

Chapter 2

Service Control Paradigms and Network Architectures

Telcos' viewpoint is "Network is King". Webcos' viewpoint is "Service is King"—the web client and its services are the center points. However, it is not easy to define what "service" is: a bunch of resource; collection of tasks; particular interactions, and semantics; or all the above. By exploring the concepts of service, we can see how services and their definitions evolve, and how their gravity center shifts to unfamiliar zones. The service control is shifting from client-server interaction mode to more open but perplexing peer-to-peer mode. This has a knock-on transforming effect on service structure, message channeling, parallel processing, etc.

Service network architecture has already undergone radical changes with the advent of Cloud and distributed platforms. A middleware layer is bringing edge of the network services back into the network, so it is no longer "stupid". The Telecom's roll out of IMS is decoupling service control from the transport network, but it perpetuates the Telecom network-centric approach. Web style services challenge the perimeterization of services—geographical scope and the Quality of Service (QoS) supremacy. The future service control paradigm must cope with multiplicity of service architectures, serving client-server, network intelligence, and peer-to-peer, all at the same time.

2.1 Introduction

Intuitively, services in the context of Information and Communications Technologies (ICT) are a set of discrete and valuable functions provided by a system. Such a system, in the context of communications, will be distributed, i.e., parts and functions of it are not placed in the same location. In addition, a multitude of users will be requesting for the service. The parts of a distributed system as well as the system and the users need to interact and hence some interaction models and mechanisms should be defined in order to well support the provision of the service

to users. Since services are provided by a software infrastructure, a general architecture of these systems (and their subsystems) needs to be identified in order to facilitate the design and the life cycles of services. ICT and Communications services presume also the ability to exchange information and hence networks are to be used.

Services can be interpreted in different ways depending on the perspective, the Telecommunication Operators (or Telcos) see services as an extension of basic communication means, Web Companies (or WebCos) see services as functionalities that they can provide/sell by means of web capabilities, other IT (Information Technology) actors consider services as software packages that can be provided and sold to users that will use them in order to accomplish a set of tasks or functionalities (e.g., a spreadsheet). It is important to have a common ground for expressing and discussing the meaning of services.

Services are accessed by the users in a very different way and the internal working of services is characterized also by different control means (i.e., the ways in which functionalities are coordinated within the “service system” and they made available to the users). Also from this perspective, the views on services are profoundly different: Telcos see services as sets of functionalities that deeply rely on the network. Actually the network is driving the service provisioning by requesting the provision of functionalities that the control layer of the network is not capable of delivering. WebCos consider services as a set of functions requested in a client—server functions (the WebCos acting as Server) that can be accomplished with a complex and highly distributed back end system. Typically a WebCo is seen as a “server”. Other approaches to services, for instance in Peer-to-Peer or overlay models, adopt different interaction and control paradigms on top of a logical platform. The platform is a means to integrate valuable functions provided by many peers or entities and generic functions needed to support a logical view of the system that is delivering the functions themselves. From this standpoint, also a system devoted to deliver services is conceptually and practically very different depending on the stakeholder that is offering the service functionalities and is building and operating a system supporting the delivery of functions. These differences result in distinct service architectures that do emphasize the approach pursued by the specific Actor and do leverage the specific assets of the Stakeholder (e.g., the network, the data center, the logical overlaying of functions).

The relationship between the network and the services is in fact an important aspect to consider and define, in order to properly understand the intricacies of ICT and Communications services. For Telcos, it is the major asset to leverage in providing services (actually services can be seen as “extensions” or enrichments of the major capability offered by the network: the network is providing the basic service, i.e., connectivity, and services rely on it and functionally extend its properties). For WebCos, the network is just a pipe that is providing a single functionality: the best effort connectivity. All the relevant functionalities are to be provided to the edge and in particular by powerful servers. In peer-to-peer systems, the network is logically abstracted and its relevant properties are offered as services to be integrated with other capabilities and functions offered by the peers.

In order to shed a light on these different approaches and their consequences, a set of examples of some services already implemented adhering to different approaches is presented. Some existing analysis, e.g., (Minerva et al. 2011; Minerva 2008a, b, c), address many of the topics of this chapter and could be considered as a sort of guide to these issues. It also considers the interaction and control paradigms for their expressive power, i.e., the capability to represent and implement the services.

2.2 Services

As a starting point for a Service definition, we will consider the World Wide Web Consortium (W3C) definition (Booth et al. 2004) because it is generally applicable to represent the concept of a service: *“a service is an abstract resource that represents a capability of performing tasks that represents a coherent functionality from the point of view of provider entities and requester entities. To be used, a service must be realized by a concrete provider agent¹”*. A service is then a complex concept that involves many actors/agents that offer resources supporting valuable functions. These functions can cooperate in order to achieve the goals of a service. According to W3C, a Service can be represented as in Table 2.1.

Figure 2.1 depicts the complexity of the service definition.

The W3C definition of a service is a generic one, but it contains some elements that pertain to the specific context and interaction model that are underpinning the web services. For instance, even if it is not cited, the underlying interaction model is client-server in which a requester agent (a client) will send messages (requests) to a provider agent (i.e., a server). The service is a resource that is strictly owned by a person or an organization; this approach is straightforward for the web context (based on a client-server interaction paradigm), but not necessarily fits for other interaction paradigms (e.g., the peer-to-peer). The actions performed by a service aim at achieving a goal state (a predetermined situation favorable to the requester and/or the provider) this is not always the case for other systems based on other interaction paradigms.

Also the Telecom related definitions of service are in a way the brain child of a biased vision of the service world, for instance: *“telecommunications service: 1. Any service provided by a telecommunication provider. 2. A specified set of user-information transfer capabilities provided to a group of users by a telecommunication system. Note: The telecommunications service user is responsible for the information content of the message. The telecommunications service provider has the responsibility for the acceptance, transmission, and delivery of the message”* (NTIA 1996). In addition the term “service” in a telecoms world is very frequently

¹An agent is a program acting on behalf of a person or organization.

Table 2.1 W3C service definition

Property of a web service	Description
• A service is a resource	• A resource is an entity that has a name, may have reasonable representations and which can be owned. The ownership of a resource is critically connected with the right to set policy on it
• A service performs one or more tasks	• A task is an action or combination of actions that is associated with a desired goal state. Performing the task involves executing the actions, and is intended to achieve a particular goal state
• A service has a service description	• A description is a set of documents that describe the interface to service and its semantics
• A service has a service interface	• An interface is the abstract boundary that a service exposes. It defines the types of messages and the message exchange patterns that are involved in interacting with the service, together with any state or condition implied by those messages
• A service has a service semantics	• It is the expected behavior when interacting with the service. The semantics express a contract (not necessarily a legal contract) between the provider entity and the requester entity. It expresses the intended effect on the real world caused by invoking the service. Service semantics may be formally described in a machine-readable form, identified but not formally defined, or informally defined via an “out of band” agreement between the provider entity and the requester entity
• A service has an identifier	• An identifier is a unique unambiguous name for a resource
• A service has one or more service roles in relation to the service’s owner	• A service role is an abstract set of tasks, which is identified to be relevant by a person or organization offering a service. Service roles are also associated with particular aspects of messages exchanged with a service
• A service may have one or more policies applied to it	• A policy is a constraint on the behavior of agents or people or organizations
• A service is owned by a person or organization	• An agent has the right and authority to control, utilize and dispose of the service
• A service is provided by a person or organization	<i>An agent has the right and authority to provide and operate the service</i>
• A service is realized by a provider agent	• A provider agent is an agent that is capable of and empowered to perform the actions associated with a service on behalf of its owner—the provider entity

interpreted as a “network” or at least as the transport service that a network provides, for instance: “3G: *Third-generation mobile network or service. Generic name for mobile network/service based on the IMT-2000 family of global*

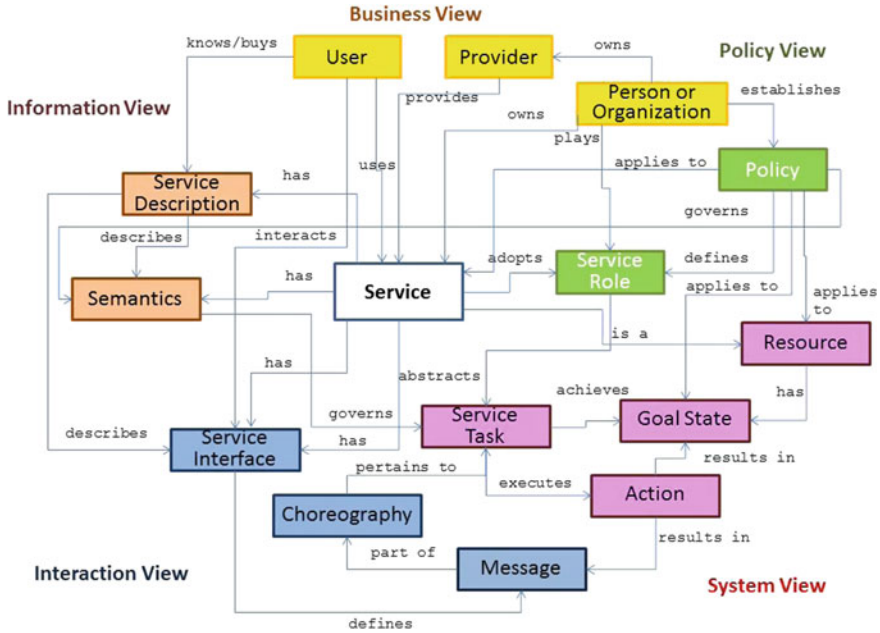


Fig. 2.1 The W3C service definition [(Booth et al. 2004) Fig. 2.8]

standards” (ITU 2011). Often also the literature follows this approach such as (Znaty and Hubaux 1997) that stated:

- “Telecommunication services is a common name for all services offered by, or over, a telecommunication network”;
- “Value added services are services that require storage/transformation of data in the network and are often marketed as stand-alone products. Examples of such services are freephone, premium rate, virtual private network and tele-banking. Many value added services can be offered by special service providers connected to the network”;
- “Teleservices include capabilities for communication between applications. Teleservices are supplied by communication software in the terminals. Telephony, audio, fax, videotex, videotelephony are examples of teleservices”.

All these definitions bring in the concept of a network supporting a major service on top of which “added value” functions and services can be instantiated as represented in Fig. 2.2.

These definitions reflect the vision of services: the network is the main asset and it provides a main service, THE voice service, on top of which several functions or services can be provided. They add value to the unique service at hand.

A more detailed study on service definition that is also considering the integration of different paradigms is given in (Bertin and Crespi 2013): “a service is

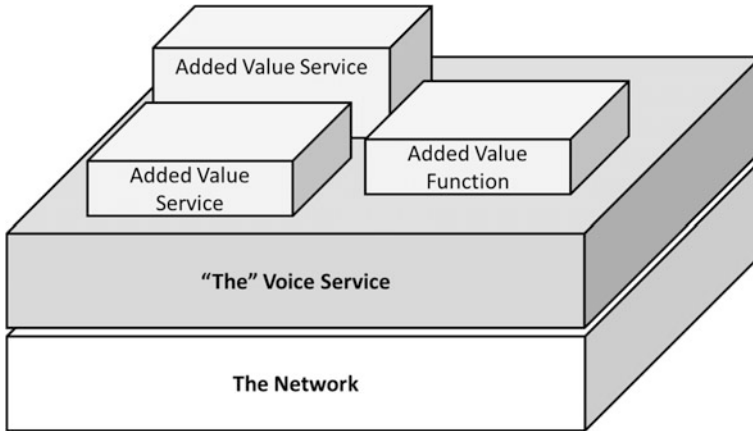


Fig. 2.2 The network, the service and added value functions/services

defined ... as an immaterial business activity made available by a service provider and offering an added value to a user".

Also in the peer-to-peer sector, the service definition strongly depends on the "view" of the system organization for supporting functionalities. For instance (Gradecki 2002) defines a service as "*predefined functionality that can be utilized by peers*" where a peer is "*an application, executing on a computer, that has the ability to communicate with similar peer applications*".

These definitions introduce another issue: the differences between services and applications. Actually services and application are sometimes used as synonymous, so an application can be defined as a set of functions that have meaning and usefulness within a specific domain or a software program. In order to differentiate the two concepts, a service is an application that can serve other applications, i.e., a service is an application that offers reusable functionalities that other applications can use over time.

Simply put, the definition of a service is a complex matter that deserves more than a mere technology approach, as stated in (Jones 2005): "*A service's intention is to undertake certain functions to provide value to the business; its specification isn't just the direct service it provides but also the environment in which it undertakes those functions*". This statement points to the problem that services have also nonfunctional goals and they could be so important to determine the meaning and the context of the service itself. Also the previous definition, for instance, could be seen as biased: it mentions the value for business, but a service could have a value for a community independently from any commercial value and business. A User Centric view on services could also change the perspective. Paraphrasing the previous sentence, it could be stated that "*a service's intention is to undertake certain functions to provide value to a person, a group of persons, a community or a business organization and its representatives; its specification isn't just the direct service it provides but also the environment in which it undertakes those functions*".

Services are becoming a crucial topic for many industries; as a consequence a new area is attracting interest from researchers: the Service Science (2013). It is an interdisciplinary approach to the study, design, and implementation of services systems (Service Science 2013). In this context, a service is even more difficult to define because the definition encompasses business, social, technology, and ecosystems issues. A service system, i.e., a system that can support services is a complex system in which specific arrangements of people and technologies take actions that provide value for others.

As stated in (Cardoso et al. 2009), “*the set of activities involved in the development of service-based solutions in a systematic and disciplined way that span, and take into account, business and technical perspectives can be referred to as service engineering: Service Engineering is an approach that provides a discipline for using models and techniques to guide the understanding, structure, design, implementation, deployment, documentation, operation, maintenance and modification of electronic services (e-services)*”. They propose a new approach to services: services are tradable and can be supported by Internet systems. This combination leads to an Internet of Services that needs to be supported by appropriate Service Architectures and by the Service Engineering.

Summarizing, services strongly depend on many factors like its goals, foreseen users, the ecosystem in which it is used, the system used to be provided and the supporting technologies and mechanisms for using it.

2.3 Interaction and Control Paradigms

Services considered in this document refer to the functions and applications that can be realized by means of ICT technologies. In this context, a service system can comprise different subsystems each one providing specific functions and executing particular tasks. In addition the system as a whole needs to interact with its users. Actually the way in which a service system interacts with users is a characterizing factor of the system itself. In fact, one first consideration is that the users can be seen as external (Fig. 2.3) to the service system or as part of it (Fig. 2.4).

In this document the way in which parts and components of the whole service system interact and cooperate is seen as the “control paradigm”, while the way in which external entities interact with the service system is termed as the “interaction paradigm”. In the case of control paradigm, the emphasis is on the architecture of the service system, while in the interaction paradigm, the focus is on the way users are accessing the system functions. This separation is not a hard one, in fact interaction and control paradigms often are synonymous, e.g., in the case of client-server systems (C-S). In addition, a complex service system could be implementing several control paradigms between its components: many web companies in fact are offering services implementing a highly distributed control paradigm between internal components and parts, and a simple client-server one for user interactions.

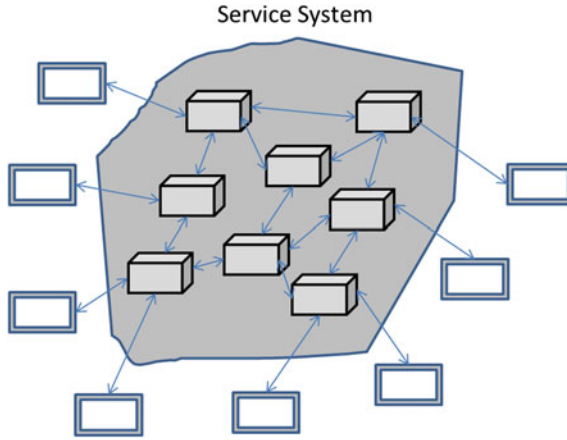


Fig. 2.3 External users' interactions with a service system

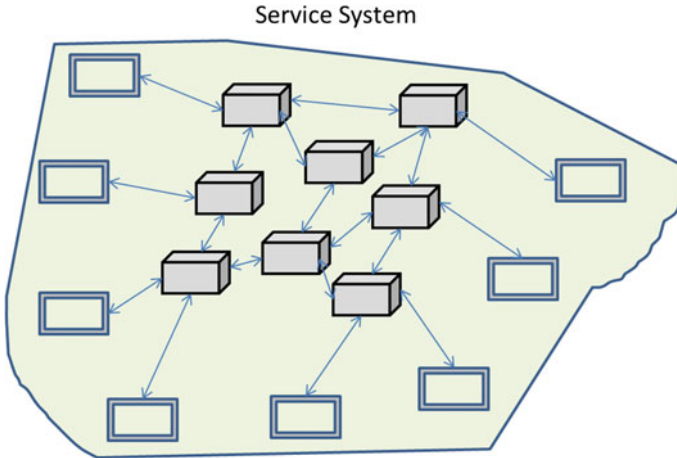


Fig. 2.4 Users as part of a service system

With reference to Figs. 2.3 and 2.4, the difference is relevant: in the first case, the relationship is clearly of a dependence, users can access the system functionalities that are owned, provided, and managed from a specific actor (the provider agent). In the second case, the ownership of the service system is blurred, since the users are providing functions and resources to the service system. It is difficult to see the users as external, without them the system would not even be able to exist. This is a great difference that points to the client-server (C-S) and the peer-to-peer (P2P) interaction paradigms as represented in Fig. 2.5.

Graphically the difference is evident: in the client-server, the central role is played by the server (that usually is seen as the service system) that is the

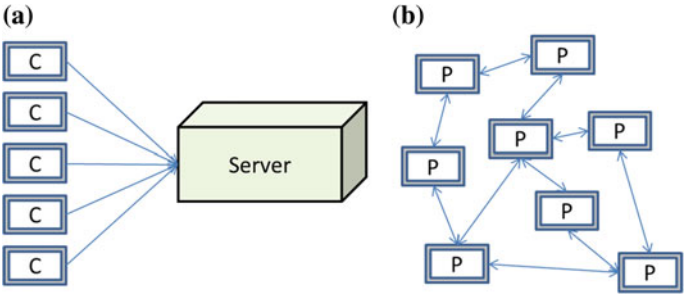


Fig. 2.5 Client–server and peer-to-peer interaction paradigms. **a** Client–server interaction paradigm. **b** Peer-to-peer interaction paradigm

centralized entity providing valuable functions to the clients that in these relationships have a lesser role. They can request functions and expect to receive a response. In the P2P paradigm, each single entity or node is equal to the others; there are not usually privileged relationships and each peer can request services from any other peer. Peers are designed and programmed for cooperation and the service system is totally decentralized. Also the communication mechanisms of P2P and C-S systems are different. In the P2P case, the communication between two communicating peers is direct in the sense that each peer knows the address of the other (even if the communication between them could go through other nodes, i.e., multi-hop manner), in the C-S case the communication is brokered, i.e., the server is in between the clients that do not have a direct reference of each other and are forced to pass through the server (Taylor and Harrison 2009). There is an extensive literature related to P2P and C-S, for example (Pavlopoulos and Cooper 2007; Orfali et al. 2007; Taylor 2005; Maly et al. 2003; Leopold 2001). Also the increasing wireless capabilities have raised the question of interaction and control paradigms (Datla et al. 2012).

The C-S paradigm is based on a very simple interaction between the clients and the server; a client sends a request (essentially a structured message) to a server and expects (nonblocking) a response from the server. Figure 2.6 depicts the simple interaction.

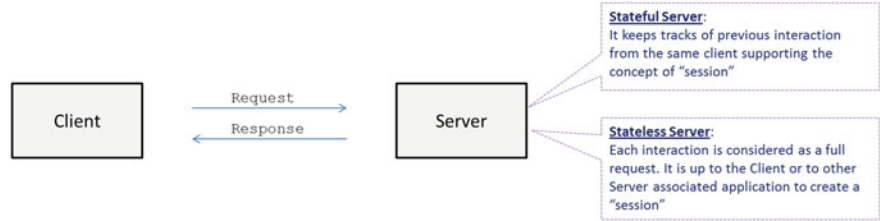


Fig. 2.6 The client–server interaction paradigm

The server is assumed to send a response to the client, however, the client cannot assume that it will be necessarily received. However, the request and response primitives are related and the client and server systems have to take care of this relationship. This means that a response cannot be received without a prior request. At the same time, a client not receiving the response could start polling the server in order to get an answer. These examples indicate that the interaction model is not balanced and there are several limitations. The server could be stateful or stateless, the difference is whether the system keeps track of the previous interactions with clients and has a finite state machine associated to the interactions going on. A stateful server is more complicated to manage especially if many clients are requesting in parallel the functions of the server. There is the REST architectural proposition (Fielding and Taylor 2002) related to web services and architecture that is promoting with a lot of success the idea of having stateless server as a principle of the architectural design.

Some interaction paradigms in proper sense are [in italics excerpts from (Foster 1995; Cachin et al. 2011)]:

- **Tasks and channels.** It is a computational model designed by Foster (1995) based on distributed tasks with logical channels connecting them. A task consists of a program, local memory, and a collection of input/output (I/O) ports. Tasks interact by sending messages through channels. A task can send local data values to other tasks via output ports. A task can receive data values from other tasks via input ports. Ports are connected by means of channels. The local memory contains the program's instructions and its private data.
- **Message passing.** *Message-passing programs create multiple tasks, with each task encapsulating local data. Each task is identified by a unique name, and tasks interact by sending and receiving messages to and from named tasks. The message-passing model does not preclude the dynamic creation of tasks, the execution of multiple tasks per-processor, or the execution of different programs by different tasks. However in practice, most message-passing systems create a fixed number of identical tasks at program startup and do not allow the tasks to be created or destroyed during program execution. These systems are said to implement a single program multiple Data (SPMD) programming model because each task executes the same program but operates on different data. The difference with tasks and channels model is that a task needs only a queue to store messages and a queue can be shared among all the tasks that need to communicate with a specific task, while in the tasks and channels model, a channel (a queue) is created for supporting the communication between two tasks.*
- **Data Parallelism.** *Data parallelism exploits concurrency by applying the same operation to multiple elements of a data structure. A data-parallel program consists of a sequence of such operations. As each operation on each data element can be thought of as an independent task, the natural granularity of a data-parallel computation is small, and the concept of "locality" does not arise naturally. Hence, data-parallel compilers often require the programmer to*

provide information about how data are to be distributed over processors, in other words, how data are to be partitioned into tasks. The compiler can then translate the data-parallel program into an SPMD formulation, thereby generating communication code automatically.

- **Shared Memory.** In the shared memory programming model, tasks share a common address space, which they read and write asynchronously. Various mechanisms such as locks and semaphores may be used to control access to the shared memory. An advantage of this model from the programmer's point of view is that the notion of data "ownership" is lacking, and hence there is no need to specify explicitly the communication of data from producers to consumers. This model can simplify program development. However, understanding and managing locality becomes more difficult, an important consideration (as noted earlier) on most shared memory architectures. It can also be more difficult to write deterministic programs.

The message-passing model is represented in Fig. 2.7. It is based on a very simple organization in which a sender forwards messages by means of a queue to a recipient.

Obviously the different mechanisms have advantages and drawback. For a deeper analysis (Kubiatowicz 1998; Foster 1995) provide more details and discussions. A short analysis of pros and cons is provided in the following:

- **Message Passing:** Message passing (MP) paradigm can be interpreted as an "interrupt with data," i.e., events are queued for processing together with associated data (they could also be large bulks of data). Receiving tasks can then operate on received data when the message is taken from the queue. There is the need of an explicit treatment of communication (e.g., "send message to task n ") and management of data (data are copied from data structures in the source of the message, transmitted to a queue and then copied to a data structure at the sink destination). Assembling and disassembling of messages can be cumbersome.
- **Shared Memory:** one of the major advantages is that the communication details between programs and the "shared memory" are hidden to the programmers, freeing them from the problem of dealing with communication (in the shared memory) and the data management (replication, caching, data coding and encoding, optimization of distribution of data). The programmers can actually focus on the parallelism and the issues to solve. However, the programs have to

Fig. 2.7 Message-passing model



adopt a sort of polling in order to synchronize with new data and new events (and this can be time and resource consuming), in addition data management is optimized having in mind a model optimization. Specific applications needing other kinds of optimization or a hybrid composition between data parallelisms and other approaches will find it difficult to implement specific application oriented policies in this context. This could be an example of abstraction that eases programming, but sometimes it is detrimental for applications that need to do “different” things.

- **Tasks and Channels:** this paradigm can be seen as a variation of the message passing. According to Foster (1995), the “task/channel” paradigm enforces the programmer to better conceptualize the communication of a parallel and distributed program. This could provide advantages in designing a large distributed system, but it also introduces more overhead in the need to control individual channels and in creating a relationship between channels and related tasks.
- **Data Parallelism:** while this paradigm can lead to significant advantages when the same instructions can be applied to different data, its general application is not easy. In fact not all the application domains offer problems that can be effectively solved with this paradigm. Actually the major drawback of this paradigm is its applicability to more restricted set of applications compared to the previous ones.

These systems are to be analyzed and used appropriately according to the specific needs of the applications. In the rest of this section, an analysis of their benefits, merits, and applicability is carried out. In large MP systems, senders and recipients share a common infrastructure made out of queues and these can be organized as a sort of message broker in charge of receiving and dispatching (in an asynchronous manner) messages between sources and sinks (Fig. 2.8).

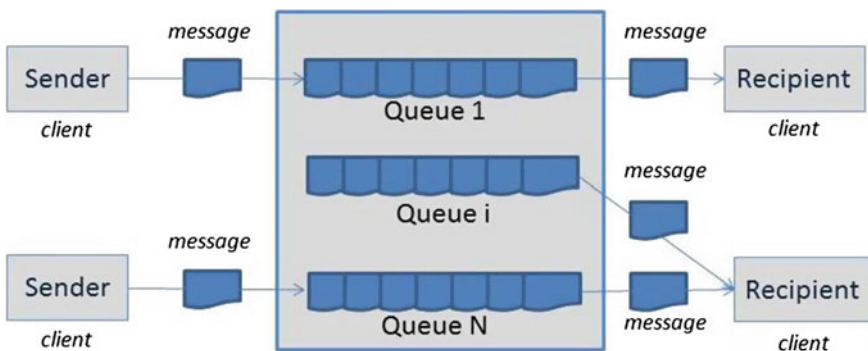


Fig. 2.8 A message-passing (MP) system

There are at least three important features.

- The model can be synchronous or asynchronous, i.e., in the first case, the sender waits for the receiver to deal with the message. In the second case, once a message has been sent, the sender can proceed with its computation. The choice of how to control the timing of operations (in a synchronous or asynchronous manner) depends on the applications requirements. The introduction of queues to store messages for later processing is a case in which asynchronous mechanisms are implemented.
- Routing: the system (especially by means of the Broker Functions) can route the messages even if the sender has not provided a full path to the destination.
- Conversion: the Broker functions can also make compatible different messages format in such a way to support the interoperability between different messaging systems.

Communication between the Sender and Receiver is not “brokered”, because the event message manager enables an asynchronous but full communication between the two parties.

The message passing, MP, and the C-S paradigms have commonalities: i.e., a sender and a receiver can be easily mapped on a client and a server. However in a MP system, the role of sender and receiver are less constraining than in a C-S system. In fact, the MP model is more granular (in a sense that the sender could be just sending messages—events—to a receiver without expecting any “response” or any service from the server). In the MP paradigm, messages are not necessarily correlated between a client and a server, and actually each entity can be a sender (and not necessarily a receiver). Under this perspective, the MP paradigm can be used in order to support a C-S model: request and response primitives can be implemented as two separate messages exchanged between a sender and a recipient. On the other side, using a C-S system for implementing a MP system introduces the logical concept that the receiving party will offer a service to the sender or at least will respond to the received message. This means that the expressiveness of the MP is greater than the C-S paradigm even at the cost of further complications² (Fig. 2.9).

The P2P paradigm is very close to the Message Passing as well as the Task and Channel models. Each peer can indeed be seen as an entity capable of being a sender and a receiver where each of them uses channels for sending and receiving data (messages, events, and flows). Actually some P2P protocols, e.g., Gnutella (Ripeanu et al. 2002), seem to be designed having the MP paradigm in mind while others are taking the Task/Channel model as the founding concept: e.g., the i2p tunneling mechanisms (Aked 2011).

²Actually a double C-S system could provide the functionalities of a MP paradigm. However the difference between the server side and the client one is such that many clients are not capable to host both the client and the server functions.

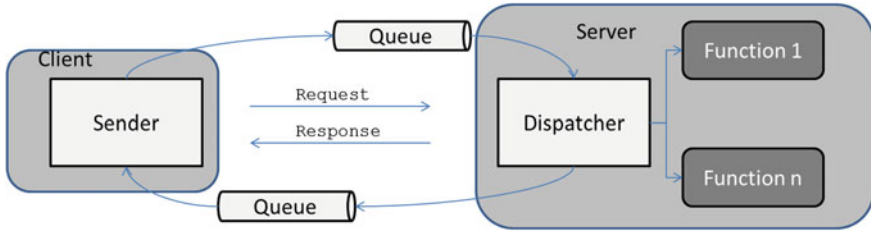


Fig. 2.9 A C-S relationship implemented by means of a MP system

Other interaction mechanisms are emerging especially for data intensive processing, for instance (Grelck et al. 2010; Margara and Cugola 2011) analyses the relations between stream processing and the Complex Event Processing.

2.4 Network Architectures and Distributed Platforms

Even if the interaction paradigms are independent from the network capabilities, in reality they are strongly influenced by the underlying communication infrastructure in several ways. As stated by Deutsch (1995) and further elaborated by Rotem-Gal-Oz (2006), (Thampi 2009), there are network fallacies that should not be ignored in designing the networked applications. Robustness and resilience of networks cannot be taken for granted and hence service design should take care of utilizing interaction, and control paradigms capable of coping with these fallacies as well as the right partition of functions within different administrative domains (including users' domain).

Another important notion related to distributed platforms is their programmability. Programmability is exerted by means of protocol interactions or more and more frequently in the Web and IT world by means of application programming interfaces (APIs). An API (Magic 2012; API 2013), is a source code-based specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables. A difference between an API and a protocol is that the protocol defines a standard way to exchange requests and responses between functional entities (to be seen as black boxes) while APIs expose a part of the software infrastructure (and organization within the functional entity). An API can be

- language-dependent, meaning it is only available by using the syntax and elements of a particular language, which makes the API more convenient to use.
- language-independent, written so that it can be called from several programming languages. This is a desirable feature for a service-oriented API that is not bound to a specific process or system and may be provided as remote procedure calls or web services.

2.4.1 Distributed Platforms as a Means to Cope with Network Fallacies

As seen for the message passing MP paradigm, there is a need to design how messages (or events) are passed between the involved entities. In addition, some extra entities besides the communication ones are usually involved in the interactions. For instance, queues and message brokers are networked elements supporting the interaction paradigm. The client-server paradigm strongly relies on the capabilities of supporting communication protocols. There are two major flavors of the client-server paradigm: the IT one and the Web one.

2.4.1.1 The Client–Server Paradigm as Supported by IT Technologies

The first one is based on remote procedure call (RPC) mechanisms as depicted in Fig. 2.10.

A few observations are useful: if the client and the server functions are synchronous, the underlying mechanisms are based on events/messages. In fact, the network drivers will fire events and messages as soon as they arrive; the underlying network is based on IP communications with two options of connection: connection-oriented (e.g., Transmission Control Protocol, TCP) or connectionless (e.g., User Datagram protocol, UDP). This means that requests can pop up asynchronously and the network driver is essentially a sort of internal queue manager; the upper layer functions are usually provided by means of application programming interfaces, API.

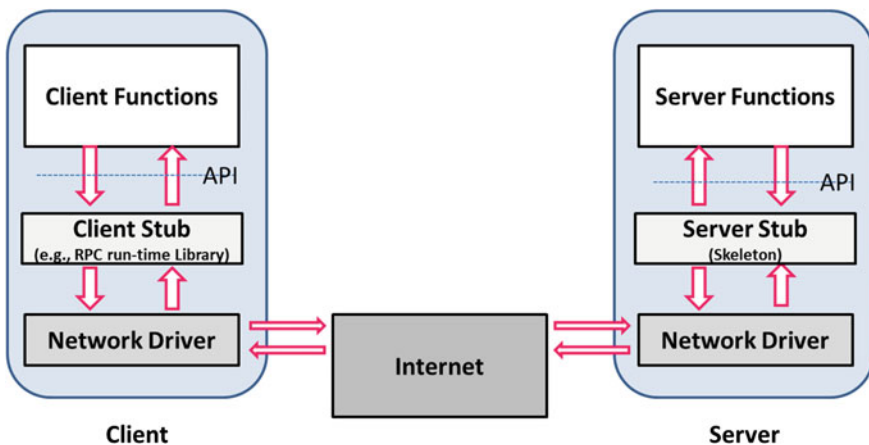


Fig. 2.10 The client–server paradigm and remote procedure call mechanisms

This approach paved the way toward middleware platforms. The initial RPC systems were in fact proprietary and confined to well-defined operating systems with machines essentially interconnected by enterprise networks. From this stage, the introduction of middleware platforms, i.e., software functions in between applications and the operating system capable of creating a sort of (distributed) unified platform supporting these functions as stated in Krakowiak (2003):

- *Hiding distribution, i.e., the fact that an application is usually made up of many interconnected parts running in distributed locations;*
- *Hiding the heterogeneity of the various hardware components, operating systems, and communication protocols;*
- *Providing uniform, standard, high-level interfaces to the application developers and integrators, so that applications can be easily composed, reused, ported, and made to interoperate;*
- *Supplying a set of common services to perform various general-purpose functions, in order to avoid duplicating efforts and to facilitate collaboration between applications.*

This approach was the basis for DCOM (Krieger and Adler 1998) and CORBA (Object Management Group 2006; Chung et al. 1998; He and Xu 2012). More recently, this approach has led to the definition of the OSGi platforms (Kwon et al. 2010, 2011) as represented in Fig. 2.11.

2.4.1.2 The Client–Server Paradigm as Supported by Web Technologies

The other trend within client-server paradigm is based on Web technologies. In this case, the basis is the Hypertext Transfer Protocol (HTTP) used as a means to send and receive requests and responses between a remote client and a server (Fig. 2.12). XML_RPC (Laurent et al. 2001) is a very simple and effective protocol for C-S

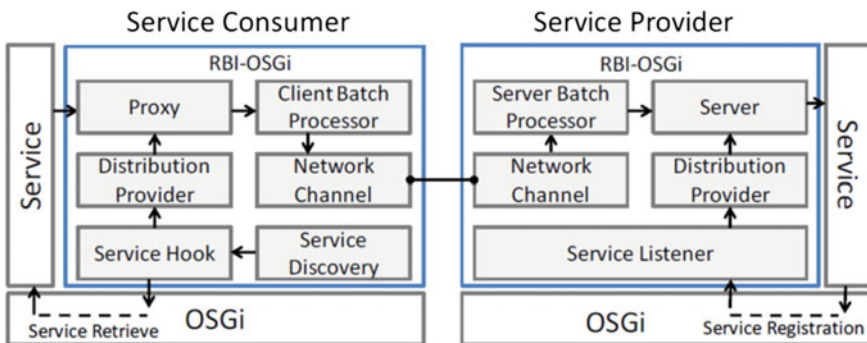
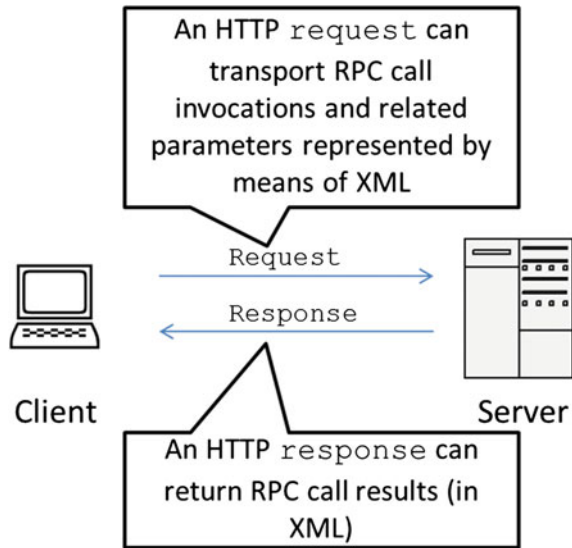


Fig. 2.11 RBI/OSGi architecture (Kwon et al. 2011)

Fig. 2.12 RPC interactions based on HTTP



interactions based on HTTP for transport and Extensible Markup Language (XML) for information representation. Remote procedure invocation and responses are coded with XML and the payload is forwarded by means of HTTP.

The simple idea that an RPC mechanism can be built using XML and HTTP (Richards 2006) has started a new view of creating web services, i.e., the possibility to dynamically invoke services in the web directly from a client remote application. This approach has led to the definition of new protocols such as Simple Object Access Protocol (SOAP) as well as to new approaches in the software architectures (e.g., REST). For a while these two approaches diverged: a lot of effort was put on the specification of Web Services Architecture and Service-Oriented Architectures. However, the increasing importance of the Representational State Transfer (REST) approach has suggested the need to integrate the two approaches under the web services definition (see Sect. 2.5).

2.4.2 Control Paradigms and Network Architectures

The combination of the communication issues, i.e., how communications capabilities are supported, offered, and used by the systems that implement services, and the interaction paradigms themselves define a set of control paradigms

- the client-server paradigm that exploits protocols, architectures, and capabilities of the Web and IT industry in order to support services by means of networked resources;

- the Network Intelligence (NI) paradigm, that combines event-based processing with the network capabilities and resources;
- the peer-to-peer (P2P) approach that combines the message passing mechanisms (or the event processing one) with distributed capabilities offered and controlled by the different peers over an overlay network.

As stated in Santoro (2006), the C-S paradigm is not properly a distributed processing paradigm: *“Incredibly, the terms “distributed systems” and “distributed computing” have been for years hijacked and (ab)used to describe very limited systems and low-level solutions (e.g., Client-Server) that have little to do with distributed computing”*. The C-S paradigm can be seen as a by-product of a set of protocols, functionalities and mechanisms that allow the distributed communication, interaction and coordinated processing. The C-S paradigm does not really contribute to the advancement of distributed processing, it just relies on those techniques. However, it is considered here for its profound impact and importance in the evolution of service platforms.

Services are built around interaction and control mechanisms. Users can access and take advantage of service functionalities by interacting with a system. For instance the C-S one is very simple but it requires the “client” to activate the service (pull) and wait for a response. Generally, it is not envisaged that a “server” will autonomously initiate an interaction toward the user (a push). This is obviously a limitation, services are designed in such a way to adhere to this interaction paradigm, and not all the services can be appropriately provided in a—C-S fashion. For instance, a simple service like “inform the client when a condition on certain data is met” (event notification) can be tricky to implement in a C-S fashion, but it could be simple to realize with another paradigm (e.g., a P2P implementation based on a message passing mechanism). The issue with the C-S paradigm is a well-known one [see for instance (Adler 1996): “Important examples include task allocation and event notification in collaborative workgroup systems, and task sequencing and routing in workflow applications”] and it has an impact especially when users/tasks need to coordinate several threads of processing and information distribution. Sometimes, in order to circumvent this issue, a continuous polling of the client on the server can be implemented, but this is not an optimized solution. Solutions like Websockets (Hickson 2011) can be seen as a way to solve this problem in the context of HTTP client server communication. Here a socket is created and the server can forward on the socket asynchronous information. In this way, the “difference” between the client and the server is reduced and this architecture can be seen as a move toward P2P solutions also for the Web. The other paradigm, the Network Intelligence one, is even more limited. It is based on the network capability to determine that the user is requesting service functionalities. The network “control” then triggers (i.e., it requests for assistance to a set of service layer functionalities) to the specific service. In this case, the service is confined to the network context in which the user is operating which is also confined by the

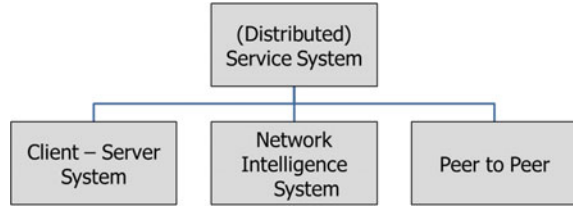
protocols and mechanisms of the network. The P2P paradigm seems to be the most adaptive, it supports a granular message-based (or event processing) approach that allows to design the service functions very effectively at a cost of representing a logical network of relationships between peers.

The paradigms have different “expressive power” (Minerva 2008a, b, c; Minerva et al. 2011), i.e., different capabilities to represent and support the control needs of a service. Actions and interactions within the parts of a service system can be represented and supported in different fashions and with different mechanisms, expressive power means here how easily the control paradigm can represent and support the needed communications and control of the components. Introducing functions to circumvent problems that could be easily solved with other paradigms could be a sign of low expressive power. Examples are the polling mechanisms for C-S systems compared to message passing systems or event-driven programming, or the over simplification of ParlayX APIs that do not allow the support of anonymity in services. An exemplification of the lack of expressive power in C-S paradigm is represented by WebSocket definition (Fette and Melnikov 2011): a new mechanism that extends the paradigm capabilities had to be introduced in order to allow the server side to autonomously notify the occurrence of events of partial data to the client side. Lack of expressive power can also yield to phenomena called Abstraction Inversion: users of a function need functionalities implemented within that are not exposed at its interface, i.e., the need to recreate functions that are available at lower levels. Abstraction Inversion, too much abstraction, and wrong control paradigm can lead to anti-patterns in intelligent systems (Laplante et al. 2007), with consequences in the ability of platforms to support services, increase in cost of software development, longer processes to delivery phase. Generally put, if a control paradigm requires too much adaptation and development of specific functions for representing and support the control flows which means that the paradigm is not well suited for such service.

The issue of expressive power is a fundamental one: the possibility to implement services with the best model for supporting the interactions between its parts should have a large importance when platforms and systems are designed, implemented, or simply bought. Instead it is often neglected and service offerings are created on the available platform totally disregarding how the final service will be provided. This has led (especially in the telecommunication world) to aberration in which highly interactive and cooperative services have been implemented with a very constraint paradigm (Minerva 2008b, 1) such as Network Intelligence [see for instance the attempt to implement Video On Demand (VoD), over IP Multimedia Subsystem (IMS) (Riede et al. 2008)]. The Interaction and Control paradigms are at the very heart of the service design and implementation, and this issue is one of the crucial considerations for determining how to create effective service platforms.

The considered control paradigms are depicted in Fig. 2.13 and are briefly discussed in this section.

Fig. 2.13 Control paradigms in the context of networked interactions



2.4.2.1 The C-S Control Paradigm

The C-S control paradigm is the simplest one; it is based on the concept that the network is essentially transparent to the functional interactions between the clients and the server. It clearly and cleverly exploits the end-to-end principle (Saltzer et al. 1984) of the Internet in order to opportunistically use the network capabilities. This principle states that *“mechanisms should not be enforced in the network if they can be deployed at end nodes, and that the core of the network should provide general services, not those tailored to specific applications”*. So the C-S paradigm is a means to move the intelligence at the edges of the network and confining the network to provide generic and reusable services related to transport. This has given rise to the concept of stupid network as proposed by (Isenberg 1998) Fig. 2.14 represents the concept of the stupid network.

As stated in Minerva et al. (2011), the C-S control paradigm has several merits

- Simple control pattern
- Implemented in stateless or stateful manner
- Decoupling of application functions from the network intricacies (service **deperimeterization**).

Its major drawback is that not always the network has an ideal behavior and there is often the need to orchestrate resources and capabilities in order to provide compelling services.

2.4.2.2 The Network Intelligence Control Paradigm

It is the typical control paradigm used by the telecoms industry to provide services. It leverages the capabilities of the network to provide services. Figure 2.15

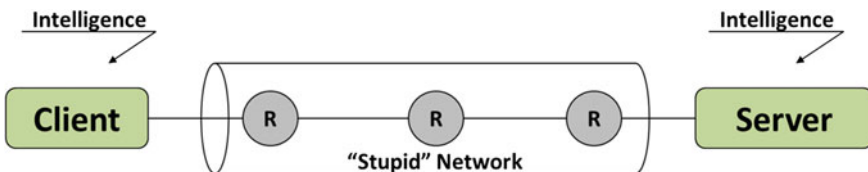


Fig. 2.14 The stupid network and the aggregation of intelligence at the edges

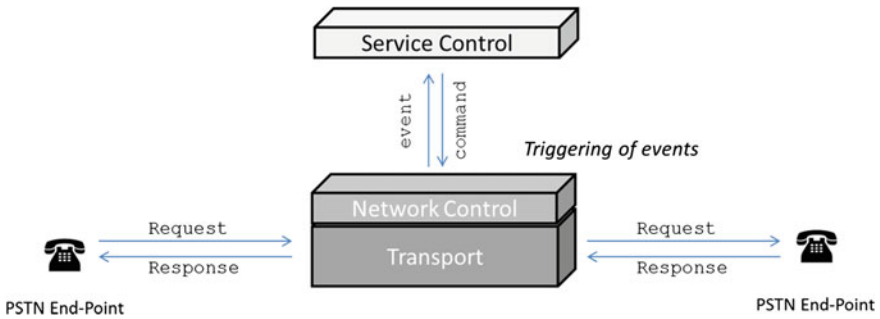


Fig. 2.15 Triggering in the NI paradigm

illustrates the basic mechanisms behind. In this case, the clients can interact with the control functions of the “network” and only when it is not possible to serve the request, i.e., when the service is “triggered”. There are two interaction models coupled in this control paradigm: a sort of C-S between the users of the “network service” and an event-driven model (Dabek et al. 2002) for the interaction between the control functions and the (added value) service functions.

The composition of interaction model is a bit complicated in this control paradigm. The caller (or the initial requestor) could be considered in a stateful C-S relation with the network. The network control triggers events toward a service control entity when it has no instructions on how to proceed with the service request. This relation is not Client-Server, because more (correlated) events can be sent by one party to the other. It is more an event-driven approach. Also the called or the destination of the communication is in an event-driven relationship with the “service system”. One evidence is that the network control is in charge of the orchestration of services. There is no way for the end points to request directly services and functions to the service control. This layer is mediated by the “network”. This approach was meaningful when the end points were stupid, but nowadays, terminals have plenty of intelligent capabilities and they could be capable of benefitting from a direct interaction with the service control and even from the direct communication with network resources. This dependence of the services from the network is a major drawback. In addition, the complexity of this model is even exacerbated by the fact that all the interactions are stateful and organized around very complex Finite State Machines (FSM) (Minerva 2004). The FSMs are associated to the concept of call and more recently session control model. Typically a service progresses along the states and the possibilities offered by a very complex and controlled model does not offer too much flexibility to the programmers (Minerva 2008b, 1), in addition the call and sessions are strongly coupled with network resource allocation. For instance in Fig. 2.16, yellow resources are allocated to a session.

Resources are made available only within a session in order to access another resource, the session control has to negotiate for it, while the end points have only the possibility to interact with the session control to request a mediated control of

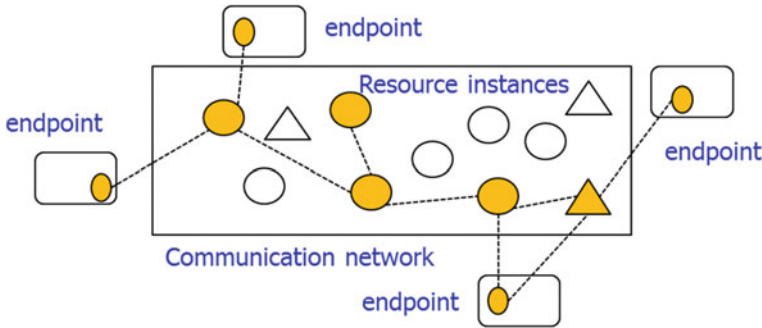


Fig. 2.16 The session concept and the allocation of resources

new resources. The possibility for endpoints (with increasing level of intelligence) of direct interaction with a specific resource is totally neglected by the Network Intelligence control paradigm. This leaves outside of the service control those elements that have the major requirements to do so.

This approach has been successful for the Intelligent Network (Faynberg et al. 1996), and it has been reiterated for many of the new control architectures for telecoms such as IP Multimedia Subsystem, IMS, (Camarillo and Garcia-Martin 2007), and Next Generation Networks in general (Copeland 2009). The IMS in particular has been considered as the new control platform to be used in order to offer the integration between telecoms and Internet services. This assumption is based on the following features offered by the IMS:

- Multi-access Networks, i.e., the capability to support several heterogeneous access networks
- Layering and decoupling of functions at the transport, control and service level
- Reuse of the successful general packet radio service (GPRS) functional architecture.

IMS is then seen as a reference architecture for Next Generation Networks. This approach has some pros and cons, such as

- Regulated and Standardized
- Support of interworking
- Constrained and limited
 - New add-ons are difficult to introduce in the software
 - Stateful model
 - Session control is just an evolution of the call control
- Complexity increases with number of new functional entities and states (for instance the presence of different types of Call State Control Functions (CSCF), gives rise to the need of several protocol interaction in order to sort out the Functions orchestrating the service).

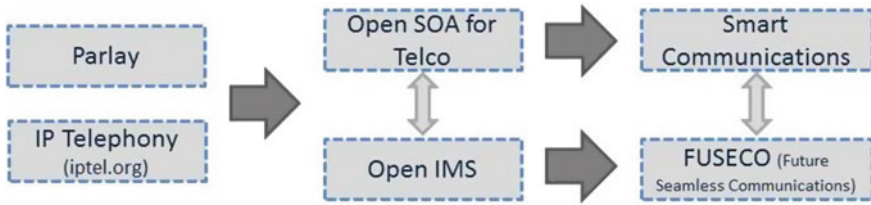


Fig. 2.17 The NGNI playground evolution

This is a very traditional approach that disregards the issues in interaction and control paradigms, the evolution of terminals and edge functionalities, the current development of the major projects in the Future Internet such as GENI (Peterson and Wroclawski 2007), AKARI (Aoyama 2009) and others (Pan et al. 2011). These new architectures consider different interaction and control paradigms (highly distribution of functions, P2P, mesh networking) that greatly differ from traditional ones. For a good overview of Network Intelligence, the interested reader could refer to the ICIN conference www.icin.co.uk that collects many contributions in this field. For a traditional approach and its mainstream developments in the exploitation of Network Intelligence, good sources are (NGNI Group 2005, 2006, 2009, 2012). Figure 2.17 represents a (desired) mainstream evolution of technologies and system from a telecoms perspective.

Summarizing, the pros and cons of the NI approach are

- Mix of different mechanisms (C-S and event—commands)
- Generally implemented in a stateful manner
- Strong dependence of application functions from networks ones (e.g., the triggering mechanism is the way in which services can be activated, creating a strong dependence of services on the status of the network).

Some other major concerns in the NI approach are: the **perimeterization** of services (i.e., services are tightly coupled to the network functionalities and its geographical extension) and the assumption that Quality of Service (QoS) is a major requirement for service. The definition of the IMS architecture can be considered as an example of the importance given by the telecommunication industry to the introduction of dynamic allocation mechanisms for better controlling the bandwidth allocation, and related QoS parameters. There is a continuous strive to find out models to introduce in the network these kinds of mechanisms (Song et al. 2007). They have been discussed and criticize in Minerva and Bell (2010) and Minerva et al. (2011) showing how operators cannot compete at a global level because their services cannot be provided outside of the deployed and owned network and arguing that QoS and more bandwidth are not always practical solutions for solving service issues (in fact, Google solved the problem of long map

downloading time by introducing AJAX solutions based on asynchronous download of data³).

2.4.2.3 The Peer-to-Peer Control Paradigm

Apart from the different topologies and characterization of structured, unstructured, and hybrid peer-to-peer networks (Lua et al. 2005; Merwe et al. 2007), two concepts are central to P2P networks: equality of peers, i.e., each node is potentially a client and a server of the network; overlaying and virtualization, i.e., the creation of a logical virtual network on top of physical resources. Another important aspect is the dynamicity of the network: nodes leave and join dynamically and the average uptime of individual nodes is relatively low. The topology of an overlay network may change all the time. Once a route is established, there is no guarantee of the length of time that it will be valid. Routing in these networks is therefore very problematic. In other terms, the P2P control paradigm is complicated by the need to manage and keep track of a highly dynamic infrastructure that changes over time. On the other side, the interaction between peers could be implemented adopting the most appropriate (for the specific service) interaction mechanism. So a P2P infrastructure could be able to encompass several interaction paradigms offering to services the most suitable mechanisms.

P2P network Systems are characterized by

- Complexity of the entire system and simplicity of each node
 - Nodes are both clients and servers
 - P2P systems are real distributed systems, they need to govern an overlay network (resource naming, addressing, location)
- Scalability and Robustness
 - P2P systems can scale to millions of nodes
 - Problems in detecting the source of issues in case of malfunctions
- Different way of programming
 - P2P systems can be implemented as event-driven system but also the communication between two terminal nodes can be client-server
- Social flavor of the solutions (Koskela et al. 2013). The willingness to share and the thresholds at which an altruistic behavior is triggered have been studied

³Actually this is another evidence that the expressive power of the C-S paradigm is not sufficient to provide services that do require responsive control on the flow of data. Introducing asynchronous (and invisible to the users) calls between the client and the server is pointing to the need for more capable control mechanisms. Actually those mechanisms are also provided in WebSocket that definitely change the C-S paradigm (in fact the server can use now the socket to notify events and messages to the client).

(Ohtsuki et al. 2006). They are particularly meaningful to understand when a P2P network will become valuable to a single user.

Summarizing, the P2P approach seems to be very flexible and promising; some pros and cons are

- Able to support many control patterns
- Generally implemented in a stateful manner (the overlay control)
- Intertwining of application functions with network ones.

2.5 Service Architectures

The interaction and control mechanisms have to be framed and supported by software architectures. In this context, a software architecture consists of a set of concepts, principles, rules, and guidelines for constructing, deploying, operating, and withdrawing services. It also describes the environment in which such services operate. The service architecture identifies components to build services, describe the way they are combined, and the way they interact. The combination of organization principles of the architecture, the underlying technology platform and the enabled interaction and control paradigms determine the expressive power of the service architectures and its capabilities in supporting several classes of services. The expressive power is the capability of services and application to represent the real world interactions and to organize the available functions in such a way to accommodate the applications and users needs.

There are some major attempts to define software architectures (Proper 2010; Lankhorst 2013) and their expressive power with respect to the problem domain representation and the involved stakeholders.

2.5.1 *The Web and IT Views on Service Architectures*

The evolution trajectory of Web and IT service architecture is defined by different factors such as

- Evolution of web servers toward application server architectures (Minch 2009);
- Evolution of Web Service (Booth et al. 2004) and Service-Oriented Architectures (Partridge and Bailey 2010) and the competition between SOAP and REST based solutions;
- The development of advanced solutions within data centers that have led to Cloud Computing (Gong et al. 2012) opposed to Grid Computing (Foster et al. 2008; Sadashiv and Kumar 2011);

- Virtualization of resources (Chowdhury and Boutaba 2010; Sahoo et al. 2010) and the Software as a Service models (Prodan and Ostermann 2009).

An attempt to standardize the mechanisms behind the simple XML_RPC approach has been done with [Simple Object Access Protocol (SOAP) (Box et al. 2000)]. SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML. SOAP consists of three parts

- The SOAP envelope defines an overall framework for expressing what is in a message, who should deal with it, and whether it is optional or mandatory.
- The SOAP encoding rules define a serialization mechanism that can be used to exchange instances of application-defined data types.
- The SOAP RPC representation defines a convention that can be used to represent remote procedure calls and responses.

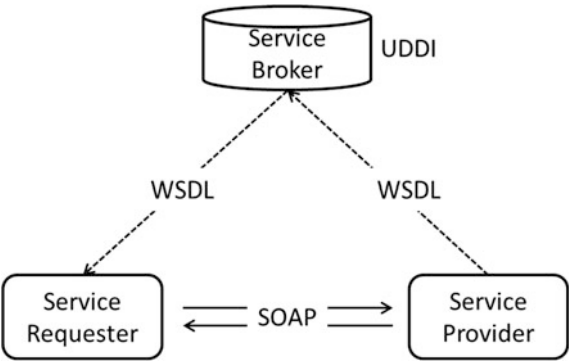
The definition of SOAP was instrumental to the growth of the Web Services Architecture. Web services provide a systematic and extensible framework for application-to-application interaction, built on top of the existing Web protocols and based on open XML standards (Curbera et al. 2002). The Web services framework is divided into three areas: communication protocols, service descriptions, and service discovery. They correspond to three protocol specifications that are the cornerstones of the Web Services Architecture (Curbera et al. 2005):

- the mentioned simple object access protocol which supports communication between Web services;
- the Web Services Description Language (WSDL) (Chinnici et al. 2007), which specifies a formal, computer-readable description of Web services;
- the Universal Description, Discovery, and Integration (UDDI) (Clement et al. 2004) directory, which is a registry of Web services descriptions.

These protocols form the skeleton of the well-known Web Service Architecture represented in Fig. 2.18. The Service Provider uses the UDDI functions in order to register and advertise its services on the Universal Registry. A service requestor can access the UDDI functionality and search for the most suitable service for its needs. It gets a WDSL description that can be used in order to correctly access the service. At this point, the service requestor and the service provider can be linked and can use the SOAP Protocol to access and provide the service functions.

The specification has been extensive and the original simplicity of the solution has led to a complex set of documents and to overall complex implementations. Essentially the UDDI service, originally intended to be used in a dynamic way, is used off-line in order to keep track of the software components made available. The SOAP protocol itself has been disregarded in favor of the simplest solutions based on the REST approach (Fielding and Taylor 2002). REST is a set of architectural principles for designing and implementing distributed systems. It proposes

Fig. 2.18 The web services architecture



the use of web-based interfaces that are described in XML and handled through HTTP. The REST principles are:

- A stateless client–server protocol
- A set of well-defined operations (e.g., GET, POST, PUT, and DELETE) to perform functions on resources
- A universal syntax for resource identification (resources are identified by URIs)
- Use of XML or HTML for describing pieces of information and their links.

REST is at the core of Web Application Development as shown in Fig. 2.19. This figure shows that whenever a simple solution is offered and it is based on well-known mechanisms then the programmers will adopt it. This is the case of REST: it is based on the established principles of web programming (each function is an addressable resource and the mechanisms to interact are those supported by the HTTP) and it is simpler than other protocol and related solutions (like SOAP). Web programmers use it much more than other solutions.

The Web Service Architecture has mingled with the definition of service-oriented architectures (SOA) (Sward and Boleng 2012; SOA-Wikipedia

Fig. 2.19 Protocol usage by APIs. Source <http://programmableweb.com>

Protocol Usage by APIs	Percentage
REST	63 %
SOAP	22%
JavaScript	7%
Atom	3%
XML-RPC	3%
Others	2%

2013). Service-oriented architecture is a software architectural concept that defines how to use services to support the requirements of software users. In a SOA environment, nodes on a network make resources available to other participants in the network as independent services that the participants access in a standardized way.

Most definitions of SOA identify the use of Web services (i.e., using SOAP or REST) in their implementation. However, SOA can be implemented using any service-based technology.

Unlike traditional object-oriented architectures, SOAs comprise loosely coupled (joined), highly interoperable application services. Because these services interoperate over different development technologies (such as Java and .NET), the software components become very reusable, due to the virtue of the interface definition being defined in a standards compliant manner (WSDL) which encapsulates/hides the vendor/language specific implementation from the calling client/service. The relations between SOA and Web Services were first of competition and then of integration, nowadays SOAs can be implemented as Web Service Architectures.

The availability within data center of storage and processing capabilities has led to the development of Cloud Computing, i.e., “a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet” (Foster et al. 2008). This concept seems to be very close to the grid computing. Very similar in scope, cloud computing and grid computing differ in at least two features that are important for this document

- The standardization and openness of the approach
- The control and optimization of the underlying resources with special emphasis on the network ones.

The grid, in fact, has pursued a standardization of its solution from its incipit and it has considered resources with holistic and broad view. Figure 2.20 represents the

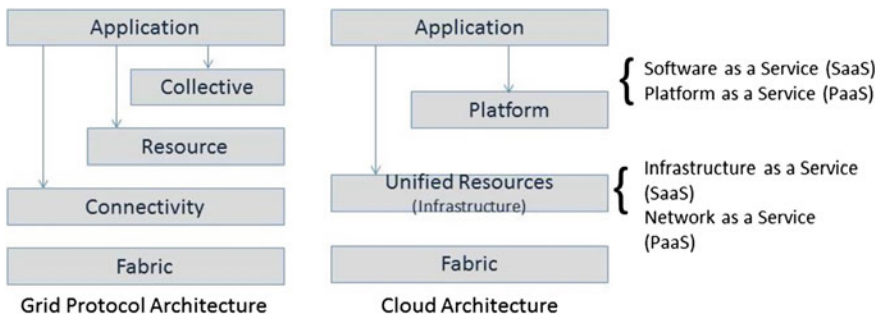


Fig. 2.20 Grid and cloud computing architectures

two different approaches: the grid aims at controlling resources and groups of them also deployed in different administrative domains, while the cloud (at least originally) was aiming to homogeneous and single administrative domains. In addition, the grid is aiming at granular control on the resources and it also fosters the control on connectivity as one major goal.

Only recently, cloud computing (and the Web Companies behind it) have started to work with the goal to achieve a better control of connectivity. For instance Google has unveiled their effort to build a large-scale controllable network (Vahdat 2012).

Contextually to the development of cloud computing, virtualization technologies have made such progresses to become a major technology trend in several application domains. Resource virtualization is the possibility to create a virtual machine that acts and behave like real resources on top of hardware and software resources. Software executed on these virtual machines is separated from the underlying hardware resources. The benefits reside on the possibility to decouple even further the software infrastructure from the underlying hardware and to segment on a cluster of machines different applications and resources limiting the occurrence that the fault of a (virtual) machine could impact on other machines and resources (Metzler 2011). Virtualization was instrumental to the possibility to introduce cloud computing (Lenk et al. 2009) and the Software as a Service (Turner et al. 2003) model to a large audience. Its extension has led to the new trend of virtualizing everything as a service [XaaS, (Schaffer 2009)] and to offer these features by means of a cloud infrastructure.

Summarizing, the Web and IT world is aiming at very large structures that can virtualize processing, storage resources, and can offer compelling services developed and organized in reusable building block. Resources are addressable with web mechanisms and they offer APIs that can be mashed up and combined in order to provide new services and applications. In spite of standardization, these solutions aim at proprietary environments in order to capitalize a technology investment and advantage.

2.5.2 The Network Intelligence View on Service Architectures

The evolution of service architecture in a Network Intelligence context is characterized by a two competing approaches and visions

- the traditional Telco one that can be seen as an evolutionary path from the ISDN, IN, and IMS architectures;
- an innovative one that stems from open platforms and in particular TINA and goes through Parlay and the service delivery platform in order to leverage Network APIs.

The two approaches have often been competing for driving the evolution of the Next Generation Networks, NGN, but nowadays they stick together because they have become rather traditional and generally obsolete. The windows of opportunity for opening up network interfaces, building a new programmable architecture and offering new services was around 2000–2002 (along the initial deliveries of Web 2.0 platforms). After that period, the window of opportunity closed as discussed in several papers that have identified the decline of the network intelligence (Minerva et al. 1998; Minerva and Moiso 2000; Licciardi et al. 2000, 2004; Manzalini et al. 2009).

The current attempts to repositioning the Operators in the Web 2.0 wave (Maes 2010) are probably useless and are consuming resources that should be spent for other efforts (see for instance Telco as a Platform Enabler and Disappearing Telco, to be discussed in Minerva and Crespi (2016)).

In the Telecommunication world, a lot of effort (in terms of joint projects, research activities and association of several enterprises) has been spent on the definition and demonstration of viable programmable platforms. A few contributions touching relevant issues posed by the book with particular emphasis on new telecom architectures are listed below:

- Architectural definitions for TINA (Berndt and Minerva 1995; Berndt et al. 1999; Yagi et al. 1995), Parlay (Walkden et al. 2002), Open Service Access (OSA) (Moerdijk and Klostermann 2003) and Service Delivery Platform, SDP (Morian Group 2004; Baglietto et al. 2012);
- Definition of Open Interfaces like Parlay (Di Caprio and Moiso 2003), ParlayX (Parlay Group 2006) and OneAPI (GSM Association 2010);
- Definition of the IMS architecture (Knightson et al. 2005) and usage of the SIP protocol (Johnston 2009) as building blocks for a capable NGN (TISPAN 2009);
- Integration of IMS and SDP (Pavlovski 2007).

A service delivery platform (or SDP) is a centralized hub for the creation and integration of all applications and services offered within a network. It is implemented by or on behalf of a network operator, and resides in the “IT” part of the operator’s infrastructure.

To avoid standardizing services and facilitate the possibility of a differentiation between operators in terms of something other than pricing, the OMA, and also the ETSI and ITU-T, opted to standardize enablers, called service capabilities by the 3GPP, service support capabilities by the ITU-T and service enablers by the OMA. The SDP allows applications to be abstracted from bearers, channels, enablers, and operational support systems. It goes hand-in-hand with an architectural framework, which describes how it interfaces with the other systems in the operator’s infrastructure, including external system. An SDP is used in order to “expose” APIs. They can be opened also to third parties. The reason for adopting the SDP is on the Telco’s infrastructure and the possibility to monetize the service exposure, i.e., the offering of APIs, also to third party developers. The SDP is filling a gap in the

definition of the IMS at the service layer. IMS service layer is essentially a collection of different application servers (SIP, OSA, CAMEL) not operating in a synergistic way. The goal of SDP is to integrate the different functions of these servers with the management infrastructure of the Telco. To simplify the reasoning behind the equation $NGN = IMS + SDP$ should represent the evolutionary strategy of operators.

2.5.3 The Peer-to-Peer View on Service Architectures

Peer-to-peer systems have the goal to optimize the usage of peer resources in order to meet the requirements of a community of users. For instance the Invisible Internet Project (I2P) aims at making the communication between peers secure and anonymized. In order to achieve this goal, the overlay logical network is organized around a few building blocks that can be combined to create an infrastructure on top of which services and applications can be executed guaranteeing anonymity to users (Aked 2011). Any peer will try to create encrypted tunnels toward the peers it wants to communicate with. Tunnels will be outbound and inbound. The peers needed for creating tunnels are selected by means of a distributed Network database. When the communication has to be created, a lease will be issued so that the inbound and inbound tunnels can be connected and the end parties can communicate. Figure 2.21 illustrates the mechanisms.

Once the network between peers has been constructed, common applications can be launched and used by taking advantage of the new infrastructure.

Another relevant P2P architecture is JXTA (Gradecki 2002), whose goal is to support the creation and the usage of large distributed application using the functionalities offered by specialized peers. JXTA decouples basic functions from services and different applications. At the lower levels, mechanisms for creating pipes

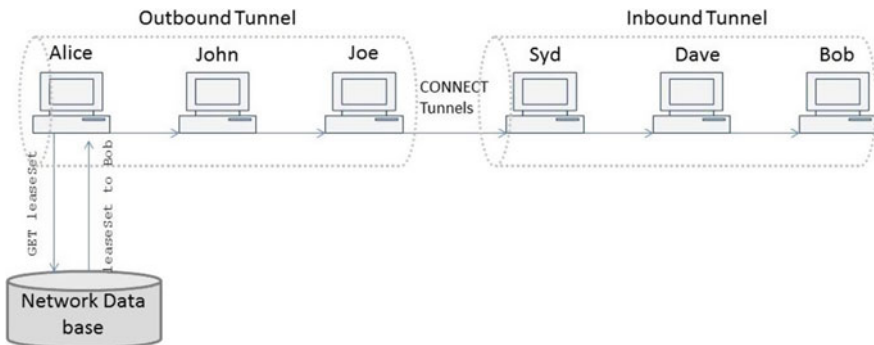


Fig. 2.21 Invisible internet project, I2P

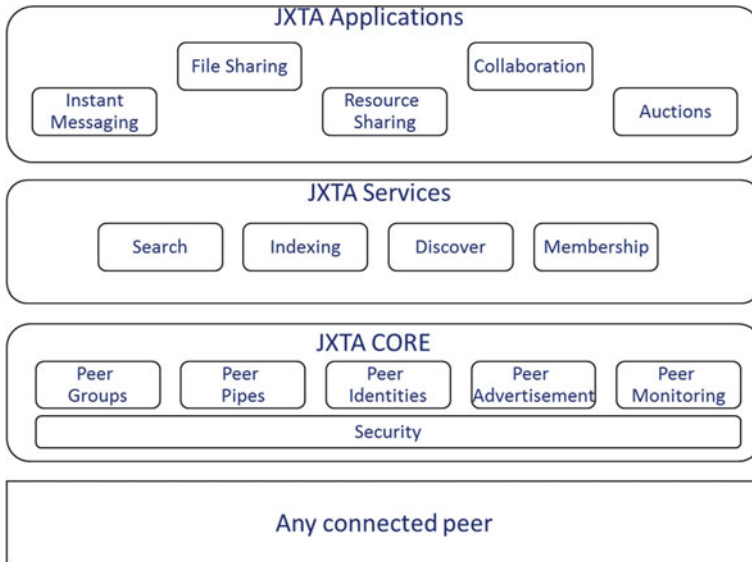


Fig. 2.22 JXTA architecture (Gradecki 2002)

between peers, grouping them, and identifying them are provided. On a higher level, services for discovery of peers, determining the group associated to a specific peer and others are provided. At the upper level, applications are created and used exploiting the JXTA functionalities. Figure 2.22 depicts the architecture.

In spite of their wide use, there are not well established and standardized P2P Platforms, However a few considerations about P2P architectures can be drawn from the previous discussion about architectures

- Complexity of the entire system and simplicity of the single node
 - While nodes are both clients and servers and provide a simple behavior, their aggregation is posing serious issues in terms of management and operation.
- P2P systems are real distributed systems,
 - they need to govern an overlay network (resource naming, addressing, location)
 - the overlay network is totally decoupled from the underlying physical infrastructure. This can introduce inefficiency in how network resources are used.
- Scalability and Robustness
 - P2P systems can scale to millions of nodes
 - Problems in detecting the source of issues in case of malfunctions

- Different ways of programming
 - P2P systems can be implemented as event-driven system but also the communication between two terminal nodes can be client-server.

An important property of P2P systems is their power (Delaney et al. 2001; Manzalini et al. 2010), and in particular their scalability (Hu and Liao 2004; Krishnamurthy et al. 2001), their availability (Bhagwan et al. 2003) and other properties such as stability (Pouwelse et al. 2005). Peer-to-peer networks offer the possibility to aggregate computing, communication and storage capabilities made available by participating peers and to scale up to very large systems. In comparison, a data center requires a lot of investments and planning in order to achieve the desired/needed capacity and scalability.

The total power of a data center is fixed and is given by the number of servers, and related processing and storage capabilities as well as the total bandwidth associated to the data center, in other terms the “power” can be indicated as power (client-server system) = $\{b_S, s_S, f_S, p_S\}$

where

b_S bandwidth of the server System

s_S storage of the server system

p_S processing in the server system

In a P2P system, the total power is given by the aggregation of individual contributions of peers, so it grows with the number of participants, in a way the “power” could be indicated as Power (P2P System) = $\sum (b_i, s_i, p_i)$

where

b_i bandwidth of node i

s_i storage of node i

p_i processing of node i.

In principle, the aggregated power of a P2P system can grow and compare to large data center when the scale of the system comprises several millions of machines and nodes. However, there are some differences to consider: heterogeneity of the machines, while in a data center there is a high degree of homogeneity (machines are the same) in a P2P system, the aggregating nodes could be very different from each other and also in similar machine the resources (processing, storage, and communications capabilities) could be shared in different ways; the availability of nodes over time, actually each single node can be turned off or leave the P2P network anytime giving a much higher degree of instability to the network. The issue of optimization of using the underlying physical resources has been tackled (V. Gurbani, V. Hilt and I. Rimal) by defining simple protocols for the allocation of resources. Virtualization and self-organization could be other meaningful ways to lessen the problem (Moiso et al. 2010; Manzalini et al. 2010).

A Taxonomy of services and application of P2P (Minerva 2008a, b, c) is depicted in Fig. 2.23.

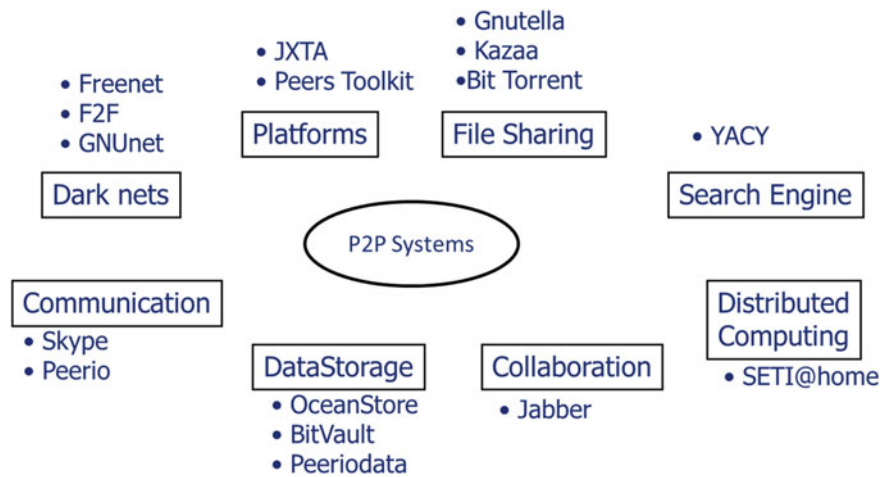


Fig. 2.23 A taxonomy of current P2P services

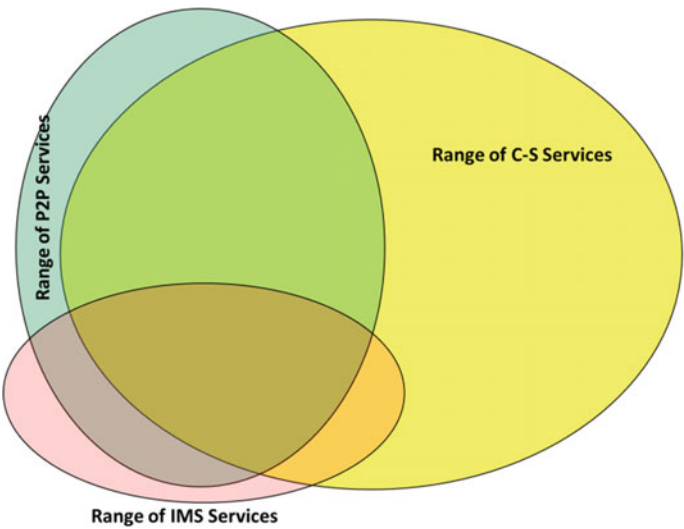


Fig. 2.24 The service span of the considered control paradigm and architectures

The current coverage of services with the three analyzed control paradigm is quite different as depicted in Fig. 2.24. However, P2P seems to be usable for providing services in a widely distributed manner and its potential capabilities are similar to the C-S paradigm, especially if a set of standardized mechanisms could be generally implemented (e.g., distributed hash tables). The value of P2P paradigm is

also related to its capability to de-perimeterize services⁴ and to disrupt the current status quo. Operators could try to benefit to revert a critical situation in the realm of service offering.

2.6 Findings

This chapter has introduced some of the approaches that are undertaken by major Actors in order to provide services. The distinctive standpoints of C-S, NI, and P2P approaches will be used to identify and discuss some possible scenarios that could occur in the future. It is very significant from a business and scientific perspectives, the increasing interest that is growing around services. The definition of a “service science” comprising several application areas is important because it is paving the way to more precise mechanisms, methodologies, and systems that can fulfill the service requirements. Under this perspective, it should be noted that the user interests and rights are still not fully considered: in fact, many “user-centered” approaches try to understand how to better offer service to the user, and not in guaranteeing to users fair services that deliver value for the right price. This is a deficiency that will be emphasized in the entire document. Chapter 6 will deal with this important aspect of services: i.e., Business Models and the way they are used by Actors of the ICT industry to exploit and take advantage of customers and users. In addition in this chapter, an essential concept related to services is appeared: the more and more proactive role of users. This is a changing factor in the industry because users can play a more relevant role in delivery services and this can have deep social and economic impacts.

Currently, different competing Stakeholders are using Services and Business Models in order to alter, increase or consolidate their positions in the value chain or to change the “status quo” yielding to new opportunities. Examples are Voice over IP (VoIP) or text messaging services like Whatsapp that are used by WebCos for acquiring new users while at the same time disrupt a consolidated market for Telcos. In addition a user-based model could lead to disruptive scenarios in which traditional Stakeholders can be bypassed in favor of service and network infrastructures totally based on user provided resources. The current trend of Fog Computing or Edge Computing (Shanhe et al. 2015) is a step toward these possibilities. Interaction and control paradigms have been introduced; they are important because they affect the way, the services are implemented and provided. By choosing a control paradigm “a priori” without considering the service, the users and their needs can push toward cumbersome solutions instead of focusing clearly

⁴Services in a P2P system can be implemented wherever there is processing and storage power (provided by users), individual peers provide also connectivity that is usable independently from the Operator that is physically supporting it. Services can be deployed and provided by making use of resources that are not necessarily attached to the specific TelCo Network; actually they can be geographical distributed.

on solving problems related to the service delivery. In this way, they are tackling the issue of how a platform supporting a specific paradigm can be used to deliver a specific service. This lack of flexibility in service design and implementation is a major drawback of many platforms that do impose very stringent control means on services and applications. As seen, Network Intelligence is especially prone to this problem because of the tight coupling of network and service functionalities. The C-S and the P2P paradigms are more flexible and can support a variety of possible interaction and control mechanisms reducing this dependency. The expressiveness of a paradigm is of paramount importance in order to create efficiently and effectively new services. Section 2.5 has discussed how the expressive power of a paradigm and its supporting infrastructure has a direct consequence on the possibility to create a large number of interesting services. This leads to the issue of service architectures. In this chapter, the basic service architecture approaches have been considered showing that the C-S one has currently an advantage, while the P2P is more complicated, but it could support in a very flexible and open way many services. The NI paradigm and related architectures seem the one that is less flexible and reusable for providing new services. In the following chapters, the issues related to these architectures will be further elaborated.

Bibliography

- Adler RM (1996) Distributed coordination models for client/server computing. *IEEE Comput* 28(4):14–22
- Aked S (2011) An investigation into darknets and the content available via anonymous peer-to-peer file sharing. In: 9th Australian information security management conference. Edith Cowan University, Perth, Australia
- Aoyama T (2009) A new generation network: beyond the internet and NGN. *IEEE Commun Mag* 47(5)
- API, wikipedia (2013) Application programming interface. Available at https://en.wikipedia.org/wiki/Application_programming_interface. Wikipedia, last accessed may 2013
- Baglietto, P, Maresca, M, Stecca M, Moiso C (2012) Towards a CAPEX-free service delivery platform. In: 16th international conference on intelligence in next generation networks (ICIN). IEEE, Berlin
- Berndt H, Darmois E, Dupuy F, Inoue Y, Lapierre M, Minerva R, Minetti R, Mossotto C, Mulder H, Natarajan N et al (1999) The TINA book: a co-operative solution for a competitive world. In: Inoue Y, Lapierre M, Mossotto C (eds) Prentice Hall
- Berndt H, Minerva R (1995) Definition of service architecture. Deliverable. TINA Consortium, Red Bank
- Bertin E, Crespi N (2013) Architecture and governance for communication services. Wiley-ISTE, London
- Bhagwan R, Savage S, Voelker GM (2003) Understanding availability. *Peer-to-peer systems II*. Springer, pp 256–267
- Booth D et al (2004) Web services architecture. W3C working group note, W3C, W3C
- Box D et al (2000) Simple object access protocol (SOAP) 1.1. Standard, W3C
- Cachin C, Guerraoui R, Rodrigues L (2011) Introduction to reliable and secure distributed programming. Springer, Berlin

- Cardoso J, Konrad V, Matthias W (2009) Service engineering for the internet of services. In: Joaquim F, José C (eds) Enterprise information systems - lecture notes in business information processing, vol 19. Springer Lecture Notes in Business Information Processing, Berlin Heidelberg, pp 15–27
- Camarillo G, Garcia-Martin MA (2007) The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds, John Wiley and Sons, New York
- Chinnici R, Moreau JJ, Ryman A, Weerawarana S (2007) Web services description language (wsdl) version 2.0 part 1: Core language. Recommendation, Boston - Geneva: W3C
- Chowdhury NM, Boutaba R (2010) A survey of network virtualization. *Comput Netw* (Elsevier) 54(5):862–876
- Chung PE et al (1998) DCOM and CORBA side by side, step by step, and layer by layer. *C++ Rep* 10(1):18–29
- Clement, L, Hatley A, Rogers C, von Riegen T et al (2004) UDDI version 3.0. 2. Specification technical committee draft, UDDI Org
- Copeland R (2009) Converging NGN wireline and mobile 3G networks with IMS. CRC Press, London
- Curbera F, Leymann F, Storey T, Ferguson D, Weerawarana S (2005) Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more. Prentice Hall PTR, Englewood Cliffs
- Curbera F, Duftler M, Khalaf R, Nagy W, M N, Weerawarana S (2002) Unraveling the web services web: an introduction to SOAP, WSDL, and UDDI. *IEEE Internet Comput* 6(2):86–93
- Dabek F, Zeldovich N, Kaashoek F, Mazières D, Morris R (2002) Event-driven programming for robust software. In: Proceedings of the 10th workshop on ACM SIGOPS European workshop. ACM, Saint-Emilion, France, pp 186–189
- Datta D et al (2012) Wireless distributed computing: a survey of research challenges. *IEEE Commun Mag* (ComSoc) 50(1):144–152
- Delaney B, Catarci T, Little TDC (2001) The power of P2P. *IEEE Multimedia* 8(2):100–103
- Deutsch, P (1995) Fallacies of distributed computing. White Paper, wikipedia
- Di Caprio G, Moiso C (2003) Web services and parlay: an architectural comparison. In: Proceedings of ICIN. ICIN, Bordeaux, France, pp 1–6
- Faynberg I, Shah NJ, Gabuzda LR, Kaplan MP (1996) The intelligent network standards: their application to services. McGraw-Hill Professional, New York
- Fette I, Melnikov A (2011) The websocket protocol. RFC 6455, W3C, Boston, Geneva
- Fielding RT, Taylor RN (2002) Principled design of the modern web architecture. *ACM Trans Internet Technol (TOIT)* (ACM) 2(2):115–150
- Foster I (1995) Designing and building parallel programs. Addison-Wesley Reading, Boston
- Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud computing and grid computing 360-degree compared. Grid computing environments workshop. IEEE, Austin, TX, USA, pp 1–10
- Gong Y, Ying, Z, Lin M (2012) A survey of cloud computing. In: Proceedings of the 2nd international conference on green communications and networks 2012 (GCN 2012). Springer, Gandia, Spain, pp 79–84
- Gradecki JD (2002) Mastering JXTA: building java peer-to-peer applications. Wiley, New York
- Grelck C, Scholz SB, Shafarenko A (2010) Asynchronous stream processing with S-Net. *Int J Parallel Prog* (Springer) 38(1):38–67
- GSM Association (2010). Home-3rd party access project-OneAPI. White Paper, GSM Association, London
- He W, Xu L (2012) Integration of distributed enterprise applications: a survey. *IEEE Trans Industr Inf* 99:1
- Hickson I (2011) The websocket api. Working draft WD—websockets. W3C, Boston, Geneva
- Hu SY, Liao GM (2004) Scalable peer-to-peer networked virtual environment. In Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games. ACM, Portland, OR, USA, pp 129–133
- Isenberg DI (1998) The dawn of the “stupid network”. *netWorker*, pp 24–31

- ITU (2011) ITU and its activities related to internet protocol (IP) networks. <http://www.itu.int>. In: ITU (ed). 4 Apr 2011. <http://www.itu.int/osg/spu/ip/glossary.html>. Accessed 26 May 2013
- Johnston AB (2009) SIP: understanding the session initiation protocol. Artech House Publishers, London
- Jones S (2005) Toward an acceptable definition of service [service-oriented architecture]. *IEEE Softw* 22(3):87–93
- Knightson K, Morita N, Towle T (2005) NGN architecture: generic principles, functional architecture, and implementation. *IEEE Commun Mag* 43(10):49–56
- Koskela T, Kassinen O, Harjula E, Ylianttila M (2013) P2P group management systems: a conceptual analysis. *ACM Comput Surv* 45(2): 20, 25
- Krakiwiak S (2003) What is middleware. www.objectweb.com. <http://middleware.objectweb.org/>. Accessed 26 May 2013
- Krieger D, Adler RM (1998) The emergence of distributed component platforms. *IEEE Comput* 31(1):43–53
- Krishnamurthy B, Wang J, Xie Y (2001) Early measurements of a cluster-based architecture for P2P systems. In: Proceedings of the 1st ACM SIGCOMM workshop on internet measurement. ACM, Burlingame, CA, USA, pp 105–109
- Kubiatowicz JD (1998) Integrated shared-memory and message-passing communication in the alewife multiprocessor. PhD Thesis, MIT, Boston
- Kwon YW, Tilevich E, Cook WR (2011) Which middleware platform should you choose for your next remote service? *Serv Oriented Comput Appl (Springer)* 5(2): 61–70
- Kwon YW, Tilevich E, Cook WR (2010) An assessment of middleware platforms for accessing remote services. In: IEEE international conference on services computing (SCC). IEEE, Miami, FL, USA, pp 482–489
- Lankhorst M (2013) Enterprise architecture at work: modelling, communication and analysis. Springer, Berlin
- Laplanche P, Hoffman RR, Klein G (2007) Antipatterns in the creation of intelligent systems. *IEEE Intell Syst* 22(1): 91–95
- Laurent SS, Johnston J, Dumbill E, Winer D (2001) Programming web services with XML-RPC. O'Reilly Media, Incorporated, Sebastopol, CA, USA
- Lenk A, Klems M, Nimis J, Tai S, Sandholm T (2009) What's inside the Cloud? An architectural map of the Cloud landscape. In: Proceedings of the 2009 ICSE workshop on software engineering challenges of cloud computing. IEEE, Vancouver, Canada, pp 23–31
- Leopold C (2001) Parallel and distributed computing: a survey of models, paradigms, and approaches. Wiley, New York
- Licciardi CA, Minerva R, Cuda A (2000) TINA is dead, long live TINA: toward programmable solutions for next generation services. In: TINA conference. TINA_C, Paris, pp 16–20
- Lua EK, Crowcroft J, Pias M, Sharma R, Lim S (2005) A survey and comparison of peer-to-peer overlay network schemes. *IEEE Commun Surv Tutorials* 7(2):72–93
- Maes SH (2010) Next generation telco service providers: Telco 2.0 and beyond. Huawei White Paper, Huawei
- Magic, instructional media +. (2012) Application programming interface. <http://www.immagic.com/>. <http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/WIKIPEDI/W120623A.pdf>. Accessed 26 May 2013
- Maly RJ, Mischke J, Kurtansky P, Stiller B (2003) Comparison of centralized (client-server) and decentralized (peer-to-peer) Networking. Semester thesis, ETH Zurich, Zurich, Switzerland, pp 1–12
- Manzalini A, et al (2010) Self-optimized cognitive network of networks. Future Network and Mobile Summit 2010. IEEE, Florence, Italy, pp 1–6
- Manzalini A, Minerva R, Moiso C (2010) Exploiting P2P solutions in telecommunication service delivery platforms. In: Antonopoulos N, Exarchakos G, Li M, Liotta A (eds) Handbook of research on P2P and grid systems for service-oriented computing: models, methodologies and applications. Information Science Reference, Hershey, PA, pp 937–955

- Manzalini A, Minerva R, Moiso C (2009) If the web is the platform, then what is the SDP? ICIN. IEEE, Bordeaux, pp 1–6
- Margara A, Cugola G (2011) Processing flows of information: from data stream to complex event processing. Proceedings of the 5th ACM international conference on distributed event-based system. ACM, New York, pp 359–360
- Merwe JVD, Dawoud D, McDonald S (2007) A survey on peer-to-peer key management for mobile ad hoc networks. ACM Comput Surv (CSUR) 39(1). Article n. 1
- Metzler J (2011) Virtualization: benefits, challenges, and solutions. White Paper, Riverbed Technology, San Francisco
- Minch R (2009) Oracle's e-business suite: an N-tier, networked application. ITM305-003. Boise State University, Boise City, ID, USA
- Minerva R (2008a) On some myths about network intelligence. In: Proceedings of international conference on intelligence in networks-ICIN2008. ICIN, Bordeaux, France, pp 1–6
- Minerva R (2008b) On the art of creating services: do different paradigms lead to different services? J Telecommun Manag 1:33–45 Henry Stewart Publications
- Minerva R (2008c) On the importance of numbers and names for a 4G service architecture. In: Annual review of wireless communication, vol 3. IEC
- Minerva R (2004) The death of network intelligence? In: Proceedings of international symposium on services and local access 2004 (ISSLS). Edinburgh
- Minerva R, Bell S (2010) Boundary blurring between telecom and the internet. Connect World
- Minerva R, Moiso C (2000) Will the “circuits to packets” revolution pave the way to the “protocols to APIs” revolution? CSELT Techn Rep 28(2):213–226
- Minerva R, Manzalini A, Moiso C (2011) Towards an expressive, adaptive and resource aware network platform. In: Prasad A, Buford J, Gurbani V (eds) Advances in next generation services and service architectures. River Publisher, pp 43–63
- Minerva R, Moiso C, Viviani G (1998) The middleware at the verge between internet and telecom services. CSELT Tech Rep 26:657–672
- Moerdijk AJ, Klostermann L (2003) Opening the networks with Parlay/OSA: standards and aspects behind the APIs. IEEE Netw 17(3):58–64
- Moiso C et al (2010) Towards a service ecology for pervasive networked environments. Future Network and Mobile Summit 2010. IEEE, Florence, Italy, pp 1–6
- Moriana Group (2004) Service delivery platforms and telecom web services—an industry wide perspective. Thought Leader Report, Moriana Group, Egham, Surrey, UK
- NGNI Group (2009) FUSECO playground. Report, Fraunhofer Fokus, Berlin
- NGNI Group (2005) Open IMS playground. Report, Fraunhofer Fokus, Berlin
- NGNI Group (2006) Open SOA Telco playground. Report, Fraunhofer Fokus, Berlin
- NGNI Group (2012) Smart communications playground. Report, Fraunhofer, Berlin
- NTIA (1996) Telecommunication service. Web page, national telecommunications and information administration. NTIA, Washington, USA
- Object Management Group (2006) CORBA Component Model 4.0. Specification formal/06-04-01. OMG, Boston
- Ohtsuki HC, Lieberman HE, Nowak AM (2006) A simple rule for the evolution of cooperation on graphs and social networks. Nature 441(7092):502–505
- Orfali R, Harkey D, Edwards J (2007) Client/server survival guide. Wiley, New York
- Pan J, Paul S, Jain R (2011) A survey of the research on future internet architectures. IEEE Commun Mag 49(7):26–36
- Parlay Group (2006) ParlayX 2.1 specification. Specification, The Parlay Group, London, UK
- Partridge C, Bailey I (2010) An analysis of services. Report Unclassified, Model Future. UK Ministry of Defence, London, UK
- Pavlovski CJ (2007) Service delivery platforms in practice [IP Multimedia Systems (IMS) Infrastructure and Services]. IEEE Communications Magazine (IEEE) 45(3):114–121

- Pavlopoulos A, Cooper R (2007) Towards a survey of the deployment space for internet applications. In: BNCOD '07 24th British national conference on databases. Springer, Glasgow, UK, pp 110–119
- Peterson L, John W (2007) Overview of the GENI architecture. GENI Design Document GDD-06-11, GENI: Global Environment for Network Innovations, Washington, DC, USA: NSF
- Pouwelse, J, Garbacki P, Epema D, Sips H (2005) The bittorrent P2P file-sharing system: measurements and analysis. Peer-to-peer systems. Springer, Berlin, pp 205–216
- Prodan R, Ostermann S (2009) A survey and taxonomy of infrastructure as a service and web hosting cloud providers. In: 10th IEEE/ACM international conference on grid computing. IEEE, Banff, Alberta, Canada, pp 17–25
- Proper E (2010) Trends in enterprise architecture research. In: Aier S, Ekstedt M, Matthes F, Proper E, Sanz JL (eds) 5th TEAR Workshop. Springer, Delft
- Richards R (2006) XML-RPC. In: Proceedings of PHP XML and web services (Apress), pp 595–631
- Riede C, Al-Hezmi A, Magedanz T (2008) Session and media signaling for IPTV via IMS. In: Proceedings of the 1st international conference on MOBILE wireless MiddleWARE, operating systems, and applications. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels. Article n. 20
- Ripeanu M, Foster I, Iamnitchi A (2002) Mapping the gnutella network: properties of large-scale peer-to-peer systems and implications for system design. arXiv preprint cs/0209028, arXiv
- Rotem-Gal-Oz A (2006) Fallacies of distributed computing explained. <http://www.rgoarchitects.com/Files/fallacies.pdf>, 20
- Sadashiv N, Dilip Kumar SM (2011) Cluster, grid and cloud computing: a detailed comparison. In: 6th international conference on computer science and education (ICCSE). IEEE, Singapore, pp 477–482
- Sahoo J, Mohapatra S, Lath R (2010) Virtualization: a survey on concepts, taxonomy and associated security issues. In: Second international conference on computer and network technology (ICCNT). IEEE, Bangkok, Thailand, pp 222–226
- Saltzer JH, Reed DP, Clark DD (1984) End-to-end arguments in system design. ACM Trans Comput Syst (TOCS) (ACM) 2(4):277–288
- Santoro N (2006) Design and analysis of distributed algorithms, vol 56. Wiley-Interscience, New York
- Schaffer HE (2009) X as a service, cloud computing, and the need for good judgment. IT Prof (IEEE) 11(5):4–5
- Service Science, Wikipedia (2013) Service science, management and engineering. Wikipedia. http://en.wikipedia.org/wiki/Service_science. Accessed 27 May 2013
- Shanhe Y, Li C, Li Q (2015) A survey of fog computing: concepts, applications and issues. Workshop on Mobile Big Data (pp. 37–42). ACM
- SOA-Wikipedia (2013) Definition of service oriented architecture. http://en.wikipedia.org/wiki/Service-oriented_architecture. Accessed 26 May 2013
- Song J, Chang MY, Lee SS, Joung J (2007) Overview of itu-t ngn qos control. In: IEEE (ed) Commun Mag 45(9):116–123
- Sward RE, Boleng J (2012) Service-oriented architecture (SOA) concepts and implementations. In: ACM SIGAda Ada letters. ACM, New York, pp 3–4
- Taylor IJ (2005) From P2P to web services and grids: peers in a client/server world. Springer, Berlin
- Taylor IJ, Harrison AB (2009) From P2P and grids to services on the web: evolving distributed communities. Springer, Berlin
- Thampi, SM (2009) Introduction to distributed systems. preprint [arXiv:0911.4395](https://arxiv.org/abs/0911.4395), arXiv preprint [arXiv:0911.4395](https://arxiv.org/abs/0911.4395)
- TISPAN (2009) NGN functional architecture. ETSI ES 282 001. ETSI, Sophia Antipolis
- Turner M, Budgen D, Brereton P (2003) Turning software into a service. IEEE Comput 36(10):38–44

- Vahdat A (2012) Symbiosis in scale out networking and data management. In: Proceedings of the 2012 international conference on management of data. ACM, Scottsdale, Arizona, USA, pp 579–580
- Walkden M et al (2002) Open service access: advantages and opportunities in service provisioning on 3G mobile networks definition and solution of proposed Parlay/OSA specification issues. OSA Specification issues, Project P1110 Technical Information EDIN. Eurescom, Heidelberg, Germany
- Yagi, H, Ohtsu K, Wakano M, Kobayashi H, Minerva R (1995) TINA-C service components. In: Proceedings of TINA95 integrating telecommunications and distributed computing-from concept to reality. TINA_C, Melbourne, Australia
- Znaty S, Hubaux JP (1997) Telecommunications services engineering: principles, architectures. ECOOP Workshops. Springer, Jyväskylä, Finland, pp 3–11

Networks and New Services: A Complete Story

Minerva, R.; Crespi, N.

2017, XIX, 186 p. 90 illus., 83 illus. in color., Hardcover

ISBN: 978-3-319-33993-1