

# Technical Aspects to Consider in an Architecture

TERJE JENSEN



Terje Jensen is Research Manager in Telenor Group Business Development and Research and Senior Researcher at Q2S/NTNU

Depending on whom you ask, an architecture could be explained in a wide range of ways. However, on a conceptual level, an architecture describes a set of components and the way they are related. This paper addresses a range of concepts to understand a technical architecture. Further details and examples are given in accompanying papers in this issue of *Teletronikk*.

## 1 Introduction

Use of the term architecture flourishes in the literature. Commonly there are different interpretations by the authors as well as the readers. It is commonly understood to mean the *description of a set of components, their functions and how they are related*. In a dictionary, however, it is often described as the art and science of building and design or style of buildings. Taking a closer look at the actual picture of building houses, one finds that every building also consists of a number of components that are interconnected in order to raise the building or for decorative purposes.

When studying standardisation documents, one may also get the impression that architecture is used as a list of things to remember when implementing a system. Applying the standards it could also be quite easy to forget that several architectures are actually referring to functions and logical entities, not physical implementation. In all, it is essential to understand different perspectives of an architecture and how to understand and apply the architecture description. This is further elaborated in Chapter 2. Chapter 3 examines different ways to decompose an architecture in order to arrive at manageable units. It is also important to see how these units are related.

Identifiers are key information as these explain how to recognize a component. Identifiers are also needed in order to define addresses and find routes to a component as Chapter 4 elaborates. Most technical architectures use protocols for exchanging information between components. Chapter 5 dives into some aspects of protocols, although this is a vast field in itself.

Protocols are for example used when interworking between domains, such as between two actors. Chapter 6 elaborates on several aspects to consider when evaluating interworking solutions that are part of the architecture.

It is also important that the technical architecture is not restricted to a static implementation, but supports dynamics in the sense that units and systems can be

introduced and removed. This is elaborated on in Chapter 7 and Chapter 8. Management aspects from TM Forum are addressed in Chapter 9. A few examples pointing to practical issues are included in Chapter 10.

As mentioned above, each of the items are huge by themselves. Hence, this paper simply scratches the surface and gives some pointers to relevant aspects. Several of these aspects are further described by accompanying papers in this issue of *Teletronikk*.

## 2 Architecture Perspectives

As addressed above, *an architecture* does

- *Place relevant components into a coherent whole; and*
- *Provide structure and relations between relevant components.*

### 2.1 Architecture Discussions

However, the description opens for several questions, such as: Which are the relevant components? What is this 'coherent whole'? How are the relations being understood? These are just a few of the immediate issues arising. In result, one may claim that making an architecture could well belong to the *art* category. However, as further elaborated in other papers in this *Teletronikk* issue, there is quite an extensive experience on defining technical architectures for the telecom area.

On the other hand, some may say that creating an architecture for a system is different from the scientific discovery. From that it could follow that architecture work rather belongs to the area of *systems engineering*. Systems engineering is a process that includes specification of requirements, invention of new approaches and a complex process of trade-offs and balance among different objectives. There are also trade-offs made among different stakeholders and constituents.

Validation of the architecture then involves a process that firstly tests specific proposals for new mecha-

nisms to better understand their limitations and relative strengths and advantages. Secondly, it tests more complete architectures to determine fitness of purpose within the multi-dimensional space of requirements.

Working on defining so-called *holistic architectures could be quite challenging*. There are several characteristics adding to the complexity:

- Architecture level requirements are often broadly defined and hard to capture.
- Requirements are often conflicting and multi-dimensional.
- Architecture requirements are frequently concerned with the behaviour of the system over a long period of time rather than a single instant or short interval.
- The architecture has to relate to the environment in several manners, also allowing for components to be introduced and removed.

The process of evaluation for a large, multi-dimensional system is itself complex. Evaluating the balance amongst a set of multi-dimensional requirements is much harder and less precise than a simple optimization of a variable. The process usually proceeds iteratively, with first design being subjected to evaluation that leads to revised and refined designs. Some concerns are illustrated in [Jens09b].

There are multiple ways to devise *components of the architecture*, some examples are:

- Functional elements
- Physical elements
- Data-oriented
- Process-oriented
- Layered (providing levels of abstractions)

Being present at a meeting discussing all these aspects at the same time could be quite entertaining – or frustrating – depending on the expectations. It is important to be aware of all the flavours of understanding the areas. This is to avoid tedious misinterpretations and wasted efforts.

## 2.2 Key Characteristics

There is no single criterion for a best-behaving architecture, although several guidelines could be obtained depending on the purpose and scope. For example, a telecom architecture, as a case of a carrier-grade structure should comply with a number of characteristics:

- *Modularity*. The architecture should be broken into building blocks that clearly define what is to be the functionality of each block.

- *Manageability*. A robust management model should be defined to discover, monitor and control each of the components.
- *Interoperability*. Building blocks from different vendors should be allowed. This ensures that each building block has predictable, well-defined behaviour.
- *Availability*. Each building block should have corresponding dependency schemes to ensure a proper level of service availability. Typically, this implies eliminating any single point of failure.
- *Testability*. Being able to test each building block and the overall system is a key carrier-grade requirement. These tests are to verify proper operation.

The architecture is to support system realisations. It is also used as guidance when planning further development of a system portfolio. The architecture has to reflect a number of key capabilities of the system at hand. In general, any larger telecom system adheres to the following characteristics (ref. [Aude08]):

- *Concurrency*. Applications require processing of a number of current processes; both within a node and in different nodes. This has been seen already from the distributed telephone network which, at its time, was the largest distributed system worldwide.
- *Weak process coupling*. Only a fairly small number of messages are exchanged between any two pairs of processes within a node or between nodes. Hence, the coupling between processes is weak.
- *Information hiding*. The interchange of messages between processes within an exchange and between exchanges requires very little knowledge of how the processes are actually designed. This means that the interchange in most cases is based on a small but consistent set of knowledge about the cooperating processes (the interface specification).
- *Weak synchronization*. Processes are weakly synchronized in the sense that a process has no *a priori* knowledge of when and with which initiation parameters it will be invoked by another process. Therefore, the process must be allowed a wide range of processing times depending on context.
- *Fault recovery*: Each process should be able to recover from faults on its own. If a process fails, all processes depending on it must be able to return to a safe state independent of the faulty process.

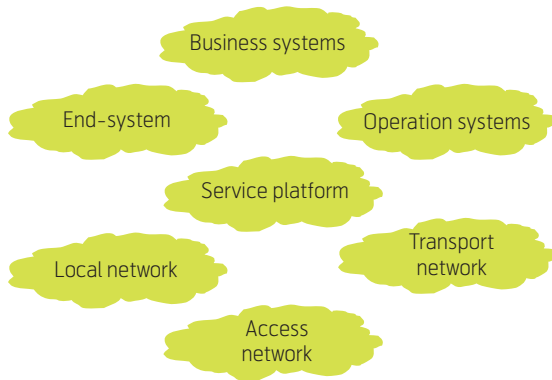


Figure 1 Illustration of domains for a traditional telecom operation

### 3 Domains, Segments and Layers

Riddle: How to study an elephant?  
 Answer: Piece by piece ...

One crucial talent is to divide a challenge at hand into smaller pieces that can be analysed and solved in order to solve the overall task. Similarly with the holistic architecture, components must be sorted into those that should be examined in further detail and those that can be placed outside the scope (for the time being).

#### 3.1 Classic Telecom Areas

A classic division into domains for telecom systems is depicted in Figure 1. As for most technical systems, this has been de-composed into a set of *domains with different purposes*:

- The end-systems represent terminals, handsets and other devices connected to a network that want to communicate.
- A local network is commonly present within a telecom customer's premises. This can be used to interconnect a number of end-systems within the building, room, etc.
- An access network connects the customers' network to a wide-area telecom service network. Note that several access networks may be combined in tandem and also in different layers. An example of the latter is found when a mobile access network utilises fixed access networks.
- The transport network does connect different sites of the wide-area network.
- Service platform typically contains data and logic for executing service adapted to users, locations and other parameters.

- Operational systems contain functions for configuration, failure handling, performance handling, etc.; basically functions for ensuring that the service is operational.
- Business systems would commonly contain customer relations, billing and other functions for interacting with customers and other partners on the business levels.

Note that not all these areas have to be present in an actual implementation of a telecom network.

Between the local network and the wired access network there has typically been a service demarcation point. This is the point where the public service is offered. However, this may not be quite the same for mobile systems. Nor does this correspond with the trends for broadband services where more of the local network's capabilities seem to be considered in order to provide an end-to-end service view.

As a result the following question appears: What is actually the service you are selling? A service is typically provided at the border of a domain. Some sources also refer to this as the service demarcation point. Note, however, that the service concept is applied between domains as well as between layers. So, the service term by itself is overloaded with different interpretations, simply something that appears on the interface between two components.

#### 3.2 Decomposing by Multiple Dimensions

A telecom network or system is by nature composed of a number of components. Each of the components has its characteristics. Moreover, some of the components can be grouped based on certain characteristics. These concerns are commonly applied when discussing appropriate solutions for the components and how they can be put together to compose a network or a system. That is, each of the components would implement certain mechanisms and be designed according to their role(s) in the architecture.

Looking at studies deriving from systematic system descriptions, one frequently observes a division into domains and strata, see Figure 2. Here, *domain* refers to geographical separation while *stratum* refers to a functional separation. These are depicted by horizontal and vertical separations. A basic example of a set of strata is the 7-layer Open System Interconnection (OSI) model. Here, however, stratum is used in a more general interpretation, although similarities with the OSI model can be recognised.

As one example, zooming in on IP-related aspects, the stratum referring to routers is commonly placed

in the centre. Several of the relations with upper and lower strata (referred to as layers) impact the behaviour of the IP stratum. So, when discussing interconnecting IP routers (and other IP elements), one should keep in mind that lower layer functionality is used for carrying the IP packets. IP packets are used for carrying higher layer information, and other functions are used for finding where to direct the information. According to OSI, IP will be placed in layer 3.

Besides carrying information from higher layers, other types of functionality are also present. Typically both control and management functions are found. The actual distinction between user data, control and management might be blurred in some cases. However, schematically this can be illustrated as in Figure 3. Some refer to this as the cube of the International Telecommunication Union – Telecommunication Standards Sector (ITU-T).

Briefly, the main purposes of each of the planes are:

- *User plane* – to convey the user information (information from higher layers);
- *Control plane* – to control traffic flows and resource configurations;
- *Management plane* – to manage network resources, including fault management, configuration management, accounting management, performance management, security management.

These are also found in ITU-T's Next Generation Network Architecture (see [Jens09n]).

Any larger network would typically have functions belonging to all these planes, although implementations vary. One objective is to find efficient complete solutions and combinations of functions that incorporate all needed functionality.

### 3.3 Domains

Domains are commonly identified according to some kind of geographical arrangement. That is, separate domains are physically dispersed. A set of domains are interconnected at certain points, see Figure 4. Interconnection points are realised at *border* sites, eg. at a border router for an IP network.

When interconnecting a customer to the IP core network, a border router is frequently called an *IP edge router*. Then, one also meets the expression *Autonomous System (AS)*. A common understanding of an AS is a set of routers under a single technical administration. This implies that an AS has its characteris-

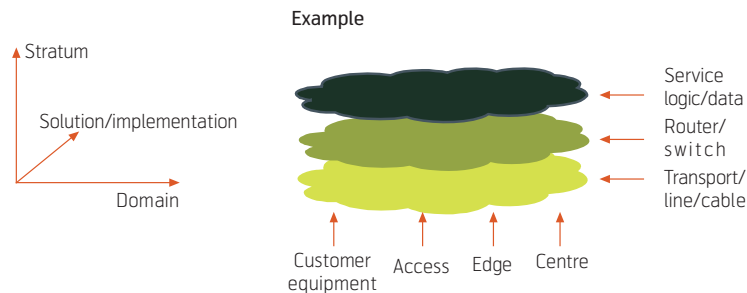


Figure 2 Identifying components in the network

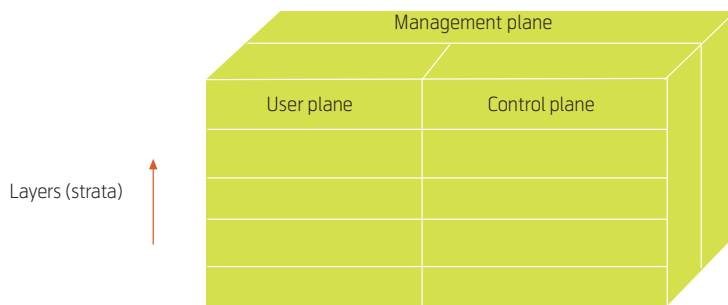


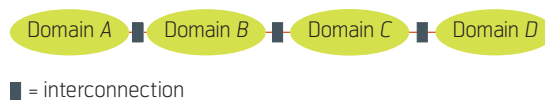
Figure 3 User, control and management activities

tics and is commonly managed by a single organisation (within the same trusted unit).

Examples of domains are seen in Figure 1. These may well have distinct characteristics implying that different solutions can be used for each of them. This also implies that the corresponding architecture guiding will differ.

For example, the limited coverage of a LAN allows for adding transmission capacity without radical increase of the cost. Moreover, a company would often own its LAN such that sophisticated mechanisms for accounting/charging may not be needed. The same goes for security solutions, although certain protection means are typically introduced, like passwords, limited access rights, and so forth.

An access line may be dedicated to a single customer, resulting in relatively low average utilisation (eg. for a residential customer). Hence, a significant fraction of the overall cost is commonly associated with the access network. Shortening the dedicated capacity introducing multiplexing/concentrating equipment is often used to seek to reduce the overall cost. For several networks the relative cost of the access portion is



■ = interconnection

Figure 4 Schematic illustration of interconnected domains

fairly high, often in the area of 60% – 80% for public networks, although this depends on the solutions used.

On the IP level the access network commonly looks like a tree or a star network, with the edge router located at the root (or in the centre). However, on the transmission layer ring structures could be used, eg. for dependability arguments.

Looking at the access link for a single user, the capacity of the links should be according to the maximum demand from that user. However, the peak load (bit rate) may very well be rather high compared to the average load, eg. during the day. An analogy is found in the telephony network where the access link is in use during a phone conversation, but idle most of the time. Installing a capacity much higher than the average load is a reason for the rather high cost of the access network.

A core domain does usually carry traffic from a greater set of customers resulting in higher capacity links and routers. A certain averaging effect of the traffic, eg. during the day is also commonly observed, for example related to the different customer types and use of services.

The edge of the core network usually has a number of features related to individual users, like access lists and monitoring mechanisms. Edge routers will then implement such features. Towards other operators, border routers, possibly with similar functionality can be present.

Providing efficient traffic handling implies that appropriate solutions must be available in all the domains involved. However, for customer equipment, it is usually expected that the cost of providing capacity is relatively low, implying that capacity is added in case a performance problem emerges. For other domains mechanisms for differentiated handling of

services (and traffic flows) seem to be gradually introduced. In addition to differentiating between services, differentiating between customers would also be likely (although this could again be seen as a kind of service differentiation).

When looking for potential performance bottlenecks, it is essential to include the end-systems as well as the processing capabilities in all domains. This means that efficiency of protocol software (as well as software of other traffic handling mechanisms) is a pivotal point. As the capacity evolution of transmission outpaces the computer capacity growth, the system designers may rather concentrate on achieving high transfer speed (putting data on the output interface) than optimised utilisation of the transmission bandwidth when seen from an end-system point of view. The network operator view may be more balanced, however, as a huge number of traffic flows will be present.

The discussion above illustrates that mechanisms in the different domains have to be considered together in order to ensure an efficient operation providing good user experience. This is where the technical architecture comes into play providing guidance on which functions that could go where. However, it has to go beyond simply providing a list of options. It must, in fact, give recommendations on which mechanisms to apply where. The latter is commonly referred to as *target architecture*, while the list of options could be considered a *reference architecture*.

### 3.4 Layering

The layering concept is often applied when introducing layers of abstraction; for example, when a layer should provide certain services to another layer (above). The above layer does not need to be aware of the details within the layer below. This notion, however, should be considered in a more general view as a system does not need to know how another

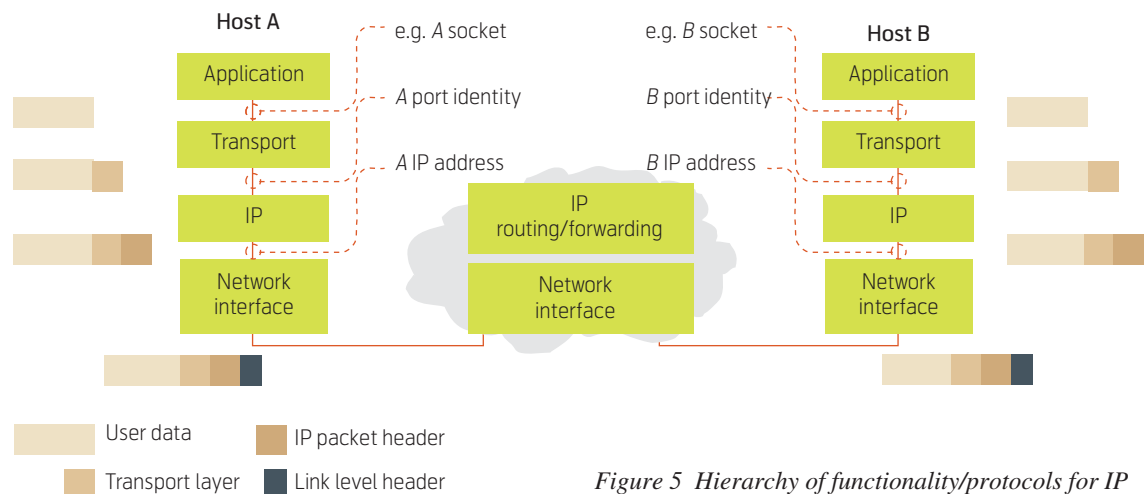


Figure 5 Hierarchy of functionality/protocols for IP

system actually works (ref. information hiding in Section 2.2). The system needs to understand how to interact in order to get the proper actions or achieve the required information.

One example of layering is found for the OSI model (Open System Interconnection). An illustration is given in Figure 5 for IP. Numerous protocols could be used below IP accessed through the network interface. A number of protocols could also be used in the transport layer, although Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are among the most popular ones.

A principle behind the layering for system interactions is that a layer entity at a destination sees the same object/message as sent by the corresponding layer entity in the source as illustrated in Figure 5.

In this manner, an application interacts with a transport protocol. The application chooses the format of data transfer. Two examples are stream (a longer flow of information) and transaction (an exchange of a single information unit). In the transport layer an association with the transport layer at the destination is established (frequently called end-to-end, although when interworking units are involved the user information could even be carried further before the final destination is reached). The transport protocol divides the data into units, sometimes called segments and transfers these units to the IP layer. In the IP layer, the data units are enveloped by the IP header information and transferred to the network interface. That interface, in a terminal/host, would be interacting with hardware drivers.

Even QoS parameters can be assigned to layers. This implies that different aspects can be discussed on certain layers, for example allowing for 'hiding' characteristics of lower layers. However, it also raises the need for finding the mapping between parameter values on the different layers. When fairly generic layers are used, such a mapping may not be straightforward, balancing the requirements of the upper layers with efficient utilisation of resources seen by layers below.

One example of the discussion of allocating functionality to layers within a network is found when IP is realised over optical. Then, three interconnection models have been sketched:

- *Overlay* model. The routing, topology distribution, signalling protocols are independent for the IP/MPLS and the optical network.
- *Augmented* model. Routing instances in the IP layer and the optical network are separated by

information exchanged (eg. IP addresses are known to the optical routing protocols).

- *Peer* model. The IP/MPLS layers act as peers to the optical network. Then a single routing protocol instance can be used for the IP/MPLS network and the optical network.

These models give a certain degree of implementation complexity; the overlay being the least complex one for near-term deployment and the peer model the most complex one. As each of the models have their advantages, an evolution path for IP over optical network may be seen.

A possible migration path is to start with the simpler functionality, meaning the domain service model with overlay interconnection and no routing exchange between the IP and the optical network. A provisioned interface would be expected. The next phase of the migration path is to exchange reachability information between IP and the optical network. This may allow for light-paths to be established in conjunction with setting up LSPs. The third phase of the migration is supporting the peer model.

On the general level, the fact that there is a number of layers present raises the question of whether these could be utilised in a coordinated manner to improve the recovery schemes. Three main classes of schemes are:

- *Uncoordinated approach*. This may be considered as the simplest approach, that is to install recovery mechanisms in several layers, but without any coordination between them. An advantage is that it is simple to install and operate.
- *Sequential approach*. Realising that some coordination would likely result in improved utilisation, different options are investigated. The next level of complexity is seen where the responsibility for resolving an anomaly is handed over to the next layer when it is clear that the current layer is not able to fully deal with the task. In principle any sequence of layers can be thought of; however two obvious ones are *bottom-up* and *top-down*.
- *Integrated approach*. This is based on a common integrated scheme for the stack of layers. Hence a full overview of all the layers is needed in order to decide which layer and which actions to take. In principle, this approach is the more flexible one; however the algorithms have to be devised and implemented.

Different mechanisms introduced at different layers may counter-act each other. For example, mapping between different service classes at the layers must be done consistently to arrive at the wanted effect. To do this the (functional/protocol) architecture provides guidance on which functionality/mechanism to place on which layer.

## 4 Identifier – Name – Address – Route

An *identifier* identifies *who* an object is, a *name* gives *what* an object is, an *address* identifies *where* it is, a *route* tells *how* to get there, a *path* says which sequence of *steps* to traverse, and a *link* would be a *step* in the path.

### 4.1 Identifier – Name – Address

There is a broad range of identifier schemes used within the telecom and internet area. Identifiers include phone numbers, e-mail addresses, IP addresses, port numbers, domain identifiers, customer identifiers, and so forth. All these do likely have their own formats and sets of rules for interpretation. As also

noticed the terminology is not consistent either, in particular the mix of identifier, name and address.

A simple example is depicted in Figure 5. Three identifiers are shown, some with only local scopes, while others have global scope. In order to allocate identifiers, certain rules and authorities may be needed. A hierarchy of authorities could well be used for some cases, for example when IP addresses are allocated. There are both global and regional registers for allocating IP address ranges to individual providers. A provider will assign IP addresses to individual users/devices, either in a dynamic or (semi-)static manner.

Take the port numbers in Figure 5. How should these be allocated? In principle there are two ways this can be done; i) universal assignment, and ii) dynamic binding. In the former, a centralised authority is typically used to assign port numbers and publish the results (could well be done hierarchically). In the latter case port numbers are assigned when needed. This implies that each program that needs a port number is assigned one on demand. In order to know the port number on a remote host, an enquiry has then to be sent to that host, which replies with the proper port number to use.

Typically a combination of the two ways of assigning port numbers has been chosen; some numbers are fixed while others are used dynamically. The ports refer to identifiers used on the transport layer (TCP/UDP). The port identifiers are included in the UDP/TCP headers.

Referring to names used on the IP layer, a Domain Name System (DNS) is used to translate between more high-level names and IP addresses. For instance, the name *viking.telenor.com* could translate into a 32 bit IP version 4 address (and the other way round). Such a naming scheme would then be used to assign names throughout the IP-based networks. It also provides a large-scale example of the client-server concept as a DNS server would be enquired in order to make a translation between the name and the address, see examples in Figure 6.

In principle, the resolution algorithm used for the translation proceeds from the top (top-level domain) and continues down. There are two ways of using the domain name system: i) by asking the name servers one by one until the resolution is complete, or, ii) by asking a name server to do the complete resolution (lower option in Figure 6). In both cases, the requester forms a query that includes the name to be resolved, in addition to other fields. When a server receives a request, it checks to see if the name is within its area (in the data base). If so, it translates the name into the

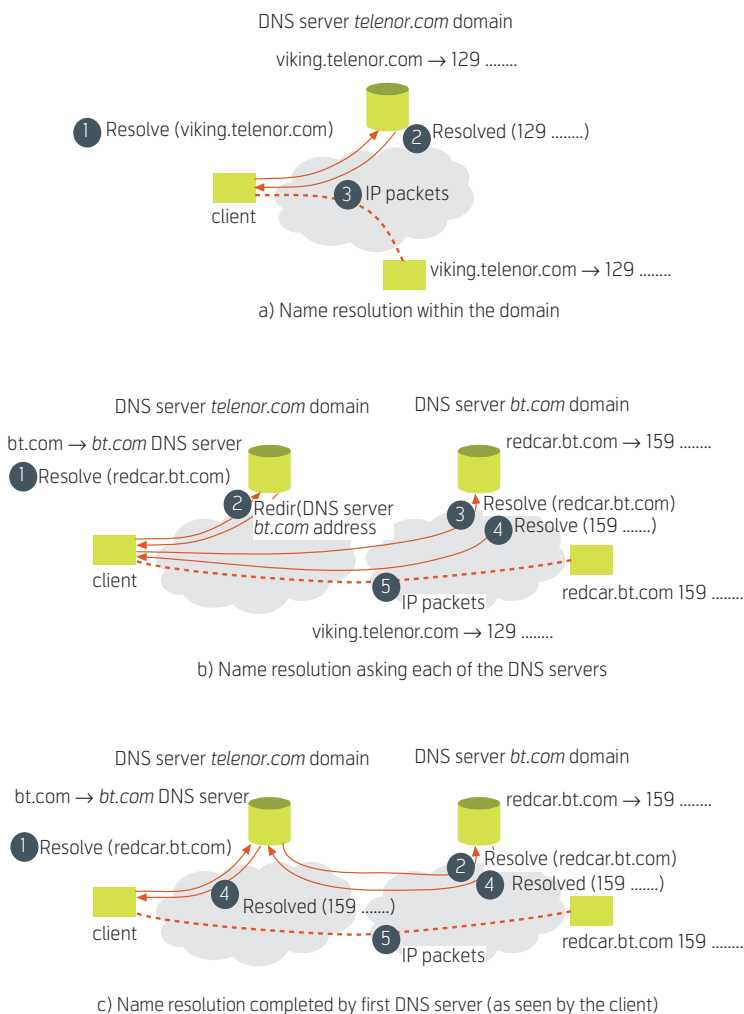


Figure 6 Examples of sequences for name resolution. Note that messages and addresses are fictitious

address and appends this result to the request before returning the reply message. If the server is not able to resolve the name, it forwards the request to the next name server (if a complete resolution was indicated in the request) and returns the result to the requester, or replies its lack of information to the requester (if no complete resolution was indicated). To initiate this procedure, all machines must know the address of at least one domain name server.

In order to have more efficient name resolutions (as well as to increase the availability of the resolution function), local name servers are commonly used. Such a local server may keep a list of all names recently being resolved (recently refers to a time less than a time-to-live which could be set per name). Asking the local server first would frequently make the resolution function more efficiently as it turns out that more requests are initiated for the same domain name. Resolving a domain name is also commonly referred to as name binding.

As seen by the example, the names/addresses are hierarchical. The last part of the name (after the last period) tells which 'area' out of the first/top-level separation. For the IP-address the first part (before the first period) has a similar meaning. However, more 'fields' are commonly needed to identify precisely the 'area'. An organisation is assigned the responsibility to administer the name range at a certain level. For example, Telenor would by itself assign names within the *telenor.com* range. Examples of some top-level domain names are *.com*, *.edu*, *.gov*, and *.org*. In addition, most countries have their own domain name. The domain names may not contain information about the physical location of a host/machine. This is one reason for the need to translate the name into an IP-address.

Inverse translation from IP address to name could also be asked for. In particular, this is often used for names written in so-called dotted-decimal form; eg. *abc.def.ghi.klm*, where all these are digits in the range [0 .. 9].

## 4.2 Route

As mentioned above, distinctions are made between names, addresses and routes; A name indicates what one seeks. An address indicates where it is. A route indicates how to get there. The IP packets deal primarily with the addresses, while higher-level protocols may take care of the mapping from names to addresses. When a hop-by-hop resolution is used, the mapping from address to route is carried out in each of the nodes on the way. This is the common approach for any IP-based network as the IP packet header carries the addresses.

The term routing refers to the process of selecting a path along which packets are sent. Conceptually, one may think of the routing table in a node as consisting of pairs; the set of addresses, and the outgoing path to use (as seen for that node). This is schematically illustrated in Figure 7 for an IP router. Observe that the complete destination address may not be specified in the routing table.

Basic properties requested from a routing algorithm are: correctness, simplicity, robustness, stability, fairness and optimality. Two major classes can be identified for routing algorithms:

- Non-adaptive algorithms that do not use measurements or estimates of current traffic load or topology in the routing decisions. These are also called static routing algorithms.
- Adaptive algorithms that change their routing decisions depending on changes in topology, perhaps also the traffic load. These may further differ in the way the information is distributed, how frequently the routing is changed and which metrics are used.

Routing information exchanged within a domain and across a domain border typically differs. One argument for this is to limit the amount of information that is revealed regarding topology and capacity within a domain. This could be for scalability and security reasons.

For example for an IP network, a set of routers can be grouped into an *autonomous system* (AS). An AS is handled by a single administrative authority, eg. a network operator. A conceptual view on two ASes using Exterior Gateway Protocols (EGPs) between them and Interior Gateway Protocols (IGPs) internally is given in Figure 8. As recognised, a gateway/border router may use two different protocols, one within the AS and another outside the AS.

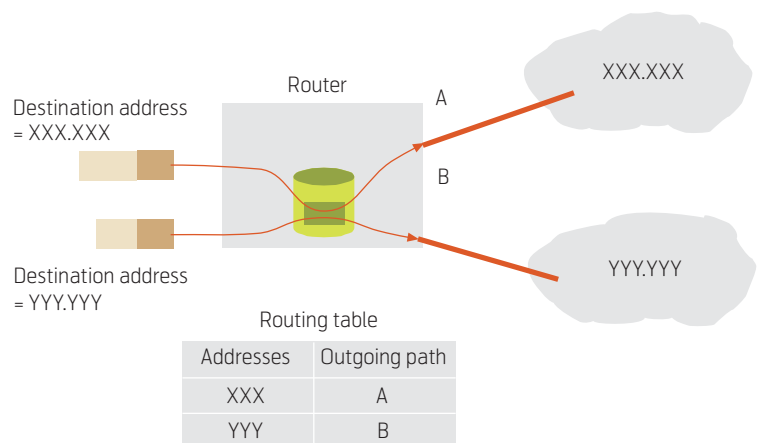


Figure 7 Illustration of routing information in a router

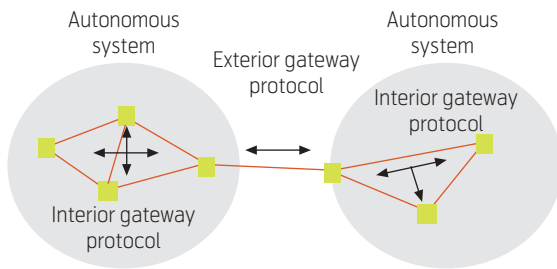


Figure 8 Use of exterior and interior gateway protocols

One of the first IGP is the Routing Information Protocol (RIP), originally designed to provide consistent routing and reachability information among hosts in a local network at the University of California at Berkeley. Using RIP, routing data for each router is broadcasted to all its neighbours periodically. Each destination in the routing table is included in the route updates. For a larger network, slow convergence may well occur, for instance when a network portion suddenly becomes unavailable (slow count to infinity). This could be alleviated by principles called split horizon and hold down. Other IGPs are OSPF and IS-IS.

Two routers that belong to different ASes are said to be exterior neighbours. The protocol exterior neighbours use to advertise reachability information to other ASes is called the Exterior Gateway Protocol (EGP). An EGP has three main features: i) support a mechanism allowing two routers to agree to communicate reachability information (acquisition), ii) a router tests whether its EGP neighbour is responding, and iii) EGP neighbours exchange reachability information. The reachability information is commonly called routing information as it is used as a basis for deciding upon routing of packets. In order to fulfil its three features a number of message types are devised, like 'acquisition', 'cease', 'hello', 'poll' and 'routing update'.

When two routers agree to exchange reachability information (acquisition), they also set initial values for a time interval to be used for testing whether the neighbour is alive (called a hello interval) and a polling interval that controls the maximum frequency of routing updates. These intervals can be changed. Moreover, they may be asymmetric, ie. different values in the two directions. Considering features ii) and iii) above, one recognises the reachability exchange has been separated from the routing information exchange. A motivation for this is that the reachability could change more frequently without influencing the routing.

Between ASes the Border Gateway Protocol (BGP) is commonly used. As seen from a BGP router, the network is made up of other interconnected BGP routers.

Two routers are connected if they share a common network. BGP can be said to be a distance vector protocol. In addition to the cost to each destination, each router also keeps information on the exact path to be used. Information on these exact paths is then exchanged.

*Efficient and flexible support for identities, names and routing* are crucial requirements in an architecture. For example, to enable swift up-scaling by introducing multiple instances of certain functionality, a location and routing function must be present. Such concerns should be included in the list of initial requirements that the architecture must comply with.

It is also important that the architecture captures differences between domain-internal and -external capabilities. Complementing an architecture at a later phase with concerns arising from business aspects would typically result in an inferior solution. One aspect of this is that mapping between the internal and the external view must be present at the border. To do this mapping all relevant information must be present.

## 5 Protocol Principles

Protocols are used to communicate between functional entities. This is a traditional concept applied for decades. Due to the layering and information hiding principles, an entity on a certain layer does not need to understand the data it is to carry. In fact allowing it to be agnostic with regard to this data reduces complexity and dependency.

As seen from a given layer, communication between two entities can carry unstructured or structured information. Examples of the former are voice and video streams. Semantic and syntax have to be defined for structured information. These rules govern the organization and the content of the communication. Note that the structured information refers to a given layer and only parts of a message may contain structured information (eg. structure of the payload may not be defined). Hence, unstructured information is commonly placed into a 'container' when carried across to the other entities. Appropriate information (eg. routing addresses, integrity checks) is then attached to the container to ensure that the information reaches the proper receivers.

According to the Open System Interconnection (OSI) model there are seven layers in the communication protocol stack. This provides mostly a theoretical reference, however, as most protocol stacks are implemented with fewer layers. In particular the layers 5-7 are commonly implemented in a single entity.

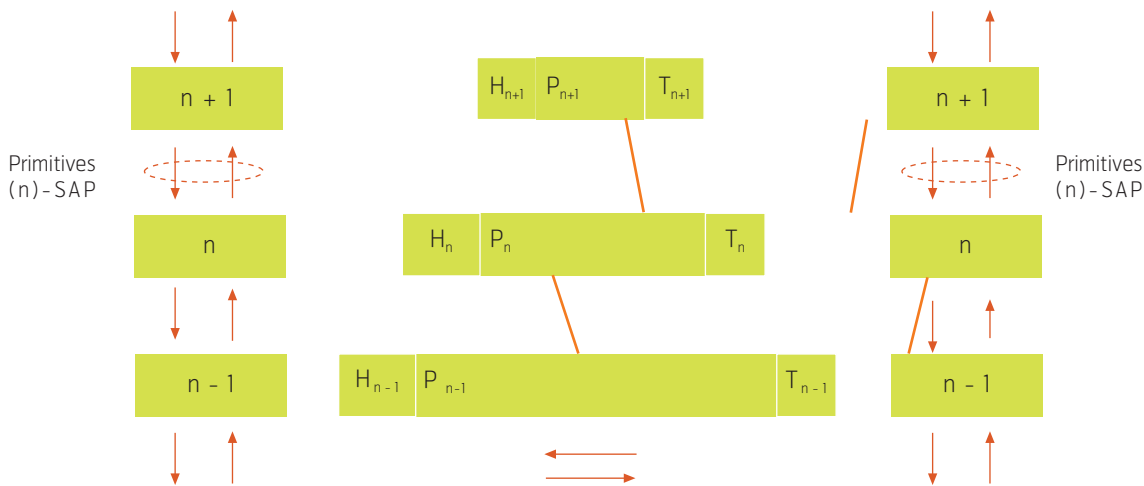


Figure 9 Protocol layering and protocol data units

Let us look at layer  $n$  in the protocol stack, see Figure 9. This layer provides services to the layer above,  $n + 1$ , and makes use of services from the layer below,  $n - 1$ . In the OSI model, these services are provided through Service Access Points (SAP). In the TCP/IP protocol stack SAPs are identified as ports (see Figure 5). There may be several SAPs between any two layers.

Then, a set of protocols can exist between layer  $n + 1$  entities in two communicating nodes. The actual communication passes through layer  $n$  (which then again may make use of services from layer  $n - 1$ , etc.). To carry out this, the protocol message types have to be mapped into primitives between the layers. Four types of primitives are defined as part of the OSI model:

- **Request.** Layer  $n + 1$  requests layer  $n$  to perform a specified task such as transferring a message to the peer entity. It may also be a local action, which is within the node itself.
- **Confirm.** Layer  $n$  acknowledges the request. Note that the confirmation may be received from the peer layer  $n + 1$  entity or from the local layer  $n$  entity.
- **Indication.** Layer  $n$  delivers a PDU to the remote layer  $n + 1$  entity. The indication primitive may be generated by either the sending entity of the layer  $n + 1$  entity or the local layer  $n$  entity.
- **Response.** Layer  $n + 1$  entity acknowledges the receipt of an indication primitive.

The combinations of primitives are depicted in Figure 10.

As seen from an implementation point of view, the primitives can be realised as a procedure/routine call

of format NAME.type, where NAME indicates the type of service and type indicates the primitive type (request, confirm, indication, response). Examples NAME can be ESTABLISH (for setting up a connection), DATA (for transferring data) and RELEASE (for releasing a connection). A number of parameters have to accompany the primitive calls, such as the identity of the peer entity at the same layer.

A protocol between two nodes (at the same layer) consists of a set of protocol data units (PDUs). A PDU is put together by a set of header elements, a set of payload elements and a set of tail elements. The payload of a PDU at one layer is the PDU at the layer above (also referred to as providing a container earlier). Following a flow down a protocol stack, this procedure is iterative. At the receiver node, the headers/tails are then stripped off at the corresponding protocol layers.

Understanding protocols is fundamental when the physical aspects of an architecture is designed. The choice of protocols and their characteristics has to be matched with other aspects so that the proper information is present. For some protocols the proper state models have to be implemented in the components that are communicating.

## 6 Interconnecting and Opening for Greater Community

### 6.1 General

Interworking between providers' domains is a major area in itself. Such interworking could be divided into syntactical and semantical aspects. The former refers to adaptation of protocol or information formats, while the latter also includes adaptation of the information transported between the domains. One example of syntactical interworking is trans-coding of voice, say between the G.711  $\mu$ -law and the G.729

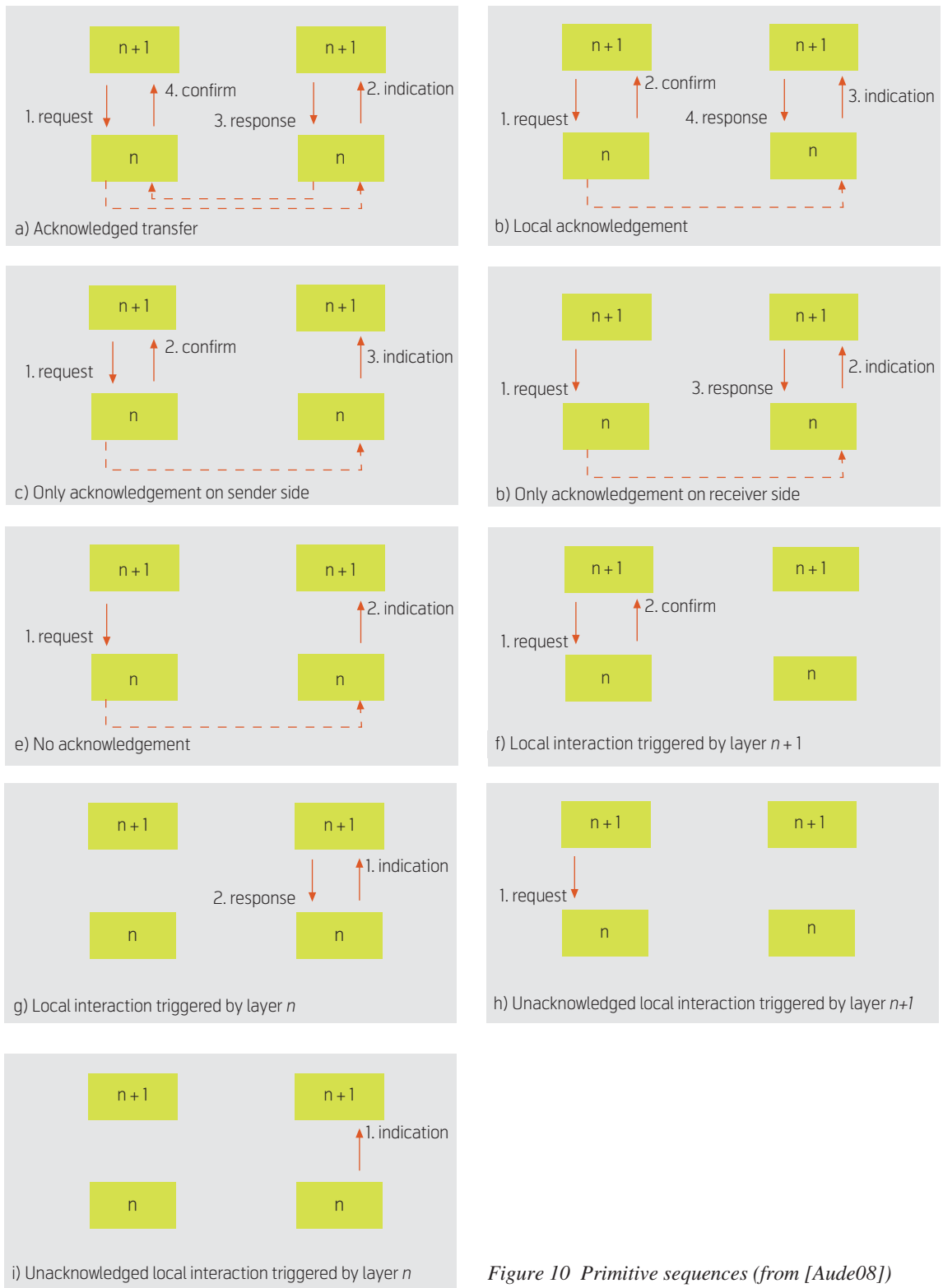


Figure 10 Primitive sequences (from [Aude08])

codec schemes. An example of semantical interworking is found when a SIP-based and an SS7-based voice network are to be connected, implying mapping between sets of SS7 messages and SIP messages. These interworking units have to be integrated into the architecture.

There is also a difference between interconnecting two technical domains managed by the same provider and interconnecting domains managed by different

providers. In the latter case, Service Level Agreements would likely be in place.

Interworking across administrative domains brings several questions that have to be addressed such as:

- What information is to be exchanged?
- Which failure reports should be forwarded?
- How to measure the service levels (if relevant)?
- On what basis is accounting to be done (if relevant)?

- Which rules exist for announcing, negotiating, ordering, configuring and releasing service features?
- What trust level should be assigned?
- Which security features need to be activated?

## 6.2 Service Levels

The term Quality of Service (QoS) has commonly been applied to refer to the service level. However, it does not capture the users' experience. Hence, Quality of Experience (QoE) has been defined as (ref. ITU-T Rec. P.10/G100):

*The overall acceptability of an application of service, as perceived subjectively by the end-user.* In order to estimate the experience aspects, measures such as the Mean Opinion Score (MOS) have been defined. This is a rate on the scale from 1 to 5.

The QoS term is often used in a confusing manner. The ITU-T Rec. E.800 defines QoS as *The collective effect of performance which determines the degree of satisfaction of a user of the service.* However, there are several tens of definitions of the term QoS only in standardisation bodies like ITU-T and ETSI,

ITU-T Rec. G.1080 provides the relation between QoE and QoS as depicted in Figure 11.

## 6.3 Growing the Pie by Interconnecting

As technical solutions evolve, the question of compatibility arises. In particular this refers to backward compatibility of equipment introduced now having to work together with existing systems. In some cases backward compatibility is not on the agenda, say when there is no reason for interworking or needing to access anyone outside a closed community.

However, driven by the network externality argument, new systems could get a flying start by being backward compatible. This may also drive a higher number of units and thereby lower the unit costs by economy of scale.

In general, requiring backward compatibility has a tendency of increasing the cost, slowing down the deployment and restricting the innovation level of the new solution.

Compatibility could also be required from authority, eg. through license conditions. This is for example found in the requirement on number portability.

Compatibility could be solved in different manners; such as by an interworking unit in the network or by a multi-solution capability in the terminal/end-system. An example of the latter is a multi-mode handset supporting GSM, UMTS and WLAN.

## 6.4 The Other Side of the Modularisation Picture

Defining a number of modules is motivated by allowing flexibility of choosing which modules to incorporate for an instantiation. The intention is also for certain module(s) to be replaced without too much impact on other modules.

There are, however, drawbacks of this. One is the overhead of service discovery and potential negotiation. That is, each of the relevant modules must become aware of which modules that deliver which services. This can belong to the 'transaction costs' as discussed by Ronald Coase in 1937 (eg. see [Mcke08]). This term was then related to a firm's interactions on a marketplace. In view of that, only looking at a single firm, a market would exist if the transaction costs plus the service costs are lower than the cost of supplying the equivalent product or service in-house. This should not be directly transferred to a system portfolio, however. Firstly, there may be a common authority for the portfolio so each of the components would not be allowed to make such autonomous optimisations. Secondly, each of the components would evolve over time which implies a development cost. Commonly there are scale/scope gains to be achieved by placing such efforts on solutions that can be re-used across the portfolio.

Still, the transaction cost analogy could provide some insight, such as its three elements:

- *Search cost.* Finding the service or product requested to support a particular function takes time and effort. Across domains one would also examine level of trust and performance. In case service descriptions are not standardised it may not be trivial to compare potential components.
- *Contracting cost.* In case of interactions across domains, price and other terms would be negotiated.
- *Co-ordination cost.* As the number of independent components increases, the complexity of managing the interactions and dependencies may also increase, such as simultaneous usage.

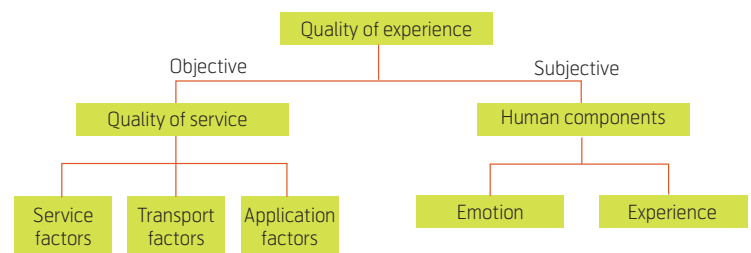


Figure 11 ITU-T Rec. G.1080 description of Quality of Experience

For a full-blown support of such a dynamic environment (see Chapters 7 and 8), these elements must be provided by relevant functions in the architecture.

## 6.5 Balancing Network Intelligence and Transparency

Adding intelligence in a system has a tendency of increasing the cost and complexity. Hence, there should be some request for each functional component. For a telecom network there has been a discussion on how much intelligence that should be implemented in the network itself versus how intelligent the end-systems/terminals should be. Different solutions are seen, for example comparing a PSTN with an IP network.

In particular for an IP network, this debate is related to the topic of network neutrality. Basically, it boils down to whether anybody should have the right to differentiate traffic types in power of their position (such as managing the access networks). According to [Aude08] the following four freedoms are included in the network neutrality:

- Freedom to access content on the network (ie. access to the information can only be regulated by the owner of the information and not by an ISP or another party not operating on behalf of the owner of the information;
- Freedom to run applications of any kind alone or together with other users;
- Freedom to attach any hardware to any network (eg. routers, servers, PCs) that satisfies the Internet specifications;
- Freedom to obtain information about all services and electronic goods available on the network.

Computing paradigms	Services	Features
• Cloud computing	• Software as a service (SaaS)	• Ubiquitous access
• Edge computing	• Infrastructure as a Service (IaaS)	• Reliability
• Grid computing	• Platform as a Service (PaaS)	• Scalability
• Utility computing	• Service-Oriented Architecture (SOA)	• Exchangeability/ Location independence • Cost-effectiveness

Table 1 Relations between computing concepts, services and features (adapted from [ITUDistr09])

The case for network neutrality is argued for by the increased level of competition (on services/applications/content area), thereby motivating for innovation and easy deployment of special services.

On the other hand, the potential lack of incentives by network operators to invest in network capacity and special features (such as bandwidth guarantees) is an argument against network neutrality. Although basically providing technical and economic arguments, it is a battle for business positions and political solutions.

The physical architecture has to take a stand on where to locate the functionality. There are other perspectives of the architecture (ref. Section 1.1) that do not consider the actual location of different components.

## 6.6 Virtual Operators/Providers, Over-the-Top Players

It is common to make a distinction between the operators providing the physical infrastructure and operators/providers without the user-near infrastructure. Examples of physical infrastructure are access links, edge routers, base stations, etc. However, the border between virtual and ‘non-virtual’ providers may be blurred in some cases. For example, some operators have their own HLR and MSC, while still being referred to as virtual operator on the condition that they do not have any radio access network.

Similarly, in case of roaming users, the home provider could be referred to as a virtual operator, as it does not have any physical infrastructure within the vicinity of the actual position of the user considered.

For an IP-based network, some refer to those providing services without having their own infrastructure as ‘over-the-top players’.

To allow for different domains and involving different players, the architecture has to include proper functionality and information. One example is that identifiers and names have to be consistently recognized. There may also be commercial concerns, such as security, service levels and accounting regimes that require the presence of corresponding functionality.

## 6.7 Resource Virtualisation

There are several understandings of the virtualization trends. Some are also attached to several of the concepts being promoted, see Table 1. It broadly belongs to the distributed systems’ paradigm<sup>1)</sup>. A distributed system is commonly referred to as a collection of independent computers that appear as a single coherent system.

<sup>1)</sup> Leslie Lamport: “You know you have a distributed system when the crash of a computer you have never heard of stops you from getting any work done”.

One objective of a distributed system is to connect users and IT resources in a transparent, open, cost-effective, reliable and scalable manner. Resources can be divided into (adapted from [ITUdistr09]):

- Physical resources:
  - Computational power
  - Storage devices
  - Communication capacity
- Virtual resources:
  - Operating systems
  - Software and licenses
  - Tasks and applications
  - Services
  - Virtual memory

To implement the virtualisation concepts (similarly to virtual providers) requires functionality as elaborated in the subsequent sections.

## 7 Supporting Dynamics and Loose Coupling

### 7.1 Components Playing Together

One objective of service orchestration is to take a set of independent service components and combine them to create an integrated service offering. Accord-

ing to the orchestration principle, a central unit coordinates and controls all service interactions required to perform a particular task. Hence, such a unit must find its place in the technical architecture.

In theory the service components do not know that they are part of a more complex service chain. The Business Process Execution Language (BPEL) is a popular way of describing the orchestration logic (see text box).

As opposed to orchestration, choreography does not have any central unit. Each service component will then need to know which operations to execute and with whom to interact. So far, choreography seems to have gained less traction from the industry. This is likely to follow from the fact that components need to be aware of each other (potentially leading to a mesh interaction pattern).

Related to network services several components could be put together in order to deliver a coherent service to a user. There are different places where such orchestration could take place, see eg. Figure 12. The one most commonly applied so far is the client orchestration where the service components are put together in the client/terminal. Common examples are Microsoft Office applications and various browser engines.

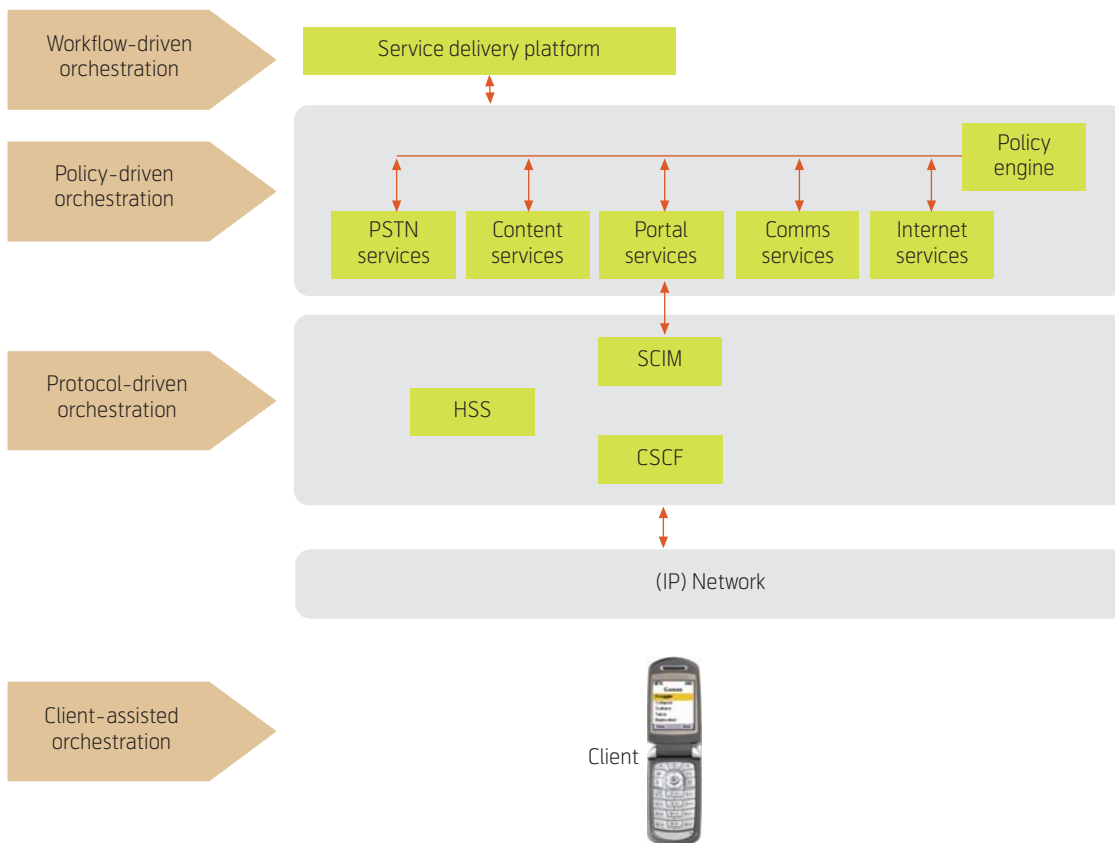


Figure 12 Service orchestration techniques (adapted from [Cuev08])

## Business Process Execution Language (BPEL)

BPEL can be used as part of Business Process Modelling (BPM). BPM has its roots in work-flow management and has also been applied in the area of orchestration. In this context, orchestration must be described by two elements; i) the way in which the sequence of steps operates, including all forms of conditionality and exception handling, and ii) the way in which the steps are enacted by the web services. BPEL supports these two steps.

The inherent capabilities are provided by i) using standard web services to invoke the partner services, ii) exposing the resulting business process as a web services, and iii) creating a language that is portable onto any platform supporting the specification.

BPEL is based on XML – in common with other web service specifications – and it invokes web services that enact the process steps using the protocol specification of SOAP (or REST) and Web Services Description Language.

BPEL describes a process via a series of constructs encapsulated via a <process> wrapper. This allows various elements of the process to be defined, internal to the process, and also allows the process itself to present a single set of interfaces to the outside world. The <process> wrapper contains the following elements (from [Musc08]):

- <partner> defines the entities that provide and use the services required to enact the process. These are bound to the process with <partnerLinks> and execute the services using <invoke> statements.
- <variables> defines the instance data used by the process.
- <correlationSets> defines the process instance identification.
- Activities defines the processing steps that constitute the process. Activities are defined in BPEL with:
  - <sequence> has one or more activities that are executed sequentially in the order they are placed within the <sequence> elements.
  - <flow> makes it possible to execute several activities in parallel.
  - A series of typical programming constructs such as <while>, <switch>, <case>, <pick>.

[Cuev08] elaborates on three network-based orchestrations:

- *Protocol-level orchestration.* This is based on the principle of spoofing signalling traffic and of modifying the behaviour in order to avoid interactions between features in different applications servers. In general this approach requires good knowledge of the features of each service and a functional element that keeps track of the state of those features. In the IP Multimedia Subsystem (IMS), the Service Capability Interaction Manager (SCIM) may take on this role.
- *Policy-driven orchestration.* This is based on the principle of applying policy-based decisions independently of applications. It is primarily driven by events and relies on control interfaces to invoke appropriate actions in response to those events. These interfaces can either be on a protocol level or on a web services level. It can work as either i) delegating policy execution to the application, or ii) leave decisions to an external policy engine.

Typical solutions are application servers and policy engines.

- *Workflow-driven orchestration.* Following the principles of Service Oriented Architecture (SOA), it is based on executing a deterministic process that is triggered by a workflow. Hence, it is based on a sequence of steps to co-ordinate a set of independent atomic elements with a defined output. The workflow itself does not try to solve interactions, it assumes well-behaved services. Today, this is typically implemented by Business Process Execution Language (BPEL).

However, according to [Cuev08] there are several technical challenges to swiftly design an orchestration arrangement:

- Most existing services have not been designed to be ‘orchestratable’. Many services can only be invoked via their network protocol interface, and most have been designed as vertical silos.
- There are no industry-wide design rules that dictate how services should be designed so that they can be easily combined to support richer, more complex service offerings.
- An individual service instance may include many different features, but each individual feature is rarely accessible as a standalone feature. From a categorisation point of view, there is no clear distinction between services, service features and service enablers, making it difficult to address orchestration at a service features or service enabler level, rather than at the level of whole services.
- The need to support service orchestration can also impact how user/subscriber data is defined, provisioned, managed and extended within the network, and how service data and session data is organised and shared. Tight co-ordination of multiple service instances raises the need to share user/subscriber data, service data and session data across those service instances.
- Non-functional aspects, including user interface aspects, provisioning, billing and management must also be considered, so that the integrated service can be treated as a first-class service in its own right.

Making components play in an automatic manner is also addressed by the so-called self-managed network. Self-managed network components have been promoted to reduce the OpEx. This includes self-optimization, self-configuration, self-protection and self-

repair. Another term is automatic networks and systems that also cover all these self-properties. To implement these features, corresponding components must be defined in the technical architecture.

## 7.2 The Enabler Concept

The concept of enabler provides an abstraction of a set of resources to a set of applications. The applications are then executed in an execution environment. The execution environment (EE), or platform as some refer to it, provides functions like process monitoring, life-cycle management, application execution, etc.

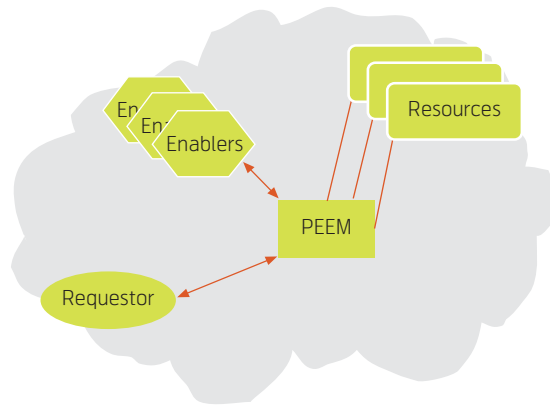


Figure 13 Illustration of actors in the context of OMA PEEM (from [OMA-PEEM])

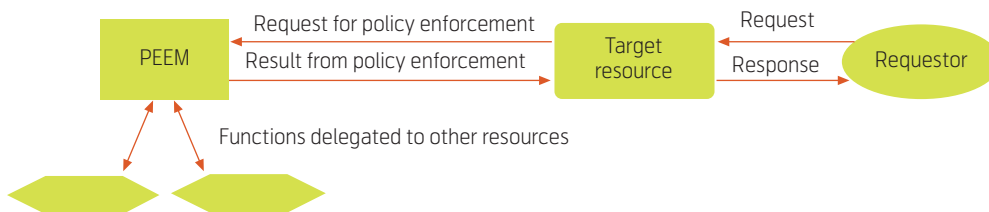
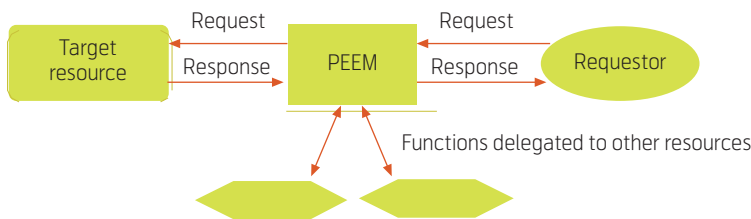


Figure 14 Logical illustration of usage patterns for PEEM; upper: PEEM as proxy, lower: callable PEEM (from [OMA-PEEM])

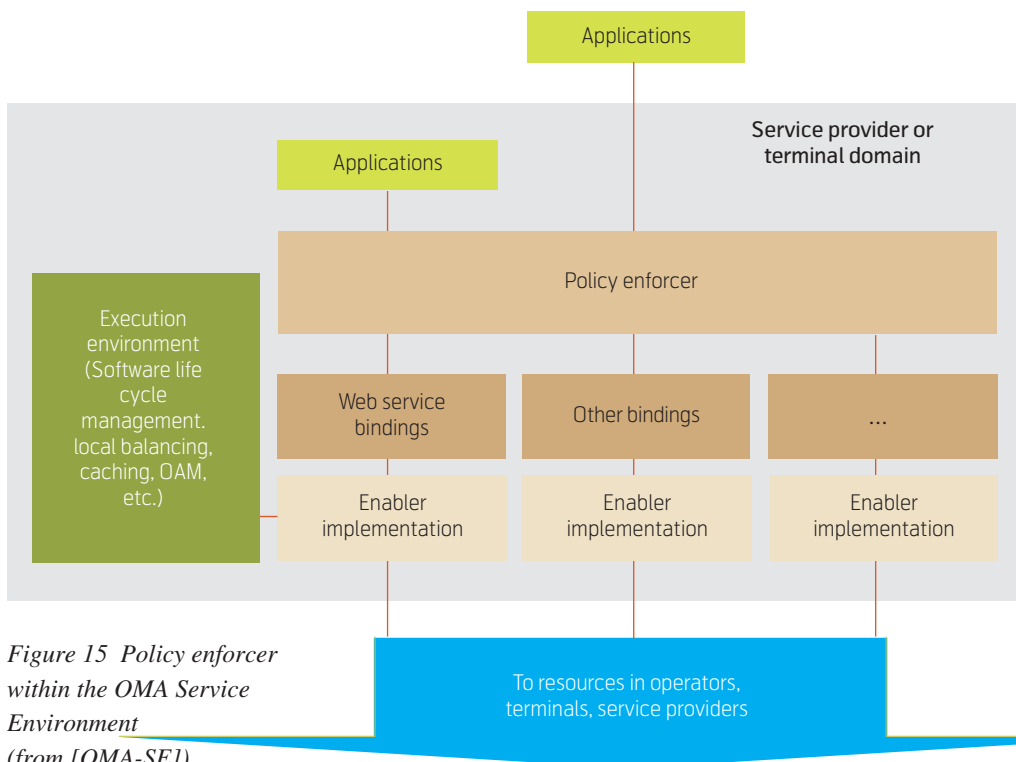


Figure 15 Policy enforcer within the OMA Service Environment (from [OMA-SE])

One example is the PEEM as described by OMA [OMA-PEEM]. This concept is illustrated in Figure 13. Two usage modes of PEEM are envisaged; either as proxy or as callable, ref. Figure 14.

As described in [OMA-SE] a service enabler is a technology intended for use in the development, deployment or operation of services. This is defined by a specification, or set of specifications. It is also published as a package. A service enabler could be implemented in different domains, including the terminal domain and the provider's domain.

An enabler may amalgamate, abstract and/or repack-age a resource<sup>2)</sup>. It should present its functions through an interface after binding to a particular syntax. Mechanisms from Web Services could be utilized for this.

Examples of enablers from the Open Mobile Alliance include location, device management, authentication, access control, discovery and directories.

The OMA also defines a policy enforcer, see Figure 15. This component provides policy-based management mechanisms to protect resources from unauthorized requests. It also manages the use of these requests through charging, logging, user privacy or preferences, or other mechanisms.

Note that 'applications' as depicted in Figure 15 also includes other enablers. That is, different enablers could be stacked (building onto each other).

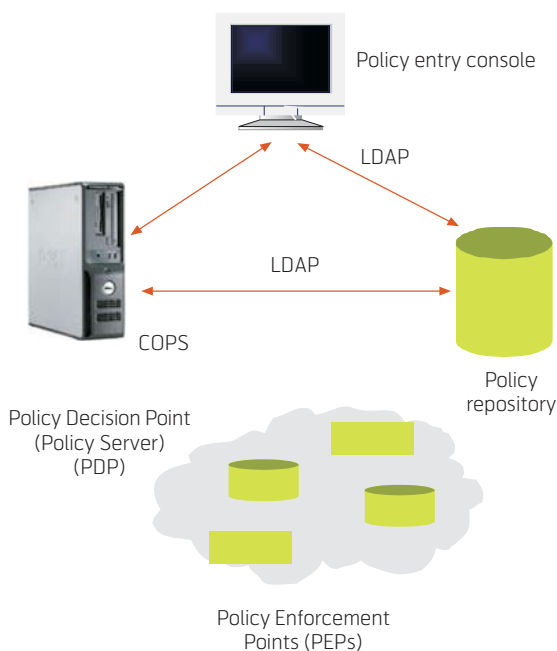


Figure 16 Example of Policy Framework components and protocols

## 8 Policies to Support Dynamics

Policy can be considered as a set of principles for usage of resources, handling of traffic or other aspects, given by business considerations. That is, the business decisions are translated into statements relevant for how the traffic, requests and resources are handled.

The semantics of a *policy rule* is a conditional imperative statement in the form

```
if <condition> then <action>
```

Thus, applying a rule means to evaluate its condition (matching the rule), and, depending on the outcome of that, either execute the action or not.

Policy-based network management would provide a centralised platform for managers. With this, one can define and distribute network policies to enforcement points at selected places. In a typical policy-based framework, see Figure 16, the manager edits policies through a policy entry console. Those policies are then stored in a policy repository. When requested, a policy server (Policy Decision Point) retrieves policies from the repository and makes policy decisions that are communicated by applying relevant protocols, eg. Common Open Policy Service (COPS). Examples of Policy Enforcement Points are routers, switches and firewalls.

In order to efficiently manage the resources a number of features of a management system are asked for:

- Centralised management view; implying the ability to carry out management activities remotely and that the management system is able to cope with all resources.
- Abstracted management data; saying that simplified views of resources should be possible, eg. 'hiding' details when they are not relevant.
- Common and consistent views/interfaces; similar procedures and similar data views should be used for similar procedures. The number of views/interfaces needed should also be limited.
- Automation of tasks; less human intervention is needed and customers may serve themselves within the authorised set of activities.

A *policy domain* (see Figure 17) can be viewed as a contiguous set of nodes that operate under a common system of administration and provide a common set of services. Each of the nodes can contain policy

<sup>2)</sup> In [OMA-SE] a resource represents a capability in a provider's domain. This may for example be a network element.

rules and/or policy information. Such a grouping is done to simplify management and ensure consistencies. A policy domain can also be seen as a container that provides scoping for a policy container, policy rules and other policy information.

A policy group class holds a property called policy roles. This represents the roles and combinations of roles that are associated with a set of policy rules. Each value represents one role combination. After identifying the relevant set of rules to be used, the rules have to be prioritised (eg. first-match, match all, priority order).

The policy to apply to an entity (eg. router) may depend on many factors such as the characteristics of the entity (eg. protocol type used) and the user connected. To describe the functions attached to an entity, the term *role* is introduced: A role represents a functional characteristic or capability of an entity (resource) to which policies are applied. Multiple roles may be assigned to a single entity, resulting in that entity's role combination. A role should be thought of in a wider sense than an entity's attribute as the role would impact on the policy selected for an entity.

A policy rule can contain ordered lists of conditions and actions. The conditions and actions may be reusable objects that reside in repositories, or they may be rule-specific embedded in the rule or a combination of both. An advantage of having reusable objects is that many policy rules may refer to the same object.

One way of thinking of a policy-controlled network is to consider the network as a state machine where policies are used to control which state each of the entities should be in or is allowed to be in at any given time.

Policy rules may be aggregated into policy groups, which may be nested to represent a hierarchy of policies. Policy groups can model intricate interactions between objects that may have involved interdependencies, like application type, user identity, interface, time of day, etc. A policy group can be reused and managed as a unit. A policy rule can be called a stand-alone policy. These can be expressed as a simple statement, eg. represented by a Management Information Base (MIB).

The set of conditions in a rule specifies when the policy rule is to be applied. The conditions can be given as sets of individual condition statements related by AND/OR. Negations can also be used. When the set of conditions associated with a policy rule is evalu-

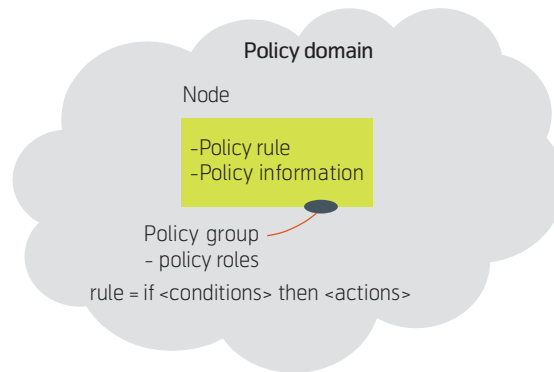


Figure 17 Policy domain and related terms

ated to TRUE, the set of actions in the rule is executed. This may either maintain the current state or imply a transition to another state. The order of execution of the actions can be specified.

Policy rules themselves can be prioritised, eg. to have an overall policy with some variations in case of exceptions.

The policies can be said to represent business goals and objectives. These goals have to be related to implementations in the network.

Supporting dynamics by introducing policies implies that the functionality mentioned needs to be included in the architecture. The different perspectives of the architecture must describe how the policy management is to be provided, including use of protocols, how to provide relevant data, and so forth.

## 9 Management Aspects

### 9.1 TM Forum, eTOM

Many people point to the TM Forum when discussion management of telecom systems, services and networks. Here we find the enhanced Telecommunication Operation Map (eTOM), for example, which defines the business processes, see Figure 18. It is intended to provide a reference into which the different processes can be related.

### 9.2 TM Forum, SID

The eTOM does not state anything about how to address the information model that is needed for the systems to interact. For that area, the *Shared Information/Data model* (SID) has been elaborated. The SID model describes an implementation-independent model of the business concepts, their characters and relationships. The SID model is a *layered architecture*; in the top level there are eight domains that can be aligned with the eTOM model.

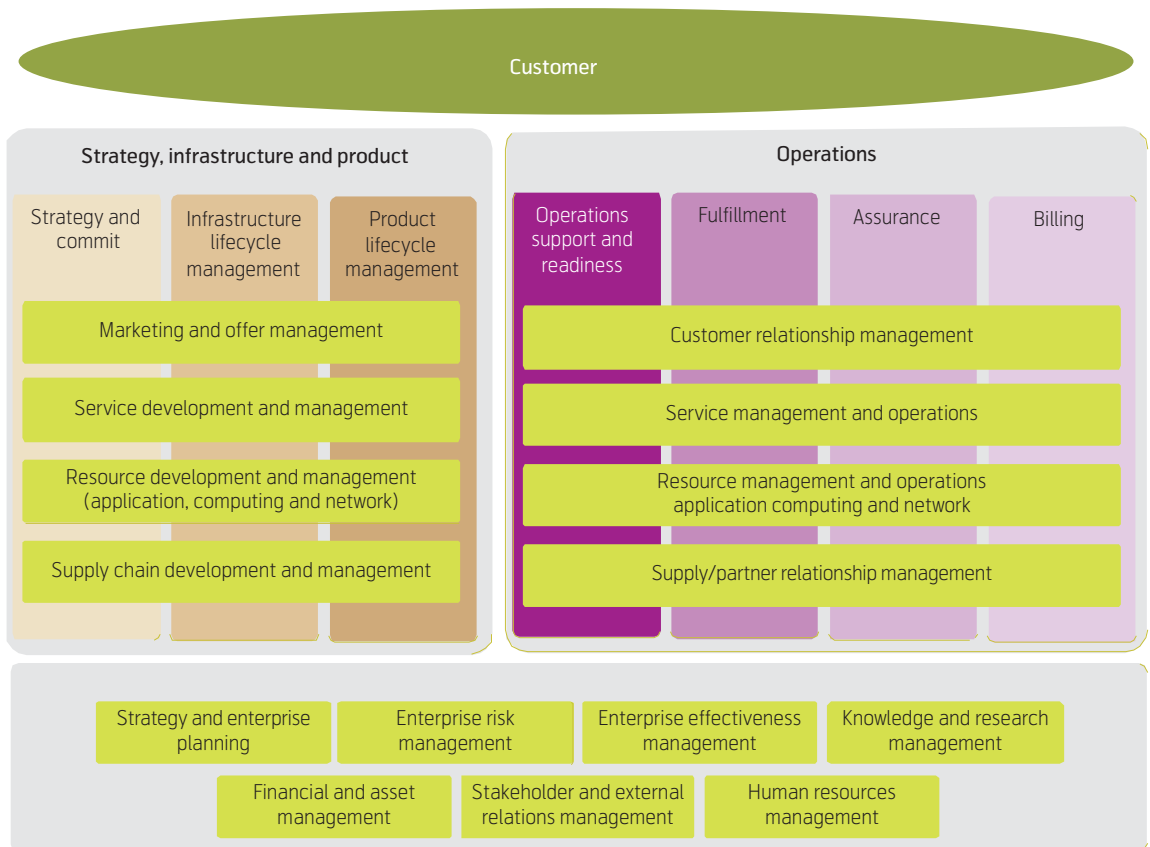


Figure 18 Enhanced Telecom Operations Map

Within the SID domains multiple Aggregate Business Entities (ABEs) are defined. An ABE contains a number of business entities. Then, the business entities contain more fine-grained business entities and associated attributes. The business entities represent things that should be modelled, see Figure 19. The ABE *Customer* for example contains a set of business entities, one of them being *CustomerAccount*. The *CustomerAccount* has associations with other business entities and includes attributes line *name* and *type* of the account.

### 9.3 TM Forum, TNA

The TM Forum is also working on a *Technology Neutral Architecture (TNA)* as part of the New Generation Operations Systems and Software (NGOSS). TNA is specified by high-level concepts allowing any technology-related issue to be resolved during implementation. The Distributed Interface-Oriented Architecture (DIOA) approach is used for specifying the TNA. This can be considered as a more general architecture than the Service-Oriented Architecture (SOA). That is, all SOAs are by default DIOAs. A DIOA is defined through: "A provider entity offering functionality across its interface that involves coordinated behaviour from both the consumer and the provider entities. The set of consumer and provider

behaviour, invoked using an interface, is an architectural construct that can be distributed."

The TNA defines a *component-based, distributed system architecture*. A set of associated mandatory services that this architecture requires is also included. Three of the key concepts are the Common Communications Vehicle (eg. a messaging bus), an Interface (representing the functionalities of a component), and a Contract (describing the services provided by the associated Interface), see illustration in Figure 20.

The following principles should be examined when assessing the compliance of an implementation with respect to TNA:

- Architecture is inherently distributed
- Architecture uses interfaces to communicate
- Architecture is componentized
- Business processes are separated from implementation of components
- Architecture is security-enabled
- Architecture is policy-enabled
- Architecture uses shared information and data
- Architecture uses a common repository

Implementing the principles of TNA comes from the reasoning that when the number of systems grows,

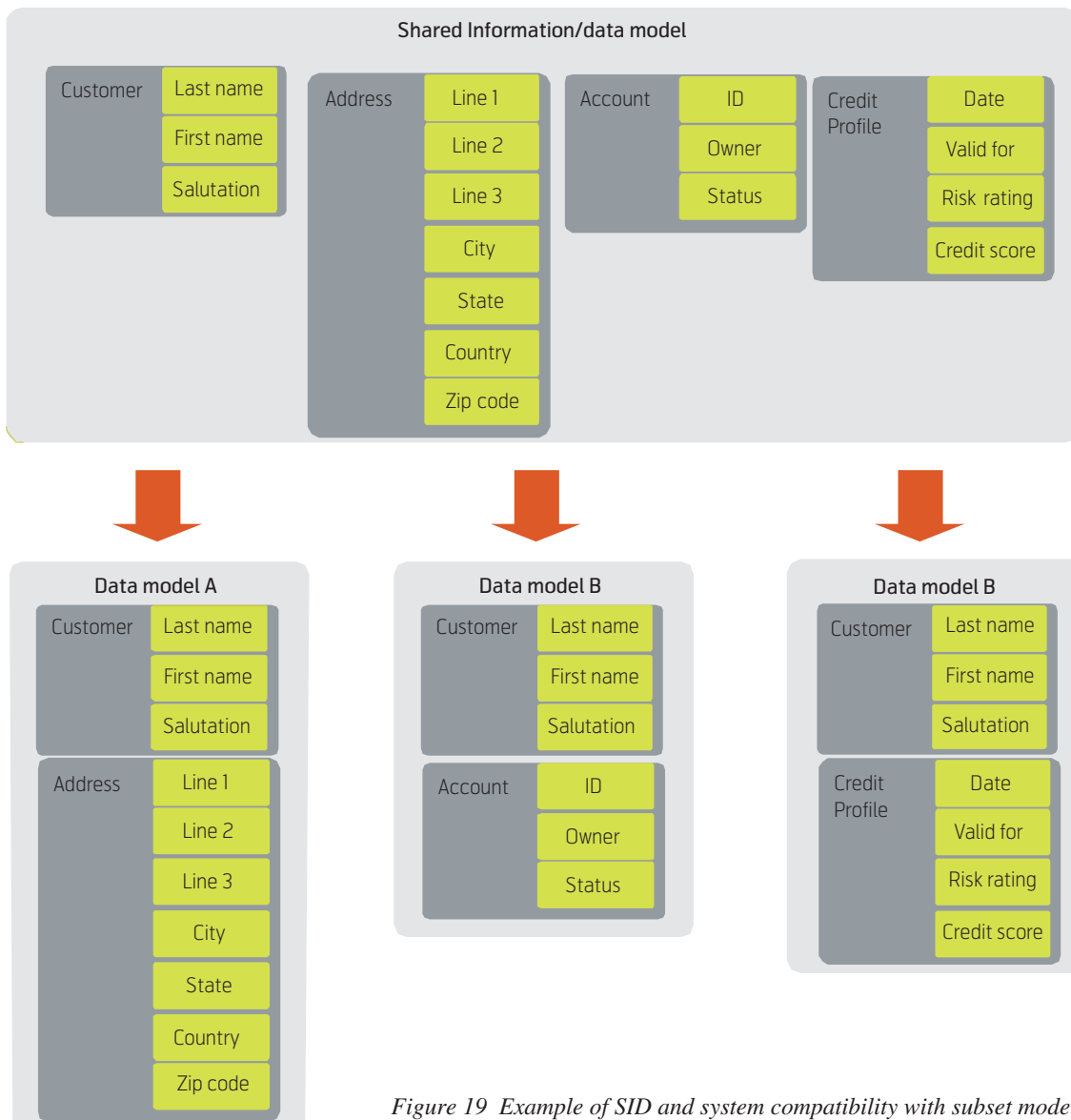


Figure 19 Example of SID and system compatibility with subset models

managing direct process flows between the systems becomes more complicated. Typically the system portfolio and tasks to resolve become easier to manage when a workflow engine is introduced. A responsibility of the workflow engine is to initiate a task in a certain system, wait for the result of the task and initiate next tasks in other systems consecutively until the required overall task is completed. In this way, the workflow engine is aware of the execution stage of the overall task.

One may also replace certain system components by introducing corresponding information in the workflow engine. As a result, the system interconnection complexity is reduced from  $n \cdot (n - 1)$  to  $n$  as illustrated by the mesh and star topology in Figure 21. Hence, the relative complexity increase for a fully meshed structure compared with a star structure goes like  $(n - 1)$ , where  $n$  is the number of systems to be interconnected. Then, the interconnection hub would add to the complexity.

The common communication system is usually implemented as a message bus. Depending on the solution such a message bus may be broadcast or publish/subscribe. In the former case, each system has to filter out unwanted messages. In the latter case a system has to subscribe to certain messages and the bus is responsible for filtering and routing.

The common communication system does not remove the challenge of translating data between systems. To alleviate this, the *common information and data model* could be introduced. Such a common data and information model will assist for the integration when each system has the same interpretation, hierarchy and relations between the shared information entities.

Irrespective of the data configuration, persistence of data must be considered. Storing the same information in multiple places poses an additional problem for keeping the different instances of the information

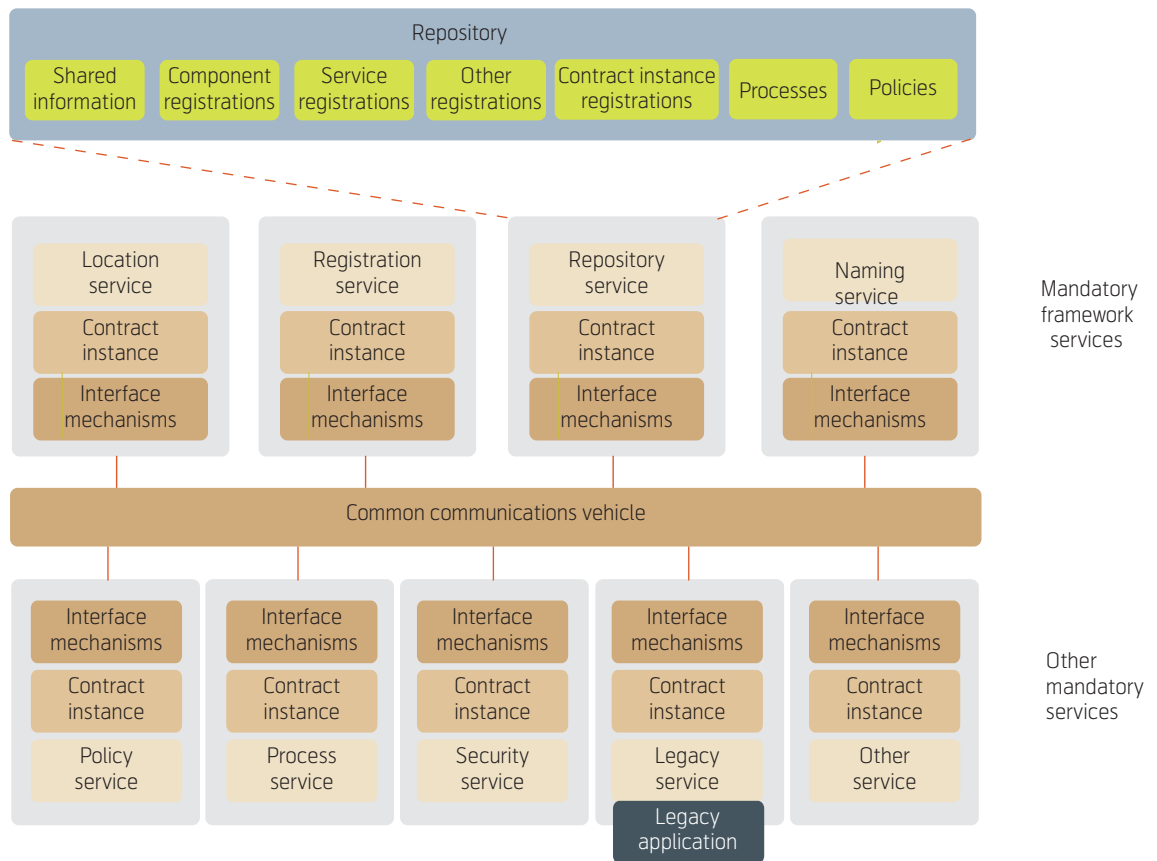


Figure 20 Technology Neutral Architecture as part of NGOSS

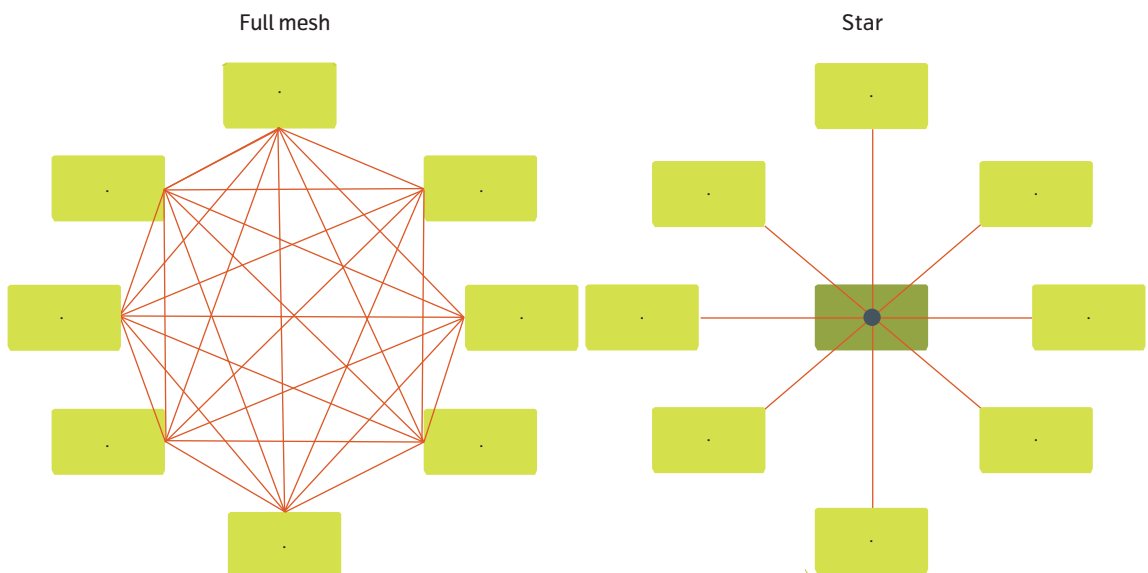


Figure 21 Illustration of system interconnection for mesh versus star topology

synchronized. The TNA defines a data steward that is responsible for ensuring the integrity of distributed data. This must then include control mechanisms to enable systems to be notified when changes are made to relevant data.

There are several approaches to follow for system integration. One example is the following steps and required functionalities:

- 1 A change is detected that triggers the tasks to be performed.
- 2 Data is exported from source systems.

- 3 Data is transferred to the system responsible for processing it.
- 4 The gathered data is processed according to integration logic.
- 5 Data is transformed to a representation that is understood by destination systems.
- 6 Data is transferred to destination systems.
- 7 Data is imported to destination systems.

Steps 5 and 6 could be swapped. In addition to these steps, an integration could be done such that the participating systems will be left in a consistent state even if the transaction that is integrated fails to execute completely.

The description above has clear resemblances with the orchestration and policy as presented above. In effect, both a looser coupling has been achieved and higher levels of dynamics can be implemented. These are characteristics that should be captured by the technical architecture if requested.

#### 9.4 ITU NGN Management, SOA

As stated in [M.3060] the NGN management architecture is a Service Oriented Architecture (SOA). SOA is a software architecture of services, policies, practices and frameworks in which components can be reused and repurposed rapidly in order to achieve shared and new functionality. This enables rapid and economical implementation in response to new requirements thus ensuring that services respond to perceived user needs.

Major goals of SOA in comparison with other architectures used in the past are to enable

- *Faster adaptation* to changing business needs;
- *Cost reduction* in the integration of new services, as well as in the maintenance of existing services.

Main features of SOA are:

- Loosely coupled, location independent, reusable services;
- Any given service may assume a client or a server role with respect to another service, depending on situation;
- The 'find-bind-execute' paradigm for the communication between services, see Figure 22;
- Published contract-based, platform and technology-neutral service interfaces. This means that the interface of a service is independent of its implementation;

- Encapsulating the lifecycle of the entities involved in a business transaction and exposing a coarser granularity of interfaces than an Object-Oriented Approach.

Although SOA represents some interesting principles, it is still a challenge to realise all the potential benefits. Take 'loose coupling', for example; the idea is that other components could call functions offered by a component. The functions need to be exposed via defined interfaces that allow a service to be accessed and invoked. One objective is to reduce inter-dependency, so components can be upgraded fairly independently of each other. A key point to realise this is the function/service registry that needs to scale and be maintained. It also becomes a question how to define the proper functions (eg. on what level of granularity). Typically, a somewhat tighter coupling is implemented, although still allowing re-use and flexibility of upgrading components.

SOA claims re-usability during run-time. There are other techniques that allow re-usability during coding/design times. One needs to reflect on whether full re-usability during run-time, compared to other levels of re-usability, returns the effort needed for communication between components respecting response times. For the extreme case, if only one component were to implement a unique function, there would also be a single-point-of-failure that would need to be addressed. Components must also be monitored to avoid becoming performance bottlenecks.

However, both resilience and performance can be easier to tackle when critical components are identified. The performance of web services, using XML and SOAP, is commonly lower than other methods such as Java Remote Method (overhead of nine times have been reported, ref. [Rich08]).

An Enterprise Service Bus (ESB) could fill the role of interconnecting systems. Typically, an ESB performs such tasks as messaging, invocation, management/

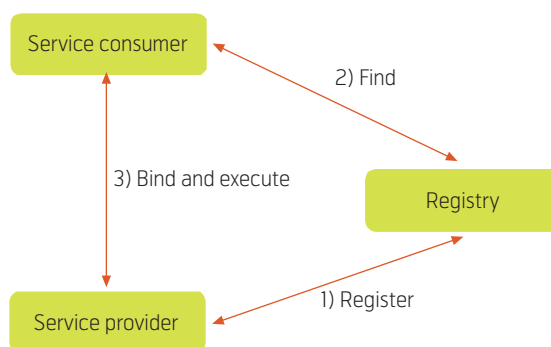


Figure 22 Find-bind-execute principles of SOA

## eXtensible Markup Language (XML)

The eXtensible Markup Language (XML) is a standard developed by the W3C. It was approved in February 1998. XML is an evolution of the HyperText Markup Language (HTML) commonly applied for web pages. There are two fundamental differences between XML and HTML:

- XML separates content from presentation. Presentation of information is described in a separated 'style sheet'.
- XML is a set of rules for how to define one's own tag sets. HTML uses pre-defined tags, while XML allows definition of more tags (hence the name *eXtensible*).

These differences lead to benefits like:

- Common style sheet: Many documents can be linked to a single style sheet. When design should be changed for a set of documents, introducing these changes in the common style sheet will suffice. Style sheets can also enable multiple views.
- Reading information in multiple ways. The same XML document could be read with different formatting, eg. different for human users and machines (translators, printer, etc.).
- Documents become applications. Processing language provided by XSLT supports interactivity and processing 'documents'.
- Efficient support for search. XML allows differentiating between books *about* Brundtland and *written by* Brundtland.
- Additional vocabularies for supporting different information types, such as voice, graphics.

The XML family consists of a number of modules like:

- Document Type Definition (DTD); This is what you are going to get (provides rules for a set of XML documents).
- Extensible Style sheet Language (XSL) and Cascading Style Sheet (CSS); This is what that should look like (style sheet languages).
- XSL Transformations (XSLT); This is how that should be processed (transformation language).
- Xlink and Xpointer; This is related to that in this way (mechanisms for composing virtual documents through pointers).
- XML Name spaces; Look there to see what this means (removing ambiguities in the meaning of tags).
- XML Schema (provides an enriched DTD).

governance, routing, business process orchestration/choreography and security. Following SOA (or more specifically, Web services) principles, XML message formats are commonly used. Correspondingly, Web Services Description Language (WSDL) is used for invocation of service components.

Some areas related to integration are:

- *Real-time versus delayed*. In real-time integration the new information is delivered to other systems immediately as it becomes available. Two common techniques to detect a change are listening and polling. Listening in real-time mode implies that a signal is always emitted when a change happens and this signal is detected by relevant systems.

For polling in delayed mode there is a periodic check on whether a change has happened. Listening has the advantage that changes are immediately propagated. However, polling may be the only option for some cases and do work sufficiently when there are no strict timing requirements.

- *Application Programming Interface (API)*. This is typically an interface where services of a certain system can be requested by other systems. For some system environments a Software Development Kit (SDK) may be provided. An advantage of using an API for system integration is that it provides a black box approach, to some extent. The other systems do not need to know details regarding internal functionalities that are executed when the API functions are called. Hence, the implementation of a system is decoupled from the API usage and may be changed without affecting the systems applying the API.
- *Data formats for interactions*. Two types of data formats are commonly found; i) eXtensible Markup Language (XML), and ii) Binary data. XML has been developed by the World Wide Web Consortium (W3C) and can be used to describe other languages and incorporate meta-information describing contents within the same document. An advantage of XML is that it is text-based. The metadata information makes XML documents properly self-documenting by descriptive structures and field names to ease the location of needed parsing information. A disadvantage of XML is that it is not suitable for data that cannot be represented as text, such as pictures (the non-textual data could then be given as a separate object). Another disadvantage is that XML is quite verbose and redundant compared to binary formats. For the binary exchange format, any type and encoding may be applied. A primary advantage is the performance as efficient coding could be applied. The structure of binary data must be exactly defined; each byte of the data must be unambiguously interpreted by all relevant systems and introducing changes would imply that the format change is propagated to every system involved.

## 10 Example – BT's Service Oriented Infrastructure

For the Information Communication Technologies area, BT has elaborated a Service Oriented Infrastructure as depicted in Figure 23. A 3-layer model has been chosen:

- *Virtualised resources*. Resources can be offered to services and applications as requested. The virtualisation allows an abstraction, hence the users of the

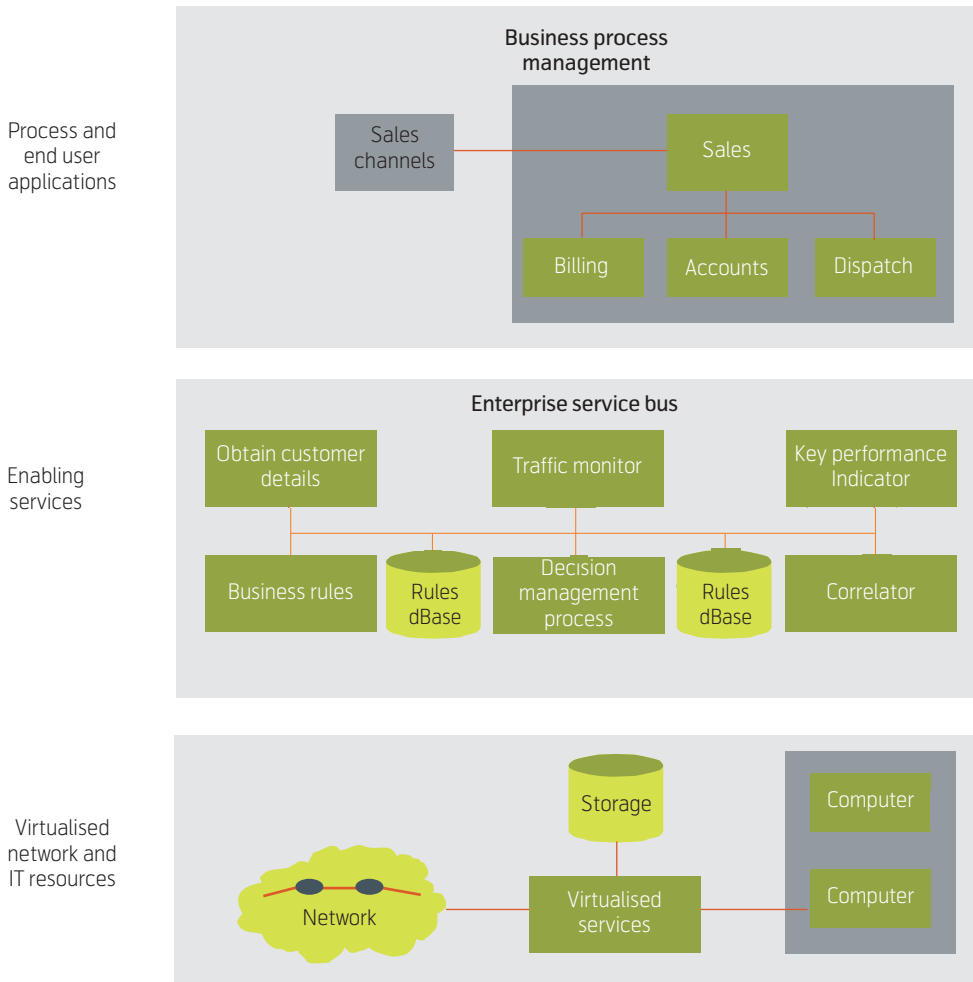


Figure 23 BT's Service Oriented Infrastructure (from [Witt08])

resources do not need to be aware of the details on how these resources have been implemented.

- *Enabling services.* This includes hosting of reusable components including mechanisms for security, resilience, orchestration, and discovery.
- *Processes and applications.* Representing applications and policies that bind components together thereby providing a usable function for the organisation.

Applying this model, IT goals can be linked to business objectives. As presented in [Smit08] the following hierarchy of goals can be identified:

- The vision, which is the overall direction the business is going in.
- High level business goals, expressed in commercial terms or business outcomes, which support achievement of the vision.
- Objectives, the second level business goals which cumulatively make up the overall business goals.

- Critical Success Factors (CSFs), which must be in place to ensure that the objectives are met.
- Key Performance Indicators (KPIs), basic measures which are indicative of the CSFs being achieved.

An essential point is to ensure that these goals are related in such a way that when lower level goals are achieved, the higher level goals will also be achieved. This may not be a trivial task, however, for a complex organization.

Overall objectives for BT are outlined in [Boot08]:

- Right fist time – How often the customer gets what they want delivered the first time;
- Cycle time – How long it takes to deliver to the customer what they want, be it an existing service or creating and delivering a new service;
- Total cost of ownership – How much it costs to give the customer what they want, covering all the costs from developing the service through delivering the service to withdrawing the service.

The general drivers behind the BT model are:

- *Virtualise*. In general, equipment is inefficiently used – so share it through virtualisation.
- *Simplify*. Managing complexity is difficult and costly to implement unless complexity is reduced, therefore simplify by standardising and limiting variability.
- *Automate*. Using manual processes costs money, generates errors and takes time – therefore automate.
- *Service*. To simplify the management of the remaining complexity organise around service, not components.
- *Carbon footprint*. The transformation must deliver a reduction in power consumption with a consequential reduction in carbon emissions.

## 11 The AKARI Architecture Requirements

[AKAR08] elaborates on a set of design requirements for a future Internet as shown in Figure 24. In this form the requirements are quite general and more details have to be supplied (ref. [AKAR08]).

Basic principles include simplicity, layered structure and end-to-end perspectives. It is motivated by enabling further diversity and expandability. It is also emphasised how to effectively map the Internet representation to real-world representation. Three elements to support this are i) separation of physical and logical addressing, ii) bi-directional authentication, and iii) traceable individuals and entities.

The design factors are considered when elaborating the new generation network requirements [AKAR08]:

- *Large capacity*. Increased speed and capacity are required to satisfy future traffic needs, which are estimated to be approximately 1000 times the current requirements in 13 years.
- *Scalability*. The devices that are connected to the network will be extremely diverse, ranging from high-performance servers to single-function sensors. Although little traffic is generated by a small device, their number will be enormous, and this will affect the number of addresses and states in the networks. Self-organizing controls and autonomous actions may be needed to maintain control in large-scale networks and networks with dynamic varying topology.
- *Openness*. The network must be open and able to support appropriate principles of competition.
- *Robustness*. High availability is crucial because the network is relied on for important services such as medical care, traffic light control and other traffic services, and bulletins during emergencies.
- *Safety*. The architecture must be able to authenticate all wired and wireless connections. It must also be designed so that it can exhibit safety and robustness according to its conditions in the event of a disaster.
- *Diversity*. The network must be designed and evaluated based on diverse communication requirements without assuming specific applications or usage trends.
- *Ubiquity*. To implement pervasive development worldwide, a recycling-oriented society must be built. A network for comprehensively monitoring the global environment from various points of view is indispensable to accomplish this.

- Peta-bit/s backbone, 10 Gbit/s fibre to the home, e-Science
- 100 billion devices, machine-to-machine, 1 million broadcasting stations
- Principles of competition and user orientations
- Essential services (medical care, transportation, emergency services), 99.99% reliability
- Safety, peace of mind (privacy, monetary and credit services, food supply traceability, disaster services)
- Affluent society, disabled person, aged society, long-tail applications
- Monitoring of global environment and human society
- Integration of communication and broadcasting, Web 2.0
- Economic incentives (business cost models)
- Ecology and sustainable society
- Human potential, universal communication



- Large capacity
- Scalability
- Openness
- Robustness
- Safety
- Diversity
- Ubiquity
- Integration and simplification
- Network model
- Electric power conservation
- Extendibility

Figure 24 Architecture design requirements derived from societal requirements (based on [AKAR08])

- *Integration and simplification.* The design must be simplified by integrating selected common parts, not by just packing together an assortment of various functions. Simplification increases reliability and facilitates subsequent extensions.
- *Network model.* To enable the information network to continue to be a foundation of society, the network architecture must have a design that includes a business-cost model so that appropriate economic incentives can be offered to service providers and businesses in the communications industry.
- *Electric power conservation.* As network performance increases, its power consumption continues to grow, and as things stand now, a router will require the electrical power of a small-scale power plant. The information-networked society of the future must be more Earth friendly.
- *Extendibility.* The network must be sustainable. In other words, it must have enough flexibility to

enable the network to be extended as society develops. This may imply that individual entities within the network can operate in a self-distributed and self-organising manner.

The AKARI project also elaborates on two options called i) the next generation, and ii) the new generation. The former takes on a migration perspective, while the latter looks at an architecture not restricted by installed systems of today (so-called *clean-slate approach*). Table 2 lists the main differences between these two perspectives.

An illustration of the concepts of new generation network is given in Figure 25.

## 12 Concluding Remarks

This paper addresses a selection of technical items to be addressed when devising a technical architecture. It is important to take on an inside-out and an outside-in view to see that the solution can be integrated

Attribute	Next Generation Network	New Generation Network
Assumed implementation time	By 2010	By 2015
Creation method	Add QoS and authentication to existing IP	Create new network without being committed to IP
Trunk link capacity	O-E-O conversion: Less than peta-bit/s capacity	All-optical: Greater than peta-bit/s capacity
Assumed terminals and applications	Integration and creation of advanced versions of existing terminals and applications such as triple- or quadruple-play services	Unknown but highly diverse, ranging from devices acting in conjunction with massive information servers to tiny communication devices such as a sensor
Power consumption	Power consumption at several mega watts (transformer substation scale)	Power conservation by a factor of at least 1/100 according to multi-wavelength optical switching
Security	Successive violations of principles such as firewalls, IPsec and IP traceback	Control spam of DoS attacks by addressing tracing and end-to-end and inter-network security
Robustness	Supported by enhancement of management function by businesses	Robustness is provided by the network itself
Routing control	Distributed centralized control following IP, MPLS required for high-speed rerouting, long fault detection time	Introduction of complete distributed control, increase in failure-resistance and adaptability, inclusion of sensor networks and ad-hoc networks
Relationship between users and the networks	Although there are some constraints on openness stipulated by UNI, ANI and NNI, reliability is increased	Provides openness from a neutral standpoint and users can bring new services
Quality assurance	Priority control for each class by using IP	Quality assurance that includes bandwidth for each flow using packet switching or paths appropriately
Layer configuration	Thick layer structure	Layer degeneracy and cross-layer control centred around a thin common layer
Integration model	Vertical integration orientation	Vertical or horizontal integration possible
Basic principles	Set from a business standpoint while using IP	Set from a clean slate to match future requirements
Sustainable evolution	Has limitations due to IP	Has sustainable evolution capability that can adapt to a changing society
Access	Up to 1 Gbit/s for each user	Over 10 Gbit/s for each user
Wired – wireless convergence	IMS	Context aware
Mobile	(Under investigation)	ID locator separation
Number of terminals	Up to 10 billion	Over 100 billion

Table 2 Differences between next generation and new generation network architecture (from [AKAR08])

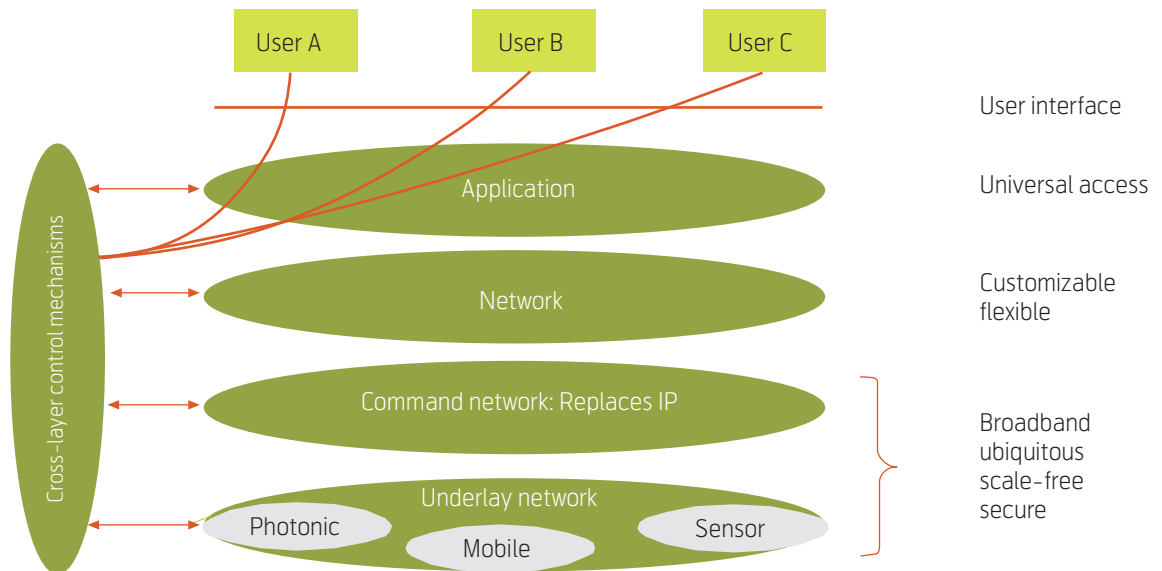


Figure 25 AKARI conceptual new generation network configuration (adapted from [AKAR08])

into the relevant environment, is able to deliver the services in question, and can be implemented in an efficient manner. It must also be flexible with respect to incorporating future features. In particular this is expected to be important when it comes to relations

between a provider and its partner. Figure 26 summarizes one questionnaire addressing what are seen as the most important factors for success of telecom operators.

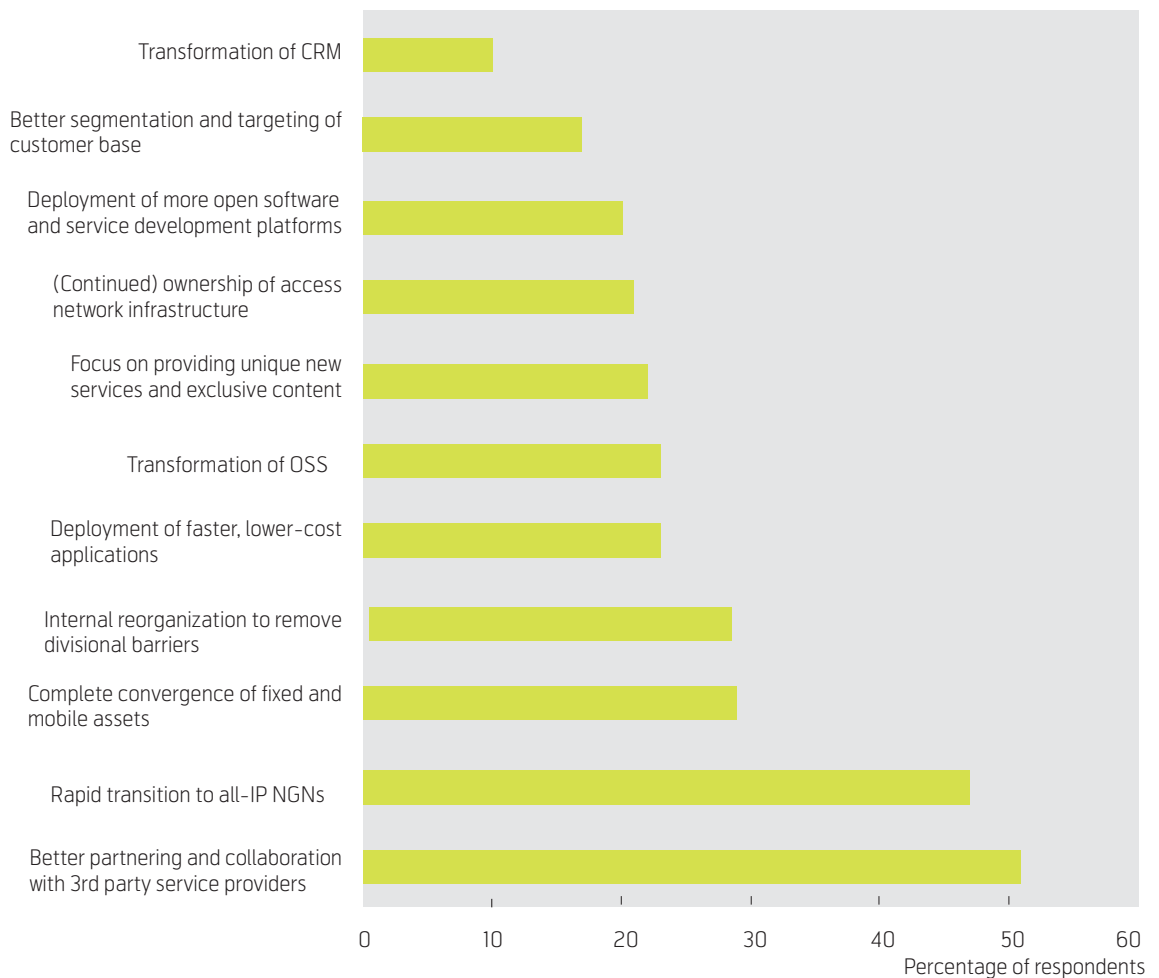


Figure 26 Most important factors for mainstream telcos' future success (from [Finn08])

Several international bodies work on issues related to technical architectures for telecom and Internet. Some of the main standards bodies to be observed are:

- International Organisation for Standardization (ISO) develops standards for computer, programming languages, data transmission and computer networking.
- International Telecommunication Union (ITU) is a United Nations body. ITU develops telecommunications standards, some jointly with the International Organization for Standardization (ISO).
- The European Telecommunication Standards Institute (ETSI) develops standard for Europe. Many of these are adaptations of ITU and/or ISO standards.
- The Institute of Electrical and Electronic Engineers (IEEE) develops worldwide standards. Popular ones are IEEE 802.11 for WLAN and 802.16 for WiMAX, although IEEE is also behind the Ethernet family of standards.
- Internet Engineering Task Force (IETF) develops protocols and mechanisms within the Internet areas.

In addition there is a wide range of other bodies developing pre-standards and de facto standards, among them the Third Generation Partnership Project (3GPP), Open Mobile Alliance (OMA), Home Gateway Initiative (HGI).

This paper is not making justices to all the items that should be considered when elaborating technical architectures. Several further examples are provided in accompanying papers in this issue of the *Teletronikk* magazine.

## References

[AKAR08] AKARI Architecture Design Project. *New Generation Network Architecture AKARI Conceptual Design* (ver. 1.1). October 2008

[Aude08] Audestad, J A. *Technologies and Systems for access and transport networks*. Artech House, 2008.

[Boot08] Boot-Handford, K. BT's virtual data centres. *BT Technology Journal*, 26 (1), 38-45, 2008.

[Cuev08] Cuevas, M et al. Using service orchestration to create integrated services in next generation networks. *BT Technology Journal*, 26 (1), 58-70, 2008.

[ITUdist09] *Distributed Computing: Utilities, Grids and Clouds*. ITU-T Technology Watch Report, 2009.

[Jens09b] Jensen, T. Business aspects impacting technical architecture. *Teletronikk*, 105 (2), 47-62, 2009. (this issue)

[Jens09n] Jensen, T. Next Generation Network architecture. *Teletronikk*, 105 (2), 134-156, 2009. (this issue)

[M3060] ITU. *Principles for the Management of Next Generation networks*. Geneva, International Telecommunication Union, March 2006. (ITU-T Rec. M.3060)

[Mckee08] Mckee, P, Fisher, M. The role of service level agreements in service oriented architectures. *BT Technology Journal*, 26 (1), 79-85, 2008.

[Musc08] Muschamp, P. Service innovation (in a software world). *BT Technology Journal*, 26 (1), 97-111, 2008.

[OMA-PEEM] Open Mobile Alliance. *Policy Evaluation, Enforcement and Management Requirements*. Ver. 1.0. June 2008.

[OMA-SE] Open Mobile Alliance. *OMS Service Environment*. Ver. 2.0. May 2005.

[Rich08] Richardson, M. SOA devil's advocate. *BT Technology Journal*, 26 (1), 123-132, 2008.

[Smit08] Smith, E A. Engineering the network for applications and services. *BT Technology Journal*, 26 (1), 25-37, 2008.