2 Background

The first part of this Chapter introduces the Web service protocol stack, the basic Web service interaction model, and Web service related protocols. The second part discusses QoS metrics and aspects. The third part introduces the notion of mobile Web services and constraints of mobile devices.

2.1 Web services

According to the Web service specification available at W3C [6], the definition of a Web services "is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web related standards".

Figure 2 depicts the basic protocol stack of Web services, consisting of HTTP for transport, SOAP (SOAP stands formerly for Simple Object Access Protocol) for XML-based message exchange, Web services description language (WSDL) for service description, Universal Description, Discovery and Interoperability (UDDI, [7]) as service registry, and Business Process Execution Language for Web Services (BPEL4WS, [8]) for Web service composition. Considering Web services

itself as a layer, it could be placed between the application and transport layer in terms of the Internet Model.

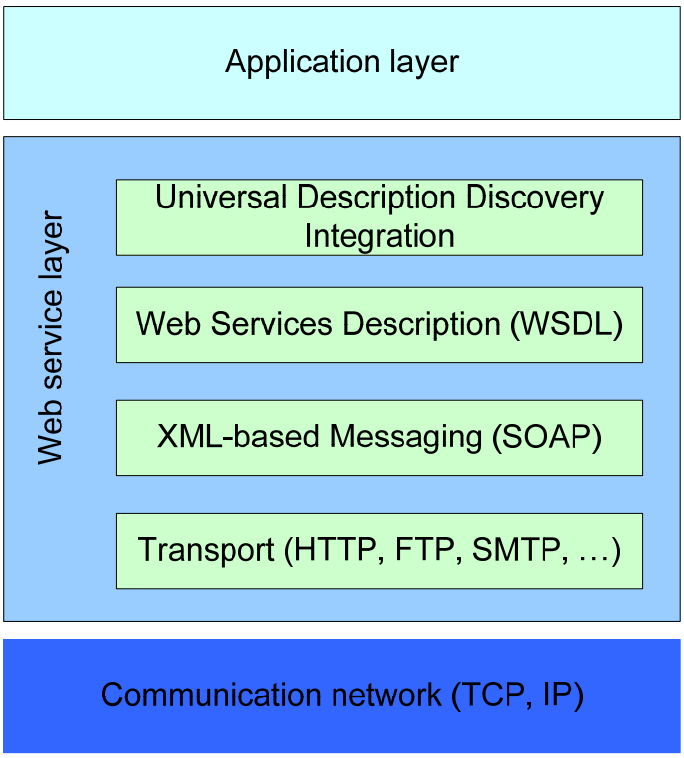

Figure 2. Web service protocol stack

With the increasing popularity and importance of Web services, a number of Web service related standards have been defined beyond the basic Web service protocol stack in different areas such as transaction [9], security [13], addressing [14], discovery [15], composition [11], etc, which will be introduced in the following subsections.

2.1.1 Basic interaction model

The basic interaction model consists of three parties, service provider, service requestor, and the service registry. As shown in Figure 3, the service provider publishes its services in the service registry; the service client finds the required service in the service registry. After discovering an appropriate service in the service registry, the service client invokes this service at the service provider.

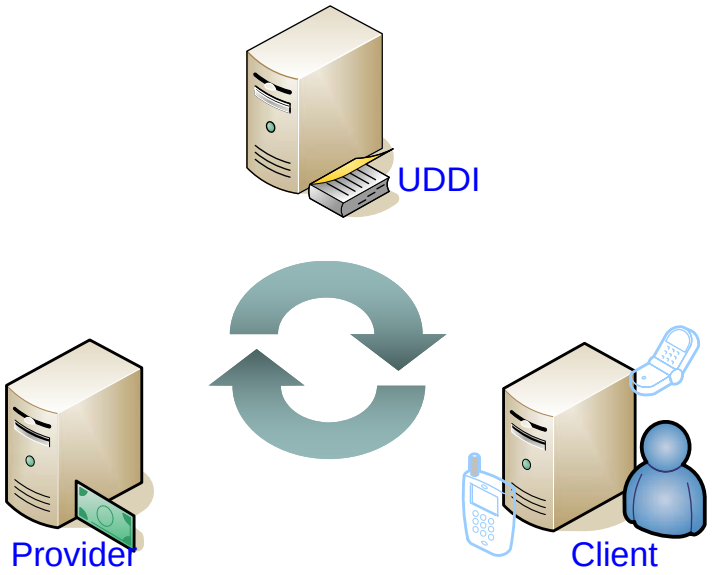

Figure 3. Basic interaction model of Web services

The easiest way for service requestors to communicate with a service provider is to invoke a service at the service provider directly. However, in many cases the service requestor does not know how to invoke the service. She may not be aware of the existence of the service, neither the uniform resource locator (URL) of the service nor how to use the service. Therefore, the service requestor needs a so called Web service description (WSD) in order to be capable to use a Web service. A WSD is a document that contains information on operations and messages, the supported network protocols, and the endpoint of a Web service.

As the basic interaction model of Web services depicts, the service provider describes its services in a WSD document using the XML-based WSDL. The service provider registers the WSD document at the service registry, e.g. a UDDI. Service requestors can now ask the service registry for available services through a public interface. If the client finds a suitable service it gets the WSD from the registry, generates a proxy object according to the WSD, implements the client software and communicates with the service provider directly [10].

The service registry or a third party discovery tool allows late binding of the service that a service requestor consumes at runtime. Late binding means service clients are able to select service providers at runtime rather than at the implementation time. A service provider implementing the same WSD can be made active at runtime and registers itself at the service registry. When a service requestor asks the service registry for currently available services before invoking a service, the service registry could suggest the service requestor to use the new service provider. The reasons could be the locality of the service provider to the service requestor or the more competitive price.

2.1.2 Web service transaction specifications

Web services will be involved in transactional activities, e.g. electronic procurement portals provide Web services to customers from different companies, which normally run different IT infrastructures. Several protocols for Web

services transactions have been specified by different companies and organizations, e.g. OASIS Business Transactions Protocol (BTP, [11]) and the Web Services Transactions (WS-Tx, [9]) specification defined by IBM. They both attempt to address the problems of running transactions with Web services. The main differences between them are that "BTP has only one model that is to be used to solve a variety of different problems. It does this by relaxing restrictions such as atomicity, durability etc. within the protocol (cohesion or atom) and allowing services to define those semantics outside of the model." WS-Tx consists of a protocol suite that separates the coordination from transactions offering a chance for a cleaner separation of concerns [11].

2.1.3 Web service security

Web services security (WS-Security, [12]) is designed to allow applications to conduct secure message exchange over SOAP. Different security models and encryption technologies can be applied for the mechanisms of the specification. WS-Security allows the association of security tokens with message exchange. As a general-purpose protocol, WS-Security does not require any specific type of security tokens.

2.1.4 Web service trust

The Web Services Trust Language (WS-Trust, [13]) defined by BEA Systems, IBM, Microsoft, etc. uses the secure messaging mechanisms of WS-Security to define additional primitives and extensions for the issuance, exchange and validation of security tokens. WS-Trust also enables the issuance and dissemination of credentials within different trust domains.

Web services-Trust extends Web services-Security in order to issue and exchange security tokens and mechanisms to establish and access the presence of trust relationships. Cooperating parties need to exchange security credentials. Applications can apply WS-Trust for secure communications designed to work with the general Web service framework [13].

2.1.5 Web service addressing

Web service Addressing (WS-Addressing) is a member submission of BEA, IBM, Microsoft, and SUN Microsystems to W3C. WS-Addressing provides transport-neutral mechanisms to address Web services and messages. Specifically, this specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner [14].

2.1.6 Web services discovery

The specification of Web services Discovery (WS-Discovery) was led by Microsoft. The specification is already supported by Microsoft, Canon, BEA Systems, and Intel. The target of WS-Discovery is to support mobile devices that are not always connected to a network with Web service interoperability.

WS-Discovery defines a multicast discovery protocol that occasionally connected devices apply in order to locate or announce services. By default, probes are sent to a multicast group, and target services that match return a response directly to the requester [15]. No DNS or UDDI are required, but they can be used if the devices and the circumstances allow this.

2.2 Quality of service

In this subsection, we introduce QoS classes and selected work targeting QoS improvement in end systems such as client and server systems. The communication path between two end systems e.g. between client and server may consist of myriad infrastructures with different characteristics. The classical technical QoS metrics are e.g. delay, jitter, packet loss rate, and throughput in the network technologies. QoS aspects and metrics can also be identified in other areas such as availability, reliability, processing time, throughput, and security of a web server, application server (hosting Web services) or end system.

Many QoS-aware network systems such as UMTS/GPRS and Differentiated Services (DiffServ) support QoS classes, which categorize QoS metrics into groups and thus simplify the treatment of traffic.

2.2.1 QoS classes

Both the 3G wireless network (UMTS) and the DiffServ support traffic classes with different QoS metrics. DiffServ as defined by the Internet Engineering Task Force (IETF) supports expedited forwarding, assured forwarding, and best effort as traffic classes. UMTS, defined by the Third Generation Partnership Project (3GPP), supports four QoS classes: conversational, streaming, interactive, and background. With the adaptation of IP as the core network protocol of UMTS, it is reasonable to argue that the mapping among the different traffic classes of the two different technologies will be essential in order to provide end to end QoS support [16].

Classes of services are often used in a Service Level Agreement (SLA), which is an agreement between service providers and customers to provide a certain level of services. Penalty clauses may apply if the SLA is not met. Parameters defined in an SLA can be mapped onto the technical systems such as wired or wireless network, web servers, and end systems.

One could categorize the QoS classes like the following:

- **Efficient service**

Target applications: Services that need high performance for quick signaling and exchanging important (light-weight) data.

Possibly mapped to: UMTS: Conversational / DiffServ: Expedited forwarding

Features:

- low delay
- low packet loss rate
- high throughput

- **Reliable service**

Target applications: Services that need high reliability, e.g. booking confirmation.

Possibly mapped to: UMTS: Interactive / DiffServ: Assured Forwarding (AF)

Features:

- high reliability
- high availability
- non-repudiation
- low packet loss rate

- **Confidential service**

Target applications: Services dealing with any confidential information such as banking, booking, and billing.

Possibly mapped to: UMTS: Interactive / DiffServ: Assured forwarding

Features:

- authentication
- encryption
- integrity
- non-repudiation
- high reliability
- high availability

It is worth noting that UMTS and DiffServ do not support all of the listed features.

- **Lookup service**

Target applications: Services providing lookup of non-confidential information such as stock quotes or price information. These should react quickly to ensure rapid proceeding of the calling service. Since there will be several similar services, availability issues are not critical, at least not for users.

Possibly mapped to: UMTS: Interactive / DiffServ: Expedited forwarding

Features:

- low delay

- **Best effort service**

Target applications: Any background traffic which is neither time critical nor confidential.

Possibly mapped to: UMTS: Background / DiffServ: Best Effort

Features:

- no specific QoS support

2.2.2 Service differentiation in web servers

There are myriad approaches towards service differentiation in end systems. A simple scheme is for example, incoming requests are dropped when a certain threshold of the end system is reached. Another approach adopts the internal resource usage due to the network conditions [17].

Voigt presented in [18] the design, implementation, and evaluation of in-kernel mechanisms for service differentiation and overload protection of web servers. Due to the fact that potentially discarded requests consume resources, the author proposed that the admission control of incoming requests should take place as early as possible in the lifetime of a web transaction.

When a server is overloaded it can not process all requests in a timely manner. Therefore, it is desirable to perform service differentiation depending on the importance of the incoming requests. Voigt demonstrated the improved efficiency and scalability of the in-kernel mechanisms compared to the same mechanisms implemented in user space.

2.2.3 Improvement of the availability of web servers

In order to improve the availability of web servers, Cotroneo et. al. presented in [19] an architecture that manages the I/O activities of all processes residing on the web server. Since performance failures of web servers stem mainly from overload, the proposed resource management strategy aims at scheduling I/O tasks. The proposed architecture is able to completely separate I/O from CPU activity. Three classes of service are defined which results in different priorities of I/O activities.

2.3 Mobile computing / Mobile devices

More and more people are relying increasingly on mobile devices in order to interact with the Internet, consuming services, and downloading videos.

Although the computing power will increase rapidly and future wireless systems such as beyond 3G will offer more data rate and bandwidth, mobile devices and wireless mobile systems will never reach the capabilities of wired devices and systems. Therefore, Web services should be designed carefully for the ever increasing usage from wireless and mobile systems.

Providing Web services in a mobile environment brings up issues of performance and QoS that are not obvious in a fixed environment to that extent. The basic constraints of mobile environments are:

- Limited bandwidth connection over the wireless link
- High probability of bit errors and packet losses
- Changing environment

The basic constraints of mobile devices are:

- Limited computing power
- Limited memory
- Limited screen size
- Limited data rate
- Limited battery lifetime
- Higher monetary charge

2.4 Conclusion

The first part of this chapter introduced background information on Web services, including the basic protocol stack and interaction model of Web services and Web service related standards. The second part discussed different aspects of QoS, including class of services, and QoS differentiation in end systems. The last part of the chapter introduced the notion of mobile Web services. With the rapidly growing number of mobile devices, mobile Web services will become an important issue when developing services. The limitations of mobile devices were classified.

In this thesis, we will present our design, implementation, and evaluation of our WS-QoS framework which provides solutions towards QoS integration into Web services. The next chapter gives an overview of selected major industrial and academic efforts towards QoS specification and management for Web services.