2 Background: Service Oriented Network Architectures

Most of the issues in the Internet arise because of inflexibility and rigidness attributes of the network architecture, which is built upon a protocol stack. The problem that is faced by the Internet is that it is hard to integrate new functionalities in it and to remove existing functionalities from it. The reason is that protocols and layers are tightly coupled between themselves as well as within each other. In addition, they are also coupled with the applications. This problem is not limited to specific protocols and mechanisms. However, it is an architectural issue.

Similar problems were seen in software engineering which has evolved to manage complexities (e.g. maintenance, integration of new functionalities, time and task management) of development process, which has direct effects in terms such as of cost, quality and development time. That is why, for designing a new software architecture for the Internet core, the principles and techniques from software engineering can be applied.

Software engineering has evolved from structural programming to service oriented programming. The design of a future network architecture can benefit from software engineering techniques to make network architecture
more flexible and easy to maintain rather than having an ossified architecture (e.g. Internet).

The Service Oriented Network Architecture (SONATE) [82][97], a clean slate network architecture, applies the principles of Service Oriented Architecture (SOA) to communication systems.

Services are the essential elements of a SOA. The protocol stack of the Internet has also been developed considering services.

2.1 Layering in Protocol Stacks

To reduce complexity and promote modularity, the protocol stack has been organized as layers. The International Organization for Standardization (ISO) had specified seven layers for the Open System Interconnection (OSI) model namely physical, data link, network, transport, session, presentation and application [144]. The TCP/IP model has 5 layers as it integrates all of the functionalities of the session, presentation and application layers of the OSI model into one layer called application layer [113]. Each layer provides services to its upper layer and consumes services from its lower layer.

An example scenario is shown in Figure 2.1 where a user started browsing Internet using the WLAN connection of his laptop. In this case, the browser sends the request to the server using Hyper Text Transfer Protocol (HTTP). The packet is then sent to the TCP protocol of the transport layer which encapsulates the HTTP packet and wraps its with its own header and trailer around it. The transport layer then sends the packet to the IP protocol of the network layer which does the same, i.e., encapsulates the TCP packet coming from transport layer and adds its own header and trailer. The packet is then sent to its lower layer so called link layer. The IEEE 802.11 protocol
Fig. 2.1 Communication using layered network stacks (adapted from [121])

of the MAC layer does the same function and sends the packet to its physical layer which changes the bits into analog signals and transmits it.

Fig. 2.2 Protocol graph in the current networks (adapted from [121])
The receiver (WLAN router) gets the analog signal, converts that into a sequence of bits and sends that to the IEEE 802.11 protocol of the MAC layer. The MAC layer extracts and modifies its header and trailer if necessary and sends the packet to the IP protocol (network layer) of the router. There is an internal IP-to-IP packet transmission inside the router. The IP protocol of the network layer then adds its own header and trailer to it and sends the packet to the Ethernet protocol of the link layer. The link layer then adds its own header and trailer to it and sends the packet to its physical layer which converts the bits into signals and sends it to the physical layer of the server. The physical layer of the server then changes the signals back into bits and sends the bits to the Ethernet protocol of the link layer. The link layer then extracts its headers and trailers and sends the IP packet to the network layer. Similarly, the IP protocol of the network layer extracts its header and trailer and sends the packet to the TCP protocol of the transmission layer. The TCP protocol of the transmission layer extracts its header and trailer and sends the packet to HTTP protocol of the application layer in the server side. The server receives the request and then sends its replies using the mentioned procedures. It is worthy to note that the router (i.e., the middle boxes) does not change the contents of the higher layers like TCP or HTTP packets.

A user usually has several applications running on his machine like browsing Internet and running a network management application. In that case, the set of protocols that are used in a sequence for all applications are not same. The sequence of protocols that are used to run an application is called a “Protocol graph”. As shown in Figure 2.2, a browser uses the red marked protocol graph (HTTP-TCP-IP-ETH) and network management application uses green marked protocol graph (SNMP-UDP-IP-ETH).
As there are not many alternative protocols in the same layer, the number of protocol graphs used today is limited.

2.2 Service Oriented Architectures (SOA)

The three main entities of a basic service oriented architecture are service provider, service consumer and service broker \[89\] as shown in Figure 2.3. In the thesis, the color codes for service provider, service broker, and service consumer are chosen as pale green, blue, and light grey respectively. A service provider creates and registers its service to the service broker. A service consumer searches its required service to the service broker. After finding the service, he binds with the service for consuming.

![Fig. 2.3 Roles and Operations of SOA \[89\]](image)

Eight main principles of SOA are as follows \[35][36]\.

Loose coupling: Coupling refers to the degree of dependencies and binding between two components. Loose coupling defines independence of a
service; where in order to execute own functionalities a service does not require to have a knowledge about other services.

Service contract: A communication agreement which is covered by service description(s) or related documents.

Autonomy: Control of a service over the logic it encapsulates characterizes the autonomy.

Abstraction: Services are independent of the logic they use and those logic is hidden from the outside world.

Re-usability: A service should be independent and fine-grained enough so that it can be used later on with no or minor modification.

Composability: An ability of a service to be coordinated to other services for forming a composite service. Composability fosters re-usability of a service.

Statelessness: A property in which services do not keep the state after request has been processed.

Discoverability: A service should be descriptive enough to be discovered easily.

SOA can provide new prospects to build a future network architecture as SOA addresses loose coupling, re-usability and autonomy of a service, which are fundamental requirements of a flexible architecture. The OSI or TCP/IP protocol stack can be decomposed into various functionalities which are described with formal contract (i.e. service description) as it makes functionalities autonomous and self-descriptive. A self-descriptive functionality has the ability to be discovered as it carries attached description which can be processed by the discovering entity. Abstraction is another point to be taken into account while decomposing a network stack (TCP/IP, UDP/IP, SCTP/IP) into various functionalities, it should be at the abstract level where
it does not rely on a particular implementation thus logic should be hidden from the users and applications. Characteristics of a functionality such as autonomy, description and re-usability, make it composable. The concept of composability fosters ease of integration of functionalities. Nevertheless, the statelessness principle of SOA might not be appropriate for all functionalities of a network architecture as some functionalities of a network do require to keep the state (e.g. reliable transmission).

2.3 Service-Oriented Network Architectures (SONATE)

Now the question is: how to apply the SOA design principles on networks? Techniques like Web-Services and XML data structures were designed for the interplay of distributed autonomous functionalities on application level as shown in Figure 2.4a. Network functionalities like routing, data encoding, or flow control itself are inherently distributed as shown in Figure 2.4b. Thus specialized concepts for building networks according to SOA principles as well as new techniques for supporting SOA are required. The following subsection 2.3.1 provides an overview of such concepts.

2.3.1 Basic Concepts of a Service Oriented Network Architecture

Services are the main elements of a SOA. A service represents the effects of an activity rather than algorithms and data structures, i.e. a service represents a higher abstraction level since different algorithms may implement the same service. A building block is the implementation of an atomic com-
munication service. A Micro-Protocol (MP) can be an example of a building block such as retransmission, data encryption (AES 256), and error correction (hamming code). Usually, each building block has several effects, for instance, reliable or confidential data transmission. But, there are also effects like increasing the end-to-end delay or reducing the maximum payload size. All the effects of a building block represent its services. The interfaces of a building block should reflect the provided services and hide the implementation details. Building blocks should also use generic interfaces so that the interaction between building blocks does not require extra adapters.

It is necessary that there are explicit service descriptions. Such descriptions should include effects and interfaces as shown in Figure 2.5. The effects of a service are offered through interfaces. The methods or operations that are exposed by web services are described using web service description
language (WSDL) along with the message format and protocol details. As a communication service is offered by a self-contained building block, the operations and message formats that are exposed by building blocks are hidden and only the resultant outcomes (i.e., effects) need to be described which necessitates a language to describe communication services. In this thesis, a communication service description language has been developed to fulfill this demand.

Flexibilty can be achieved if we apply the SOA principles into the network. A network architecture should be flexible in two means. Firstly, networks should be able to adapt to specific customer or application needs and changing environmental conditions. Secondly, networks should be able
to evolve, i.e. to add a new functionality, to change, update, or remove an existing functionality. This flexibility is achieved by composing several (smaller) services to a more complex and specialized service. In today’s networks, complex protocols are organized in layers, building a static protocol graph [87]. Service oriented network architectures aim at supporting dynamic composition of services, i.e. dynamic protocol graphs as shown in Figure 2.6. Without being dependent on a static protocol graph, it is easier to make use of new protocols (i.e. building blocks) and to reuse functionalities on different levels. Having dynamic protocol graphs implies that there is no static placement of functionalities as defined by the layers of the OSI reference model. In this sense such networks will be layerless including compression/encryption can be used for application payload only or also for some protocol headers. Furthermore, it is not necessary that protocols are processed in sequence, for example, there might be different branches in the protocol graph to handle different but related data types within one flow, e.g. signalling and streaming media. In order to enable dynamic protocol graphs the interaction between building blocks should not be defined by executable code, but by description which can be easily changed.

2.3.2 SOA Principles in Networks

In order to fulfill the SOA principles, it is crucial to design services and building blocks appropriately. Nevertheless, the basic concepts of a service oriented network architecture described above support a service oriented design. Using service as the basic element for the design of a system instead of algorithms or protocols foster loose coupling and abstraction. Service descriptions represent the service contracts and are also used to discover
Description and Selection of Communication Services for Service Oriented Network Architectures
Khondoker, R.
2016, XXII, 211 p. 34 illus., Softcover
ISBN: 978-3-658-12741-1