RELOAD – Usages for P2P Data Storage and Discovery

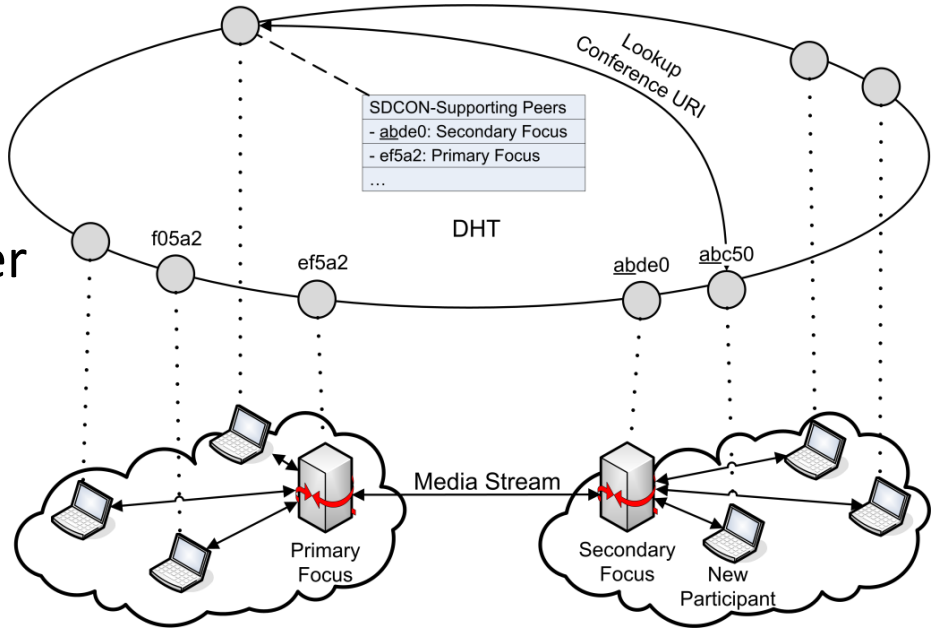
AW2-presentation from

Alexander Knauf

Alexander.Knauf@Haw-Hamburg.de

Review

- Infrastructure Independent Conferencing:
  - Decentralized fashion
  - Multiple conference controller
  - Signaling with SIP
  - Connected through DHT
  - Proximity-aware
  - Mesh building



Outline

- Introduction into RELOAD
- RELOAD Usages
  - TURN advertisement
  - SIP Registration
  - Service Discovery
- Conclusion & Outlook

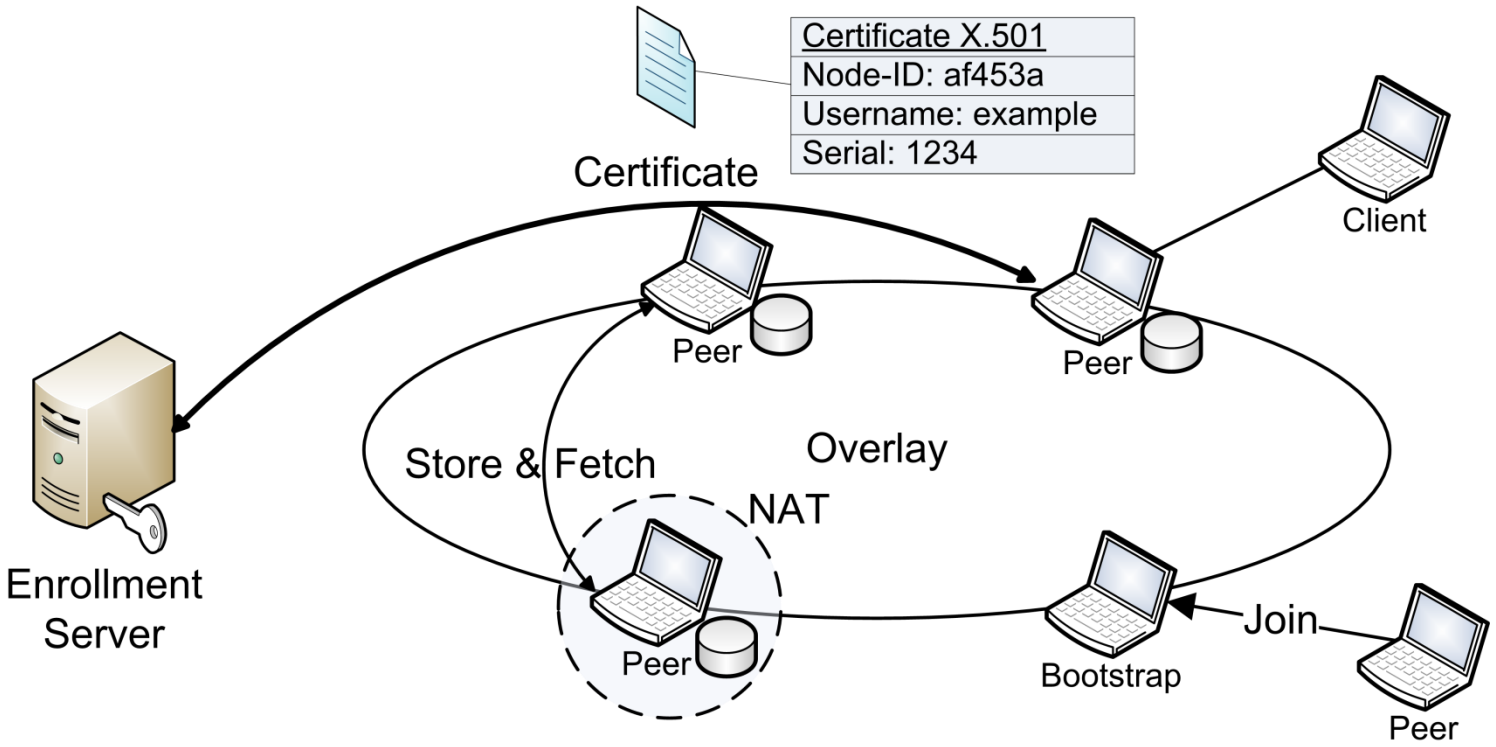
RELOAD Overview(1)

- RELOAD = REsource LOcation And Discovery
- (Defines)* a signaling protocol for P2P networks
- Features:
  - Data storage and retrieval in a P2P network
  - Message and routing services
  - Designed for a variety of applications
  - Pluggable overlay algorithm
  - High security definitions
  - NAT and Firewall traversal

*Not a standard jet

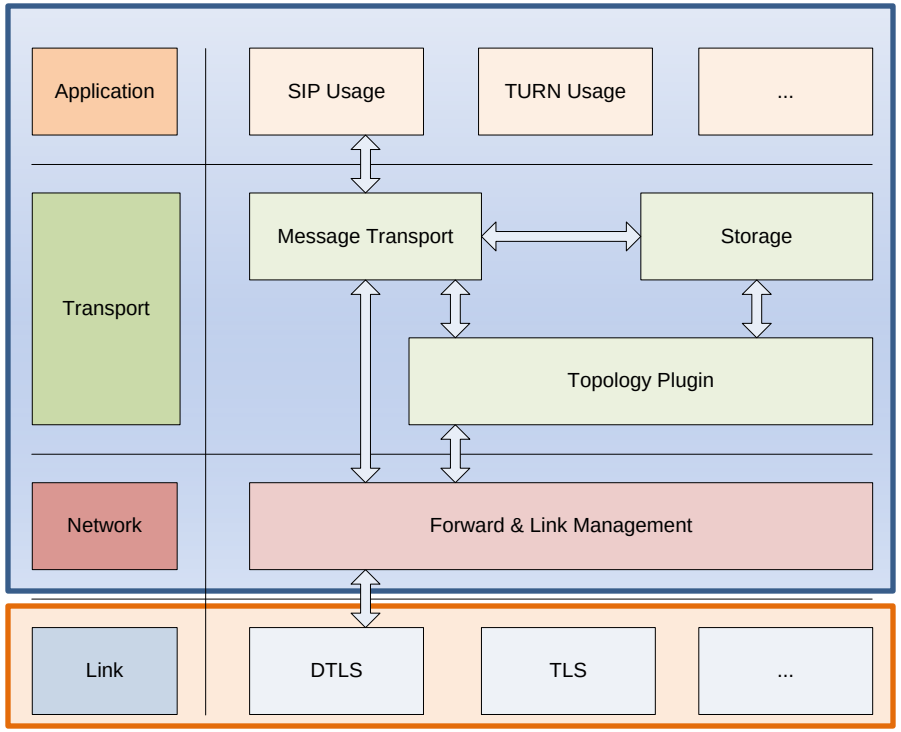
RELOAD Overview(2)

- RELOAD overlay instance: Functionality overview



RELOAD Architecture

OSI application layer!

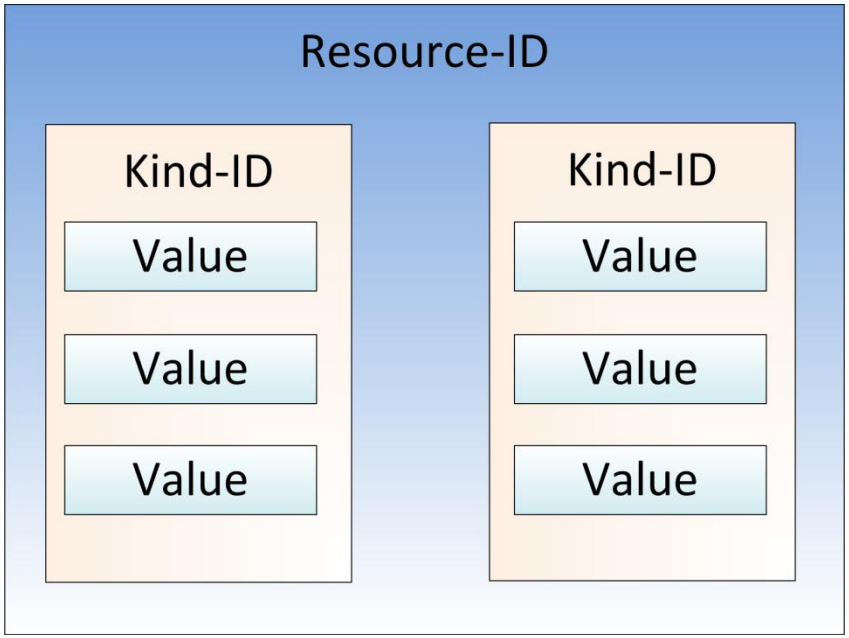


OSI transport layer!

- Application:
  - The application specific behaviors, called Usages
- Transport:
  - Defines messages(store, fetch, join, etc)
  - Defines storage types and permissions
  - Pluggable Overlay
- Network:
  - Creates, maintains and deletes connections
- Link:
  - Underlying secure transport protocols

Kind Definition

- Each stored data (called Resource) belongs to a Kind:
  - Resource-ID= Hash(Resource name) , e.g. SIP URI
  - But: Other Resources with different Kinds may produce same Resource-ID!
  - Solution: Each Kind has its own Kind-ID
- Its up by the application to add new Kinds (data structures)



Usages

- *"A usage is an application that wishes to use the overlay for some purpose"* [1]
- A Usage must specify:
  - Kind-IDs registration
  - Kind data structure definition
  - Access Control Rule
  - How a Resource name is formed
  - Merge of data, if its partitioned on multiple nodes
  - Defines types of connections

TURN Usage(1)

- Problem: Many (if not most) peers behind NAT
- Solution: Communication through intermediate relay peer
  - TURN = Traversal Using Relays around NAT)
- RELOAD Usage: Each overlay peer may act as TURN server
  - After overlay join, a peer knows if it has an public IP

```
if(IP==public):
    for d in range('1', turnDensity)
        store(TurnServerKind(hash(Node-ID+d)))
```

  - Advertises this peer as TURN server "randomly" in the Overlay

TURN Usage(2)

- Finding a TURN server (TS):
  - Peer uses a Find request messages

    find(TURNServerKind (random (Resource-ID)))
  - Returns 0 if no TURN server is known
  - Returns the Resource-IDs for this Kind
  - Then a peer uses a Fetch request for the returned Resource-ID
- Because every peer with public IP will become TS, a random search will quickly succeed

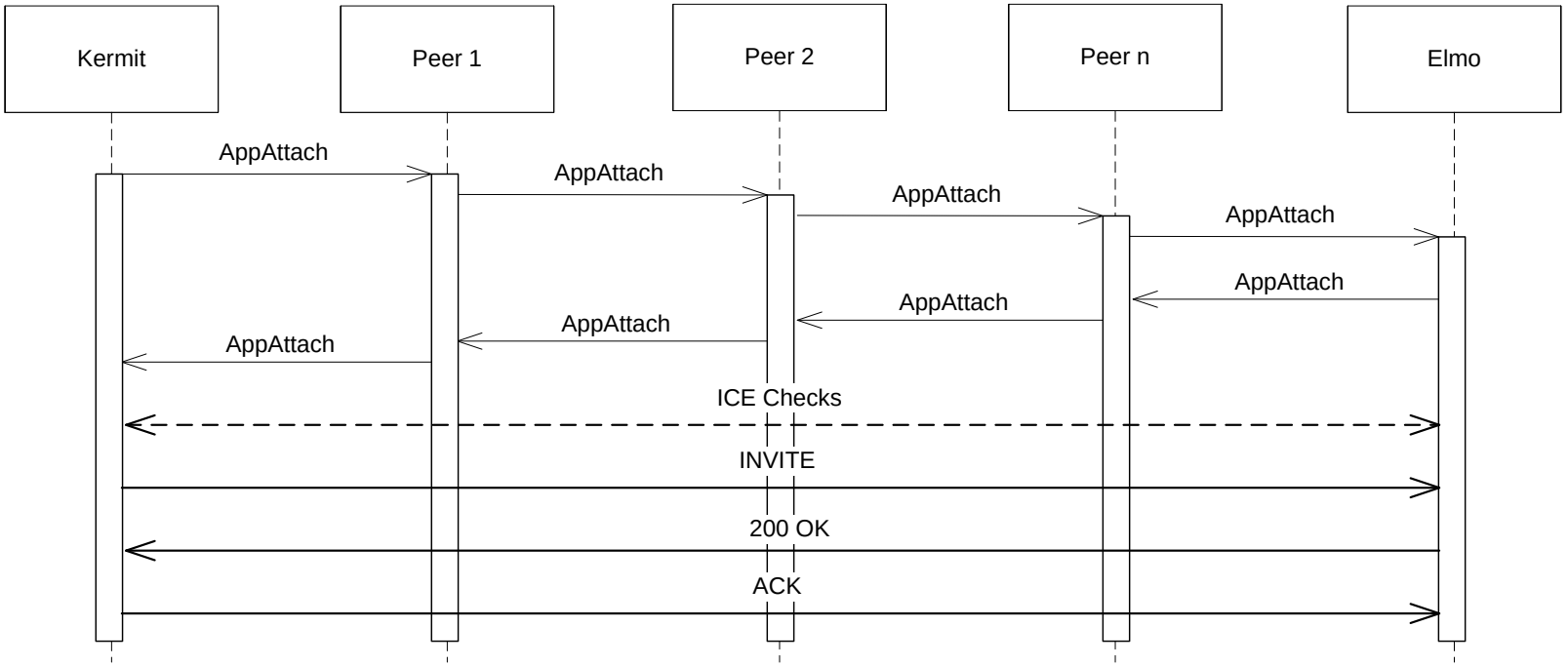
SIP Usage(1)

- Goal: Replace SIP registrars and proxies
- SIP Usage: Store mappings SIP URI → Node-ID
  - Example: *sip:elmo@muppets.com* → *e4f3a*
- User lookup:

  fetch(SIPKind(hash(‘*sip:elmo@muppets.com*’)))
  - Returns: e4f3a
- Establishing a connection:
  1. Use RELOAD AppAttach request to set up a transport
  2. Establish SIP connection using this transport connection

SIP Usage(2)

- Establishing a SIP Dialog:
  - ICE Checks for NAT and Firewall traversal

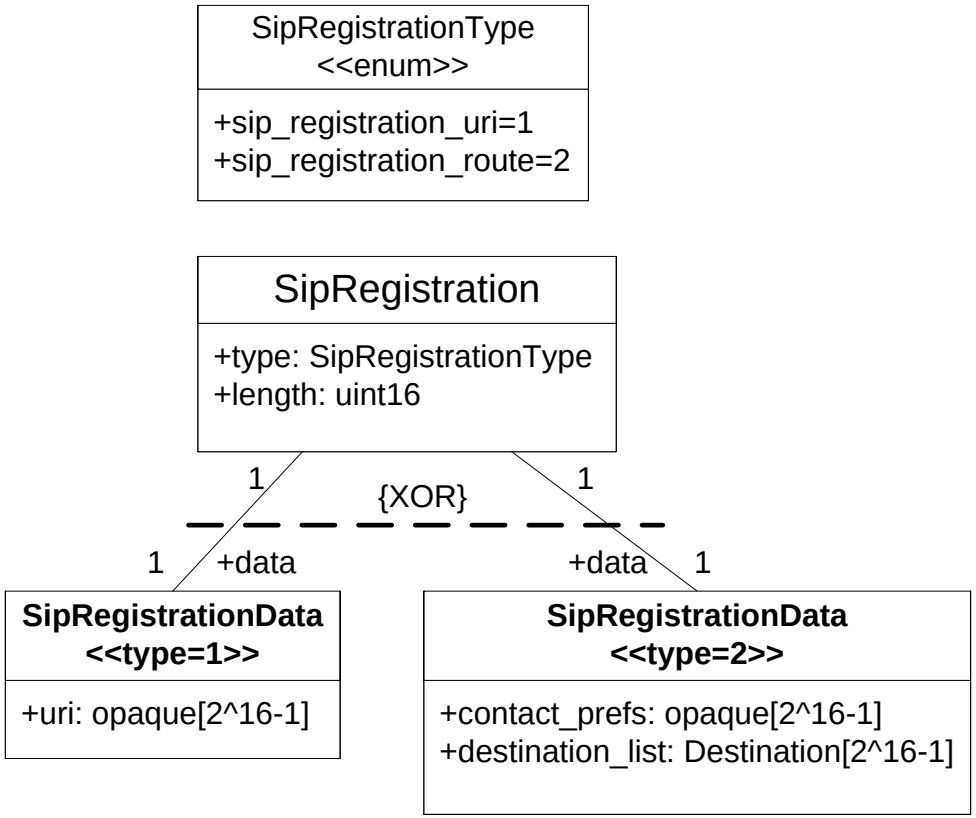


SIP Usage(3)

SIP Usage defines a new Kind data structure

- SipRegistration:
  - SIP URI as string

    XOR
  - Destination List with preferred contacts
- Note: a user can have multiple locations (Home, Office, etc..)



Service Discovery Usage(1)

- Problem: How to find a service in a RELOAD overlay?
  - Naive solution: Store all service-provider under well known Resource ID
    - ⟶Scales linearly 😐
    - ⟶Owner of Resource ID may store an amount of data ☹
    - ⟶Requests could overload ID owner ☹ ☹
- Solution: Building service discover tree structures:
    - ⟶Scales logarithmic
    - ⟶Storage and requesting load distributed

Service Discovery Usage(2)

- Each service provided in the overlay is identified by its own namespace
- Service registration: hash(namespace, i, j), with i is the level (depth) of the tree and j Node-ID at this level
  - Example: *hash('voice-mail', '1', 'ef3a1')*
- Then, service provider stores mappings up and down the tree
  - Stops if: *others IDs <= n.ID <= other IDs*

    OR
  - Stops if: its the only leaf in the tree
- Each leaf has its own peer responsible for a range of Node-IDs

Service Discovery Usage(3)

- Service lookup:
  - Fetch operation on hash(‘voice-mail’, ‘1’, ‘3af2b’)
  - Analog: Queries the tree up and down
  - Repeat Fetch() until a node is found with the closest Node-ID to the node searching for the service
- Then: Establish transport connection to this node and proceed application procedures

Conclusion & Outlook

- Conclusion:
  - RELOAD base protocol for P2P signaling
  - Ready to add new applications
  - Applications MUST define Usages, Kinds, etc..
  - Some Usages already defined
  - No Usage for conferencing
- Outlook:
  - RELOAD potential base for distributed conferencing
  - My draft writing had already begun ☺

Questions and Discussion

Thanks for your Attention!

References

[1] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, “REsource Location And Discovery (RELOAD) Base Protocol“, IETF, Internet-Draft work in progress 08, March 2010.

[2] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, “A SIP Usage for RELOAD“, IETF, Internet-Draft work in progress 04, March 2010.

[3] J.Meanpe, G. Camarillo, “Service Discovery Usage for REsource LOcation And Discovery (RELOAD)" IETF, Internet-Draft work in progress 00, January 2010.