P2PSIP, ICE, AND RTCWEB

T-110.5150
APPLICATIONS AND SERVICES IN INTERNET
OCTOBER 11TH, 2011

JOUNI MÄENPÄÄ
 NOMADICLAB, ERICSSON RESEARCH
AGENDA

› Peer-to-Peer SIP (P2PSIP)
› Interactive Connectivity Establishment (ICE)
› Real-Time Communication between Web browsers (RTCWeb)
› Extending SIP
› SIP extensions
PEER-TO-PEER SIP OVERVIEW

› 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 and Client Protocols
  – 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 Peer**
- Participates in the P2PSIP overlay and provides storage and transport services to other nodes.
- Interacts with the P2PSIP overlay through its associated peer using the Client Protocol. Does not run the distributed database algorithm.

**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

**NAT**

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

**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.

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

**Enrollment Server**
- Helps in the enrollment process of the joining peer.
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**
    - To 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
/// 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**: hash(*alice@example.com*) = 32B4A7F02C
/// - Locate the peer that is responsible for that Resource-ID
/// - Store a <Resource-ID, Node-ID> 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
///   \[\text{hash}(\text{alice@example.com}) = 32B4A7F02C\]
/// - Locate the peer that is responsible for that Resource-ID in the P2PSIP overlay
///   \[\text{A P2PSIP Resource Record with contact information is obtained: alice@example.com} \rightarrow \text{Alice’s Node-ID}\]
/// - 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

Alice
sip:alice@p2psip.net
Resource-ID: hash(alice@p2psip.net) =2
Node-ID: 11

Bob’s Resource Record
Resource-ID: 4
Content: Bob’s Node-ID=15
EXAMPLE: ALICE CALLING BOB (2/3)

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

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

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

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

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

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

3. INVITE

(1) SIP INVITE

(2) SIP 200 OK

(3) SIP 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.

- **SIP Usage**
- **XMPP Usage**

**Usage layer**

**Message Transport**
- Processes messages related to storage and retrieval of data.
- Implements an overlay algorithm.
- Provides packet forwarding services. Handles setting up connections across NATs using ICE.

**Messaging API**
- Usages use RELOAD through Messaging API

**Storage**

**Topology Plugin**

**Forwarding and Link Management**

**Overlay Link API**
- TLS
- DTLS
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

1. Request
   Dest: D

2. Request
   Dest: D Via: A

3. Request
   Dest: D Via: A, B

4. Response
   Dest: C, B, A

5. Response
   Dest: B, A

6. Response
   Dest: A
INTERACTIVE CONNECTIVITY ESTABLISHMENT (ICE)
NETWORK ADDRESS TRANSLATION (NAT)

› Network Address Translation (**NAT**)  
  - Mapping of IP addresses from one realm to another  
  - E.g., connect an isolated address realm with **private addresses** to an external realm with **globally unique addresses**  
  - Thanks to NAT, a host in a private network can transparently communicate with destinations on an external network  
    › And vice versa  

› Types of address and port **mapping**  
  - Endpoint independent mapping  
  - Address dependent mapping  
  - Address and port dependent mapping  

› Types of **filtering**  
  - Endpoint-independent filtering  
  - Address-dependent filtering  
  - Address and port dependent filtering
SIP, RELOAD, and RTCWeb use Interactive Connectivity Establishment (ICE) to set up connections across NATs.

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
- Obtain a relayed address
- Control the operation of a relay

ICE is used to discover a working path through NATs
- (1) Gather candidate addresses
- (2) Exchange candidates
- (3) Perform connectivity checks
COMMUNICATION SCENARIO FOR ICE
NAT TRAVERSAL FOR MEDIA IN SIP (2/2)

› 1-2: Alice gathers ICE candidates
› 3-5: Alice sends her ICE candidates to Bob
› 6-7: Bob gathers ICE candidates
› 8-11: Bob sends his candidates to Alice
› 12: Alice and Bob perform ICE connectivity checks
› 13: ICE has found a working path, RTP media starts flowing between Alice and Bob
REAL-TIME COMMUNICATION BETWEEN WEB BROWSERS (RTCWEB)
Voice and video telephony and conferencing in **HTML5**
- HTML5: the 5th revision of the HTML standard
- Interoperable, no plugins required

Some aspects of video conferencing in HTML5
- Getting multimedia streams from local devices
- Recording streams locally
- Connecting to remote peers using NAT traversal
- Sending streams to remote peers and receiving streams
- Displaying the streams using HTML5 `<video>` or `<audio>` elements
- Sending arbitrary data to remote peers

**RTCWeb** WG in the **IETF**
- Scope: the protocols that browsers talk to each other
- For WG charter, see [1]

**WebRTC** in **W3C** (World Wide Web Consortium)
- Scope: APIs that are offered to Javascript applications to take advantage of the browser’s functionality
- For current API draft, see [2]
RTCWEB

› IETF RTCWeb WG focuses on the protocols
› Functionality groups
  – Data transport – sending and receiving data, NAT traversal
  – Data framing – RTP and SRTP (Secure Real-Time Protocol)
  – Data formats – codecs, format specifications
  – Connection management – setting up, negotiating, and tearing down connections
  – Presentation and control – W3C API effort, user control over browser’s interaction with input/output devices
  – Local system support functions – e.g., echo cancellation, volume control
RTCWEB TRAPEZOID

WEB SERVER

SIGNALLING PATH (SIP)

WEB SERVER

WEB

WEB

PROPRIETARY OVER HTTP/WEBSOCKETS

PROPRIETARY OVER HTTP/WEBSOCKETS

JS/HTML/CSS

JS/HTML/CSS

BROWSER

BROWSER

ALICE

BOB

MEDIA PATH

PEERCONNECTION
**CALL ESTABLISHMENT IN RTCWEB**

1. **Download** a video communication web application (Javascript)
2. **Open a PeerConnection** (among other things)
   - Allows two users to communicate directly, browser-to-browser
   - `new PeerConnection(configuration, signalingCallback)`
     - `configuration`: address of a STUN/TURN server
3. **Use a signaling protocol** over bidirectional HTTP or WebSocket to talk to server
   - Bidirectional HTTP: e.g., long polling, HTTP streaming
   - WebSocket: bi-directional, full-duplex communication channel over a single TCP socket
     - Implemented in web browsers and web servers
   - The signaling protocol could be a subset of SIP
   - Support for SDP and offer/answer model is mandatory
     - ICE candidates in SDP
4. **Servers** may talk **SIP** to each other
5. **Media path** directly between browsers
   - Over PeerConnection
   - ICE negotiation
   - RTP (Real-Time Protocol) for media transport
REFERENCES

› [1] RTCWeb charter
   - http://tools.ietf.org/wg/rtcweb/charters

› [2] RTCWeb API
   - http://dev.w3.org/2011/webrtc/editor/webrtc.html