Aus dem Institut für Telematik
der Universität zu Lübeck

Direktor:
Prof. Dr. rer. nat. Stefan Fischer

# Entwicklung einer Service-orientierten Architektur zur vernetzten Kommunikation zwischen medizinischen Geräten, Systemen und Applikationen

Inauguraldissertation
zur
Erlangung der Doktorwürde
der Universität zu Lübeck
– Aus der Technisch-Naturwissenschaftlichen Fakultät –

Vorgelegt von
Herrn Dipl.-Inf. Stephan Pöhlsen
aus Hamburg

Lübeck 2010

| | |
|---|---|
| Erster Berichterstatter: | Prof. Dr.-Ing. Christian Werner |
| Zweiter Berichterstatter: | Prof. Dr.-Ing. Jörg-Uwe Meyer |

Tag der mündlichen Prüfung: 18.01.2010

Zum Druck genehmigt. Lübeck, den 20.01.2010

# Kurzfassung

Die Integration medizinischer Geräte und Systeme zur vernetzten Kommunikation ist von Interoperabilitätsproblemen geprägt. Das Konzept einer Service-orientierten Architektur (SOA), alle Dienste über eine standardisierte Schnittstellentechnologie als „Service“ anzusprechen, hat zum Ziel, diese Probleme zu lösen. In der vorliegenden Arbeit wird eine solche SOA entwickelt, die Plug-and-Play-Prinzipien verfolgt und als technologische Basis Web Services verwendet. Web Services zeichnen sich durch ihre Interoperabilität und ihre weite Verbreitung aus.

Die Anforderungen, die im medizinischen Bereich an einen Verzeichnisdienst gestellt werden, sind mit bestehenden Lösungen jedoch nicht zu erfüllen. Daher wird in dieser Arbeit eine Discovery-Architektur entwickelt. Diese setzt auf WS-Discovery auf, welche die Basis für Geräte-Plug-and-Play im Web-Service-Umfeld ist. WS-Discovery wird für die Discovery-Architektur so erweitert, dass innerhalb einer Klinik – auch über lokale Netzwerkgrenzen hinweg – ein Auffinden von Services möglich ist.

In dieser SOA ist eine zuverlässige Kommunikation patientenaher Geräte notwendig. Insbesondere soll sie nicht von anderen Systemen eingeschränkt werden können, wenn diese zum Beispiel mit einem Computer-Virus oder -Wurm befallen sind, welche versuchen, das Netzwerk zur weiteren Verbreitung zu nutzen. Für die SOA wird deshalb eine Switch-Architektur entwickelt, die durch eine gleichmäßige Verteilung der Datenrate die Übertragung von Nutzdaten garantiert. Auf diese Weise lässt sich sicherstellen, dass an einem solchen Switch angeschlossene Geräte miteinander kommunizieren können.

Außerdem ist es notwendig, dass in einer SOA, die im medizinischen Bereich eingesetzt werden soll, Zugriffskontrollen vorhanden sind. Das entwickelte Sicherheitskonzept verfolgt einen dezentralen Ansatz, bei dem Geräte eigene Sicherheitsinformationen bereitstellen, um einen gegenseitigen Zugriff zu reglementieren. Dazu gehören neben einer Identität insbesondere die Service-Typen, die angeboten bzw. genutzt werden dürfen.

Weiterhin spielt die Performance von Web Services eine wichtige Rolle. Es werden vergleichende Performance-Messungen von Data-Distribution-Service- und Web-Service-Implementierungen durchgeführt. Dabei soll aufgezeigt werden, wo Verbesserungen für eine Verwendung im medizinischen Bereich notwendig sind, bzw. ob und wie sich diese mit Web-Service-Technologien umsetzen lassen.

Zusammengenommen stellt die Arbeit eine SOA vor, die technische Interoperabilität im medizinischen Bereich ermöglicht. Sie bildet eine Basis für darauf aufbauende semantische Interoperabilität.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Kapitel 1

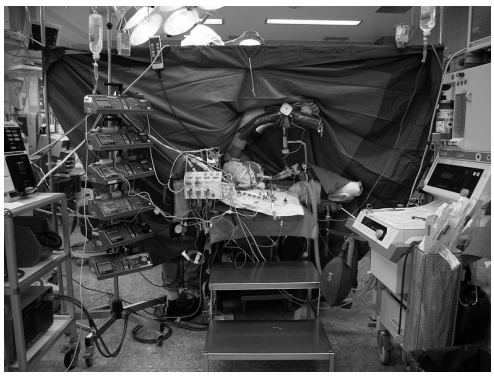
# Einleitung

Medizinische Geräte sind ein unentbehrlicher Bestandteil moderner Medizin. Messungen von Vitalparametern, Verabreichung von intravenösen Medikamenten, Röntgen- und Ultraschalluntersuchungen sowie weitere, teils lebenswichtige Maßnahmen, sind Routineaufgaben, die mithilfe medizinischer Geräte durchgeführt werden.

Abbildung 1.1a zeigt einen Operationssaal aus dem Jahre 1956, der für damalige Verhältnisse modern war. Der Patient steht im Mittelpunkt und bekommt die volle Aufmerksamkeit. Seitdem hat mit dem technischen Fortschritt die Anzahl an medizinischen Geräten stetig zugenommen. Die Geräte dienen teils der Verbesserung der medizinischen Versorgung, teils entlasten sie das medizinische Personal.

Ein aktueller Operationssaal ist in Abbildung 1.1b zu sehen. Zahlreiche Geräte sind an der Behandlung des Patienten beteiligt. Der Patient steht zwar noch immer im Mittelpunkt, jedoch nimmt die Bedienung der Geräte und die Beobachtung der Monitore immer mehr Aufmerksamkeit des medizinischen Personals in Anspruch.

(a) Historischer Operationssaal; Quelle: Bundesarchiv, Bild 183-35596-0002; Foto: Erich Höhne und Erich Pohl | 17. Januar 1956



(b) Heutzutage sind medizinische Geräte unentbehrlich; Foto: Klinik für Anästhesiologie der Universität zu Lübeck

Abbildung 1.1: Technisierung der Operationssäle

## 1.1 Interoperabilität medizinischer Geräte

Unter Interoperabilität versteht man die Fähigkeit von Geräten oder Systemen, möglichst nahtlos miteinander zusammenzuarbeiten. Nach Lesh et al. [82] ist Interoperabilität eine kontinuierliche Größe, die sich anhand ihrer Komplexität bestimmen lässt. Am unteren Ende, mit der geringsten Komplexität, befindet sich die physikalische Interoperabilität. Hierbei ist es möglich, zwei Geräte miteinander zu verbinden und bestimmte Funktionen des anderen Gerätes zu benutzen. Die höchste Komplexität bezüglich Interoperabilität wird Geräten auf semantischer Ebene abverlangt. Sie müssen übertragene Daten bzw. Informationen verstehen und geeignet darauf reagieren können.

Mit dem technischen Fortschritt bei Medizingeräten konnte die Interoperabilität der Geräte untereinander nicht mithalten. Viele Geräte wurden als eigenständige Einheiten konzipiert und kommunizieren meist in keiner Weise mit anderen Geräten.

Werden dennoch Geräte miteinander verbunden, so handelt es sich hierbei oft um Punkt-zu-Punkt-Verbindungen mit proprietären Protokollen eines Herstellers. Diese werden bei größeren Geräten, wie Anästhesiegeräten, benutzt, um einzelne Module anzubinden oder Informationen an andere Geräte weiterzureichen. Ein ganzheitliches, herstellerübergreifendes Konzept existiert nicht, sodass immer öfter zusätzliche Verbindungen zwischen existierenden Systemen hinzugefügt werden.

Durch mehr Interoperabilität zwischen medizinischen Geräten können Arbeitsabläufe optimiert werden. So ist es beispielsweise möglich, durch eine automatische Protokollierung eine vollständigere Patientenakte zu erstellen. Eine Voraussetzung hierfür ist allerdings, dass Geräte und Systeme auch herstellerübergreifend miteinander kommunizieren.

Durch automatisierte Abläufe können Fehlerquellen reduziert und gleichzeitig Kosteneinsparungen erreicht werden. Insofern ist es verständlich, dass sich Betreiber von Gesundheitseinrichtungen mehr Interoperabilität von den Medizingeräteherstellern wünschen. Um Interoperabilität zwischen Geräten unterschiedlicher Hersteller zu ermöglichen, müssen sowohl technische als auch rechtliche Aspekte beachtet werden.

### 1.1.1 Technische Aspekte

Bei der Datenübertragung sind im medizinischen Umfeld unterschiedliche Schnittstellen verbreitet, von seriellen RS-232-Verbindungen bis hin zu Netzwerken auf Ethernet-Basis. Diese Schnittstellenvielfalt sorgt bereits für Probleme bei der technischen Interoperabilität.

Ein Beispiel aus dem Alltag ist die Stromversorgung. Bei Steckdosen haben sich weltweit verschiedene Standards etabliert. Nicht alle Länder haben die gleichen Steckdosen,

sodass auch hier Interoperabilitätsprobleme vorhanden sind. Diese lassen sich jedoch meist mit entsprechenden Reise-Stromadaptern beheben.

Bei neueren Medizingeräten, welche Daten austauschen sollen, setzen sich Ethernet-Schnittstellen durch. Ein Grund hierfür ist, dass bei Ethernet mehr als zwei Geräte miteinander kommunizieren können, und nicht nur Punkt-zu-Punkt-Verbindungen, wie bei seriellem RS-232, möglich sind. Außerdem kann in vielen Krankenhäusern die bereits vorhandene Ethernet-Verkabelung der IT-Infrastruktur mitbenutzt werden. Selbst mit einer vereinheitlichten Netzwerk-Infrastruktur können heutige medizinische Geräte jedoch nicht zwangsläufig miteinander kommunizieren, da unterschiedliche proprietäre Protokolle auf den höheren Ebenen des ISO/OSI-Modells für den Nachrichtenaustausch verwendet werden.

In einigen speziellen Bereichen war der Bedarf an Interoperabilität in der Vergangenheit groß genug, dass sich dort anwendungsspezifische Standards etabliert haben. Als Beispiel ist hier der DICOM-Standard [25] zu nennen. Er wird sowohl für bildgebende als auch für bildverarbeitende Systeme verwendet und spezifiziert das Format zur Speicherung und Übertragung von Bilddaten.

Der Begriff *HL7 (Health Level 7)* wird je nach Zusammenhang unterschiedlich verwendet. Auf der einen Seite handelt es sich um *Health Level Seven, Inc.* [51], eine Organisation, die Normen im Gesundheitswesen entwickelt. Auf der anderen Seite werden die von ihr entwickelten Standards HL7 genannt, da sie auf der siebten Schicht des ISO/OSI-Modells einzuordnen sind. HL7-Standards werden vor allem innerhalb von Krankenhausinformationssystemen (KIS) und deren Subsystemen, wie Laborinformationssystemen (LIS), verwendet.

Am bekanntesten sind die HL7 v2.x Standards, von denen v2.6 [5] der aktuellste ist. Dies sind aufeinander aufbauende, abwärtskompatible Standards zum Nachrichtenaustausch. Spezifiziert ist dabei ein Nachrichtenformat bzw. -container. Dabei handelt es sich beispielsweise um ADT-Nachrichten (Admission, Discharge, Transfer), die administrative Informationen – wie den Aufenthaltsort eines Patienten im Krankenhaus – angeben, oder um Befundmitteilungen, die von einem LIS zum KIS gesendet werden.

In v2.x traten Probleme bezüglich der Semantik auf, woraufhin ein *Referenzinformationsmodell (RIM)* entwickelt und als ISO/HL7 21731 [67] standardisiert wurde. Es ist die Basis von HL7 Version 3, mit der durch einen gemeinsamen Wortschatz semantische Interoperabilität erreicht werden soll.

Für die Gerätekommunikation existiert die Reihe der ISO/IEEE-11073-Standards. Eine Einführung geben Galarraga et al. in [45]. ISO/IEEE 11073-10101 [69] spezifiziert eine Nomenklatur, um semantische Interoperabilität zu ermöglichen. Hierzu werden 32-Bit-Integers als identifizierende Bezeichner angegeben. Diese werden in ISO/IEEE 11073-10201 [70] in dem dort spezifizierten *Domain Information Model*

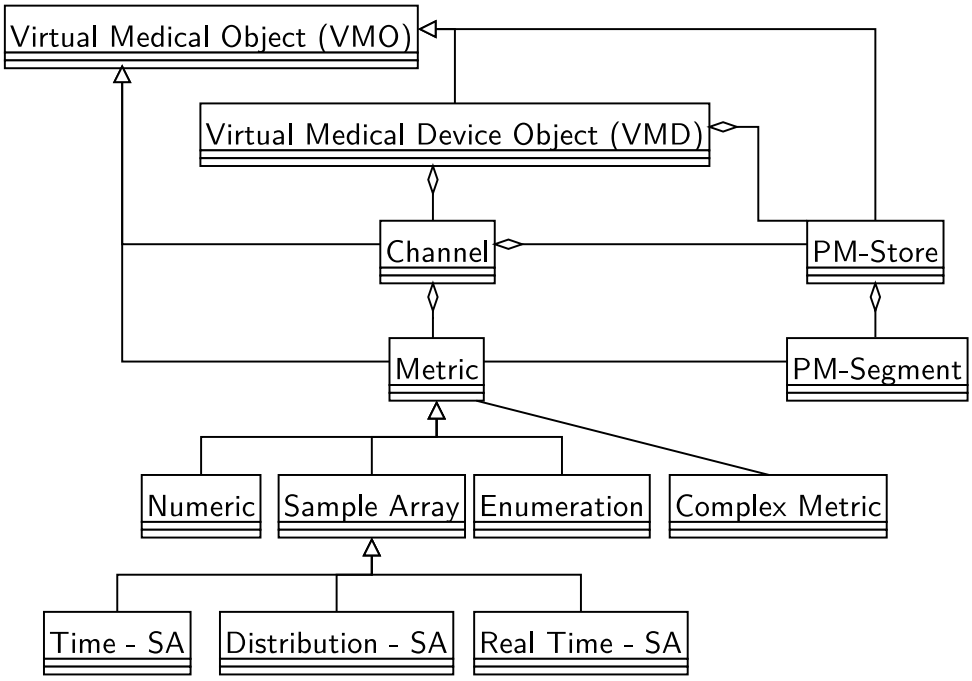



Abbildung 1.2: Klassendiagramm des *Medical Package* aus ISO/IEEE 11073-10201 DIM

*(DIM)* verwendet. Im DIM werden Objekte mit dazugehörigen Attributen sowie ihr Zusammenhang untereinander definiert.

In Abbildung 1.2 ist das *Medical Package* aus dem DIM auszugsweise dargestellt. Alle Objekte erben vom *Virtual Medical Object (VMO)*. Das *Virtual Medical Device Object (VMD)* ist eine Abstraktion von einem medizinischen Gerät. Dieses Gerät kann mehrere Kanäle haben, die wiederum mehrere Messobjekte haben. Ein Messobjekt ist selbst beispielsweise ein einzelner numerischer Wert oder eine Messdatenreihe. In den ISO/IEEE-11073-1030x-Standards wird das abstrakte VMD aus dem DIM verwendet, um bestimmte Geräteklassen, wie beispielsweise Infusionspumpen (11073-10301) oder Beatmungsgeräte (11073-10303), konkret zu modellieren.

Die Standardserie ISO/IEEE 11073-2xxxx dient dem Nachrichtenaustausch. Zusammen mit den ISO/IEEE-11073-1xxxx-Spezifikationen deckt sie die Schichten fünf bis sieben im ISO/OSI-Modell ab. Neben dem existierenden Basis-Standard in ISO/IEEE 11073-20101 [71] befinden sich zwei weitere Normen im Entwurf. In ISO/IEEE 11073-20201 soll ein Basismodell für ereignisgetriebene Kommunikation mit *event reports* definiert werden, während in ISO/IEEE 11073-20202 ein Abfragemodell für *polling* standardisiert werden soll.

Für die physikalische Verbindung sowie für den Nachrichtentransport sind die Spezifikationen aus dem Bereich 11073-3xxxx vorgesehen. Die beiden vorhandenen Standards ISO/IEEE 11073-30200 [72] bzw. 30300 [73] nutzen eine kabelgebundene bzw. drahtlose Verbindung auf Basis der IrDA-Protokolle *(Infrared Data Association)*. Die IrDA-basierten Verbindungen haben bei den Medizingeräteherstellern bisher keine ausreichende Verbreitung gefunden und entsprechen nicht mehr dem Stand der Technik.

### 1.1.2 Rechtliche Aspekte

Die Herstellung, Inverkehrbringung, der Betrieb und die Anwendung von Medizinprodukten unterliegen einer Regulierung und Normung. Bei der Interoperabilität von medizinischen Geräten sind daher auch rechtliche Aspekte zu berücksichtigen. Durch die Kommunikation zwischen Geräten können Fehler verursacht werden, die vorher nicht möglich waren. Deswegen ist es notwendig, die rechtlichen Grundsätze für medizinische Geräte von Anfang an zu beachten, wenn man Netzwerkprotokolle für Medizingeräte definieren möchte.

Die europäische Richtlinie 93/42/EWG über Medizinprodukte regelt die Aspekte Marktzugang und Patientensicherheit beim Einsatz medizinischer Geräte. In Deutschland wird sie durch das Medizinproduktegesetz (MPG) in nationales Recht umgesetzt, welches unter anderem durch die Medizinprodukte-Verordnung (MPV) und die Medizinprodukte-Betreiberverordnung (MPBetrV) ergänzt wird. Die EU-Richtlinie unterteilt medizinische Geräte in Abhängigkeit ihres Risikos in verschiedene Klassen.

Möchte ein Hersteller ein Medizinprodukt in Verkehr bringen, so benötigt er für dieses unabhängig von der Risikoklasse eine CE-Kennzeichnung. Hierfür ist eine erfolgreiche Durchführung eines Konformitätsbewertungsverfahrens notwendig, was voraussetzt, dass das Gerät allen anwendbaren EU-Richtlinien entspricht. Für Medizinprodukte bestimmter Klassen ist zusätzlich eine Zertifizierung durch eine „Benannte Stelle“ notwendig. Diese führt je nach Wahl des Herstellers entweder eine Baumusterprüfung des Gerätes durch oder auditiert und überwacht das vollständige Qualitätssicherungssystem des Herstellers.

In den USA kontrolliert die *Food and Drug Administration (FDA)* bzw. deren *Center for Devices and Radiological Health (CDRH)* die Inverkehrbringung von Medizinprodukten. Medizinische Geräte werden auch hier in Klassen entsprechend ihres Risikos eingeteilt. Die meisten Geräte der höchsten Risikoklasse sowie nicht klassifizierte Geräte benötigen einen *Pre-market approval (PMA)*. Dabei handelt es sich um einen Prozess der FDA, der das Risiko und die Wirksamkeit eines Gerätes überprüft. Dies ist ein sehr aufwendiger Prozess, da ein empirischer wissenschaftlicher Nachweis geführt werden muss.

Ist ein medizinisches Gerät bereits klassifiziert worden und benötigt keine PMA, so muss für ein neu zuzulassendes Gerät nur eine *Pre-market Notification* nach „510k“ durchgeführt werden. Bei der 510k handelt es sich um eine Ähnlichkeitserklärung, die besagt, dass das neue Gerät mindestens so wirksam und sicher ist, wie ein bereits zugelassenes. Für die meisten Geräte der untersten Risikoklasse genügt die Einhaltung der *Good Manufacturing Practice (GMP)* und eine Registrierung bei der FDA.

Der Nachweis der Konformität zu grundlegenden Anforderungen der rechtlichen Regulierung ist aufwendig. Daher existieren Normen, die mit den rechtlichen Richtlinien harmonisiert sind. Das bedeutet, dass bei einem Einhalten dieser Normen von Konformität ausgegangen wird: Eine Konformitätsvermutung liegt vor.

Die ISO-Norm 13485 [28] legt Anforderungen an ein Qualitätsmanagementsystem von Medizinprodukten fest. Sie harmoniert mit der allgemeinen Qualitätsmanagementnorm ISO 9001 und wird meist als erster Schritt des Zulassungsverfahrens betrachtet. Ein Risikomanagement ist hier vor allem in der Produktentwicklungszeit vorgesehen. Für die Zeit danach sind entsprechende Normen, wie beispielsweise ISO 14971 [29], zu erfüllen. Diese spezifiziert einen Prozess für Hersteller, um Gefahren von medizinischen Geräten zu identifizieren, die dazugehörigen Risiken abzuschätzen und zu kontrollieren sowie die Effektivität dieser Kontrolle zu überwachen. Der Prozess soll dabei den ganzen Lebenszyklus eines Gerätes, einschließlich der Zeit nach seiner Produktion und Inverkehrbringung, umfassen. Wird ein Risikomanagement nach ISO 14971 durchgeführt, so gilt die Konformitätsvermutung im Sinne des MPG.

Ein Hersteller muss seine Risikomanagement-Aktivitäten laut ISO 14971 nach einem selbst zu erstellenden Risikomanagementplan durchführen. Dieser beinhaltet alle Aktivitäten des Risikomanagements und gibt an, in welcher Phase des Prozesses welche Aktivität durchzuführen ist. Unter anderem werden Verantwortungen verteilt sowie Anforderungen für Überprüfungen und Kriterien für akzeptable Risiken festgelegt.

Die Norm enthält eine schematische Darstellung des Risikomanagement-Prozesses. Als Erstes wird eine Risikoanalyse durchgeführt. Sie dient der Identifizierung von Gefährdungen und der Einschätzung der Risiken für jede Gefährdungssituation. Anschließend folgt eine Risikobewertung, woraufhin eine Risikominderung bzw. Maßnahmen zur Risikobeherrschung durchgeführt werden. Es folgen die Bewertung von Restrisiken und die Erstellung von Risiko-Nutzen-Analysen.

Darauf aufbauend wird die Akzeptanz des Gesamt-Restrisikos bewertet. Der Hersteller entscheidet anhand der im Risikomanagementplan aufgestellten Kriterien, ob das Gesamt-Restrisiko für das Medizinprodukt akzeptabel ist. Ist dies zu bejahen, so wird ein Risikomanagementbericht erstellt und der Risikomanagementakte hinzugefügt. Nach der Entwicklung, d. h. während und nach der Produktion, soll der Hersteller Informationen über Vorkommnisse beim Betrieb des Produktes zusammentragen. Diese sind hinsichtlich neuer Risiken bzw. Änderungen der Risikobewertung durchzusehen.

Das Risikomanagement nach ISO 14971 ist für Medizingerätehersteller gedacht, die ein Medizinprodukt in Verkehr bringen wollen. Bei Medizinprodukten kann es sich auch um Systeme handeln, die aus mehreren Einzelkomponenten bestehen. Es kommt nur darauf an, dass der Hersteller das Gesamt-Restrisiko verantworten kann. Dementsprechend ist keine Teilzulassung möglich.

Eine Besonderheit in Deutschland ist die durch das MPG erlaubte In-Haus-Herstellung [11], die in der EU-Richtlinie nicht enthalten ist. Hierbei kann eine Gesundheitseinrichtung Medizinprodukte bzw. Systeme für die Verwendung in eigenen Räumen herstellen. Wie bei der Inverkehrbringung durch einen Hersteller muss jedoch der Betreiber ein Konformitätsbewertungsverfahren durchführen. Hierbei wird zugunsten des Betreibers auf die CE-Kennzeichnung und die Einbindung einer „Benannten Stelle“ verzichtet.

Trotz der Problematik der Gesamt-Restrisikobewertung existieren Systeme, in denen Produkte unterschiedlicher Hersteller miteinander verbunden sind. Üblicherweise bestehen diese Systeme überwiegend aus Geräten eines Herstellers, sodass dieser in der Lage ist, ein Gesamt-Restrisiko zu bestimmen. Er ist auch für die Integration von weiteren Fremdgeräten verantwortlich, die meist erst nach und nach für sein System zugelassen werden. Bei den vorhandenen Integrationskonzepten handelt es sich im Allgemeinen um proprietäre Systeme.

International steigt jedoch der Druck, mehr Interoperabilität zwischen medizinischen Geräten unterschiedlicher Hersteller zu ermöglichen. Krankenhausbetreiber wollen gemäß der „Best-of-Breed“-Strategie für jeden Teilbereich das jeweils beste Medizinprodukt verwenden und nicht zwangsweise alle Geräte von einem Hersteller beziehen.

Das Risikomanagement eines Herstellers nach ISO 14971 deckt jedoch nur Systeme ab, die aus Medizinprodukten bestehen, die für eine gemeinsame Nutzung zugelassen wurden. Bei Systemen, die aus Medizinprodukten unterschiedlicher Hersteller bestehen, die nicht für eine gemeinsame Nutzung zugelassen wurden, kann ein Gesamt-Restrisiko nach ISO 14971 nicht von einem einzelnen Hersteller bestimmt werden. Ein Risikomanagement für solche Netzwerke kann nur von dem Betreiber des Netzwerks, der Gesundheitseinrichtung selbst, durchgeführt werden. Hierfür befindet sich unter dem Titel *„Application of risk management for IT-networks incorporating medical devices“* ein Normenvorschlag für IEC 80001 in Arbeit, der momentan als *Committee Draft* vorliegt [74].

IEC 80001 soll ISO 14971 ergänzen und beginnt dort, wo das Risikomanagement eines Herstellers aufhört. Werden in einem isolierten Netzwerk ausschließlich medizinische Geräte nach Vorgabe eines Medizingeräteherstellers betrieben, so erfolgt das Risikomanagement beim Hersteller nach ISO 14971. Das ist auch der Fall, wenn Fremdgeräte von diesem Hersteller für sein isoliertes Netzwerk zugelassen wurden. In gemeinsam genutzten Netzwerken dagegen soll die IEC 80001 zur Anwendung kommen. Hier hat der Betreiber das Risikomanagement zu erbringen, der Gerätehersteller muss ihm

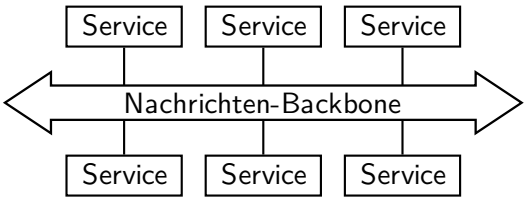



Abbildung 1.3: Services sind miteinander über ein Nachrichten-Backbone verbunden, um miteinander zu kommunizieren

jedoch die hierfür notwendigen Restrisiko-Informationen bereitstellen, die über sein Risikomanagement nach ISO 14971 hinausgehen.

## 1.2 Service-orientierte Architekturen

Interoperabilitätsprobleme gibt es in vielen Bereichen. Bei der Vernetzung von IT-Systemen haben sich Service-orientierte Architekturen (SOAs) durchgesetzt. Bei einer SOA handelt es sich um ein Integrationskonzept, bei dem alle Systeme über eine standardisierte Schnittstellentechnologie als „Service“ in das Gesamtsystem eingebunden werden.

Dieses Konzept stammt aus dem Unternehmensumfeld. Betrieb und Wartung von IT-Systemen sind dort sehr aufwendig und damit kostenintensiv. Die Geschäftsprozesse des Unternehmens müssen auf die IT-Systeme abgebildet werden. Änderungen an Geschäftsprozessen führen zu neuen IT-Systemen, die oft neue Schnittstellen haben. Eine Datenmigration und Abschaltung alter IT-Systeme ist problematisch, sodass die IT-Landschaft permanent wächst.

Die Services innerhalb einer SOA sind, wie in Abbildung 1.3 dargestellt, über ein gemeinsames Nachrichten-Backbone miteinander verbunden. Ob es sich hierbei um ein Netzwerk oder beispielsweise um einen Kommunikationsserver handelt, ist dabei offen gelassen und hängt von der konkreten Umsetzung ab. Ein Kommunikationsserver ist eine zentrale Komponente, über die alle Nachrichten laufen. Dieser kann beispielsweise Nachrichten zwischenspeichern, falls Empfänger gerade nicht erreichbar sind. Außerdem ist es möglich, Nachrichten von einem proprietären in ein standardisiertes Format zu konvertieren, um so Altsysteme anzubinden. Eine Alternative zur Integration von Altsystemen in eine SOA sind sogenannte Software-Wrapper. Diese sprechen auf der einen Seite das Altsystem an und stellen auf der anderen Seite den Dienst als standardisierten Service im Sinne einer SOA zur Verfügung.

Aus den einzelnen Services können in einem Unternehmen durch sogenannte Orchestrierung Geschäftsprozesse modelliert werden. Unter Orchestrierung ist im Allgemeinen

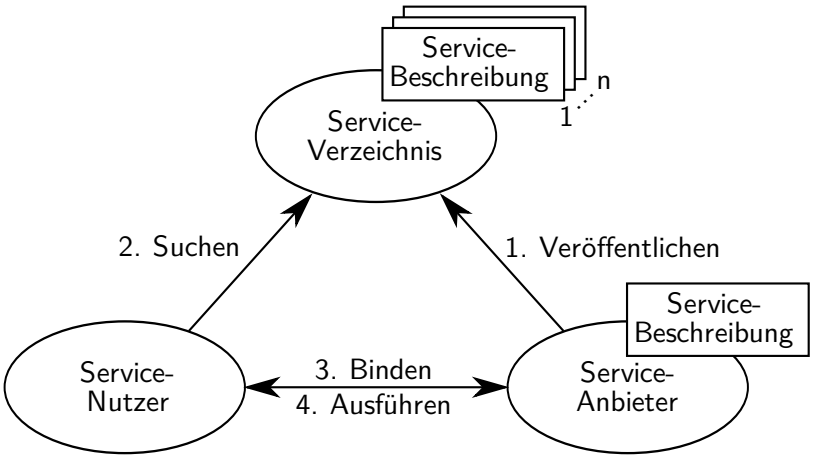



Abbildung 1.4: SOA-Rollenmodell: Interaktionen zwischen den Rollen

das Zusammenfügen einzelner Dienste zu einem neuen, höherwertigen Dienst zu verstehen. Ändern sich Geschäftsprozesse, so können die einzelnen Services wiederverwendet werden und müssen, gegebenenfalls mit weiteren Services, nur neu orchestriert werden. Auf diese Art und Weise ist eine relativ einfache Anpassung der Unternehmens-IT an geänderte Geschäftsprozesse möglich.

Dazu ist es notwendig, dass die Services möglichst lose gekoppelt sind, d. h., dass sie einen möglichst geringen Grad an Abhängigkeit untereinander haben. Haben geringfügige Änderungen eines Service Änderungen an anderen Services zur Folge, so sind sie eng gekoppelt. Sind jedoch nur selten Änderungen an anderen Services notwendig, so sind die Services lose gekoppelt.

In einer SOA gibt es drei Rollen: Service-Anbieter, Service-Nutzer und Service-Verzeichnis. Die Interaktionen der Rollen untereinander ist im sogenannten SOA-Rollenmodell in Abbildung 1.4 dargestellt. Als Erstes veröffentlichen Service-Anbieter ihre Service-Beschreibung in einem Service-Verzeichnis, um ihren Dienst bekannt zu machen. Ohne diesen Schritt wären sie für Service-Nutzer nicht auffindbar. Anschließend haben Service-Nutzer die Möglichkeit, in dem Service-Verzeichnis nach einem passenden Service-Anbieter zu suchen. Der Service-Nutzer erhält die Service-Beschreibung und die Adresse, unter welcher der Dienst angeboten wird. Mithilfe der Service-Beschreibung ist der Service-Nutzer in der Lage, sich an den Service-Anbieter zu binden. Als letzter Schritt erfolgt dann die Ausführung des Service.

Der verwandte Begriff *Service-oriented Device Architecture (SODA)* wurde 2006 von de Deugd et al. definiert [22]. Dabei handelt es sich ebenfalls um das Konzept einer SOA, allerdings werden nicht nur typische Dienste aus dem Unternehmensumfeld bereitgestellt, sondern auch solche von „physikalischen Geräten“. Darunter sind sowohl einfache Sensoren und Aktoren als auch komplexere digitale Geräte und

Steuerungssysteme zu verstehen. Als Beispiele seien hier RFID-Lesegeräte, EKG-Monitore, GPS-Empfänger und Klimaanlagen-Kontrollsysteme genannt.

Eine SODA-Implementierung besteht aus mehreren Komponenten. Ein Geräteadapter kommuniziert mit einem physikalischen Sensor und stellt ein abstraktes Service-Modell zur Verfügung. Ein Bus-Adapter stellt die Daten aus dem Service-Modell im Netzwerk bereit, indem er das abstrakte Service-Modell in der konkreten SOA-Implementierung abbildet. Eine SODA-Implementierung auf einem Gerät kann daher mit einem Software-Wrapper für Altsysteme im Unternehmensumfeld verglichen werden, nur dass in diesem Fall nicht ein Altsystem, sondern ein physikalisches Gerät als Service eingebunden wird.

## 1.3 Motivation

Nach Lesh et al. [82] ist Interoperabilität praktisch ein nicht vorhandenes Merkmal medizinischer Geräte. Zwar existieren einige Standards, jedoch haben diese nicht die gewünschte Interoperabilität erreicht. Standards wie HL7 oder DICOM sind für spezielle Bereiche konzipiert, während ISO/IEEE 11073 von den meisten Herstellern nicht angenommen wurde.

Der Markt wird von herstellerspezifischen Schnittstellen dominiert, die nicht miteinander interoperabel sind. Die volle Funktionalität eines Systems kann meist nur dann erreicht werden, wenn es ausschließlich aus Geräten eines Herstellers besteht. Müssen später neue Geräte hinzugekauft werden, so kommt der sogenannte Lock-in-Effekt zum Tragen: Die neuen Geräte müssen vom gleichen Hersteller gekauft werden, selbst wenn die Konkurrenz bessere Geräte anbietet. Eine komplette Umstellung auf das neue System ist aus Kostengründen oft nicht sinnvoll. Aus Kundensicht ist jedoch die „Best-of-Breed“-Strategie von großem Interesse.

Momentan wird versucht, die Interoperabilitätsprobleme mit Konverter-Boxen zu lösen. Auf diese Weise lassen sich einzelne Punkt-zu-Punkt-Verbindungen zwischen Geräten herstellen. Sie sind nicht nur für herstellerübergreifende Verbindungen notwendig, sondern teilweise auch für Geräte des selben Herstellers. Zukäufe bzw. Übernahmen sowie neue Gerätegenerationen sorgen dafür, dass bei einem Hersteller durchaus mehr als ein Protokoll im Einsatz ist.

Unternehmen wie *Capsule Technologie* [14] haben sich darauf spezialisiert, Hard- und Software zu entwickeln, um nicht kompatible Geräte verschiedener Hersteller miteinander zu verbinden. Mit Adapterkabeln und Modulen lassen sich Geräte mit seriellem Ausgang physikalisch anbinden, während die *DataCaptor Device Interfaces (DDIs) Library* die proprietären Protokolle übersetzen kann. Auf diese Weise lassen sich über 400 unterschiedliche Medizingeräte miteinander verbinden.

Die Kosten für Geräteintegration sind immens. Eine Analyse [78] von Kaiser Permanente kommt zu dem Ergebnis, dass in ihrem Unternehmen die Kosten zur Geräteintegration momentan etwa 40 % der Kosten eines Gerätes ausmachen, das entspricht jährlich etwa 40 Millionen USD für die nächsten 10 Jahre. Ein Standard für Geräte-Kommunikation könne die Kosten um bis zu 30 % reduzieren, was einer Summe von 12 Millionen USD jährlich gleich kommt.

Mehrere Initiativen beschäftigen sich mit der Interoperabilität im medizinischen Bereich. Zu nennen ist beispielsweise die IHE *(Integrating the Healthcare Enterprise)* [63] als eine Organisation, die sich zum Ziel gesetzt hat, den Nachrichtenaustausch zwischen medizinischen Systemen zu verbessern. Dazu werden in einem Prozess sogenannte IHE-Profile erarbeitet, die präzise angeben, wie existierende Standards (z. B. DICOM, HL7, W3C-Standards) zu verwenden sind.

Das *Medical Device Plug-and-Play (MD-PnP) Program* [88] wurde 2004 vom *Massachusetts General Hospital (MGH)* und dem *Center for Integration of Medicine and Innovative Technology (CIMIT)* initiiert. Zwei Aspekte von Interoperabilität werden hier behandelt: Zum einen der bidirektionale Datenaustausch zwischen elektronischen Patientenakten und medizinischen Geräten, der zu einer umfangreichen Datenerfassung beiträgt und dadurch Fernüberwachungen und Entscheidungsunterstützungssysteme *(Decision Support Systems)* ermöglicht. Zum anderen eine Gerätesteuerung zur Integration mehrerer Medizingeräte in ein fehlerresistenteres Gesamtsystem durch sogenannte *Safety Interlocks*. Safety Interlocks zwischen medizinischen Geräten sorgen dafür, dass die Geräte sich gegenseitig so überwachen und steuern, dass eine erhöhte Patientensicherheit ermöglicht wird.

Im Anwendungsfall von Arney et al. [7] wird während einer Operation ein Röntgenbild benötigt. Um eine möglichst scharfe Aufnahme des Röntgenbildes zu erreichen, dürfen sich Bauch und Brustkorb des Patienten nicht bewegen, d. h., die künstliche Beatmung muss unterbrochen werden. Typischerweise hält der Anästhesist die Beatmung an, sodass die Aufnahme gemacht werden kann. Anschließend muss er die künstliche Beatmung wieder fortsetzen. Dabei entsteht dem Patienten im Normalfall kein Schaden.

Dieser Anwendungsfall bezieht sich auf einen Artikel von Lofsky [86], in dem von einem Unglücksfall während einer Cholezystektomie (Gallenblasenentfernung) berichtet wird. Aufgrund unglücklicher Zufälle hatte es der Anästhesist versäumt, die Beatmung nach der Röntgenaufnahme rechtzeitig wieder zu aktivieren, sodass eine 32-jährige Patientin starb. In diesem Fall kann eine automatische Synchronisation von Anästhesie- und Röntgengerät die Patientensicherheit erhöhen, da die Beatmung – wenn überhaupt – nur kurzzeitig unterbrochen werden muss und auf jeden Fall wieder automatisch fortgesetzt wird.

Interoperabilitätsprobleme sind nicht spezifisch für den medizinischen Bereich. Bei Computersystemen, insbesondere im Unternehmensumfeld, werden SOAs erfolgreich

eingesetzt und sind dort Stand der Technik. Die Einführung einer SOA im medizinischen Bereich, insbesondere für die Geräte-Kommunikation, schafft die besten Voraussetzungen, um die Interoperabilitätsprobleme zu beseitigen.

Anstatt unterschiedliche proprietäre Schnittstellen zu verwenden, bieten alle Systeme ihre Dienste über eine einheitliche Service-Schnittstelle an. Entsprechend dem „SODA-Gedanken“ handelt es sich bei den Systemen nicht nur um Computersysteme, sondern auch um medizinische Geräte. Durch die lose Kopplung der Services können vorhandene Services bzw. Geräte auf einfache Weise ausgetauscht und neue hinzugefügt werden. Anstatt dabei Geschäftsprozesse zu modellieren, können medizinische Arbeitsabläufe umgesetzt werden. Als Beispiel sei hier der oben genannte Anwendungsfall zu erwähnen, in dem eine Röntgenaufnahme bei einem künstlich beatmeten Patienten angefertigt werden soll.

Auf Herstellerseite führt die Einführung einer SOA zu einer Modularisierung des Geräteparks. In Komplettsystemen können einzelne Geräte durch neue Generationen ausgetauscht werden, ohne dass Anpassungen an anderen Geräten notwendig sind.

Wird eine SOA mit herstellerübergreifenden Gerätebeschreibungen verwendet, so haben die Betreiber die Möglichkeit, nach der „Best-of-Breed“-Strategie, für jede Geräteklasse das jeweils geeignetste Gerät, unabhängig vom Hersteller, zu beziehen. Auf diese Art entstehen Gesamtsysteme aus Geräten unterschiedlicher Hersteller, die ein Risikomanagement nach dem sich im Entwurf befindenden IEC 80001 benötigen. Die Hersteller liefern dann nur noch „Teile“ des Gesamtsystems, sodass man auf Herstellerseite von „Teilzulassungen“ sprechen kann. Schließlich ist der Betreiber für die abschließende Gesamt-Restrisikobewertung verantwortlich, die auf den vom Hersteller bereitgestellten Informationen über die einzelnen Geräte aufsetzt.

SOAs für ein technisches Umfeld, wie die Vernetzung medizinischer Geräte und Systeme, liefern neue Herausforderungen. Verglichen damit sind im klassischen SOA-Umfeld bei der Anwendung in Unternehmen eher wenige, größere Services in relativ statischen Kontexten im Einsatz.

Eine SOA zur vernetzten Kommunikation zwischen medizinischen Geräten, Systemen und Applikationen muss mit einer komplexen und gleichzeitig dynamischen Service-Struktur umgehen können. Durch jedes Ein- oder Ausschalten eines Gerätes tauchen Services im Netzwerk auf bzw. verschwinden wieder. Hinzu kommt ein hoher Anspruch an die Dienstgüte der angebotenen Services im Netzwerk. Im Gegensatz zu Unternehmensanwendungen, bei denen es oft um die Gesamtperformance geht, muss im medizinischen Bereich jede einzelne Aktion für sich garantiert funktionieren. Außerdem ist ein allumfassendes Sicherheitskonzept durch die Anzahl verschiedener Hersteller für das Risikomanagement eines Betreibers unverzichtbar.

Wenn es um die Umsetzung einer SOA zum Vernetzen von medizinischen Geräten oder technischen Systemen im Allgemeinen geht, entstehen neue Fragestellungen, die in dieser Form bei Unternehmensanwendungen nicht vorhanden sind.

## 1.4 Zielsetzung

In dieser Arbeit geht es um die Entwicklung einer SOA zur vernetzten Kommunikation zwischen medizinischen Geräten, Systemen und Applikationen. Im Vordergrund steht dabei, eine interoperable und herstellerübergreifende SOA-Lösung zu entwickeln, die – wann immer möglich – auf etablierte Standard-Technologien zurückgreift.

Weitverbreitete Standard-Technologien sind üblicherweise generisch gehalten, sodass nicht für alle Probleme bereits Lösungen entwickelt wurden. Diese Arbeit beschäftigt sich mit den Problembereichen, die in einer SOA für technische Systeme noch nicht abgedeckt werden. Zusätzlich kommen noch Sicherheitsaspekte für das Risikomanagement hinzu, welche im medizinischen Bereich unerlässlich sind.

Um die Verwendung von medizinischen Geräten zu vereinfachen, soll die SOA soweit wie möglich Plug-and-Play (PnP)-Prinzipien umsetzen. Dazu gehören neben einer Adressierung der Geräte insbesondere deren Auffinden, sowie die Möglichkeit, eine Gerätebeschreibung abzufragen. Anschließend kann eine Gerätesteuerung sowie eine Datenverteilung darauf aufsetzen.

Die zu entwickelnde SOA soll technische Interoperabilität zwischen medizinischen Geräten und Systemen jeglicher Art gewährleisten. Sie dient als Basis für mögliche weiterreichende semantische Interoperabilität, die nicht Bestandteil dieser Arbeit ist. Für semantische Interoperabilität könnten beispielsweise die Nomenklatur und das DIM der ISO/IEEE 11073 sowie das Referenzinformationsmodell von HL7 v3 verwendet werden.

## 1.5 Aufbau dieser Arbeit

Im nächsten Kapitel folgen Grundlagen zu Web Services, mit denen sich eine interoperable SOA umsetzen lässt, da sie weitverbreitete Internet-Technologien benutzen. Als Erstes wird ein Überblick über existierende Web-Service-Standards gegeben. Anschließend werden die drei Basistechnologien vorgestellt, mit den sich das SOA-Rollenmodell abbilden lässt. Das Kapitel schließt mit dem Thema Web-Service-Profile ab. Diese gewährleisten Interoperabilität zwischen verschiedenen Implementierungen. Insbesondere das Geräte-PnP-Konzept des *Device Profile for Web Services (DPWS)* spielt eine wesentliche Rolle für die zu entwickelnde SOA.

Kapitel 3 beschreibt die Entwicklung einer Web-Service-Discovery-Architektur, die in der entwickelten SOA die Rolle des Verzeichnisdienstes übernimmt. Damit ist die Discovery-Architektur elementarer Bestandteil der SOA, da Services erst miteinander kommunizieren können, nachdem sie einander gefunden haben. Entsprechend der PnP-Prinzipien soll das Auffinden von Services ohne Konfiguration einzelner

Geräte funktionieren. Nach der Architekturbeschreibung werden zwei verschiedene Realisierungen vorgestellt und ihre Eignung für die SOA bewertet.

Das vierte Kapitel befasst sich mit der Optimierung der Dienstgüte im Netzwerk. Die Kommunikation zwischen Services muss grundsätzlich funktionieren und sollte nicht von anderen Netzwerkteilnehmern gestört werden, d. h. auch bei einem Fehlverhalten dieser. Vor allem in Verbindung mit Web Services sollte dies transparent ermöglicht werden, da Web Services typischerweise keinen Einfluss auf die darunterliegenden Schichten haben. Es wird eine Switch-Architektur vorgestellt, die entwickelt wurde, um Übertragungsgarantien zu liefern und so unter anderem eine zuverlässige Gerätesteuerung in der SOA zu ermöglichen.

Kapitel 5 beschäftigt sich mit der Entwicklung eines Sicherheitskonzepts. In einem gemeinsam genutzten Netzwerk mit vielen unterschiedlichen medizinischen Geräten und Computern ist ein Sicherheitskonzept für das Risikomanagement unerlässlich. Nicht jeder Teilnehmer in einem solchen Netzwerk darf jeden vorhandenen Service nutzen. Gerade die Services, die direkte Auswirkungen auf einen Patienten haben, müssen vor unbefugtem Zugriff geschützt werden. Die hierfür entwickelte Zugriffskontrolle zeichnet sich durch ihren dezentralen Ansatz aus, der unter anderem PnP ermöglicht.

Im sechsten Kapitel wird die Performance von Web Services mit der einer Publish-Subscribe-Middleware für Echtzeitsysteme verglichen. Die Evaluierung wird anhand von zwei Szenarien durchgeführt. Bei dem einen Szenario handelt es sich um die Fernsteuerung von Geräten, während das andere eine Verteilung von Messwerten beschreibt. Hier geht es um unterschiedliche Übertragungsarten, die in der SOA für verschiedene Bereiche benötigt werden.

Im letzten Kapitel werden die einzelnen Teile dieser Arbeit zusammengefasst, die daraus aufgebaute SOA als Ganzes vorgestellt und ein Ausblick über mögliche weiterführende Arbeiten gegeben.

# Kapitel 2

# Web-Service-Grundlagen

Web Services bieten standardisierte Kommunikation zwischen verschiedenen Software-Anwendungen unabhängig von den darunterliegenden Plattformen. Sie zeichnen sich durch ihre Interoperabilität und Erweiterbarkeit sowie eine maschinenlesbare Service-Beschreibung aus. Daher eignen sich Web Services für die Umsetzung einer SOA.

Web Services basieren auf Technologien wie beispielsweise *Extensible Markup Language (XML)* [183], die aus dem *World Wide Web (WWW)* bekannt sind. Die Standardisierung wird vornehmlich vom *World Wide Web Consortium (W3C)* [193] und von der *Organization for the Advancement of Structured Information Standards (OASIS)* [130] vorangetrieben. Das W3C wurde 1994 von Tim Berners-Lee gegründet, um das langfristige Wachstum des WWWs durch die Entwicklung von Standards und Richtlinien sicherzustellen. OASIS wurde 1993 als Konsortium gegründet, um Richtlinien für den XML-Vorläufer *SGML (Standard Generalized Markup Language)* [66] zu entwickeln.

Während sich das W3C in erster Linie mit der Architektur und den Basistechnologien von Web Services beschäftigt, kümmert sich die OASIS eher um Web-Service-Erweiterungen. Dies macht sich dadurch bemerkbar, dass die Spezifikationen vom W3C eng aufeinander abgestimmt sind, während bei der OASIS die Spezifikationen primär auf den Kernspezifikationen der W3C-Architektur aufbauen.

Im folgenden Abschnitt wird der Web-Service-Protokollstapel des W3C vorgestellt. Es handelt sich dabei um ein Architekturkonzept, das prinzipiell keine bestimmte Technologie vorschreibt. Es werden jedoch für die einzelnen Bereiche entsprechende Spezifikationen vorgestellt, um so einen Überblick zu erlangen.

Im zweiten Abschnitt folgen die Basistechnologien, mit denen sich das Konzept einer SOA mit Web Services umsetzen lässt. Als Nachrichtenformat wird SOAP verwendet, die Service-Beschreibung erfolgt mithilfe von WSDL, während UDDI die Rolle des Verzeichnisdienstes einnimmt.

Im letzten Abschnitt geht es um Interoperabilitätsprofile für Web Services. Prinzipiell sind Web Services interoperabel, jedoch führen unterschiedliche Interpretationen eines Standards unter Umständen zu inkompatiblen Systemen.

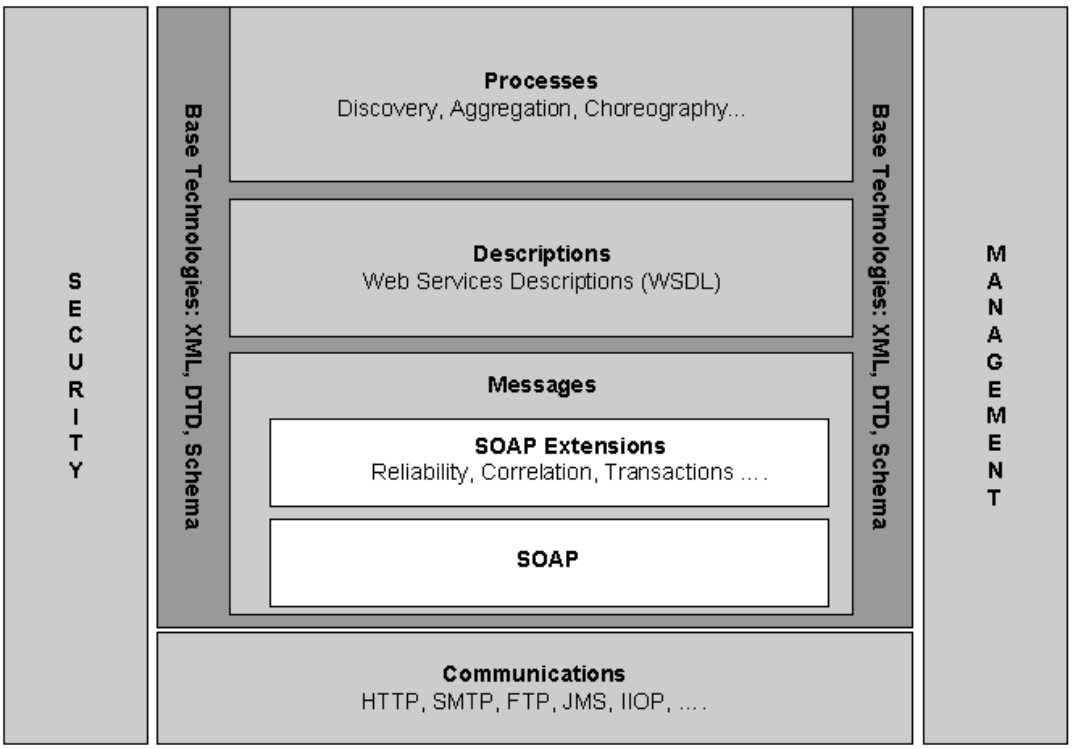



Abbildung 2.1: Web-Service-Protokollstapel aus [190]

## 2.1 Übersicht über den Web-Service-Protokollstapel

Die *Web Services Architecture Working Group* des W3C beschäftigt sich mit der Architektur von Web Services. Sie definiert Begriffe und deren Beziehungen untereinander, um in abstrakten Modellen einzelne Aspekte der Web-Service-Architektur zu beschreiben. Die Architektur im Überblick ist als Protokollstapel in Abbildung 2.1 zu sehen.

Die *Communications*-Schicht ist zu unterst im Stapel angeordnet. Sie bildet die Grundlage zum Nachrichtenaustausch zwischen Web Services. Üblicherweise werden hier Internetprotokolle wie HTTP, SMTP und FTP verwendet, von denen HTTP jedoch am weitesten verbreitet ist. Prinzipiell ist aber kein bestimmtes Protokoll vorgeschrieben, sodass theoretisch auch andere Protokolle verwendet werden können.

Strenggenommen ist *Java Message Service (JMS)* [159] zwar ein Java API (*Application Programming Interface*) für ein Kommunikationsprotokoll, jedoch wird es oft mit einem Kommunikationsprotokoll gleichgesetzt. Das *Internet Inter-ORB Protocol (IIOP)* wurde für die *Common Object Request Broker Architecture (CORBA)* [127] spezifiziert, um dort eine Kommunikation über das Internet zu ermöglichen.

Bei JMS sowie bei IIOP handelt es sich um sehr spezielle Protokolle, die nur für interne Web-Service-Kommunikation nützlich sind. Schließlich können Web Services nur miteinander kommunizieren, wenn sie das gleiche Kommunikationsprotokoll verwenden. Sollen trotzdem JMS, IIOP oder andere Protokolle verwendet werden, so kann ein Kommunikationsserver Nachrichten von einem Kommunikationsprotokoll in ein anderes überführen.

Über der *Communications*-Schicht wird XML mit den dazugehörigen Beschreibungssprachen *Document Type Definition (DTD)* [183] und *XML Schema* [174] als Basis für die weiteren Schichten verwendet. XML Schema wird hierbei gegenüber DTD zunehmend bevorzugt, da es eine mächtigere Semantik sowie elementare und komplexe Datentypen unterstützt.

Die *Messages*-Schicht baut auf der *Communications*-Schicht auf. Diese Schicht behandelt Nachrichten im Allgemeinen und SOAP als Nachrichtenformat im Speziellen. Dies wird im folgenden Abschnitt 2.2 über Basistechnologien detailliert vorgestellt. Die Arbeitsgruppe des W3C stellt in ihrer Architektur ein nachrichtenorientiertes Modell vor. Es behandelt alle Aspekte einer Nachricht und ihrer Verarbeitung, jedoch nicht die Semantik oder die Beziehung der Nachrichten untereinander. Obwohl die Arbeitsgruppe des W3C davon ausgeht, dass es auch Alternativen zu SOAP geben kann, rät sie dennoch aufgrund von möglichen Interoperabilitätsproblemen stark davon ab, etwas anderes zu verwenden. SOAP ist somit die Kern-Technologie von Web Services: ein plattform- und sprachunabhängiges Nachrichtenformat.

SOAP wurde als erweiterbares Nachrichtenformat konzipiert und unterstützt somit auf Nachrichtenebene diverse Erweiterungen. Eine weitverbreitete Erweiterung ist *Web Services Addressing (WS-Addressing)* [177]. Mit WS-Addressing ist es möglich, unabhängig von dem darunter liegenden Protokoll in der *Communications*-Schicht, Web-Services zu adressieren. Dazu gehören beispielsweise ein Empfänger, eine Nachrichten-ID oder ein Absender.

Um Nachrichten zuverlässig zu übertragen, gibt es die konkurrierenden OASIS-Standards *Web Services ReliableMessaging (WS-RM) v1.2* [121] und *WS-Reliability (WS-R) v1.1* [103]. Diese garantieren den übergeordneten Schichten beispielsweise, dass eine Nachricht genau einmal beim Empfänger ankommt. Für Transaktionen gibt es hingegen *WS-AtomicTransaction v1.2* [119].

Über der Nachrichtenschicht ist die *Descriptions*-Schicht angesiedelt. Spezifikationen dieser Schicht sind dafür zuständig, einen Web Service zu beschreiben. Erst durch eine möglichst genaue Beschreibung eines Web Services ist es möglich, ihn aufzurufen. Die Arbeitsgruppe hat hier ein Service-orientiertes Modell entwickelt, in dem es um den Service selbst und um die Aktionen geht, welche ausgeführt werden können. Als Aktion ist dabei alles zu verstehen, was durch einen Nachrichtenempfang ausgelöst wird oder zu einem Nachrichtenversand führt. Es wird also angegeben, welche Nachrichten verschickt werden und wie diese miteinander zusammenhängen. Auf diese Weise

lassen sich sogenannte Web-Service-Operationen oder auch -Methoden definieren. Typischerweise wird hierfür die *Web Service Description Language (WSDL)* [171] verwendet, die ebenfalls im Abschnitt 2.2 zum Thema Basistechnologien erläutert wird.

Die Beschreibung eines Web Services kann um Richtlinien erweitert werden. Diese werden im sogenannten Richtlinien-Modell beschrieben. Hierfür hat das W3C das *Web Services Policy 1.5 – Framework (WS-Policy)* [182] geschaffen. Damit ist es beispielsweise möglich, Sicherheitsanforderungen in die Beschreibung eines Dienstes aufzunehmen.

Oberhalb der *Descriptions*-Schicht ist die *Processes*-Schicht angesiedelt. Dort geht es um Prozessabläufe, wie beispielsweise das Auffinden von Web Services. Die OASIS sieht hierfür die *Universal Description, Discovery and Integration (UDDI)* [124] vor. UDDI wird als dritte Komponente für die Umsetzung einer SOA im folgenden Abschnitt beschrieben.

Ein weiterer Teil der *Processes*-Schicht ist die Dienstkomposition von Services. Diese lässt sich in Orchestrierung und Choreographie unterteilen. Bei der Orchestrierung werden einzelne Web Services zu höherwertigen Services aggregiert, um so etwa Geschäftsprozesse zu modellieren. Mit der *Business Process Execution Language (WS-BPEL)* [114] lassen sich diese Prozesse in XML beschreiben, die anschließend von einer BPEL-Engine ausgeführt werden können.

Die Choreographie verfolgt einen etwas anderen Ansatz. Anstatt einen ausführbaren Prozess zu modellieren, soll der Nachrichtenaustausch zwischen verschiedenen Web Services beschrieben werden. Auf diese Weise lässt sich die Zusammenarbeit mehrerer Web Services von einem globalen Standpunkt aus beschreiben. Spezifiziert wird dies in der *Web Services Choreography Description Language (WS-CDL)* [167], auch WS-Choreography oder WS-Chor genannt.

Einige Erweiterungen lassen sich nicht eindeutig in die Schichten des Protokollstapels einsortieren, da sie in mehreren Ebenen vertreten sind. Im Web-Service-Protokollstapel aus Abbildung 2.1 sind daher die Bereiche *Security* und *Management* seitlich angegliedert.

Auf der linken Seite befindet sich der *Security*-Bereich. In der Kommunikationsschicht kann beispielsweise HTTPS [151] anstelle von HTTP verwendet werden. HTTPS verwendet *Transport Layer Security (TLS)* [27], um einen sicheren Kanal zwischen zwei Knoten aufzubauen, in welchem der HTTP-Transfer geschützt stattfinden kann.

Wird dagegen in den auf XML basierenden Schichten Sicherheit benötigt, so muss auf *XML Encryption* [172] und *XML Signature* [184] zurückgegriffen werden. Beide Standards haben mit Web Services per se nichts zu tun, bilden aber die Grundlage für *Web Services Security (WS-Security)* [111]. WS-Security beschreibt, wie mithilfe von *XML Encryption* und *XML Signature* eine SOAP-Nachricht verschlüsselt und signiert

werden kann. Zusätzlich ist in Security-Profilen spezifiziert, wie weitere Informationen, beispielsweise Sicherheitszertifikate oder andere Sicherheitstokens, im SOAP-Header platziert werden müssen. WS-Security bildet somit die Kernkomponente, was Sicherheit bei Web Services betrifft.

Auf WS-Security setzen *WS-Trust* [116], *WS-SecureConversation* [115] und *Web Services Federation Language (WS-Federation)* [85] auf. In WS-Trust ist ein Framework spezifiziert, das vorgibt, auf welche Art ein *Security Token Service (STS)* umgesetzt werden kann. Dieser ermöglicht es, als Vermittler Sicherheitstokens zu vergeben. WS-SecureConversation spezifiziert Erweiterungen, um einen gemeinsamen Sicherheitskontext zu ermöglichen. Von diesem lassen sich *session keys* für eine Verschlüsselung einzelner Nachrichten ableiten. WS-Federation setzt die Idee einer *Federated Identity* für Web Services um. Sicherheitsinformationen können über Sicherheitsdomänen hinweg ausgetauscht werden, um so fremden Services Zugriff auf eingene Ressourcen zu gewähren.

Welche Sicherheitsanforderungen ein Web Service an seine Aufrufer stellt, lässt sich mithilfe der *WS-SecurityPolicy* [122] in die Web-Service-Beschreibung einbetten. Es kann zum Beispiel angegeben werden, dass Sicherheit entweder auf Transportebene oder auf Nachrichtenebene mit Verschlüsselung benötigt wird, um so Vertraulichkeit zu gewährleisten.

Der *Management*-Bereich des Protokollstapels beschäftigt sich mit der Verwaltung von Web Services und den von ihnen angebotenen Ressourcen. Es gibt verschiedene Spezifikationen, die sich mit diesem Thema beschäftigen. Einige davon konkurrieren miteinander, zum Beispiel der *Web Services for Management (WS-Management/WS-Man)* [32] DTMF[1]-Standard mit den beiden OASIS-Standards *Management Using Web Services (MUWS) 1.1* [109] und *Management of Web Services (MOWS) 1.1* [108], die unter dem Namen *Web Services Distributed Management (WSDM)* zusammengefasst werden.

Konkurrierende Spezifikationen, nicht nur im Managementbereich, stellen ein Problem von Web Services dar. Deshalb haben bedeutende Hersteller von Web-Service-Software (HP, IBM, Intel und Microsoft) in Zusammenarbeit eine gemeinsame Roadmap zur Harmonisierung erstellt, die März 2006 veröffentlicht wurde [18]. Ein Überblick von IBM vom Mai 2007 liefert weitere Informationen [56].

Die aktuelle Situation dieser Harmonisierung ist in Abbildung 2.2 dargestellt. Die grau hinterlegten Spezifikationen aus den unteren Schichten befinden sich seit 2009 beim W3C im Standardisierungsprozess der *Resource Access Working Group*.

*Web Services Eventing (WS-Eventing)* [186] beschreibt ein Protokoll, mit dem sich Web-Service-Nutzer bei Web-Service-Anbietern für Ereignisse, sogenannte *Events*, re-

[1] Die DMTF (Distributed Management Task Force, Inc.) [31] ist eine internationale Vereinigung von IT-Unternehmen, um Managementnormen zu entwickeln.

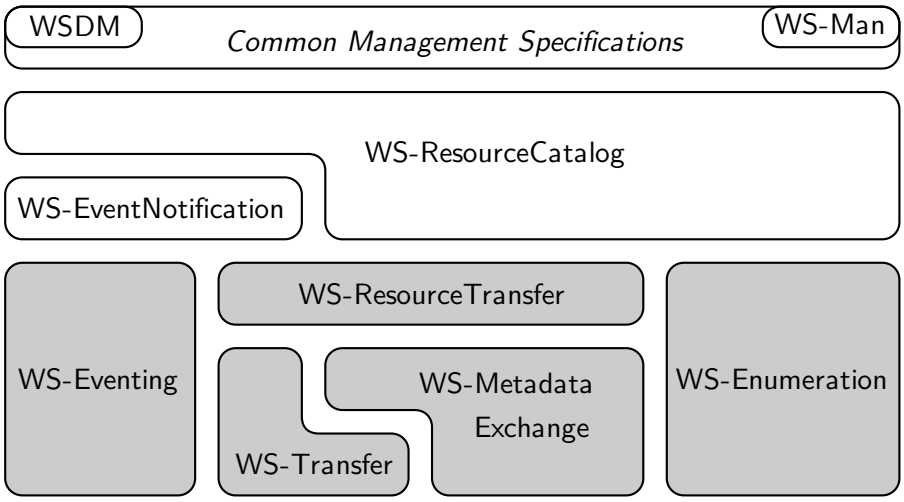



Abbildung 2.2: Spezifikationen für die Verwaltung von Web Services und den von ihnen angebotenen Ressourcen

gistrieren können. Es ist die Basis für die noch nicht spezifizierte *WS-EventNotification*. Diese soll weitere Funktionen aus dem konkurrierenden Ansatz von *Web Services Notification (WSN/WS-Notification)* übernehmen und diesen langfristig ersetzen. WS-Notification besteht aus den drei dadurch zukünftig obsoleten OASIS-Spezifikationen: *Web Services Base Notification (WS-BaseNotification) v1.3* [106], *Web Services Brokered Notification (WS-BrokeredNotification) v1.3* [107] und *Web Services Topics (WS-Topics) v1.3* [113]. Des Weiteren wurde von HP *Web Services Events (WS-Events)* [15] spezifiziert, welches jedoch bereits vor der Harmonisierung zugunsten von WS-Notification aufgegeben wurde.

Mit *Web Services Transfer (WS-Transfer)* [189] wird der Zugriff auf eine Ressource durch eine XML-Repräsentation ermöglicht. Die vier Basis-Operationen sind: *GET*, *PUT*, *DELETE* und *CREATE*. Mit diesen lassen sich Web Services nach den Prinzipien des *Representational State Transfer (REST)* übertragen. Dabei handelt es sich um einen Architekturstil von Fielding [43], der sich insbesondere durch die Zustandslosigkeit der Client/Server-Verbindung auszeichnet, sodass der Server keine Session verwalten muss.

*Web Services Metadata Exchange (WS-MetadataExchange)* [187] nutzt die GET-Operation von WS-Transfer, um Metadaten, wie zum Beispiel die WSDL-Beschreibung eines Web Services, zu übertragen. Die auf WS-Transfer aufbauende *Web Services Resource Transfer (WS-RT/WS-ResourceTransfer)* [188] ermöglicht es, nur Teile der XML-Repräsentation einer Ressource zu übertragen. Im Gegensatz dazu ist die *Web Services Enumeration (WS-Enumeration)* [185] geeignet, um beispielsweise über Datensätze einer Datenbank oder einer Log-Datei zu iterieren. Die WS-ResourceTransfer-

Spezifikation soll das OASIS *Web Services Resource Framework (WSRF)* ablösen. Das WSRF besteht aus fünf Spezifikationen: WS-Resource, WS-ResourceProperties, WS-ResourceLifetime, WS-ServiceGroup und WS-BaseFaults. Diese werden im WSRF-Primer [100] mit Beispielen beschrieben.

Der *Web Services Resource Catalog (WS-ResourceCatalog/WS-RC)* [99] liegt als initialer Entwurf von HP, IBM, Intel und Microsoft vor. Er soll als Basis für eine zukünftige gemeinsame Management-Spezifikation dienen. Wenn möglich, soll diese auf WSDM und WS-Man als Altlasten verzichten, d. h. diese komplett ersetzen.

Zusammenfassend lässt sich sagen, dass die Menge an vorhandenen Web-Service-Spezifikationen unübersichtlich ist. Jedoch ist absehbar, dass sich bei den unterschiedlichen Erweiterungen allgemein anerkannte Standards etablieren werden.

## 2.2 Basistechnologien

Im Web-Service-Bereich gelten die Technologien SOAP, WSDL und UDDI als Basistechnologien zur Umsetzung einer SOA. Für die Kommunikation zwischen den Services über ein gemeinsames Nachrichten-Backbone wird SOAP als Nachrichtenformat verwendet. Für die Beschreibung eines Web Services steht WSDL zur Verfügung, während die Rolle des Verzeichnisdienstes von UDDI eingenommen wird.

Entsprechend dem SOA-Rollenmodell aus der Einleitung (vgl. Abbildung 1.4) sendet der Service-Anbieter im ersten Schritt eine SOAP-Nachricht an den UDDI-Verzeichnisdienst. Die Nachricht enthält neben einer allgemeinen Service-Beschreibung einen Verweis (URL) auf seine Web-Service-Beschreibung (WSDL). Daraufhin kann der Service-Nutzer per SOAP-Nachricht eine Suchanfrage an das UDDI-Verzeichnis senden. Das Suchergebnis beinhaltet für jeden Service-Anbieter neben einer allgemeinen Beschreibung seine Adresse sowie einen Verweis auf seine Web-Service-Beschreibung. Mithilfe der Adresse des Web Services und seiner Web-Service-Beschreibung ist der Service-Nutzer in der Lage, sich an den Service-Anbieter zu binden. Der anschließende Nachrichtenaustausch erfolgt wiederum im SOAP-Format.

### 2.2.1 SOAP

Der Name SOAP war bis einschließlich Version 1.1 [169] eine Abkürzung für *Simple Object Access Protocol (SOAP)*. Im Rahmen der Standardisierung durch die *W3C XML Protocol Working Group* wurde aus dem Akronym ein eigenständiger Name. Obwohl SOAP 1.2 [179] seit 2003 ein W3C-Standard ist, findet die Version 1.1 noch weite Verbreitung.

SOAP 1.2 bietet nicht mehr Funktionalität als SOAP 1.1: Bei den meisten Änderungen handelt es sich nur um Präzisierungen zur Verbesserung der Interoperabilität. Die

Änderungen zwischen SOAP 1.1 und 1.2 führt das W3C in [178, Kap. 6] und [168] auf.

Die Spezifikation von SOAP 1.2 ist in zwei Teile eingeteilt: *Part 1: Messaging Framework* [179] und *Part 2: Adjuncts* [180]. Im ersten Teil geht es um ein abstraktes, erweiterbares und transportunabhängiges Nachrichtenformat. Es wird die Struktur, die Verarbeitung und die Erweiterbarkeit von SOAP-Nachrichten sowie ein sogenanntes *Binding Framework* spezifiziert. Das Binding Framework gibt allgemeine Regeln vor, wie SOAP-Nachrichten an ein Transportprotokoll zu koppeln sind.

Im zweiten Teil werden Ergänzungen zum ersten Teil spezifiziert, um eine konkrete technische Realisierung zu ermöglichen. Es wird angegeben, wie nicht-XML-basierte Daten in SOAP-Nachrichten eingebettet werden können, um so beispielsweise *Remote Procedure Calls (RPC)* zu ermöglichen (*rpc/encoded*). Liegen die Daten bereits in XML vor, so können sie direkt eingebettet werden (*document/literal*). Außerdem wird ein sogenanntes *Transport Binding* für HTTP definiert. In diesem wird angegeben, auf welche Art eine SOAP-Nachricht über das HTTP-Protokoll übertragen wird.

#### Nachrichtenformat

Die Struktur einer SOAP-Nachricht ist in Abbildung 2.3 schematisch dargestellt, wobei die im Standard spezifizierten Teile grau hinterlegt sind. Eine SOAP-Nachricht besteht aus genau einem *Envelope*-Element. Dieses wiederum beinhaltet ein oder zwei Kindelemente: als Erstes ein optionales *Header*-Element, gefolgt von einem verbindlichen *Body*-Element.

Das *Header*-Element kann beliebig viele sogenannte *Header*-Blöcke enthalten. Diese sind im Standard selbst nicht spezifiziert und ermöglichen es anderen Spezifikationen, SOAP zu erweitern. Allerdings sind für *Header*-Blöcke einige Attribute definiert, um Web Services bei der Verarbeitung von Nachrichten zu beeinflussen.

Mit dem *mustUnderstand*-Attribut kann ein Header-Block markiert werden. Dadurch wird angezeigt, ob ein Web Service diesen Eintrag verarbeiten muss, oder ihn ignorieren darf. Auf diese Weise ist es möglich, SOAP-Erweiterungen zu entwickeln, die nicht von Web Services ignoriert werden dürfen, nur weil diese die Erweiterung nicht unterstützen. Standardmäßig ist es Web Services nämlich erlaubt, alle ihnen unbekannten *Header*-Blöcke zu ignorieren.

Das *role*-Attribut wird benötigt, wenn eine SOAP-Nachricht über Zwischenknoten, sogenannte *Intermediaries*, zu ihrem eigentlichen Empfänger, dem *Ultimate Receiver*, gelangt. Mit dem Attribut wird angegeben, welcher Web Service diesen *Header*-Block verarbeiten soll. In SOAP 1.1 heißt das Attribut *actor* und hat die gleiche Bedeutung.

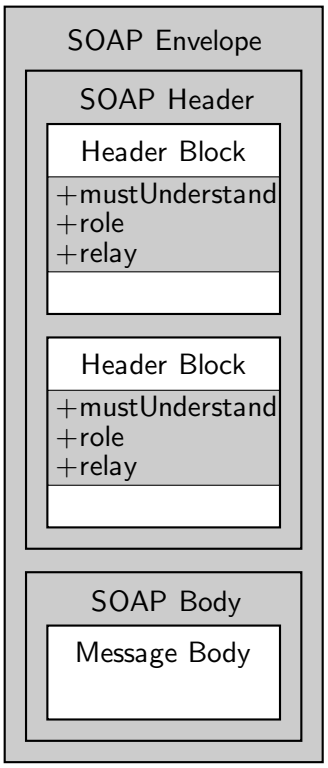



Abbildung 2.3: Schematische Darstellung einer SOAP-Struktur

Sobald ein *Header*-Block seinen Empfänger erreicht hat, wird dieser normalerweise verworfen. Mit dem *relay*-Attribut kann angegeben werden, dass der *Header*-Block weitergesendet werden soll, wenn dieser nicht verarbeitet wurde.

SOAP-1.2-Nachrichten sind als sogenanntes XML Infoset [173] spezifiziert. Sie lassen sich entsprechend XML 1.0 [183] serialisieren. In Version 1.1 ist dagegen die XML-Serialisierung verbindlich. Sinnvoll sind alternative Serialisierungen, wenn beispielsweise Komprimierung benötigt wird oder viele Binärdaten übertragen werden sollen.

Um Binärdaten in Form von Dateianhängen zu übertragen, gibt es *SOAP Messages with Attachments (SwA)* [170] und *SOAP Message Transmission Optimization Mechanism (SOAP-MTOM)* [175], wobei sich letzterer durchgesetzt hat. Bei beiden Spezifikationen werden die SOAP-Nachricht und ihre Anhänge in eine Nachricht vom Typ *MIME multipart/related* [84] verpackt. Ebenso wird mit Dateianhängen bei E-Mails verfahren. Der Unterschied der beiden Spezifikationen liegt in der Art der Referenzierung der Anhänge. In SwA wird der *Content-ID*-MIME-Header nach RFC 2111 [83] verwendet, während für MTOM der W3C-Standard *XML-binary Optimized Packaging (XOP)* [176] vorgesehen ist.

#### Nachrichtenübertragung

Für die Übertragung von SOAP-Nachrichten lässt sich prinzipiell fast jedes Transportprotokoll verwenden. Im zweiten Teil des SOAP-1.2-Standards wird eine konkrete Umsetzung für ein Transport-Binding mit HTTP beschrieben. Dieses befolgt die Regeln aus dem ersten Teil des Standards, die im *SOAP Protocol Binding Framework* angegeben sind.

Es werden zwei verschiedene Arten von sogenannten *Message Exchange Patterns (MEP)* verwendet. Im Request-Response-MEP schickt der Service-Nutzer zunächst eine SOAP-Nachricht als Anfrage an den Service-Anbieter und erwartet als Antwort eine SOAP-Nachricht oder – je nach Anwendungsfall – einfach nur eine Empfangsbestätigung zurück. Beim Response-MEP ist die Anfrage keine SOAP-Nachricht, sondern ein GET-HTTP-Request, auf den als Antwort eine SOAP-Nachricht zurückgeliefert wird.

Wie das SOAP-Nachrichtenformat und das HTTP-Transport-Binding letztendlich zur Schnittstellenbeschreibung eines Web Services verwendet werden, wird mithilfe von WSDL angegeben.

### 2.2.2 WSDL

Die *Web Services Description Language (WSDL)* ist eine XML-Sprache zur Beschreibung von Web Services. WSDL 2.0 [181] ist die aktuelle Version, die vom W3C nach längeren Diskussionen im Juni 2007 als Standard veröffentlicht wurde. In Verwendung ist allerdings zur Zeit die ältere Version 1.1 [171], und es ist nicht absehbar, ob sich das in naher Zukunft ändern wird, da die meisten Frameworks nur WSDL 1.1 unterstützen.

Die Änderungen zwischen WSDL 1.1 und 2.0 sind umfangreicher als die zwischen den SOAP Versionen 1.1 und 1.2. Deswegen wurde der W3C-Standard nicht – wie ursprünglich geplant – als Version 1.2 veröffentlicht. Das grundlegende Konzept ist jedoch bei beiden WSDL-Versionen gleich. In Abbildung 2.4 werden WSDL 1.1 und 2.0 miteinander in einer schematischen Darstellung verglichen. Teile, die nicht zu WSDL gehören, aber verwendet werden, sind dabei weiß dargestellt. Als Erstes wird ein Web Service mit einer abstrakten Schnittstelle beschrieben, bevor angegeben wird „wie“ die Schnittstelle konkret umgesetzt wurde und „wo“ der Service zu erreichen ist.

Ein abstrakte Schnittstellenbeschreibung beinhaltet zunächst die Nachrichten-Typen, die ausgetauscht werden, sogenannte *types*. In WSDL 1.1 ist hierfür XML Schema [174] vorgegeben. Zusätzlich ist es in WSDL 2.0 erlaubt, weitere Schemasprachen – wie DTD oder RELAX NG [68] – mithilfe von Erweiterungen zu verwenden. In WSDL 1.1 werden

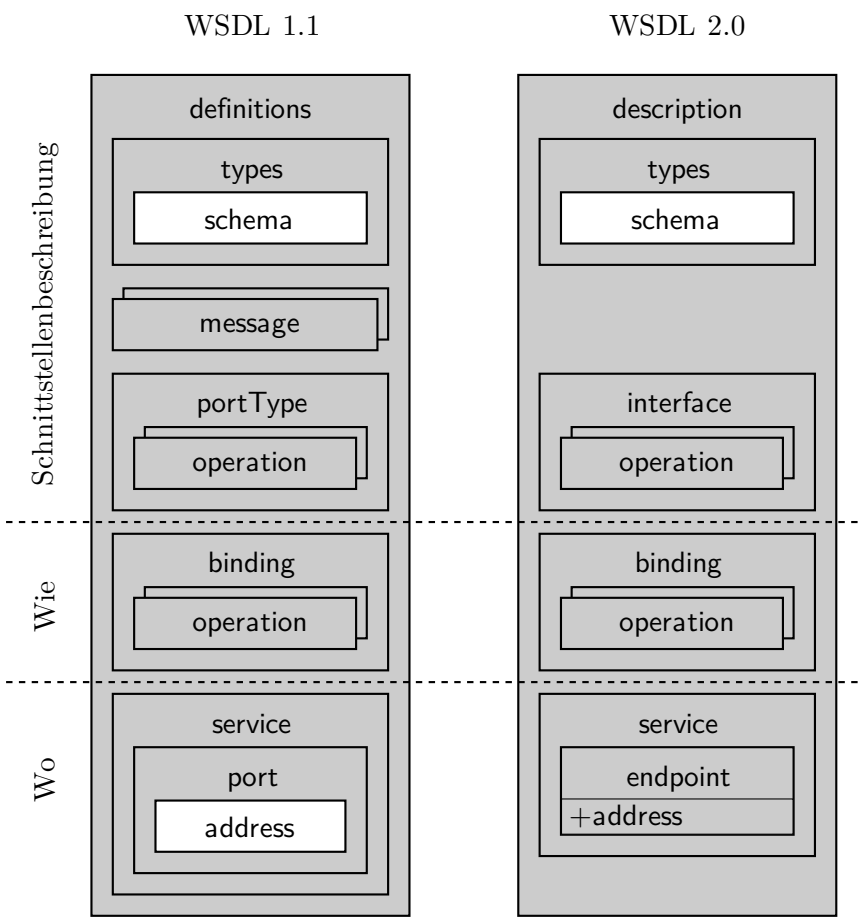



Abbildung 2.4: Vergleich von WSDL 1.1 und 2.0

diese Nachrichten-Typen anschließend einzeln jeweils in einer *message* referenziert, während bei WSDL 2.0 dieser Schritt entfällt.

Der Zusammenhang von Nachrichten, die für eine Operation notwendig sind, lässt sich mit MEPs ausdrücken. In WSDL 1.1 sind vier MEPs definiert, die nicht erweitert werden können: *One-way*, *Request-response*, *Solicit-response* und *Notification*. Im zweiten Teil von WSDL 2.0 werden *In-Only*, *Robust In-Only* und *In-Out* spezifiziert, wobei *In-Only* dem *One-way* und *In-Out* dem *Request-response* aus WSDL 1.1 entspricht. Bei den beiden MEPs *Solicit-response* und *Notification* sind lediglich die Rollen von Service-Nutzer und Service-Anbieter vertauscht. Für WSDL 2.0 werden sie als *Out-In* und *Out-Only* mit den drei zusätzlichen MEPs *In-Optional-Out*, *Robust Out-Only* und *Out-Optional-In* in einer WSDL-2.0-Erweiterung [191] spezifiziert.

Die einzelnen Operationen eines Web Services werden innerhalb einer abstrakten Schnittstelle zusammengefasst. Wie in Abbildung 2.4 zu sehen, heißt diese in WSDL 1.1 *portType*, während sie für WSDL 2.0 in *interface* umbenannt wurde.

Auf der abstrakten Schnittstellenbeschreibung setzt ein Transport-Binding auf, im WSDL als *binding* spezifiziert. In diesem wird angegeben, „wie“ die abstrakten Nachrichten mit einem bestimmten Transport-Protokoll zu übertragen sind. Hier wird meistens SOAP mit einem HTTP-Binding verwendet.

Als Letztes folgt in einer *service*-Definition die Angabe, „wo“ sich der Web Service befindet. In WSDL 1.1 wird hierzu für einen Service seine Adresse in einem *port* angegeben. Für WSDL 2.0 wurde der *port* in *endpoint* umbenannt und die Adresse wird als Attribut angegeben.

### 2.2.3 UDDI

Die Spezifikation für einen Verzeichnisdienst, *Universal Description, Discovery, and Integration (UDDI)*, liegt aktuell in Version 3.0.2 [124] vor. Sie definiert ein Datenmodell sowie eine Schnittstelle zum Veröffentlichen und Auffinden von Service-Beschreibungen. Damit bildet sie eine Kernkomponente bei der Umsetzung einer SOA.

Genau genommen ist ein UDDI-Verzeichnisdienst selbst ein Web Service. Es werden Web-Service-Schnittstellen spezifiziert, mit denen auf das darunterliegende Datenmodell zugegriffen werden kann.

#### Datenmodell

Im UDDI-Datenmodell gibt es vier Haupt-Datentypen: *businessEntity*, *businessService*, *bindingTemplate* und *tModel*. Mit ihnen lassen sich die Fragen „wer?“, „was?“, „wo?“ und „wie?“ beantworten.

Die *businessEntity* ist der übergeordnete Datentyp. Er enthält den Namen einer Firma oder Organisation und kann zusätzliche Beschreibungen dieser enthalten. Außerdem ist es möglich, Kontaktpersonen zu dieser Firma hinzuzufügen. Mit einer *businessEntity* lässt sich also die Frage beantworten, „wer“ einen Service anbietet. Die angebotenen Services dieser Firma werden als Liste von *businessService*s angegeben.

In einem *businessService* wird angeben, „was“ für ein Service angeboten wird. Dazu wird der Service auf semantischer Ebene mit betriebswirtschaftlichen Fachbegriffen in natürlicher Sprache beschrieben. Für die technische Beschreibung des *businessService* wird eine Liste mit *bindingTemplate*s angegeben.

Ein *bindingTemplate* beschreibt eine Web-Service-Instanz. Deswegen wird im sogenannten *accessPoint*-Feld des Datentypen die URL angegeben, unter welcher der Web Service zu erreichen ist. Das *bindingTemplate* gibt damit an, „wo“ ein Web Service zu finden ist. Zusätzlich kann informativer Text hinzugefügt werden.

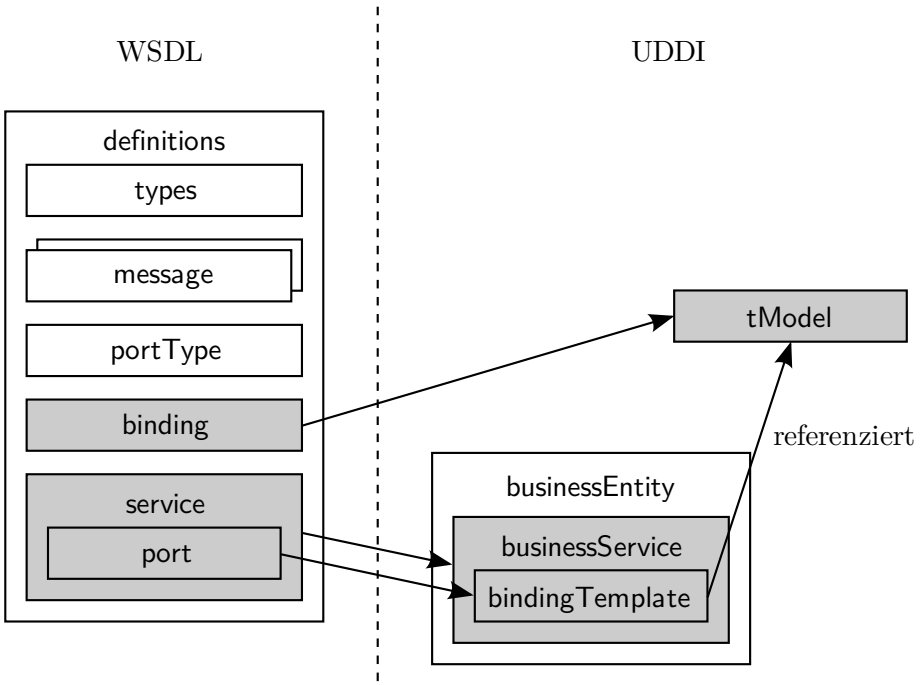



Abbildung 2.5: Web-Service-Beschreibungen im WSDL-Format lassen sich im UDDI-Datenmodell abbilden

Die Liste mit *tModel*-Instanz-Informationen hat beim *bindingTemplate* besondere Bedeutung. In jeder der Instanz-Informationen wird genau ein *tModelKey* angeben, um ein *tModel* zu referenzieren. Alle *tModelKey*s eines *bindingTemplate*, und damit einer Web-Service-Instanz, ergeben zusammen seinen sogenannten „technischen Fingerabdruck“. Web-Service-Instanzen mit dem gleichen Fingerabdruck sind kompatibel, d. h., Web-Service-Nutzer können sie gegeneinander austauschen, ohne weitere Änderungen durchführen zu müssen. Schließlich bieten alle Instanzen mit dem gleichen technischen Fingerabdruck die gleichen Services an. Von einer gleichen Semantik kann in diesem Fall ausgegangen werden.

Das *tModel* beschreibt, „wie“ technisch auf den Web Service zugegriffen werden kann. Es beinhaltet vor allem das Transportprotokoll, das Nachrichtenformat sowie Informationen über Nachrichten-Sequenzen (Operationen). Ein Service-Nutzer benötigt diese Informationen, um zu entscheiden, ob er mit dem Service-Anbieter kommunizieren kann. Leider gibt der UDDI-Standard nicht vor, wie diese Beschreibung genau zu erfolgenden hat. Stattdessen wird auf verschiedene andere Standards verwiesen. Im *tModel* existiert für Web Services die Möglichkeit, das *useType*-Attribut des *overviewURL*-Feldes auf „wsdlInterface“ zu setzen und anschließend eine WSDL-Datei zu referenzieren.

In der Abbildung 2.5 ist zu erkennen, wie eine Web-Service-Beschreibung im WSDL-Format in einem UDDI-Verzeichnis abgebildet werden kann. Das Binding im WSDL entspricht dem *tModel* bei UDDI. Es gibt an, wie mit einem bestimmten Transportprotokoll Nachrichten übertragen werden können. Die verwendeten MEPs entsprechen dabei den Nachrichten-Sequenzen, welche im *tModel* zur technischen Beschreibung gehören. Der Service aus der WSDL-Beschreibung wird in das *businessService*-Feld geschrieben, während die Adresse des Web Services im *bindingTemplate* hinterlegt wird.

### APIs

Auf die Daten in einem UDDI-Verzeichnisdienst kann man über verschiedene APIs zugreifen. Mit ihnen ist es möglich, die Daten auf unterschiedliche Art und Weise abzufragen und zu verändern. Bei den UDDI-APIs handelt es sich um in WSDL beschriebene Web-Service-Schnittstellen. Anbieter und Nutzer von Web Services können dadurch mit der gleichen Technologie auf den Verzeichnisdienst zugreifen, die sie selbst auch für die Kommunikation untereinander verwenden.

Ein Verzeichnisdienst zur Realisierung einer SOA muss das Veröffentlichen und Suchen von Services unterstützen. Bei UDDI sind hierfür die *Publication API* und die *Inquiry API* vorgesehen. Zusätzlich bieten die meisten UDDI-Verzeichnisse eine Web-Oberfläche zur Administration. Über diese können ebenfalls Services gesucht und hinzugefügt werden.

Nach seinem Start hat ein Service-Anbieter mit der *Publication API* die Möglichkeit, sich automatisch beim Verzeichnisdienst zu registrieren. Meistens wird jedoch vom Entwickler der pragmatische anstelle des programmatischen Weges gewählt und der Dienst manuell im Verzeichnis registriert. Viele Dienste im Unternehmensumfeld werden einmalig eingerichtet und bleiben dann für lange Zeit bestehen. Es lohnt sich in diesem Fall nicht, eine automatische Dienstregistrierung zu entwickeln.

Auf der Seite eines Service-Nutzers sieht es anders aus. Er kann nicht beeinflussen, wie lange ein benötigter Service unverändert zur Verfügung steht und muss daher zur Laufzeit nach dem Service-Anbieter suchen. Die *Inquiry API* ist damit die wichtigste API eines UDDI-Verzeichnisdienstes und wird deswegen im Folgenden näher erläutert.

Das Suchen eines Service ist in zwei Schritte unterteilt: Im Ersten geht es darum, eine passende Service-Schnittstellen-Beschreibung zu finden. Die *Inquiry API* bietet hierfür die *find_tModel*-Operation an. In Abbildung 2.6 ist die dazu passende Suchanfrage zu sehen. Es wir nach allen *tModel*s gesucht, welche sich entsprechend der „UDDI Best Practice“ in der Kategorie „wsdlSpec“ einsortiert haben.

```
<find_tModel xmlns="urn:uddi-org:api_v3">
  <categoryBag>
    <keyedReference keyValue="wsdlSpec"
                    tModelKey="uddi:uddi.org:categorization:types" />
  </categoryBag>
</find_tModel>
```

Abbildung 2.6: UDDI-Suchanfrage nach allen WSDL *tModel*s

```
<find_binding xmlns="urn:uddi-org:api_v3">
  <tModelBag>
    <tModelKey>uddi:some.specific.example:tmodelkey</tModelKey>
  </tModelBag>
</find_service>
```

Abbildung 2.7: UDDI-Suchanfrage nach allen Web-Service-Instanzen eines *tModel*s

Die Ergebnisliste beinhaltet neben allen *tModelKey*s den *tModel*-Namen und optionale Beschreibungen. Mit der *get_tModelDetail*-Operation können dann die Service-Schnittstellen-Beschreibungen abgerufen werden. Aus diesen muss sich der Service-Nutzer eine passende Schnittstelle heraussuchen.

Im zweiten Schritt geht es darum, zu einer Service-Schnittstellen-Beschreibung passende Web-Service-Instanzen zu finden. Im einfachsten Fall ist es für den Service-Nutzer unerheblich, wer einen Service bereitstellt, sodass direkt nach *bindingTemplate*s gesucht werden kann, unabhängig von *businessEntity* und *businessService*. In Abbildung 2.7 ist eine solche Suchanfrage exemplarisch aufgeführt. Im *tModelBag* wird einer der im ersten Schritt ermittelten *tModelKey*s angegeben. Sucht ein Service-Nutzer einen Service-Anbieter, der gleich mehrere solcher Schnittstellen in einer Instanz umgesetzt hat, so können mehrere *tModelKey*s im *tModelBag* angeben werden. Hierbei handelt es sich um den sogenannten technischen Fingerabdruck, wie er auch schon im UDDI-Datenmodell vorgestellt wurde.

Neben der *Inquiry API* und der hier nicht näher erläuterten *Publication API* gibt es noch weitere APIs. So wird die Replikation und Rechteübertragung von Services zwischen Verzeichnisdiensten unterstützt. Außerdem gibt es eine *Security API*, um möglicherweise Dienste zu autorisieren, welche Daten im Verzeichnis ändern wollen. Die *Subscription API* erlaubt es Dienstnutzern, sich über Aktualisierungen innerhalb des UDDI-Verzeichnisses zu informieren. Auf diese Weise kann ein Dienstnutzer auf das Erscheinen neuer Services reagieren.

## 2.3 Profile für Web-Service-Interoperabilität

Die Verwendung von Web Services sorgt nicht automatisch für Interoperabilität zwischen den einzelnen Services. Es sind viele verschiedene Web-Service-Spezifikationen notwendig, um den Protokollstapel aus Abschnitt 2.1 abzudecken. Diese sind nicht von vornherein interoperabel, da sie teils von Gruppen unterschiedlicher Unternehmen entworfen und zudem bei verschiedenen Organisationen zur Standardisierung eingereicht wurden. Außerdem liegen viele dieser Spezifikationen in unterschiedlichen Versionen vor, sodass nicht automatisch jeder Web Service technisch in der Lage ist, mit anderen Web Services zu kommunizieren.

Um die dadurch entstandenen Interoperabilitätsprobleme im Web-Service-Umfeld zu lösen, können sogenannte Web-Service-Profile verwendet werden. Services, die das gleiche Web-Service-Profil unterstützen, sind interoperabel. Unter einem Web-Service-Profil ist eine benannte Gruppe von Web-Service-Standards bestimmter Versionen zu verstehen, welche mit zusätzlichen Vereinbarungen ergänzt wurden. Diese Vereinbarungen geben in erster Linie an, wie diese Standards zusammenarbeiten, um Interoperabilität zu gewährleisten.

### 2.3.1 WS-I-Profile

Bei den meisten Web-Service-Profilen handelt es sich um sogenannte WS-I-Profile. Sie wurden von der *Web Services Interoperability (WS-I) Organization* [192] spezifiziert. Die WS-I selbst ist ein offener Industrieverband mit dem Ziel, „Best Practices“ für Web-Service-Interoperabilität zu entwickeln.

Das *Basic Profile 1.0 (BP 1.0)* [194] aus dem Jahre 2004 ist das erste vom WS-I veröffentlichte Profil. Für den Nachrichtentransfer setzt es auf SOAP 1.1 [169] in Verbindung mit dem dort spezifizierten HTTP-Transport-Binding. Die Service-Beschreibung erfolgt in WSDL 1.1 [171] auf Basis von W3C XML Schema [174]. Als Verzeichnisdienst wird UDDI [124] in der älteren Version 2.0 verwendet, sodass das BP 1.0 damit alle Basistechnologien für die Umsetzung einer SOA abdeckt. Zusätzlich wird noch angegeben, wie HTTPS verwendet werden kann, auch wenn es für das Profil nicht verbindlich ist.

Das *Basic Profile 1.1 (BP 1.1)* [197] basiert auf dem BP 1.0 und dessen Korrekturen. Es deckt allerdings nur den Teil des BP 1.0 ab, der sich nicht mit der Serialisierung von SOAP-Nachrichten beschäftigt. Dieser Teil ist in das *Simple SOAP Binding Profile 1.0 (SSBP 1.0)* [195] ausgelagert, sodass BP 1.1 und SSBP 1.0 zusammen in etwa dem BP 1.0 mit Korrekturen entsprechen.

Die Aufteilung ermöglicht es, anstelle des SSBP 1.0 beispielsweise das *Attachments Profile 1.0 (AP 1.0)* [196] zu verwenden. Dann können SOAP-Nachrichten mit Anhängen entsprechend der SwA-Spezifikation [170] verschickt werden.

Das *Basic Profile 1.2 (BP 1.2)* [199] befindet sich noch im Entwurf. Es baut auf dem BP 1.1 auf und übernimmt Teile des SSBP 1.0 für die Nachrichten-Serialisierung. Die entscheidende Neuerung von BP 1.2 ist die Unterstützung für WS-Addressing (vgl. Abschnitt 2.1) und SOAP-MTOM für Anhänge (vgl. Abschnitt 2.2.1).

Ebenfalls im Entwurfsstadium befindet sich das *Basic Profile 2.0 (BP 2.0)* [201]. Es gibt im Gegensatz zum BP 1.2 die Kompatibilität zu BP 1.x auf und verwendet SOAP 1.2 anstelle von 1.1. Bei WSDL wird dagegen Version 1.1 beibehalten, anstatt auf die neue Version 2.0 umzusteigen.

Für den Bereich Sicherheit existieren die Profile *Basic Security Profile 1.0 (BSP 1.0)* [198] und BSP 1.1 [200]. Sie decken die Sicherheit sowohl auf Verbindungsebene mittels HTTPS als auch auf Nachrichtenebene ab. Das BSP 1.0 verwendet für Nachrichten-Sicherheit die OASIS *Web Services Security v1.0 (WS-Security)* [102] Standards, während das BSP 1.1 die WS-Security-v1.1-Reihe [111] verwendet.

Das *Reliable Secure Profile Version 1.0 (RSP 1.0)* [202] liegt ebenfalls als Entwurf vor. Es ermöglicht sichere und zuverlässige Kommunikation. Dazu verwendet es WS-RM [121]. RSP 1.0 lässt sich sowohl mit BP 1.2 und 2.0 als auch mit BSP 1.0 und 1.1 Profilen kombinieren.

### 2.3.2 DPWS

Das *Devices Profile for Web Services (DPWS)* wurde erstmals im Mai 2004 von Microsoft veröffentlicht. Es ist eine Sammlung von Web-Service-Spezifikationen, die sowohl für ressourcenbeschränkte Geräte, wie beispielsweise Netzwerk-Drucker, als auch für umfangreichere Implementierungen geeignet sind. Ziel ist es, PnP in lokalen Netzwerken zu ermöglichen. Seit Juli 2009 ist DPWS in Version 1.1 [117] ein OASIS-Standard.

In Abbildung 2.8 ist die Anordnung von Clients und Geräten entsprechend der DPWS-Spezifikation dargestellt. Auf einem Gerät läuft ein sogenannter *Hosting Service*, der mehrere *Hosted Services* im Netzwerk anbietet. Für den Nachrichtentransport wird SOAP 1.2 verwendet.

Bevor ein Client auf ein Gerät zugreifen kann, muss dieses gefunden werden. DPWS verwendet hierfür *Web Services Dynamic Discovery (WS-Discovery)* [120]. Dabei handelt es sich um ein Protokoll, mit dessen Hilfe sich Web Services in Ad-hoc-Netzwerken auffinden lassen. Hierfür werden Suchanfragen nicht an einen zentralen Verzeichnisdienst gesendet, sondern im lokalen Netzwerk als Rundruf verteilt. Antworten kommen anschließend von allen zur Suchanfrage passenden Geräten bzw. deren Hosting-Services. DPWS sieht außerdem vor, dass Service-Anbieter, sobald sie in

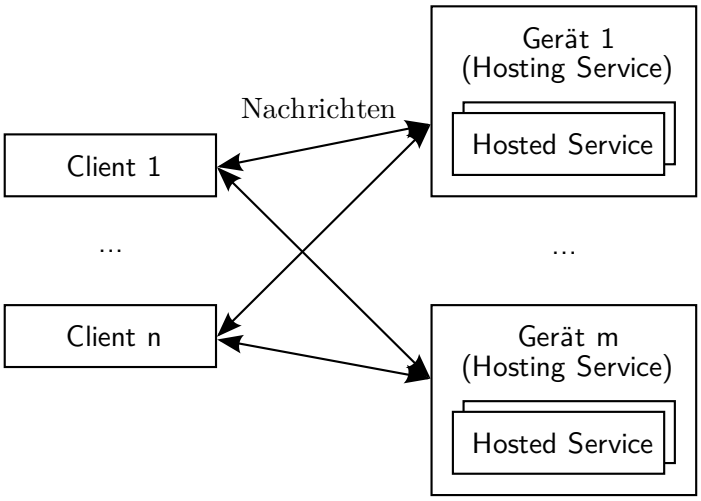



Abbildung 2.8: Anordnung von Clients und Geräten entsprechend der DPWS-Spezifikation, nach [117]

einem Netzwerk verfügbar sind, dies automatisch mit einem Rundruf bekannt geben. Auf diese Weise kann auf ein regelmäßiges Suchen (sogenanntes *Polling*) bei Clients verzichtet werden.

Nachdem ein Client ein Gerät gefunden hat, besteht für ihn die Möglichkeit, Metadaten dieses Gerätes abzufragen. Hierfür verwendet DPWS die WS-MetadataExchange-Spezifikation [187] in Verbindung mit WS-Transfer [189] (vgl. Abschnitt 2.1). Ist die Metadaten-Anfrage an den Hosting-Service adressiert, so werden allgemeine Informationen zu dem Gerät, wie Hersteller oder Modell, zurückgegeben. Zusätzlich ist typischerweise eine Liste mit allen Hosted-Services enthalten, sodass im nächsten Schritt von diesen die Metadaten abgefragt werden können. Die Metadaten eines Hosted-Service enthalten idealerweise eine WSDL-Beschreibung des Service, sodass es möglich ist, ohne Vorwissen alle weiteren notwendigen Metadaten zu ermitteln.

Für ereignisorientierte Kommunikation verwendet DPWS WS-Eventing [186]. Es ermöglicht Clients, Ereignisse von Web Services zu abonnieren. Sobald ein solches Ereignis auftritt, versendet der Web Service eine entsprechende Nachricht an alle Clients, die dieses abonniert haben.

DPWS empfiehlt für sichere Kommunikation die Verwendung von HTTPS, wann immer dies möglich ist. Da WS-Discovery größtenteils jedoch kein HTTP verwendet, muss auf Nachrichtenebene Sicherheit ermöglicht werden. Hierfür ist das in WS-Discovery spezifizierte *Compact Signature Format* vorgesehen [120, Abschnitt 8.2].

### 2.3.3 HL7 Web Services Profile

Im medizinischen Bereich wurde im Rahmen der Spezifizierung von HL7 Version 3 ein Web-Service-Profil für den Nachrichtentransport entworfen. Bei der aktuellen Version [53] handelt es sich um einen Entwurf, der aufgrund von stichhaltigen Problemen zunächst zurückgerufen wurde, jedoch den aktuellen Stand der Technik wiedergibt. Ziel des Profils ist es, Interoperabilität beim Austausch von HL7-Version-3-Nachrichten auf Basis von Web Services zu ermöglichen.

In dem Profil werden Implementierungsentscheidungen in vier Abschnitte eingeteilt: *Basic*, *Addressing*, *Security* und *Reliable Messaging*.

Im Teil *Basic Profile* werden sowohl SOAP 1.1 als auch SOAP 1.2 für den Nachrichtenaustausch zugelassen, wobei eine HL7-Nachricht das einzige Body-Element einer solchen SOAP-Nachricht ist. Für die Service-Beschreibung in WSDL 1.1 muss das WS-Addressing *action*-Attribut verwendet werden. Außerdem wird eine WSDL-Namenskonvention angegeben. Mit dieser lassen sich aus den Namen der HL7-Version-3-Nachrichten die einzelnen Elemente im WSDL benennen, sodass jeder, der sich an die Konvention hält, genau die gleiche WSDL-Datei generiert und für den Nachrichtenaustausch verwendet.

Prinzipiell soll die Kommunikation konform zum WS-I BP 1.0 erfolgen. Der einzige Unterschied besteht in der Signatur von Operationen. Beim BP 1.0 müssen alle Operationen unterschiedliche Root-Elemente im Nachrichten-Body besitzen. Im Gegensatz dazu genügt im HL7-Profil das Root-Element nicht zur eindeutigen Differenzierung. Erst in Verbindung mit dem SOAP-*action*-Attribut lässt sich eine Operation eindeutig ermitteln.

Einzige Ausnahme ist, dass die Signatur einer Operation nicht, wie üblich, nur das Root-Element des Nachrichten-Bodies ist, sondern in Verbindung mit der SOAP-*action* ermittelt wird.

Der *Addressing-Profile*-Teil gibt an, dass WS-Addressing auch innerhalb von SOAP-Nachrichten und nicht nur im WSDL verwendet werden muss. Außerdem wird darauf hingewiesen, dass bei der Angabe eines Absenders (WS-Addressing-From-Element) darauf zu achten ist, dass dieser Angabe nur in geschützten Umgebungen zu vertrauen ist.

Das *Security Profile* basiert auf WS-Security, WS-Trust, WS-SecureConversation, WS-SecurityPolicy sowie WS-Policy (vgl. Abschnitt 2.1). Des Weiteren muss der Web Service zum WS-I BSP konform sein (vgl. Abschnitt 2.3.1). In dem Sicherheitsabschnitt ist aber nicht angegeben, wie Sicherheit praktisch umgesetzt werden soll, sondern nur welche Techniken dafür zur Verfügung stehen. Was für eine Zertifizierungstelle es beispielsweise gibt oder wie „Vertrauen“ zwischen Geräten ermittelt wird, ist offen

gelassen. Es ist einzig und allein vorgeschrieben, dass jeder Client ein eigenes Zertifikat für Identifizierungszwecke haben muss.

Für *Reliable Messaging* wird WS-RM (vgl. Abschnitt 2.1) angegeben und darauf hingewiesen, dass ein Nachrichtentransport sogar mehrere Tage oder Wochen in Anspruch nehmen kann. Schließlich ist es denkbar, dass die Nachrichten per Diskette transportiert werden.

# Kapitel 3

# Entwicklung einer Web-Service-Discovery-Architektur

Bevor Services miteinander kommunizieren können, müssen sie voneinander wissen, was sie anbieten und wo sie zu finden sind. In einer SOA ist für das Veröffentlichen und Finden von Services die Rolle des Verzeichnisdienstes vorgesehen. Ohne die Funktion eines solchen Verzeichnisdienstes können Service-Nutzer und -Anbieter nicht automatisch zueinanderfinden.

Das Codieren von Adressen einzelner Services in den Programm-Code schränkt die Vorteile einer SOA deutlich ein und fördert nicht die gewünschte Interoperabilität zwischen Services. Die Services würden dabei zu eng miteinander gekoppelt. Ein automatisches Suchen und Finden ist somit ein elementarer Bestandteil einer SOA.

In Abschnitt 1.3 über die Motivation dieser Arbeit wurde ein Anwendungsfall skizziert, bei dem ein Röntgenbild aufgenommen wird, während ein Anästhesiegerät den Patienten künstlich beatmet. Üblicherweise wird hierzu die Beatmung durch einen Arzt unterbrochen, dann ein Röntgenbild gemacht und anschließend die Beatmung manuell wieder gestartet. Durch eine automatische Kommunikation zwischen Anästhesie- und Röntgengerät könnte auf die manuelle Unterbrechung und Wiederaufnahme der Beatmung verzichtet werden, sodass der Arbeitsablauf vereinfacht und die Patientensicherheit erhöht würde.

Bei den verwenden Röntgengeräten handelt es sich typischerweise um fahrbare Geräte, sogenannte C-Bögen. Dadurch ist es möglich, die Röntgengeräte nicht nur in einem einzigen Operationssaal zu benutzen, sondern ein Gerät bei Bedarf zu holen. Das hat zur Folge, dass Anästhesiegerät und Röntgengerät sich selbstständig während einer Operation suchen und finden müssen. Eine Discovery-Architektur muss es Geräten ermöglichen, andere Geräte in einem dynamischen Umfeld zur Laufzeit zuverlässig zu finden, um so eine Art PnP von medizinischen Geräten zu ermöglichen.

Unabhängig von dieser PnP-Funktionalität sollte es auch möglich sein, Web Services innerhalb des gesamten Klinikums und nicht nur innerhalb eines Operationssaals zu suchen. Dies ist etwa nötig, wenn einzelne medizinische Geräte auf das KIS zugreifen wollen, um benötigte Patientendaten abzufragen, oder wenn zur Wartung

beispielsweise Statusinformationen von Geräten abgefragt werden sollen. Die Discovery-Architektur sollte folglich vom lokalen PnP bis hin zu einer organisationsweiten Discovery-Architektur skalieren.

In diesem Kapitel werden zunächst verwandte Technologien und Arbeiten vorgestellt, in denen es um das Auffinden von Web Services geht. Die Architekturen haben unterschiedliche Ansätze, wobei WS-Discovery für die Vernetzung von medizinischen Geräten am vielversprechendsten ist.

Anschließend wird die vom Autor entworfene Web-Service-Discovery-Architektur auf Basis von WS-Discovery beschrieben. Zwei mögliche Realisierungen werden vorgestellt und bewertet. Zum Abschluss werden die Ergebnisse zusammengefasst.

Die Ideen und Konzepte aus diesem Kapitel wurden im Wesentlichen vorab in [140] und [133] veröffentlicht.

## 3.1 Verwandte Technologien

Die beiden wichtigsten verwandten Technologien zum Thema Suchen von Web Services sind zweifellos UDDI und WS-Discovery, die beide recht gegensätzliche Ziele verfolgen. Es existieren noch weitere Technologien und Arbeiten zu diesem Thema, die allerdings aufgrund der Tatsache, dass sie sich kaum durchgesetzt haben, nur kurz in einem gemeinsamen Abschnitt aufgeführt werden.

### 3.1.1 UDDI

Eine technische Einführung zu UDDI erfolgte bereits in Abschnitt 2.2.3. An dieser Stelle wird die Verwendung von UDDI zum Auffinden von Services bewertet.

Als UDDI im Jahre 2000 entwickelt wurde, lag der Fokus auf der Schaffung eines weltweiten Verzeichnisdienstes – der *UDDI Business Registry (UBR)* [123]. Mit dem Aufbau wurde von IBM, Microsoft und SAP noch im selben Jahr begonnen. Damals hatte man die Vision, auf diese Weise Geschäftspartner für E-Commerce zu erreichen. Deswegen beinhaltet ein UDDI-Verzeichnis auch allgemeine Geschäftsdaten wie Firmenname, Postanschrift, Telefon- und Faxnummern sowie eine Branchenangabe.

Bei einem UDDI-Verzeichnis handelt es sich um einen zentralen Dienst, der sich zwar mittels Replikation auf weitere Knoten verteilen lässt, aber zentral verwaltet wird. Werden einzelne replizierte Knoten nicht zentral, sondern jeweils von einem anderen Betreiber verwaltet, so ist nicht sichergestellt, dass alle Knoten die gleichen Informationen zur Verfügung stellen. In diesem Falle entstünde das Problem, dass zum Auffinden eines Service zunächst der „richtige“ Knoten des replizierten Verzeichnisdienstes gefunden werden müsste.

Insgesamt hat der zentrale Ansatz von UDDI dafür gesorgt, dass Unternehmen – insbesondere für kritische Anwendungen – einen eigenen Verzeichnisdienst im Haus benötigen. Diesen können sie selbst verwalten und haben daher volle Kontrolle über alle angebotenen Services. Damit ist die Grundidee, die hinter UDDI stand, fehlgeschlagen. Auch wenn dies nicht von den Betreibern der UBR direkt gesagt wird, so ist doch die Abschaltung der UBR im Januar 2006 ein eindeutiges Zeichen dafür [91].

Erschwerend kommt hinzu, dass UDDI ausschließlich von der OASIS und nicht vom W3C standardisiert wurde, obwohl dieses Spezifikationen der Web-Service-Kernkomponenten abdeckt. In der Web-Service-Architektur des W3C wird UDDI nur indirekt erwähnt: Es wird in der Prozessebene des Protokollstapels unter dem Begriff *Discovery* eingeordnet.

### 3.1.2 WS-Discovery

*Web Services Dynamic Discovery (WS-Discovery)* [120] definiert ein Protokoll zum Auffinden von Web Services. Es ist zusammen mit DPWS Mitte 2009 durch die OASIS standardisiert worden und bildet die Basis für Web-Service-PnP in lokalen Netzwerken. Damit verfolgt WS-Discovery einen ganz anderen Ansatz als UDDI.

Das Protokoll unterstützt zwei Betriebsmodi: den Ad-hoc-Modus und den Managed-Modus. Diese haben nichts mit der IP-Adressenvergabe zu tun, da bei WS-Discovery davon ausgegangen wird, dass gültige IP-Adressen vorhanden sind. Wie üblich kann eine IP-Adressenvergabe entweder durch statische Konfiguration, dynamische Zuweisung mittels des *Dynamic Host Configuration Protocol (DHCP)* [35] oder durch eine automatische Konfiguration [16, 160] stattfinden.

Der Ad-hoc-Modus von WS-Discovery kommt ohne zentrale Komponenten aus und nutzt die Multicast-Funktionalität moderner Netzwerke zum Verteilen von Nachrichten. Alle Web Services treten entsprechend WS-Discovery einer lokalen Multicast-Gruppe bei und haben so die Möglichkeit, mit minimalem Aufwand untereinander Nachrichten auszutauschen. Für IPv4 ist die Gruppe 239.255.255.250 spezifiziert, während dies für IPv6 FF02::C ist. Um SOAP-Nachrichten per Multicast zu verschicken, wird das *User Datagram Protocol (UDP)* benötigt. WS-Discovery verwendet deswegen für die Kommunikation SOAP-over-UDP [118]. Dabei handelt es sich um ein UDP-Transport-Binding, das eine SOAP-Nachricht als Nutzlast eines UDP-Datagramms verschickt.

Das PnP-Konzept von WS-Discovery ist nicht neu. Vieles wurde vom *Service Location Protocol (SLP)* [48] und vom *Simple Service Discovery Protocol (SSDP)* [46] übernommen. Bei SLP handelt es sich um eine Spezifikation (RFC), die in lokalen Netzwerken dazu verwendet werden kann, etwa Netzwerkdrucker zu finden. SSDP dagegen bildet die Discovery-Basis von *Universal Plug and Play (UPnP)* [161] und

wurde von Microsoft erstmals mit Windows ME ausgeliefert. UPnP wird beispielsweise für die Ansteuerung von Routern und Media-Centern verwendet.

Die wesentliche Aufgabe von WS-Discovery ist, das Auffinden (*Probe*) von Web Services, die in der Spezifikation *Target Services* genannt werden, zu ermöglichen. Die Suche nach Target Services kann durch Angabe eines oder mehrerer *Types* sowie *Scopes* eingeschränkt werden. Unter einem *Type* ist ein Bezeichner zu verstehen, der für eine bestimmte Gruppe von Nachrichten steht, die ein Target Service empfängt oder sendet. Als Beispiel lassen sich hier WSDL-portTypes nennen; sie geben an, welche abstrakten Methoden ein Service anbietet.

Mit der Angabe eines *Scope* kann der Bereich eingeschränkt werden, in dem gesucht werden soll. Die Einschränkung basiert nicht auf der Netzwerkstruktur, sondern auf einer willkürlichen Einteilung in logische Gruppen. Ein Scope ist ein beliebiger *Uniform Resource Identifier (URI)* und kann beispielsweise als einfacher String zeichenweise verglichen werden (*case-sensitive*). Es ist aber auch möglich, den Scope als LDAP-String vergleichen zu lassen, sodass sich beliebige Untergruppen von Target Services bilden lassen.

Als Antwort erhalten Clients auf eine Probe-Anfrage ProbeMatches-Nachrichten. Jeder Target Service, der allen Types und Scopes der Probe-Nachricht entspricht, antwortet mit einer solchen ProbeMatches-Nachricht. Folglich kann ein Client keine, eine oder mehrere Antworten auf eine Suchanfrage bekommen. In der ProbeMatches-Nachricht sind für einen Target Service neben seinen Types und Scopes ein Bezeichner und mögliche Transportadressen angegeben.

Der Discovery-Prozess sieht vor, dass Target Services einen gleichbleibenden Bezeichner haben, der für jede seiner Netzwerkschnittstellen gültig ist. Daher werden üblicherweise *Universally Unique Identifier (UUID)* [81] anstelle einer Transportadresse verwendet. Diese ermöglichen es, dass ein Target Service auch nach einem Neustart den selben Bezeichner verwendet, selbst wenn er eine andere IP-Adresse zugewiesen bekommen hat.

Ist der gesuchte Target Service mit seinen Types und Scopes bereits bekannt, so kann statt einer Suche (Probe) eine Namensauflösung (*Resolve*) durchgeführt werden, d. h., nach seinem Bezeichner gesucht werden. Der Target Service mit dem entsprechenden Bezeichner antwortet auf die Resolve-Anfrage mit einer ResolveMatches-Nachricht. Diese enthält – wie die ProbeMatches-Nachricht – neben Types, Scopes und Bezeichner eine oder mehrere Transportadressen.

Abgesehen von Probe- und Resolve-Anfragen von Clients zum aktiven Suchen gibt es noch Bekanntmachungen (*Announcements*) von den Target Services. Ein Target Service verschickt an die Multicast-Adresse eine sogenannte *Hello*-Nachricht beim Betreten des Netzwerks, welche die gleichen Informationen wie eine ProbeMatches-Nachricht enthält. Auf diese Weise wird es Clients ermöglicht, anstelle eines regelmäßigen Polling nach neuen Services die Hello-Nachrichten auszuwerten.

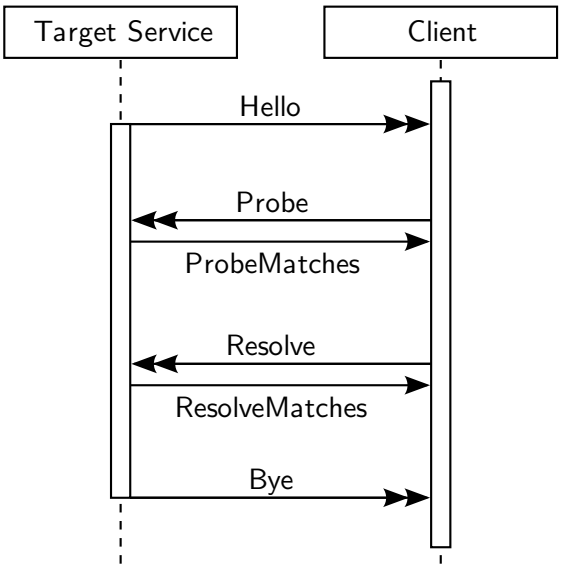



Abbildung 3.1: Nachrichtenaustausch im Ad-hoc-Modus nach [120]. Bekanntmachungen und Anfragen werden per Multicast verschickt, während Antworten direkt (unicast) zugestellt werden.

Bevor ein Target Service das Netzwerk verlässt, verschickt er eine *Bye*-Nachricht, sodass auch dies den Clients mitgeteilt wird. Diese haben so die Möglichkeit, immer auf dem aktuellen Stand über die vorhandenen Services zu bleiben, ohne regelmäßig nach neuen Target Services zu suchen.

Der Nachrichtenaustausch von WS-Discovery im Ad-hoc-Modus zwischen einem Target Service und einem Client ist in Abbildung 3.1 zu sehen. UDP-Datagramme, die per Multicast verschickt werden, sind mit einem doppelten Pfeil abbildet, während die per Unicast verschickten einen einfachen Pfeil haben. Target Services senden Hello- und Bye-Nachrichten beim Betreten und Verlassen des Netzwerks. Ansonsten antworten sie nur dann auf Probe- und Resolve-Nachrichten eines Clients, wenn sie aufgrund ihrer Types und Scopes bzw. ihres Bezeichners der Anfrage entsprechen. Dazu verschicken sie entsprechend der Anfrage eine *ProbeMatches* bzw. *ResolveMatches*-Antwort.

Unter Umständen unterstützt ein Target Service so viele Types oder befindet sich in so vielen Scopes, dass er nicht alle Daten in einer ProbeMatches-Antwort über UDP verschicken kann. In diesem Fall bindet der Target Service nur die relevanten Types und Scopes ein, nach denen in der Multicast-Probe-Nachricht gefragt wurde. Um alle Types und Scopes eines Target Services zu ermitteln, unterstützt WS-Discovery zusätzlich HTTP-Probe-Anfragen (*directed Probe*). Ist ein Target Service einem Client bereits bekannt, so kann dieser per HTTP eine Probe-Anfrage an den Target Service stellen, welcher nun die Möglichkeit hat, mit allen seinen Types und Scopes zu antworten, da keine Nachrichtenbeschränkung vorhanden ist.

Der Managed-Modus von WS-Discovery ist für größere Netzwerke gedacht. Abgesehen von den beiden Rollen Client und Target Service gibt es hierbei eine dritte Rolle, den *Discovery Proxy*. Die Aufgabe des Discovery Proxy ist es, Informationen über Target Services zwischenzuspeichern, um diese den Clients zur Verfügung zu stellen. Clients haben nun die Möglichkeit, gezielt den Discovery Proxy zu kontaktieren, anstatt mit einer Multicast-Anfrage alle Target Services zu befragen. Clients verwenden hierfür einen *directed Probe*, wie auch beim direkten Abfragen eines Target Service. Auf diese Weise tragen Discovery Proxies zu einer Reduzierung des Paket-Aufkommens bei.

Der Wechsel zwischen Ad-hoc-Modus und Managed-Modus kann für einen Client dynamisch erfolgen. Zunächst hat ein Client keine Kenntnis über einen vorhandenen Discovery Proxy. Sobald er seine erste Anfrage per Multicast verschickt, bekommt er nicht nur die Antworten von Target Services, sondern zusätzlich eine Hello-Nachricht von einem vorhandenen Discovery Proxy. Anschließend kann der Client seine Anfragen direkt an den Discovery Proxy schicken und produziert keinen weiteren Multicast-Verkehr. Sollte der Discovery Proxy auf Client-Anfragen nicht mehr reagieren, so wechselt der Client automatisch wieder in den Ad-hoc-Modus zurück.

Für Target Services ist in der Spezifikation kein automatischer Wechsel zwischen Ad-hoc- und Managed-Modus vorhanden. Schließlich ist für einen Target Service nach dem Bekanntmachen mit einer Hello-Nachricht im Managed-Modus bis auf die Bye-Nachricht kein weiterer Multicast-Nachrichtenaustausch vorgesehen.

Im Gegensatz zur ursprünglichen WS-Discovery-Spezifikation [9] aus dem Jahre 2005 ist in der von OASIS standardisierten WS-Discovery Version 1.1 von 2009 ein expliziter Managed-Modus aufgeführt. In diesem werden alle Nachrichten per HTTP direkt an den Discovery Proxy geschickt, auch von Target Services. Wie dieser Discovery Proxy allerdings ermittelt wird, ist nicht spezifiziert. Es wird lediglich angegeben, dass diese Information beispielsweise über explizite Konfiguration, Probe-Anfragen nach Discovery Proxies, *Domain Name System (DNS)* [93, 94] oder DHCP ermittelt werden kann [120, Abschnitt 2.2.2].

Der Fokus von WS-Discovery liegt eindeutig auf dem Ad-hoc-Modus, der für PnP eine unentbehrliche Basis bildet.

### 3.1.3 Weitere Technologien und verwandte Arbeiten

Im Juli 2000 hat Microsoft einen Entwurf von *Discovery of Web Services (DISCO)* veröffentlicht [90]. Dabei handelt es sich um einen Discovery-Algorithmus, der eine URL als Eingabe benötigt und ein Discovery-Dokument ausgibt. Auf der URL wird ein HTTP-GET-Request ausgeführt und in Abhängigkeit des empfangenen Dokumentes weiterverfahren. Je nachdem, was zurückgegeben wird, muss ein XML-Stylesheet angewendet oder aber eine HTML-Seite ausgewertet werden.

Ein ähnliches Protokoll für Web Services, *The Advertisement and Discovery of Services (ADS) protocol for Web services* [96], wurde 2000 von IBM veröffentlicht. Es verwendet zusätzlich die *Well Defined Services (WDS) Specification*, die Teil des IBM *Web Services Toolkit (WSTK)* [57] ist. Diese ermöglicht es, die nicht-funktionalen Teile eines Service zu beschreiben, und dient dazu, mit dem Betreiber des Service in Kontakt zu treten, anstatt direkt mit dessen Services zu kommunizieren. Nach ADS werden Web Services wie bei DISCO über einen Webserver bekannt gegeben. Dazu wird entweder im Root-Verzeichnis des Webservers eine Datei mit dem Namen `svcsadvt.xml` abgelegt oder aber mit einem HTML-Meta-Element auf eine solche Datei verwiesen. Bei der Datei handelt es sich entweder um eine WSDL-Datei, eine WDS-Datei oder eine XML-Datei, in der mehrere WDS-Dateien referenziert sind.

Im darauffolgenden Jahr haben Microsoft und IBM die *Web Services Inspection Language (WS-Inspection) 1.0* veröffentlicht [8]. Dabei handelt es sich um die Zusammenführung der Konzepte von DISCO und ADS. Kern der Spezifikation ist ein XML-Schema, das die Struktur einer Inspection-XML-Datei (WSIL-Datei) beschreibt. In einer solchen Inspection-Datei werden Zeiger auf weitere Web-Service-Beschreibungen aufgelistet. Ein solcher Zeiger kann eine URL für eine WSDL- oder WSIL-Datei sein, sowie eine Referenz auf ein UDDI-Verzeichnis und den dazugehörigen *serviceKey* eines einzelnen Service.

Durch die Verwendung von WS-Inspection und UDDI liegen die Informationen verteilt und sind nicht gut auffindbar. Daher haben Zhang et al. [205] ein Framework zur Aggregation entwickelt, welches dynamisch diese Dokumente durchsucht, um aktuelle Services zu finden. Zu Demonstrationszwecken ist gleichzeitig ein Portal entstanden, auf dem die Daten zur Verfügung gestellt wurden.

Einen anderen Ansatz verfolgen Snell und Donoho [156] mit *DNS Endpoint Discovery (DNS-EPD)*. Sie führen zwei neue *DNS Resource Records (RR)* ein, um Web Services im Internet auffindbar zu machen. Bei dem einen handelt es sich um einen *Endpoint Reference (EPR) RR*, der die Adresse und den portType eines Web Service beinhaltet. Zum anderen ist *Endpoint Extension (EPX) RR* als zusätzlicher RR gedacht, um dem ersten RR weitere Service-beschreibende Daten hinzuzufügen. Innerhalb eines EPX RR kann entweder mittels einer URL auf eine WSDL-Datei verwiesen werden oder direkt XML eingebettet werden. Das Hauptproblem dieses Ansatzes ist, dass gängige DNS-Server diese RRs nicht unterstützen, auch wenn Alor-Hernandez et al. [4] eine prototypische Implementierung vorgelegt haben.

## 3.2 Architektur

Bei der für diese Arbeit entworfenen Architektur handelt es sich um eine hybride Discovery-Architektur. Abbildung 3.2 gibt einen Überblick.

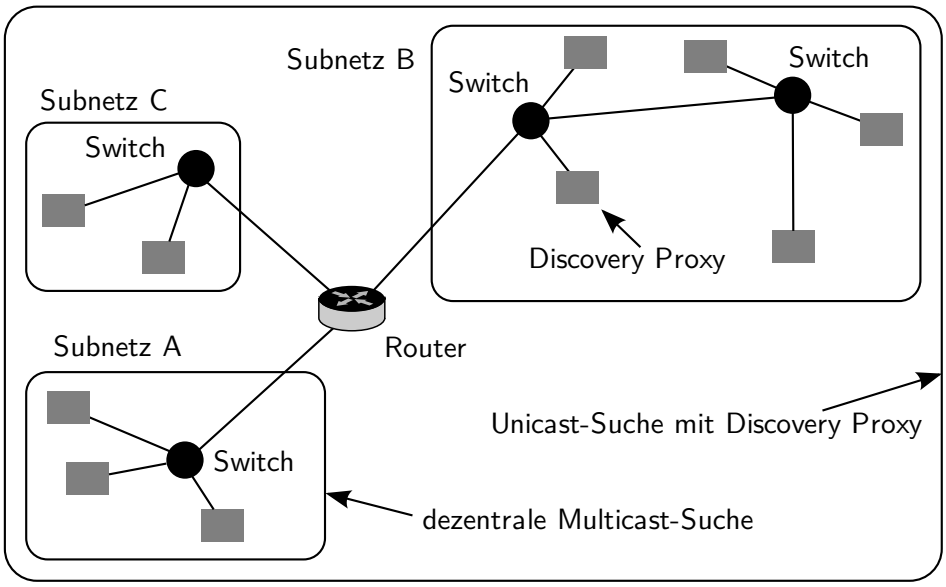



Abbildung 3.2: Die zwei Suchebenen der hybriden Discovery-Architektur: lokale Suche dezentral, globale Suche zentral.

Für lokale Suchen innerhalb eines Subnetzes wird der Ad-hoc-Modus von WS-Discovery verwendet. Im Ad-hoc-Modus werden Hello- und Bye-Nachrichten verschickt, die ein regelmäßiges Suchen nach neuen Services vermeidbar machen. Dies ist eine elementare Eigenschaft, um die Reaktionsgeschwindigkeit der Services untereinander zu erhöhen.

Für die lokale Suche wird bewusst auf einen Discovery Proxy zum Zwischenspeichern von Informationen über vorhandene Target Services verzichtet. Zwar kann eine Unicast-Verbindung zu einem zentralen Discovery Proxy zuverlässiger sein; die dezentrale Lösung mit Multicast ist jedoch robuster, da im Falle eines partiellen Netzwerkausfalls der unbeschädigte Teil des Netzwerks weiterlaufen kann. Auf diese Weise wird ein sogenannter *Single Point of Failure (SPOF)* vermieden. Bei einem zentralen Discovery Proxy besteht zudem die Gefahr, dass veraltete Informationen gespeichert sind. Wird ein Gerät beispielsweise einfach ausgeschaltet, so kann es sich nicht mehr beim Discovery Proxy abmelden, sodass dieser zukünftig falsche Suchergebnisse liefert.

Zusammenfassend lässt sich über die lokale Suche in dieser Architektur sagen, dass sie zuverlässig funktioniert, solange Client und Target Service miteinander kommunizieren können. Dabei kann es sich auch um ein Subnetz mit nur zwei Geräten handeln, die mit einem Crossover-Ethernet-Kabel verbunden sind.

Für die Suche von Target Services außerhalb eines Subnetzes wird ein zentraler Discovery Proxy verwendet (vgl. Abbildung 3.2). Eine Multicast-Kommunikation, wie sie innerhalb eines Subnetzes stattfindet, ist hier nicht sinnvoll. Deswegen ist die IPv6-Multicast-Adresse bei WS-Discovery auch eine sogenannte *Link-Local*-Adresse

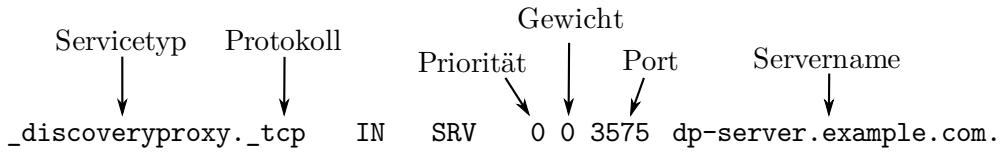



Abbildung 3.3: Beispiel eines Service (SRV) RR

(FF02::C). Die IPv4-Multicast-Adresse (239.255.255.250) ist aus dem administrativen Bereich und sollte nur für ein Subnetz verwendet werden [150]. Die Adresse kann aber durchaus auch über Subnetzgrenzen hinweg im ganzen Unternehmen bzw. Krankenhaus geroutet werden. In diesem Fall können die Hello- und Bye-Nachrichten zu datenschutzrechtlichen Problemen führen, da eine unternehmensweite Kontrolle über das Ein- und Ausschalten von Geräten möglich wird. Ein weiterer Aspekt wäre die steigende Netzwerklast. Alle Hello- und Bye-Nachrichten müssten im gesamten Netzwerk verteilt werden, da jeder Target Service der Multicast-Gruppe beitritt.

Damit der zentrale Discovery Proxy von Target Services sowie Clients erreicht werden kann, muss seine Adresse bekannt gemacht werden. Hierfür wird in dieser Architektur auf DNS [93, 94] zurückgegriffen. In einer ersten Realisierung wurde stattdessen DHCP [35] verwendet, jedoch hat sich dabei gezeigt, dass dieser Ansatz weniger geeignet ist. Details sind in der späteren Bewertung zu finden.

Im DNS lassen sich zu einer Domain verschiedene Services definieren. Für ein Unternehmen mit der Domain `example.com` ist etwa der Webserver unter `www.example.com` zu finden. Hierbei handelt es sich allerdings nur um eine allgemein akzeptierte Konvention.

Ein Mail-Server zum Empfangen von E-Mails wird mit einem eigenen Eintrag, dem *Mail Exchange (MX) RR*, im DNS eingetragen. Gegenüber oben beschriebener Konvention hat der MX RR den Vorteil, dass dieser standardisiert ist und zusätzlich noch die Angabe einer Priorität für Backup- bzw. Lastverteilungsszenarien ermöglicht.

Um für neue Service-Typen nicht jedes Mal einen neuen RR-Typen spezifizieren und implementieren zu müssen, gibt es einen generischen *Service (SRV) RR*. Ein solcher wird beispielsweise für Internettelefonie über das *Session Initiation Protocol (SIP)* verwendet. In Abbildung 3.3 ist ein SRV RR mit Beispielwerten für diese Architektur dargestellt. Auf der linken Seite sind der Service-Typ sowie das verwendete Protokoll angegeben. Für den Service-Typ muss ein Name spezifiziert werden, im Falle dieser Architektur ist es `discoveryproxy`, während beispielsweise das SIP-Protokoll `sip` verwendet. Das Protokoll ist entweder `tcp` oder `udp`. Anschließend wird angegeben, dass es sich um einen Internet (IN) Service (SRV) RR handelt.

Wo ein Service zu erreichen ist, wird auf der rechten Seite mit Port und Server-Namen angegeben. Die Werte für die Priorität und Gewichtung sind für Backup und Lastverteilungszwecke nützlich. Möchte man die Lastverteilungsmechanismen verwenden,

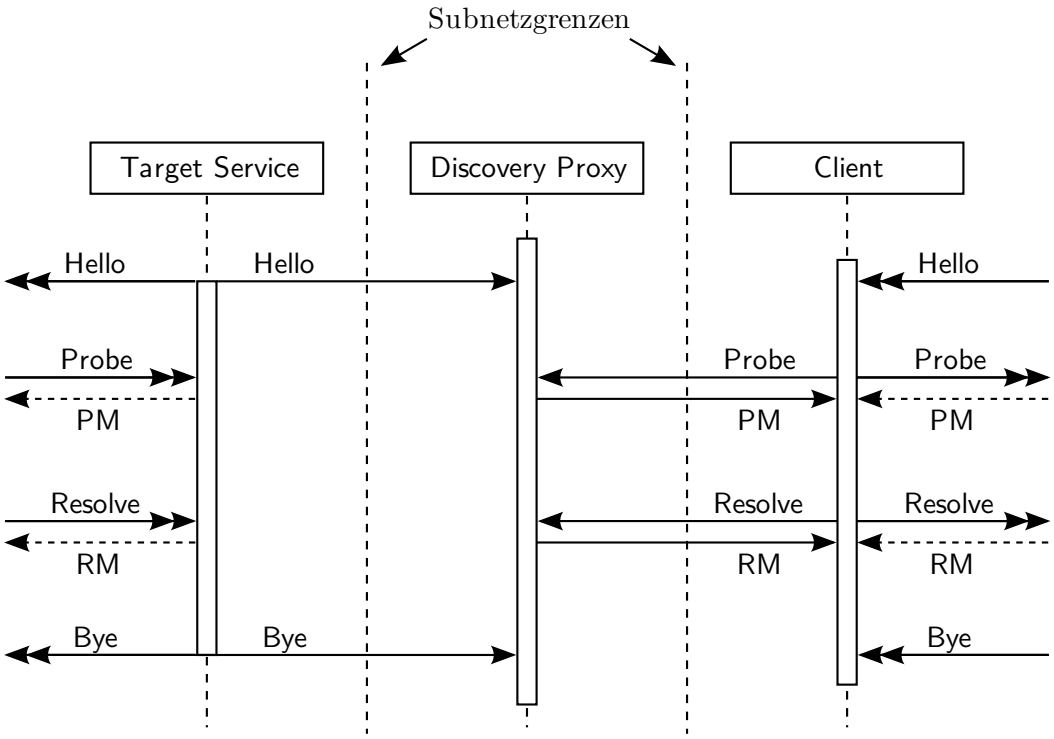



Abbildung 3.4: Nachrichtenaustausch für globales Suchen erfolgt mit Unicast über Subnetzgrenzen, während lokales Suchen per Multicast erfolgt.

werden für ein und denselben Service mehrere SRV RRs angegeben. Einträge mit einem kleineren Prioritätswert sollen dabei bevorzugt verwendet werden. Bei gleicher Priorität soll zufällig zwischen den Einträgen entsprechend ihrer Gewichtung eine Auswahl getroffen werden.

In der hier vorgestellten Discovery-Architektur ist vorgesehen, den zentralen Discovery Proxy mit einem SRV RR im DNS zu hinterlegen. Clients und Target Services haben nun die Möglichkeit, den zentralen Discovery Proxy über DNS zu ermitteln und mit ihm im expliziten Managed-Modus zu kommunizieren.

Der Nachrichtenaustausch der hybriden Discovery-Architektur ist in Abbildung 3.4 dargestellt. Target Services senden von sich aus beim Betreten des Netzwerks eine Hello-Multicast-Nachricht im Subnetz und registrieren sich mit einer weiteren Hello-Nachricht bei dem zentralen Discovery Proxy. Entsprechend wird die Bye-Nachricht beim Verlassen des Netzwerks sowohl Multicast als auch Unicast (per HTTP) versendet. Ansonsten antwortet der Target Service auf *directed-Probe*-Anfragen und lokale Multicast-Anfragen mit ProbeMatches-(PM)- und ResolveMatches-(RM)-Nachrichten.

Clients haben zwei Möglichkeiten, aktiv nach Target Services zu suchen. Die eine ist die lokale Suche mithilfe von Multicast-Anfragen. Die andere Variante nutzt den Discovery Proxy, um Target Services zu finden, die sich dort registriert haben. Unabhängig davon werden Clients über neu auftauchende Target Services in ihrem Subnetz informiert. Dies ist besonders für PnP-Szenarien interessant und ein elementarer Bestandteil der Architektur.

Um den im DNS hinterlegten SRV RR abzufragen, wird dessen *Fully Qualified Domain Name (FQDN)* benötigt. Er beginnt mit der Angabe des Service-Typs und des Protokolls (`_discoveryproxy._tcp`) und endet mit der lokalen Domain. Die lokale Domain eines Netzwerkgerätes lässt sich über seinen eigenen FQDN ermitteln. Dieser setzt sich aus dem Hostnamen und der lokalen Domain zusammen. Bei einer statischen IP-Konfiguration muss diese mit angegeben werden.

In den meisten Netzwerken erfolgt eine dynamische IP-Adressen-Zuweisung auf Basis von DHCP. Einem Gerät wird dabei nicht nur die IP-Adresse mit der dazugehörigen Netzmaske übermittelt, sondern meist auch ein Hostname, die dazugehörige lokale Domain, das Gateway und Nameserver-Adressen. Bei einer dynamischen IP-Adressen-Zuweisung werden folglich alle nötigen Daten geliefert, um mit der hier vorgestellten Architektur über Subnetzgrenzen hinweg nach Geräten zu suchen. Eine explizite Konfiguration der Geräte entfällt.

## 3.3 Realisierung

Im Rahmen dieser Arbeit ist nicht nur die beschriebene Architektur auf DNS-Basis implementiert worden, sondern zunächst eine, die ausschließlich DHCP verwendet. Üblicherweise werden lokale Dienste in speziellen Feldern bei einer DHCP-Konfiguration zusätzlich angegeben. Abgesehen von den üblichen Angaben für Gateway und DNS-Server können hier beispielsweise Adressen von NetBIOS-Nameservern, Zeitservern sowie Server für Sun's *Network Information Services (NIS)* angegeben werden.

### 3.3.1 DHCP-basierte Realisierung

Für die Realisierung wird der DHCP-Server des *Internet Systems Consortium (ISC)* verwendet [64]. Er erlaubt es, eigene Optionen für die Konfiguration zu verwenden. Die unterschiedlichen Optionen werden anhand ihres *Option Code*, einem 8-Bit-Integer, unterschieden. Die Option für die Subnetzmaske hat beispielsweise den Code 1, während die Nameserver-Option den Code 5 hat. Für die eigene Verwendung stehen *site-specific* Codes von 128 bis 254 zur freien Verfügung [2]. Dementsprechend ist der DHCP-Server so konfiguriert worden, dass er unter einem Options-Code aus diesem Bereich die IP-Adresse des zentralen Discovery Proxy zurück gibt.

Für die DHCP-Anfragen soll zunächst ein DHCP-Client in den Web Service integriert werden. Dieser soll DHCP-Anfragen an den DHCP-Server schicken, um so an die IP-Adresse des zentralen Discovery Proxy zu gelangen. In der DHCP-Spezifikation ist festgelegt, dass Nachrichten vom DHCP-Server an den Client immer an Port 68 geschickt werden sollen. Da dieser Port aber bereits vom „richtigen“ DHCP-Client des Systems belegt ist, hat sich diese Möglichkeit als unbrauchbar erwiesen.

Um an DHCP-Konfigurationsdaten zu gelangen, bleibt folglich nur der Umweg über den bereits vorhandenen System-DHCP-Client. Der unter Linux verwendete DHCP-Client des ISC protokolliert die empfangenen Daten in `/var/lib/dhcp3/dhclient-eth0.lease`. Target Services und Clients können folglich diese Datei parsen, um an die Adresse des zentralen Discovery Proxy zu gelangen.

Der verwendete Discovery Proxy ist nicht selbst entwickelt, sondern vom WS4D-JavaME-Stack [204] übernommen worden. Dabei handelt es sich um eine Discovery-Proxy-Implementierung in Java, die bei der hier vorgestellten Lösung keinerlei Konfiguration bedarf.

### 3.3.2 DNS-basierte Realisierung

Die DNS-basierte Realisierung entspricht der obigen Architekturbeschreibung. Im Bind9-DNS-Server des Instituts für Telematik der Universität zu Lübeck ist für die lokale Domain `itm.uni-luebeck.de` der folgende SRV RR hinterlegt worden: `_discoveryproxy._tcp.itm.uni-luebeck.de. IN SRV 0 0 8888 itm01.itm.uni-luebeck.de.` Wie bei der DHCP-basierten Realisierung wird der Discovery Proxy des WS4D-JavaME-Stacks verwendet. In diesem Fall ist der Port auf den im SRV RR angegebenen Port 8888 gesetzt.

Für Web Services und Clients in Java wird auf die DNS-Implementierung dnsjava [33] in Version 2.0.6 zurückgegriffen. Dabei handelt es sich um eine Java-Bibliothek, die DNS-Abfragen auf verschiedenen Abstraktionsebenen ermöglicht. Auf diese Weise ist es möglich, den verwendeten WS4D-JavaMe-Stack so anzupassen, dass er auch mit einem zentralen Discovery Proxy kommunizieren kann.

Die DNS-basierte Realisierung funktioniert sogar weltweit, wenn sowohl der DNS-Server als auch der zentrale Discovery Proxy aus dem Internet erreichbar sind. Zu Demonstrationszwecken ist deshalb noch ein Webbrowser-Plug-in für den Webbrowser Firefox geschrieben worden. Dieses zeigt für die Domain der aktuell angezeigten Internetseite die dazu gefundenen Web Services an.

Abbildung 3.5 zeigt einen Screenshot des Plug-ins. Hier ist die Internetseite des Instituts für Telematik aufgerufen worden. Sie ist im oberen Teil des Fensters zu sehen. Das Plug-in extrahiert aus der Adresszeile des Webbrowsers die Domain, in diesem Fall `itm.uni-luebeck.de`. Für diese wird anschließend versucht, im DNS

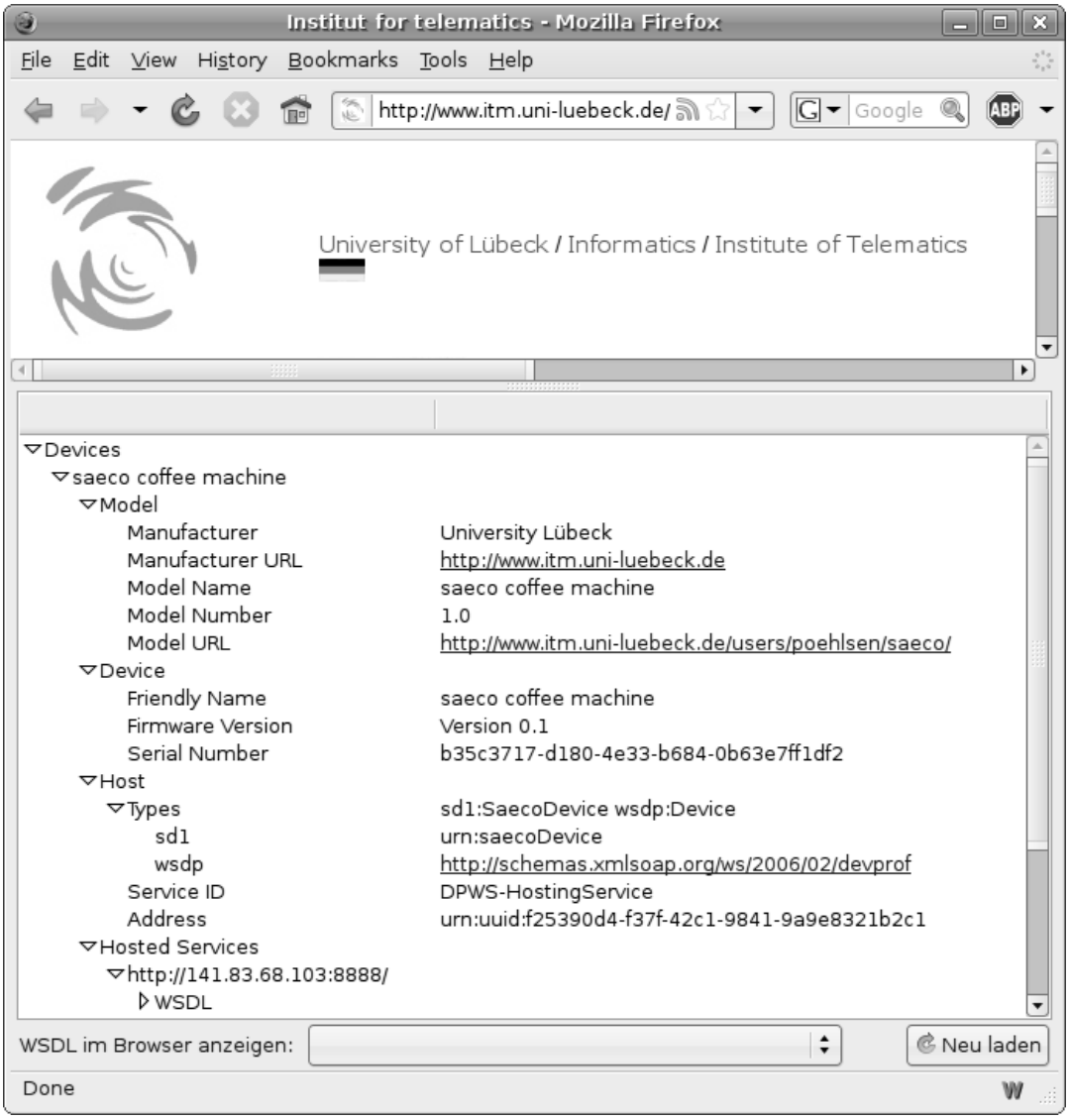



Abbildung 3.5: Screenshot vom Web-Service-Discovery-Plug-in mit aktivierter Ansicht der gefundenen Web Services des Institutes.

einen Discovery-Proxy-SRV-RR zu finden. Ist dies erfolgreich, so ändert sich die Farbe des „W“ unten rechts in der Statuszeile von Grau zu Grün.

Nach einem Mausklick auf das grüne „W“ öffnet sich eine Ansicht, in der die Metadaten aller Web Services angezeigt werden, die der zentrale Discovery Proxy der Internet-Domain zurückliefert. Auf dem Screenshot ist zu sehen, dass sich ein Web-Service-Gerät mit dem Namen „saeco coffee machine“ registriert hat. Hierbei handelt es sich um die Saeco-Kaffeemaschine des Instituts.

Über das Drop-Down-Feld direkt über der Statuszeile lässt sich für die einzelnen Web Services die WSDL-Beschreibung abrufen. Diese wird in einem neuen Webbrowserfenster angezeigt.

Der Firefox-Webbrowser bietet intern keine Möglichkeit, DNS-Anfragen zu stellen. Allein die Auflösung von Domain-Namen zu IP-Adressen wird unterstützt. Das Web-Service-Discovery-Plug-in muss deswegen in Abhängigkeit des darunterliegenden Betriebssystems selbst die DNS-Auflösungen für den SRV RR implementieren. Unter Unix (Linux und Mac OS X) steht hierfür die Funktion `res_query()` zur Verfügung. Für Windows wird auf die Systemfunktion `DnsQuery_UTF8()` der DNS API zurückgegriffen, die dort für DNS-Anfragen zuständig ist.

## 3.4 Bewertung

Beide Realisierungen erfüllen ihre Aufgabe, die Adresse eines zentralen Discovery Proxy zu ermitteln. Der DNS-basierte Ansatz ist zudem nicht nur innerhalb eines Netzwerks nutzbar, sondern auch über das Internet.

### 3.4.1 Bewertung der DHCP-basierten Realisierung

Die Umsetzung auf Basis einer DHCP-Option, die den zentralen Discovery Proxy angibt, erscheint zunächst als die naheliegendere Variante. Technische Details des DHCP-Standards sorgen jedoch für Schwierigkeiten bei der Umsetzung, da nicht-System-Programme eines Computers keine Möglichkeit haben, direkt mit einem DHCP-Server zu kommunizieren. Im Standard ist vorgesehen, dass DHCP-Pakete vom Server an den Client immer an den UDP-Port 68 geschickt werden müssen. Dieser steht aber nur Diensten mit administrativen Rechten zur Verfügung und ist meist schon vom System belegt. Das hat zur Folge, dass DHCP-Optionen grundsätzlich über den bereits laufen DHCP-Client ermittelt werden müssen.

Problematisch ist die DHCP-Variante folglich für Framework-Entwickler. Damit das Framework auf möglichst vielen Plattformen läuft, müssen unterschiedliche DHCP-Clients unterstützt werden. Beim DHCP-Client des ISC können die Daten beispielsweise aus einer Protokoll-Datei gelesen werden. Der Microsoft DHCP-Client unter

Windows bietet dagegen eine API an, um an die benötigte DHCP-Option zu gelangen.

Da DHCP-Anfragen üblicherweise keine Subnetzgrenzen überschreiten, kommt aus administrativer Sicht das Problem hinzu, dass jedes Subnetz eines Unternehmens mit der Adresse des zentralen Discovery Proxy konfiguriert werden muss. Hierbei ist insbesondere nicht sichergestellt, dass bereits verwendete DHCP-Server in der Lage sind, eigene Optionen hinzuzufügen, wie dies beim Server der ISC der Fall ist.

### 3.4.2 Bewertung der DNS-basierten Realisierung

Die in Abschnitt 3.2 vorgestellte Architektur auf DNS-Basis hat gegenüber einer reinen DHCP-Realisierung wesentliche Vorteile. Auf administrativer Seite muss lediglich ein SRV RR im DNS eingetragen werden, um auf einen zentralen Discovery Proxy zu verweisen. Bei diesem Eintrag handelt es sich um einen standardisierten RR, der im Gegensatz zu der EPR-RR-Variante von Snell und Donoho [156] bei DNS-Servern bereits implementiert ist. Daher ist sichergestellt, dass durch die WS-Discovery-Architektur keine zusätzlichen Interoperabilitätsprobleme entstehen.

Für eine DNS-Anfrage wird die lokale Domain benötigt. Ohne diese ist ein Web Service nicht in der Lage, nach dem zentralen Discovery Proxy zu fragen. Abgesehen von manueller Konfiguration ist die lokale Domain über den DHCP-Server zu beziehen. In verwalteten Netzwerken wird sie praktisch immer gesetzt und steht den Clients damit automatisch zur Verfügung.

Die realisierte Architektur macht die Web Services nicht nur innerhalb eines Unternehmensnetzwerks erreichbar, sondern auch weltweit über das Internet. Voraussetzung hierfür ist allerdings, dass sowohl der DNS-Server mit dem SRV RR als auch der zentrale Discovery Proxy aus dem Internet erreichbar sind. Die Funktion des zentralen Discovery Proxy kann in diesem Fall mit einem UDDI-Verzeichnisdienst im Unternehmen verglichen werden, der externe Zugriffe erlaubt.

Auf diese Weise lassen sich Web Services auch außerhalb des Netzwerks sichtbar machen. Dabei ist zu beachten, dass unter Umständen unternehmensinterne Daten preisgegeben werden.

## 3.5 Ergebnis

In diesem Kapitel wurde eine neue hybride Discovery-Architektur für Web Services vorgestellt. Die besondere Herausforderung bestand darin, dass die Architektur in einem sehr dynamischen und gleichzeitig heterogenen Umfeld funktionieren muss. Im Gegensatz zu einem typischen Unternehmensumfeld gibt es im medizinischen Bereich zusätzlich viele kleine Services, die nur ab und zu in Betrieb sind. Hinzu kommt, dass

sie in immer anderen Konstellationen miteinander vernetzt werden, zum Beispiel wenn ein Medizingerät in einem anderen Operationssaal eingesetzt wird.

Die Architektur unterstützt PnP, indem sie auf dem Ad-hoc-Modus von WS-Discovery aufbaut. Auf diese Weise ist es möglich, in sich abgeschlossene, getrennte Netzwerke zu unterstützen. Diese können beispielsweise mit einem Crossover-Ethernet-Kabel für zwei Geräte oder auf Basis eines Switches für mehrere Geräte realisiert werden. Eine Anbindung an das Enterprise-Netzwerk ist nicht erforderlich. Ob die IP-Adressen hierbei vergeben werden (statisch bzw. DHCP) oder Geräte selbst automatisch konfiguriert werden, spielt dabei keine Rolle. Auf den optionalen Discovery Proxy wird in einem lokalen Netzwerk verzichtet, da hierdurch das Auffinden von Services robuster gegenüber Störungen ist und keine eventuell veralteten Informationen zwischengespeichert sind.

Durch die Kompatibilität der Architektur zum Ad-hoc-Modus von WS-Discovery ist gewährleistet, dass auch mit Geräten kommuniziert werden kann, welche die in dieser Arbeit vorgestellte Architektur nicht unterstützen. Dies ist beispielsweise für Netzwerkdrucker interessant, die WS-Discovery bereits im Rahmen von DPWS unterstützen und so auch in dieser Architektur lokal zur Verfügung stehen.

Für Discovery-Szenarien außerhalb eines Subnetzes wird auf einen zentralen Discovery Proxy zurückgegriffen, da die Discovery-Architektur sonst nicht skalieren würde. An diesem melden sich alle zur Verfügung stehenden Services an und ab, sodass jederzeit bekannt ist, welche Services genutzt werden können. Wird ein Gerät einfach ausgeschaltet, so wird dies vom Discovery Proxy nicht erkannt, und er liefert unter Umständen veraltete Service-Einträge. Mit Hilfe des zentralen Discovery Proxy ist es Clients jedoch möglich, im gesamten Unternehmensnetzwerk nach Services zu suchen. Im Krankenhausumfeld ist dies beispielsweise für die Gerätewartung oder auch für eine Integration zentraler Services, wie einem KIS, relevant.

Um die Adresse des zentralen Discovery Proxy zu ermitteln, wird diese im DNS hinterlegt. Hierzu wird unter der Domain der Organisation ein SRV RR für den Discovery Proxy angelegt. SRV RRs sind so aufgebaut, dass es möglich ist, bei Bedarf weitere Discovery Proxies zu Redundanzzwecken zu referenzieren.

Anstatt eines DNS-Eintrages kann auch DHCP zur Referenzierung des zentralen Discovery Proxy verwendet werden. Wie sich jedoch gezeigt hat, bietet dieser Ansatz im Gegensatz zur gewählten Architektur mit DNS keine wesentlichen Vorteile. Im Gegenteil, es haben sich ausschließlich Nachteile, wie zum Beispiel ein erhöhter administrativer Aufwand, herausgestellt.

Die in diesem Kapitel entworfene Architektur ist nicht nur für eine SOA zur Vernetzung von medizinischen Geräten geeignet, auch wenn sie für diesen Zweck entworfen wurde. Der DNS-Eintrag für den zentralen Discovery Proxy kann auch aus dem Internet erreichbar sein. Erlaubt die Firewall eines Unternehmens den Zugriff auf seinen Discovery Proxy, so können beliebige andere Unternehmen ebenfalls auf diesen

zugreifen. Auf diese Weise kann ein Unternehmen seine Web Services gebündelt an einem Ort publizieren: dem zentralen Discovery Proxy.

Das im Rahmen dieser Arbeit entwickelte Webbrowser-Plug-in demonstriert dies. Es zeigt für die aktuell besuchte Domain mithilfe des dazugehörigen Discovery Proxy alle Web Services des Unternehmens an. Oft liegt für Web-Service-Entwickler nahe, wo bestimmte Services angeboten werden. Ein Web Service zum Bestellen von Büchern kann beispielsweise bei Amazon erwartet werden.

Bei UDDI handelt sich um eine Technologie zur Implemtierung von universellen Verzeichnisdiensten. Ein Discovery Proxy hingegen ist speziell zum Auffinden von dort registrierten Web Services entwickelt. Es ist präzise definiert, wie dieser zu verwenden ist: Es wird ein Type (QName) angegeben, und der Discovery Proxy liefert die dazu gefundenen Web Services. Bei UDDI hingegen ist die Handhabung von Web Services unzureichend spezifiziert. Innerhalb eines Unternehmens ist ein zentraler Discovery Proxy daher UDDI vorzuziehen.

# Kapitel 4

# Optimierung der Dienstgüte in Netzwerken

Heutzutage werden immer mehr Systeme vernetzt. Die Anforderungen an das Netzwerk steigen dabei zunehmend. Bei einer klassischen Punkt-zu-Punkt-Verbindung steht die Leitung ausschließlich den beiden Teilnehmern zur Verfügung und sie können allein entscheiden, wie sie diese verwenden. In größeren Netzwerken dagegen wird meist vorgegeben, welche Dienstgüte-Kriterien das Netzwerk erfüllt. Das kann z. B. ein kompletter Verzicht auf Dienstgüte-Kriterien im Sinne einer Best-Effort-Übertragung sein, oder es erfolgt eine pauschale Priorisierung einzelner Dienste. Die einzelnen Netzwerkteilnehmer müssen sich so gut wie möglich damit arrangieren.

Im IT-Umfeld hat sich *TCP/IP* für die Vernetzung von Computersystemen durchgesetzt. Dabei handelt es sich nicht um ein einzelnes Protokoll, sondern um eine Protokollfamilie. Die drei wichtigsten Protokolle sind das *Internet Protocol (IP)* [142], das *Transmission Control Protocol (TCP)* [143] und das *User Datagram Protocol (UDP)* [141]. TCP/IP setzt bei kabelgebundenen Verbindungen für die darunter liegenden Schichten üblicherweise auf Ethernet [62] auf. Für drahtlose Kommunikation wird oft ein *Wireless Local Area Network (WLAN)* [59] verwendet.

Im klinisch-medizinischen Bereich setzt sich derzeit ebenfalls die Kombination aus TCP/IP und Ethernet durch, und es werden immer mehr Medizingeräte mit einer solchen Netzwerkschnittstelle ausgestattet. Dies ermöglicht unter anderem eine Anbindung an das Enterprise-Netzwerk des Krankenhauses. Die Kommunikation mit dem KIS stellt keine besonderen Netzwerkanforderungen. Hier sind die im Enterprise-Bereich üblichen Wiederholungen der Nachrichtenübertragung mithilfe von Reliable-Messaging-Spezifikationen ausreichend.

Bei der Interaktion von Medizingeräten untereinander gibt es höhere Anforderungen an ein Netzwerk. Im skizzierten Anwendungsfall (siehe Abschnitt 1.3) sollen ein Anästhesiegerät und ein Röntgengerät miteinander kommunizieren. Je nach Beatmungsmodus und Röntgenaufnahme reicht eine Synchronisation der beiden Geräte nicht aus. Unter

Umständen muss deshalb die künstliche Beatmung durch das Röntgengerät unterbrochen werden. Um eine möglichst hohe Patientensicherheit zu gewährleisten, ist es notwendig, die Übertragung der Nachrichten zu garantieren.

Prinzipiell ist die heute zur Verfügung stehende Datenrate bei Ethernet für die Vernetzung von Medizingeräten ausreichend. Es kann jedoch vorkommen, dass einzelne Geräte die gesamte Datenrate beanspruchen und so möglicherweise wichtigen anderen Datenverkehr verzögern oder vollständig blockieren. Hierfür können beispielsweise größere Dateitransfers der bildgebenden Medizin verantwortlich sein.

Es ist aber auch vorstellbar, dass einzelne Systeme im Netzwerk mit einem Computervirus oder -wurm befallen sind und nicht mehr korrekt funktionieren. Da in vielen Medizingeräten derzeit weitverbreitete Betriebssysteme verwendet werden und diese nicht immer vom Betreiber aktualisiert werden dürfen, steigt das Risiko hierfür erheblich. Im ungünstigsten Fall versucht ein mit einem Computerwurm befallenes Gerät, weitere Geräte zu infizieren, und blockiert dadurch die Netzwerkverbindung vollständig.

In diesem Kapitel werden zunächst die Grundlagen zur Dienstgüte erläutert, die bei Ethernet und TCP/IP bereits vorhanden sind, um die Übertragung in einem Netzwerk zu verbessern. Danach werden verwandte Technologien und Arbeiten vorgestellt, die sich mit dem Thema Echtzeit-Ethernet befassen, um einen Überblick über bestehende Lösungen zu geben.

Anschließend wird eine selbst entworfene faire Switch-Architektur vorgestellt, die auch bei fehlerhaften Sendern im Netzwerk Übertragungsgarantien gibt und keinerlei Modifikation aufseiten der Geräte benötigt. In dem darauffolgenden Abschnitt wird die dazugehörige Implementierung auf einem Linux-System mit mehreren Netzwerkkarten vorgestellt. Diese wird mit einem Störsender, der die Verbreitung eines Computerwurmbefalls simuliert, evaluiert. Das Kapitel endet mit einer Zusammenfassung der Ergebnisse.

Die Idee der fairen Switch-Architektur für Übertragungsgarantien sowie wesentliche Ergebnisse dazu wurden in [134] vom Autor bereits vorab veröffentlicht.

## 4.1 Grundlagen

Bei Netzwerken auf Basis von Ethernet und TCP/IP handelt es sich um *Best-Effort*-Netzwerke. Darunter ist zu verstehen, dass die Datenübertragung so gut wie möglich ausgeführt wird, jedoch keinerlei Garantien für eine Übertragung gegeben werden.

Echtzeitsysteme benötigen dagegen Garantien für ihre Datenübertragung. Unter Echtzeit wird hierbei verstanden, dass ein System vor Ablauf einer festen Zeitgrenze, der sogenannten *deadline*, auf ein Ereignis reagiert. Im Rahmen der Datenübertragung

bedeutet dies, dass sie in einer vorgegebenen Zeitspanne erfolgreich stattgefunden haben muss. Die dadurch entstehenden Anforderungen werden üblicherweise unter dem Begriff *Quality of Service (QoS)*, auch Dienstgüte genannt, zusammengefasst.

### 4.1.1 Ethernet-Dienstgüte

Bei Ethernet [62] wird das Netzwerk als sogenanntes *shared medium* verwendet, d. h., alle Teilnehmer nutzen gemeinsam die zur Verfügung stehende Bandbreite. Der Algorithmus für den Zugriff auf das Medium ist *Carrier Sense Multiple Access/Collision Detection (CSMA/CD)*. Darunter ist zu verstehen, dass ein Teilnehmer, der Daten senden möchte, zunächst ermittelt, ob auf dem Medium bereits gesendet wird. Ist dies der Fall, so wartet er eine zufällige Zeit und versucht es dann erneut.

Durch den hierbei entstehenden Nichtdeterminismus des CSMA/CD-Algorithmus ist Ethernet nicht echtzeitfähig. Derzeit wird das Zugriffsproblem durch Vollduplex-Betrieb in Verbindung mit einem voll geswitchten Netzwerk umgangen. Alle Netzwerkteilnehmer erhalten dabei eine eigene Verbindung zum Switch, der für die Kollisionsvermeidung zuständig ist.

Damit ein Ethernet-Switch Kollisionen vermeiden kann, muss er die empfangenen Daten gegebenenfalls zwischenspeichern, bevor er sie zum eigentlichen Empfänger weiterleitet. Der Nichtdeterminismus für den Medienzugriff verschiebt sich hierdurch in den Queueing-Mechanismus des Switches.

Managebare Switches bieten meist die Möglichkeit, die Dienstgüte mittels einer Priorisierung zu erhöhen. Dabei wird die Reihenfolge des Datenversandes vom Switch beeinflusst. Die Priorisierung kann statisch oder dynamisch erfolgen. Bei der statischen Priorisierung werden feste Regeln innerhalb des Switches vorgegeben. Für die dynamische Priorisierung ist es notwendig, dass in den eingehenden Daten, den *Ethernet Frames*, eine Priorisierung angegeben ist. Dies erfordert auf allen betroffenen Netzwerkknoten eine Unterstützung der Ethernet-Erweiterung *Virtual Local Area Network (VLAN)* nach IEEE 802.1Q [61].

Diese Erweiterung dient primär der logischen Einteilung eines Netzwerks in mehrere, voneinander unabhängige Teil-Netzwerke. Die Erweiterung sieht in jedem Ethernet Frame ein sogenanntes *VLAN-Tag* vor. Abgesehen von der Angabe eines VLAN-Bezeichners stehen drei Bit für die Angabe einer Priorität zur Verfügung, dem *Priority Code Point (PCP)*.

Wie dieser PCP zu interpretieren ist, wird unter anderem in IEEE 802.1D angegeben [60]. Standardmäßig ist das Feld auf 0 gesetzt. Eine Priorität wird mit Werten von 1 bis 7 angegeben, wobei 7 der höchsten Priorität entspricht. Bei der Priorität handelt es sich um sogenannte *Traffic Types* von *Background* (1) bis zu *Network Control* (7).

Je nachdem, wie viele Queues zur Verfügung stehen, können die Daten in unterschiedliche Klassen aufgeteilt werden, wobei jeder Queue eine Klasse zugeordnet wird. Im Idealfall gibt es für jeden Typ eine eigene Klasse, andernfalls müssen sich mehrere Typen eine Klasse bzw. Queue teilen. Durch die Einteilung der Daten in Klassen wird bei dieser Art der Priorisierung auf Ethernetebene auch von *Class of Service (CoS)* gesprochen. Eine solche Priorisierung erhöht zwar die Dienstgüte für einzelne Service-Klassen, jedoch werden auf diese Weise noch keine Garantien für eine Datenübertragung gegeben.

### 4.1.2 IP-Dienstgüte

Auch auf IP-Ebene kann die Dienstgüte beeinflusst werden. Hierfür ist bei IPv4 [142] im Header das zweite Byte vorgesehen. Ursprünglich wurde das Feld bei IPv4 für die Angabe des *Type of Service (ToS)* [3] standardisiert, während es heutzutage für *Differentiated Services (DiffServ)* [98] und *Explicit Congestion Notification (ECN)* [148] spezifiziert ist.

Das ToS-Feld wird vom sendenden System gesetzt. Die ersten drei Bit (Bit 0–2) geben dabei die Priorität an. Hier sind Werte von 000 (Routine) bis 111 (Network Control) möglich, ähnlich den Typen bei Ethernet. Das folgende Bit (Bit 3) wird gesetzt, wenn eine niedrige Latenz benötigt wird. Bit 4 gibt an, dass ein hoher Durchsatz gewünscht ist, während Bit 5 eine zuverlässige Übertragung fordert. Eine Minimierung der Kosten kann mit Bit 6 angegeben werden.

DiffServ und ECN lösen nicht nur das ToS-Feld im IPv4-Header ab, sondern spezifizieren zugleich, wie das *Traffic-Class*-Feld bei IPv6 [24] gesetzt werden soll. Die ersten sechs Bit erhält DiffServ, während die letzten beiden Bit für ECN zur Verfügung stehen.

Bei DiffServ handelt es sich um einen klassenbasierten Mechanismus zum Traffic-Management. Der *DiffServ Code Point (DSCP)* ist sechs Bit lang und ermöglicht somit die Unterscheidung von 64 Klassen ($2^6$). Um ein möglichst einheitliches Verständnis über die Bedeutung der einzelnen Klassen zu erreichen, sind bereits einige DSCPs spezifiziert.

Das *Default per-hop behavior (Default PHB)* mit dem Code Point 000000 muss von allen DiffServ-kompatiblen Routern unterstützt werden. Hierbei handelt es sich um die normale Best-Effort-Weiterleitung.

Als Vorgänger von DiffServ kann die Priorität (Bit 0–2) aus dem ToS-Feld angesehen werden. Hierbei handelt es sich bereits um eine Klasseneinteilung. Diese ist in das *Class Selector PHB* übernommen worden und hat die Code Points $xxx000$. Die ersten drei Bit des ToS-Feldes wurden also unverändert in DiffServ übernommen.

Weitere PHBs sind möglich und können durch weitere Spezifikationen ergänzt werden. Die *Assured Forwarding (AF) PHB* [52] beinhaltet vier Klassen mit unterschiedlicher Priorität. Die Pakete innerhalb einer dieser Klassen werden nochmals in drei Stufen eingeteilt (*drop precedence*), die angeben, welche von ihnen bei Überlastung ihrer Klasse zuerst verworfen werden sollen. Auf diese Weise entstehen 12 Klassen für das AF PHB. Die *Expedited Forwarding (EF) PHB* [21] hat nur eine Klasse (Code Point 101110). Diese soll priorisiert übertragen werden, um so eine niedrige Latenz, geringe Latenzschwankungen sowie eine niedrige Paketverlustrate zu gewährleisten.

Die beiden Bit, die dem DSCP folgen, sind für ECN vorgesehen. Sie ermöglichen es einem Router, eine Markierung im IP-Paket zu setzen, welche anzeigt, dass eine Überlastung verursacht wird. Die beteiligten Knoten sollen diese Information dabei so auffassen, als ob das Paket verloren gegangen wäre. Mit ECN lässt sich bei TCP mithilfe des Routers die Geschwindigkeit, mit der gesendet wird, kontrolliert herunterregeln.

Inzwischen unterstützen alle modernen Betriebssysteme ECN, jedoch ist es standardmäßig deaktiviert. Der Grund hierfür sind fehlerhafte Implementierungen, welche IP-Pakete verwerfen, in denen ECN-Support aktiviert ist [89]. Da hierdurch manche TCP-Verbindungen verhindert werden, ist es praktikabler, ECN zu deaktivieren.

Für Dienstgüte auf IP-Ebene lässt sich zusammenfassend festhalten, dass DiffServ mit seinem klassenbasierten Ansatz ähnlich funktioniert, wie dies bei Ethernet der Fall ist. Es hebt sich durch eine höhere Anzahl an Klassen (64 statt 8) sowie durch eine bessere Semantik der Klassen von Ethernet-PCPs ab. ECN ist aufgrund bestehender Inkompatibilitäten ausschließlich in selbst verwalteten Netzwerken sinnvoll zu nutzen. Durch ECN werden Empfänger von Paketen lediglich informiert, dass eine Überlastung des Netzwerks vorliegt. Folglich liefern sowohl die Klassifizierung als auch ECN keine Übertragungsgarantien. Es ist ausschließlich eine Priorisierung von Datenverkehr möglich.

## 4.2 Verwandte Arbeiten und Technologien

Seit den 1980er Jahren werden für die Kommunikation im industriellen Sektor Echtzeit-Netzwerke verwendet. Es gibt verschiedene solcher Netzwerk-Technologien, die unter dem Namen Feldbus bekannt sind. *Profibus* [147] ist derzeit einer der verbreitetsten. Ein weiterer geläufiger Feldbus ist *Controller Area Network (CAN)* [13].

Feldbusse wurden entwickelt, um die Parallelverkabelung in Anlagen und Fabriken zu ersetzen und auf diese Weise die Kosten zu senken. CAN und Profibus können für die Datenübertragung zum Beispiel RS-485 [6] verwenden, das bei Feldbussen gebräuchlich ist. Es handelt sich dabei um einen Schnittstellen-Standard für die serielle Datenübertragung über ein Adernpaar.

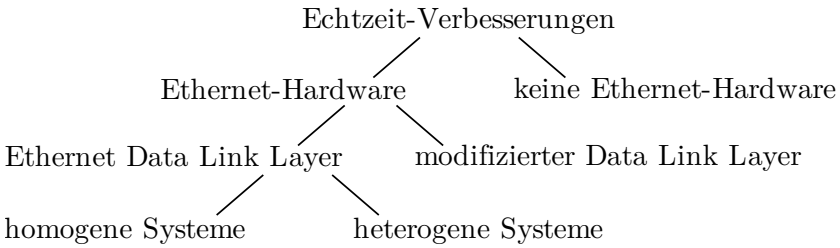



Abbildung 4.1: Klassifikationshierarchie für Industrial Ethernet

Zunehmend werden Echtzeit-Ethernet- bzw. *Industrial-Ethernet*-Lösungen anstelle klassischer Feldbusse verwendet. Dabei handelt es sich nicht um einen einzelnen Standard, sondern um zahlreiche inkompatible Ansätze, die miteinander konkurrieren. Detaillierte Beschreibungen bestehender Standards sind unter anderem in [75], [23] und [41] zu finden.

Zur besseren Vergleichbarkeit werden die verschiedenen Ansätze in dieser Arbeit klassifiziert. In Abbildung 4.1 ist eine hierarchische Struktur skizziert.

Einige Industrial-Ethernet-Ansätze haben nur die Verkabelungstechnik mit standardisierter Ethernet-Hardware gemeinsam. Die verbauten Chips auf den einzelnen Knoten im Netzwerk sind spezielle ASICs (*Application Specific Integrated Circuits*). Der bekannteste Vertreter dieser Klasse ist EtherCAT (Ethernet for Control Automation Technology) [38].

Ein EtherCAT-Master-Knoten sendet spezielle Ethernet Frames, die von einem Slave-Knoten zu anderen weitergeleitet werden. Dies erfolgt bereits während des Empfangs der Frames, sodass diese nicht zwischengespeichert werden. Es werden lediglich einige Daten (beispielsweise 2 Bit) vom Slave eingefügt oder gelesen. Am Schluss sendet der letzte Slave in der Reihe den Frame wieder zurück, sodass der erste Slave dem Master einen sogenannten *Response Frame* übergibt. EtherCAT ermöglicht damit eine kurze Zyklendauer mit geringen Schwankungen, um so eine möglichst exakte Synchronisation der Knoten zu erreichen.

Andere Lösungen verwenden herkömmliche Ethernet-Hardware und unterscheiden sich in der Art des Medienzugriffs. Anstatt des bei Ethernet üblichen CSMA/CD-Algorithmus wird hier auf ein deterministisches Verfahren gesetzt, um so Übertragungsgarantien zu ermöglichen. Powerlink [39] ist ein Beispiel für diesen Ansatz. Ein *Managing Node (MN)* sendet am Anfang eines jeden Zyklus einen speziellen *Start-of-Cycle*-Frame. Anschließend werden alle übrigen *Controlled Nodes* der Reihenfolge nach aufgefordert, einen Powerlink Ethernet Frame zu senden. Dieser isochronen Phase schließt sich eine asynchrone Phase an, in der jeder Knoten maximal ein Ethernet Frame mit TCP/IP-Daten verschicken darf. Auf diese Weise ist es möglich, zusätzlich TCP/IP in einem Powerlink-Ethernet zu verwenden.

In der Forschung werden weitere Ansätze verfolgt, die Dienstgüte bei Ethernet zu optimieren, ohne den *Data Link Layer* mit CSMA/CD zu verändern. Die entsprechenden Arbeiten lassen sich in Lösungen einteilen (vgl. Abbildung 4.1), bei denen alle Knoten im Netzwerk angepasst werden müssen (homogene Systeme), und solche, die auch mit unmodifizierten Netzwerk-Teilnehmern zurechtkommen (heterogene Systeme).

Bei RETHER [165] handelt es sich um eine Lösung für ein homogenes System, das ein Token-Protokoll einsetzt. Dadurch, dass nur der Inhaber des Tokens senden darf, kommt es auf dem Medium nicht mehr zu Kollisionen. In Verbindung mit einem deterministischen Verfahren zum Weiterreichen des Tokens, können so Übertragungsgarantien gegeben werden.

Kweon et al. [80] verfolgen einen anderen Ansatz für ein homogenes System. In der Automatisierungstechnik werden in der Regel periodisch kleine Datenmengen versendet. Hierbei handelt es sich meist um Abtastwerte von einzelnen Controllern. Da diese Werte üblicherweise überabgetastet werden, schadet es nicht, wenn ab und zu ein Paket verloren geht. Es genügt, wenn eine statistische Übertragungsgarantie gegeben wird. Um dies zu gewährleisten, muss sporadisch auftretender Datenverkehr, bevor er ins Netzwerk gelangt, von dem sendenden Knoten geglättet werden. Diese Art der Übertragungsgarantie kann nur dann funktionieren, wenn alle Knoten kooperativ sind, es sich also um ein homogenes System handelt.

Für heterogene Systeme müssen Switches verwendet werden. Nur so lässt sich der nichtdeterministische Zugriff von CSMA/CD umgehen. EtheReal [164] verwendet einen speziellen Switch, um benötigte Datenraten bereits vor einer Übertragung zu reservieren. Hierzu muss auf dem sendenden System mithilfe des EtheReal-Echtzeit-Daemon die durchschnittliche Datenrate sowie deren maximale Burst-Größe für die Verbindung auf dem Switch reserviert werden. Anschließend steht eine Proxy-IP zur Verfügung, um Daten über die reservierte Leitung zu übertragen. Ist die zur Verfügung stehende Datenrate bereits ausgebucht, so sind keine weiteren Reservierungen für Echtzeit-Übertragungen möglich.

Es existieren auch sogenannte Echtzeit-Ethernet-Lösungen, die auf Ethernet und TCP/IP aufsetzen. Diese können folglich keine Übertragungsgarantien liefern und werden in der Automatisierungstechnik für die manuelle Bedienung durch Menschen verwendet [42]. Zu diesen gehören beispielsweise *MODBUS TCP/IP* [95] und *EtherNet/IP* [40]. Bei MODBUS TCP/IP handelt es sich um eine Erweiterung für die klassische Variante von MODBUS, die eine serielle Verbindung benutzt.

EtherNet/IP spezifiziert ein Protokoll für die Applikationsebene. Dies ist kompatibel zu den seriellen Feldbussen DeviceNet und ControlNet. Um Echtzeitkommunikation zu ermöglichen, gibt es *CIP Sync* als Erweiterung zur Zeitsynchronisation auf Basis des *Precision Time Protocol (PTP)*, das in IEEE 1588 [58] spezifiziert ist. Nach einer Synchronisierung der Geräte können mit Zeitstempeln versehene Steuerbefehle verschickt werden, um so eine koordinierte Aktion auszuführen. Um einen möglichst

zuverlässigen Ablauf innerhalb eines Netzwerks mit weiterem Datenverkehr zu gewährleisten, sollen die Pakete von EtherNet/IP in einem voll geswitchten Netz bevorzugt weitergeleitet werden. Hierfür wird auf das VLAN-Tagging nach IEEE 802.1Q, wie im Abschnitt 4.1.1 über Ethernet-Dienstgüte erläutert, zurückgegriffen.

## 4.3 Kernidee für Übertragungsgarantien

Die Kernidee der in diesem Abschnitt vorgestellten Switch-Architektur besteht darin, die vorhandene Datenrate im Netzwerk fair, d. h. gleichmäßig, zwischen allen Teilnehmern aufzuteilen. Jedes medizinische Gerät, das am Switch angeschlossen ist, bekommt dabei die gleiche Übertragungsgarantie, es gibt keine Bevorzugung. Aufgrund der hohen Datenrate von Ethernet im Vergleich zu seriellen Verbindungen ist die garantierte Datenrate sowohl für die Steuerung von anderen Geräten als auch für die Visualisierung von Echtzeit-Messwerten auf anderen Geräten ausreichend. So erhalten beispielsweise 100 Teilnehmer bei Fast Ethernet (100 MBit/s) für jede Punkt-zu-Punkt-Verbindung eine Übertragungsgarantie von 1 MBit/s.

In Ethernet-Switches erfolgt üblicherweise eine Klassifizierung des Datenverkehrs anhand seines Service-Typs. Das funktioniert nur, so lange keine Fehlklassifikationen stattfinden und keine Klasse überbucht wird. Sind alle Pakete in der höchsten Prioritätsklasse, so hat das die gleiche Wirkung, wie der komplette Verzicht auf Priorisierung.

Die vorhandene Netzwerkkapazität lässt sich beliebig aufteilen, von einer fairen Verteilung der Geräte untereinander, bis hin zu einer fairen Verteilung nach Service-Typ. Abbildung 4.2 verdeutlicht dies: 60 % der Datenrate werden zwischen den Geräten aufgeteilt, während die restlichen 40 % für eine Verteilung zwischen den Services zur Verfügung stehen. Je nach Anwendungsbereich kann das Verhältnis unterschiedlich gewählt werden.

Aktuelle Switches unterstützen die Verteilung zwischen Services, um auf diese Weise Echtzeit-Traffic dem normalen Datenverkehr vorzuziehen. Die hier vorgestellte Switch-Architektur dagegen konzentriert sich ausschließlich auf eine gleichmäßige Verteilung der Datenrate auf die einzelnen Geräte. Sollte ein Gerät die ihm garantierte Datenrate nicht verwenden, so steht sie den restlichen Geräten gleichmäßig zur Verfügung. Dies ermöglicht es den Geräten, mit einer höheren Datenrate zu übertragen als ihnen garantiert wird.

Da an einem Switch mehrere Pakete zur gleichen Zeit ankommen können, muss er Pakete zwischenspeichern können. Switches können Pakete am Eingang oder Ausgang eines Ports sowie an beiden in sogenannten Queues speichern. Dies ist abhängig davon, ob die Switching-Entscheidung vor oder nach dem Zwischenspeichern erfolgt, bzw. ob sowohl davor als auch danach gespeichert wird.

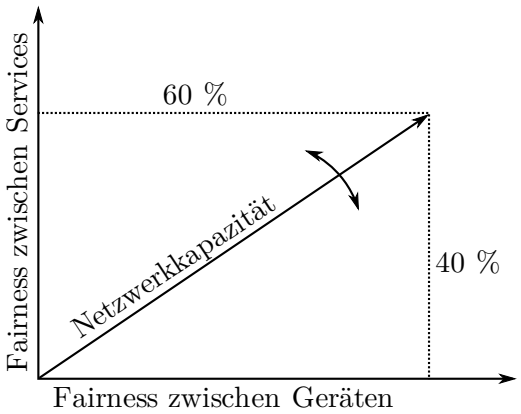



Abbildung 4.2: Die Netzwerkkapazität kann für eine faire Verteilung zwischen Services oder Geräten verwendet werden

Switches mit Output-Queues ermöglichen den maximalen Durchsatz. Alle Pakete werden sofort in die Queue des zum Ziel gehörenden Ports eingereiht. Das Problem hierbei ist, dass ein Störsender die Output-Queue eines Ports mit seinen Paketen füllen kann und somit Pakete anderer Teilnehmer verdrängt. Switches, die Ethernet-CoS unterstützen, haben deswegen an jedem Port nicht nur eine Output-Queue, sondern entsprechend der Anzahl der unterstützen Prioritätsklassen mehrere, um so eine gegenseitige Beeinflussung zu verhindern.

Input-Queueing wird in Switches immer dann verwendet, wenn der Switch nicht schnell genug ist, um von allen Ports eingehende Pakete gleichzeitig zu verteilen. Die Pakete werden in diesem Fall so lange in den Input-Queues aufbewahrt, bis sie verteilt werden können. Handelt es sich um reine Input-Queueing-Switches, so sind am Ausgang eines Ports keine Queues vorhanden und es entstehen unter Umständen unnötige Wartezeiten. Dies ist immer dann der Fall, wenn das zweite Paket in einer Input-Queue an einen momentan freien Port weitergeleitet werden könnte, jedoch erst warten muss, bis das erste Paket verschickt ist. Aus diesem Grund werden Input-Queues meist nur in Verbindung mit Output-Queues verwendet.

Abbildung 4.3 zeigt die entworfene Switch-Architektur für einen Switch mit $n$ Ports. Um den Paketfluss besser darstellen zu können, ist der Eingang eines Ports auf der linken Seite und der dazugehörige Ausgang des selben Ports auf der rechten Seite abgebildet. Die Switch-Architektur muss eine gegenseitige Beeinflussung, d. h., ein Verdrängen von eingehenden Paketen anderer Ports, verhindern, um eine faire Verteilung der Datenrate nicht zu behindern. Folglich werden für jeden der $n$ „Eingänge“ eines Switches eine eigene Queue an jedem der $n$ „Ausgänge“ benötigt. Das hat zur Folge, dass ein Switch mit $n$ Ports insgesamt $n^2$ Queues benötigt, das sind $n$ pro Port.

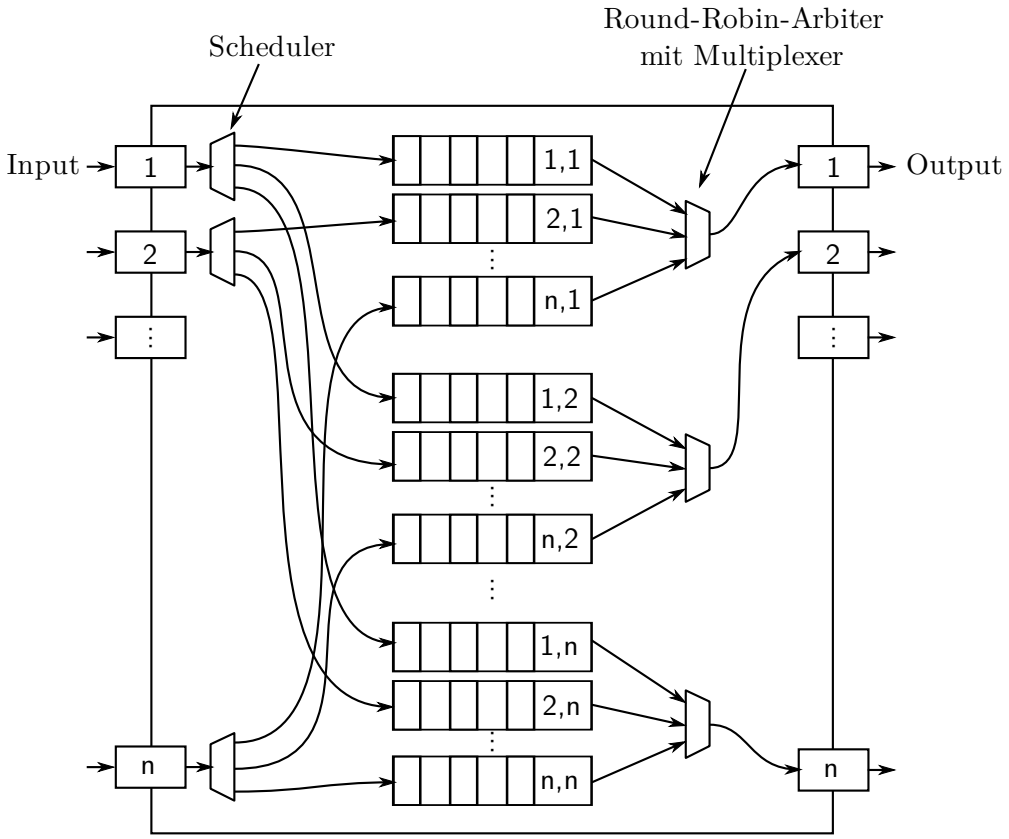



Abbildung 4.3: Architektur des fairen Switches mit $n^2$ Queues

Ethernet Frames kommen auf der linken Seite der Abbildung an einem Port an und werden dann von dem dazugehörigen *Scheduler* in die passende Queue für den Ziel-Port weitergeleitet. Da die Switching-Entscheidung vor dem Zwischenspeichern getroffen wird, handelt es sich bei dem entwickelten Switch um einen Output-Queueing-Switch.

In der Output-Queue werden Ethernet Frames so lange zwischengespeichert, bis sie vom Multiplexer für den nächsten Versand vorgesehen sind. Der Multiplexer wird von einem Round-Robin-Arbiter gesteuert. Dabei handelt es sich um einen Prioritätencoder, der reihum die einzelnen Queues auswählt und jeweils ein Ethernet Frame pro Queue verschicken lässt. Ist eine Queue leer, so wird gleich mit der nächsten fortgefahren, sodass keine Bandbreite verschwendet wird. Diese Art der Bandbreitenverteilung nennt sich *max-min fairness* [87].

Durch diese Architektur ist jedem direkt am Switch angeschlossenen Knoten pro Ziel-Port eine Übertragungsrate von $1/n$ der vorhandenen Datenrate garantiert. Bei einem Fast-Ethernet-Switch mit 96 Ports entspricht der garantierte Anteil über 1 MBit/s für jede einzelne Verbindung. In Summe wird einem Knoten damit sogar die volle Datenrate garantiert, so lange er alle Ports gleichmäßig anspricht.

Werden zwei solcher Switches hintereinander geschaltet, so sinkt die garantierte Übertragungsrate deutlich. Es wird in diesem Fall nur noch $\frac{1}{n^2}$ garantiert. Bei zwei 96-Port-Switches entspricht das für Geräte, die über zwei voll ausgelastete Switches miteinander verbunden sind, nur noch etwas mehr als 10 KBit/s.

## 4.4 Implementierung eines fairen Switches

Die vorgestellte faire Switch-Architektur ist implementiert worden, um sie mit bestehender Switch-Hardware zu vergleichen. Als Hardware-Basis wird ein Desktop-Rechner mit drei Intel-Netzwerkkarten vom Typ *EtherExpress FastEthernet* verwendet. Als Software wird Debian GNU/Linux eingesetzt. Der Linux-Kernel unterstützt Ethernet-Bridging mit individuell konfigurierbarem Scheduling [54]. Zur Bridge-Konfiguration wird *bridge control* (`brctl`) verwendet, während das faire Scheduling mit dem *Traffic Controller* (`tc`) umgesetzt wird.

In Abbildung 4.4 ist der vollständige Aufbau der Implementierung eines 3-Port-Switches skizziert. Die Ethernet Frames durchlaufen die Struktur dabei von links nach rechts. Die eigentliche Umsetzung des fairen Switches erfolgt dabei in einer sogenannten *classful queueing discipline (qdisc)*, die in der Abbildung im oberen Bereich vergrößert dargestellt ist.

Eine classful Qdisc hat die Aufgabe, Ethernet Frames in unterschiedliche Klassen einzuteilen, anstatt sie lediglich in eine gemeinsame Queue zu schreiben. Bei einer solchen Klasse handelt es sich im einfachsten Fall um eine Queue. In der Abbildung sind diese mit `1:10`, `1:11` und `1:12` bezeichnet. Soll eine komplexere Klassifizierung durchgeführt werden, so kann eine Klasse selbst wieder eine Qdisc sein und die Frames in weitere Unterklassen einteilen. Auf diese Weise entsteht eine Baumstruktur, bei der alle Blätter einer Queue und die übrigen Knoten einer classful Qdisc entsprechen.

Eine classful Qdisc besitzt Filter, um die eingehenden Frames den unterschiedlichen Klassen zuzuteilen. Für den fairen Switch benötigt jeder Port-Ausgang für jeden Port-Eingang eine eigene Queue. Außerdem muss die Qdisc die Filterung anhand des eingehenden Ports durchführen. Da jedoch einem Ethernet Frame nicht zu entnehmen ist, an welchem Port er empfangen wurde, ist es notwendig, diesen zu markieren.

Für diese Markierung wird auf die Ethernet-Firewall *ethernet bridge tables (ebtables)* [155] zurückgegriffen. Diese markiert alle Frames beim Empfang, wie dies auf der linken Seite der Abbildung zu sehen ist. Anschließend trifft der Linux-Kernel eine Switching-Entscheidung und leitet den Frame an die root Qdisc des Ziel-Ports weiter. Die Qdisc ist nun in der Lage, anhand der Markierung den Frame in der passenden Queue abzulegen.

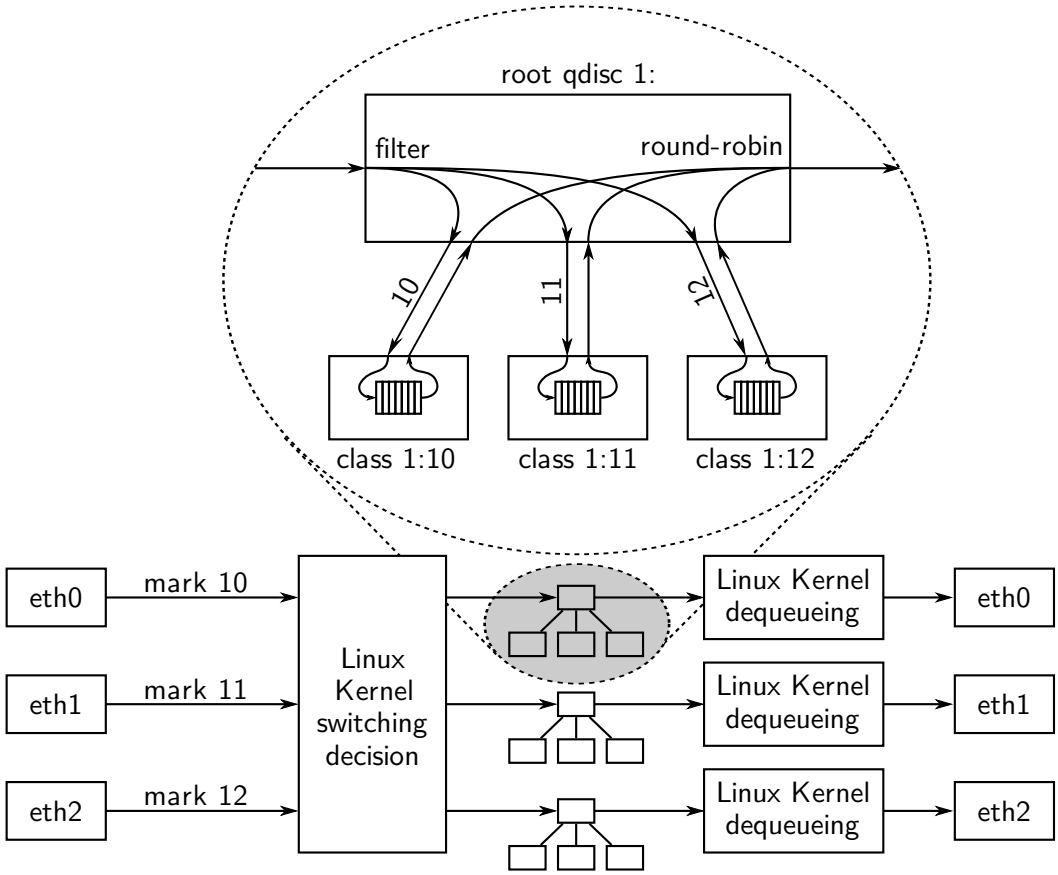



Abbildung 4.4: Aufbau der Switch-Implementierung im Linux-Kernel

Ein Ethernet Frame verbleibt so lange in der Queue, bis der Linux-Kernel von der Netzwerkkarte informiert wird, dass ein neuer Frame verschickt werden kann. Der Linux-Kernel greift anschließend auf die root Qdisc zu, um den nächsten Ethernet Frame anzufordern. Je nachdem, um was für eine Qdisc es sich handelt und wie diese konfiguriert ist, wird aus einer Queue ein Frame entnommen.

Im Linux-Kernel sind Qdiscs mit unterschiedlichen Queueing-Strategien, wie beispielsweise gewichtetem Round-Robin oder hierarchischem Token-Bucket, vorhanden. Gewichtetes Round-Robin unterscheidet sich von Round-Robin dadurch, dass nicht alle Queues nacheinander gleich oft an die Reihe kommen, sondern entsprechend ihrer jeweiligen Gewichtung unterschiedlich häufig. Auf diese Weise lassen sich einzelne Queues anderen gegenüber bevorzugen. Für die faire Switch-Implementierung wird die classful Qdisc mit gewichtetem Round-Robin gewählt, wobei alle Klassen das gleiche Gewicht bekommen. Das sorgt dafür, dass der Switch die gewünschte max-min Fairness erreicht.

## 4.5 Evaluation

Um die Implementierung der Architektur zu bewerten, sind im Rahmen dieser Arbeit mehrere Messungen zum Vergleich der neuen Architektur mit herkömmlichen Geräten durchgeführt worden. Dabei ist zu beachten, dass die zur Verfügung stehenden Switches nicht für das umgesetzte Szenario optimiert sind. Die Ergebnisse repräsentieren folglich nicht die Grundperformance eines Switches, sondern ausschließlich sein Verhalten im umgesetzten Szenario.

Die folgenden Switch-Modelle standen für die Evaluation zur Verfügung:

- Managebare Geräte:
  - 3Com Corporation, SuperStack II Switch 3000, 10Base-T/100Base-TX 12 Port
  - Extreme Networks, Summit 24 (Extreme 13011), expandable 24 Port Fast Ethernet Switch
  - Cisco Systems Inc., WS-C2916M-XL, 2900 Series, expandable 16 Port Ethernet Switch
- Consumer Geräte:
  - Allied Telesis Inc., AT-FS708, 10Base-T/ 100Base-TX 8 Port Ethernet Switch
  - D-Link Corporation, DES-1008D (G2), 8-Port 10/100 Fast Ethernet Switch
  - Netgear Inc., ProSafe FS108v2, 8 Port Fast Ethernet Switch

Alle Switches werden in ihren Werkeinstellungen betrieben und werden im Folgenden ohne namentliche Zuordnung verwendet. Es ist nicht das Ziel, einzelne Produkte zu vergleichen, gerade da ein sehr spezieller Fall untersucht wird.

Zusätzlich zur fairen Switch-Implementierung aus dem vorherigen Abschnitt wird der Linux-Rechner mit einem Standard-Bridging-Setup aufgesetzt. Dabei handelt es sich um einen Switch mit einer Output-Queue pro Port, der damit gut als Vergleich verwendet werden kann.

### 4.5.1 Testaufbau

Das Schlimmste, was ein Gerät im Netzwerk verursachen kann, ist, dass es dieses mit Datenpaketen überlastet. Deswegen haben zahlreiche Business-Switches eine sogenannte *Flood Protection*. Diese ist jedoch meist nur in der Lage, Broadcast- und Multicast-Nachrichten zu unterdrücken. Die Flood Protection verwirft hierzu bei starker Netzwerkauslastung diese Nachrichten, um das Netzwerk zu entlasten. Nachrichten, die direkt an einzelne Geräte gesendet werden, lassen sich auf diese

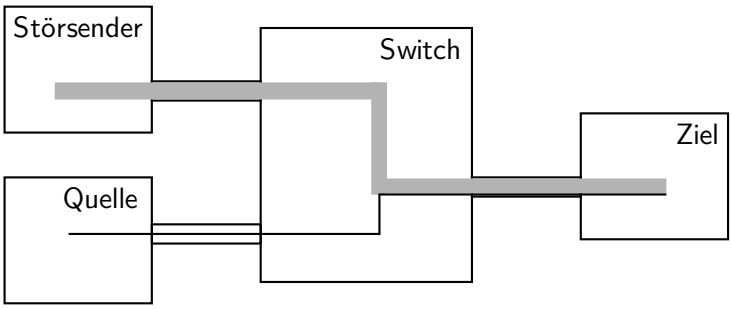



Abbildung 4.5: Versuchsaufbau für die Switch-Evaluation

Weise nicht unterdrücken, da ein Switch nicht unterscheiden kann, ob es sich um einen regulären Datenstrom handelt oder ob das Gerät überlastet wird. Bei starkem Broadcast- und Multicast-Verkehr wird bei einer Flood Protection dagegen von einem Missbrauch des Netzwerks ausgegangen.

Eine Priorisierung mittels CoS, wie Ethernet sie anbietet, kann in diesem Fall nicht helfen, da das störende Gerät die Pakete mit höchster Priorität verschicken kann. Um den Testaufbau zu vereinfachen, wird deswegen ohne CoS gemessen. Das Einzige, was den Schaden durch einen Störsender begrenzen kann, ist eine faire Aufteilung der Datenrate, da er die Nutzdaten anderer nicht mehr verdrängen kann.

In Abbildung 4.5 ist der Versuchsaufbau abgebildet. An einem Switch sind drei Linux-Computer direkt mit Vollduplex per Fast Ethernet angebunden. Auf dem Computer des Störsenders werden mithilfe von *netcat* im UDP-Modus unentwegt IP-Pakete generiert, die an den Zielrechner geschickt werden, um die Netzwerkverbindung voll auszulasten. Der TCP-Algorithmus würde hingegen seine Datenrate zurückfahren, wenn anderer Datenverkehr auf der Leitung herrscht, um so eine Überlastung zu vermeiden. Eine TCP-Verbindung ist daher auf Seiten des Störsenders für den Versuchsaufbau nicht geeignet.

Für die Messungen zwischen Quelle und Ziel werden Iperf und Netperf verwendet, die beide in der Debian-Distribution enthalten sind.

### 4.5.2 TCP-Durchsatzmessungen

In Fast-Ethernet-Netzwerken liegt der TCP-Durchsatz ohne Störsender laut Netperf zwischen 93,8 und 94,0 MBit/s, unabhängig davon, welcher Switch verwendet wird. Mit einem Störsender reduziert sich der Durchsatz drastisch. In Tabelle 4.1 sind die Messwerte für den TCP-Durchsatz von Netperf aufgeführt. Die Messreihen sind fünf mal mit einer Dauer von jeweils 20 Sekunden durchgeführt worden.

| Switch | Durchsatz |
|---|---|
| Consumer 1 | 27,88 |
| Consumer 2 | 20,60 |
| Consumer 3 | 1,01 |
| Manageable 1 | 2,63 |
| Manageable 2 | 0,22 |
| Manageable 3 | 47,24 |
| Linux Switch | 0,29 |
| Fairer Switch | 46,97 |

Tabelle 4.1: TCP-Durchsatz in MBit/s mit einem Störsender

Der Sender einer TCP-Verbindung ist im Vergleich zu Sendern anderen Datenverkehrs kooperativ, da der TCP-Algorithmus eine Blockierung des Netzwerks zu vermeiden versucht. Hierzu reduziert er die Datenrate kontinuierlich, bis keine weiteren Pakete mehr verloren gehen. Deutlich wird dies an dem zu erwartenden Verhalten der Linux, Consumer 3, Manageable 1 und Manageable 2 Switches: Die TCP-Verbindung kommt fast vollständig zum Erliegen.

Die faire Switch-Implementierung und der Manageable 3 Switch erreichen dagegen eine faire Aufteilung der zur Verfügung stehenden Datenrate. Wollen zwei Quellen Daten zum gleichen Ziel senden, so ist es fair, jedem die Hälfte der maximalen Datenrate zur Verfügung zu stellen. Beide Switches regulieren den Störsender, indem sie seine Pakete verwerfen, obwohl er weiter mit voller Datenrate sendet. Die kooperative TCP-Verbindung hat so die Möglichkeit, mit gleicher Datenrate zu übertragen. Mittels TCP wird ein Durchsatz von 47 MBit/s erreicht, was der Hälfte des verfügbaren Durchsatzes von 94 MBit/s entspricht.

Die TCP-Durchsätze vom Consumer 1 und Consumer 2 Switch sind zwischen diesen beiden Grenzwerten anzuordnen. Das Verhalten dieser Switches wird später im Kapitel, zusammen mit den Messungen für den UDP-Durchsatz, erklärt. Anhand der in dieser Messreihe ermittelten Werte kann noch keine Aussage getroffen werden.

### 4.5.3 Latenzmessungen

In Request-Response-Szenarien ist eine niedrige Latenz sinnvoll. Noch wichtiger ist aber, dass eine maximale Latenz in Verbindung mit einer garantierten Datenrate eingehalten wird.

| Switch | UDP | | TCP | |
|---|---|---|---|---|
| | ohne | mit | ohne | mit |
| Consumer 1 | 1519 | 1107 | 1541 | 1092 |
| Consumer 2 | 1525 | 551 | 1542 | 546 |
| Consumer 3 | 1508 | 77,9 | 1559 | 77,8 |
| Manageable 1 | 1508 | 46,2 | 1588 | 23,7 |
| Manageable 2 | 1478 | 66,8 | 1552 | 42,3 |
| Manageable 3 | 1545 | 61,7 | 1541 | 61,6 |
| Linux Switch | 1869 | 8,7 | 1793 | 7,8 |
| Fairer Switch | 1879 | 65,0 | 1769 | 64,9 |

Tabelle 4.2: Request-Response-Transaktionen pro Sekunde mit und ohne Störsender

Netperf ist in der Lage, Request-Response-Messungen sowohl für UDP-Nachrichten als auch für TCP-Verbindungen durchzuführen. Die Nutzlast der Nachrichtenpakete für die Messungen beträgt hierfür standardmäßig ein Byte. Netperf zählt die Anzahl an Transaktionen pro Sekunde, wobei eine Transaktion der Kombination aus einem Request (Anfrage) und der dazugehörigen Response (Antwort) entspricht. Die Messergebnisse für UDP und TCP, sowohl mit als auch ohne Störsender, sind in Tabelle 4.2 aufgelistet.

Ohne Störsender werden zwischen 1478 und 1879 Transaktionen von den verschiedenen Switches übertragen. Die Dauer einer Transaktion variiert zwischen 0,53 ms und 0,67 ms. Auffällig ist hier, dass die PC-Hardware die Pakete schneller durchleitet.

Mit einem Störsender nimmt die Anzahl der Transaktionen deutlich ab. Das deutet darauf hin, dass die Switches eine Output-Queue besitzen.

Die PC-Hardware für die Switch-Implementierung verwendet Netzwerkadapter mit einem Ringpuffer, der 128 Pakete für die Übertragung zwischenspeichern kann. Da dieser meist auf der Netzwerkkarte selbst angesiedelt ist, kann der Ringpuffer nicht direkt vom Scheduler beeinflusst werden. Dies ist der Grund, warum die faire Switch-Implementierung nur 65 Request-Response-Transaktionen pro Sekunde erreicht (vgl. Tabelle 4.2). Eine Transaktion benötigt etwa 15,4 ms.

In Fast-Ethernet-Netzwerken können $\frac{100{,}000{,}000}{1530*8} \approx 8170$ Ethernet Frames pro Sekunde übertragen werden. In 15,4 ms sind das etwa 126 Pakete, das entspricht ungefähr der Größe des Ringpuffers von 128 Paketen. Switches mit mehreren Output-Queues pro Port, wie dies bei CoS-fähigen Switches und der fairen Switch-Implementierung der Fall ist, sollten deswegen keinen weiteren Puffer hinter dem Multiplexer verwenden. Nur so lassen sich Ethernet Frames mit einer möglichst geringen Latenz verschicken.

### 4.5.4 UDP-Durchsatzmessungen

In medizinischen Systemen werden häufig periodisch Daten von Echtzeit-Messkurven versendet. Für diese Art der Daten bietet sich das UDP-Protokoll an. Insbesondere, wenn mehrere Empfänger per Multicast oder Broadcast gleichzeitig erreicht werden sollen, ist UDP die optimale Lösung. Deswegen ist es wichtig, im Netzwerk einen zuverlässigen UDP-Durchsatz zu gewährleisten.

Der UDP-Durchsatz lässt sich sowohl mit Netperf als auch mit Iperf messen. Bei Iperf muss die zu sendende Datenrate angegeben werden, während Netperf grundsätzlich mit der maximalen Datenrate sendet. Das Verhalten von Netperf spiegelt nicht die Übertragungsart von Echtzeit-Messkurven wieder, da üblicherweise nicht permanent mit voller Datenrate gesendet wird. Für Messungen mit unterschiedlicher Datenrate kommt daher nur Iperf infrage.

Die zu übertragende Datenmenge kann die maximale Nutzlast eines Ethernet Frames überschreiten. Dies hat die Aufspaltung des Datenpakets in mehrere Fragmente zur Folge. Daher werden im Rahmen dieser Arbeit zusätzlich Messungen mit unterschiedlicher Nutzdatengröße durchgeführt. Diese zeigen die Auswirkungen von gebündelt ankommenden Ethernet Frames auf die Übertragung im Switch. Auch für diese Messungen wird auf Iperf zurückgegriffen, da damit zusätzlich zur Datenrate die Nutzdatengröße variiert werden kann.

#### Messungen mit unterschiedlicher Datenrate

Die Messungen bei unterschiedlicher Datenrate werden mit einer UDP-Nutzlast von 1470 Byte durchgeführt, sodass keine IP-Fragmentierung auftritt. Daraus folgt, dass die gemessene UDP-Verlustrate der Ethernet-Frame-Verlustrate entspricht. Die dazugehörigen Messergebnisse sind in Tabelle 4.3 aufgelistet.

Die faire Switch-Implementierung verhält sich wie erwartet. Es werden alle Pakete ohne Verlust übertragen, solange mit einer Datenrate gesendet wird, die unter der Hälfte der insgesamt verfügbaren Datenrate bleibt, da die Datenrate zwischen Störsender und Nutzdatenstrom aufgeteilt wird. Ab einer Datenrate von 50 MBit/s beginnt der faire Switch, Daten zu verwerfen, da die effektive Datenrate etwas weniger als 100 MBit/s beträgt.

Der Manageable 3 Switch zeigt ein ähnliches Verhalten, jedoch erhält der Störsender weniger als die halbe Datenrate. Dies ist daran zu erkennen, dass der Nutzdatenstrom mit 50 MBit/s übertragen wird, obwohl die tatsächliche Durchsatzrate für UDP bei 95-96 MBit/s liegt. Eine mögliche Erklärung ist, dass der Router eine Historie über zuvor übertragene Daten speichert und den Nutzdatenstrom bevorzugt überträgt, weil dieser mit einer geringeren Datenrate sendet. Damit schon bei Messbeginn Daten

| MBit/s | Cons 1 | Cons 2 | Cons 3 | Man 1 | Man 2 | Man 3 | Linux | Fair |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0,05 % | 0 | 54,8 % | 0 | 2,6 % | 0 |
| 10 | 0 | 0 | 0,01 % | 0,3 % | 31,7 % | 0 | 15,0 % | 0 |
| 20 | 0 | 4,3 % | 0,23 % | 1,1 % | 36,2 % | 0 | 24,2 % | 0 |
| 25 | 0 | 8,3 % | 0,13 % | 1,2 % | 38,3 % | 0 | 29,1 % | 0 |
| 30 | 0 | 15,2 % | 0,08 % | 2,3 % | 40,0 % | 0 | 31,7 % | 0 |
| 40 | 0,001 % | 22,7 % | 0,24 % | 3,8 % | 41,9 % | 0 | 37,9 % | 0 |
| 45 | 0,05 % | 26,3 % | 0,45 % | 4,7 % | 43,8 % | 0 | 39,9 % | 0 |
| 50 | 6,1 % | 30,8 % | 15,03 % | 5,6 % | 44,8 % | 0,02 % | 42,9 % | 4,0 % |

Tabelle 4.3: UDP-Paketverlustrate im Referenzdatenstrom bei unterschiedlichen Datenraten

vom Störsender versendet werden, ist es notwendig, diesen bereits vor Messbeginn zu starten.

Der geringfügige Paketverlust des Consumer 1 Switches bei 40 und 45 MBit/s wird aus einer *Random Early Detection (RED)* resultieren, welche zum aktiven Queue-Management verwendet wird [44]. RED dient dazu, einzelne Pakete bereits vor dem Überlaufen einer Queue zu verwerfen, um so dafür zu sorgen, dass nicht alle TCP-Verbindungen gleichzeitig die Geschwindigkeit reduzieren. Das dadurch entstehende nichtdeterministische Verhalten des Switches ist kontraproduktiv für Übertragungsgarantien.

Der Manageable 2 Switch arbeitet unvorhersehbar, insbesondere die extrem hohe Verlustrate bei 1 MBit/s überrascht. Switches mit einem solchen Scheduling-Verhalten sind offensichtlich völlig ungeeignet für Bereiche, in denen Übertragungsgarantien benötigt werden.

#### Messungen mit unterschiedlicher Nutzdatengröße

In dieser Messreihe werden Pakete mit unterschiedlicher Nutzdatengröße versendet, wobei die Datenrate bei durchschnittlich 1 MBit/s bleibt. Mit zunehmender Größe werden die Pakete folglich seltener verschickt, da sonst die Datenrate steigen würde. Tabelle 4.4 beinhaltet die Verlustrate für unterschiedlich große UDP-Pakete. In der ersten Spalte ist die Nutzlast des UDP-Pakets angegeben. Die zweite Spalte zeigt die daraus resultierende Anzahl an IP-Fragmenten, die direkt nacheinander an den Switch gesendet werden.

Während des Tests sendet der Iperf-Client Daten zum Iperf-Server. Eine Evaluierung ist nur in dem Falle möglich, wo Client und Server ihre Daten miteinander austauschen können. Da jedoch das Netzwerk von einem Störsender überlastet wird, gelingt diese

| Byte | Frag | Con 1 | Con 2 | Con 3 | Man 1 | Man 2 | Man 3 | Linux | Fair |
|---|---|---|---|---|---|---|---|---|---|
| 1470 | 1 | 0 | 0 | 0,1 % | 0 | 54,8 % | 0 | 3,0 % | 0 |
| 2940 | 2 | 0 | 0 | 87,2 % | 0 | 59,6 % | 0 | 5,5 % | 0 |
| 4410 | 3 | 0 | 0 | fehlg. | 0 | 68,8 % | 0 | 15,7 % | 0 |
| 5880 | 4 | 0 | 12,6 % | | 0 | 72,8 % | 0 | 36,1 % | 0 |
| 7350 | 5 | 42,5 % | 0,1 % | | 0 | 77,6 % | 0 | 41,2 % | 0 |
| 8820 | 6 | fehlg. | 17,5 % | | 0 | 80,3 % | 0 | 56,3 % | 0 |
| 10290 | 7 | | 49,5 % | | 0 | 81,8 % | 0 | fehlg. | 0 |
| 11760 | 8 | | 49,6 % | | 0 | 86,4 % | 0 | | 0 |
| 14700 | 10 | | 0,8 % | | 0 | fehlg. | 0 | | 0 |
| 17640 | 12 | | 42,2 % | | 0 | | 0 | | 0 |
| 22050 | 15 | | fehlg. | | 0 | | 0 | | 0 |
| 61740 | 42 | | | | 0 | | 0 | | 0 |
| 63210 | 43 | | | | fehlg. | | 0 | | 0 |
| 65507 | 45 | | | | | | 0 | | 0 |

Tabelle 4.4: UDP-Paketverlustrate im Referenzdatenstrom bei unterschiedlicher Nutzdatengröße

Kommunikation nicht immer. Sind zu viele Pakete verloren gegangen, ist Iperf nicht in der Lage, eine Messung durchzuführen. In der Tabelle ist in diesem Fall „fehlg.“ (fehlgeschlagen) eingetragen.

Aus den Messergebnissen ist ersichtlich, dass bereits ein einzelner Störsender andere Übertragungen beeinflusst. Dies ist selbst bei einer niedrigen Datenrate von 1 MBit/s bei einigen Switches der Fall. Alle Consumer-Switches scheinen recht kurze Queues zu haben, da diese leicht überlaufen.

Der Consumer 2 Switch zeigt zusätzlich noch ein weiteres Phänomen. Er hat eine besonders kleine Verlustrate für Datenpakete mit 5 oder 10 Fragmenten. Der *Longest-Queue-First-(LQF)*-Algorithmus wird die Ursache für dieses Verhalten sein. Die Queue scheint die Länge eines Vielfachen von 5 Fragmenten zu haben, sodass der eingehende Datenstrom die Queue optimal füllt. Das führt zu einer niedrigen Verlustrate im Vergleich zu den anderen Messwerten dieses Switches.

Die managebaren Switches variieren ziemlich stark bei den Ergebnissen der Durchsatzmessungen dieses UDP-Tests. Der Manageable 1 Switch schlägt nur bei sehr großen UDP-Paketen fehl, während der Manageable 2 bei seinem unberechenbaren Verhalten bleibt. Nur der Manageable 3 Switch verhält sich in den Messungen entsprechend der fairen Switch-Implementierung. Er scheint damit das geeignetste Gerät zu sein. Jedoch ist nicht sichergestellt, dass er grundsätzlich eine gleichmäßige Verteilung der

Datenrate erreicht, da der Switch ohne weitere Informationen vom Hersteller nur als „Black Box“ betrachtet werden kann.

## 4.6 Ergebnis

Drahtgebundene Netzwerke basieren heutzutage praktisch immer auf Ethernet. Das ist auch im medizinischen Umfeld nicht anders, wo die Anzahl der Geräte, die an die IT-Infrastruktur angebunden werden, stetig steigt. Bei Ethernet handelt es sich aber um ein Best-Effort-Netzwerk, das folglich keine Übertragungsgarantien liefert.

Inzwischen wird Ethernet bevorzugt als voll geswitchtes Netzwerk mit Vollduplex-Verbindungen eingesetzt, damit keine Kollisionen auf dem Medium entstehen. Dies erhöht den Durchsatz im Netzwerk, verschiebt jedoch die Problematik auf das Queue-Management innerhalb der Switches, welches nun dafür verantwortlich ist, zu entscheiden, in welcher Reihenfolge Pakete weitergeleitet werden.

Im medizinischen Bereich sind Übertragungsgarantien von zentralem Interesse, da eine Fernsteuerung von Geräten aus Gründen der Patientensicherheit zuverlässig funktionieren muss. Hinzu kommt, dass viele Medizingeräte auf Standard-Betriebssystemen basieren. Die Betreiber sind üblicherweise nicht berechtigt, selbstständig sicherheitskritische Updates einzuspielen, sodass diese Geräte dadurch anfälliger für Computerviren und Würmer sind als herkömmliche PCs.

Bei Ethernet ist zur Optimierung der Dienstgüte eine Priorisierung auf Basis von Service-Klassen vorgesehen. Dies funktioniert nur so lange zuverlässig, wie hoch priorisierte Klassen nicht zu viel Datenverkehr abbekommen. Ist ein medizinisches Gerät mit einem Computerwurm befallen und beginnt, andere Rechner anzugreifen, so kann das Netzwerk überlastet werden. Werden die Pakete des Computerwurms zudem mit hoher Priorität versendet, so funktioniert die Priorisierung nicht mehr wie gewünscht. Im Gegenteil, der Wurm wird sogar bevorzugt.

Vorhandene Echtzeit-Ethernet-Lösungen liefern – wie klassisches Ethernet – keine Übertragungsgarantien oder benötigen modifizierte Endgeräte. Eine Verwendung von modifizierten Endgeräten ist jedoch unpraktikabel, wenn es um eine Integration in bestehende Systeme geht. Es bleibt damit nur die Möglichkeit, mithilfe von Switches Übertragungsgarantien zu leisten.

Hierfür wurde in diesem Kapitel eine Switch-Architektur vorgestellt, die die zur Verfügung stehende Datenrate gleichmäßig zwischen den Geräten verteilt. Auf diese Weise wird Fairness zwischen den Geräten im Netzwerk erreicht, anstatt auf eine Service-Klassen-Priorisierung zu setzen, die von den Endgeräten vorgegeben wird.

Der spezielle Switch, der hierfür benötigt wird, muss $n^2$ Queues besitzen, um eine gegenseitige Beeinflussung der angeschlossenen Geräte zu vermeiden. Diese Queues

werden entsprechend des Round-Robin-Verfahrens der Reihe nach bearbeitet, sodass automatisch eine maximale Fairness erreicht wird. Auch EtheReal verwendet einen speziellen Switch, der jedoch eine explizite Reservierung durch die Endgeräte benötigt. Dadurch ist EtheReal auf die Zusammenarbeit der einzelnen Knoten angewiesen. Auch kann keine Übertragung garantiert werden, wenn die verfügbare Datenrate vollständig reserviert ist.

Die in dieser Arbeit vorgestellte Switch-Architektur lässt sich gut in einem Operationssaal zur Vernetzung von medizinischen Geräten einsetzen. Ein entsprechender Switch garantiert jedem angeschlossenen Gerät, dass es mit jedem anderen kommunizieren kann. Bei $n$ Geräten an einem solchen Switch wird jedem Gerät eine Übertragung von $\frac{1}{n}$ der maximalen Datenrate je Empfänger garantiert. Sind zum Beispiel 48 Geräte an einem Fast-Ethernet-Switch angeschlossen, entspricht das einer garantierten Übertragungsrate von etwa 2 MBit/s.

# Kapitel 5

# Entwicklung eines Sicherheitskonzepts mit verteilter Zugriffskontrolle

In der Medizin spielt Sicherheit eine große Rolle, wobei verschiedene Arten von Sicherheit dabei von Bedeutung sind. Dies reicht von Datenschutz, der gewährleistet, dass keine Patientendaten an unbefugte Dritte gelangen, bis hin zur Patientensicherheit, bei der es darauf ankommt, einem Patienten keinen Schaden durch Maßnahmen der Gesundheitsversorgung zuzufügen.

Sicherheitskonzepte aus dem Unternehmensumfeld werden üblicherweise bereits im Bereich der Krankenhausverwaltung verwendet, etwa um Abrechnungsdaten in einem KIS zu schützen. Bei der Gerätekommunikation dagegen steht die Patientensicherheit im Vordergrund. Das notwendige Risikomanagement der Hersteller sieht meist in sich abgeschlossene Systeme für die Gerätekommunikation vor. Wenn Kommunikation nach außen erlaubt ist, findet sie über Gateway-Rechner statt und ermöglicht nur unkritische Aktionen, die das Risiko einen Patienten zu schädigen in keiner Weise erhöhen.

Die Betreiber medizinischer Geräte wünschen sich zunehmend auch herstellerübergreifend mehr Interoperabilität. Ein wichtiges Problem hierbei stellt das Risikomanagement dar, welches ein Hersteller in diesem Fall nicht mehr für das gesamte medizinische System vorab durchführen kann. Mithilfe der sich im Entwurf befindlichen IEC 80001 [74] soll es Betreibern ermöglicht werden, ein Risikomanagement durchzuführen, um die Geräteintegration in einem Netzwerk zu ermöglichen. Dieses Risikomanagement baut auf den Restrisiken auf, welche die Hersteller zuvor mit der ISO 14971 bestimmt haben.

Aus der Geräteintegration und der daraus resultierenden Vernetzung entstehen neue Risiken für die Patientensicherheit. Es stellt sich die Frage, welches Gerät welche Funktionen anderer Geräte nutzen darf. Bei dem in der Einleitung beschriebenen Anwendungsfall sollen ein Röntgengerät und ein Anästhesiegerät miteinander kommunizieren und dadurch die Patientensicherheit erhöhen (vgl. Abschnitt 1.3). Es spielt eine Rolle, ob das Röntgengerät dazu berechtigt ist, die Beatmung kurzzeitig unterbrechen zu lassen, um eine qualitativ bessere Aufnahme anzufertigen.

Es besteht folglich der Bedarf an einem Sicherheitskonzept, das die Zugriffskontrolle der Geräte untereinander im Netzwerk regelt. Dies ist nicht nur für das Risikomanagement des Betreibers, sondern auch für den Hersteller von Interesse. Sollte trotz aller Vorsichtmaßnahmen ein Patient zu Schaden kommen, so entstehen möglicherweise Schadensersatzansprüche. Mithilfe des Sicherheitskonzepts soll ein Schaden eindeutig dem verursachenden Gerät zugeordnet werden können.

In diesem Kapitel werden zunächst die Grundlagen der Informationssicherheit eingeführt. Dazu gehören die Sicherheitsmechanismen einer Zugriffskontrolle, mit denen sich die notwendige Sicherheit gewährleisten lässt. Anschließend werden relevante Technologien vorgestellt, die für die Zugriffskontrolle bei Web Services von Bedeutung sind. Verwandte Arbeiten, die sich mit Informationssicherheit im medizinischen Bereich beschäftigen, werden aufgeführt.

Im Anschluss daran wird ein neues Sicherheitskonzept einer verteilten Zugriffskontrolle vorgestellt, das sich durch seinen dezentralen Ansatz gut für verteilte Systeme eignet. Ein Anwendungsfall wird exemplarisch auf zwei verschiedene Arten umgesetzt. In der anschließenden Bewertung folgt ein Vergleich, wie sich die verwendeten Technologien zur Umsetzung eignen. Das Kapitel schließt mit einer Zusammenfassung der Ergebnisse ab.

Die in diesem Kapitel beschriebenen Ideen und Konzepte hat der Autor auch in einem Zeitschriftenartikel [139] zusammengefasst, der in Kürze veröffentlicht werden wird.

## 5.1 Grundlagen

In diesem Abschnitt werden die Grundlagen des Sicherheitskonzepts beschrieben. Im ersten Teil geht es um Konzepte der Informationssicherheit, zu denen unter anderem die Vertraulichkeit von Informationen gehört. Der zweite Teil stellt die verschiedenen Sicherheitsmechanismen vor, mit denen sich die grundlegenden Konzepte umsetzen lassen.

### 5.1.1 Informationssicherheit

Die Informationssicherheit basiert auf sogenannten Schutzzielen. Die drei zentralen Ziele sind Vertraulichkeit (*Confidentiality*), Integrität (*Integrity*) und Verfügbarkeit (*Availability*). Sie sind als das sogenannte *CIA Triad* bekannt [157].

Vertraulichkeit hat zum Ziel, die Daten selbst zu schützen. Hierbei ist nicht Schutz vor Beschädigung gemeint, sondern der Schutz vor Kenntnisnahme der „geheimen“ Daten durch Unberechtigte. Missbrauch der Daten soll auf diese Weise vermieden werden. Zu den zu schützenden Daten gehören unter anderem neben persönlichen Daten auch Konfigurationsdateien, die einen Angriff möglicherweise erleichtern.

Integrität garantiert die Richtigkeit und Vollständigkeit von Daten. Daten, bei denen die Integrität gewährleistet ist, sind vor Veränderung, beispielsweise durch Manipulation, sowie vor einem Datenverlust geschützt. In einer Datenbank ist es notwendig, dass die Einträge nicht nur richtig, sondern auch vollständig vorhanden sind, d. h., dass kein Datenverlust aufgetreten ist.

Verfügbarkeit bzw. Ausfallsicherheit ist das dritte Schutzziel der Informationssicherheit. Ziel ist es, die Funktionalität eines sicherheitskritischen Systems zu gewährleisten. Dazu müssen Systemausfälle vermieden und Angriffe auf das System, die seine Verfügbarkeit einschränken, abgewehrt werden. Die Verfügbarkeit eines Netzwerks zur Kommunikation zwischen medizinischen Geräten war bereits Thema des vorherigen Kapitels.

Abgesehen von den drei Schutzzielen Vertraulichkeit, Integrität und Verfügbarkeit, die als Basis in der Informationssicherheit gelten, gibt es zusätzliche Ziele. Darunter gewinnt Zurechenbarkeit (*Accountability*) zunehmend an Bedeutung. Insbesondere bei elektronischen Verträgen hat sie die Aufgabe, die Verwertbarkeit einer Information vor Gericht zu garantieren. Zurechenbarkeit setzt sich aus Authentizität und Nachweisbarkeit zusammen.

Unter Authentizität (*Authenticity*) ist die eindeutige Zuordnung einer Identität zu einer oder mehreren Informationsquellen zu verstehen, sodass die Echtheit der Information gewährleistet ist. Bei der Nachweisbarkeit (*Non-Repudiation*) geht es darum, den Versand oder den Empfang einer Information belegen zu können. Diese sind erst in Verbindung mit Authentizität des Senders bzw. Empfängers rechtlich verbindlich.

### 5.1.2 Sicherheitsmechanismen

Bei der Zugriffskontrolle muss zwischen Authentifizierung und Autorisierung unterschieden werden.

Mit der Authentifizierung kann die Authentizität eines Benutzers oder Systems nachgewiesen werden. Dieser Identitätsnachweis kann zum einen auf Basis gemeinsamen Wissens eines Passworts stattfinden. In einer zentralen Benutzerdatenbank sind Benutzername und Passwort der zu authentifizierenden Benutzer hinterlegt. Diese Art der Authentifizierung wird typischerweise bei zentralistischen Systemen verwendet.

Zum anderen ist es möglich, dass sich Benutzer oder Geräte durch den Besitz eines kryptografischen Schlüssels ausweisen. Dieser Schlüssel kann etwa in einem digitalen Zertifikat angegeben oder auf einer elektronischen Karte gespeichert sein. Zertifikate werden im Internet für die Webserver-Authentifizierung verwendet. Der Webserver einer Bank kann auf diese Weise seine Authentizität nachweisen, damit der Anwender sicherstellen kann, dass seine Anmeldedaten nicht in falsche Hände geraten.

Autorisierung baut auf der Authentifizierung auf. Sie legt anhand der nachgewiesenen Identität fest, welche Funktionen genutzt werden dürfen. Für die Autorisierung wird in Betriebssystemen oft auf Benutzergruppen und -rollen zurückgegriffen. Entsprechend der Zugehörigkeit zu diesen werden einzelne Funktionen freigegeben.

Autorisierung bildet somit die Basis für das Sicherheitsmerkmal Vertraulichkeit. Nur berechtigten, d. h. autorisierten Benutzern, wird der Zugriff gestattet. Erfolgt dieser Zugriff über ein unsicheres Medium, so muss nicht nur der Zugriff, sondern auch die Übertragung geschützt werden. Hierfür stehen zwei Möglichkeiten zur Verfügung: Zum einen ist es möglich, einen sicheren Kanal zwischen den beteiligten Kommunikationspartnern aufzubauen, über den die zu schützenden Daten übertragen werden. Zum anderen können die Nachrichten selbst geschützt werden, bevor sie übertragen werden. Im Internet werden für webbasiertes Online-Banking geschützte Kanäle auf Basis von HTTPS [151] verwendet, in denen die Daten anschließend im Klartext übertragen werden. Sicherheitskritische E-Mails werden dagegen typischerweise selbst verschlüsselt und anschließend ohne weitere Sicherheitsmaßnahmen übertragen.

Für die Zurechenbarkeit einer übertragenen Nachricht wird neben der Authentizität die Nachweisbarkeit benötigt. In einem gesicherten Kanal wird durch Authentifizierung beim Verbindungsaufbau Authentizität erreicht. Ein Problem des sicheren Kanals ist die Nachweisbarkeit: Es kann im Nachhinein nicht ohne Weiteres bewiesen werden, dass eine Nachricht durch einen bestimmten sicheren Kanal übertragen wurde. Eine Protokollierung der Nachricht ist nicht ausreichend, da nicht nachgewiesen werden kann, dass das Protokoll unverfälscht ist. Für die Nachweisbarkeit einer Nachricht ist daher eine digitale Signatur nötig.

Eine digitale Signatur enthält alle Informationen, um die Integrität und die Authentizität einer Nachricht zu überprüfen. Folglich reicht eine Protokollierung einer Nachricht mit der dazugehörigen digitalen Signatur, um die Zurechenbarkeit zu gewährleisten.

## 5.2 Relevante Technologien

Die in diesem Kapitel vorgestellte Zugriffskontrolle basiert auf existierenden Technologien. Gerade im Bereich der Sicherheit ist es üblich, existierende Standards zu verwenden, die in einem neuen Sicherheitskonzept miteinander vereint werden. Der X.509-Standard für Zertifikate hat sich durchgesetzt und ist ein elementarer Baustein für verschiedene Sicherheitskonzepte. Ein weiterer Baustein sind sogenannte Sicherheitsinformationen, die sich beispielsweise mithilfe von SAML beschreiben und austauschen lassen. Im Rahmen von Web Services gilt WS-Security als Kernkomponente. Es spezifiziert, wie Sicherheitsinformationen in SOAP-Nachrichten eingebettet werden.

### 5.2.1 X.509-Zertifikate

X.509 [19] spezifiziert eine Public-Key-Infrastruktur (PKI). In einer PKI wird Kommunikation mithilfe asymmetrischer Kryptografie abgesichert. Hierbei besitzt jeder Teilnehmer ein Schlüsselpaar aus einem privaten und einem öffentlichen Schlüssel.

Bei der Kommunikation müssen Teilnehmer darauf vertrauen, dass der zu einem öffentlichen Schlüssel passende private Schlüssel dem richtigen Kommunikationspartner gehört. Dieses Vertrauen wird durch die Verwendung von Public-Key-Zertifikaten erreicht, in denen öffentlich verfügbare Informationen gespeichert werden. Damit sich die Informationen in einem Zertifikat nicht verändert lassen, muss jedes einzelne Zertifikat digital signiert werden.

Der X.509-Standard sieht hierfür ein streng hierarchisches System mit vertrauenswürdigen Zertifizierungsstellen vor. An oberster Stelle steht dabei das Zertifikat der Root-Zertifizierungsstelle. Dieses wird meist dazu verwendet, einige Zwischenzertifikate für weitere Zertifizierungen zu erstellen. Der Grund hierfür ist, dass das Root-Zertifikat einen besonders hohen Schutz benötigt, da alle Teilnehmer diesem Zertifikat vertrauen.

Bevor ein Teilnehmer einem Zertifikat trauen kann, muss er sich von dessen Gültigkeit überzeugen. Dazu überprüft er zunächst die Signatur des Zertifikats und kontrolliert anschließend, ob das Zertifikat, das für die Signierung verwendet wurde, ebenfalls gültig ist. Dieser Prozess wird so lange wiederholt, bis ein Zertifikat als Root-Zertifikat einer vertrauenswürdigen Zertifizierungsstelle erkannt wird. In diesem Fall kann allen Zertifikaten in diesem Zertifizierungspfad getraut werden, auch dem ursprünglich zu überprüfenden Zertifikat.

In Abbildung 5.1 ist ein X.509-Zertifikat schematisch dargestellt. Es beinhaltet Angaben über den Aussteller, den Gültigkeitszeitraum sowie den Inhaber (das Subjekt) des Zertifikats. Die im Standard angegebenen technischen Details, wie z. B. Angaben über die verwendeten kryptografischen Algorithmen, sind nicht dargestellt. Seit 1996 liegt der Standard in Version 3 vor und unterstützt zusätzlich zu den existierenden Feldern Erweiterungen. Das Feld ist in der Abbildung weiß hinterlegt, da hier – im Gegensatz zu den anderen Feldern – nicht standardisierte Daten abgelegt werden können.

Bei einer X.509v3-Zertifikatserweiterung handelt es sich um ein Containerformat, in dem beliebige öffentliche Informationen abgelegt werden können. Zur Identifizierung besitzt eine Erweiterung einen *Object Identifier (OID)* [129]. Zusätzlich gibt eine Markierung an, ob die Erweiterung als kritisch oder unkritisch anzusehen ist. Als kritisch markierte Erweiterungen müssen von Anwendungen unterstützt und verarbeitet werden, im Gegensatz zu unkritischen, die ignoriert werden dürfen.

Die standardisierte Erweiterung *Basic Constraints* mit der OID 2.5.29.19 gibt an, ob es sich um ein Zertifikat einer Zertifizierungsstelle handelt. Ist dies der Fall, so


Abbildung 5.1: Schematische Darstellung eines X.509-Zertifikats

wird eine maximale Pfadlänge angegeben, die festlegt, wie viele Zwischenzertifikate erlaubt sind. Die Erweiterung wird ausgehend vom Root-Zertifikat dazu verwendet, um einzuschränken, mit welchen Zertifikaten neue Zertifikate erstellt werden dürfen. Andernfalls wäre es möglich, mit jedem Zertifikat neue Zertifikate zu signieren.

Eine weitere standardisierte Erweiterung ist die *Extended Key Usage* mit der OID 2.5.29.37. Hierbei handelt es sich um eine Erweiterung, die eine Liste mit OIDs enthält, welche jeweils für einen bestimmten Verwendungszweck stehen, für den das Zertifikat benutzt werden darf. Befindet sich beispielsweise die OID 1.3.6.1.5.5.7.3.1 in dieser Liste, so kann das Zertifikat für eine TLS-basierte Webserver-Authentifizierung verwendet werden, wie dies im Internet für HTTPS-Verbindungen notwendig ist.

### 5.2.2 SAML

Die *Security Assertion Markup Language 2.0 (SAML 2.0)* [104] wurde von der OASIS standardisiert. Sie definiert eine Sprache für den Austausch von Sicherheitsinformationen (*Security Assertions*) zwischen verschiedenen Systemen. Bei diesen handelt es sich um beliebige Authentifizierungs- und Autorisierungsinformationen, welche sich in einem standardisierten XML-Format übertragen lassen. Die mit SAML verfolgten Ziele [101] sind unter anderem webbasiertes *Single Sign-on* und *Federated Identity*, sowie die Unterstützung von Web Services.

Um Single Sign-on bei webbasierten Anwendungen zu ermöglichen, müssen Authentifizierungsinformationen zwischen Systemen übertragen werden. Hierfür werden im

Internet typischerweise HTTP-Cookies verwendet. Um Single Sign-on über Domaingrenzen hinweg zu ermöglichen, ist es notwendig, die Sicherheitsinformationen anders zu übertragen, da Cookies auf eine Domain begrenzt sind. Hier setzt SAML mit seinen Security Assertions an, um eine standardisierte Lösung zu bieten.

Nachdem sich ein Anwender für Single Sign-on auf einem zentralen System angemeldet hat, werden alle Hyperlinks auf andere Seiten so präpariert, dass der Webbrowser des Anwenders eine signierte SAML Assertion mit den notwendigen Sicherheitsinformationen an das neue System überträgt. Auf diese Weise wird der Anwender automatisch von seinem Webbrowser auf dem neuen System angemeldet. Hierzu ist es lediglich notwendig, dass das neue System den übertragenen Sicherheitsinformationen vertraut.

Greift ein Anwender direkt auf ein System zu, für das er sich noch nicht authentifiziert hat, so wird er von diesem zu einer zentralen Anmeldemaske weitergeleitet. Nach einer erfolgreichen Authentifizierung wird er auf die ursprüngliche Seite zurückgeleitet. Eine signierte SAML Assertion mit den notwendigen Sicherheitsinformationen ist in der Weiterleitung enthalten.

Komplizierter ist der Ablauf für eine *Federated Identity*. Hier muss der Anwender einmalig seine Identitäten unterschiedlicher Systeme miteinander verbinden. Beim Log-in auf einem System wird festgestellt, dass der Anwender bereits bei einem anderen System angemeldet ist. Ihm wird nun angeboten, die Identitäten miteinander zu verbinden, sodass es in Zukunft ausreicht, sich auf dem ersten der beiden Systeme anzumelden.

Gehören die Systeme zu unterschiedlichen Unternehmen, so können zum Schutz der Privatsphäre Pseudonyme und kurzlebige IDs verwendet werden. Keines der Unternehmen muss dabei Kundendaten preisgeben. Es hat aber die Möglichkeit, beliebige Sicherheitsinformationen in der SAML Assertion abzulegen.

Eine schematische Darstellung einer SAML Assertion befindet sich in Abbildung 5.2. Die im Standard spezifizierten Bereiche sind grau hinterlegt. Beliebige Sicherheitsinformationen lassen sich als Attribut in die SAML Assertion integrieren. Dabei handelt es sich um ein generisches Erweiterungsfeld, in welchem beliebige XML-Daten gespeichert werden können. Hier kann beispielsweise angegeben werden, dass es sich um einen „Premium“-Kunden handelt.

Die übrigen Felder einer SAML Assertion sind mit den Feldern eines X.509-Zertifikats vergleichbar. Auch hier wird ein Subjekt (Inhaber), ein Aussteller (*Issuer*) und ein Gültigkeitszeitraum angegeben. Damit eine SAML Assertion nicht verändert werden kann bzw. eine Änderung bemerkt wird, muss sie vom Aussteller signiert werden. Dies ist immer dann notwendig, wenn die SAML Assertion nicht direkt zwischen Aussteller und Empfänger auf sicherem Weg ausgetauscht werden kann, wie dies bei webbasiertem Single Sign-on der Fall ist. Dort werden die SAML Assertions immer zuerst an den Webbrowser gegeben, damit dieser sie auf der nächsten Seite weiterverwendet.

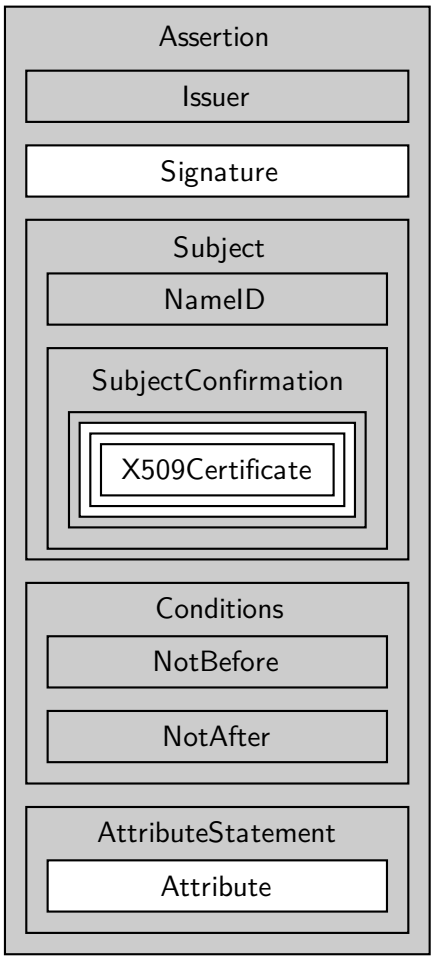



Abbildung 5.2: Schematische Darstellung einer SAML Assertion

Ein weiterer zu berücksichtigender Sicherheitsaspekt ist die Bindung der Assertion an das Subjekt. Die Angabe einer *NameID* ist nur ausreichend, wenn diese auf anderem Wege überprüft werden kann. Oft handelt es sich jedoch um einen nicht weiter verifizierbaren Wert, sodass es sinnvoll ist, die Zugehörigkeit durch eine Angabe in der Assertion nachweisbar zu machen. In der schematischen Darstellung in Abbildung 5.2 erfolgt durch den Besitz eines Zertifikats der Nachweis der Zugehörigkeit der SAML Assertion. Dies wird durch den Besitz des privaten Schlüssels erreicht, der zu dem im Zertifikat angegebenen öffentlichen Schlüssel passt.

Wie SAML Assertions übertragen werden, ist in der Kernspezifikation nicht angegeben. Sie spezifiziert nur das Format der Assertion sowie ein abstraktes Nachrichten-Protokoll. Eine Bindung des Nachrichten-Protokolls an verschiedene Transportprotokolle erfolgt in der SAML-Binding-Spezifikation [105]. In dieser Spezifikation sind verschiedene

HTTP-Bindings für SAML spezifiziert. Diese eignen sich für eine Umsetzung von webbasiertem Single Sign-on, bei dem SAML Assertions mithilfe des Webbrowsers übertragen werden. Sollen die SAML Assertions dagegen direkt zwischen Systemen ausgetauscht werden, so wird dafür ein Transportprotokoll benötigt. Für diesen Zweck ist in der SAML-Binding-Spezifikation das *SAML SOAP Binding* spezifiziert. Es wird angegeben, wie Systeme untereinander SAML-Nachrichten mithilfe von SOAP austauschen können. Dabei sind die einzelnen SAML-Nachrichten im Body einer SOAP-Nachricht platziert.

Unabhängig von der SAML-Binding-Spezifikation hat OASIS ein *Web Services Security: SAML Token Profile 1.1* [110] spezifiziert. In diesem wird angegeben, wie sich SAML Assertions als Sicherheitstoken in WS-Security verwenden lassen. Die SAML Assertion dient in diesem Fall zur Authentifizierung eines der beteiligten Kommunikationspartners. Das *SAML Token Profile* darf nicht mit dem zuvor erklärten *SAML SOAP Binding* verwechselt werden, bei dem Web Services als Hilfmittel für den Transport verwendet werden.

### 5.2.3 WS-Security

Die OASIS-Spezifikation *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)* [111] wird für Sicherheit auf Nachrichtenebene verwendet. Sie ist die *Core*-Spezifikation von WS-Security und definiert, wie Nachrichtenintegrität und Vertraulichkeit in SOAP-Nachrichten umgesetzt werden soll. Die Spezifikation bildet jedoch ausschließlich ein Grundgerüst, das für verschiedene Sicherheitstokens, Vertrauensmodelle etc. verwendet werden kann.

In separaten Sicherheitsprofilen wird spezifiziert, wie verschiedene Sicherheitstokens entsprechend der Core-Spezifikation verwendet werden können. Beispiele sind das bereits erwähnte *SAML Token Profile* oder das *X.509 Certificate Profile* [112]. Die Spezifikation erlaubt es, Sicherheitstokens in einer Nachricht mitzuschicken, um so Nachrichtenintegrität und Vertraulichkeit zu ermöglichen.

Hierfür spezifiziert die Core-Spezifikation einen *Security*-Header-Eintrag. Dieser ist in Abbildung 5.3 in einer schematischen Darstellung mit einer eingebetteten SAML Assertion abgebildet. Die im Core-Standard spezifizierten Felder sind dabei grau hinterlegt. Die dargestellte SAML Assertion steht für eine Authentifizierung zur Verfügung.

Mit dem *SecurityTokenReference*-Element ist es möglich, im Security-Header beliebige XML-Sicherheitstokens zu referenzieren. XML-Tokens sind in den jeweiligen Profilen spezifiziert, von denen sie benötigt werden. Im Core sind dagegen die BinarySecurityTokens spezifiziert. Sie werden für X.509-Zertifikate sowie weitere,

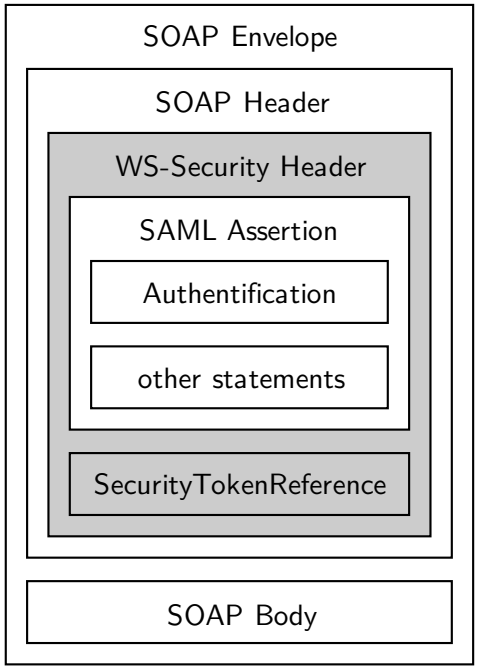



Abbildung 5.3: Schematische Darstellung einer SOAP-Nachricht mit WS-Security und SAML nach [101]

nicht-XML-basierte Tokens verwendet und sind üblicherweise Base64-codiert. Außerdem gibt es *EncryptedData*-Tokens, die verschlüsselte Tokens entsprechend *XML Encryption* [172] enthalten.

Abgesehen von den unterschiedlichen Tokens sind im Security-Header noch *Signature*-Elemente entsprechend *XML Signature* [184] vorgesehen. Für die Verschlüsselung von Daten stehen *EncryptedKey* und *ReferenceList* aus der *XML-Encryption*-Spezifikation [172] zur Verfügung. Im EncryptedKey-Element kann im Header bereits der Schlüssel angegeben werden. Die ReferenceList ist als Subelement eines EncryptedKey-Elementes gedacht, kann bei WS-Security jedoch auch direkt eingesetzt werden, um unterschiedliche Schlüssel für die verschlüsselten Datenbereiche zu verwenden.

Das *EncryptedHeader*-Element ist neben dem Security-Header der zweite Header-Block, der im Core-Standard spezifiziert wird. Er wird benötigt, wenn ein ganzer SOAP-Header-Block verschlüsselt werden muss, damit seine Attribute nicht sichtbar sind. Das EncryptedHeader-Element enthält in einem EncryptedData-Element den Header-Block in verschlüsselter Form. Diese Schachtelung ist notwendig, um das mustUnderstand-Attribut der SOAP-Spezifikation zu berücksichtigen (vgl. Abschnitt 2.2.1).

## 5.3 Verwandte Arbeiten

In der Medizin gibt es verschiedene Bereiche, die sich mit dem Thema Informationssicherheit beschäftigen. Den weitaus größten Teil nimmt dabei der institutionsübergreifende Nachrichtenaustausch ein.

Der Verband Deutscher Arztinformationssystemhersteller und Provider e. V. (VDAP) hat ein Kommunikationskonzept mit dem Namen *VDAP Communication Standard (VCS)* [166] entwickelt, das seit 2001 als technische Regel der DIN vorliegt [30]. Dabei handelt es sich um ein geschütztes Intranet, das von zertifizierten VCS-Providern betrieben wird. In diesem Netz werden E-Mails entsprechend des S/MIME-Standards verschickt. Ein VCS-spezifisches Kommunikationsprotokoll setzt darauf auf, um den Versand und Empfang von Nachrichten zu quittieren. Die Verbreitung von VCS zur Sektor-übergreifenden Kommunikation stellt ein Problem dar, weil viele Software-Hersteller von Arztinformationssystemen VCS nicht unterstützen, sodass alternative Lösungen zur Anbindung gesucht werden [132, 50].

Die Deutsche Röntgengesellschaft (DRG) hat 2003 eine „Arbeitsgemeinschaft für Informationstechnologie“ (@GIT) gegründet, die sich mit dem sicheren Austausch von DICOM-Daten beschäftigt. Das Ergebnis ist ein E-Mail-basiertes System, das OpenPGP [12] verwendet, um so die Sicherheitsanforderungen, die an die Übertragung gestellt werden, zu erfüllen [92, 37]. Vorgestellt wurde es 2004 auf dem 85. Deutschen Röntgenkongress in Verbindung mit einer Kryptokampagne für die Erstellung der notwendigen PGP-Schlüssel.

*Doctor to Doctor (D2D)* [20] ist die Telematik-Plattform der Kassenärztlichen Vereinigung (KV). Dabei handelt es sich um eine Kommunikationsplattform für Praxis- und Klinik-Software, um einen sicheren Austausch von KV-Abrechnungen, Arztbriefen, Überweisungen und weiteren Gesundheitsdaten zu ermöglichen. Die Kommunikation erfolgt über ISDN oder ein *Virtual Private Network (VPN)* der KVen. Die Nachrichten werden verschlüsselt mithilfe TCP/IP übertragen und werden auch auf zentralen D2D-Servern verschlüsselt gespeichert.

An kryptografischen Karten sind in Deutschland im medizinischen Bereich derzeit die elektronische Gesundheitskarte (eGK) und der Heilberufsausweis (HBA) in Entwicklung. Der HBA ist mit der *Health Professional Card (HPC)* verwandt, die in der europäischen Vornorm CEN ENV 13729 spezifiziert ist und sich in der Überarbeitung zu einem internationalen ISO-Standard befindet [131]. Die zu ihrer Nutzung erforderliche Public-Key-Infrastruktur ist in der ISO-17090-Reihe [65] standardisiert.

Informationssicherheit im Bereich der Geräte-Kommunikation ist die Ausnahme. Im Bereich der bildgebenden und bildverarbeitenden Geräte, bei denen durch DICOM momentan am meisten Interoperabilität gewährleistet wird, gibt es jedoch erste Ansätze.

Im DICOM-Standard selbst sind zwar keine Sicherheitsrichtlinien enthalten, jedoch werden in Teil 15 [26] sogenannte Sicherheitsprofile definiert. Beispielsweise wird im Anhang B.1 das *Basic TLS Secure Transport Connection Profile* spezifiziert. Ein DICOM-Gerät, das zu diesem Profil konform ist, muss RSA-basierte Zertifikate, wie sie bei X.509 üblich sind, akzeptieren sowie die kryptografischen Algorithmen SHA, Triple DES EDE und CBC unterstützen. Außerdem wird dringend empfohlen, Port 2762 für TLS-Verbindungen zu verwenden.

In dem Profil wird nicht spezifiziert, wie eine TLS-Verbindung aufzubauen ist. Auch wird nicht angegeben, welche Bedeutung einem ausgetauschten Zertifikat zukommt. Das Profil beschränkt sich damit auf ein Minimum an Interoperabilität.

Im Gegensatz dazu berücksichtigen Forschungsarbeiten im Bereich der Geräteintegration Sicherheitsanforderungen, die in Netzwerken im Rahmen der IEC 80001 zu beachten sind. Das Service-orientierte Integrationskonzept von Ibach et al. [55] sieht für die Gerätekommunikationen in orthopädischen Operationssälen einen zentralen Service-Manager vor. Dieser nimmt eine vermittelnde Rolle ein und ist nicht nur für ein Informationsmanagement, sondern auch für eine Zugangskontrolle zuständig. Aus dieser Arbeit wird momentan eine Anwendungsregel mit dem Arbeitstitel „implementation of IT-networks: use case operating theatres“ erstellt, die der IEC 80001 als Empfehlung angefügt werden soll. Der Autor konnte hier bereits einige Teile seines Konzepts einbringen, sodass die Zugangskontrolle in der Anwendungsregel nicht mehr notwendigerweise zentral erfolgen muss.

## 5.4 Exemplarische Umsetzung des Sicherheitskonzepts

Ein Sicherheitskonzept für die Vernetzung von medizinischen Geräten muss zwischen verschiedenen Geräten unterscheiden, damit nicht jedes Gerät beliebig die Services anderer Geräte verwenden kann. Es wird folglich ein Zugriffsschutz benötigt. Klassischerweise wird hierfür ein zentraler Verzeichnisdienst verwendet, in dem die Berechtigungen der einzelnen Geräte abgelegt sind. Sobald neue Services dem Netzwerk hinzugefügt werden, müssen diese im Verzeichnisdienst eingetragen werden.

Möchte ein Gerät auf den Service eines anderen Gerätes zugreifen, so benötigt es eine Zugriffserlaubnis. Hierfür ist es notwendig, auf dem zentralen Verzeichnisdienst die Berechtigung zu überprüfen. Es findet eine vermittelte Authentifizierung statt, die gleichzeitig eine Autorisierung enthält. Bei einem solchen Dienst handelt es sich um einen sogenannten *Security Token Service (STS)*.

Im Web-Service-Protokollstapel existiert hierfür die WS-Trust-Spezifikation [116]. Diese setzt auf WS-Security auf und spezifiziert, wie Sicherheitstokens bei einem STS beantragt und von ihm erhalten werden können.

Im Gegensatz dazu soll das hier vorgestellte Sicherheitskonzept dezentral arbeiten. Hierdurch ergibt sich eine bessere Skalierbarkeit und Ausfallsicherheit der Zugriffskontrolle, und es wird auf technischer Ebene PnP ermöglicht. Die Idee dabei ist, dass einem Gerät die relevanten Sicherheitsinformationen bereits mitgeliefert werden. Somit entfällt ein Abfragen der Sicherheitsinformationen von einem STS.

Eine exemplarische Umsetzung erfolgt anhand eines Anwendungsfalls, in dem der Krankenhausbetreiber Geräte unterschiedlicher Hersteller in einem Netzwerk verwendet. Die integrierte Oberfläche eines Anästhesiegerätes bietet die Möglichkeit, Infusionspumpen anderer Hersteller zu steuern. Durch ein ganzheitliches Bedienkonzept soll die Arbeit des Anästhesisten erleichtert werden.

Das Risikomanagement der Gerätehersteller sieht vor, dass ausschließlich die Kommunikation erlaubt wird, die im Rahmen des Netzwerk-Risikomanagements abgedeckt ist. Das Anästhesiegerät muss sich darauf verlassen können, dass die Infusionspumpe wie erwartet funktioniert. Außerdem benötigt das Anästhesiegerät eine zuverlässige Angabe darüber, wie präzise die Medikamentendosierung in der Pumpe angegeben werden kann. Der Hersteller der Infusionspumpe dagegen will sichergehen, dass das Gerät von dem Anästhesiegerät gesteuert werden darf.

In diesem Anwendungsfall müssen den Geräten verschiedene Informationen zur Verfügung stehen, um den Zugriff kontrollieren zu können. Da Geräte Services entsprechend des SOA-Konzepts anbieten und nutzen, ist es sinnvoll anzugeben, für welche Service-Typen ein Gerät zugelassen ist, bzw. wofür es dem Risikomanagement entsprechend verwendet werden darf.

Bei Web Services werden diese Service-Typen in Form einer Service-Beschreibung mithilfe von WSDL angegeben. Der Service-Typ eines Gerätes wird dabei in abstrakter Form im sogenannten WSDL-portType angegeben. Der portType hat einen *qualified name (QName)* und lässt sich daher gut referenzieren, um den Typ eines Service anzugeben.

Zusätzlich zum Service-Typ ist auch zu berücksichtigen, welche Rolle – entsprechend dem SOA-Rollenmodell – das Gerät für diesen Service-Typ einnimmt: Service-Anbieter oder Service-Nutzer. Service-Anbieter überprüfen, ob mit einer passenden Berechtigung auf ihre Service-Schnittstellen zugegriffen wird. Service-Nutzer dagegen, die einen anderen Service verwenden möchten, achten darauf, dass sich der Service-Anbieter auch korrekt ausweisen kann.

Im Falle der Infusionspumpe ist zusätzlich zur Angabe des Service-Typs und der Rolle noch die Angabe der Dosierungsgenauigkeit wichtig. Folglich ist es notwendig, zusätzlich noch beliebige weitere Service-spezifische Informationen zu ermöglichen, sodass sich beispielsweise unterschiedliche Güteklassen für einen Service realisieren lassen.

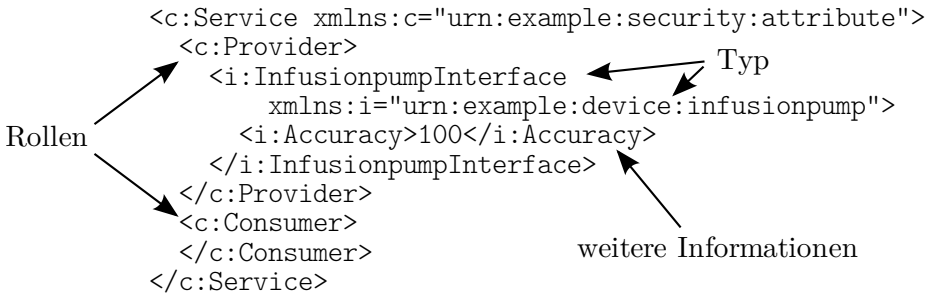



Abbildung 5.4: Beispiel einer Datenstruktur mit Autorisierungsinformationen für die Zugriffskontrolle einer Infusionspumpe

Für eine Zugriffskontrolle lassen sich zusammenfassend folgende Eigenschaften eines Gerätes festhalten: Service-Typ (QName eines WSDL portType), Rolle (Service-Anbieter oder Service-Nutzer) und weitere Service-spezifische Informationen (Güteklassen, ...).

In Abbildung 5.4 ist eine Beispiel-Datenstruktur mit den entsprechenden Informationen für eine Infusionspumpe dargestellt. In dem Grundgerüst werden die Services für das Gerät entsprechend ihrer Rolle im `Provider` oder `Consumer`-Bereich abgelegt. In diesem Fall bietet die Infusionspumpe einen Service an, nutzt aber keinen, sodass nur im oberen Bereich ein Service aufgelistet ist. Hierbei handelt es sich um einen generischen *InfusionpumpInterface*-Service, der mit seinem dazugehörigen Namensraum angegeben ist. Außerdem ist ein Service-spezifischer Wert angegeben, um den beteiligten Geräten weitere Informationen für den Zugriff zu liefern. In diesem Fall ist es die Genauigkeit, mit der die Pumpe Infusionen verabreichen kann.

Die Sicherheitsinformationen der Zugriffskontrolle müssen entsprechend des Risikomanagements ermittelt werden. Anschließend müssen sie für eine dezentrale Zugriffskontrolle von den Geräten selbst bereitgestellt werden.

Die verteilte Zugriffskontrolle lässt sich auf Basis von Zertifikatserweiterungen von X.509-Zertifikaten umsetzen. In Abbildung 5.5 ist ein solches Zertifikat in textueller Darstellung abgebildet. Die notwendigen Sicherheitsinformationen für ein Gerät (vgl. Abbildung 5.4) sind in einem selbst-spezifizierten Erweiterungsfeld abgelegt. Authentifizierung sowie Autorisierung erfolgen dezentral auf Basis von Geräte-Zertifikaten. Bei der Ausstellung eines Zertifikats besteht die Möglichkeit, dem jeweiligen Gerät alle notwendigen Informationen für die Zugriffskontrolle mitzugeben.

Alternativ zu einer Zertifikatserweiterung nach X.509 lässt sich SAML verwenden, um die nötigen Sicherheitsinformationen bereitzustellen. Die Authentifizierung findet dabei weiterhin auf Basis von X.509-Zertifikaten statt, nur dass diese hier keine besonderen Erweiterungen benötigen.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=DE, CN=CAName
        Validity
            Not Before: Oct  1 00:00:00 2009 GMT
            Not After : Oct  1 00:00:00 2012 GMT
        Subject: C=DE, CN=www.example.com/emailAddress=
                b2ee5a30-3d98-4fcd-8103-a9d45c67e430@example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:95:1e:12:e7:b7:db:a2:bd:c3:7e:cc:a9:0e:27:...
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            2.25.173516653110309551815905509064444331291:
                <c:Service xmlns:c="...">...</c:Service>
    Signature Algorithm: sha1WithRSAEncryption
    Signature Value:
        21:3d:33:d9:21:9c:9d:40:8f:5d:8c:fe:ff:26:46:e7:12:97:...
```

Autorisierungsinformationen unter einer selbst spezifizierten OID abgelegt

Abbildung 5.5: Textuelle Darstellung eines X.509-Zertifikats mit Erweiterung für Autorisierungsinformationen (vgl. Abbildung 5.4)

Eine textuelle Darstellung einer entsprechenden SAML Assertion ist in Abbildung 5.6 zu sehen. Die Informationen für ein Gerät sind hier innerhalb des `AttributeStatement` in einem `Attribute` untergebracht. Bei SAML ist es notwendig, eine Assoziation zwischen SAML Assertion und X.509-Zertifikat herzustellen. In der abgebildeten SAML Assertion wird dies durch die Angabe in Zeile 8 erreicht, die die Assertion auf den Inhaber des in Zeile 12 angegebenen X.509-Zertifikats einschränkt. Um zusätzlich noch eine Modifikation an der Assertion selbst auszuschließen, wurde sie vom Aussteller digital signiert (Zeile 3).

## 5.5 Bewertung

Mit den in den Zertifikaten bzw. SAML Assertions zur Verfügung stehenden Informationen kann die Web-Service-Kommunikation adäquat abgesichert werden. Bei der Kommunikation zwischen Anästhesiegerät und Infusionspumpe können alle Nachrichten, die die Dosierungsgeschwindigkeit der Pumpe beeinflussen, digital signiert und protokolliert werden. Sollte einem Patienten durch eine falsche Dosierung geschadet werden, so ließe sich der Fehler eindeutig einem Gerät zuordnen.

Mit dem vorgestellten Sicherheitskonzept für eine Zugriffskontrolle von Web Services lassen sich von den CIA-Schutzzielen Vertraulichkeit und Integrität gewährleisten.

```
 1 <Assertion ID="_a75adf55-01e7-40cc-929f-dbd8472ebdfc"
      IssueInstant="2009-10-01T00:00:00Z" Version="2.0"
      xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
 2  <Issuer>http://www.example.com/IdentityProvider</Issuer>
 3  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
 4  <Subject>
 5    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
 6      b2ee5a30-3d98-4fcd-8103-a9d45c67e430
 7    </NameID>
 8    <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
 9      <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
10        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
11          <ds:X509Data>
12            <ds:X509Certificate>MIICyjCCAjO...</ds:X509Certificate>
13          </ds:X509Data>
14        </ds:KeyInfo>
15      </SubjectConfirmationData>
16    </SubjectConfirmation>
17  </Subject>
18  <Conditions NotBefore="2009-10-01T00:00:00Z"
                NotOnOrAfter="2012-10-01T00:00:00Z">
19  </Conditions>
20  <AttributeStatement>
21    <Attribute>
22      <c:Service xmlns:c="...">...</c:Service>
23    </Attribute>
24  </AttributeStatement>
25 </Assertion>
```

Signatur

Bindung an den Inhaber eines Zertifikats

Autorisierungsinformationen

Abbildung 5.6: Textuelle Darstellung einer SAML Assertion

Verfügbarkeit – als drittes Schutzziel – kann nicht garantiert werden, da sie unter anderem von der Verfügbarkeit des Netzwerks abhängt. Erst in Verbindung mit Übertragungsgarantien (vgl. Kapitel 4) lässt sich Verfügbarkeit gewährleisten.

Das vorgestellte Sicherheitskonzept verwendet eine PKI entsprechend des X.509-Standards mit den dazugehörigen Zertifikaten. X.509-Zertifikate haben sich zur Authentifizierung durchgesetzt und werden im Rahmen von Web Services auf Transportebene für sichere HTTPS-Verbindungen verwendet. Auch auf Nachrichtenebene werden sie durch das X.509-Zertifikat-Profil für WS-Security unterstützt. OpenPGP als alternative PKI, wie sie beispielsweise die Arbeitsgemeinschaft für Informationstechnologie der DRG für den sicheren E-Mail-Austausch verwendet, wird weder von HTTPS unterstützt, noch gibt es eine Spezifikation für Web Services.

Durch die Verwendung von Zertifikaten ermöglicht das Sicherheitskonzept eine digitale Signatur von Nachrichten. In Verbindung mit individuellen Zertifikaten für jedes Gerät lässt sich so eine Nachricht eindeutig einem Gerät zuordnen. Das entspricht auch den Anforderungen des HL7-Web-Service-Profils, welches für jedes Gerät ein eigenes Zertifikat vorsieht. Wichtig ist dies insbesondere für die Zurechenbarkeit, die durch eine Protokollierung der signierten Nachrichten im Sicherheitskonzept auf einfache Weise erreicht wird.

Das vorgestellte Sicherheitskonzept verwendet einen dezentralen Ansatz für die Zugriffskontrolle. Es unterscheidet sich damit grundsätzlich von anderen Ansätzen, die eine zentrale Instanz für das Zugriffsmanagement vorsehen [55]. Durch seine Dezentralität erhöht das Sicherheitskonzept die Ausfallsicherheit, weil es keine Verbindung zu einem zentralen STS benötigt. Außerdem skaliert es dadurch besser, da nicht regelmäßig Sicherheitstokens vom STS abgefragt werden müssen. Als weiteren Aspekt bietet es auf technischer Ebene PnP. Somit ist auch eine direkte Punkt-zu-Punkt-Verbindung zwischen Geräten möglich und es kann auf eine Anbindung an das organisationsweite Netzwerk verzichtet werden. Beispielsweise ließen sich im skizzierten Anwendungsfall die Infusionspumpen direkt mit einem Ethernet-Kabel am Anästhesiegerät anschließen.

Für die dezentrale Umsetzung eignen sich sowohl SAML Assertions als auch Zertifikatserweiterungen nach X.509v3. Beide Ansätze verwenden einen Container, der beliebig mit Informationen angereichert werden kann. Folglich muss in beiden Fällen eine eigene Datenstruktur spezifiziert werden, in der die notwendigen Sicherheitsinformationen, wie Service-Typ, Rolle und weitere Restriktionen, für die Autorisierung abgelegt werden können. Im Falle von SAML ist dies notwendigerweise XML, während sich im Zertifikat beliebige Daten speichern lassen. Ein mögliches Grundgerüst hierfür ist exemplarisch realisiert worden.

X-509v3-Zertifikatserweiterungen haben durch ihren schlichten Ansatz gegenüber SAML einen wesentlichen Vorteil. Es muss keine weitere Spezifikation umgesetzt werden, da Zertifikatserweiterungen bereits im X.509v3-Standard enthalten sind. Es ist lediglich bei der Zertifikatserstellung anzugeben, welche Zugriffsberechtigungen für das Gerät gelten. Diese haben zwangsläufig die gleiche Gültigkeitsdauer, wie das Zertifikat selbst. Sollte sich etwas an den Zugriffsbeschränkungen ändern, so wird ein neues Zertifikat benötigt.

Die höhere Komplexität von SAML geht mit seiner größeren Funktionsvielfalt einher. SAML Assertions sind für einen allgemeinen Austausch von Sicherheitsinformationen konzipiert und unterstützen verschiedene Szenarien, wie Single Sign-on oder Federated Identities.

Unabhängig davon, welcher Ansatz verfolgt wird, ist von entscheidender Bedeutung, wer für die Zugriffskontrolle zuständig ist. Für ein Netzwerk, in dem entsprechend der IEC 80001 Geräte miteinander vernetzt werden, wird das Risikomanagement vom Betreiber durchgeführt. Dieser muss dafür Sorge tragen, dass durch die Vernetzung keine unvertretbaren Risiken für einen Patienten entstehen.

Ibach et al. [55] sehen vor, dass aus diesem Grund der Betreiber mithilfe eines zentralen Service-Managers für die Verwaltung der Zugriffskontrolle zuständig ist. Entsprechend des hier vorgestellten dezentralen Ansatzes wäre folglich der Betreiber dafür zuständig, die Geräte-Zertifikate mit den damit verbundenen Zugriffsinformationen in einer Erweiterung oder einer SAML Assertion auszustellen. Die Geräte-Hersteller

verlassen sich hierbei auf das Risikomanagement des Betreibers. Die Zurechenbarkeit gewinnt in diesem Fall enorm an Bedeutung.

Alternativ dazu wäre es möglich, dass die Geräte-Hersteller die Zertifikate mitliefern, um so PnP zu ermöglichen. In den Geräten ließe sich so der „Intended Use“ eines medizinischen Gerätes angeben, sodass andere Geräte die Service-Schnittstellen so verwenden, wie dies vom Hersteller vorgesehen ist. Hierbei wäre es jedoch notwendig, dass sich die Hersteller auf eine gemeinsame Root-Zertifizierungstelle einigen, da X.509 eine strenge Hierarchie von Zertifizierungsstellen vorsieht. Andernfalls würde ein Wildwuchs an Zertifizierungsstellen entstehen, wie dies im Internet der Fall ist, wo unzählige parallel existieren.

Möchte ein Hersteller zusätzlich bei einigen seiner Geräte bestimmte Funktionen nur exklusiven Kooperationspartnern zur Verfügung stellen, so wird eine zusätzliche Zugriffskontrolle innerhalb des Gerätes benötigt. Eine herstellerübergreifende Zugriffskontrolle reicht hier nicht aus. Der Hersteller muss zusätzlich eigene Regeln im Gerät ablegen, die festlegen, welche anderen Geräte spezielle Funktionen verwenden dürfen.

## 5.6 Ergebnis

Informationstechnische Sicherheitsaspekte werden im medizinischen Umfeld bisher fast ausschließlich auf Enterprise/KIS-Ebene behandelt. Dabei geht es primär um Datenschutzaspekte bezüglich Patientendaten. Zum einen spielt der institutionsübergreifende Austausch von Daten eine Rolle, zum anderen aber auch der Zugriff auf Patientendaten innerhalb eines Betriebs, der eine Authentifizierung und Autorisierung des medizinischen Personals erfordert.

Auf Geräteebene sind bisher überwiegend in sich abgeschlossene Systeme auf dem Markt. Mit zunehmender Öffnung dieser Systeme, die unter anderem durch die IEC-80001-Norm ermöglicht werden soll, entsteht Bedarf an informationstechnischen Sicherheitskonzepten. Bisher liegt der Fokus auf Geräteebene ausschließlich auf der Patientensicherheit. Da die gewünschte herstellerübergreifende Interoperabilität noch nicht vorhanden ist, wurden Sicherheitsaspekte auf Geräteebene bisher nicht genauer betrachtet. Erst durch eine zunehmende Vernetzung werden Sicherheitsaspekte wie Zurechenbarkeit eine immer größere Rolle spielen, sie sollten dabei jedoch von Anfang an berücksichtigt werden.

In diesem Kapitel wurde ein Sicherheitskonzept für eine dezentrale Zugriffskontrolle vorgestellt. Die Grundidee dabei ist, dass einem Gerät alle nötigen Sicherheitsinformationen, die es für den Zugriff auf andere Geräte benötigt, gleich mitgegeben werden. Dabei handelt es sich unter anderem um den Service-Typ und die Rolle, die das Gerät bei der Kommunikation einnimmt. So lässt sich für eine Infusionspumpe angeben, dass

sie einen Infusionspumpen-Service anbieten darf, während ein Anästhesiegerät diesen Service dagegen nutzen darf. Zusätzlich ist es möglich, ergänzende Informationen anzugeben, um so weitere sicherheitskritische Restriktionen zu ermöglichen.

Die Zugriffskontrolle lässt sich auf verschiedene Weisen technisch realisieren. Zum einen ist es möglich, ausschließlich X.509-Zertifikate mit entsprechenden Erweiterungen für die Zugriffskontrolle zu verwenden, um so eine sehr schlanke Umsetzung zu ermöglichen. Es werden keinerlei weitere Spezifikationen benötigt. Zum anderen können SAML Assertions verwendet werden, die üblicherweise für komplexere Zugriffskontrollen eingesetzt werden.

Das Sicherheitskonzept ist insbesondere für eine Zugriffsverwaltung von medizinischen Geräten in einem gemeinsam genutzten Netzwerk geeignet. Dies ist unabhängig davon, ob der Betreiber oder die Hersteller die Zertifikate ausstellen. Sobald ein Gerät mit einem Zertifikat ausgestattet ist, ist eine PnP-Kommunikation möglich. Das vorgestellte Sicherheitskonzept ist nicht nur für Medizingeräte geeignet, sondern lässt sich allgemein in verteilten Systemen verwenden. Der Autor konnte zeigen, dass mit einer dezentralen Zugriffskontrolle ein PnP-Konzept umsetzbar ist.

# Kapitel 6

# Performance von Web Services

Für Web Services wird schlechte Performance häufig als Nachteil aufgeführt, sie haben den Ruf, „langsam“ zu sein [36, 77]. Dennoch werden sie in Forschungsprojekten für die Kommunikation zwischen Medizingeräten eingesetzt [7, 97, 158].

Web Services eignen sich besonders für die Fernsteuerung von Geräten. Auf einfache Weise lassen sich Parameter eines Gerätes abfragen oder setzen. Im Anwendungsfall aus dem vorigen Kapitel ermöglicht die integrierte Oberfläche des Anästhesiegerätes eine Fernsteuerung angeschlossener Infusionspumpen. Beim anderen Anwendungsfall aus der Einleitung dieser Arbeit soll das Röntgengerät eine Synchronisierung der Geräte bzw. eine Unterbrechung der künstlichen Beatmung veranlassen.

Bei Fernsteuerungen handelt es sich üblicherweise um synchronen Nachrichtenaustausch, bei dem Anfragen gestellt und Rückmeldungen erwartet werden. Realisiert wird dies bei Web Services mit dem Request-Response-MEP. Bei der Geräte-Kommunikation müssen jedoch auch Echtzeitkurven und Messwerte von Monitoring-Geräten übertragen werden. Es bietet sich an, anstelle eines regelmäßigen Abfragens neuer Werte eine ereignisgesteuerte Kommunikation zu verwenden. Die Übertragung muss in diesem Fall vom Service-Anbieter aus erfolgen und nicht vom Service-Nutzer, wie dies bei Web Services üblich ist. Für diese Art von Kommunikation bietet sich das Publish-Subscribe-Paradigma an.

Bei der Kommunikation zwischen medizinischen Geräten gibt es also zwei verschiedenartige Übertragungen: Zum einen das klassische Request-Response, wie es von *Remote Procedure Calls (RPCs)* bekannt ist, zum anderen ein ereignisgesteuertes Publish-Subscribe für gemessene Vitalparameter, Alarme und Ähnliches.

Für vergleichende Performance-Messungen wird *Data Distribution Service for Real-time Systems (DDS)* [126] verwendet. DDS ist eine Publish-Subscribe-Middleware, die für Echtzeitsysteme konzipiert wurde. Sie ist insbesondere im militärischen Anwendungsbereich bekannt und wird von der US Navy für ihr *Open Architecture Computing Environment (OACE)* empfohlen [144, 153], sodass DDS auf amerikanischen Kriegsschiffen als Nachrichten-Backbone Verwendung findet. Außerdem bietet

DDS verschiedene Quality-of-Service-Merkmale, sodass es als Konkurrent zu Web Services angesehen werden kann.

In diesem Kapitel soll zum einen untersucht werden, ob die Performance von Web Services für eine Verwendung im medizinischen Bereich ausreicht, bzw. ob und wie sich Web Services verwenden lassen, um zum Beispiel Publish-Subscribe umzusetzen.

Nachfolgend werden relevante Technologien vorgestellt, die im Zusammenhang mit den Performance-Messungen stehen. Anschließend werden verwandte Arbeiten aufgeführt, die sich mit Web-Service- bzw. DDS-Performance beschäftigen. Im dritten Abschnitt werden die zu vergleichenden Szenarien im Detail mit den dazugehörigen Implementierungen vorgestellt. Im Anschluss daran folgt eine Evaluierung der Messwerte. Der letzte Abschnitt fasst die Ergebnisse der Performance-Messungen zusammen.

Die in diesem Kapitel vorgestellten Ideen und deren Ergebnisse wurden zur Veröffentlichung unter [138] eingereicht.

## 6.1 Relevante Technologien

Für die Performance-Messungen werden verschiedene Technologien verwendet: zum einen SOAP-over-UDP als alternatives Transportprotokoll für Web Services und zum anderen DDS als Publish-Subscribe-Middleware für Echtzeitsysteme.

### 6.1.1 SOAP-over-UDP

SOAP-over-UDP wurde von OASIS im Rahmen von DPWS standardisiert [118]. Bei der Spezifikation handelt es sich um ein Transport-Binding, das angibt, wie sich SOAP-Nachrichten in UDP-Datagrammen verschicken lassen.

Einige Anwendungen werden aufgrund der Semantik des Nachrichtenaustausches besser auf UDP basierend entwickelt. Sie sind beispielsweise nicht auf TCP-Übertragungsgarantien angewiesen oder benötigen die Multicast-Funktionalität von UDP. Dies ist z. B. bei WS-Discovery der Fall, welches Multicast-Nachrichten versendet, um Services im lokalen Netz zu finden.

SOAP-over-UDP unterstützt sowohl One-way-Nachrichten als auch das Request-Response-MEP. Nachrichten lassen sich dabei an Unicast- oder Multicast-Adressen verschicken, wobei die Spezifikation für die Response ausschließlich Unicast vorsieht. Dabei ist zu beachten, dass eine SOAP-Nachricht in einem UDP-Datagramm inklusive IP- und UDP-Header eine Größe von 65.536 ($2^{16}$) Byte nicht überschreiten darf. Andernfalls lässt sie sich nicht verschicken bzw. wird abgeschnitten. Eine Aufteilung einer SOAP-Nachricht auf mehrere UDP-Datagramme ist nicht vorgesehen.

Der SOAP-over-UDP-Standard schreibt die Verwendung von WS-Addressing vor. Zum einen ermöglicht dies die Angabe einer SOAP-Action, zum anderen ist es notwendig, um eine Message-ID hinzuzufügen, die bei SOAP-over-UDP grundsätzlich vorhanden sein muss. Die Message-ID findet dabei für zwei Aufgaben Verwendung: Erstens wird sie dazu benötigt, in einer Response-Nachricht die dazugehörige Request-Nachricht im RelatesTo-Feld zu referenzieren. Zweitens ermöglicht die Message-ID das Erkennen einer wiederholt übertragenen Nachricht.

Ein wiederholtes Übertragen von Nachrichten ist vorgesehen, um möglichen Paketverlusten – insbesondere bei Multicast – vorzubeugen. Hierfür ist im Anhang der Spezifikation ein Algorithmus angegeben, mit dem ermittelt werden kann, wie lange bis zur erneuten Übertragung eines Pakets zu warten ist.

Für die Adressierung ist bei SOAP-over-UDP das folgende URI-Schema nach RFC 3986 [10] vorgesehen: `soap.udp://<host>:<port>[/<rel_path>][?<query>]`. Für das Versenden eines UDP-Datagramms werden hingegen ausschließlich Host und Port der URI verwendet.

Im Gegensatz zu HTTP lässt sich bei UDP keine Sicherheit mittels TLS erreichen. Zwar existiert die RFC-Spezifikation *Datagram Transport Layer Security (DTLS)* [152], jedoch ist diese abgesehen von SIP-Internettelefonie nicht verbreitet und spielt im Web-Service-Bereich momentan keine Rolle. DTLS unterstützt ausschließlich UDP-Unicast. Im Web-Service-Bereich wird UDP jedoch meist aufgrund seiner Multicast-Funktionalität verwendet.

Folglich ist es notwendig, WS-Security einzusetzen, um Informationssicherheit zu erreichen. Bei UDP ist dies von großer Relevanz, da sich der Absender eines UDP-Datagramms fälschen lässt. Es ist zu beachten, dass durch WS-Security die SOAP-Nachricht größer wird und trotzdem die maximale Größe eines UDP-Datagramms nicht überschreiten darf. Daher wird in der Spezifikation empfohlen, Sicherheitstokens nicht mitzusenden. Diese müssen im Vorwege bereits bekannt sein oder mithilfe des HTTP-Transport-Bindings ausgetauscht werden.

### 6.1.2 DDS

Die Spezifikation *Data Distribution Service for Real-time Systems (DDS)* wurde von der *Object Management Group (OMG)* standardisiert [126]. Diese ist insbesondere für die Entwicklung der *Common Object Request Broker Architecture (CORBA)* und der *Unified Modeling Language (UML)* bekannt.

Bei DDS handelt es sich um eine Middleware zur Kommunikation von Echtzeit-Applikationen. *Quality of Service (QoS)* ist ein elementarer Bestandteil des Gesamtkonzepts von DDS. DDS garantiert dabei selbst keine Echtzeit, im Gegenteil, es ist

auf Performance ausgelegt und meldet ausschließlich eine QoS-Verletzung. Es ist vorgesehen, dass eine Applikation Angaben darüber macht, „was“ sie an QoS-Merkmalen benötigt und nicht angibt, „wie“ diese erreicht werden sollen. In DDS sind hierfür verschiedene QoS-Regeln spezifiziert, mit denen die Applikation die nötigen Angaben machen kann.

Die Middleware ist für Netzwerke mit tausenden Knoten konzipiert. Daher ist es notwendig, Sender und Empfänger voneinander zu entkoppeln. Das ist nicht nur für die Skalierbarkeit wichtig, sondern auch, um die nötige Flexibilität zu erreichen. Schließlich sollen einzelne Knoten hinzugefügt werden können, ohne das ganze System ändern zu müssen.

Klassischerweise wird für eine Entkopplung von Sendern und Empfängern ein gemeinsam genutzter, verteilter Speicher verwendet, in dem sie ihre Daten ablegen. Im Netzwerk lässt sich dies schwierig mit hoher Performance implementieren, da Schreibzugriffe synchronisiert werden müssen. Als Alternative hat sich in vielen Fällen das Publish-Subscribe-Modell durchgesetzt.

DDS basiert auf einem sogenannten *Data-Centric-Publish-Subscribe-(DCPS)*-Modell, das die Hauptschicht der Middleware darstellt. Hierbei wird das Konzept eines „*Global Data Space*“ implementiert. Applikationen, die dazu beitragen wollen, verbreiten ihre Informationen mithilfe eines sogenannten *Publisher*. Um Teile des Global Data Space zu lesen, bedarf es eines *Subscriber*. Die Middleware übernimmt dabei die Aufgabe, die Informationen an alle interessierten Subscriber zu verteilen.

Dem Global Data Space liegt ein Daten-Modell zugrunde. Dabei handelt es sich um typisierte Datenstrukturen, die aus Sicht der Middleware ohne Bezug zueinander existieren. Im Global Data Space werden sie anhand ihres *Topic* und des dazugehörigen Datentyps identifiziert. Dieser wird mit der *CORBA Interface Definition Language (IDL)* spezifiziert.

Um Daten in einem bestimmten Topic zu verbreiten, verwendet der Publisher einen *DataWriter*. Mit diesem wird es der Applikation ermöglicht, Daten in den Global Data Space zu „schreiben“ (*write*-Operation). Auf der Empfängerseite wird für jedes Topic ein eigener *DataReader* für den Subscriber benötigt. Mit diesem ist es möglich, abonnierte Daten aus dem Global Data Space „herauszunehmen“ (*take*-Operation).

Im DataReader sowie im DataWriter lassen sich QoS-Richtlinien angeben. Die Middleware achtet darauf, dass die notwendigen Bedingungen eingehalten werden. Das bedeutet für Publisher, dass sie Daten nach dem „*Fire-and-forget*“-Prinzip in den Global Data Space schreiben können.

Wenn das *RELIABILITY*-QoS-Merkmal auf *RELIABLE* gesetzt ist, dann wiederholt die Middleware die Datenübertragung, falls es notwendig ist. Dies ist genau dann der Fall, wenn der DataWriter eine zuverlässige Übertragung unterstützt und der DataReader eine solche beansprucht. Andernfalls wird *BEST_EFFORT* übertragen,

etwa dann, wenn der DataWriter zwar eine zuverlässige Übertragung unterstützt, der DataReader jedoch nur *BEST_EFFORT* benötigt.

Weitere QoS-Merkmale sind *HISTORY* und *PARTITION*. Das *HISTORY*-QoS kontrolliert, wieviele „alte“ Datensätze in einem Topic zur Verfügung stehen sollen, falls der DataReader sie nicht schnell genug herausnimmt. Standardmäßig wird nur ein Datensatz aufbewahrt. Es ist aber auch möglich, diesen Wert zu erhöhen oder sogar alle Datensätze von der Middleware aufbewahren zu lassen.

Mit dem *PARTITION*-QoS können mehrere DataReader und DataWriter logisch separiert werden. Dies erfolgt zusätzlich zur Trennung auf Netzwerkebene mit unterschiedlichen *DOMAINS*. DataReader können folglich nur die Daten empfangen, die in ihr Topic geschrieben wurden und sich in ihrer logischen Partition innerhalb ihrer Netzwerk-Domain befinden.

Auf der DCPS-Schicht mit den QoS-Richtlinien kann eine optionale *Data Local Reconstruction Layer (DLRL)* von der Middleware aufgesetzt werden. Sie ist dafür zuständig, aus den Daten, die DataReader empfangen haben, eine lokale Rekonstruktion zu erstellen. Diese kann anschließend von der Applikation so verwendet werden, als ob die Daten lokal vorlägen. Eine Anwendung muss die Daten nicht mehr selbst aus dem Global Data Space einzeln herausnehmen.

DDS ermöglicht mit der DCPS- und der DLRL-Schicht Kompatibilität auf Applikationsebene. Es ist ausschließlich ein API spezifiziert. Ein einheitlicher Datenaustausch über ein Netzwerk ist als separate Erweiterung verfügbar. Diese ist von OMG als *Data Distribution Service Interoperability Wire Protocol (DDSI)* [128] standardisiert.

## 6.2 Verwandte Arbeiten

Im Bereich von Web Services wurden bereits zahlreiche Performance-Messungen durchgeführt. Sie unterscheiden sich anhand ihrer Zielsetzung. Einige Arbeiten vergleichen Web Services mit anderen Technologien, während andere eine Optimierung von Web Services thematisieren.

Elfwing et al. [36] haben 2002 einen Vergleich von Web Services und CORBA durchgeführt. Hierfür haben sie die in Java enthaltene CORBA-Implementierung mit dem Java-Axis-Framework der Apache Foundation verglichen. Nach Beseitigung einiger Performance-Probleme der Web-Service-Implementierung war die Übertragung mit der CORBA-Implementierung immer noch etwa 7-mal schneller als die Web-Service-Realisierung.

Ein Vergleich von Web Services mit Java *Remote Method Invocation (RMI)* wurde von Juric et al. [77] durchgeführt. Zusätzlich zu einer ungeschützten Datenübertragung wurde auf Seite von Java RMI die SSL-gesicherte Variante RMI-SSL verwendet. Auf

Web-Service-Seite wurde eine Nachricht mit WS-Security sowohl signiert als auch verschlüsselt. RMI schneidet im Vergleich zu Web Services deutlich besser ab. Gerade im Bereich der Sicherheit sind Web Services deutlich langsamer, unter anderem da bei ihnen Sicherheit auf Nachrichtenebene und nicht auf Transportebene verwendet wurde, wie dies bei RMI-SSL der Fall ist.

Chiu et al. [17] haben einen speziellen Aspekt von Web-Service-Performance untersucht: die Übertragung von Fließkommazahlen. Da SOAP-Nachrichten üblicherweise als XML serialisiert werden, müssen Fließkommazahlen für den Transport zunächst von der Binärdarstellung in ASCII-Darstellung konvertiert werden und auf Empfängerseite anschließend wieder zurück. Besonders aufwendig ist die Konvertierung für Fließkommazahlen doppelter Genauigkeit aufgrund der notwendigen Rundungsmodi. Die Autoren schlagen deshalb vor, mehrere Kommunikationsprotokolle zu unterstützen. Standard-konformes SOAP lässt sich dazu verwenden, zu ermitteln, welche proprietären Erweiterungen von beiden Kommunikationspartnern unterstützt werden, um so Fließkommazahlen effizienter auszutauschen. Unterstützt der eine Kommunikationspartner die notwendige Erweiterung nicht, dann empfehlen sie, unverändertes SOAP zu verwenden.

Um die Performance von Web Services zu erhöhen, schlagen Kangasharju et al. [79] die Verwendung einer Komprimierung sowie das aufrechterhalten einer TCP-Verbindung für einen weiteren Nachrichtenaustausch vor. Für die Komprimierung verwenden sie GNU zip (gzip) [49] und für das Aufrechterhalten der Verbindung wird ein selbst entworfenes Transport-Binding für TCP eingesetzt.

Von praktisch größerem Belang sind Arbeiten, die kompatibel zu bestehenden Web-Service-Spezifikationen sind. Abu-Ghazaleh und Lewis [1] haben eine differenzielle Deserialisierung entwickelt, die auf Seiten eines Service-Anbieters die Ähnlichkeit eingehender Nachrichten ausnutzt, um so die Zeit für ein Deserialisieren von Nachrichten zu reduzieren. Bei den verarbeiteten Nachrichten handelt es sich um Arrays mit ganzen Zahlen und Fließkommazahlen, die portionsweise verarbeitet werden. Die differenzielle Deserialisierung reduziert die Verarbeitungszeit bei Nachrichtenportionen ab 32 Byte, in den Messreihen, bei denen sich nur 25 % und 50 % der Zahlen im Array geändert haben. Bei einer Änderung von 75 % der Zahlen kann die Verarbeitungszeit durchaus länger sein. Folglich lässt sich die differenzielle Deserialisierung nur in sehr speziellen Fällen nutzen.

Govindaraju et al. [47] haben die Performance von Web-Service-Toolkits miteinander verglichen. Die dabei verwendeten Toolkits waren: gSOAP 2.4, AxisC++ (CVS-Version vom 28. Mai 2004), AxisJava 1.2, .NET 1.1.4322 und XSOAP4/XSUL 1.1. Am schnellsten war gSOAP [162], das auch in der vorliegenden Arbeit als Web-Service-Toolkit für die Performance-Messungen verwendet wird. Für gSOAP wurden von van Engelen und Zhang [163] weitere Performance-Messungen durchgeführt, mit denen unter anderem ermittelt wurde, an welchen Stellen Optimierungen für WS-Security möglich sind.

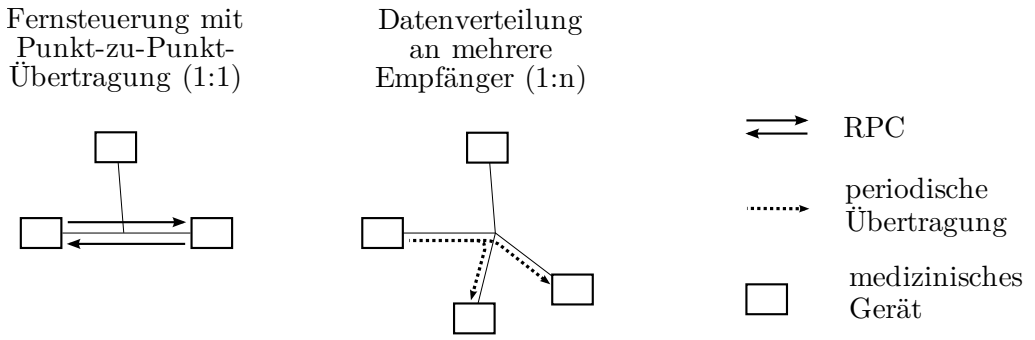



Abbildung 6.1: Szenarien für Performance-Messungen

DDS wurde von Xiong et al. [203] im Rahmen des *DDS Benchmark Project* [34] untersucht. Dabei ging es nicht nur um die Performance der Implementierungen von *Real-Time Innovations (RTI)* [149], *PrismTech* [145] und *Object Computing, Inc. (OCI)* [125], sondern auch um die darunter liegenden Architekturen zur Umsetzung von DDS. Die Performance-Messungen behandeln die Latenz und Schwankungen der Paketlaufzeit (Jitter). Diese wurden sowohl innerhalb eines Rechners als auch über ein Gigabit-Netzwerk gemessen.

In einem DDS-Tutorial von Schmidt und Parson [154] wird nach einer Einführung ein Benchmark-Szenario aufgeführt, um die Performance von DDS zu demonstrieren. Es werden ein nicht näher spezifizierter „Notification Service“, JMS, gSOAP und drei DDS-Implementierungen miteinander verglichen. Round-Trip-Zeiten von verschieden großen Nachrichten werden angegeben, bei denen gSOAP trotz XML-Serialisierung gut abschneidet. Bei kleinen Nachrichten ist nur eine DDS-Implementierung schneller, jedoch steigt mit zunehmender Nachrichtengröße die Latenz von gSOAP. Leider ist nicht ersichtlich, ob UDP oder HTTP für den SOAP-Transport verwendet wurde. Im Ganzen sind die Messwerte nicht aussagekräftig.

## 6.3 Realisierung

Wie in der Motivation in Abschnitt 1.3 aufgeführt, sieht das MD-PnP-Programm [88] zwei Arten von Interoperabilität vor: Zum einen den Austausch von diskreten und zeitkontinuierlichen Parametern und zum anderen die Fernsteuerung von Geräten. Für die Performance-Messungen werden deshalb beide Szenarien entsprechend Abbildung 6.1 umgesetzt. Bei der Fernsteuerung eines Gerätes handelt es sich um eine Punkt-zu-Punkt-Übertragung, bei der das steuernde Gerät eine Rückmeldung erwartet. Dagegen ist es bei der Datenverteilung von Messwerten üblich, mehrere Empfänger zu haben. Deswegen werden in diesem Szenario Messwerte periodisch an beliebig viele Empfänger versendet (1:n-Übertragung).

Die Web-Service-Implementierungen für die Performance-Messungen verwenden *WS4D-gSOAP* [204]. Dabei handelt es sich um eine DPWS-Implementierung, die auf gSOAP [162] aufsetzt, welches für diese Arbeit in Version 2.7.13 verwendet wird. Es wird WS4D-gSOAP eingesetzt, da es bereits die benötigten Web-Service-Spezifikationen umsetzt und sich durch die hohe Performance von gSOAP auszeichnet.

Die Realisierungen auf DDS-Basis verwenden *OpenSplice DDS* [146] von PrismTech in der *Community Edition V4.1.090513*. Es handelt sich dabei um die einzige frei erhältliche DDS-Implementierung der im DDS-Benchmark-Projekt vertretenen Hersteller.

Im Folgenden werden beide Szenarien beschrieben und es wird angegeben, welche Art von Nachrichtenaustausch stattfindet. Außerdem wird erläutert, wie die Umsetzung mit Web Services und DDS erfolgt.

### 6.3.1 Fernsteuerung

Bei der Fernsteuerung von Geräten wird der interne Status eines Gerätes modifiziert. Dabei können nicht nur einzelne Einstellungen, sondern auch der Gesamtzustand eines Gerätes verändert werden. Die Fernsteuerung muss dabei nicht von einem entfernten Raum aus erfolgen. Der häufigere Fall dürfte eine vereinheitlichte, integrierte Bedienoberfläche in unmittelbarer Nähe zum Gerät sein.

Außerdem ist es möglich, dass die Fernsteuerung von einem anderen Gerät durchgeführt wird. Ein Röntgengerät könnte etwa, wie im Anwendungsfall in Abschnitt 1.3 beschrieben, die Beatmung durch ein Anästhesiegerät kurzzeitig unterbrechen lassen und so den Arbeitsablauf automatisieren. Voraussetzung für die Fernsteuerung ist allerdings, dass das steuernde Gerät dazu berechtigt ist.

Bei den zur Fernsteuerung von Geräten übertragenen Nutzdaten handelt es sich oft um einzelne Werte, die Einstellungen ändern oder das Gerät dazu veranlassen, in einen anderen Betriebszustand zu wechseln. Die Übertragung muss bei einer Fernsteuerung zuverlässig erfolgen, d. h., jeder mögliche Kommunikationsfehler muss festgestellt werden. Deswegen ist eine Rückmeldung vom zu steuernden Gerät notwendig.

Im Test-Szenario soll eine Nutzlast von einem 32-Bit-Integer übertragen werden. Er repräsentiert eine Alarmgrenze oder eine andere Einstellung eines medizinischen Gerätes, wie zum Beispiel die Konzentration eines Narkosegases. Es ist nicht notwendig, einen zusätzlichen Bezeichner zu übertragen, der angibt, um was für einen Wert es sich handelt, da sich diese Information aus dem Nachrichten-Typ ableiten lässt: Bei Web Services ist diese Information im Operationsnamen enthalten, während bei DDS der Name des Topic verwendet wird.

Das Szenario sieht vor, dass eine Antwortnachricht geschickt wird, die in diesem Fall ebenfalls ein 32-Bit-Integer enthält. Diese kann einen Status-Code oder den vorherigen

Wert einer Einstellung entsprechend der Semantik beinhalten, die für diese Messungen allerdings keine weitere Rolle spielen.

Für die Web-Service-Realisierung des Fernsteuerungs-Szenarios wurde das übliche HTTP-Transport-Binding verwendet. Dieses kann auf verschiedene Weisen konfiguriert werden. So ermöglicht HTTP die Aktivierung von *Keep-Alive*. Hiermit ist es möglich, eine bestehende TCP-Verbindung aufrecht zu erhalten, um weitere HTTP-Requests zu übertragen. Mit deren Übertragung kann so schneller begonnen werden.

Anstelle von HTTP kann auch HTTPS verwendet werden, welches einen geschützten Transport der Nachricht gewährleistet. Allerdings ermöglicht HTTPS nur Sicherheit auf Transportebene, sodass keine Zurechenbarkeit der übertragenen Nachrichten gegeben ist (vgl. Abschnitt 5.1.2). Hierfür ist es notwendig, die Nachricht selbst entsprechend der WS-Security-Spezifikation zu sichern.

gSOAP unterstützt alle genannten Erweiterungen: Keep-Alive, HTTPS und WS-Security. Von WS-Security wird jedoch nur die XML-Signatur unterstützt. XML-Verschlüsselung ist nicht möglich, sodass Vertraulichkeit ausschließlich auf Transportebene erreicht werden kann. Für die Realisierung der Web-Service-Fernsteuerung können die drei HTTP-Erweiterungen in jeder beliebigen Kombination aktiviert werden, sodass acht verschiedene Messungen möglich sind.

DDS unterstützt von sich aus keine Request-Response-Szenarien, sodass dies emuliert werden muss. Das erste Problem, dass bei der Realisierung mit DDS entsteht, ist die Adressierung. Bei DDS sind Publisher und Subscriber so stark entkoppelt, dass sie sich nicht gegenseitig identifizieren können. Folglich ist es notwendig, jedem Gerät eine ID zu geben. Hierfür werden, wie dies DPWS für Web-Service-Geräte vorsieht, UUIDs vergeben, die einen Teilnehmer identifizieren und so eine Adressierung ermöglichen.

Ein weiterer Aspekt ist die Referenzierung von Nachrichten. Eine Response-Nachricht muss eine Referenz auf eine Request-Nachricht enthalten, sodass auf Applikationsebene die Zuordnung hergestellt werden kann. Auch hierfür werden UUIDs verwendet, wie dies bei Web Services entsprechend der WS-Addressing-Spezifikation vorgesehen ist.

Bei DDS gibt es drei Varianten, eine Adressierung umzusetzen: Die ID des Empfängers kann in den Namen einer logischen Partition oder eines Topic eingebettet sowie in der Nachricht selbst angegeben werden. In der DDS-Realisierung dieses Szenarios wird die Empfänger-ID in der Nachricht angegeben, da dies im Vergleich zu den anderen beiden Varianten die flexibelste Lösung ist. Es ist nicht notwendig, mehrere DataWriter und Publisher zu verwenden, um Nachrichten an andere Empfänger zu adressieren.

Im Gegensatz zur Web-Service-Umsetzung, bei der entsprechend WS-Addressing die IDs im SOAP-Header vorhanden sind, müssen diese in der Nutzlast der DDS-Nachricht untergebracht werden. Die Request-Nachrichten enthalten also zusätzlich zu dem 32-Bit-Integer für die Fernsteuerung eine Empfänger-ID (`toID`), eine Nachrichten-ID (`msgID`) und eine ID, an welche die dazugehörige Antwort geschickt werden soll

(`replyToID`). Die Response-Nachricht enthält die gleichen Angaben, jedoch wird anstelle der ID für eine Rückantwort (`replyToID`) die Nachrichten-ID der Anfrage als Bezugsinformation (`relatesTo`) angegeben.

### 6.3.2 Datenverteilung

Das zweite untersuchte Szenario ist die Datenverteilung. Diese unterscheidet sich von einer Fernsteuerung dadurch, dass sie nicht den internen Zustand eines Gerätes verändert. Es werden ausschließlich anderen Geräten Daten zur Verfügung gestellt.

Beispielsweise können so alle Daten, die auf einem Gerät dargestellt werden, einem anderen System zur Verfügung gestellt werden, um eine entsprechende Remote-Darstellung zu ermöglichen. Dazu gehören insbesondere EKG-Kurven und periodisch erhobene Messwerte, aber auch Geräteeinstellungen, wie Alarmgrenzen und weitere interne Zustandsinformationen. Die meisten generierten Daten stammen vom Patienten-Monitoring. Diese sollen nicht nur lokal auf einem Bildschirm zur Verfügung gestellt werden, sondern auch in anderen Räumen. Schon heute ist es üblich, dass auf Intensiv- und Überwachungsstationen im zentralen Stationszimmer alle Monitoring-Daten angezeigt werden. Die gemessenen Vitalparameter sind nicht nur für eine Anzeige auf weiteren Geräten wichtig, sondern auch für eine Protokollierung in der elektronischen Patientenakte.

Bei der Datenverteilung werden periodisch Daten verschickt. Die erhobenen Messwerte sollen dabei nicht einzeln, sondern in regelmäßigen Abständen gebündelt übertragen werden. Um die Auswirkung der Datenmenge zu ermitteln, soll ein Integer-Array als Nutzlast übertragen werden, welches in der Größe variiert werden kann. So lässt sich der Einfluss der Serialisierung auf die Übertragung ermitteln.

In einigen Fällen liegen auf dem sendenden Gerät bereits alle Daten in einem eigenen Binärformat vor und können direkt übertragen werden. Um auch diesen Fall abzudecken, soll neben einer Übertragung eines Integer-Array auch eine Übertragung mit einem Byte-Array variierender Länge möglich sein. Für Web Services wird das Byte-Array Base64-codiert [76] im XML abgelegt. Bei DDS dagegen werden Byte-Arrays nativ unterstützt. Ein Nachteil dieser Variante ist, dass die Binärdaten nicht vom Framework dekodiert werden können. Der Entwickler einer Anwendung muss Programm-Code schreiben, der die Binärdaten in geeignete Datenstrukturen überführt. Das Framework bietet hierbei keinerlei Unterstützung. Trotzdem soll dies bei den Performance-Messungen berücksichtigt werden.

Die Web-Service-Realisierung verwendet das SOAP-over-UDP-Transport-Binding für dieses Test-Szenario, da durch die Verwendung von Multicast die Daten gleichzeitig von mehreren Empfängern abonniert werden können. Ein mehrfacher Versand der Daten per HTTP ist dadurch nicht notwendig und eine höhere Latenz durch eine zunehmende Anzahl an Abonnenten wird so vermieden. Die Verwendung von Multicast

anstelle von HTTP hat den Nachteil, dass die Pakete nicht zuverlässig übertragen werden. Aufgrund des periodischen Versendens der Daten kann der Empfänger einen Paketverlust jedoch selbst feststellen und die Daten bei Bedarf abfragen. Daher wird generell keine Empfangsbestätigung benötigt.

UDP unterstützt kein Keep-Alive, da keine Verbindung zwischen Sender und Empfänger zustande kommt. Auch UDP-Transportsicherheit spielt bei Web Services keine Rolle. Die Daten werden, wie bei DDS, einfach versendet. Eine Signierung der Daten ist die einzige Konfigurationsmöglichkeit, die von gSOAP für SOAP-over-UDP unterstützt wird. Die Umsetzung erfolgt so, dass der gesamte SOAP-Body signiert und das Zertifikat im SOAP-Header als Sicherheitstoken eingebettet wird.

Bei DDS muss für die Datenverteilung zusätzlich zu den Daten-Arrays eine Absender-ID angegeben werden, damit sich die Daten eindeutig einem Gerät zuordnen lassen. Von den verfügbaren QoS-Merkmalen ist RELIABILITY von Bedeutung, sodass die Implementierung sowohl BEST_EFFORT als auch RELIABLE unterstützt.

## 6.4 Evaluation

Der Versuchsaufbau besteht aus zwei PCs mit AMD-Athlon-XP-2700+-(2.167-GHz)-Prozessoren. Die PCs sind mit einem Crossover-Ethernet-Kabel direkt über ihre On-board-Netzwerkkarten verbunden. Bei diesen handelt es sich um 100 MBit/s Fast-Ethernet-Karten des Typs VT6102 [Rhine-II] (rev 78) von VIA. Jeder PC hat 1 GByte RAM und läuft unter Debian 5.0.1 (Kernel 2.6.26-2-686) im *Single User Mode*. Nach dem Booten wird der udev-Daemon beendet, um eine Beeinträchtigung durch diesen zu unterbinden.

Für die DDS-Messungen muss der OpenSplice-Daemon laufen. Er ist dafür zuständig, der Applikation die Publish-Subscribe-Funktionalität bereitzustellen. Für die Messungen wird er in der Standardeinstellung betrieben, in der er alle Nachrichten per Broadcast verteilt.

Die Verwendung von Broadcast sorgt dafür, dass die Anzahl der Empfänger keinen Einfluss auf die Übertragung des Senders hat. Dies gilt jedoch nur so lange, wie kein Paketverlust auftritt und Pakete nicht wiederholt versendet werden müssen. Unter der Annahme, dass kein Paket verloren geht, ist eine Messung mit zwei PCs auch für eine 1:n-Übertragung repräsentativ. Das Gleiche gilt für die Datenverteilung auf Basis von SOAP-over-UDP, wenn Multicast verwendet wird.

Für die HTTPS-Übertragungen und die digitale Signatur werden X.509-Zertifikate mit 1024-Bit-RSA-Schlüsseln verwendet. Hash-Werte werden mit SHA1 bestimmt.

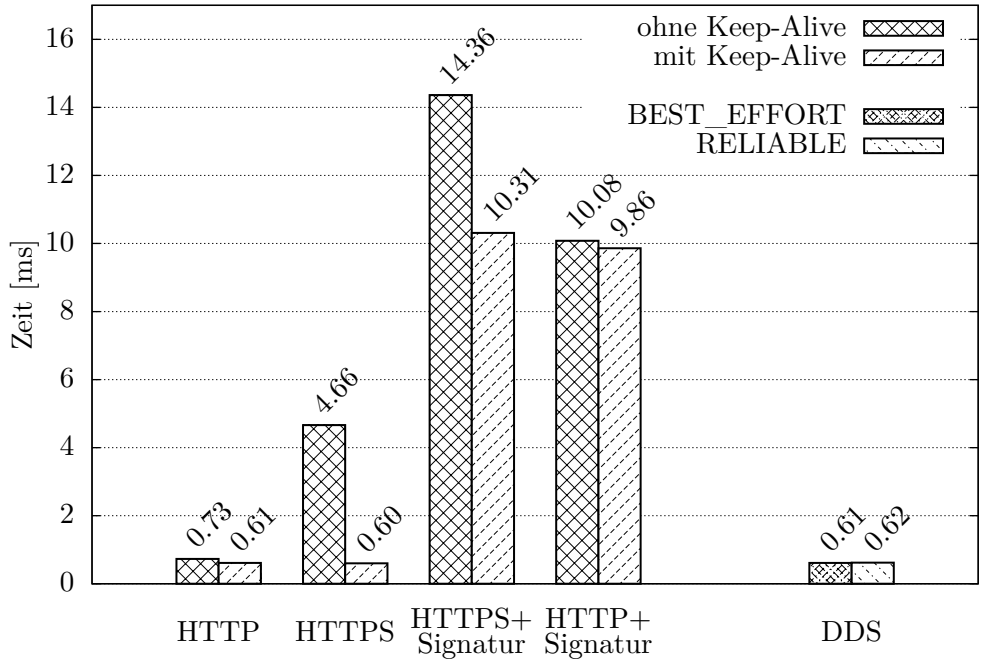



Abbildung 6.2: Round-Trip-Zeit für die Fernsteuerung von Geräten

### 6.4.1 Fernsteuerung

Die Performance-Messungen für die Fernsteuerung sind einfach durchzuführen. Im Client-Programm wird die Laufzeit des entfernten Funktionsaufrufes gemessen. Für jede mögliche Konfiguration werden 11.000 Messungen durchgeführt. Von diesen werden die Ersten 1.000 verworfen, um einen Einfluss der Initialisierung zu vermeiden, der insbesondere bei aktivierten Sicherheitsfunktionen auftritt.

Abbildung 6.2 zeigt den Median der Messwerte für jede einzelne Konfiguration. Auf der linken Seite sind die acht möglichen Web-Service-Konfigurationen abgebildet, während die zwei DDS-Konfigurationen rechts dargestellt sind.

Die Performance von Web Services auf Basis von HTTP ist vergleichbar mit den Ergebnissen des emulierten RPCs unter DDS. Keep-Alive sorgt für eine Performancesteigerung, die sich insbesondere bei HTTPS-Verbindungen bemerkbar macht. Dies ist am zweiten und dritten Säulenpaar deutlich zu sehen. Der Grund hierfür ist der rechenintensive Verbindungsaufbau der Transportsicherheit mit asymmetrischen RSA-Schlüsseln. Dieser muss einmalig am Anfang der Verbindung durchgeführt werden, um einen symmetrischen Sitzungsschlüssel zu erzeugen, der anschließend eine wesentlich schnellere Verschlüsselung ermöglicht.

Die Verwendung von Keep-Alive hat einen weiteren Effekt. Bei aktiviertem Keep-Alive ist die Datenübertragung über eine HTTPS-Verbindung schneller als über eine HTTP-Verbindung. Zunächst erscheint dies widersprüchlich. Es wirkt wie ein Messfehler, da eine zusätzliche Verschlüsselung eine Übertragung nicht beschleunigen kann. Die kürzeste Round-Trip-Zeit, die für HTTPS mit Keep-Alive gemessen worden ist, ist 0,55 ms, während sie für HTTP mit Keep-Alive 0,61 ms beträgt.

Mit einem Netzwerk-Sniffer konnte herausgefunden werden, dass die TLS-Verbindung, welche die Grundlage für HTTPS ist, beim Verbindungsaufbau automatisch eine Deflate-Datenkomprimierung aktiviert. Da sich die darauf folgenden SOAP-Nachrichten nur anhand ihrer Nachrichten-ID unterscheiden, kann die Kompression sehr effizient arbeiten. Die komprimierte SOAP-Nachricht passt in ein einzelnes Ethernet Frame, im Gegensatz zur unkomprimierten Nachricht, die auf zwei Frames verteilt wird.

Die Messungen haben gezeigt, dass die Kompression mit anschließender Verschlüsselung in diesem Versuchsaufbau die Übertragungszeit verkürzt. Mit einem langsameren Prozessor oder einem schnelleren Netzwerk könnte die Round-Trip-Zeit mit Kompression und anschließender Verschlüsselung dagegen größer sein.

### 6.4.2 Datenverteilung

Die Datenverteilung von Messwerten erfolgt unidirektional. Messungen der Paketlaufzeit sind deswegen nicht ohne Weiteres durchzuführen, da eine Zeitsynchronisation immer einer gewissen Ungenauigkeit unterliegt. Erschwerend kommt hinzu, dass die Uhren einen zusätzlichen Drift haben, d. h., dass sich deren Versatz im Laufe der Zeit vergrößert oder verkleinert. Deswegen wird zusätzlich eine gleiche Übertragung in umgekehrter Richtung durchgeführt, sodass sich die Round-Trip-Zeit mit einer Uhr messen lässt. Die Hälfte der gemessenen Zeit sollte eine gute Annäherung für die Übertragungszeit bei der Datenverteilung wiedergeben.

Wie bei den Messungen der Fernsteuerung werden auch hier 11.000 Messungen durchgeführt, von denen die ersten tausend verworfen werden. Anschließend werden die Messwerte halbiert, um die Übertragungszeit für eine Richtung wiederzugeben. Abbildung 6.3 zeigt den Median für Messreihen mit verschieden großer Nutzlast. Ein 32-Bit-Integer wird in der Nutzlast grundsätzlich mit 4 Byte angerechnet, unabhängig davon, wie viel er bei einer XML-Serialisierung benötigt.

Die DDS-Messungen werden für Byte- und Integer-Arrays jeweils mit und ohne RELIABILITY durchgeführt. Da für alle vier Messreihen die Ergebnisse nur im Rahmen der Messgenauigkeit variieren, ist der Übersichtlichkeit halber nur eine Linie für DDS gezeichnet.

Es ist nicht überraschend, dass die DDS-Messergebnisse für Byte-Arrays und Integer-Arrays gleich sind, da eine binäre Serialisierung verwendet wird. Diese benötigt

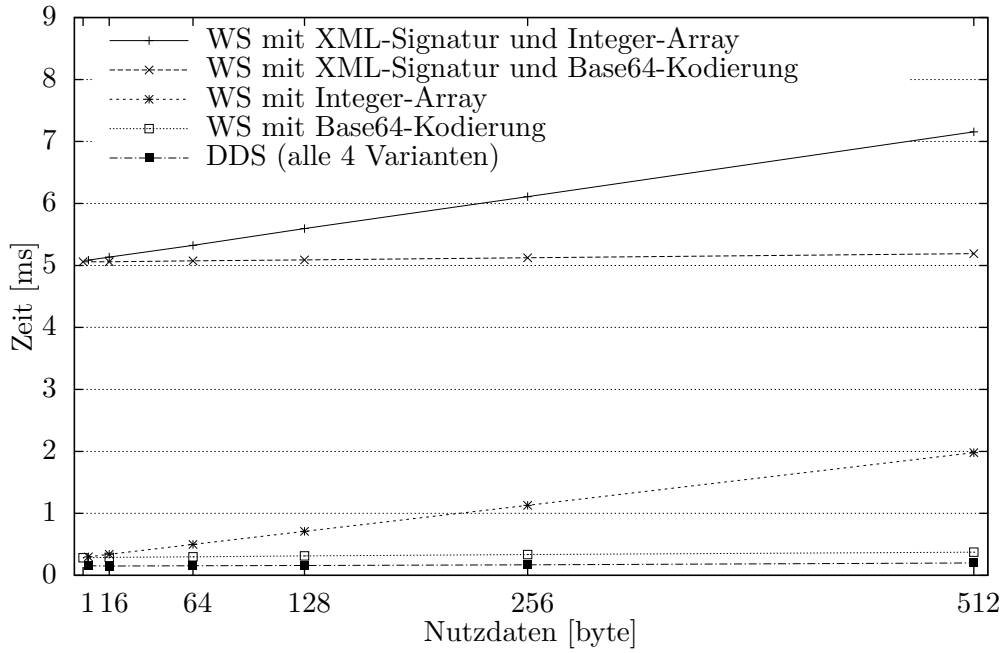



Abbildung 6.3: Übertragungszeiten für die unidirektionale Datenverteilung

für Daten gleicher Größe die gleiche Serialisierungszeit. Die Tatsache, dass sich die Zeiten für eine zuverlässige Übertragung von einer Best-Effort-Übertragung nicht unterscheiden, ist plausibel, da auf der Crossover-Ethernet-Verbindung keine Pakete verloren gegangen sind.

Abbildung 6.3 ist zu entnehmen, dass das Signieren einer SOAP-Nachricht und das dazugehörige Überprüfen der Signatur auf Empfängerseite – unabhängig von der Nutzlast – etwa 5 ms in Anspruch nimmt. Außerdem ist der Overhead durch eine XML-Serialisierung des Integer-Array zu erkennen. Für jeden Integer muss ein öffnendes und schließendes XML-Tag eingefügt werden. Bei der Base64-Codierung des Byte-Array dagegen spielt die Größe der Nutzdaten eine untergeordnete Rolle.

Ein Vergleich der Web-Service-Übertragung Base64-codierter Binärdaten mit den DDS-Übertragungen ist auf Basis des Diagramms schwierig. Für Web Services vergrößert sich die Übertragungszeit von 0,29 ms für 1 Byte auf 0,38 ms für 512 Byte, während sie bei DDS für die entsprechenden Nutzlasten von 0,15 ms auf 0,20 ms steigt. In Verbindung mit der Übertragung ist die binäre DDS-Codierung von OpenSplice doppelt so schnell wie die Base64-Codierung für Web Services von gSOAP. Auch hier muss berücksichtigt werden, dass sich das Verhältnis der Übertragungszeiten bei anderer Prozessorleistung und Netzwerkgeschwindigkeit verschieben kann.

DDS unterstützt keine kryptografischen Sicherheitsmechanismen, sodass kein Vergleich mit Web Services vorgenommen werden kann.

## 6.5 Ergebnis

Die Performance von Web Services wird häufig aufgrund der Verwendung von XML kritisiert. Web Services gelten als „langsam“ im Vergleich zu anderen Middlewares. Dies zeigen auch diverse oben aufgeführte Arbeiten. Allerdings gibt es Implementierungen, die auf Performance ausgelegt sind, wie dies beispielsweise bei gSOAP der Fall ist. Im Vergleich zu anderen Web-Service-Implementierungen ist sie performant.

Um die Performance von Web-Services in Relation zu anderen Middlewares zu ermitteln, sind exemplarische Vergleichsmessungen durchgeführt worden. Hierfür ist DDS, eine Middleware für Datenverteilung in Echtzeitsystemen, verwendet worden, die sich durch ihre Geschwindigkeit und QoS-Unterstützung auszeichnet. Zwei praxisnahe Szenarien sind evaluiert worden: die Fernsteuerung eines Gerätes sowie die periodische Datenverteilung von Echtzeitkurven und anderen gemessenen Werten.

Die Messergebnisse zeigen, dass die Performance von Web Services im Vergleich zu DDS für eine Verwendung im medizinischen Bereich gut genug ist. Das zum Vergleich herangezogene DDS hat teilweise eine „schnellere“ Datenübertragung, was in Anbetracht der Tatsache, dass es für hohe Performance entwickelt ist, nicht überrascht. Die Performance-Unterschiede sind jedoch nicht so groß, dass sie einen Wechsel zu DDS begründen würden. Vor allem, da viele Projekte im Bereich der Vernetzung von medizinischen Geräten bereits Erfahrungen im Umgang mit Web Services haben, ist es sinnvoll, diese Technologie auch weiterhin einzusetzen.

Web Services werden in dieser Arbeit vor allem aufgrund ihrer Interoperabilität verwendet, die bei der Vernetzung von medizinischen Geräten im Mittelpunkt steht. DDS dagegen wurde als Middleware für Echtzeitsysteme mit Fokus auf Performance entwickelt. Die Möglichkeit zur Interoperabilität ist erst später mit der DDSI-Spezifikation hinzugefügt worden.

Bei der Umsetzung des Fernsteuerungs-Szenarios hat sich gezeigt, dass *lose Kopplung* nicht uneingeschränkt sinnvoll ist. Reine Publish-Subscribe-Middlewares – wie DDS – entkoppeln Sender und Empfänger zu stark voneinander, als dass noch eine direkte Adressierung einzelner Empfänger möglich wäre. In diesem Fall ist eine Kopplung auf Anwendungsebene notwendig, um RPC realisieren zu können.

Die Ergebnisse der Datenverteilung zeigen, dass eine XML-Serialisierung für große Integer-Arrays ungeeignet ist. Jeder Integer muss in einem eigenen XML-Tag eingeschlossen und von der Binär- in die ASCII-Darstellung konvertiert werden. So lange keine binäre Serialisierung bzw. Kompression auf SOAP-Ebene vorhanden ist, muss abgewogen werden, ob Arrays mit numerischen Daten nicht Base64-codiert eingebettet werden sollen, um so eine höhere Performance zu erreichen.

Insgesamt ist die Performance von Web Services jedoch ausreichend, um Systeme im technischen Umfeld miteinander zu verbinden. Web Services eignen sich damit

nicht nur für die Kommunikation auf KIS-Ebene, sondern auch für eine Vernetzung von medizinischen Geräten untereinander zur gegenseitigen Fernsteuerung und Datenverteilung.

# Kapitel 7

# Zusammenfassung und Ausblick

Das Ziel dieser Arbeit ist die Entwicklung einer SOA zur vernetzten Kommunikation zwischen medizinischen Geräten, Systemen und Applikationen. Die SOA soll die Kommunikation im gesamten medizinischen Bereich abdecken, um so bestehende Interoperabilitätsprobleme langfristig beseitigen zu können.

Web Services haben sich im Unternehmensumfeld als technologische Basis zur Umsetzung einer SOA durchgesetzt. Sie zeichnen sich insbesondere durch ihre Interoperabilität aus, da sie weder an eine Plattform, noch an eine Programmiersprache gebunden sind. Daher ist die vermehrte Verwendung von Web Services im medizinischen Bereich auf Enterprise-/KIS-Ebene zu erwarten. Im Rahmen der HL7-Standardisierung wird bereits eine Transport-Spezifikation für Web Services entwickelt [53]. Der Bereich der Geräteintegration wird hingegen durch Web Services bisher nicht vollständig abgedeckt. Aus diesem Grund wurden in dieser Arbeit Fragestellungen aus diesem Bereich untersucht, um so eine ganzheitliche SOA-Umsetzung zu ermöglichen.

Das in dieser Arbeit entwickelte Gesamtkonzept einer SOA ist in Abbildung 7.1 dargestellt. Die Ebenen lassen sich in zwei Gruppen einteilen. Die „Plug“-Gruppe beinhaltet alle Ebenen mit vorbereitenden Maßnahmen, die im SOA-Rollenmodell (vgl. Abbildung 1.4) mit dem Binden der Services aneinander abschließen. In der „Play“-Gruppe sind die Ebenen zusammengefasst, welche die Ausführung von Services betreffen.

Die Architektur ist in Auszügen bereits vorab in [135] bzw. [136] sowie in [137] veröffentlicht worden. Sie basiert auf DPWS, welches eine Web-Service-Kommunikation für ressourcenbeschränkte Geräte vorsieht. Das PnP-Konzept von DPWS ist beibehalten worden, während gleichzeitig eine Integration in den Enterprise-/KIS-Bereich ermöglicht wird.

Das SOA-Rollenmodell sieht einen Verzeichnisdienst zum Auffinden von Services vor. Es besteht Bedarf an einer PnP-Discovery-Lösung, die zum einen lokal funktioniert und zum anderen ein klinikweites Auffinden von Web Services ermöglicht. Bestehende Technologien werden den Ansprüchen nur partiell gerecht. Ein umfassendes, funktionelles Konzept existierte bisher nicht.

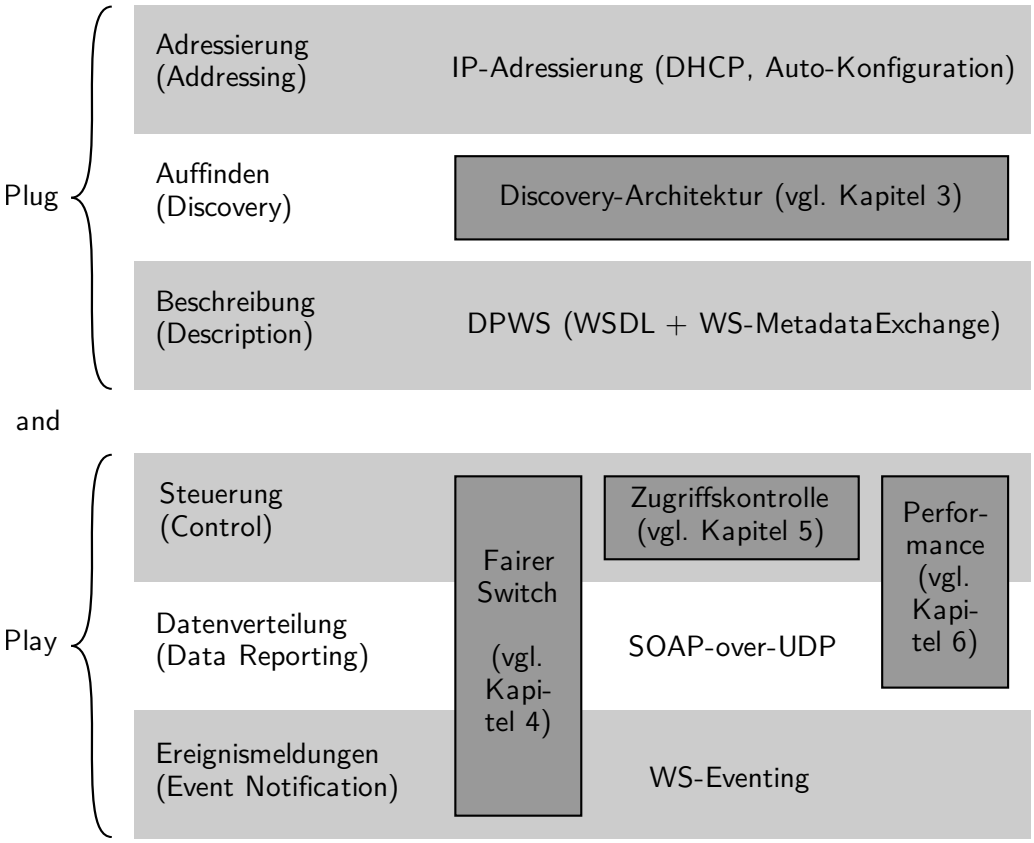



Abbildung 7.1: Entwickelte SOA im Überblick

Die in dieser Arbeit entworfene Discovery-Architektur (vgl. Kapitel 3) ist speziell für die genannten Anforderungen konzipiert worden und unterstützt PnP. Die lokale Suche nach Web Services bzw. Medizingeräten ist durch die Verwendung von Multicast robust, im Gegensatz zu zentralen Komponenten, die ausfallen können. Zusätzlich ermöglicht die Architektur das Auffinden von DPWS-konformen Geräten, wie Netzwerkdruckern, um deren Integration zu ermöglichen. Für ein klinikweites Auffinden von Web Services wird in der Architektur ein zentraler Discovery Proxy verwendet. Dieser ist notwendig, da die Architektur sonst nicht skalieren würde.

Die vorgestellte Discovery-Architektur nimmt die Rolle des Verzeichnisdienstes der in dieser Arbeit entwickelten SOA ein. In der Abbildung 7.1 bildet die Discovery-Architektur die „Auffinden“-Ebene. Sie verwendet die Adressierung, die auf DHCP-Basis bzw. bei separaten Netzwerken auf der IP-Auto-Konfiguration erfolgt. Für die Beschreibung wird auf DPWS zurückgegriffen, dass mittels WS-MetadataExchange diverse Metadaten zum Gerät selbst und für seine Services WSDL-Beschreibungen bereitstellt.

Eine vollständige Vernetzung von medizinischen Geräten – entsprechend dem SOA-Konzept – birgt Risiken. Geräte, die in die Behandlung eines Patienten involviert sind, können bei Fehlfunktion dem Patienten Schaden zufügen. Gerade in diesem Bereich ist es wichtig, garantierte Datenübertragungen zu ermöglichen, insbesondere dann, wenn ein Gerät von einem anderen ferngesteuert wird. Im Enterprise-Bereich sind dagegen Lösungen auf Basis einer Reliable-Messaging-Spezifikation ausreichend, da dort ein einfaches Wiederholen der Übertragung genügt. Bestehende Ethernet- und TCP/IP-Ansätze für QoS gehen von kooperativen Teilnehmern aus. Da dies aber nicht gewährleistet ist, werden QoS-Merkmale meist deaktiviert.

Um garantierte Datenübertragungen zwischen medizinischen Geräten zu gewährleisten, wurde eine spezielle Switch-Architektur entwickelt (vgl. Kapitel 4). Diese ermöglicht es, bei einem $n$-Port-Switch mithilfe von $n^2$ Queues und einem fairen Verteilen der Datenrate, Übertragungen zu garantieren. Dabei wird eine Kooperation der beteiligten Geräte nicht vorausgesetzt. Für jede Verbindung zwischen zwei Geräten, die an diesem Switch angeschlossen sind, wird $\frac{1}{n}$ der maximalen Datenrate garantiert.

Für die in dieser Arbeit entwickelte SOA lässt sich der Switch z. B. in Operationssälen oder am Patientenbett auf Intensiv- und Überwachungsstationen einsetzen, um die Patientensicherheit zu erhöhen. Der Switch garantiert allen über ihn kommunizierenden Geräten eine fixe Datenrate und trägt damit zur Verfügbarkeit des Netzes bei. In der entwickelten SOA wird er daher für die Web-Service-Kommunikation der Services untereinander verwendet (vgl. Abbildung 7.1 „Play“-Gruppe).

Neben der Verfügbarkeit der Services ist es notwendig, in der SOA eine Zugriffskontrolle umzusetzen. Auf KIS-Ebene ist eine Authentifizierung und Autorisierung des Personals notwendig, da aus datenschutzrechtlichen Gründen die Vertraulichkeit von Patientendaten gewährleistet sein muss. Auf Geräteebene sind dagegen andere Anforderungen an eine Zugriffskontrolle zu stellen. Die Zugriffskontrolle betrifft hier die Kommunikation der Services bzw. Geräte untereinander. Dabei stehen die Patientensicherheit und das dazugehörige Risikomanagement im Vordergrund. Auch ist eine Zurechenbarkeit wichtig, damit sich Hersteller von Geräten gegenüber anderen Herstellern absichern können.

Das entwickelte Sicherheitskonzept für die SOA sieht eine dezentrale Zugriffskontrolle vor, um PnP zu ermöglichen (vgl. Kapitel 5). Die Geräte selbst enthalten alle notwendigen Sicherheitsinformationen, um sich gegenseitig Zugriff zu gewähren bzw. zu verweigern. Ein klinikweiter Authentifizierungs- und Autorisierungsdienst ist nicht notwendig. Ziel ist, dass die Geräte entsprechend des PnP-Prinzips ihre Services nutzen können, sobald sie miteinander vernetzt worden sind und sie die entsprechende Berechtigung haben.

Die Sicherheitsinformationen beinhalten sowohl die Service-Typen und die jeweilige Rolle (Service-Anbieter bzw. Service-Nutzer), die das Geräte dabei einnimmt, als auch weitere gerätespezifische Informationen. Sie lassen sich entweder vom Hersteller oder

vom Betreiber eines Gerätes hinterlegen. Speichern die Hersteller die notwendigen Sicherheitsinformationen auf dem Gerät, so enthalten diese die Zweckbestimmung des Gerätes. Dies scheint im Vergleich zu einer Zugriffskontrolle durch den Betreiber die sinnvollere Variante zu sein. Sie ermöglicht PnP, da eine Konfiguration durch den Betreiber entfällt. Das notwendige Risikomanagement des Betreibers ist sinnvollerweise bereits vor Inbetriebnahme eines Gerätes soweit erfolgreich durchzuführen, dass einem Testen des Gerätes im Netzwerk nichts mehr im Wege steht. Anschließend kann das Gerät ohne Konfigurationsaufwand ins Netzwerk eingebunden und das Risikomanagement abgeschlossen werden.

Die Zugriffskontrolle für Web Services ist insbesondere für eine Fernsteuerung von Geräten von Bedeutung, weswegen sie in Abbildung 7.1 in der „Steuerung“-Ebene angesiedelt ist.

Darüber hinaus wurde ermittelt, wo bei Web Services Performance-Verbesserungen notwendig bzw. möglich sind (vgl. Kapitel 6). Zwei Szenarien wurden betrachtet, die für die entwickelte SOA von Bedeutung sind: zum einen eine Fernsteuerung von medizinischen Geräten und zum anderen eine periodische Datenverteilung von Messwerten verschiedener Geräte. Für beide Szenarien wurden Performance-Vergleichsmessungen zwischen Web Services und DDS, einer Middleware für Echtzeitsysteme, durchgeführt.

Die Messungen haben gezeigt, dass standardisierte Web Services auf Basis von HTTP einen ausreichend schnellen Datentransfer für die Fernsteuerung von Geräten ermöglichen. Das zeigt sich auch daran, dass der Overhead von Web Services im Vergleich zu notwendigen Sicherheitsmechanismen gering ist. Andere Middlewares zur Kommunikation könnten hier nur einen kleinen Performance-Gewinn verbuchen. Außerdem haben die Messungen ergeben, dass SOAP-over-UDP für eine periodisch auftretende Datenverteilung mit hoher Frequenz, wie z. B. beim Patienten-Monitoring, geeignet ist. Dementsprechend wird in der entwickelten SOA für die Datenverteilung SOAP-over-UDP verwendet, während WS-Eventing für Ereignismeldungen vorgesehen ist, da es speziell dafür konzipiert ist (vgl. Abbildung 7.1).

Nach wie vor stellt die Performance bei der Übertragung von Integer-Arrays mittels XML-Serialisierung einen Schwachpunkt dar. Es existieren bereits Forschungsarbeiten speziell zu diesem Thema, jedoch hat sich bisher noch keine Lösung durchsetzen können. Eine binäre Serialisierung bzw. Kompression auf SOAP-Ebene ist allgemein hilfreich, da die übliche XML-Serialisierung ersetzt wird und somit eine bessere Performance erreicht werden kann.

Die vorgestellte SOA bildet ein technisches Grundgerüst, um im medizinischen Bereich technische Interoperabilität auf Basis von Web Services zu ermöglichen. Darauf aufbauend ist es notwendig, generische Service-Beschreibungen für Geräteklassen zu entwickeln. Diese können in Geräteprofilen verwendet werden, um semantische Interoperabilität zu erreichen. Hierfür ist es sinnvoll, bestehende Semantiken aus der

ISO-Reihe 11073 und HL7 soweit wie möglich zu übernehmen und auf Basis der hier vorgestellten Architektur umzusetzen.

Zusammenfassend lässt sich festhalten, dass in dieser Arbeit eine SOA entwickelt wurde, um technische Interoperabilität im medizinischen Bereich zu ermöglichen. Die SOA deckt dabei den Bereich der Kommunikation von einzelnen Geräten bis zum Enterprise/KIS-Bereich ab und lässt sich als Basis für weitergehende semantische Interoperabilität nutzen.

# Literaturverzeichnis


[1] Abu-Ghazaleh, Nayef ; Lewis, Michael J.: Differential Deserialization for Optimized SOAP Performance. In: *Proceedings of the 2005 ACM/IEEE SC|05 Conference (SC'05)*. Seattle, WA, USA : IEEE Computer Society, November 2005. – DOI 10.1109/SC.2005.24. – ISBN 1–59593–061–2

[2] Alexander, Steve ; Droms, Ralph: *DHCP Options and BOOTP Vendor Extensions*. RFC 2132 (Draft Standard). `http://tools.ietf.org/html/rfc2132`. Version: März 1997

[3] Almquist, Philip: *Type of Service in the Internet Protocol Suite*. RFC 1349 (Proposed Standard). `http://tools.ietf.org/html/rfc1349`. Version: Juli 1992. – Obsoleted by: RFC 2474

[4] Alor-Hernandez, Giner ; Posada-Gomez, Ruben ; Aguilar-Lasserre, Alberto A. ; Abud-Figueroa, MA. A.: Web Services Discovery and Invocation by using DNS-EPD. In: *Proceedings of the Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007)*. Cuernavaca, Morelos, Mexico : IEEE Computer Society, September 2007. – DOI 10.1109/CERMA.2007.139. – ISBN 0–7695–2974–7, 695–700

[5] ANSI/HL7 V2.6-2007: *Health Level Seven Standard Version 2.6 - An Application Protocol for Electronic Data Exchange in Healthcare Environments (revision of ANSI/HL7 V2.5.1-2007)*. Oktober 2007

[6] ANSI/TIA/EIA-485-A-98: *Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems*. März 1998

[7] Arney, David ; Goldman, Julian M. ; Whitehead, Susan F. ; Lee, Insup: Synchronizing an X-ray and Anesthesia Machine Ventilator: A Medical Device Interoperability Case Study. In: *Proceedings of the International Conference on Biomedical Electronics and Devices (BioDevices)*. Porto, Portugal : INSTICC Press, Januar 2009. – ISBN 978–989–8111–64–7, S. 52–60

[8] Ballinger, Keith ; Brittenham, Peter ; Malhotra, Ashok ; Nagy, William A. ; Pharies, Stefan: *Web Services Inspection Language (WS-Inspection) 1.0*. `http://www.ibm.com/developerworks/library/specification/ws-wsilspec/`. Version: November 2001

[9] Beatty, John ; Kakivaya, Gopal ; Kemp, Devon ; Kuehnel, Thomas ; Lovering, Brad ; Roe, Bryan ; John, Christopher S. ; Schlimmer, Jeffrey ; Simonnet,

Guillaume ; Walter, Doug ; Weast, Jack ; Yarmosh, Yevgeniy ; Yendluri, Prasad: *Web Services Dynamic Discovery (WS-Discovery)*. http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf. Version: April 2005

[10] Berners-Lee, Tim ; Fielding, Roy T. ; Masinter, Larry: *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986 (Standard). http://tools.ietf.org/html/rfc3986. Version: Januar 2005

[11] Bundesgesetz: *Gesetz über Medizinprodukte (Medizinproduktegesetz – MPG)*. August 2002. – Neufassung

[12] Callas, Jon ; Donnerhacke, Lutz ; Finney, Hal ; Shaw, David ; Thayer, Rodney: *OpenPGP Message Format*. RFC 4880 (Proposed Standard). http://tools.ietf.org/html/rfc4880. Version: November 2007

[13] CAN in Automation (CiA): *Controller Area Network (CAN)*. http://www.can-cia.org/, Abruf: 2009-10-15

[14] Capsule Technologie: http://www.capsuletech.com/

[15] Catania, Nicolas ; Kumar, Pankaj ; Murray, Bryan ; Pourhedari, Homayoun ; Vambenepe, William ; Wurster, Klaus: *Web Services Events (WS-Events) Version 2.0*. Version: Juli 2006. http://xml.coverpages.org/WS-Events20030721.pdf, Abruf: 2009-10-27

[16] Cheshire, Stuart ; Aboba, Bernard ; Guttman, Erik: *Dynamic Configuration of IPv4 Link-Local Addresses*. RFC 3927 (Proposed Standard). http://tools.ietf.org/html/rfc3927. Version: Mai 2005

[17] Chiu, Kenneth ; Govindaraju, Madhusudhan ; Bramley, Randall: Investigating the Limits of SOAP Performance for Scientific Computing. In: *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing HPDC-11 2002 (HPDC'02)*. Edinburgh, Scotland : IEEE Computer Society, Juli 2002. – DOI 10.1109/HPDC.2002.1029924. – ISBN 0–7695–1686–6, S. 246–254

[18] Cline, Kevin ; Cohen, Josh ; Davis, Doug ; Ferguson, Donald F. ; Kreger, Heather ; McCollum, Raymond ; Murray, Bryan ; Robinson, Ian ; Schlimmer, Jeffrey ; Shewchuk, John ; Tewari, Vijay ; Vambenepe, William: *Toward Converging Web Service Standards for Resources, Events, and Management*. Version: März 2006. http://msdn.microsoft.com/en-us/library/aa480724.aspx, Abruf: 2009-09-08

[19] Cooper, David ; Santesson, Stefan ; Farrell, Stephen ; Boeyen, Sharon ; Housley, Russell ; Polk, Tim: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280 (Proposed Standard). http://tools.ietf.org/html/rfc5280. Version: Mai 2008

[20] D2D: *Die Telematik-Plattform der Kassenärztlichen Vereinigungen*. http://www.d2d.de/, Abruf: 2009-11-10

[21] Davie, Bruce ; Charny, Anna ; Bennett, Jon ; Benson, Kent ; Le Boudec, Jean-Yves ; Courtney, Bill ; Davari, Shahram ; Firoiu, Victor ; Stiliadis, Dimitrios: *An Expedited Forwarding PHB (Per-Hop Behavior).* RFC 3246 (Proposed Standard). `http://tools.ietf.org/html/rfc3246`. Version: März 2002

[22] de Deugd, Scott ; Carroll, Randy ; Kelly, Kevin ; Millett, Bill ; Ricker, Jeffrey: SODA: Service Oriented Device Architecture. In: *IEEE Pervasive Computing* 5 (2006), Juli–Sep, Nr. 3, S. 94–96, c3. – DOI 10.1109/MPRV.2006.59. – ISSN 1536–1268

[23] Decotignie, Jean-Dominique: Ethernet-Based Real-Time and Industrial Communications. In: *Proceedings of the IEEE* 93 (2005), Juni, Nr. 6, S. 1102–1117. – DOI 10.1109/JPROC.2005.849721. – ISSN 0018–9219

[24] Deering, Stephen E. ; Hinden, Robert M.: *Internet Protocol, Version 6 (IPv6) Specification.* RFC 2460 (Draft Standard). `http://tools.ietf.org/html/rfc2460`. Version: Dezember 1998

[25] DICOM PS 3.1-2008: *Digital Imaging and Communications in Medicine (DICOM) – Part 1: Introduction and Overview.* Virginia, USA, 2008

[26] DICOM PS 3.15-2008: *Digital Imaging and Communications in Medicine (DICOM) – Part 15: Security and System Management Profiles.* 2008

[27] Dierks, Tim (Hrsg.) ; Rescorla, Eric (Hrsg.): *The Transport Layer Security (TLS) Protocol Version 1.2.* RFC 5246 (Proposed Standard). `http://tools.ietf.org/html/rfc5246`. Version: August 2008

[28] DIN EN ISO 13485:2007-10: *Medizinprodukte – Qualitätsmanagementsysteme – Anforderungen für regulatorische Zwecke (ISO 13485:2003); Deutsche Fassung EN ISO 13485:2003+AC:2007.* Oktober 2007

[29] DIN EN ISO 14971:2007-07: *Medizinprodukte – Anwendung des Risikomanagements auf Medizinprodukte (ISO 14971:2007); Deutsche Fassung EN ISO 14971:2007.* Juli 2007

[30] DIN PAS 1011:2001-03: *VCS Kommunikationskonzepte für das Gesundheitswesen.* Technische Regel, März 2001

[31] Distributed Management Task Force, Inc.: `http://www.dmtf.org/`

[32] DMTF Specification: *Web Services for Management (WS-Management).* `http://www.dmtf.org/standards/published_documents/DSP0226_1.0.0.pdf`. Version: April 2006

[33] dnsjava: `http://www.dnsjava.org/`, Abruf: 2009-10-09

[34] DOC Group: *Pub/Sub Data Distribution Service Benchmarking.* Version: April 2007. `http://www.dre.vanderbilt.edu/DDS/`, Abruf: 2009-10-29

[35] Droms, Ralph: *Dynamic Host Configuration Protocol.* RFC 2131 (Draft Standard). `http://tools.ietf.org/html/rfc2131`. Version: März 1997

[36] Elfwing, Robert ; Paulsson, Ulf ; Lundberg, Lars: Performance of SOAP in Web Service Environment Compared to CORBA. In: *Proceedings of the Ninth Asia-Pacific Software Engineering Conference (APSEC'02)*. Gold Coast, Australia : IEEE Computer Society, Dezember 2002. – DOI 10.1109/APSEC.2002.1182978. – ISBN 0–7695–1850–8, S. 84–93

[37] Engelmann, Uwe ; Schütze, Bernd ; Schröter, André ; Weisser, Gerald ; Walz, Michael ; Kämmerer, Marc ; Mildenberger, Peter: Teleradiologie per DICOM-E-Mail: Der empfohlene Minimalstandard der Deutschen Röntgengesellschaft. In: *Telemed 2005 – Tagungsband zur 10. Fortbildungsveranstaltung und Arbeitstagung*, 2005

[38] EtherCAT Technology Group (ETG): *Technical Introduction and Overview*. Version: August 2009. `http://www.ethercat.org/en/technology.html`, Abruf: 2009-10-15

[39] Ethernet POWERLINK Standardization Group (EPSG): *POWERLINK*. `http://www.ethernet-powerlink.org/`, Abruf: 2009-10-15

[40] EtherNet/IP: `http://www.ethernetip.de/`, Abruf: 2009-10-16

[41] Felser, Max: Real-Time Ethernet—Industry Prospective. In: *Proceedings of the IEEE* Bd. 93, 2005. – DOI 10.1109/JPROC.2005.849720. – ISSN 0018–9219, S. 1118–1129

[42] Felser, Max: Real Time Ethernet for Automation Applications. In: *Embedded Systems Handbook, Second Edition: Networked Embedded Systems*. CRC Press, Juni 2009. – ISBN 1–4398–0761–2, Kapitel 21

[43] Fielding, Roy T.: *Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine, Diss., 2000. `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`

[44] Floyd, Sally ; Jacobson, Van: Random Early Detection Gateways for Congestion Avoidance. In: *IEEE/ACM Transactions on Networking* 1 (1993), August, Nr. 4, S. 397–413. – DOI 10.1109/90.251892. – ISSN 1063–6692

[45] Galarraga, Miguel ; Serrano, Luis ; Martínez, Ignacio ; Toledo, Paula de: Standards for Medical Device Communication: X73 PoC-MDC. In: *International Council on Medical & Care Compunetics, ICMCC Event*. La Haya, Holland, Juni 2006

[46] Goland, Yaron Y. ; Cai, Ting ; Leach, Paul ; Gu, Ye ; Albright, Shivaun: *Simple Service Discovery Protocol/1.0 – Operating without an Arbiter*. IETF Internet draft. `http://www3.tools.ietf.org/html/draft-cai-ssdp-v1-03`. Version: Oktober 1999. – expired

[47] Govindaraju, Madhusudhan ; Slominski, Aleksander ; Chiu, Kenneth ; Liu, Pu ; Engelen, Robert van ; Lewis, Michael J.: Toward Characterizing the Performance of SOAP Toolkits. In: *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*. Pittsburgh, PA, USA

: IEEE Computer Society, November 2004. – DOI 10.1109/GRID.2004.60. – ISSN 1550–5510, S. 365–372

[48] Guttman, Erik ; Perkins, Charles E. ; Veizades, John ; Day, Michael: *Service Location Protocol, Version 2*. RFC 2608 (Proposed Standard). `http://tools.ietf.org/html/rfc2608`. Version: Juni 1999

[49] gzip: *The gzip home page*. Version: Juli 2003. `http://www.gzip.org/`, Abruf: 2009-11-08

[50] Haas, Peter ; Eckenbach, Mandy ; Plagge, Hermann ; Schiprowski, Witold: Integration von Anwendungssystemen des stationären und ambulanten Versorgungssektors am Beispiel des Projektes Mamma@kte.nrw. In: *Enterprise Application Integration 2005, Proceedings of the 2nd GI-Workshop on Enterprise Application Integration (EAI-05)* Bd. 141. Marburg, 2005 (CEUR Workshop Proceedings)

[51] Health Level Seven, Inc.: `http://www.hl7.org/`

[52] Heinanen, Juha ; Baker, Fred ; Weiss, Walter ; Wroclawski, John: *Assured Forwarding PHB Group*. RFC 2597 (Proposed Standard). `http://tools.ietf.org/html/rfc2597`. Version: Juni 1999

[53] HL7 Version 3 Standard ; Ruggeri, Roberto (Hrsg.): *Transport Specification - Web Services Profile, Release 2*. `http://www.hl7.org/v3ballot/html/infrastructure/transport/transport-wsprofiles.htm`. Version: August 2009. – Draft Document

[54] Hubert, Bert (Hrsg.): *Linux Advanced Routing & Traffic Control HOWTO*. Version: März 2004. `http://lartc.org/howto/`, Abruf: 2009-10-19

[55] Ibach, Bastian ; Kanert, Achim ; Zimolong, Andreas ; Bulitta, Clemens ; Radermacher, Klaus: Concept of a service-oriented integration architecture for the orthopaedic operating theatre. In: *International Journal of Computer Assisted Radiology and Surgery* 3 (2008), Juni, Nr. 1, S. 446–447. – DOI 10.1007/s11548–008–0209–6. – ISSN 1861–6429

[56] IBM: *WSDM/WS-Man Reconciliation – An Overview and Migration Guide*. Version: Mai 2007. `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-wsdmmgmt/wsdmmgmt_v2.pdf`

[57] IBM alphaWorks: *Web Services Toolkit*. Version: Juli 2000. `http://www.alphaworks.ibm.com/tech/webservicestoolkit`, Abruf: 2009-11-03

[58] IEEE 1588-2002: *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. IEEE Standard, 2002

[59] IEEE 802.11-2007: *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access*

*Control (MAC) and Physical Layer (PHY) Specifications.* IEEE Standard, Juni 2007

[60] IEEE 802.1D-2004: *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges.* IEEE Standard, Juni 2004

[61] IEEE 802.1Q-2005: *IEEE Standard for Local and metropolitan area networks – Virtual Bridged Local Area Networks.* IEEE Standard, Mai 2006

[62] IEEE 802.3-2008: *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications.* IEEE Standard, Dezember 2008

[63] IHE International: *Integrating the Healthcare Enterprise.* `http://www.ihe.net/`

[64] Internet Systems Consortium: *ISC DHCP.* `https://www.isc.org/software/dhcp`, Abruf: 2009-10-08

[65] ISO 17090:2008: *Health informatics – Public key infrastructure.* Februar 2008

[66] ISO 8879:1986: *Information processing – Text and office systems – Standard Generalized Markup Language (SGML).* 1986

[67] ISO/HL7 21731:2006-08: *Medizinische Informatik – HL 7 Version 3 – Referenzinformationsmodell, Ausgabe 1.* 2006

[68] ISO/IEC 19757-2:2008: *Information technology – Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG.* 2008

[69] ISO/IEEE 11073-10101:2004: *Health informatics – Point-of-care medical device communication – Part 10101: Nomenclature.* 2004

[70] ISO/IEEE 11073-10201:2004: *Health informatics – Point-of-care medical device communication – Part 10201: Domain information model.* 2004

[71] ISO/IEEE 11073-20101:2004: *Health informatics – Point-of-care medical device communication – Part 20101: Application profiles – Base standard.* 2004

[72] ISO/IEEE 11073-30200:2004: *Health informatics – Point-of-care medical device communication – Part 30200: Transport profile – Cable connected.* 2004

[73] ISO/IEEE 11073-30300:2004: *Health informatics – Point-of-care medical device communication – Part 30300: Transport profile – Infrared wireless.* 2004

[74] ISO/TC 215 – IEC/SC 62A JWG 7: *Application of risk management for IT-networks incorporating medical devices.* 2008. – Committee Draft (CD2)

[75] Jasperneite, Jürgen ; Neumann, Peter: How to Guarantee Realtime Behavior using Ethernet. In: *11th IFAC Symposium on Information Control Problems in*

*Manufacturing (INCOM'2004)* Bd. 1. Salvador-Bahia, Brazil : Elsevier, April 2004. – ISBN 0–08–044249–8, S. 91–96

[76] Josefsson, Simon: *The Base16, Base32, and Base64 Data Encodings.* RFC 4648 (Proposed Standard). `http://tools.ietf.org/html/rfc4648`. Version: Oktober 2006

[77] Juric, Matjaz B.; Rozman, Ivan; Brumen, Bostjan; Colnaric, Matjaz; Hericko, Marjan: Comparison of performance of Web services, WS-Security, RMI, and RMI–SSL. In: *Journal of Systems and Software* 79 (2006), Mai, Nr. 5, S. 689–700. – DOI 10.1016/j.jss.2005.08.006. – ISSN 0164–1212

[78] Kaiser Permanente: Analysis of Implementing Integrated Systems. In: *MDPnP Booklet February 2007* (2007), Februar, 12–14. `http://mdpnp.org/Publications.html`

[79] Kangasharju, Jaakko; Tarkoma, Sasu; Raatikainen, Kimmo: Comparing SOAP Performance for Various Encodings, Protocols, and Connections. In: *Personal Wireless Communications* Bd. 2775. Venice, Italy : Springer, September 2003 (Lecture Notes in Computer Science). – DOI 10.1007/b12004. – ISBN 978–3–540–20123–6, S. 397–406

[80] Kweon, Seok-Kyu; Shin, Kang G.; Zheng, Qin: Statistical Real-Time Communication over Ethernet for Manufacturing Automation Systems. In: *Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium (RTAS'99).* Vancouver, BC, Canada : IEEE Computer Society, Juni 1999. – DOI 10.1109/RTTAS.1999.777672. – ISBN 0–7695–0194–X, S. 192–202

[81] Leach, Paul J.; Mealling, Michael; Salz, Richard: *A Universally Unique IDentifier (UUID) URN Namespace.* RFC 4122 (Proposed Standard). `http://tools.ietf.org/html/rfc4122`. Version: Juli 2005

[82] Lesh, Kathy; Weininger, Sandy; Goldman, Julian M.; Wilson, Bob; Himes, Glenn: Medical Device Interoperability-Assessing the Environment. In: *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP).* Cambridge, Massachusetts, USA : IEEE Computer Society, Juni 2007. – DOI 10.1109/HCMDSS–MDPnP.2007.22. – ISBN 0–7695–3081–8, S. 3–12

[83] Levinson, Edward: *Content-ID and Message-ID Uniform Resource Locators.* RFC 2111 (Proposed Standard). `http://tools.ietf.org/html/rfc2111`. Version: März 1997

[84] Levinson, Edward: *The MIME Multipart/Related Content-type.* RFC 2387 (Proposed Standard). `http://tools.ietf.org/html/rfc2387`. Version: August 1998

[85] Lockhart, Hal; Andersen, Steve; Bohren, Jeff; Sverdlov, Yakov; Hondo, Maryann; Maruyama, Hiroshi; Nagaratnam, Nataraj; Boubez, Toufic; Morrison, K S.; Nanda, Arun; Schmidt, Don; Walters, Doug; Wilson,

Hervey ; Burch, Lloyd ; Earl, Doug ; Baja, Siddharth ; Prafullchandra, Hemma ; Nadalin, Anthony (Hrsg.) ; Kaler, Chris (Hrsg.): *Web Services Federation Language (WS-Federation).* Version: Dezember 2006. http://www.ibm.com/developerworks/library/specification/ws-fed/, Abruf: 2009-10-28

[86] Lofsky, Ann S.: TURN YOUR ALARMS ON! In: *Newsletter – The Official Journal of the Anesthesia Patient Safety Foundation* 19 (2004), Nr. 4, S. 43

[87] Massoulie, Laurent ; Roberts, James: Bandwidth Sharing: Objectives and Algorithms. In: *Proceedings of 18th Annual Joint Conference of the IEEE Computer and Communications Societies. (INFOCOM '99)* Bd. 3. New York, NY, USA, März 1999. – DOI 10.1109/INFCOM.1999.752159. – ISBN 0–7803–5417–6, S. 1395–1403

[88] MD PnP: *Medical Device "Plug-and-Play" Interoperability Program.* Version: 2004. http://mdpnp.org/MD_PnP_Program.html, Abruf: 2009-09-08

[89] Medina, Alberto ; Allman, Mark ; Floyd, Sally: Measuring Interactions Between Transport Protocols and Middleboxes. In: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (IMC '04).* Taormina, Sicily, Italy : ACM, Oktober 2004. – DOI 10.1145/1028788.1028835. – ISBN 1–58113–821–0, S. 336–341

[90] Microsoft Corporation: *Discovery of Web Services (DISCO).* http://lists.ebxml.org/archives/ebxml-transport/200007/msg00007.html. Version: Juli 2000. – Entwurf

[91] Microsoft Corporation: *UDDI Business Registry Shutdown FAQ.* http://uddi.microsoft.com/about/FAQshutdown.htm. Version: Dezember 2005

[92] Mildenberger, Peter ; Kämmerer, Marc ; Engelmann, Uwe ; Ruggiero, Stephan ; Klos, Gordon ; Runa, Alain ; Schröter, Andre ; Weisser, Gerald ; Walz, Michael ; Schütze, Bernd: Teleradiologie mit DICOM E-mail: Empfehlungen der @GIT. In: *RöFo - Fortschritte auf dem Gebiet der Röntgenstrahlen und der bildgebenden Verfahren (Fortschr Röntgenstr)* 177 (2005), Nr. 5, S. 697–702. – DOI 10.1055/s–2005–858049. – ISSN 1438–9029

[93] Mockapetris, Paul: *Domain Names – Concepts and Facilities.* RFC 1034 (Standard). http://tools.ietf.org/html/rfc1034. Version: November 1987

[94] Mockapetris, Paul: *Domain Names – Implementation and Specification.* RFC 1035 (Standard). http://tools.ietf.org/html/rfc1035. Version: November 1987

[95] Modbus-IDA: *MODBUS Messaging on TCP/IP Implementation Guide V1.0b.* Version: Oktober 2006. http://www.modbus.org/specs.php, Abruf: 2009-10-16

[96] Nagy, William ; Curbera, Francisco ; Weerawarana, Sanjiva: *The Advertisement and Discovery of Services (ADS) protocol for Web services.* http:

//www.ibm.com/developerworks/library/ws-ads.html. Version: Oktober 2000

[97] Nee, Oliver ; Hein, Andreas ; Gorath, Torsten ; Hülsmann, Nils ; Laleci, Gokce B. ; Yuksel, Mustafa ; Olduz, Mehmet ; Tasyurt, Ibrahim ; Orhan, Umut ; Dogac, Asuman ; Fruntelata, Ana ; Ghiorghe, Silviu ; Ludwig, Ralf: SAPHIRE: Intelligent Healthcare Monitoring based on Semantic Interoperability Platform - Pilot Applications. In: *IET Communications* 2 (2008), Februar, Nr. 2, S. 192–201. – DOI 10.1049/iet–com:20060699

[98] Nichols, Kathleen ; Blake, Steven ; Baker, Fred ; Black, David L.: *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*. RFC 2474 (Proposed Standard). http://tools.ietf.org/html/rfc2474. Version: Dezember 1998

[99] Nosov, Alexander ; Hately, Andrew ; Reistad, Brian ; Murray, Bryan ; Davis, Doug ; Kreger, Heather ; Niblett, Peter ; McCollum, Raymond ; Tewari, Vijay ; Kumbalimutt, Vishwa ; Vambenepe, William: *Web Services Resource Catalog (WS-RC)*. http://schemas.xmlsoap.org/ws/2007/05/ResourceCatalog/WS-ResourceCatalog.pdf. Version: Mai 2007

[100] OASIS Committee Draft 02 ; Banks, Tim (Hrsg.): *Web Services Resource Framework (WSRF) – Primer v1.2*. http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf. Version: Mai 2006

[101] OASIS Security Services TC: *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. Committee Draft. http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf. Version: März 2008

[102] OASIS Standard ; Nadalin, Anthony (Hrsg.) ; Kaler, Chris (Hrsg.) ; Hallam-Baker, Phillip (Hrsg.) ; Monzillo, Ronald (Hrsg.): *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)*. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf. Version: März 2004

[103] OASIS Standard: *WS-Reliability 1.1*. http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf. Version: November 2004

[104] OASIS Standard ; Cantor, Scott (Hrsg.) ; Kemp, John (Hrsg.) ; Philpott, Rob (Hrsg.) ; Maler, Eve (Hrsg.): *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf. Version: März 2005

[105] OASIS Standard: *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf. Version: März 2005

[106] OASIS Standard: *Web Services Base Notification 1.3 (WS-BaseNotification).* http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.htm. Version: Oktober 2006

[107] OASIS Standard: *Web Services Brokered Notification 1.3 (WS-BrokeredNotification).* http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.htm. Version: Oktober 2006

[108] OASIS Standard ; Wilson, Kirk (Hrsg.) ; Sedukhin, Igor (Hrsg.): *Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1.* http://docs.oasis-open.org/wsdm/wsdm-mows-1.1-spec-os-01.htm. Version: August 2006

[109] OASIS Standard ; Bullard, Vaughn (Hrsg.) ; Vambenepe, William (Hrsg.): *Web Services Distributed Management: Management Using Web Services (MUWS 1.1) Part 1.* http://docs.oasis-open.org/wsdm/wsdm-muws1-1.1-spec-os-01.htm. Version: August 2006

[110] OASIS Standard ; Monzillo, Ronald (Hrsg.) ; Kaler, Chris (Hrsg.) ; Nadalin, Anthony (Hrsg.) ; Hallem-Baker, Phillip (Hrsg.): *Web Services Security: SAML Token Profile 1.1.* http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SAMLTokenProfile.pdf. Version: Februar 2006

[111] OASIS Standard: *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004).* http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf. Version: Februar 2006

[112] OASIS Standard: *Web Services Security X.509 Certificate Token Profile 1.1.* http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-x509TokenProfile.pdf. Version: Februar 2006

[113] OASIS Standard: *Web Services Topics 1.3 (WS-Topics).* http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.htm. Version: Oktober 2006

[114] OASIS Standard ; Alves, Alexandre (Hrsg.) ; Arkin, Assaf (Hrsg.) ; Askary, Sid (Hrsg.) ; Barreto, Charlton (Hrsg.) ; Bloch, Ben (Hrsg.) ; Curbera Francisco (Hrsg.) ; Ford, Mark (Hrsg.) ; Goland, Yaron (Hrsg.) ; Guízar Alejandro (Hrsg.) ; Kartha, Neelakantan (Hrsg.) ; Liu, Canyang K. (Hrsg.) ; Khalaf, Rania (Hrsg.) ; König, Dieter (Hrsg.) ; Marin, Mike (Hrsg.) ; Mehta, Vinkesh (Hrsg.) ; Thatte, Satish (Hrsg.) ; Rijn, Danny van d. (Hrsg.) ; Yendluri, Prasad (Hrsg.) ; Yiu, Alex (Hrsg.): *Web Services Business Process Execution Language Version 2.0.* http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html. Version: April 2007

[115] OASIS Standard: *WS-SecureConversation 1.3.* http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.3/ws-secureconversation.html. Version: März 2007

[116] OASIS Standard; Nadalin, Anthony (Hrsg.); Goodner, Marc (Hrsg.); Gudgin, Martin (Hrsg.); Barbir, Abbie (Hrsg.); Granqvist, Hans (Hrsg.): *WS-Trust 1.3*. http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf. Version: März 2007

[117] OASIS Standard: *Devices Profile for Web Services Version 1.1*. http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html. Version: Juli 2009

[118] OASIS Standard: *SOAP-over-UDP Version 1.1*. http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.html. Version: Juli 2009

[119] OASIS Standard: *Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.2*. http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec.html. Version: Februar 2009

[120] OASIS Standard: *Web Services Dynamic Discovery (WS-Discovery) Version 1.1*. http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html. Version: Juli 2009

[121] OASIS Standard: *Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.2*. http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.2-spec-os.html. Version: Februar 2009

[122] OASIS Standard: *WS-SecurityPolicy 1.3*. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.html. Version: Februar 2009

[123] OASIS UDDI: *Introduction to UDDI: Important Features and Functional Concepts*. http://uddi.xml.org/files/uddi-tech-wp.pdf. Version: Oktober 2004

[124] OASIS UDDI Specification Technical Committee; Clement, Luc (Hrsg.); Hately, Andrew (Hrsg.); Riegen, Claus von (Hrsg.); Rogers, Tony (Hrsg.): *UDDI Version 3.0.2*. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm. Version: Oktober 2004

[125] Object Computing, Inc.: *Webseite*. http://www.ociweb.com/, Abruf: 2009-10-29

[126] Object Management Group: *Data Distribution Service for Real-time Systems, Version 1.2*. http://www.omg.org/spec/DDS/1.2/. Version: Januar 2007

[127] Object Management Group: *Common Object Request Broker Architecture (CORBA)*. http://www.omg.org/spec/CORBA/3.1/. Version: Januar 2008

[128] Object Management Group: *The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification*. http://www.omg.org/spec/DDSI/2.1/. Version: Januar 2009

[129] OID Repository: *Object Identifier (OID) Repository.* Version: Juni 2009. `http://www.oid-info.com/`, Abruf: 2009-10-23

[130] Organization for the Advancement of Structured Information Standards (OASIS): `http://www.oasis-open.org/`

[131] Pharow, Peter ; Blobel, Bernd ; Goetz, Christoph ; Engel, Kjeld: Karten im deutschen Gesundheitswesen und der Einfluss der internationalen Standardisierung. In: *50. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (gmds), 12. Jahrestagung der Deutschen Arbeitsgemeinschaft für Epidemiologie.* Freiburg im Breisgau, September 2005

[132] Plagge, Hermann ; Haas, Peter ; Heupel, Rudolf ; Naujokat, Frederic: Entwicklung eines Health Telematic Brokers (HTB) zur Realisierung einer sektorübergreifenden Interoperabilität. In: *50. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (gmds), 12. Jahrestagung der Deutschen Arbeitsgemeinschaft für Epidemiologie.* Freiburg im Breisgau : German Medical Science, September 2005

[133] Pöhlsen, Stephan ; Buschmann, Carsten ; Werner, Christian: Integrating a Decentralized Web Service Discovery System into the Internet Infrastructure. In: *Proceedings of the Sixth IEEE European Conference on Web Services.* Dublin, Irland : IEEE Computer Society, November 2008. – DOI 10.1109/ECOWS.2008.15. – ISBN 978–0–7695–3399–5, S. 13–20

[134] Pöhlsen, Stephan ; Franz, Frank ; Kück, Kai ; Meyer, Jörg-Uwe ; Werner, Christian: Fair-Queued Ethernet for Medical Applications. In: *Proceedings of the 2008 Seventh IEEE International Symposium on Network Computing and Applications.* Cambridge, MA, USA : IEEE Computer Society, Juli 2008. – DOI 10.1109/NCA.2008.19. – ISBN 978–0–7695–3192–2, S. 152–159

[135] Pöhlsen, Stephan ; Schlichting, Stefan ; Strähle, Markus ; Franz, Frank ; Werner, Christian: A Concept for a Medical Device Plug-and-Play Architecture based on Web Services. In: *2nd Joint Workshop on High Confidence Medical Devices, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability.* San Francisco, CA, USA, April 2009

[136] Pöhlsen, Stephan ; Schlichting, Stefan ; Strähle, Markus ; Franz, Frank ; Werner, Christian: A Concept for a Medical Device Plug-and-Play Architecture based on Web Services. In: *SIGBED Review* 6 (2009), Juli, Nr. 2, S. 7. – ISSN 1551–3688

[137] Pöhlsen, Stephan ; Schlichting, Stefan ; Strähle, Markus ; Franz, Frank ; Werner, Christian: A DPWS-based Architecture for Medical Device Interoperability. In: *World Congress on Medical Physics and Biomedical Engineering (WC 2009)* Bd. 25. München, Germany, September 2009 (IFMBE Proceedings V). – ISBN 978–3–642–03903–4, S. 82–85

[138] Pöhlsen, Stephan ; Schlichting, Stefan ; Werner, Christian: A Comparison of DDS and Web Services for PoC Inter-Device Connectivity. Eingereicht bei: *4th International Conference on Pervasive Computing Technologies for Healthcare 2010*. München, März 2010

[139] Pöhlsen, Stephan ; Schlichting, Stefan ; Werner, Christian: Praktische Umsetzung einer sicheren Zugriffskontrolle für Web-Services am Beispiel medizinischer Geräte. Erscheint in: *PIK – Praxis der Informationsverarbeitung und Kommunikation* 33 (2010). – ISSN 0930–5157

[140] Pöhlsen, Stephan ; Werner, Christian: Robust Web Service Discovery in Large Networks. In: *Proceedings of the 2008 IEEE International Conference on Services Computing (SCC 2008)* Bd. 2. Honolulu, HI, USA : IEEE Computer Society, Juli 2008. – DOI 10.1109/SCC.2008.36. – ISBN 978–0–7695–3283–7–02, S. 521–524

[141] Postel, Jon (Hrsg.): *User Datagram Protocol*. RFC 768 (Standard). `http://tools.ietf.org/html/rfc768`. Version: August 1980

[142] Postel, Jon (Hrsg.): *Internet Protocol*. RFC 791 (Standard). `http://tools.ietf.org/html/rfc791`. Version: September 1981

[143] Postel, Jon (Hrsg.): *Transmission Control Protocol*. RFC 793 (Standard). `http://tools.ietf.org/html/rfc793`. Version: September 1981

[144] PrismTech: *Defense and Aerospace*. `http://www.opensplice.com/section-item.asp?snum=3&sid=351`, Abruf: 2009-10-27

[145] PrismTech: *Webseite*. `http://www.prismtech.com/`, Abruf: 2009-10-29

[146] PrismTech: *OpenSplice DDS Community Edition*. `http://www.opensplice.org/`. Version: 2009

[147] *PROFIBUS*. `http://www.profibus.com/technology/profibus/`, Abruf: 2009-10-15

[148] Ramakrishnan, Kadangode K. ; Floyd, Sally ; Black, David L.: *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168 (Proposed Standard). `http://tools.ietf.org/html/rfc3168`. Version: September 2001

[149] Real-Time Innovations (RTI): *Webseite*. `http://www.rti.com/`, Abruf: 2009-10-29

[150] Rekhter, Yakov ; Moskowitz, Robert G. ; Karrenberg, Daniel ; Groot, Geert J. ; Lear, Eliot: *Address Allocation for Private Internets*. RFC 1918 (Best Current Practice). `http://tools.ietf.org/html/rfc1918`. Version: Februar 1996

[151] Rescorla, Eric: *HTTP Over TLS*. RFC 2818 (Informational). `http://tools.ietf.org/html/rfc2818`. Version: Mai 2000

[152] Rescorla, Eric ; Modadugu, Nagendra: *Datagram Transport Layer Security*. RFC 4347 (Proposed Standard). http://tools.ietf.org/html/rfc4347. Version: April 2006

[153] RTI: *RTI's NDDS Publish-Subscribe Middleware Powers Navy Open Architecture Test Facility – Navy OA Technical Architecture recommends new Data Distribution Service for Real-Time Systems standard (DDS) from the Object Management Group (OMG)*. Version: Februar 2004. http://www.rti.com/company/news/NavyOA_Feb04.html, Abruf: 2009-10-27

[154] Schmidt, Douglas C. ; Parsons, Jeff: *Overview of the OMG Data Distribution Service*. Version: Dezember 2005. http://www.cs.wustl.edu/~schmidt/DDS.ppt, Abruf: 2009-10-29

[155] Schuymer, Bart D. (Hrsg.): *ebtables*. http://ebtables.sourceforge.net/, Abruf: 2009-10-20

[156] Snell, James ; Donoho, Andrew: *DNS Endpoint Discovery (DNS-EPD)*. Internet-Draft. http://www.watersprings.org/pub/id/draft-snell-dnsepd-01.txt. Version: November 2004. – Expired

[157] Stamp, Mark: *Information Security: Principles and Practice*. John Wiley & Sons, 2005. – ISBN 0–471–73848–4

[158] Strähle, Markus ; Ehlbeck, Mareike ; Prapavat, Viravuth ; Kück, Kai ; Franz, Frank ; Meyer, Jörg-Uwe: Towards a Service-Oriented Architecture for Interconnecting Medical Devices and Applications. In: *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPNP)*. Cambridge, Massachusetts, USA : IEEE Computer Society, Juni 2007. – DOI 10.1109/HCMDSS–MDPnP.2007.27. – ISBN 0–7695–3081–8, 153–155

[159] Sun Microsystems, Inc.: *Java Message Service Specification*. http://java.sun.com/products/jms/docs.html. Version: April 2002. – Version 1.1

[160] Thomson, Susan ; Narten, Thomas ; Jinmei, Tatuya: *IPv6 Stateless Address Autoconfiguration*. RFC 4862 (Draft Standard), September 2007

[161] UPnP Forum: *UPnP Device Architecture 1.0*. http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf. Version: Oktober 2008

[162] van Engelen, Robert A. ; Gallivan, Kyle A.: The gSOAP Toolkit for Web Services and Peer-to-Peer Computing Networks. In: *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*. Berlin, Germany : IEEE Computer Society, Mai 2002. – DOI 10.1109/CCGRID.2002.1017120. – ISBN 0–7695–1582–7, 128–135

[163] van Engelen, Robert A. ; Zhang, Wei: An Overview and Evaluation of Web Services Security Performance Optimizations. In: *Proceedings of the 2008 IEEE International Conference on Web Services (ICWS)*. Beijing, China : IEEE

Computer Society, September 2008. – DOI 10.1109/ICWS.2008.102. – ISBN 978–0–7695–3310–0, 137–144

[164] Varadarajan, Srinidhi ; Chiueh, Tzi-cker: EtheReal: A Host-Transparent Real-Time Fast Ethernet Switch. In: *6th International Conference on Network Protocols (ICNP '98)*. Austin, TX, USA : IEEE Computer Society, Oktober 1998. – DOI 10.1109/ICNP.1998.723721. – ISBN 0–8186–8988–9, S. 12–21

[165] Venkatramani, Chitra ; Chiueh, Tzi-cker: Design, Implementation, and Evaluation of a Software-based Real-Time Ethernet Protocol. In: *SIGCOMM Computer Communication Review* 25 (1995), Oktober, Nr. 4, S. 27–37. – DOI 10.1145/217391.217404. – ISSN 0146–4833

[166] Verband Deutscher Arztinformationssystemhersteller und Provider e. V. (VDAP): *VCS – der Standard für die elektronische Arzt-Arzt-Kommunikation*. http://www.vdap.de/?nav=89, Abruf: 2009-09-08

[167] W3C Candidate Recommendation ; Kavantzas, Nickolas (Hrsg.) ; Burdett, David (Hrsg.) ; Ritzinger, Gregory (Hrsg.) ; Fletcher, Tony (Hrsg.) ; Lafon, Yves (Hrsg.) ; Barreto, Charlton (Hrsg.): *Web Services Choreography Description Language Version 1.0*. http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/. Version: November 2005

[168] W3C Communications Team: *From SOAP/1.1 to SOAP Version 1.2 in 9 points*. http://www.w3.org/2003/06/soap11-soap12.html. Version: Juni 2003

[169] W3C Note: *Simple Object Access Protocol (SOAP) 1.1*. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/. Version: Mai 2000

[170] W3C Note: *SOAP Messages with Attachments*. http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211. Version: Dezember 2000

[171] W3C Note ; Christensen, Erik (Hrsg.) ; Curbera, Francisco (Hrsg.) ; Meredith, Greg (Hrsg.) ; Weerawarana, Sanjiva (Hrsg.): *Web Services Description Language (WSDL) 1.1*. http://www.w3.org/TR/2001/NOTE-wsdl-20010315. Version: März 2001

[172] W3C Recommendation ; Eastlake, Donald (Hrsg.) ; Reagle, Joseph (Hrsg.): *XML Encryption Syntax and Processing*. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/. Version: Dezember 2002

[173] W3C Recommendation: *XML Information Set (Second Edition)*. http://www.w3.org/TR/2004/REC-xml-infoset-20040204/. Version: Februar 2004

[174] W3C Recommendation: *XML Schema Part 1: Structures Second Edition*. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/. Version: Oktober 2004

[175] W3C Recommendation: *SOAP Message Transmission Optimization*

*Mechanism*. http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/. Version: Januar 2005

[176] W3C Recommendation: *XML-binary Optimized Packaging*. http://www.w3.org/TR/2005/REC-xop10-20050125/. Version: Januar 2005

[177] W3C Recommendation ; Gudgin, Martin (Hrsg.) ; Hadley, Marc (Hrsg.) ; Rogers, Tony (Hrsg.): *Web Services Addressing 1.0 – Core*. http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/. Version: Mai 2006

[178] W3C Recommendation: *SOAP Version 1.2 Part 0: Primer (Second Edition)*. http://www.w3.org/TR/2007/REC-soap12-part0-20070427/. Version: April 2007

[179] W3C Recommendation: *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. http://www.w3.org/TR/2007/REC-soap12-part1-20070427/. Version: April 2007

[180] W3C Recommendation: *SOAP Version 1.2 Part 2: Adjuncts (Second Edition)*. http://www.w3.org/TR/2007/REC-soap12-part2-20070427/. Version: April 2007

[181] W3C Recommendation: *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. http://www.w3.org/TR/2007/REC-wsdl20-20070626/. Version: Juni 2007

[182] W3C Recommendation: *Web Services Policy 1.5 – Framework*. http://www.w3.org/TR/2007/REC-ws-policy-20070904/. Version: September 2007

[183] W3C Recommendation ; Bray, Tim (Hrsg.) ; Paoli, Jean (Hrsg.) ; Sperberg-McQueen, C. M. (Hrsg.) ; Maler, Eve (Hrsg.) ; Yergeau, François (Hrsg.): *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. http://www.w3.org/TR/2008/REC-xml-20081126/. Version: November 2008

[184] W3C Recommendation: *XML Signature Syntax and Processing (Second Edition)*. http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/. Version: Juni 2008

[185] W3C Working Draft: *Web Services Enumeration (WS-Enumeration)*. http://www.w3.org/TR/2009/WD-ws-enumeration-20090625/. Version: Juni 2009

[186] W3C Working Draft: *Web Services Eventing (WS-Eventing)*. http://www.w3.org/TR/2009/WD-ws-eventing-20090625/. Version: Juni 2009

[187] W3C Working Draft: *Web Services Metadata Exchange (WS-MetadataExchange)*. http://www.w3.org/TR/2009/WD-ws-metadata-exchange-20090625/. Version: Juni 2009

[188] W3C Working Draft: *Web Services Resource Transfer (WS-RT)*. http://www.w3.org/TR/2009/WD-ws-resource-transfer-20090625/. Version: Juni 2009

[189] W3C Working Draft: *Web Services Transfer (WS-Transfer)*. http://www.w3.org/TR/2009/WD-ws-transfer-20090625/. Version: Juni 2009

[190] W3C Working Group Note ; Booth, David (Hrsg.) ; Haas, Hugo (Hrsg.) ; McCabe, Francis (Hrsg.) ; Newcomer, Eric (Hrsg.) ; Champion, Michael (Hrsg.) ; Ferris, Chris (Hrsg.) ; Orchard, David (Hrsg.): *Web Services Architecture*. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/. Version: Februar 2004

[191] W3C Working Group Note: *Web Services Description Language (WSDL) Version 2.0: Additional MEPs*. http://www.w3.org/TR/2007/NOTE-wsdl20-additional-meps-20070626/. Version: Juni 2007

[192] Web Services Interoperability Organization (WS-I): http://www.ws-i.org/

[193] World Wide Web Consortium (W3C): http://www.w3.org/

[194] WS-I Final Material ; Ballinger, Keith (Hrsg.) ; Ehnebuske, David (Hrsg.) ; Gudgin, Martin (Hrsg.) ; Nottingham, Mark (Hrsg.) ; Yendluri, Prasad (Hrsg.): *Basic Profile Version 1.0*. http://www.ws-i.org/Profiles/BasicProfile-1.0.html. Version: April 2004

[195] WS-I Final Material: *Simple SOAP Binding Profile Version 1.0*. http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html. Version: August 2004

[196] WS-I Final Material: *Attachments Profile Version 1.0*. http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html. Version: April 2006

[197] WS-I Final Material: *Basic Profile Version 1.1*. http://www.ws-i.org/Profiles/BasicProfile-1.1.html. Version: April 2006

[198] WS-I Final Material ; McIntosh, Michael (Hrsg.) ; Gudgin, Martin (Hrsg.) ; Morrison, K. S. (Hrsg.) ; Barbir, Abbie (Hrsg.): *Basic Security Profile Version 1.0*. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html. Version: März 2007

[199] WS-I Working Group Approval Draft: *Basic Profile Version 1.2*. http://www.ws-i.org/Profiles/BasicProfile-1_2(WGAD).html. Version: Oktober 2007

[200] WS-I Working Group Approval Draft ; McIntosh, Michael (Hrsg.) ; Gudgin, Martin (Hrsg.) ; Morrison, K. S. (Hrsg.) ; Barbir, Abbie (Hrsg.): *Basic Security Profile Version 1.1*. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.1-2007-02-20.html. Version: Februar 2007

[201] WS-I Working Group Draft: *Basic Profile Version 2.0*. http://www.ws-i.org/Profiles/BasicProfile-2_0(WGD).html. Version: Oktober 2007

[202] WS-I Working Group Draft ; Durand, Jacques (Hrsg.) ; Karmarkar, Anish (Hrsg.) ; Pilz, Gilbert (Hrsg.): *Reliable Secure Profile Version 1.0*. http://

`www.ws-i.org/Profiles/ReliableSecureProfile-1.0.html`. Version: Mai 2008. – revision 16

[203] Xiong, Ming ; Parsons, Jeff ; Edmondson, James ; Nguyen, Hieu ; Schmidt, Douglas C.: Evaluating Technologies for Tactical Information Management in Net-Centric Systems. In: *Defense Transformation and Net-Centric Systems 2007* Bd. 6578. Orlando, FL, USA : SPIE, April 2007 (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series 1). – DOI 10.1117/12.719679. – ISSN 0277–786X, S. 65780A

[204] Zeeb, Elmar ; Bobek, Andreas ; Bohn, Hendrik ; Prüter, Steffen ; Pohl, Andre ; Krumm, Heiko ; Lück, Ingo ; Golatowski, Frank ; Timmermann, Dirk: WS4D: SOA-Toolkits making embedded systems ready for Web Services. In: *Proceedings on the Second International Workshop on OSSPL07: Open Source Software and Product Lines 2007*. Limerick, Ireland, Juni 2007, S. 33–42

[205] Zhang, Liang-Jie ; Zhou, Qun ; Chao, Tian: A Dynamic Services Discovery Framework for Traversing Web Services Representation Chain. In: *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*. San Diego, California, USA : IEEE Computer Society, Juli 2004. – DOI 10.1109/ICWS.2004.5. – ISBN 0–7695–2167–3, S. 632–639

# Index