SIP-Specific Event Notification (1/2)

- 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

SIP-Specific Event Notification (2/2)

- Example: Bob subscribes to presence status of Alice



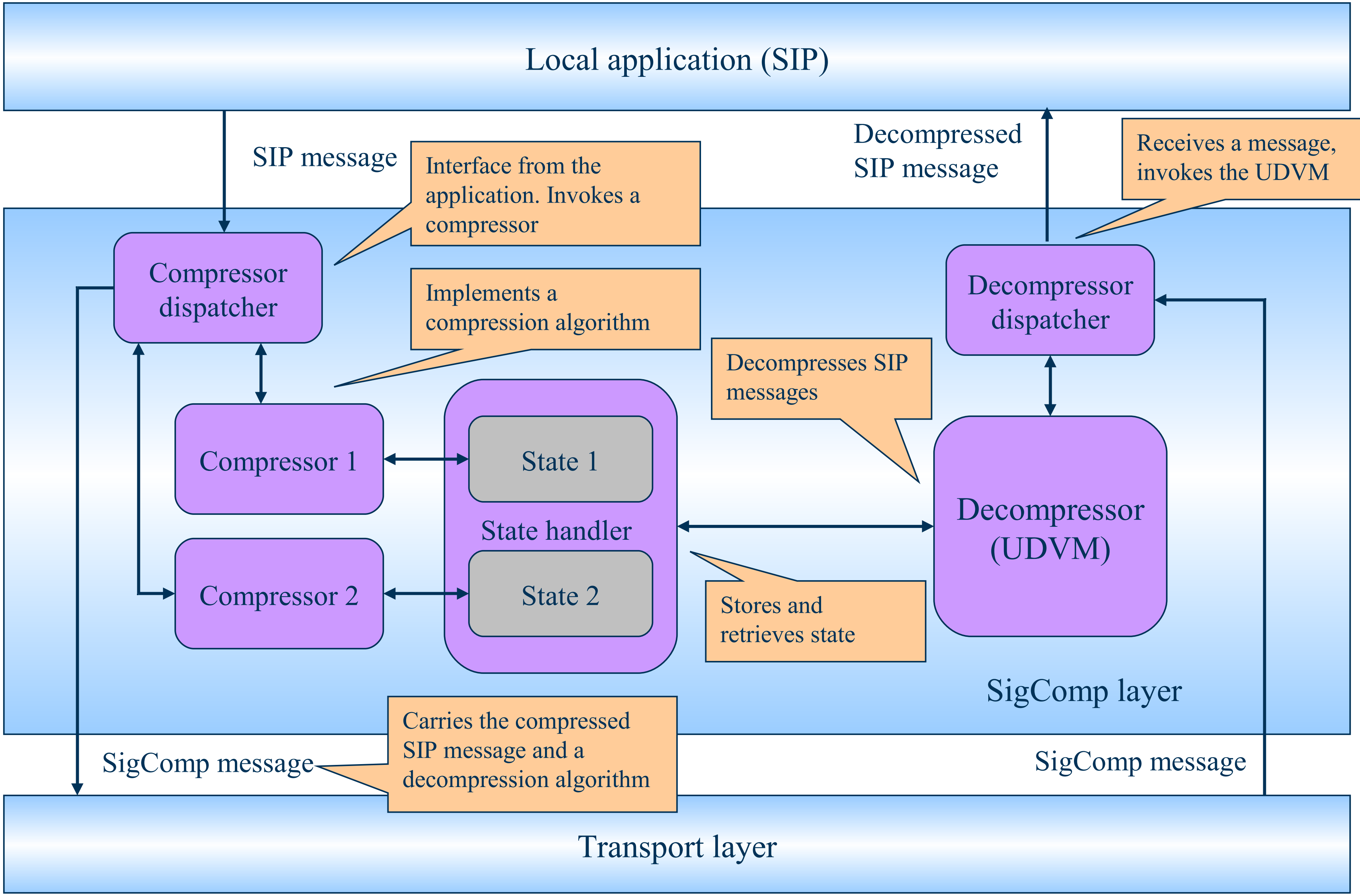
Signaling Compression (SigComp) (1/5)

- SIP is not an efficient protocol regarding message size
  - Problematic e.g., in 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: cb03a0s09a2sdfglkj490333
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=fmtp: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=fmtp:97 mode-set=0,2,5,7; mode-change-period=2
a=rtpmap:96 telephone-event
a=maxptime:20
```

Signaling Compression (2/5)



Signaling Compression (3/5)

- 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

```
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
```

Signaling Compression (4/5)

- 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
- In dynamic compression, compression is done relative to messages sent prior to the current compressed message
- In 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
```

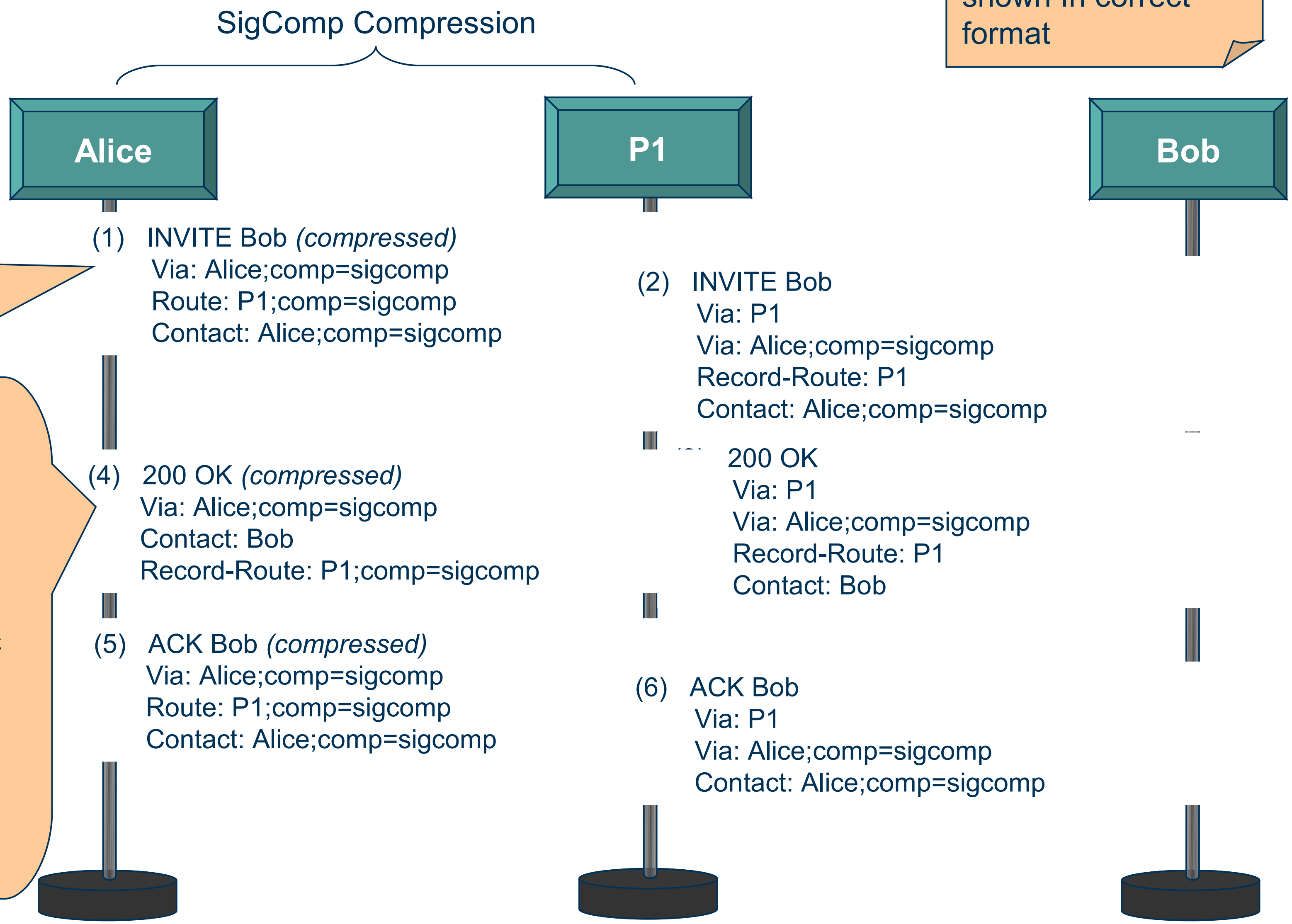
```
SIP/2.0 200 OK
Via: SIP/2.0/UDP p1.domain.com:5060;branch=xyz
       ;received=123.1.0.5
Via: SIP/2.0/UDP c1.domain2.com:5060;branch=abc;
       ;received=123.0.100.4
From: Bob <sip:Bob@domain2.com>;tag=123
To: Alice <sip:Alice@domain.com>;tag=987
Call-ID: 123456789@123.0.100.4
Cseq: 1 INVITE
Contact: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

v=0
o=Alice 2890844545 2890844545 IN IP4 123.0.0.5
s=-
c=IN IP4 123.0.0.5
t=0 0
m=audio 30000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Signaling Compression (5/5)

Note: in order to save space, header fields are not shown In correct format

UAC 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



ERICSSON

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>
```

Inclusion of message/external-body MIME type in Accept header indicates support for content indirection. UAs supporting content indirection must support content indirection of application/sdp MIME type.

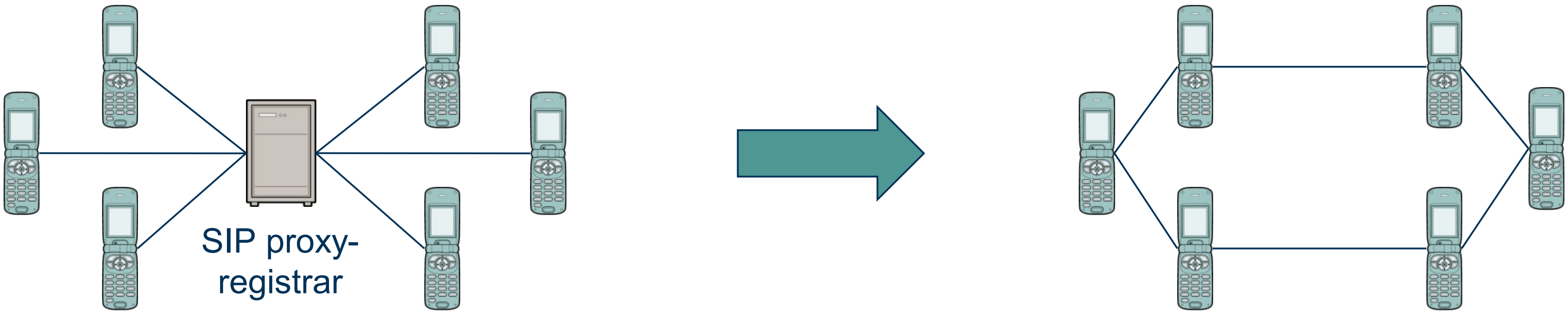
The access-type parameter indicates that the external content is referenced by a URI. The ”expiration” parameter specifies the time period for which the URI is valid.

The purpose of the indirected content. Here it describes a session.

Specifies versioning information for the URI. If the content referred to by a URI changes, the Content-ID must change also.

Peer-to-Peer SIP Overview

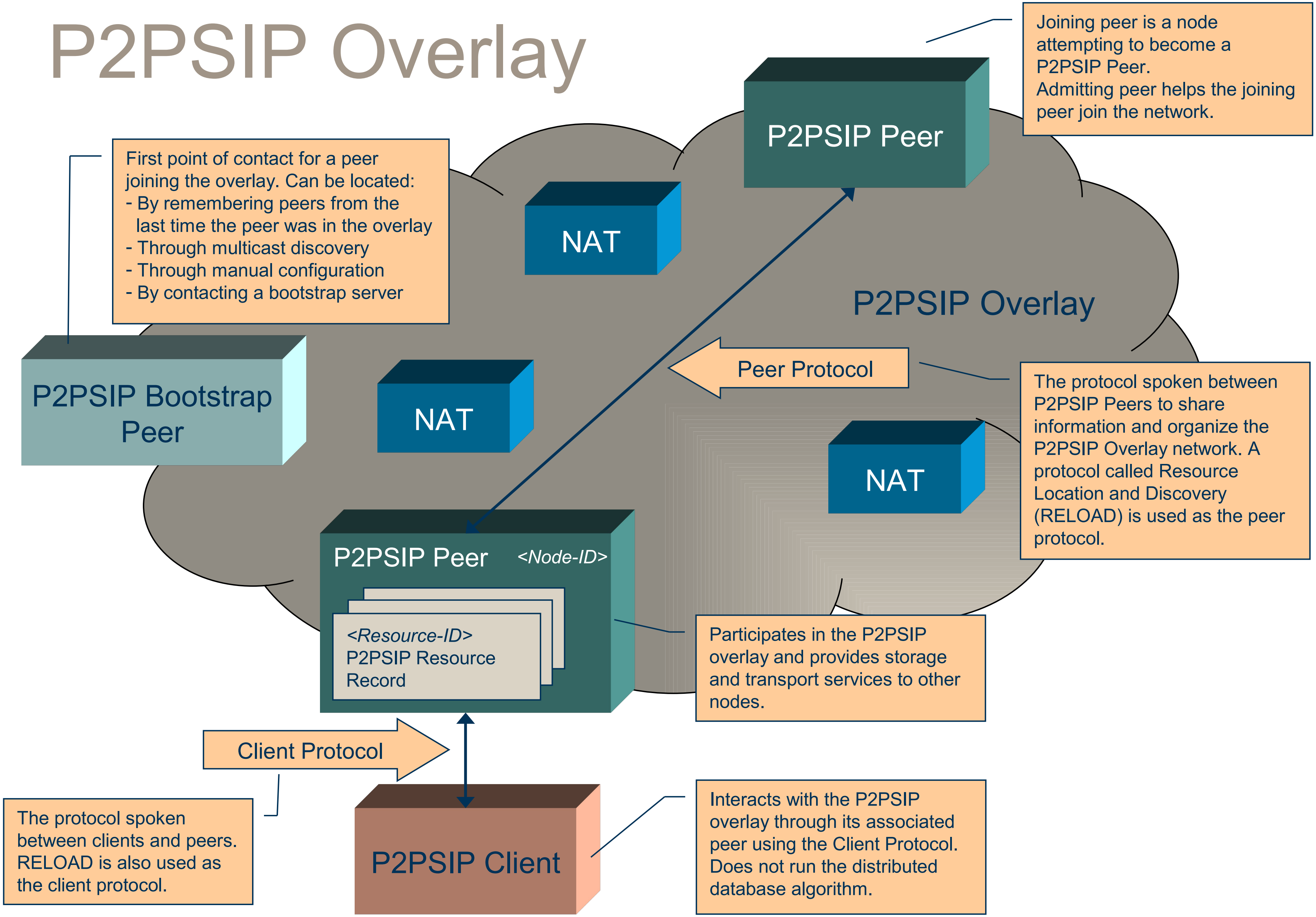
- In conventional client/server SIP, there is a relatively fixed hierarchy of SIP proxies and SIP UAs
- In Peer-to-Peer SIP (P2PSIP), SIP is used in an environment where the traditional proxy-registrar and message routing functions are replaced by a distributed mechanism
  - This mechanism can be e.g., a distributed hash table (DHT)
- In a peer-to-peer (P2P) overlay network, address-of-record to contact URI mappings are distributed amongst the peers in the overlay



Peer-to-Peer SIP in IETF

- The details of P2PSIP are being worked out in the P2PSIP working group of the Internet Engineering Task Force (IETF). The working group will:
  - Define concepts, terminology, rationale, and use cases for P2PSIP
  - Standardize a P2PSIP Peer Protocol
  - Optionally, standardize a P2PSIP Client Protocol
  - Produce 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



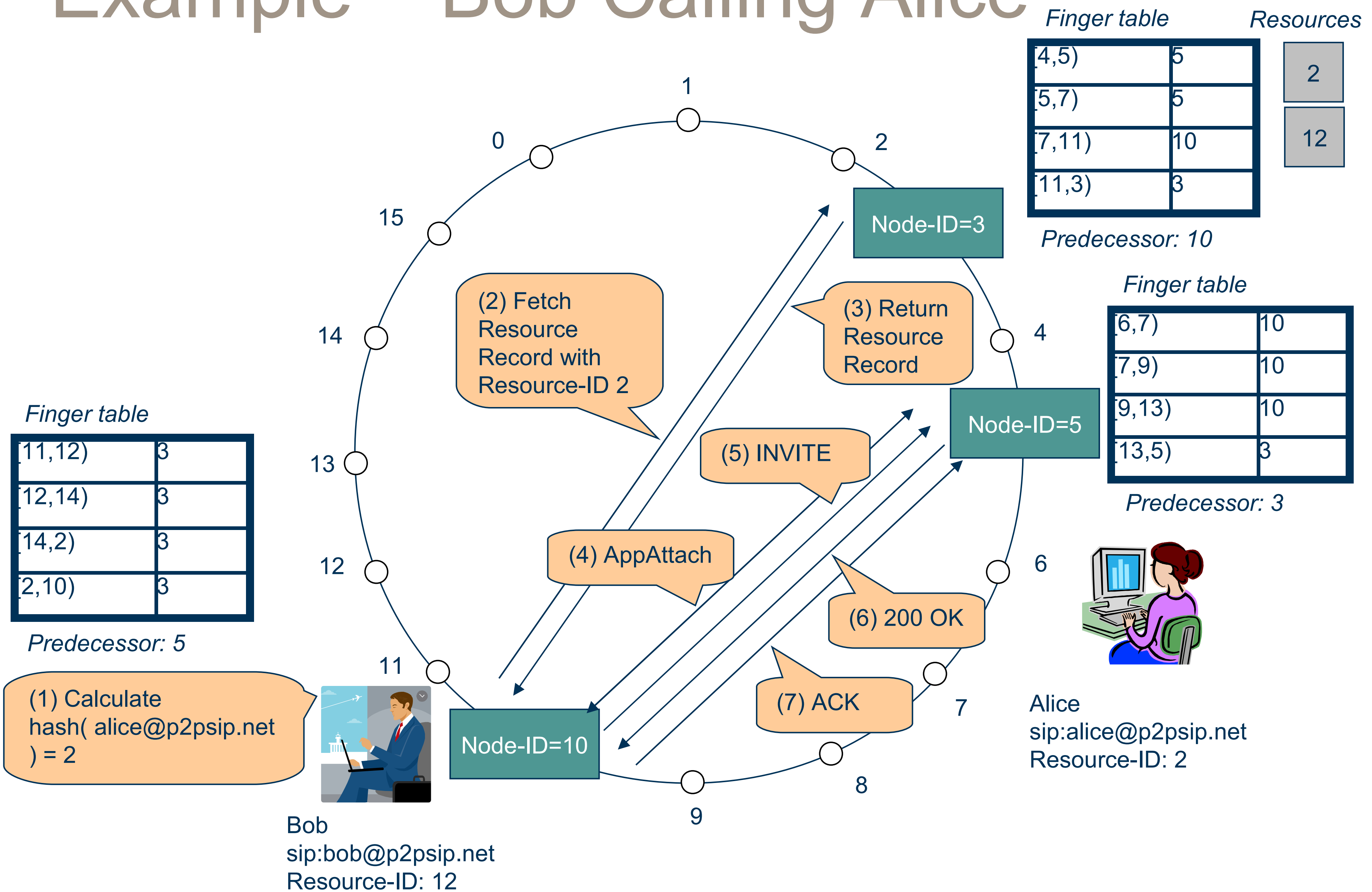
P2PSIP Operations (1/2)

- P2PSIP peers are capable of performing operations such as:
  - Joining and leaving
  - Store and fetch
  - Storing information on behalf of the overlay
  - Transporting messages
- To join a P2PSIP overlay, a joining peer needs to:
  - Contact an overlay configuration server
  - Obtain a certificate and a 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)

- 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
- 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 – Bob Calling Alice



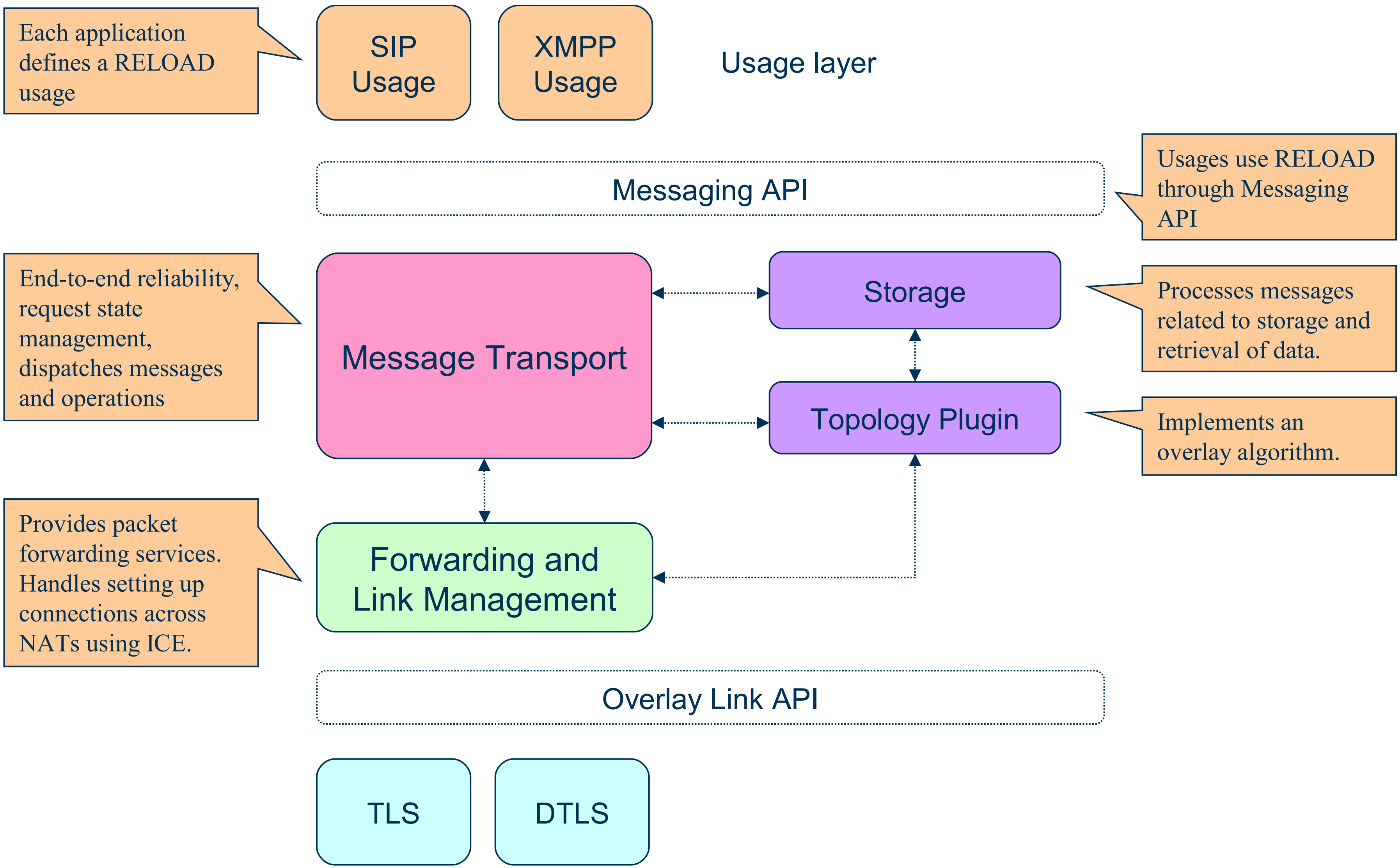
Challenges for P2PSIP

- Security and identity assertion
  - No fully distributed system for security exist which would be as robust as a centralized solution
  - IETF proposes a centralized entity contacted at enrollment time
- Performance
  - Increased resource lookup delay
  - Classical client-server has a *O(1)* lookup cost
  - E.g. Chord has a lookup latency of *O(log(N))*
- Regulatory issues
  - Lawful intercept, emergency calls

Resource Location and Discovery (RELOAD)

- A P2P signaling protocol specified by the P2PSIP working group
- 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

RELOAD Architecture

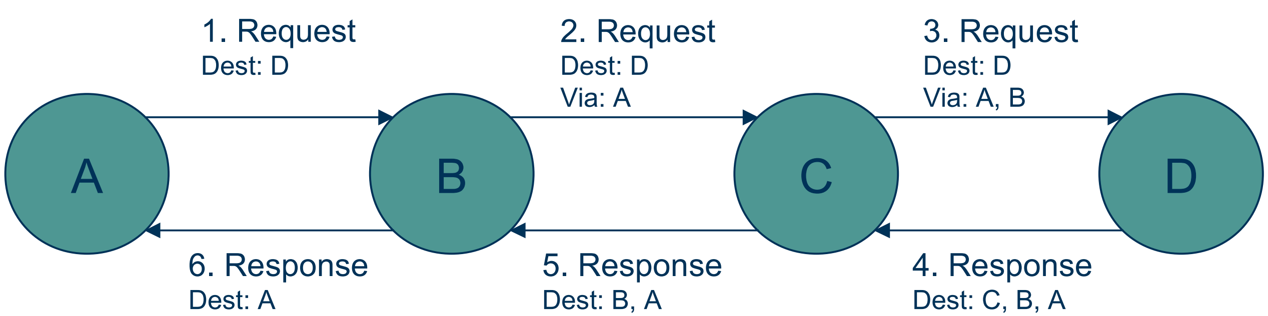


RELOAD Features (1/2)

- 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

RELOAD features (2/2)

- 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

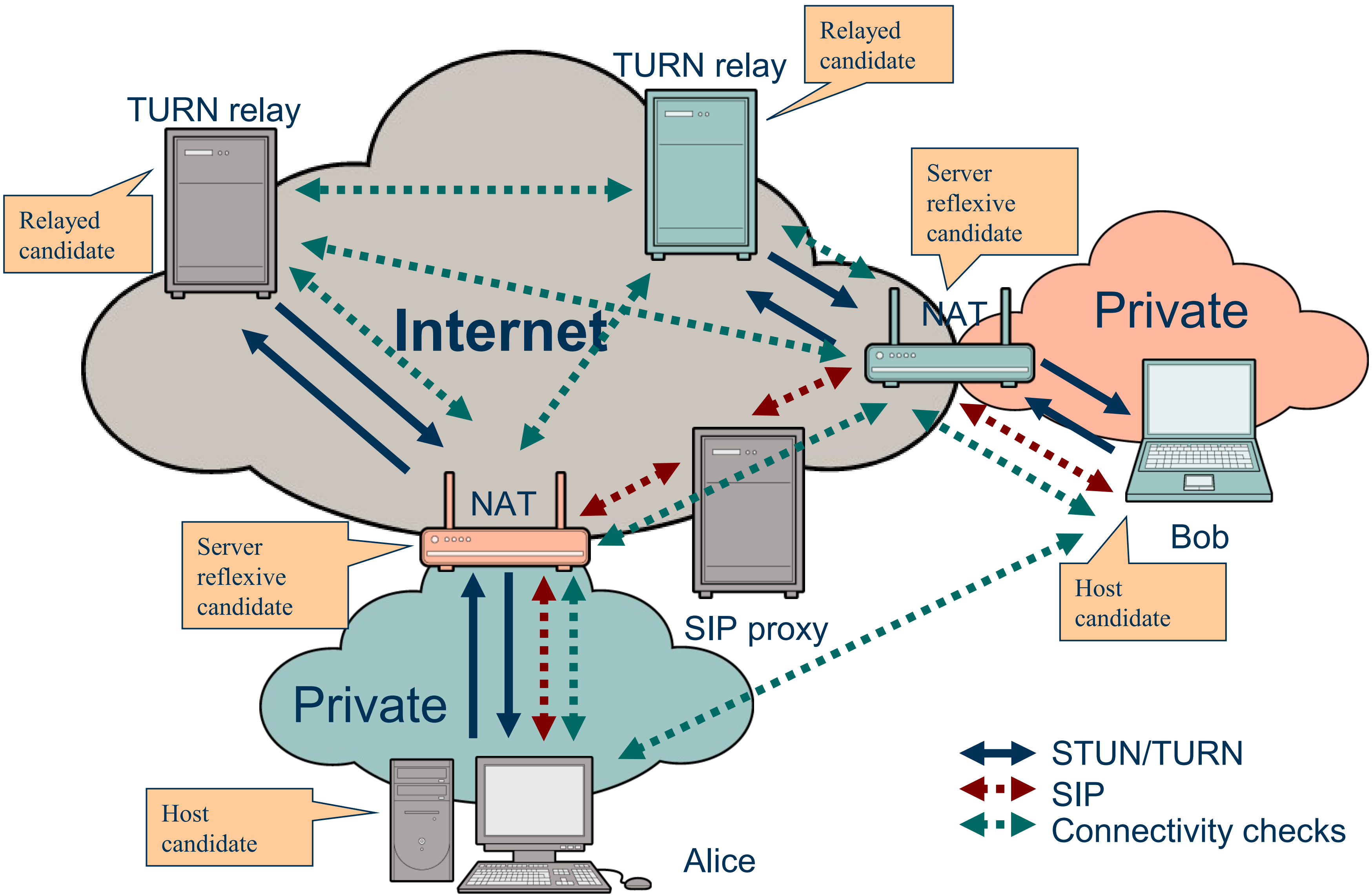


ERICSSON

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
  - Gather candidate addresses
  - Exchange candidates
  - 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



ERICSSON

ERICSSON

TAKING YOU FORWARD