Design of a SIP Outbound Edge Proxy (EPSIP)

Sixth FRUCT seminar
Helsinki, Finland on 3-6 November 2009.

Sergio Lembo

Department of Communications and Networking (TKK)

Jani Heikkinen, Sasu Tarkoma

Department of Computer Science and Engineering (TKK)

Scope of work

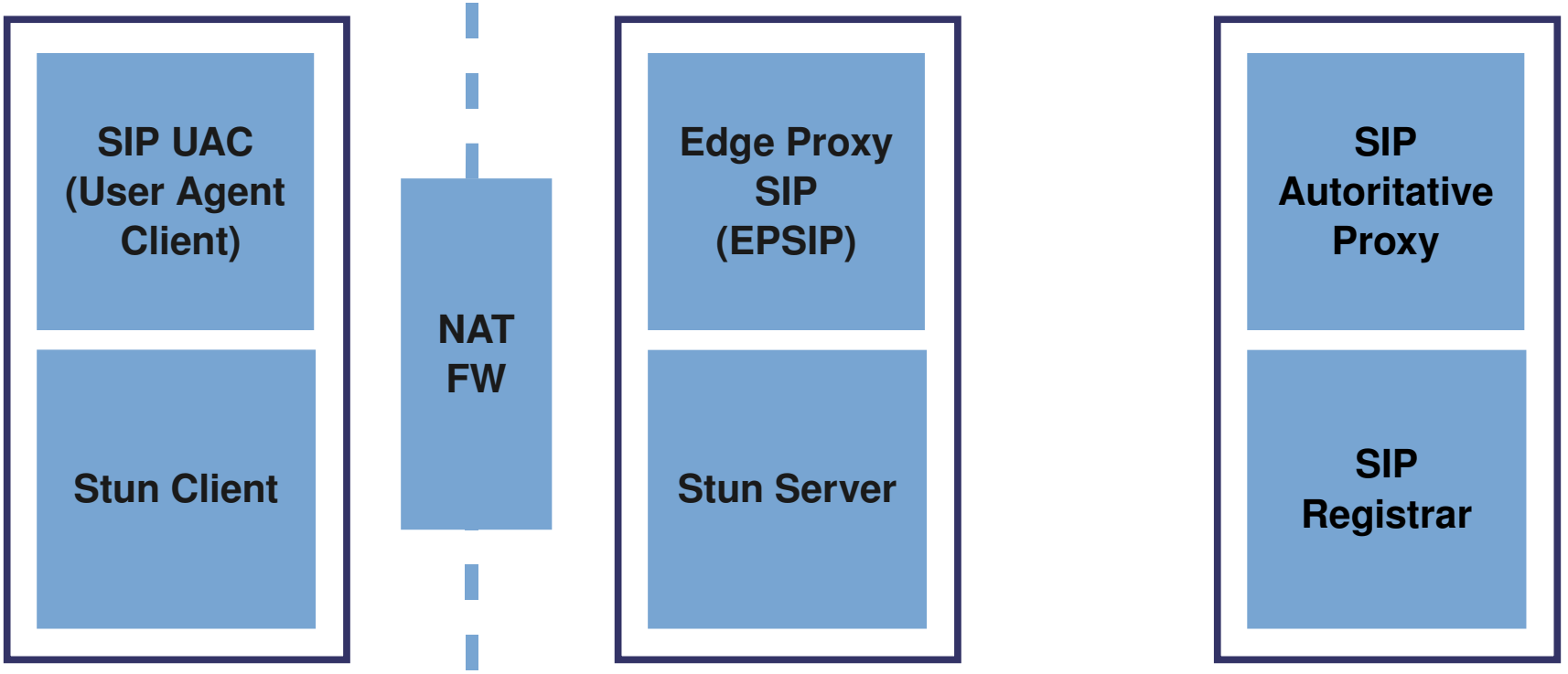
- Design of a SIP Outbound Edge Proxy that follows the requirements stated in IETF Internet-Draft SIP Outbound.

- Ready-to-implement design, thus bridging the gap between the requirements in the draft and an actual and feasible implementation of these requirements in a real system.

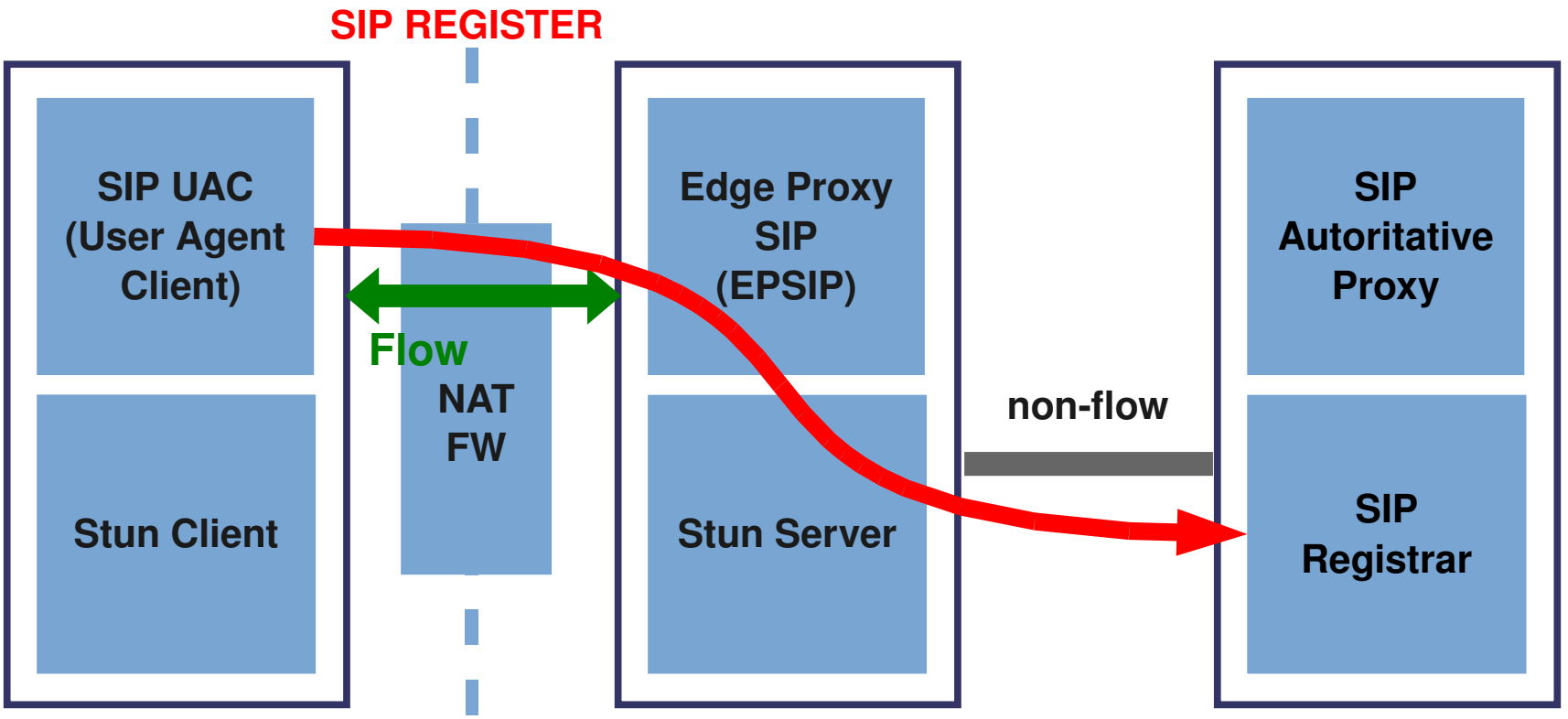
Introduction

- SIP (Session Initiation Protocol)
- Without special considerations SIP User Agents behind a Firewall or Network Address Translator (NAT) are unable to receive incoming SIP requests due to the presence of these network elements.
- We focus here on a particular solution provided by an IETF draft:
  - *Internet-Draft SIP Outbound*.
- SIP Outbound enables incoming SIP requests to a User Agent behind a Firewall or Network Address Translator (NAT) with a mechanism that requires a particular kind of SIP proxy, namely, SIP Edge Proxy.
- We present a design of such SIP Edge Proxy conforming to SIP Outbound draft
  - EPSIP (Edge Proxy SIP).

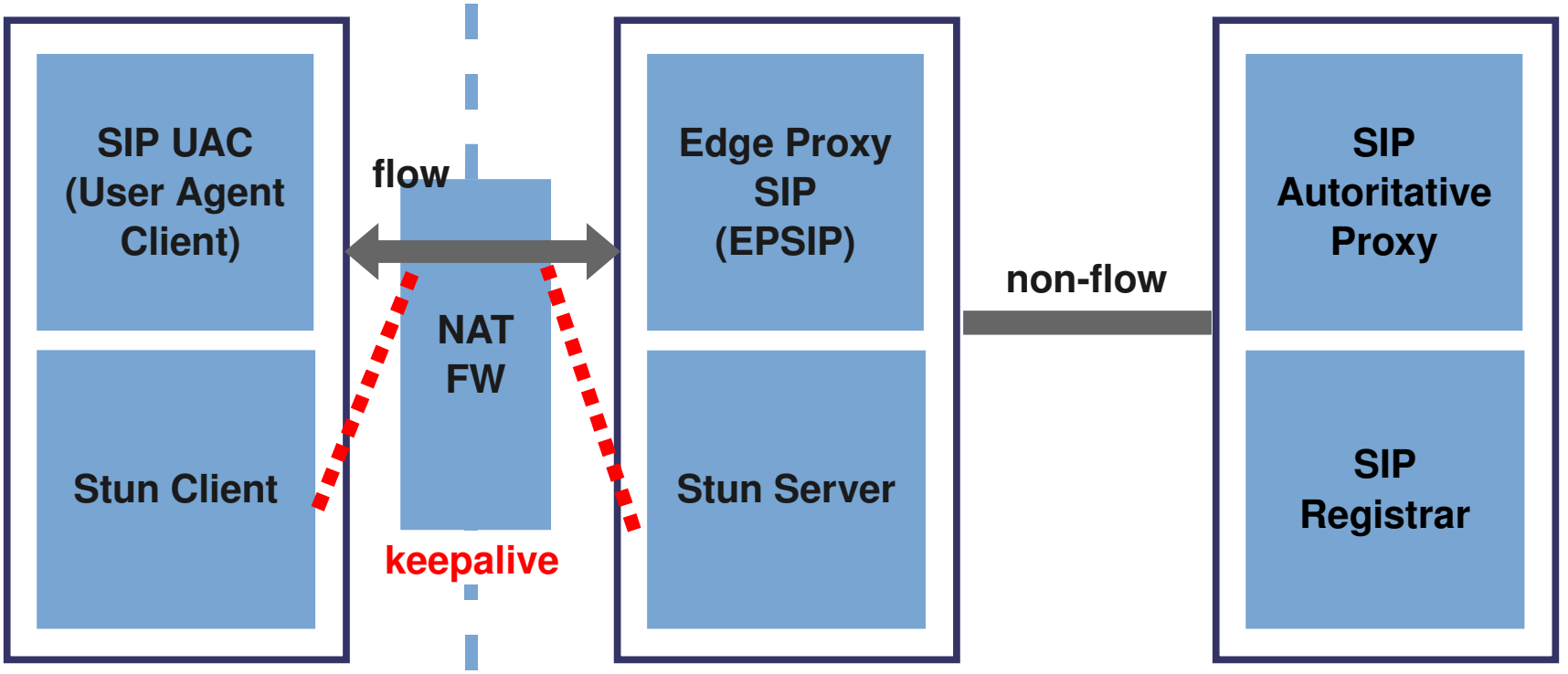
SIP Outbound and Functionality of a SIP Outbound Edge Proxy



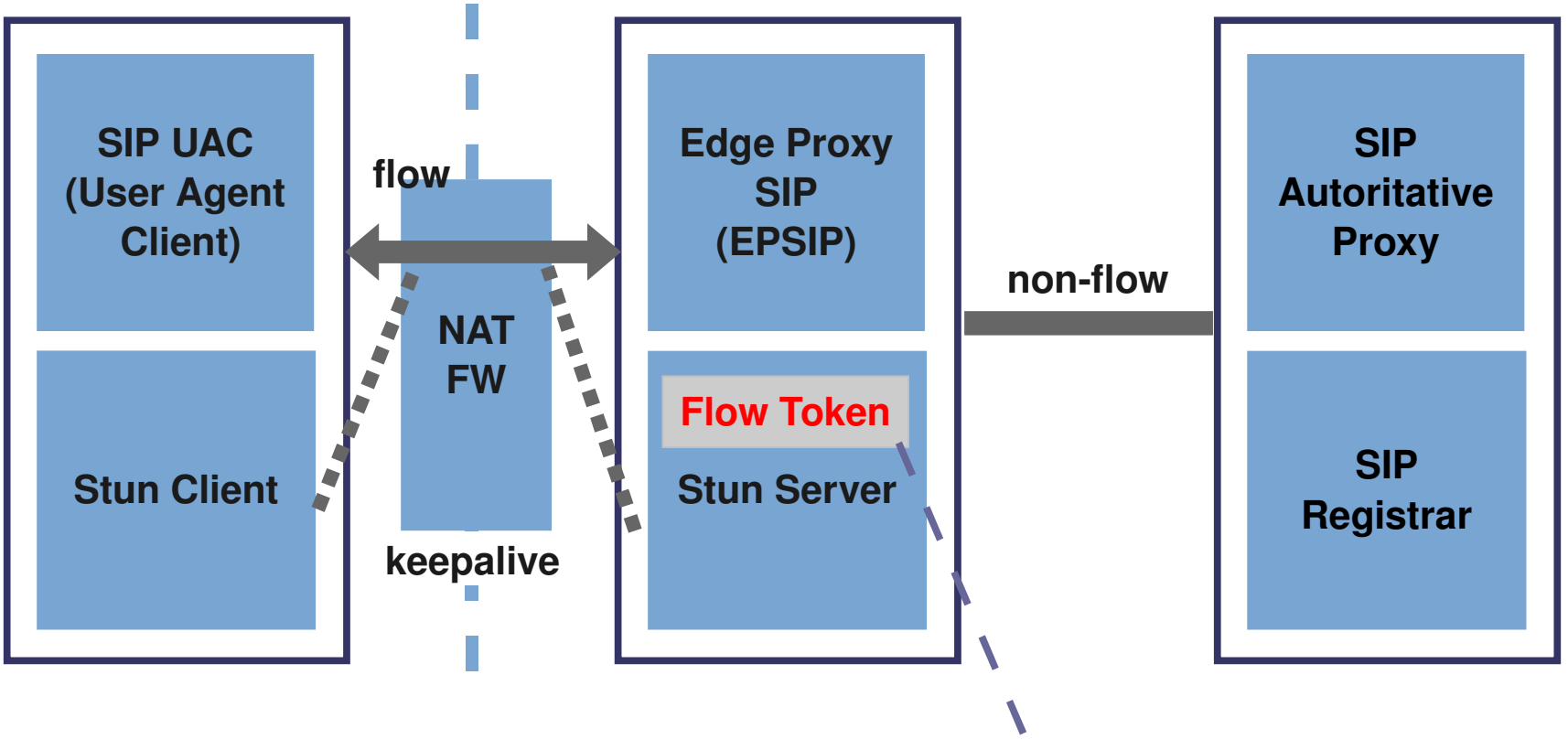
SIP Outbound and Functionality of a SIP Outbound Edge Proxy



SIP Outbound and Functionality of a SIP Outbound Edge Proxy



SIP Outbound and Functionality of a SIP Outbound Edge Proxy



- *Flow Token*
- (To map future requests back to the correct flow)

Scope of the design

- Fulfill SIP Outbound draft.
- Working model of SIP Outbound
- Suitable to handle several transport protocols
- Be relatively more than a proof of concept
  - serve as a basic prototype for real production scenarios.
- Main principles:
  - handle flows by processes
  - map flow-tokens to processes through local sockets

Principles in the design of a SIP Outbound Edge Proxy

- The proxy should be based on a stateless SIP proxy.
- The architecture must be capable to maintain TCP connections in time.
- TCP connections (flows) must be managed with a concurrent approach.
  - associate a connection to a process or thread that will be responsible to handle the particular connection (flow).
  - In this sense we introduce the concept that the proxy will have one process or thread running per established flow.
- Messages to be forwarded over a TCP connection (flow) must reach the corresponding flow by a suitable mechanism that maps the content in the flow-token to the module in charge of the TCP connection.
  - processes or threads must be reached appropriately when it comes the time to forward an arriving message over a flow
  - i.e the content of the flow-token must be mapped to some reference that will make the message reach the appropriate process or thread in charge of the flow.

Transport layer architecture design

- The design presented here is based on implementing concurrency in the proxy by using multiple processes.

- Main process

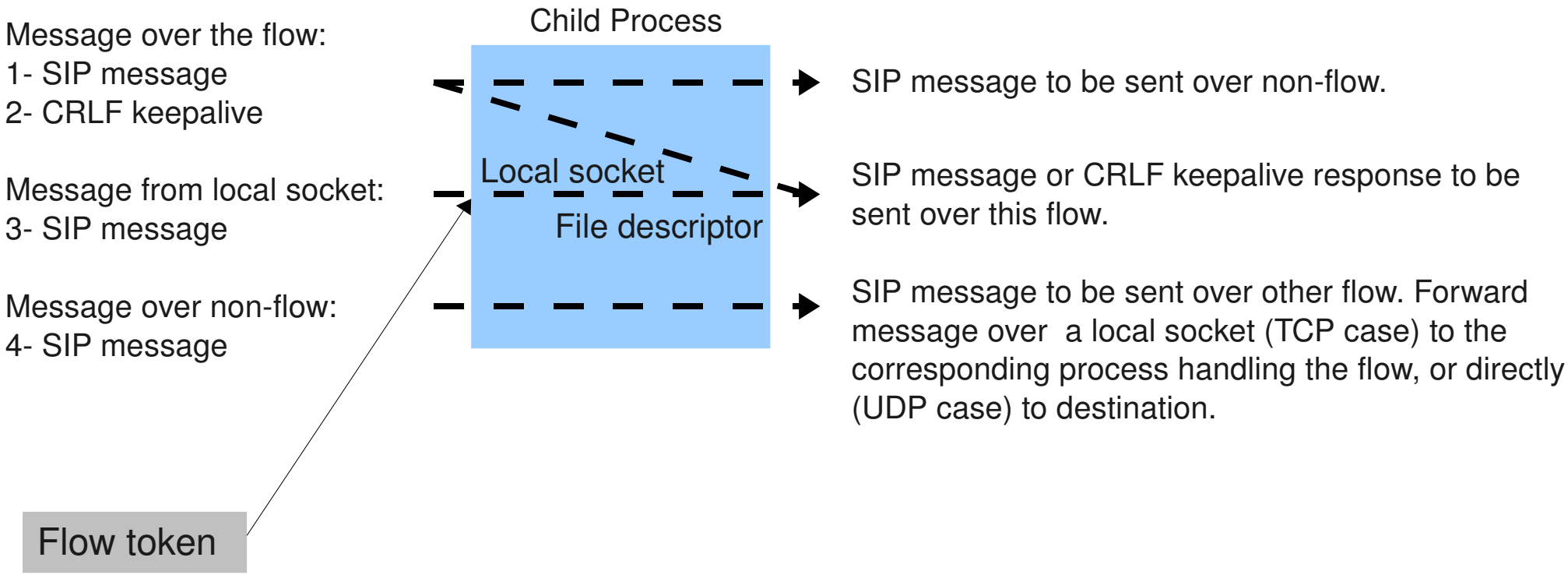
Listening for incoming messages:

- Messages over connectionless transport protocols (UDP flows)
- New messages over connection-oriented transport protocols (TCP flows)

The system will have one process per flow

Transport layer architecture design

- Child processes to handle connections for TCP transport protocol.



Transport layer architecture design

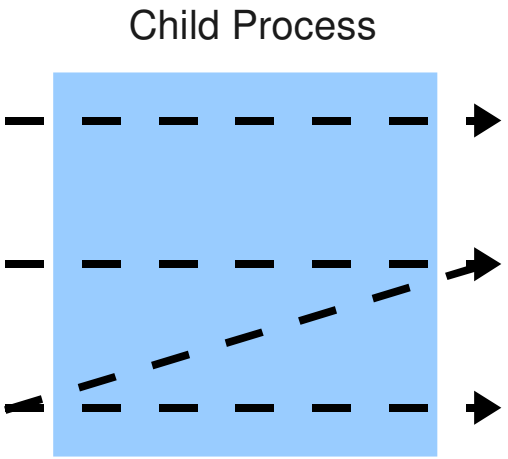
- Child processes to handle datagrams for UDP transport protocol.

Message over the flow:

1- SIP message

2- STUN keepalive

Message over non-flow:
3- SIP message

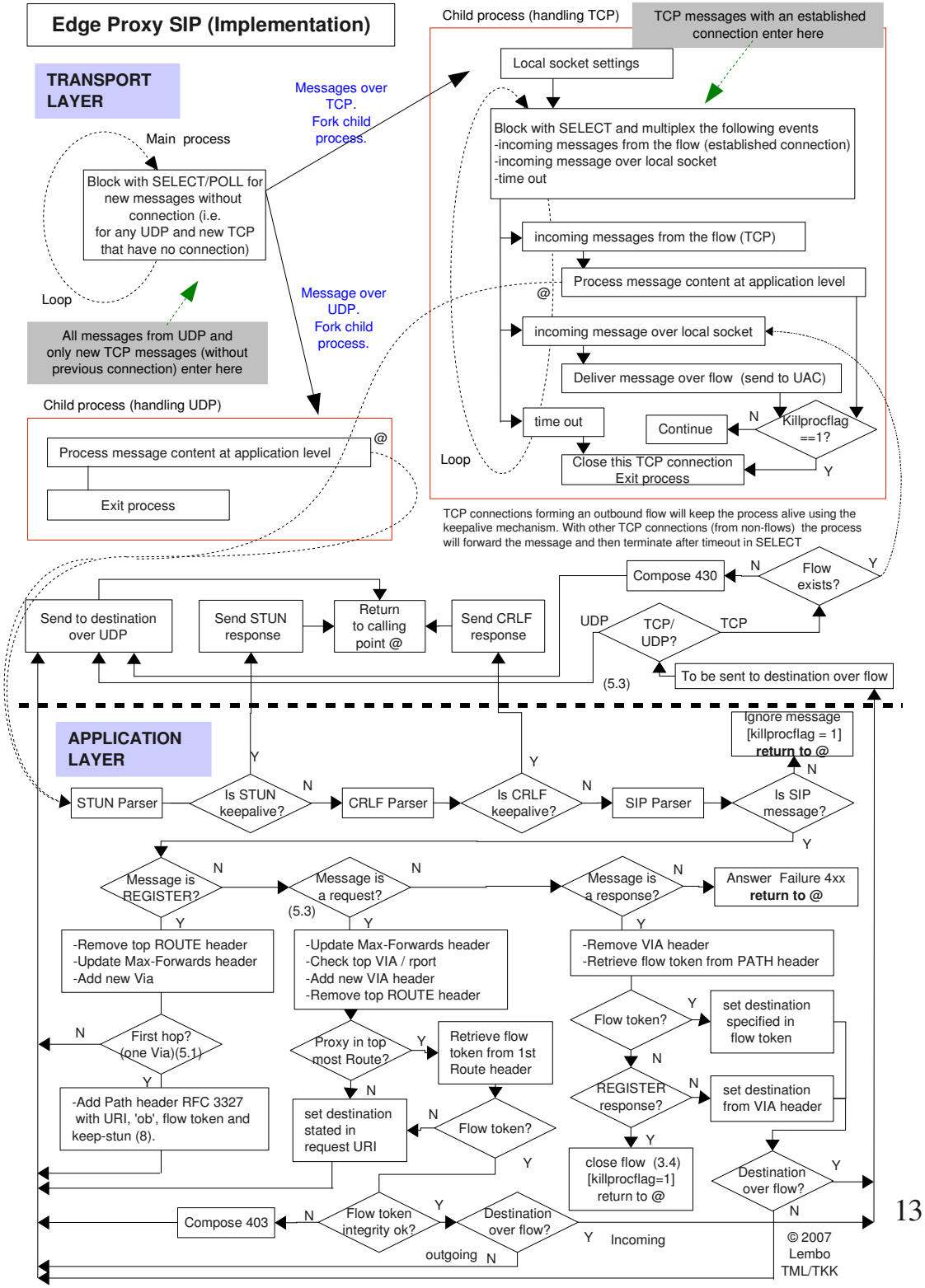


SIP message to be sent over non-flow.

SIP message or STUN response to be sent over a UDP flow.

SIP message to be sent over a TCP flow.
Forward message over a local socket to the corresponding process handling the flow.

Transport and application layer design



Conclusions

- We designed a SIP Outbound Edge Proxy and :
  - introduced the concept of using a dedicated process per flow.
  - proposed a method to map a flow-token to a TCP connection
    - Relating a process to a local socket with a name equivalent to the information stored in the flow token.
    - Relating a file-descriptor to a process
  - the design implements a multiple-process concurrent approach to handle independently multiple flows.
  - composed a flow chart that resumes in one figure the design of the proxy and SIP Outbound logic.
- The proxy is suitable also as a multi-transport-protocol SIP proxy with the advantage of offering a design that contemplates the use of transport oriented protocols.
- The design was implemented in a real proxy and we successfully verified its behavior in a test scenario.