Trustful hyper-linked entities in dynamic networks

reTHINK / 5G
Service Delivery via dynamic web-based hyper-linked entities

Anastasius Gavras, Eurescom GmbH
Paulo Chainho, Altice Labs (PT Inovação)
Brussels, April 6th 2016
Main goals of reTHINK

• Design and prototype a new Web-centric peer to peer Service Architecture
• Enable dynamic trusted relationships among distributed applications called Hyperlinked Entities (“Hyperties”)
• Support use-cases beyond commodity voice such as contextual and social communications, D2D, M2M/IoT applied to smart cities use cases
• Main drivers of reTHINK:
  – Agile Service Architecture, with a light weight standardization needs (and very few protocols)
  – Innovative Identity model, flexible and reliable
  – Bringing QoS features on demand on the new networks
Principles

• The reTHINK Service Delivery framework follows
  – Microservices Architecture patterns
  – Edge Computing paradigms

• reTHINK envisages
  – A global network of interconnected microservices called Hyperties
  – Executed in end-user devices or edge-network servers, on behalf of users
Main reThink concepts

- **Hyperty**: dynamic web-based service downloaded on client side or edge server, and linked to an identity
- **Protocol-on-the-fly** (ProtoFly) concept to avoid using standard network protocols (No need for normalization and standards)
- Independent **identity management**
Hyperty

- Implements a Service Logic
- An instance is associated to a “User” through an identity
- The Identity is decoupled from the Service Provider
- Hyperties are implemented in Javascript ECMA5/6 (currently)
Protocol on-the-fly – Protofly

- Protocol on-the-fly leverages the code on-demand support by Web runtimes (e.g. Javascript)
- The most appropriate protocol stack is dynamically selected loaded and instantiate during run-time
Protofly Procedure
Protofly Procedure

Bob Service Provider

Catalog

H2
P2

Alice Service Provider

Catalog

H2
P2

Alice Device

Hyperty Sandbox

H1

Protostub Sandbox

Bob Service Provider

Msg Node

Alice Service Provider

Msg Node

Protostub Sandbox

Runtime UA

Msg BUS

WebRTC Engine
Protofly Procedure

Bob Service Provider

Alice Service Provider

WebRTC Engine

Protostub Sandbox

Msg Node

Hyperty Sandbox

P2

P1

Msg BUS

Media Stream
• Identity tokens are generated, inserted in Messages sent by Hyperties, and validated by dedicated modules within recipient Hyperty Runtimes before delivered to the target Identity.
• Identity management procedures are performed according to applicable policies managed by the end-user.
Trustful hyper-linked entities in dynamic networks

Thank you

https://rethink-project.eu/publications/deliverables/

https://github.com/reTHINK-project
Trustful hyper-linked entities in dynamic networks

Backup slides
How it works (1/2)

Alice

Device A1

Register/log-in

Alice’s CSP

Hyperty instances Registry

Alice’s Identity Provider

Hyperty A

Alice’s Identity Assertion

Bob

Device B1

Register/log-in

Bob’s CSP

Hyperty B

Bob’s Identity Provider

Hyperty instances Registry

Bob’s Identity Provider

Associates Alice’s Identity to Hyperty

Associates Bob’s Identity to Hyperty
How it works (2/2)

1. **Search Bob**
   - « Bob » « T-Labs » « Berlin »
   - Gets the Hyperty address to join Bob (+ Bob’s Identifier @ Domain)

2. **Call Bob**
   - + Alice’s Identity Assertion
   - + Bob’s Identifier
   - Updates device runtime with Bob’s ProtoFly

3. **ProtoFly to Join Bob**

4. **Discovery service**
   - Checks Alice’s Assertion
   - Alice’s Identity Provider

5. **Bob’s CSP**

6. **Bob’s Hyperties instances Registry**

7. **Catalog**

---

**Communication**

- **Device A1**
- **Hyperty A**
- **Hyperty B**
- **Device B1**

**reTHINK Project M9 Review – WP2**
Functional Architecture

- Hyperties are provisioned to Catalogues from where they can be discovered and downloaded to devices or network edge server. The instantiated versions of hyperties are registered in a Registry, which represents reachable authenticated peers. The Registry serves as a location manager and is used for peer discovery.

- A peer can be a human being, a group or, a connected object (e.g. building room with sensors). Peers have independent identities that are maintained by Identity Management functionalities.
Hyperty Core Runtime

- Service Provider
- Service Provider Catalogue Interface
- Runtime Device
- Service Provider Sandbox
- Core Sandbox
- Runtime User Agent
- Catalogue ProstStub
- Policy Engine (PDP/IFEP)
- (Incl. Authorisation + Policies Repository)
- Message
- BUS
- Authorise
- Policies
- Registry
- GoS User Agent
- Graph Connector
- Identity Module
Runtime Architecture Security Analysis

Platform Stack Layers
- Sandbox
  - Runtime
    - Process
      - Operating System
        - Hardware

L1 attacks
L2 attacks
L3 attacks
L4 attack
L5 attack

Approved

Browser Platform
- L1
- L2
- L3
- L4
- L5

D2
- A1
- A2
- A3

D3
- A4
- A5
- A6

Regular user
Advanced user
Power user

reTHINK Project M9 Review
Message Delivery Layers

Domain level
Message Delivery

Runtime level
Message Delivery

Sandbox level
Message Delivery

Any Messaging Protocol (Protofly)

Native Standard Hyperty JSON Messaging

Hyperty Runtime

P2P
Message Delivery

- The Message delivery is based on a simple message Router functionality.
- Routers deliver messages to all found listeners, which can be other Routers or end-points (Hyperties)
- Listeners are programmatically registered and unregistered by Routing Management functionalities
A Resource Oriented Messaging Framework is used for communication. Publish/subscribe and request/response messaging patterns are supported.
Hyperties cooperate and communicate each other via P2P Synchronisation of JSON Data Objects.

Hyperty Reporter only have to write on a JSON data object everytime there is a need to communicate with other Hyperties.
Hyperty Data Object Schema

Each Hyperty Data Object is formally described by a JSON-Schema that is identified by a Catalogue URL. This allows to check whether two different Hyperties are compliant by cross checking each supported Hyperty Data Object schema.

Eg Hyperties supporting Connection schema are able to handle WebRTC Peer Connections independently of the signalling protocol used. The URL Scheme for Connection Data Objects is "connection" (example: "connection://example.com/alice/bob201601290617")
Hyperties supporting Context Data Schema are able to produce or consume Context Data, usually collected from sensors. The URL Scheme for Communication Data Objects is "ctxt". Example: `ctxt://example.com/room/temperature201601290617`).
Mobility

Abstraction from Network or Hyperty Adresses Facilitates object and hyperty mobility