

Connecting Objects in the Internet of Things

INGE GRØNBÆK



Inge Grønbaek is Senior Adviser in Telenor R&I

The architecture includes a secure Application Programming Interface (API), a backbone and separate device networks with standard interface to the backbone. The API decouples innovation of services and service logic from protocols and network elements. It also enables service portability between systems, i.e. a service may be allocated to end-systems or servers, with possible relocation and replication throughout its lifecycle. Machine-to-machine services for Connected Objects (CO) could benefit the society in many areas, including environmental, health care, trade, transportation, alarms and surveillance. However, such development depends on powerful communications features with global interoperability for service ubiquity. Interoperability is required not only for a standard Quality of Service (QoS) controlled Internet Protocol (IP) bearer, but also for cross domain security, mobility, multicast, location, routing and management, including fair compensation for utility provisioning. The proposed architecture with its API not only includes these critical elements but also caters for multi-homing, mobile networks with dynamic membership and third party persistent storage based on indirection. The API supports end-to-end service control and offers capability features as a vehicle for service development and ubiquitous deployment. The architecture is more generic than traditional hierarchical sensor and actuator networks as it supports grids and autonomous neural type of networks.

Introduction

This article describes the core of a service oriented architecture offering generic functionality for Connecting Objects in the Internet of Things (IoT). The architecture supports network and/or end system based services and service features.

The evolving IoT may potentially cause an explosion in the number of communicating Connected Objects (CO). Even when most of these objects contribute with limited traffic, the sum may well become the major source on the Internet. CO services could benefit the society in many areas, including environmental, health care, trade, transportation, alarms and surveillance. However, such development depends on the availability of global interoperability and powerful communications features.

There are urgent actions to be taken to pave the way for this development to take place, in order to benefit individuals, interested parties and the community as a whole. The pivotal point is establishment of a global market through service ubiquity. This requires new attitudes with respect to cooperation, coordination and standardisation in order to ensure inter-carrier interoperability. The architecture is therefore introduced in ETSI in order to contribute to this process [1].

The functionality offered to objects via an API is either peer-to-peer (P2P) or enabled by a minor set of new generic functional entities. These include a gateway and an anchor point entity class. The gateway

can be instantiated for interconnect of a rich variety of COs, including layer two proprietary COs. It can also be designed to support interoperability for native General Packet Radio Service (GPRS) devices on GPRS networks. The anchor point entity handles global mobility management and mobile M:N multicast. Additionally, these new entities support presence and location based services. Locations of COs can be identified, and the set of COs at a specified location may be found. Privacy is also handled by the same entities.

The architecture is functionally layered, with protocols and components identified for each layer of the well known Open Systems Interconnect (OSI) model. Diverse protocol stacks are supported by relating the stack profile and the CO identity. Deployed Internet protocols and protocols under development by the Internet Engineering Taskforce (IETF) are adopted.

The two critical elements for the support of ubiquitous services are interoperability for services and bearers. Bearer interoperability requires both interoperability for the IP bearer (user plane) and for the bearer control (control plane).

The background for the strong interoperability demand is described by Metcalfe's Law, stating that the usefulness of a network increases by the square of the number of nodes (users or devices) connected to the network. Furthermore, Reed's Law states that the utility of large networks, particularly social networks, can scale exponentially with the size of the network.

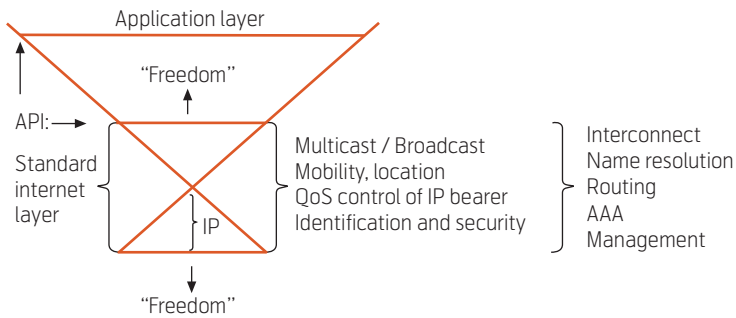


Figure 1 Internet layer with API

Formation of such groups is gradually happening, e.g. the social utility Facebook which connects people with friends and others who work, study and live around them. These strong positive network externalities, where the benefits are an increasing function of the number of other users, represent a huge commercial potential which may be severely reduced by the lack of interconnect [2] and missing ubiquity in service support and provisioning.

The major area in demand for harmonization belongs to what is here termed the Internet layer (Figure 1). The Internet layer is the IP network layer, extended to include the inter-domain functionality required for end-to-end maintenance of QoS, Security, Mobility, Location, Multicast, Name resolution, Routing, Authentication Authorisation and Accounting (AAA), and Management. This functionality is required to interoperate across interconnected domains, i.e. administrative and technology domains, for the IoT to support ubiquitous multifunctional services. In order for higher layer functionality and applications to get access to this functionality, there is a need for Application Programming Interfaces (API) at selected layers of the protocol stack (e.g. as depicted for the application and the Internet layer of Figure 1).

The architecture is more generic than traditional hierarchical sensor and actuator networks, as it supports grids and autonomous neural types of communications. The following description of the architecture is centred on the API and the functionality required for its support. This is starting with object identification in chapter 1. Layering with functional aggregation, and primitive (method) design principles in chapter 2. Chapter 3 describes an example API service element and provides an overview of the service functionality (Appendix A goes into more detail). Network elements and functional components required for support of the API are described in chapter 4.

1 Object Identification

Applications benefit from objects being identifiable and/or locatable through different mechanisms. The most important ones are:

- By identifiers
- By the location (e.g. geographical confinement)
- By an object's profile or element(s) thereof, e.g. in combination with location.

The major challenge is to define and standardise a globally unique namespace supporting ubiquitous services. A solution is proposed by [3] and [4], introducing a new flat namespace based on public and private key pairs. This namespace is called the Host Identity namespace. It fills the gap between the IP and Domain Name Service (DNS) namespaces. The Host Identity namespace consists of Host Identifiers (HIs), the public key of an asymmetric key-pair. Each host will have at least one Host Identity. Each Host Identity uniquely identifies a single host.

For efficiency purposes a Host Identity Tag (HIT) is defined as a 128-bit representation of a Host Identity. It is created by taking a cryptographic hash over the corresponding Host Identifier. A HIT presents the identity in a consistent format to a protocol, independent of the cryptographic algorithms used.

2 API and Layering Principles

The API describes capabilities and services between objects. Capabilities and services may freely be allocated to end systems (COs) or to servers. This enables functional allocations at the discretion of the developer. The same API may be used for network centric services and for P2P services. This enables services and service components to move, i.e. the architecture is agnostic to functional allocation and location.

Generic service elements may serve as building blocks for the full class of event oriented (i.e. data centric) and streaming oriented CO services.

The actual CO protocol (e.g. monitoring or remote control) is carried as payload by the user plane service elements of the API. These protocols may be proprietary, related to actual sensors or controllers, or adhere to standards. The architecture allows multiple application specific protocols to coexist, and it allows new applications and protocols to be defined without changing the basic API or its primitives (i.e. methods).

The service logically provided between COs, via the service API (Figure 2), shall be flexible in also offering subsets of the functionality. The idea, in particu-

lar applicable to device networks, is for the implementation to apply the simplest and most efficient protocol stack meeting the service requirements, with no or minimum processing and transmission overhead. It shall further be possible to increase the level of functionality by adding or including functional (sub) layers as required. Figure 2 shows aggregation of functionality from the set of service layers and protocol entities in the protocol stack. Any of the layers shown may be functionally transparent, depending on the protocol stack profile in use. Flexibility is enabled by resolution of the CO characteristics from the CO identifier.

The aggregated IP bearer service is logically provided at the top of the enhanced Network layer (i.e. to the Transport layer as shown in Figure 2). However, in an operational configuration, any of the layers may apply no protocol for communication with its protocol peer(s), i.e. the layer is empty except possibly from an inter layer service mapping (Figure 5).

A standardised API and protocols at the Internet layer (Figure 2) is key for support of inter operator ubiquitous services.

2.1 Service Aggregation

The functionality of the API service offered to COs is aggregated from the services offered by the sub-layers shown in Figure 2. The description adheres to the

well known ISO OSI model, and the approach relies on the same principles as the Platform-Based Design Methodology explained in [5]. The aggregated API functionality represents the Platform Design-Space Export.

The aggregated network service, i.e. the Aggregated IP bearer at Layer 3+, enhances the basic IP bearer service, which may be limited to offering a best effort service (IPv4 or IPv6).

2.2 Service Primitive Design Principles

According to the Representational State Transfer (REST) [6] principle, all basic service primitives shall offer a container for carrying data of a separately defined type. This creates an environment where clients and servers (i.e. objects) that encode their information the same way work together (i.e. share a common data definition). The uniform API interface can evolve over time. That is why it is built from three different parts, serving different and independent purposes:

- Identifiers (e.g. HITs),
- Methods (i.e. service primitives),
- (Document/data) types.

Each part is designed to change independently of the other parts. For example, new methods do not require the addition of new (data) types, and new (data) types

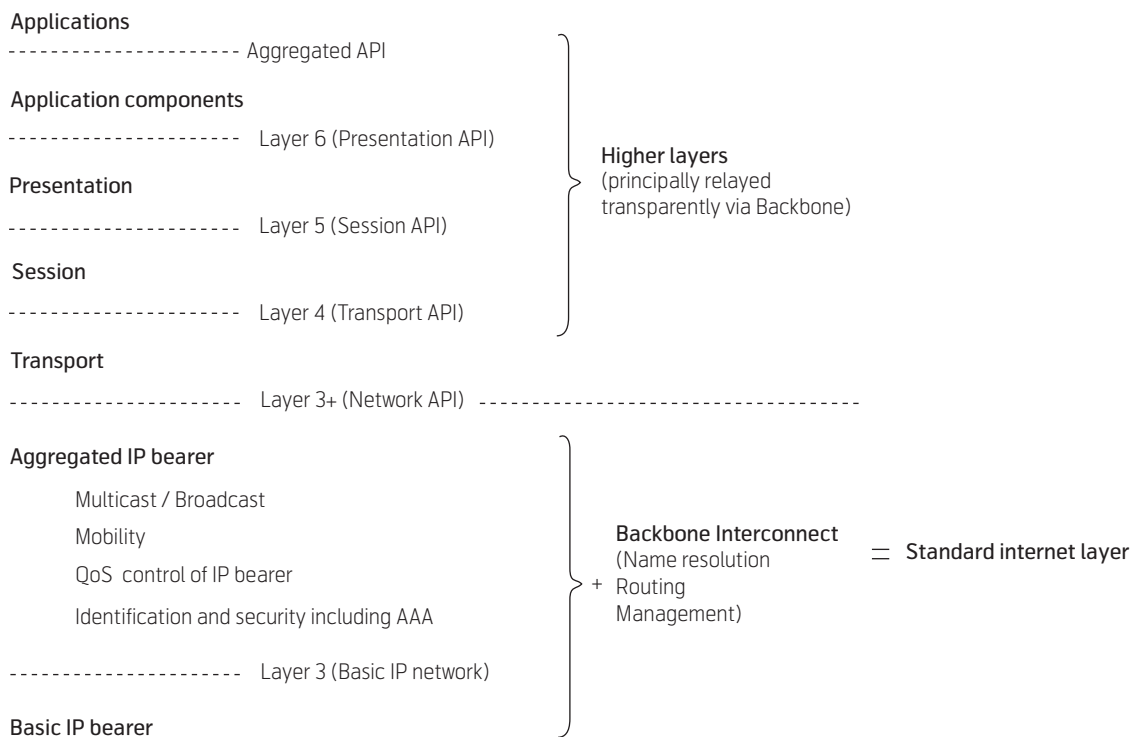


Figure 2 API and service aggregation

The layers above the Layer 3+ are the Transport, Session, Presentation and the applications. The service primitives offered by these layers are termed higher layer service primitives

do not require the addition of new methods. In object-oriented terms there is a single base class that defines the methods, but most methods have an “insert data here” slot. The information you want to transfer is added to this slot, along with the name of its (data) type. This avoids multiplying the number of standard methods by the number of (data) types.

3 API Service Elements

This chapter describes an example API service element (Micro-payment), and gives a brief overview of the complete service (Table 1). Appendix A describes selected service primitives, and an in-depth presentation is given in [7].

3.1 Micro-Payment Management Service

Settlement and compensation is allocated to the Management domain, as the functionality may be applicable to all layers of the protocol stack.

Fair and flexible compensation schemes between cooperating and competing parties are required to correlate resource consumption and cost, in order to avoid anomalous resource consumption and blocking of incentives for investing in infrastructure.

The proposed functionality is based on micro-payments as defined in [8] and is based on hash-chain trees, and involves minimal computational costs for packet authentication. The use of hash chain trees allows efficient generation and storage of hash values within a simple object with limited storage capacity. The number of cryptographic keys required is minimised, still allowing for fast verification of authentication data.

An object generates a hash chain of length N by applying a hash function N times to a random secret value P_N , the root of the hash chain, to obtain a final

hash P_0 , the anchor of the hash chain. The object commits to the chain by digitally signing the anchor with the private key. For each payment, the object releases a pre-image of the last hash value. For example, the object releases the hash value P_1 for the first payment. The receiver of the payment can apply the same hash function to the value P_1 to obtain the anchor P_0 . Since the hash function is one-way, only the object could have generated the hash value.

Figure 3 shows the overall payment process, where an object generates a hash chain of length N . The object commits to the anchor of the chain P_0 , the length of the chain, the value of each hash, and the vendor at which he/she wishes to spend the chain. Prior to payment, the object forwards the commitment to the vendor, who can verify its authenticity offline. For each micro-payment, the object releases the next (number of) payment hash(es) in the chain. The vendor can redeem the hashes at the Broker with whom the object has an account at a later date, by presenting the highest payment hash along with the signed commitment.

For additional efficiency it is proposed in [8] to apply an Unbalanced One-Way Binary Tree (UOBT) scheme, where the root of each chain is derived from another hash chain. The scheme is ideal when a large number of hashes are needed and the device has limited storage capabilities. Only the single value of the tree root has to be stored on the device to be able to reconstruct the entire tree.

A service provider generates revenue by charging for usage of network resources e.g. from objects, and may also provide other value-added services.

The employed micro-payment technology allows services to be charged for, and also allows the routers within the (access) network to authenticate packets by including a new hash value in each packet. This eliminates the need for any long-lived contracts between an object and a service provider.

3.1.1 Micro-Payment Primitives

The following service primitive initiates the acquisition of Length number of tokens (returned in Tokens), each valued Value and honoured by Vendor (e.g. via a macro transaction):

- Buy-Tokens (Broker-ID, Length, Value, Vendor-ID, P_0)

The Commit-Token is used to initiate payment, and it allows the Vendor to verify the validity of the tokens via the Broker:

- Commit-Token (Vendor-ID, Broker-ID, Length, Value, P_0)

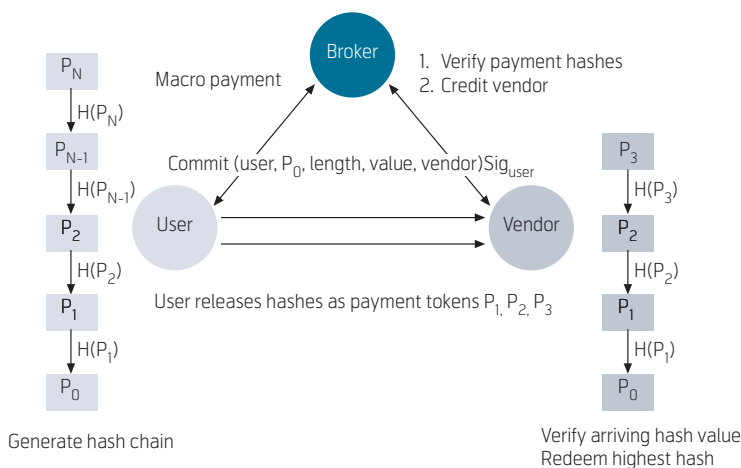


Figure 3 Micro-payments using hash chains

Explicit micro-payment is made by invoking the primitive:

- Submit-Tokens (Vendor-ID, Length, Tokens)

Length number of tokens is transferred to the Vendor.

The Submit-Tokens functionality may be integrated in other primitives, e.g. in the Data transmission primitives for real-time payment of utilized transmission resources.

The Vendor applies the following primitive for redeeming the tokens (e.g. macro-payment):

- Redeem-Token (Broker-ID, Tokens)

3.2 Service overview

Table 1 summarises the service functionality at each layer. Detailed examples are included in Appendix A.

4 The Topology and Network Elements

The IoT comprises two logically distinct, but closely coupled, network domains (Figure 4), i.e. the backbone and the device network domains. Both domains may be serving COs directly.

The backbone network offers ubiquitous interconnect for services at the Internet layer (Figure 2), while connected objects interconnect at the Application layer, i.e. the higher layer protocols are transparent to the Internet layer.

Device networks connecting clusters of objects must be flexible in technology and topology. A limited set of interfaces need to be standardised for interconnect with and via the backbone network (Figure 4, Figure 5). Device networks and their devices are expected to develop continuously for a longer period, and their technology basis is an important topic for continued research. There is a need to integrate simple low-end devices, e.g. with limitations in functionality and

Description	Service
<p>Application component sub-layer The application component sub-layer functionality may be offered as either complete services or as additional building blocks for in-house or third party services. The following represent example services.</p>	<p>Storage, retrieval Event reporting Presence Application routing Etc.</p>
<ul style="list-style-type: none"> • Presentation service. Defines the vocabulary for (control of) CO service applications; i.e. the data structures and commands required for COs to interoperate, e.g. for an advanced control and surveillance application. The actual monitoring or control protocol may be proprietary, related to actual sensors or controllers, or standards may be applied. 	<p>Application dependent. (Pending standardisation)</p>
<ul style="list-style-type: none"> • Session service. In this context a session shall be understood to represent the state of active communication between connected objects; i.e. it is not required to be established by e.g. the Session Initiation Protocol (SIP). 	<p>Invite (Destination, Profile)</p>
<ul style="list-style-type: none"> • Transport service. The abstract service for explicit transport protocol selection (TCP, UDP, MQTT, NIL) 	<p>Transport-Selection (Destination-CO, Protocol)</p>
<ul style="list-style-type: none"> • Network service at the Internet layer The Internet layer in Figure 2 is the IP network layer, extended to include the inter-domain functionality required for end-to-end maintenance of QoS control, Security, Mobility, Location, Multicast, Name resolution, Routing and Management. This functionality is required to be maintained across interconnected domains of the IoT, to support ubiquitous multifunctional services. This functionality is therefore critical for end-to-end service provisioning, since network elements in interconnected domains need to contribute to the functionality on a hop-by-hop basis. 	<p>Data transmission Multicast Mobility QoS control ID and Security Location & status Basic IP bearer (This equates to the IP sockets interface)</p>
<p>Management The management comprises functionality of generic nature that cannot logically be confined to a specific layer of the OSI stack.</p>	<p>Compensation Software upgrades Functional configuration Provisioning/fulfilment Assurance/fault handling</p>

Table 1 Services at each layer

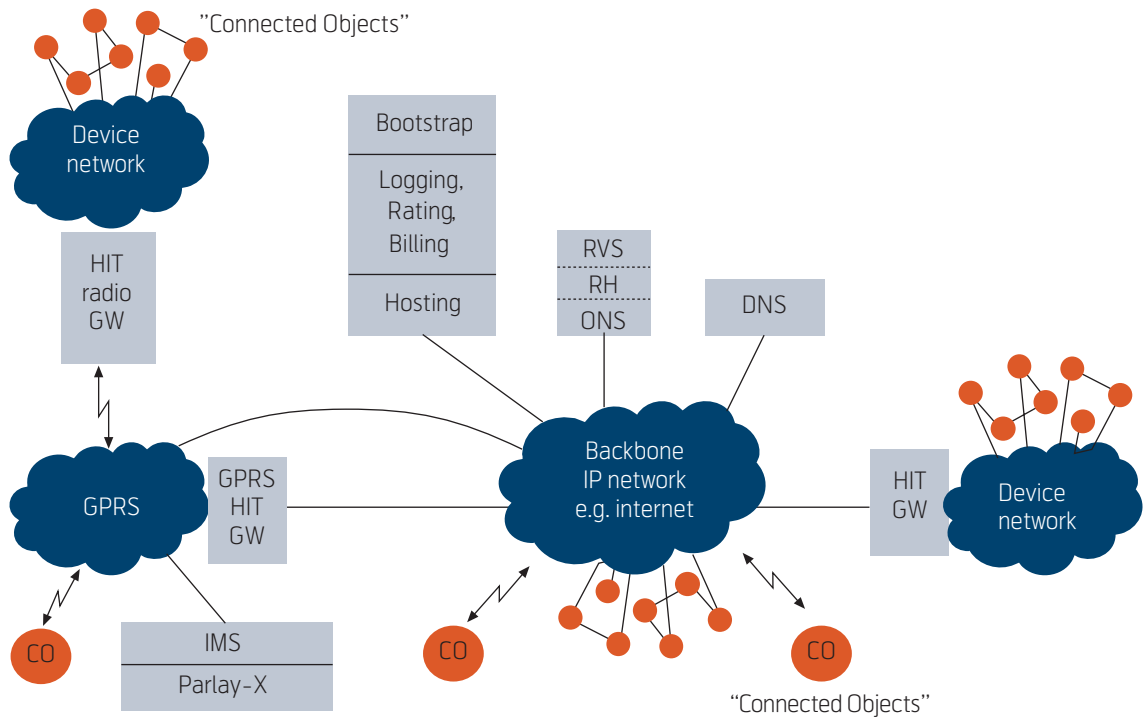


Figure 4 Components and their relation

The naming reflects the use of the Host Identity Protocol (HIP) protocol for e.g. obtaining name/address separation, security and mobility for objects

power supply. Such elements are interfaced efficiently through a flexible gateway architecture proposed in [9] and depicted in Figure 5. The naming reflects the use of the HIP protocol for e.g. obtaining name/address separation, security and mobility for objects.

The backbone architecture must include a limited set of interfaces for device network access and interconnect. A baseline proposal is defined in [9].

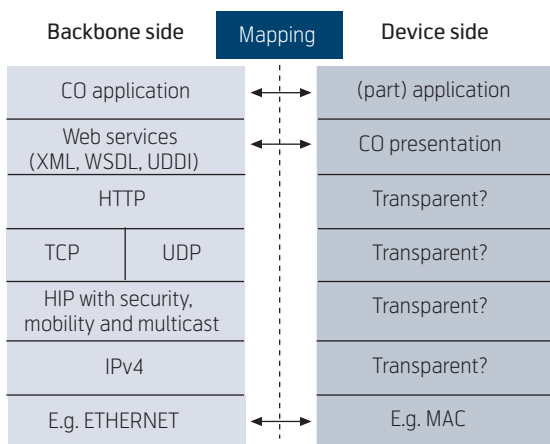


Figure 5 HIT gateway architecture

The communication scheme on the device network side of the gateway can be based on local lightweight protocols, e.g. at the link layer. The layers denoted "Transparent?" may or may not be instantiated at the device side of the gateway

All interconnect with non backbone compliant architectures is carried out at the rim of the backbone, and this effectively eliminates the need for the N square interconnect arrangements and gateways for N different technologies (e.g. service specific networks). The common backbone architecture will also reduce the number of different and variant network elements.

4.1 Gateways

The Host Identity Tag (HIT) gateway (Figure 5) is based on the Host Identity Protocol [3] and allows global addressing of COs while maintaining the use of IPv4 addresses. This is achieved by allocating a single public IP-address to a potentially large group of COs under control of a single HIT gateway. This is the address of the HIT gateway. This is similar to a care-of address in mobile IPv4. The gateway applies the HIT for addressing and/or identifying the actual CO. The HIT gateway also supports localized mobility management, as the IP-address of a CO would only change when the CO moves outside the control of its current gateway. The HIT gateway shall keep track of the location of all COs under its control. Each gateway shall be allocated a coverage area, allowing identification of objects within that area. The same HIT gateway may serve more areas in parallel in order to increase the granularity of CO identification.

Each gateway shall furthermore keep track of all its physical gateway neighbours, to allow extended area

search for COs. The HIT gateway may be multi-homed, e.g. via interfaces applying different network technologies.

The architecture of the gateway is depicted in Figure 5. A cluster of objects (i.e. a mobile network) may share mobility management through common gateway objects representing the cluster members towards the surroundings. The HIT gateway represents the members of the device network. In its basic version, the members of the mobile network are static. This may well meet the requirement of most device networks. However, there are situations where dynamic and hierarchical formation of such networks is highly desirable and efficient.

The mechanisms required for such dynamic mobility management is multi level registrations and indirection in the resolution handling at the Resolution Handler (RH). In addition to the basic name-to-address resolution, there is a need for a name-to-name resolution. That is, when an object (or objects) joins a mobile network it registers the name (HIT) of the caring gateway at its Rendezvous Server (RVS) at the RH.

The HIT Radio gateway is functionally equivalent to the HIT gateway, except for applying a radio interface (e.g. GSM, EDGE, UTRAN) for access to the GPRS network from single COs or CO clusters.

The GPRS HIT gateway offers interconnect of HIT based COs, e.g. on the Internet with Universal Mobile Telecommunications Service (UMTS)/GPRS native (non HIT) devices. The gateway handles both control and data/user-plane mapping. The control plane mapping is carried out for mobility management. The local mobility within the GPRS network will be transparent to the HIP based mobility management. A special requirement is that a CO attached to GPRS needs its local IP address to be registered at the gateway and at the mobility anchor point RVS (i.e. the Rendezvous server) in Figure 4. This registration must be carried out when the GPRS Attach / Packet Data Protocol (PDP) context creation takes place.

The user-plane mapping requires relaying of the IP user-plane between the GPRS Tunnelling Protocol user plane (GTP-U) and the HIP/IP protocol. This implies that the CO security association and the HIP protocol will terminate at the gateway.

4.2 Servers (RVS, RH, ONS)

The basic functionality of the Rendezvous Server (RVS) is to offer mobility anchoring, i.e. maintenance of the HIT to address bindings. It may also be engaged in traffic forwarding in cases where privacy is required. Event reporting shall also be handled by

the RVS serving the target CO, i.e. the CO at which events are monitored for reporting. This implies that the registrar and notification functionality shall be implemented at the RVS as well.

The Resolution Handler (RH) is an RVS extension offering generic name resolution from a flat namespace (e.g. HIT to address resolution). Retrieval of CO characteristics is part of the functionality (e.g. identification of protocol stack and other capabilities):

URI -> (HI -> HIT) -> IP address or HIT for indirection -> CO characteristics (e.g. protocol stack support)

This resolution is carried out by the following method:

- Resolve (HIT, Address)

In case of indirection, the HIT of an object will first resolve to the HIT of the caring gateway (GW). The RH recognises that this is not an address, so it repeatedly resolves all HITs until an address is finally encountered.

When a gateway receives a packet with a HIT that cannot be found in the local HIT to address binding table (i.e. routing table), it has to check if the HIT is inside a subordinate gateway. This is done by resolving the HIT relative to the current gateway by applying the following method:

- Resolve-Relative (HIT, Current-GW, Next-GW)

The resolution process stops when the resolution reaches current gateway and returns its predecessor as the Next-GW. The address of the Next-GW is subsequently locked up in the local routing table, and the packet may be forwarded.

For multi-homed objects, and potentially also for replicated "service or data objects", the HIT resolves to an IP address and a CO characteristic for each individual interface or instance. Each of these interfaces is identified by a locally scoped identifier (i.e. relative to the HIT).

The RH may be implemented as a self contained unit or integrated with the RVS. The RH is based on the Data-Oriented Network Architecture [4], introduced to obtain e.g. persistence in naming of data and services together with strong authentication. This functionality is not implemented in the legacy DNS based name resolution. In [4] RHs are networked and tiered, and the architecture is claimed to scale to serve the complete Internet.

The Object Naming Service (ONS) is part of the EPC Global Network [10]. The ONS may be integrated with the RVS entity, or can be implemented as a self-contained entity. The ONS offers name resolution for Electronic Product Codes (EPC):

EPC -> EPC-IS (i.e. the URL of the interface to the owner of the EPC manager code).

4.3 IMS/Parlay-X

NGN/IMS may implement the functionality of the CO service-primitives. NGN/IMS supports high volume streaming well, but small amounts of real-time data are not handled efficiently within the session oriented framework of IMS. The use of the SIP MESSAGE method for such data exchange may be proposed. However, the solution would require mechanisms like QoS differentiation, charging, multicast, flow- and congestion-control to be implemented for that purpose, and at the IMS signalling/control plane.

The solution is therefore a general, QoS controlled, connectionless service at the network layer, i.e. the Network API in Figure 2. This could be realized within the framework of NGN by adding a new connectionless subsystem to IMS. This is perceived equivalent to basing the CO service on a general QoS controlled IP network, except for the integration of OSS and BSS.

IMS/Parlay-X functionality is offered via the CO API for access to information supplied via IMS. This is described in more detail in [7].

4.4 Support Features

Bootstrap functionality shall be available to cover functionality like Dynamic Host Configuration, Identity management, Service discovery, Network and neighbour discovery.

Logging Rating and Billing are not further described in this presentation. The required functionality described in [7] covers logging of statistics, e.g. for planning purposes, and the information shall be sufficient as the basis for non-repudiation, rating and billing. Real-time rating is required to support Advice of Charge (AoC) and Prepaid.

Hosting, e.g. for third parties, shall be based on evolving technologies for virtualization. Persistent storage may be supported by the indirection as described for the RH.

5 Conclusion and Recommendations

The architecture urgently needs standardisation to take place in order to create a global (i.e. cross operator and service provider) market for end-to-end CO services. Since the architecture with the API shields applications from the underlying technology, it reduces efforts involved in service development, and at the same time allows services and technology platforms to evolve independently. Adoption of the architecture will allow the effect of economic network externalities to increase the total value of the market by supporting ubiquitous services on an end-to-end basis. This will provide efficiency in scale and scope in service infrastructures, service production and service development.

The Next Generation Network (NGN) and IP Multimedia Subsystem (IMS) initially only support the class of session oriented applications for use by COs. It is therefore important to implement the non-session (data) related part with interconnect to GPRS based networks. The architecture allows ubiquitous services also via GPRS/UMTS, and for native GPRS devices.

The architecture may serve as a vehicle for migration to the true all-IP IoT. However, this presupposes avoiding the mistakes made by Internet Service Providers (ISPs) for the Internet [2], i.e. by not cooperating and coordinating between operators in order to allow service and transport level interoperability at the Internet layer. This is required for service ubiquity, in order to enlarge the total global market, thereby benefiting the whole Information and Communications Technology (ICT) industry.

The IoT may evolve through a holistic approach to integration of COs, stimulating fair competition, economic growth and innovation. The value of the IoT market will grow due to new service offerings related to COs, but maybe more by the increase in value caused by positive network externalities resulting from general Internet service ubiquity.

Acknowledgement

The author is grateful for enlightening discussions, comments and contributions from Jan Audestad, Terje Jensen, and Anne-Grethe Kåråsen, all with Telenor.

References

- 1 Grønbæk, I. *Architecture for the Internet of Things: Focus on API*. ETSI Workshop on Machine to Machine Standardization, 4-5 June 2008. Sophia Antipolis, France, ETSI. http://www.etsi.org/Website/NewsAndEvents/2008_M2MWORKSHOP.aspx.

- 2 Faratin, P et al. Complexity of Internet Interconnections: Technology, Incentives and Implications for Policy. In: *Thirty Fifth Telecommunication Policy Research Conference (TPRC-07)*, The National Center for Technology & Law, George Mason University School of Law, September 28-30, 2007.
- 3 IETF. *Host Identity Protocol (HIP) Architecture*. May 2006. (RFC 4423)
- 4 Koponen, T et al. A Data-Oriented (and Beyond) Network Architecture. *SIGCOMM'07*, Kyoto, Japan, 27-31 August 2007.
- 5 Sangiovanni-Vincentelli, A L. Quo Vadis SLD: Reasoning about Trends and Challenges of System-Level Design. *Proceedings of the IEEE*, 95 (3), 467-506, March 2007.
- 6 *REST*. http://en.wikipedia.org/wiki/Representational_State_Transfer. Accessed 12.08.2008.
- 7 Grønbaek, I et al. *Abstract Service API for Connected Objects*. Fornebu, Telenor R&I, 2007. (R&I Report R 18/2007)
- 8 Tewari, H, O'Mahony, D. Real-Time Payments for Mobile IP. *IEEE communications-magazine*, February 2003, 126-136.
- 9 Grønbaek, I, Jakobsson, S. *High level architecture for support of CO services*. Fornebu, Telenor R&I, 2007. (R&I Report R 37/2007)
- 10 *Electronic Product Code*. http://www.epcglobal-inc.org/standards/architecture/architecture_1_2-framework-20070910.pdf.

independent manner by applying indirection. The name resolution from the HIT to address is carried out by Resolve method:

- Resolve (HIT, Address)

In case of indirection, the HIT of an object will first resolve to the HIT of the caring (HIT) database server. The resolver recognises that this is not an address, so it repeatedly resolves all HITs until an address is finally encountered.

This functionality is similar to applying Distributed Hash Tables (DHTs) for data management, except for giving the network full control of the allocation and location of database resources. Data may be replicated by applying the same resolution mechanisms as for multi-homing, i.e. the HIT would resolve to multiple addresses. This mechanism can support data resilience as well as load distribution for multiple retrievals.

The following primitives are used to store, retrieve and delete data-objects:

- Store-Object (OID, Data-Object, Scheme, Result)

Scheme indicates time to live, need for replication and any access restrictions and key(s).

The Result parameter returns the status of the operation.

- Retrieve-Object (OID, Data-Object, Scheme, Result)
- Delete-Object (OID, Scheme, Result)

In the Retrieve and Delete primitives the Scheme parameter mainly conveys the required access rights for the operations.

Event Reporting

The following primitive is used for event subscription at the event server (Registrar-CO-ID), for events at the Target-CO-ID (both could be the same CO):

- Event-Subscription-Send (Registrar-CO-ID, Target-CO-ID, Parameters)

An event subscription shall be confirmed by an event notification from the Registrar. The event notification indicates an event at the Target-CO-ID, or a subscription for events.

- Event-Notification-Send (Subscriber-CO-ID, Target-CO-ID, Parameters)

The Event-Report primitive is used for carrying reports from a CO (e.g. a simple sensor) to the recipient (e.g. the Registrar event server).

- Event-Report-Send (Registrar-CO-ID, Parameters)

Appendix A Example Service Primitives

A description of how the IMS/Parlay-X service can be offered as a part of the API can be found in [7].

Application Component Sub-layer

The application component sub-layer functionality may be offered as either complete services or as additional building blocks for in-house or third party services. The following represent example services.

Storage and Retrieval

An object may not only be a node in the Internet of things. It may also be a data object, e.g. stored in a node (i.e. in an end-system or in a network element). By assigning a Host Identity Tag (HIT) to a data object, it may be stored and retrieved in a location

The Parameters may specify a distinct value or a set of values (e.g. representing upper and lower limits). Events may also be complex in that more COs and parameters may be involved.

CO Presence

The following primitive is used for dynamic CO presence registration:

- Register-Send (Registrar-CO-ID, Parameters)

A registration shall be confirmed by an event notification to the registering party, and to all parties having subscribed on this registration event.

Presentation Service

This layer defines the vocabulary for (control of) CO service applications; i.e. the data structures and commands required for COs to interoperate, e.g. for an advanced control and surveillance application.

The actual monitoring or control protocol may be proprietary, related to actual sensors or controllers, or standards may be applied. The identification or standardization of such protocols is for further study, but architecturally the vocabulary of such protocols is allocated to the Presentation layer.

Session Service

In this context a session shall be understood to represent the state of active communication between connected objects; i.e. it is not required to be established by e.g. the Session Initiation Protocol (SIP). A session may thus even be represented by a link layer association.

Transport Service

The abstract service for explicit transport protocol selection is composed of the following primitive:

- Transport-Selection (Destination-CO-ID, Protocol).

The initial protocol offerings will be UDP, TCP, MQTT(s) or NIL.

Primitives from subordinate layers will be applied, e.g. for sending and receiving data.

Network Service at the Internet Layer

The Internet layer in Figure 2 is the IP network layer, extended to include the inter-domain functionality required for end-to-end maintenance of QoS control, Security, Mobility, Location, Multicast, Name resolution, Routing and Management. This functionality is required to be maintained across interconnected domains of the IoT, to support ubiquitous multifunctional services. This functionality is therefore critical for end-to-end service provisioning, since network

elements in interconnected domains need to contribute to the functionality on a hop-by-hop basis.

The network service aggregates the functionality from each of its subordinate layers into the service provided to the transport, or directly to COs.

Data Transmission

The following primitives are defined for sending and receiving unsecured and secured data:

- Send-ID (CO-ID, Data)
% sends to the identified CO (CO-ID)
- Receive-ID (CO-ID, Data)
- Send-SA (SA, Data)
% sends to the specified Security Association (SA)
- Receive-SA (SA, Data)

A combined mechanism for sending data and paying for its transfer is defined as follows:

- Send-ID-Token (CO-ID, Data, Length, Token)
- Send-SA-Token (SA, Data, Length, Token)

The Length parameter specifies the token rate to be submitted.

Tokens are acquired and redeemed by the primitives for Settlement and compensation defined previously.

Multicast

The service for managing multicast for COs is composed of the following primitives:

- MC-Group-Open (Group-ID, Profile)

Profile specifies the characteristics, e.g. security level.

- MC-Group-Close (Group-ID)

The basic Send and Receive primitives are used to send and receive secured or unsecured data.

The joining and leaving of a multicast group is achieved by the following primitives:

- MC-Group-Join (Group-ID)
- MC-Group-Leave (Group-ID)

Mobility

The API for specifying mobility management to be applied for a CO is composed of the following primitives:

- MM-Register (CO-ID)
- MM-End (CO-ID)

QoS Control

The API for specifying a default QoS to be applied is composed of the following primitives:

- QoS-Set-Default (Profile)
- QoS-Set-Default-Confirm (Profile, AoC)

Advice of Charge (AoC), as returned from the network, provides the information required for estimation of the charging rate.

The API for explicit QoS control is composed of the following primitives:

- QoS-Set-Path (Destination-CO-ID, Profile)
- QoS-Set-Path-Confirm (Destination-CO-ID, Profile, AoC)

ID and Security

The API for security association creation is composed of the following set of control primitives:

- SA-Create (CO-ID, Profile, SA)

The Profile shall indicate both the security requirement and the involved algorithms.

Location & Status

The service for network assisted reporting of status and location of user terminals includes the following set of control primitives:

- Location-Request (CO-ID-Set, Scheme)
% The CO-ID-Set identifies one or more COs
- Location-Response (CO-ID-Set, Scheme, Status, Coordinates)

The Scheme defines the desired coordinate system. Status indicates the validity of the returned coordinates.

The following primitives are used to request the network to identify object(s) known to be at a specific location:

- ID-Location-Request (Scheme, Coordinates)
- ID-Location-Response (Scheme, Coordinates, Status, CO-ID-Set)

The CO-ID-Set identifies zero or more COs.

Reporting of end-system supplied location and status is done by applying the following primitives (e.g. to Geographic Information Systems (GIS) central information portal):

- End-Location-Report (CO-ID, Scheme, Status, Coordinates)
- End-Location-Report-Indication (GIS-ID, Scheme, Status, Coordinates)

Basic IP Bearer

The network service of the basic IP bearer is simply composed of the following primitives for sending and receiving data:

- Send-IP (To-IP-address, Data)

- Receive-IP (From-IP-address, Data)

This equates to the IP sockets interface. A more complete overview is given in [7].

Abbreviations

AAA	Authentication Authorisation and Accounting
AoC	Advice of Charge
API	Application Programming Interface
BSS	Business Support System
CO	Connected Objects
DHT	Distributed Hash Table
DNS	Domain Name Service
EDGE	Enhanced Data rates for Global Evolution
EPC	Electronic Product Code
GIS	Geographic Information System
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GTP-U	GPRS Tunnelling Protocol user plane
GW	Gateway
HIP	Host Identity Protocol
HIT	Host Identity Tag
ICT	Information and Communications Technology
ID	Identifier
IETF	Internet Engineering Taskforce
IMS	IP Multimedia Subsystem
IoT	Internet of Things
IP	Internet Protocol
ISP	Internet Service Provider
M2M	Machine to Machine
MQTT	MQ Telemetry Transport
NGN	Next Generation Network
NIL	Nothing; zero
OID	Object ID
ONS	Object Naming Service
OSI	Open Systems Interconnect
OSS	Operations Support System
P2P	Peer-to-peer
PDP	Packet Data Protocol
QoS	Quality of Service
REST	Representational State Transfer
RH	Resolution Handler
RVS	Rendezvous Server
SA	Security Association
SIP	Session Initiation Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications Service
UOBT	One-Way Binary Tree
URI	Universal Resource Identifier
URL	Uniform Resource Locator
UTRAN	UMTS Terrestrial Radio Access Network

Inge Grønbæk received his M.Sc. degree in computer science from the Technical University, Trondheim, Norway, in 1977. He has since been involved in protocol design and implementation for systems involving circuit switching, packet switching, and message handling. He was for a period seconded from Siemens Norway A/S to NATO at SHAPE Technical Centre (STC) in the Hague, Holland, where he was working in the area of protocol standardisation. In 1994 he took a position as Chief of technical development at the Networks division of the Norwegian Telecom. In September 1998 he transferred to Telenor (former Norwegian Telecom) R&D as Senior Adviser. In the period 1999 to 2002 he was the Telenor representative in the international 3G.IP Focus Group aiming at applying IP technology in third generation mobile systems. He recently represented Telenor in the TISPAN WG4. His current focus is in the area of IP based network evolution, in particular on architecture and service production for M2M and connected objects.

email: inge.gronbak@telenor.com