DISTRIBUTED VOICE SERVICE

Martin Jaroš

Master Degree Programme (1), FEEC BUT

E-mail: xjaros32@stud.feec.vutbr.cz

Supervised by: Václav Zeman

E-mail: zeman@feec.vutbr.cz

Abstract: This work proposes a communication protocol and its reference implementation for an Internet telephony service based on a distributed architecture. The service provides a secure peer to peer voice streaming network between its clients. The implementation focuses on support of embedded devices.

Keywords: Voice over IP, Distributed hash table, audio streaming

1. PREFACE

The primary idea behind this service is a decentralized communication network between its clients. This means there are no requirements to run high performance dedicated servers, the service is more robust against network failures and more secure as it does not rely on any external resources. User experience should not be altered as on the user level the service behaves as any other voice service. The protocol is very lightweight in comparison to a classical VoIP protocol suite and easy to implement. All external communication is done over a single UDP socket, there are three layers of the protocol. The routing layer creates a logical overlay network over the physical IP network and handles all the low level networking, nodes are addressed by keys in this overlay. The session layer handles the communication between logical nodes, verifies keys and creates secure logical tunnels. The media layer handles media streaming over the logical tunnels and all low level media compression and interfacing.

2. OVERLAY NETWORK

The figures 1 and 2 show the network abstraction created by the routing layer. The underlying network socket is subjected to a dynamic address translation, in the overlay network the nodes are addressed statically by their public keys and any node may open a communication channel to any other node if its ID is provided. The mapping of the IDs to the underlaying network addresses and port numbers are done by a distributed hash table [1].

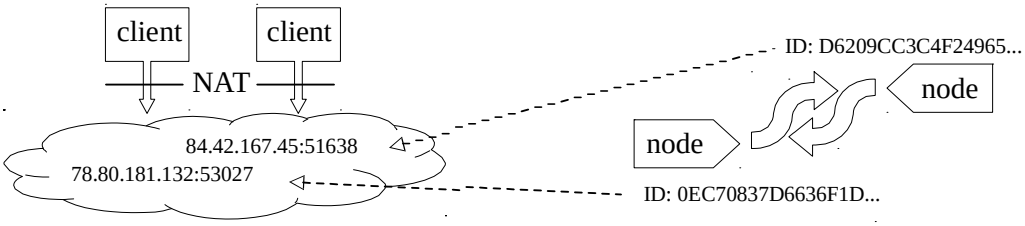

Figure 1: Physical IP network

Figure 2: Logical overlay network

The distributed hash table implementation uses a XOR based metric with the SHA-1 hash method, the resultant address space is therefore 160 bits large with 20 entries per bucket. The XOR metric is symmetrical and has the triangle property:

$$x \oplus x = 0,\ x \oplus y = y \oplus x,\ x \oplus y + y \oplus z \geq x \oplus z\,. \tag{1}$$

The routing table resembles a binary search tree with a lookup complexity of O(log n). The table is subdivided into buckets based on the prefix, for example if the user's ID starts with the 0110 binary sequence, then the first bucket will contain the 1* address space, second bucket the 00* address space, the third bucket the 010* address space and the fourth bucket the remaining 011* address space. If there is at least one route entry in each bucket, then the lookup will converge; the route entry is defined as an IP address + user ID combination. The lookup is performed recursively using a single request / response plain text message containing the origin ID, the destination ID and in the response a list of k closest route entries, where k is the bucket size. The relative distance is defined by the metric. The lookups are done in parallel and the routing table is updated based on the flowing traffic. Full-cone NAT traversal is possible as the translation mapping is held for a single source address and port tuple. Given the nature of the hash function, the routing layer is anonymous, does not leak any personal information and may be shared across implementations. The peer's identity is verified at the end of the lookup process during the key exchange process. In order to increase stability and to have some protection against address spoofing, table entries are prioritized based on uptime.

3. SECURITY CONSIDERATIONS

Each peer is defined by its public key represented as a X.509 [2] SubjectPublicKeyInfo structure. The peer's unique ID is a SHA-1 digest of this structure in DER encoding, this is the ID used for addressing. The actual key algorithm may be any one supported by the standard, either RSA, DSA or ECDSA defined by the structure. During session initialization the Diffie-Hellman key exchange is performed, the implementation defaults to elliptic keys over the secp256r1 curve for both signing and key derivation. The key exchange message contains two SubjectPublicKeyInfo structures in DER encoding, the first is for key derivation, the second is the peer's public key. The message is appended by a signature as shown in the figure 3. Peer's public key is verified against its SHA-1 digest, then the signature is verified and a shared encryption key is calculated.

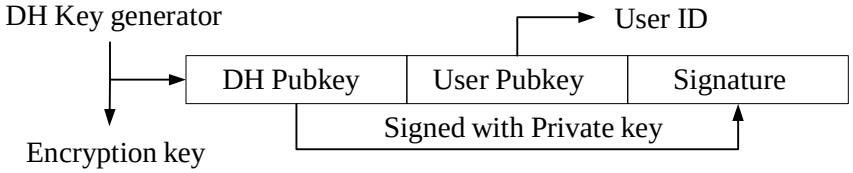

Figure 3: Key exchange

After the successful authentication a virtual stream is created for the media transfer, this stream is encrypted with an AES-256 cipher in Galois counter mode. The cipher key is a SHA-256 derivation of the shared secret, the counter IV is a sequence number of the packet. The sequence number is also used for packet reordering and loss detection. The packet contains the 32bit sequence number, the 64bit authentication tag and a variable length ciphertext, each packet is authenticated by the tag.

4. MEDIA

The media implementation focuses on a high quality and low latency streaming with adaptive bitrate control, main media type is an audio stream. The audio stream uses a hybrid cosine transform and linear prediction coding as defined in the Opus standard [3]. There is the SILK speech coder, the CELT low latency audio coder and a packetizer. The source stream uses fullband

48kHz sampling with 20ms buffer time, the bitstream uses variable bitrate from 6 to 510 kbps. The bitstream is self descriptive with no additional data required, the bandwidth and bitrate allocation

may change dynamically on the fly and the receiver may decode any packet individually and

replace missing packets with loss concealment. The implementation uses ALSA interface for hardware abstraction, the overall latency is nominally bellow 50ms, packet sizes are typically around 80 bytes per 20ms for high quality voice streams. Other media types such as video or

instant messages are reserved for future implementations. The complete payload overhead is below 14%; if the UDP+IPv4 header is also taken into account, the overhead gets close to 30% which is still reasonably good in comparison to the classical VoIP protocols such as SRTP.

5. IMPLEMENTATION

The implementation is divided into two separate processes, the service daemon and the user client. This partitioning allows modular deployment without reliance on the user-space environment. The service is build on top of the Linux kernel for portability on embedded platforms using the GNU build system. Libraries used are libasound for sound architecture abstraction, libcrypto (which is a part of the OpenSSL project) for cryptography and libopus for audio coding. The system service is implemented as an event driven static state machine, the client is connected via a system pipe. The reference implementation uses wrapper for a websocket interface over the control pipes with web based GUI created as a HTML5 / Javascript page for portability, native user interfaces are also possible. The graphical user interface resembles traditional IM voice client interface, hiding the underlaying distributed topology. The figure 4 shows a typical call session, all signaling is done in-band within the encrypted logical stream. The user A dials the ID of the user B, the routing layer performs a distributed table lookup and the session layer authenticates the logical stream. Then the classical ring / answer session handling takes place followed by media stream.

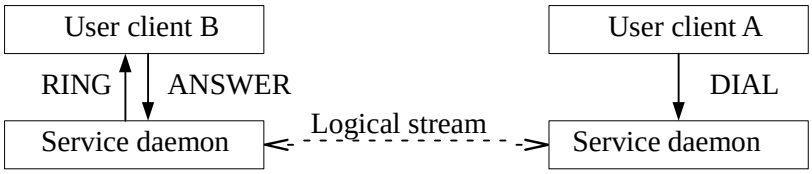

Figure 4: User interactions

6. CONCLUSION

The implementation was tested on PC and various embedded systems such as Beaglebone or the popular Raspberry Pi with great success due to the Linux kernel abstractions. The source code is openly distributed using the Git versioning tool [4] under the GPL license. The proposed service has high performance, simulations have shown that even with very high number of clients (hundred thousands) the lookup latency is still feasible in comparison to classical server-client based systems. The distributed hash table implementation provides robust topology resistant to network failures or denial of service attacks. The protocol provides perfect forward secrecy for its peers with relatively low transport and computational overhead. The Opus codec is a state of the art speech and audio codec with high quality and low latency, the implementation has optimizations for both the x86 and ARM architectures.

REFERENCES

[1] GHODSI, Ali. *Distributed k-ary System: Algorithms for Distributed Hash Tables.* Stockholm: The Royal Institute of Technology, School of Information and Communication Technology, 2006. Available: https://www.sics.se/~ali/thesis/dks.pdf

[2] Network Working Group. *RFC 5280: Internet X.509 Public Key Infrastructure Certificate*. The Internet Engineering Task Force, 2008. Available: https://tools.ietf.org/html/rfc5280

[3] The Internet Engineering Task Force. *RFC 6716: Definition of the Opus Audio Codec*. The Internet Engineering Task Force, 2012. Available: https://tools.ietf.org/html/rfc6716

[4] JAROS, Martin. Distributed voice service. [online]. 2014. Available: https://github.com/martinjaros/dvs