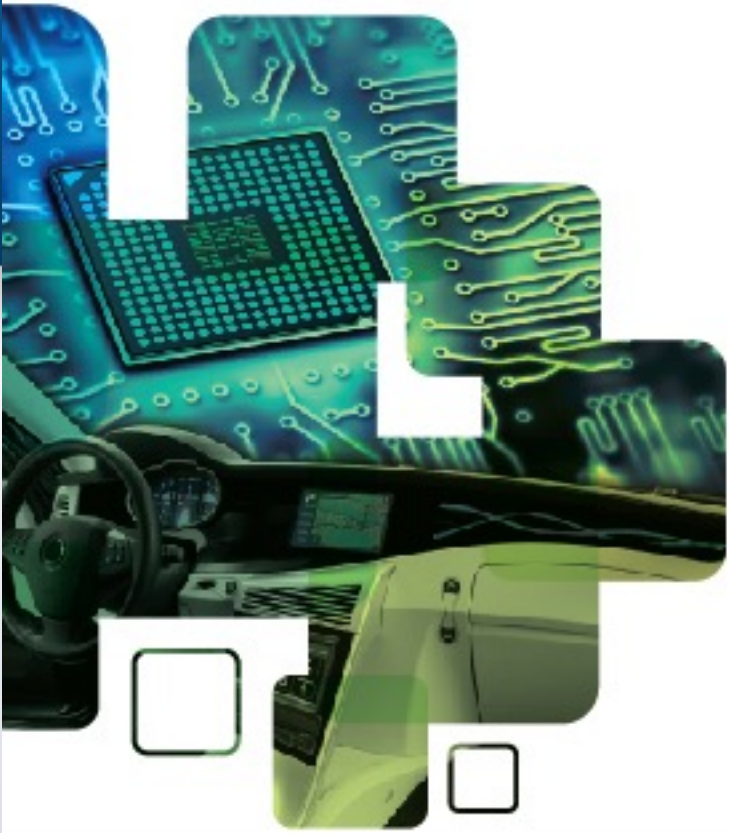


Emulating the Essence of Erlang in RVI

2016-09-08

Ulf Wiger
GENIVI

Dashboard image reproduced with the permission of Visteon and 3M Corporation

Outline

- Briefly, What is GENIVI?
- Briefly, What is RVI?
- Problem Description
- Essence of Erlang?
- Solution

BRIEFLY, WHAT IS GENIVI?

GENIVI Alliance

Developing an open standard for aligning automotive and consumer infotainment cycles

GENIVI Alliance

GENIVI

GENIVI Members

Original car manufacturers

BMW Group

Great Wall

HONDA The Power of Dreams

HYUNDAI MOTOR GROUP

JAGUAR LAND ROVER

JOHN DEERE

DAIMLER

NISSAN

PSA PEUGEOT CITROËN

RENAULT

SAIC 上汽集团 SAIC MOTOR

VOLVO Volvo Cars

First tiers

AW AISIN AW CO.,LTD.

ALPINE

ALPS

AppCarousel

clarion

Continental

DELPHI

DENSO

HARMAN

DESAY SV AUTOMOTIVE

MOBIS

LG

MAGNETI MARELLI

MITSUBISHI ELECTRIC

peiker

Pioneer

BOSCH Invented for life

Visteon

OSV, Middleware, HW, Svc Suppliers

accenture High performance. Delivered.

ACCESS

Advanced Driver Information Technology

Airbiquity

AllGo embedded

ALTERA

Aricent

ARKAMYS

arynga

Advanced Telematic Systems

audiokinetic

高德 AutoNavi

BearingPoint.

BLACKDUCK

cinemo

Codethink

COGENTEMBEDDED

COLLABORA

CTAG Centro Tecnológico de Automoción de Galicia

CYBERCOM GROUP

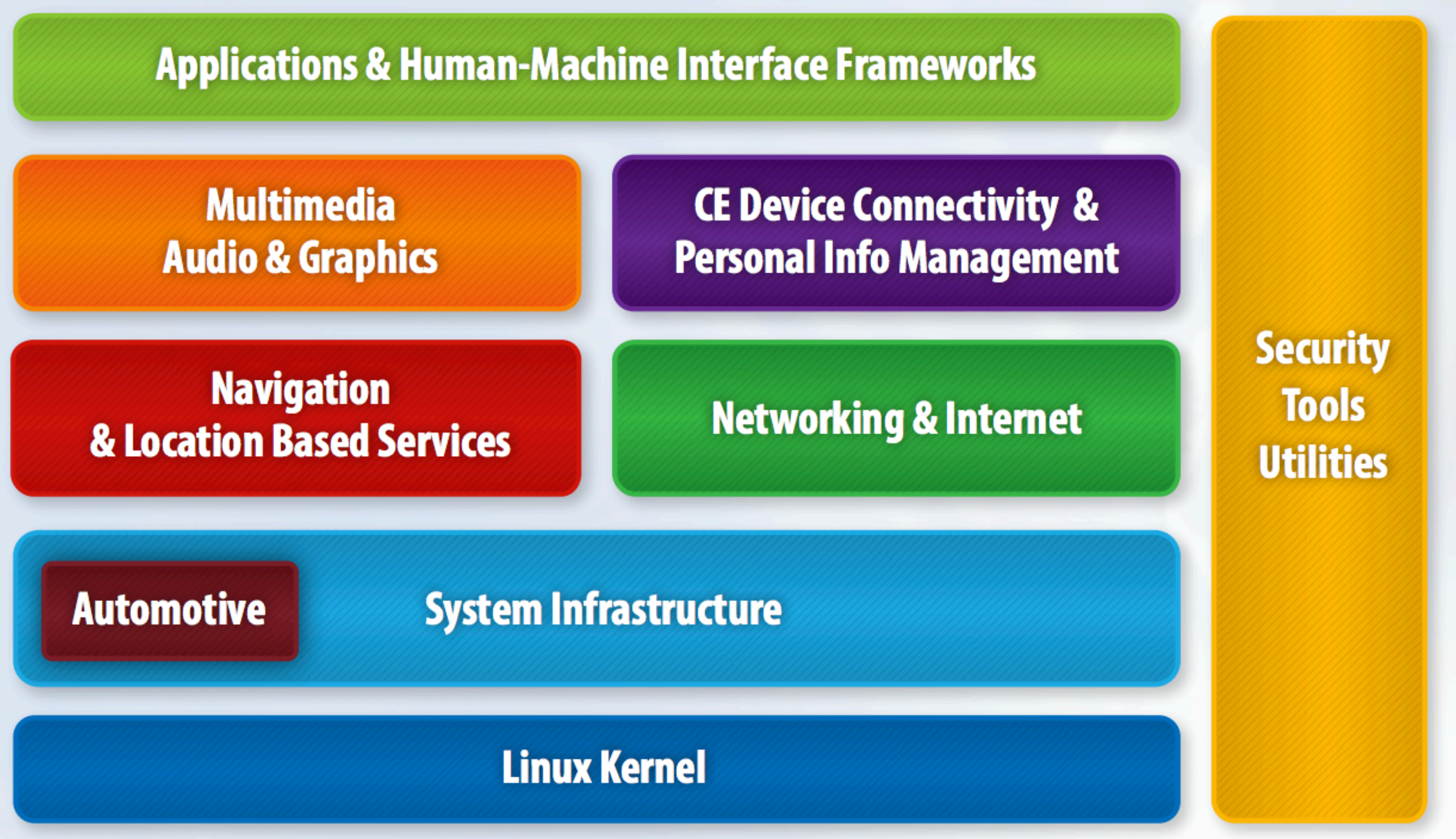
EnGIS EnGIS Technologies, Inc.

ERICPOL

ERICSSON

8 September 2016

GENIVI Projects



Applications & Human-Machine Interface Frameworks

Multimedia Audio & Graphics

CE Device Connectivity & Personal Info Management

Navigation & Location Based Services

Networking & Internet

Automotive

System Infrastructure

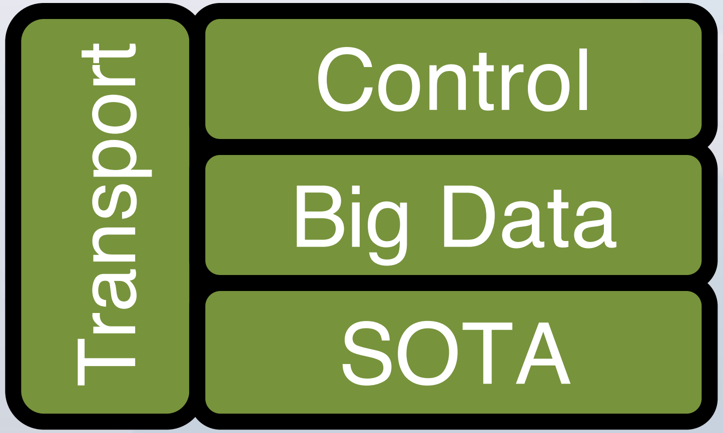
Linux Kernel

Security Tools Utilities

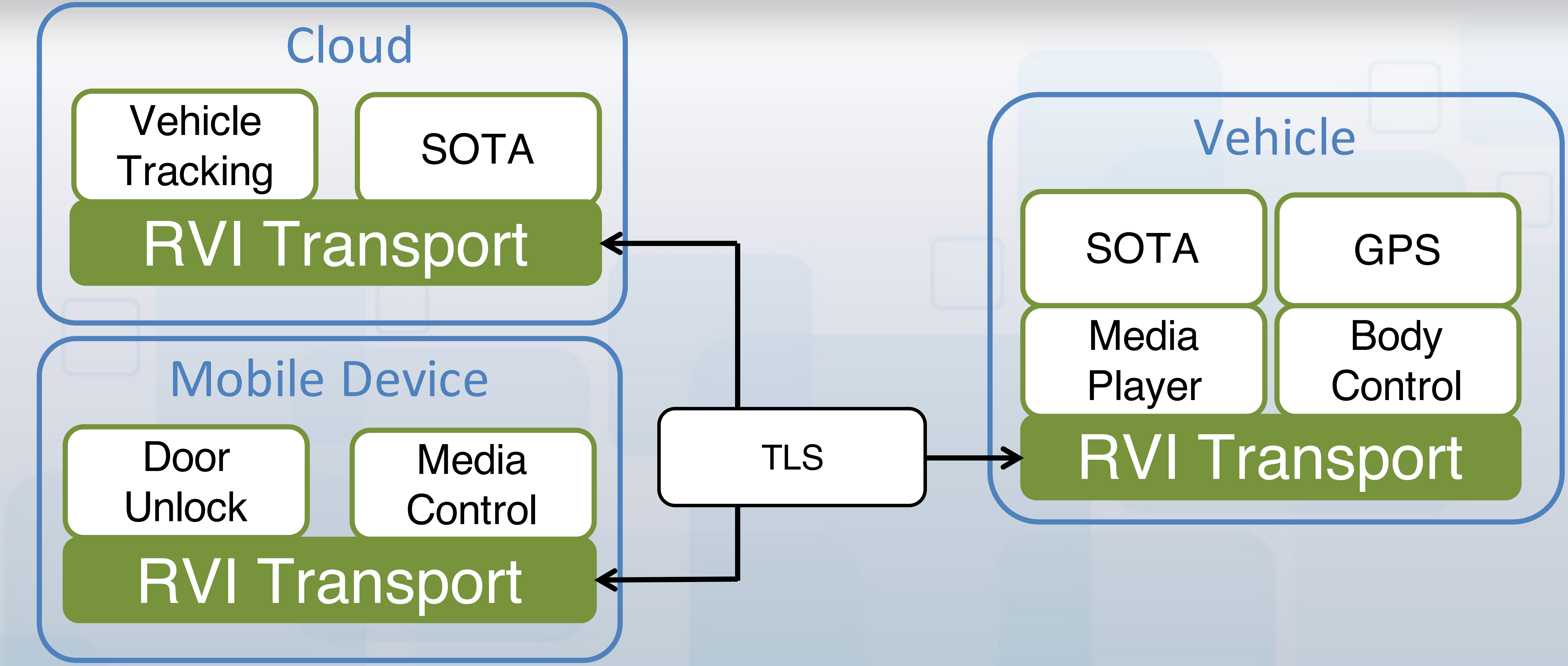
BRIEFLY, WHAT IS RVI?

RVI – Remote Vehicle Interaction

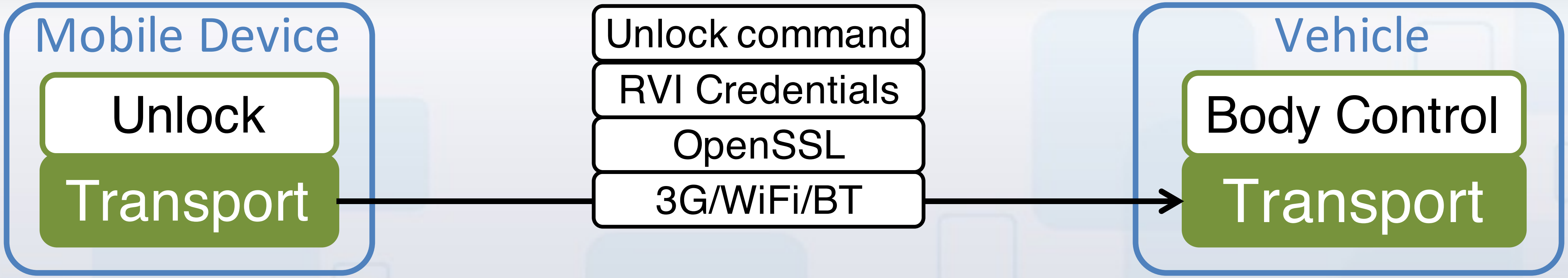
Provide P2P based provisioning, authentication, authorization, discovery and invocation between services running inside and outside a vehicle.



Schematics


Cloud
Vehicle Tracking
SOTA
RVI Transport
Mobile Device
Door Unlock
Media Control
RVI Transport
TLS
Vehicle
SOTA
GPS
Media Player
Body Control
RVI Transport

Security

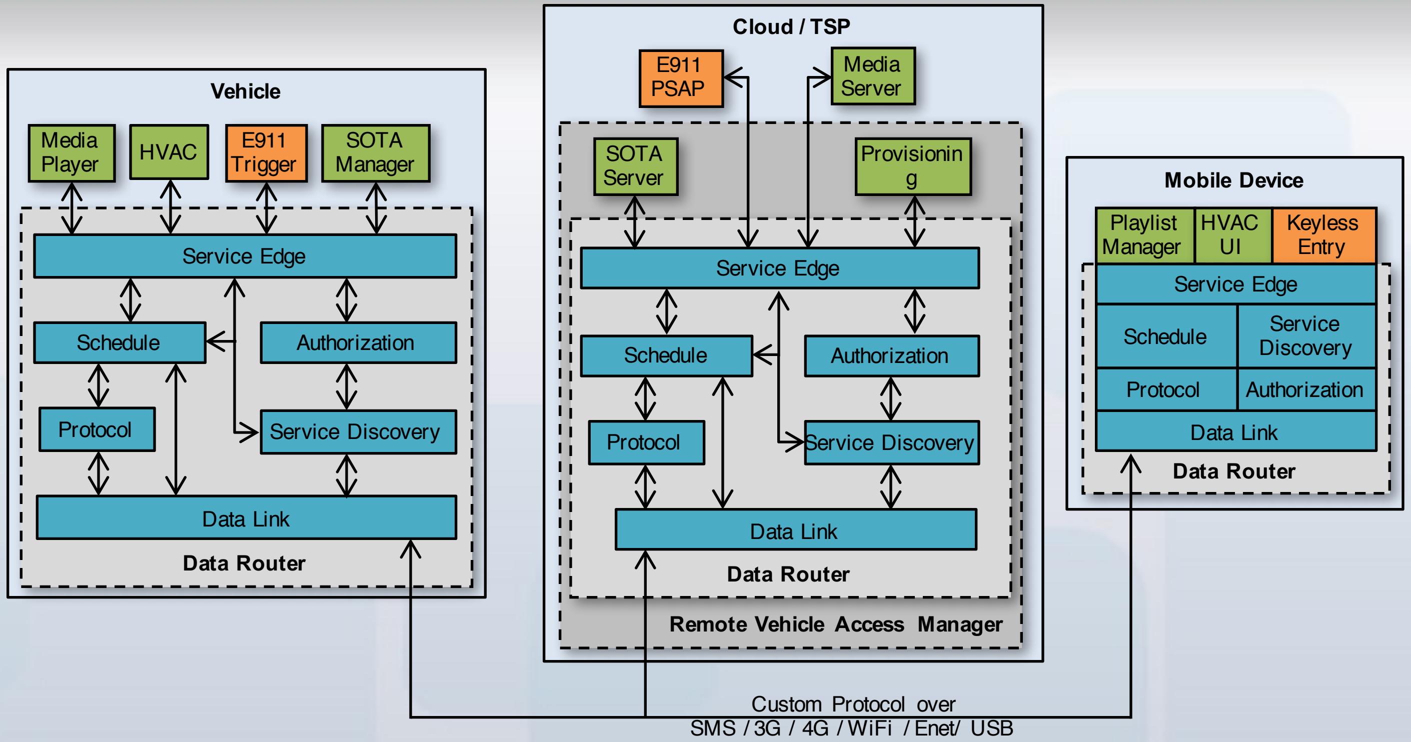


- **OpenSSL**
  TLS provides authentication, core eavesdropping and MITM attack protection

- **RVI Credentials**
  Signed by root server. Verifiable access control lists.

- **Unlock**
  Will only be accepted if access control succeeds

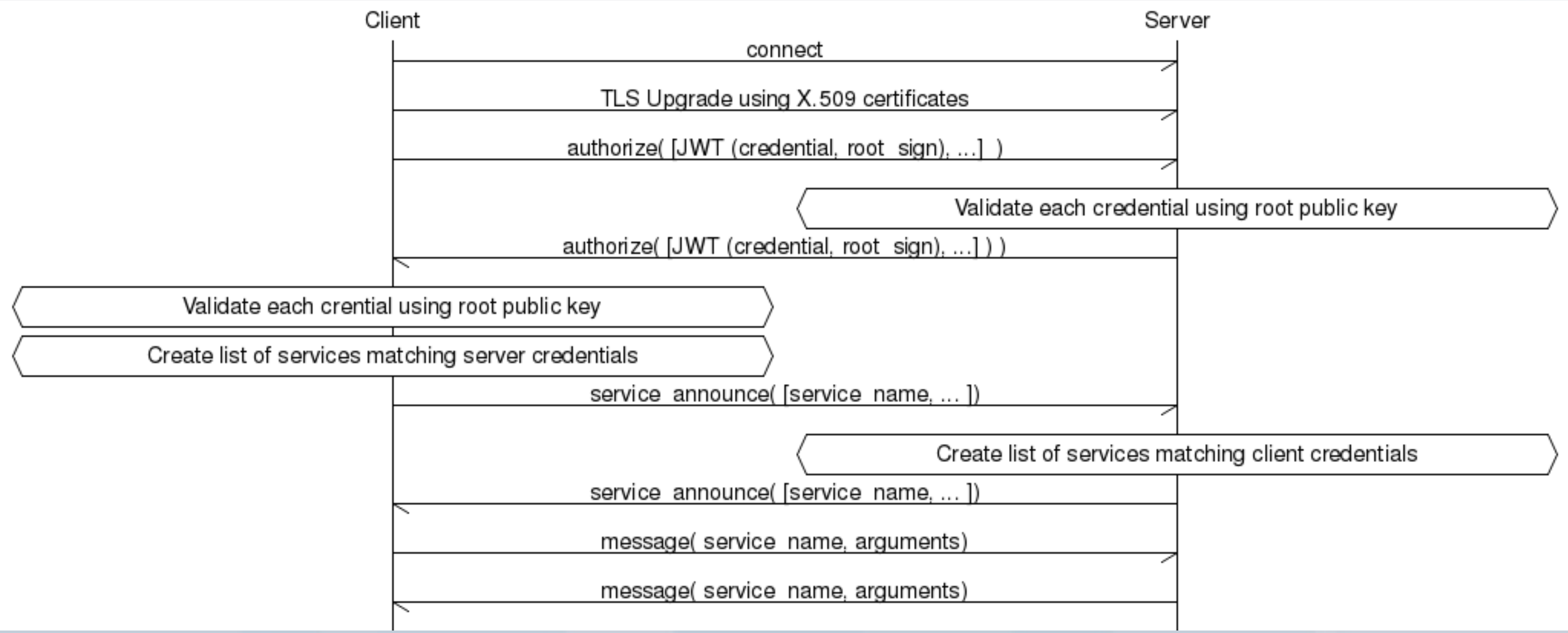
RVI Transport – A Closer Look



🟦 GENIVI-developed component (OSS) 🟧 OEM/3rd party proprietary component 🟩 3rd party reference component (OSS)

https://github.com/GENIVI/rvi_core

RVI Protocol



Client

Server

connect

TLS Upgrade using X.509 certificates

authorize([JWT (credential, root sign), ...])

Validate each credential using root public key

authorize([JWT (credential, root sign), ...]))

Validate each crential using root public key

Create list of services matching server credentials

service announce([service name, ...])

Create list of services matching client credentials

service announce([service name, ...])

message(service name, arguments)

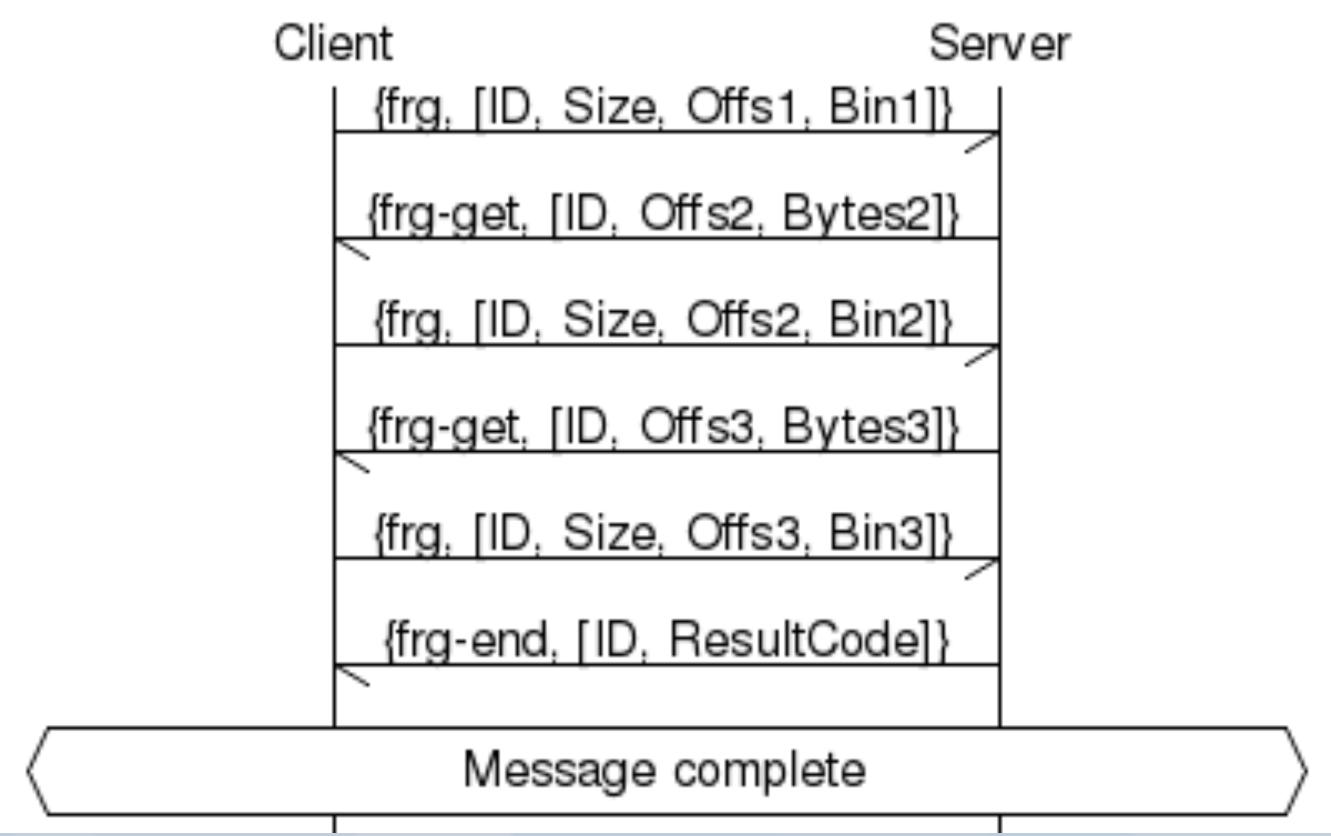
message(service name, arguments)

RVI Credentials

- Encoded as JSON Web Tokens (JWT) by provisioning service
- Validated using root cert public key
- Inform service announcements
- Service invocations authorized through pattern-matching (wildcards optional)

```
{
    "create_timestamp": 1439925416,
    "right_to_invoke": [
        "genivi.org/vin/"
    ],
    "right_to_receive": [
        "genivi.org/backend/sota"
    ],
    "id": "insecure_cert",
    "iss": "genivi.org",
    "device_cert": "",
    "validity": {
        "start": 1420099200,
        "stop": 1925020799
    }
}
```

Message Chunking


Client
Server
{frg, [ID, Size, Offs1, Bin1]}
{frg-get, [ID, Offs2, Bytes2]}
{frg, [ID, Size, Offs2, Bin2]}
{frg-get, [ID, Offs3, Bytes3]}
{frg, [ID, Size, Offs3, Bin3]}
{frg-end, [ID, ResultCode]}
Message complete

Service Edge API

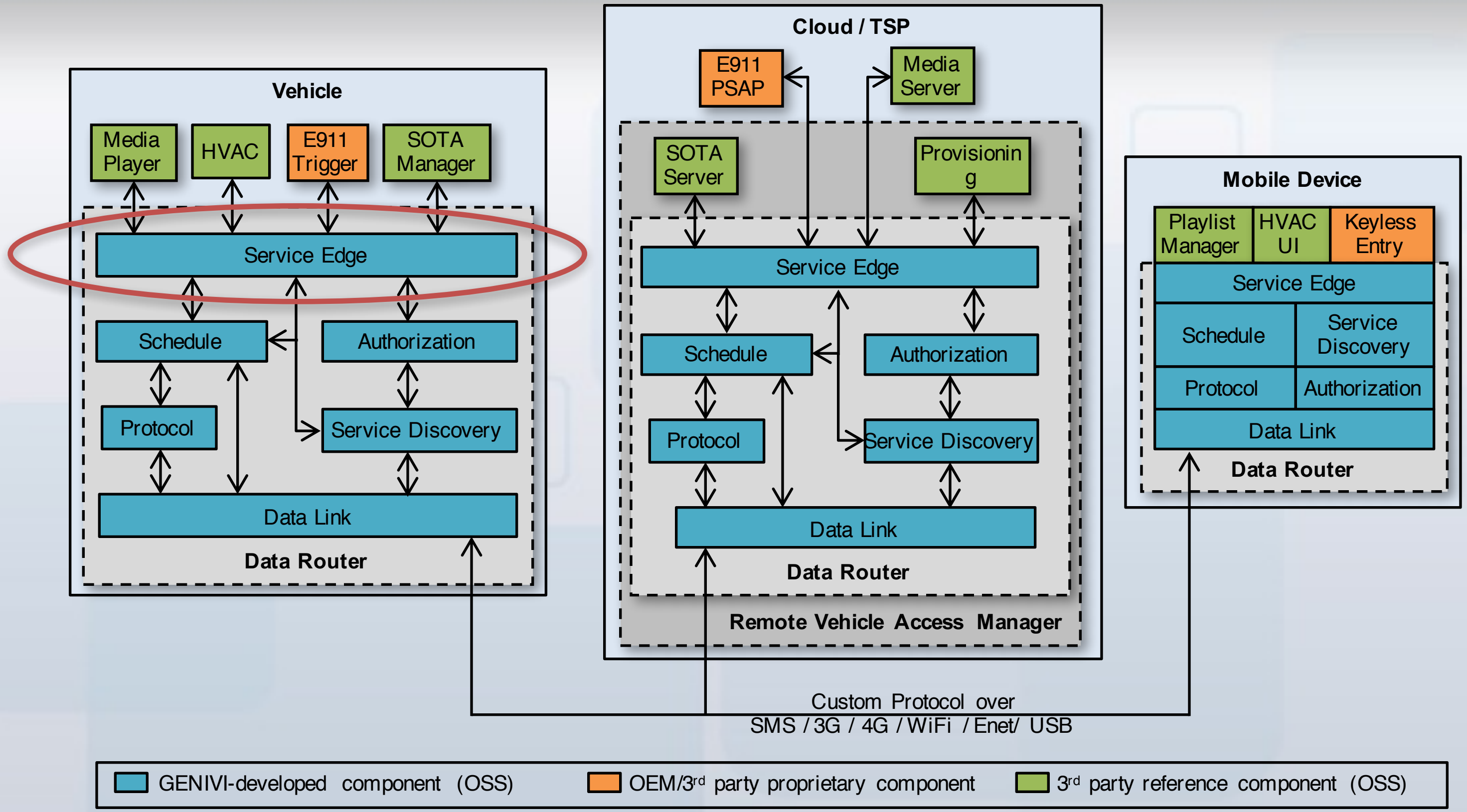
RPCs:

```
register_service(Svc, Addr)
unregister_service(Svc)
get_available_services()
message(Svc, Timeout, Params)
```

Notifications:

```
service_available(Svc)
service_unavailable(Svc)
handle_remote_message(
  IP, Port, Svc, Timeout, Params)
handle_local_timeout(Svc, TID)
```

- JSON-RPC
- Websocket
- (msgpack-RPC soon)



PROBLEM DESCRIPTION

Problem Description

- Synchronous service invocation
  - RPC semantics simplify client code
  - ...but RVI is asynchronous & store/forward!
  - Currently, client often passes reply URL, or registers reply service
- Instant failure reporting
  - RVI timeouts can be long (even days)
  - Complex message processing chain

Problem description, cont...

- Portability
  - Erlang, Android, iOS, C implementations
- Security
  - Reply URL is a security risk
  - Advertising a reply service is ugly & complex

ESSENCE OF ERLANG

8 September 2016

Erlang Client-Server

- Synchronous wrapper
- One-way monitor
- Unique reply tag

```
call(S, Req, Timeout) ->
    Ref = erlang:monitor(process, S),
    S ! {'$call', {self(), Ref}, Req},
    receive
        {Ref, Reply} ->
            Reply;
        {'DOWN', Ref, _, _, Reason} ->
            error(Reason)
    after Timeout ->
        error(timeout)
    end.
```

Gen_server delayed response

```
handle_call({...} = Req, {Pid,_} = From, #st{pending = Pend} = S) ->
    Mref = erlang:monitor(process, Pid),
    dispatch_req(Req, From, ...),
    {noreply, S#st{pend = [{From, Mref} | Pend]}};
...

handle_info({delayed_reply, From, Reply}, #st{pending = Pend} = S) ->
    case lists:keytake(From, 1, Pend) of
        {value, {_, Mref}, Rest} ->
            erlang:demonitor(Mref),
            gen_server:reply(From, Reply),
            {noreply, S#st{pending = Rest}};
        false ->
            {noreply, S}
    end.
```

Analysis

- Things to Emulate
  - Synchronous call wrapper
  - Delayed response
  - Monitor
- Different from Erlang
  - Security
  - Store & forward
  - Multi-node relay

SOLUTION

RVI Service names

- MQTT syntax
- Each RVI node has a unique UUID

```
genivi.org/vehicle/[UUID]/HVAC/set_temp
```

- Names starting with '$' are internal (as in MQTT)
  - Cannot be accessed through the service edge

```
$RVI/node/[UUID]/reply/[Ref]
```

Sequence

- Client calls 'message' RPC in Service Edge
  - If {"reply": false} (default), returns immediately
  - If {"reply": true}, RPC waits for service reply
    - Internal reply service created & added to msg

Sequence, cont ...

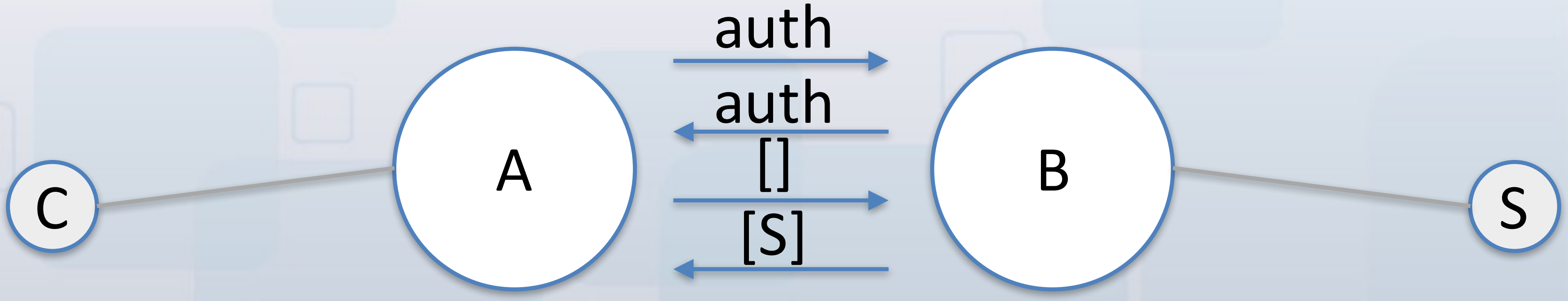
- If service is local, message is dispatched immediately
  - Otherwise, scheduled for remote dispatch
- If {"reply": true}, service reply routed back to reply service
  - Otherwise, reply ignored

RVI Event Service

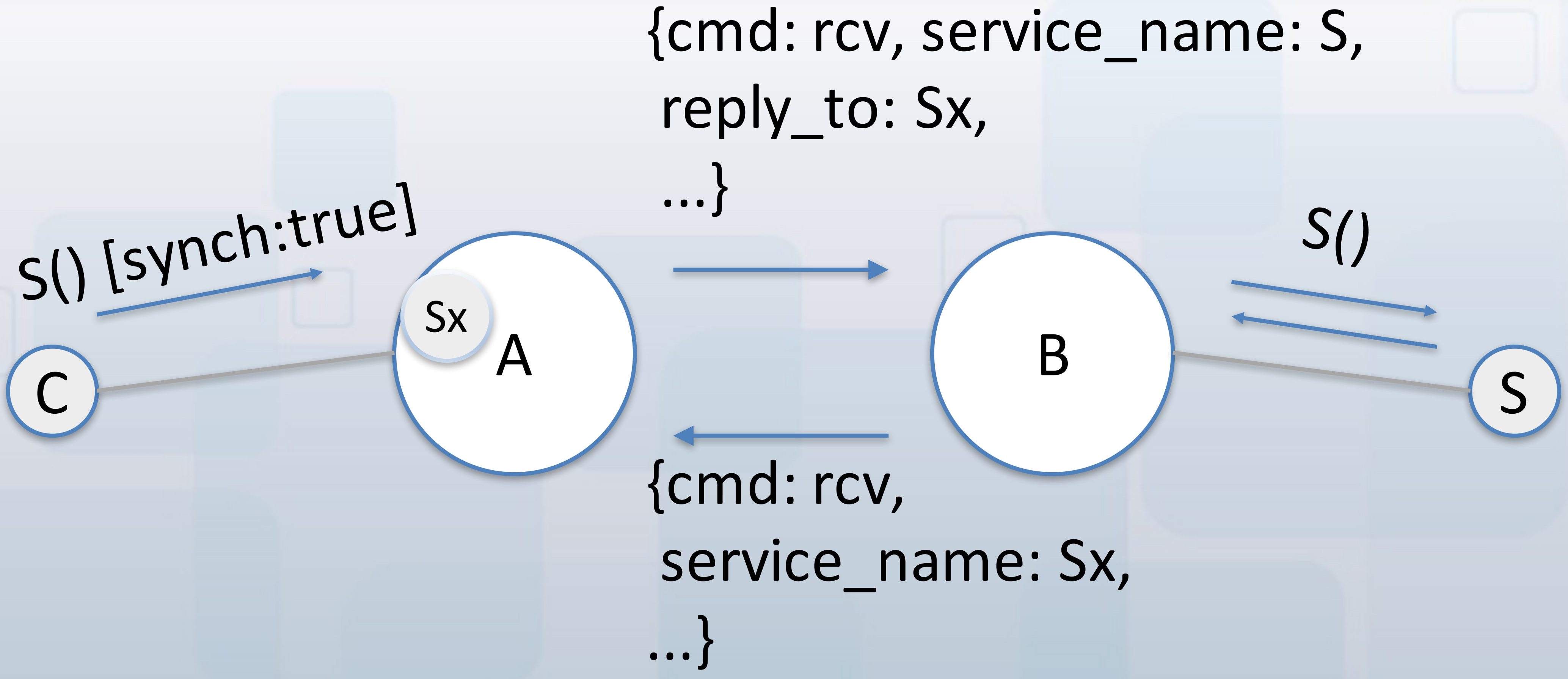
- High-level events with severity indicators
  - 0: info, 1: success, 2: warning, 3: error
- Log ID included in message, threaded across nodes
- If reply service AND error: issue an error response (normal service dispatch, but to internal service point)

```
19:13:53.178 svc_edge:7-zPWi  0  svc_edge   local_message: genivi.org/vehicle/.../HVAC/set_temp
19:13:53.180 svc_edge:7-zPWi  1  authorize   local msg allowed: Cred=36cecde8-...
19:13:53.180 svc_edge:7-zPWi  0  svc_edge   schedule message (.../HVAC/set_temp)
```

Example



Example



S() [synch:true]

{cmd: rcv, service_name: S,
reply_to: Sx,
...}

S()

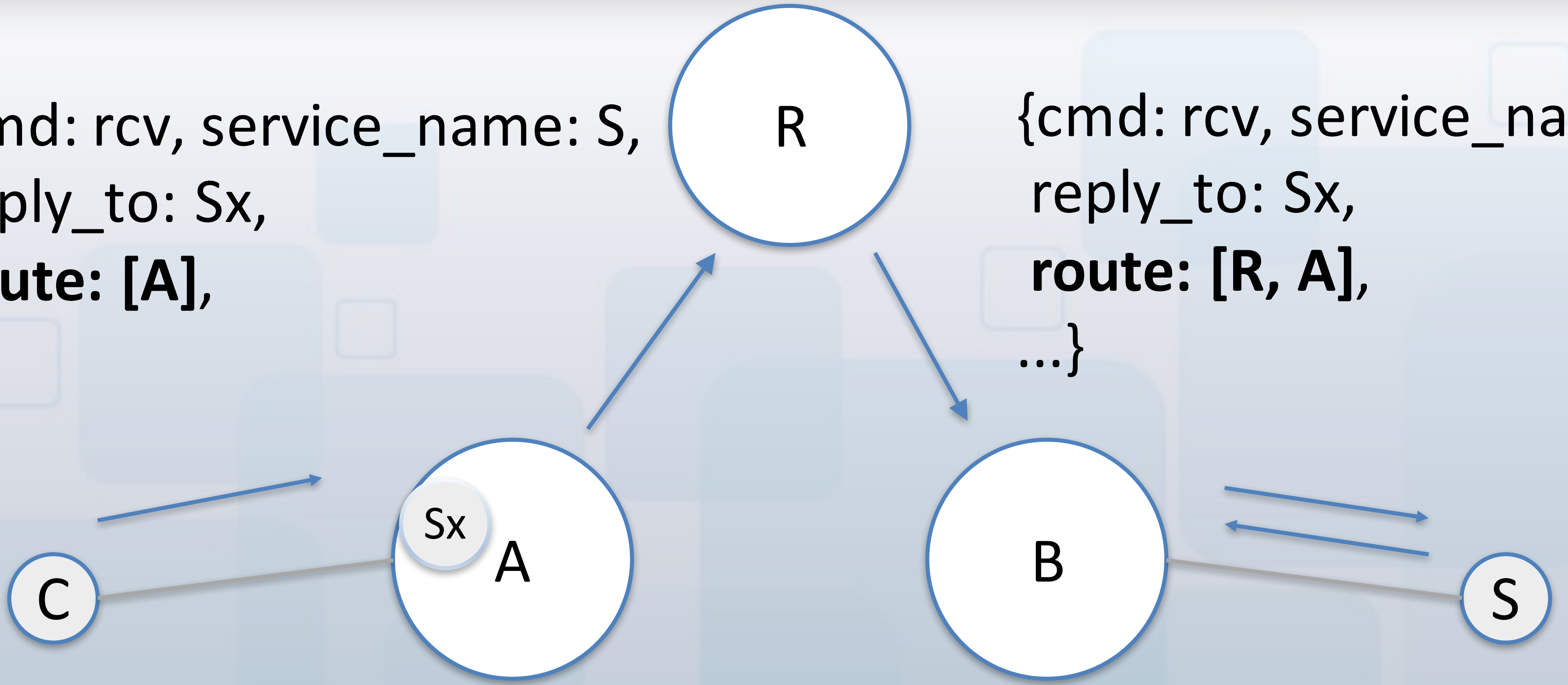
{cmd: rcv,
service_name: Sx,
...}

Adding Multi-Node Relay

- Service announcements based on access ctl lists
  - But internal service points not announced
  - Reply routing based on unique node ID
  - Reply along the same path as request dispatch
- SIP uses a "record-route" header
  - Each relay node pushes itself onto the path stack
  - When replying, pop the stack, get the next hop

Adding Service Relay

{cmd: rcv, service_name: S,
reply_to: Sx,
route: [A],
...}



{cmd: rcv, service_name: S,
reply_to: Sx,
route: [R, A],
...}

RVI Status

- Version 0.5.0 avaiable (License: MPL 2.0)
- rvi_core (RVI Transport) usable for pilots
  - Debian and Raspian packages built
  - Automated test suite (Common Test)
- Android SDK
- iOS SDK
- Python support libs
- Dynamic agent demos

RVI Status, cont...

- Next version:
  - Lots of code cleanup
  - Robustness focus
  - Synch RPC wrappers
  - Multi-node message relay
  - Delegated provisioning authority
  - C client (rvi_lib)

Thank You!