ComConsult Research

Der Netzwerk Insider

Januar 2017

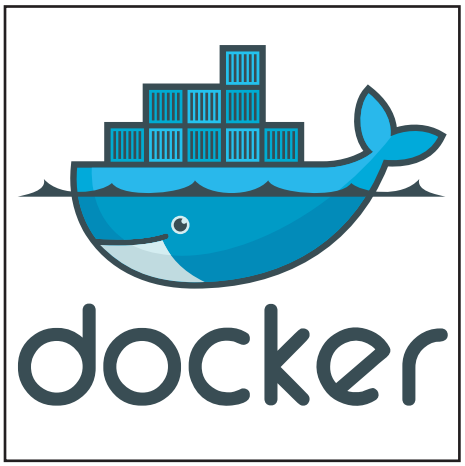
Systematische Weiterbildung für Netzwerk- und IT-Professionals

Data Center

Container-Networking

von Markus Schaub

Schon die Bezeichnung „Container“ legt einen Vergleich mit den 12,192 m langen, 2,438 m breiten und 2,591 m hohen ISO 668 Containern nahe. Dabei würde sich ein Kuchen als Analogie viel eher eignen. Warum? Dazu später mehr. Aber auch diese Analogie hat natürlich ihre Grenzen. Weder aus der einen noch aus der anderen lässt sich ableiten, was Container eigentlich sind, wofür man sie nutzt und warum sie nicht nur zunehmend auf die Entwicklung und den Betrieb von Software Auswirkungen haben, sondern auch für Netzwerke nicht ohne Konsequenzen sind.



Die Kernidee

Um die Idee von Containern zu verstehen, zäumt man das Pferd am besten von hinten auf und betrachtet das fertige Produkt, den Container selbst. Diese werden gerne mit virtuellen Maschinen verglichen. Abbildung 1 zeigt den strukturellen Aufbau von Containern und stellt diesen dem der virtuellen Maschinen gegenüber. Im Großen und Ganzen scheinen beide sehr ähnlich zu sein, da sie bis auf eine Schicht identisch sind: dem Guest-OS, also dem Betriebssystem, das in einer VM läuft. Dieses fehlt bei Containern.

weiter auf Seite 7

Wireless LAN

Was bringt der neue WLAN-Standard?

von Dr. Joachim Wetzlar

Kurz vor Weihnachten hat das IEEE den neuen WLAN-Standard IEEE 802.11-2016 veröffentlicht. Was? Schon wieder neue WLAN-Techniken? Gehören damit unsere WLAN-Komponenten, die teilweise noch aus dem letzten Jahrzehnt stammen, endgültig aufs Abstellgleis, sprich in den Elektroschrott? Keine Sorge: Der genannte Standard bringt eigentlich nichts Neues. Und dennoch gibt es neue Ideen und Entwicklungen im WLAN-Umfeld, die ich in diesem Artikel etwas beleuchten möchte.

Die letzte Ausgabe des WLAN-Standards – IEEE 802.11-2012 – war schon gewaltig: Insgesamt knapp 2.800 Seiten. Selbst auf Bibeldruckpapier wäre das noch ziemlich unhandlich. Immerhin kann man sich das Dokument als PDF beim IEEE herunterladen. Die Regel ist, dass ca. ein halbes Jahr nach der Veröffentlichung der Download kostenlos wird. Hierzu hat das IEEE das „Get Program“ eingerichtet, dessen Website sich unter dem Link **http://standards.ieee.org/about/get/** aufrufen lässt.

weiter auf Seite 17

Geleit

RZ kontra Cloud:

vergessen Sie IaaS, vergessen Sie Kosten, es geht um die Zukunft unserer Anwendungen:

naht das Ende von Client-Server?

auf Seite 2

Standpunkt

WLAN Clients senden weiter als man denkt!

auf Seite 15

Aktueller Kongress

ComConsult Netzwerk-Forum 2017

ab Seite 4

Aktuelles Seminar

Leistungsfähige, skalierbare, hochverfügbare, sichere und wirtschaftliche Speicherlösungen

auf Seite 16

Geleit

RZ kontra Cloud:

vergessen Sie IaaS, vergessen Sie Kosten, es geht um die Zukunft unserer Anwendungen:

naht das Ende von Client-Server?

ComConsult Research hat eine aktuelle Untersuchung abgeschlossen, deren Ergebnisse exklusiv auf dem Netzwerk Forum 2017 diskutiert wird. Dabei geht es um die Frage, warum und wie auch große Unternehmen Teile ihrer IT in die Cloud verlagern. Die Untersuchung hat wieder einmal unterstrichen, dass speziell bei der Diskussion „Cloud kontra Rechenzentrum“ einige zentrale Fakten nicht übersehen werden dürfen. Gleichzeitig wurden aber auch zentrale IT-Architekturen der Vergangenheit in Frage gestellt.

Bei der Diskussion der Cloud, speziell hier der Aspekt PaaS, darf nie übersehen werden, dass wir hier über einen klar definierten Typ von Anwendung sprechen:

- Es geht in der Regel um Web-Applikationen
- Mikro-Service-Architekturen lösen dabei monolithische Anwendungs-Architekturen ab
- Automatische Skalierung (Elastizität) spielt dabei eine große Rolle (siehe AWS Lambda)

Diese Rahmenparameter sind die Basis für eine Reihe von Folgeproblemen. Der Übergang zu Mikro-Services ist in der Anwendungsentwicklung der Status-Quo, da er die Voraussetzung für eine deutlich agilere Pflege von Anwendungen ist und die Ausfallzeiten durch Wartung pro Jahr dramatisch reduziert. Kombiniert man dies mit speziellen Diensten der Cloud-Plattformen wie zum Beispiel die automatische Migration auf neue Versionen, dann erreichen auch große Applikations-Betreiber heute Ausfallzeiten, die nur noch im Bereich weniger Minuten pro Jahr liegen. Leider haben Mikro-Services aber auch Nachteile. Einer der Kernnachteile liegt in der Lizenzierung. Die Anbieter von Software müssen bereit sein, sich auf diese Art von Architektur mit ihren Preis-Modellen einzulassen. Das ist nicht immer der Fall und im Zweifel erfordert dies den Wechsel des Software-Produkts. Gleiches gilt für die Eignung von Datenbanken in diesem Umfeld. Es gibt sicher Gründe warum dominante Anbieter wie Oracle in den Cloud-Datenbanken zum Beispiel von Amazon eine Gefahr sehen.

Damit ist aber auch klar, dass bei der Diskussion der Cloud eine 1:1 Migration bestehender Altanwendungen in der Regel ausgeschlossen werden kann. Unsere Untersuchung hat gezeigt, dass selbst im Fall von neuen Anwendungen, die bereits Mikroservice-Architekturen haben, erhebliche Anpassungen an die Cloud-Plattform erforderlich sind (diese sind aber bei bereits vorliegenden Mikroservice-Architekturen in der Regel schnell umsetzbar).

In unserer Untersuchung hat es damit auch eine sehr naheliegende Diskussion gegeben, die wir auch auf ComConsult Netzwerk Forum 2017 vertiefen wollen: wo stehen Client-Server-Architekturen. Die Aufteilung von Anwendungen in einen Client- und einen Server-Teil erfolgte historisch, um die Server zu entlasten und speziell x86-Server einer breiteren Nutzung zuzuführen. Diese Art von Architektur war dann auch über die letzten 15 Jahre gesehen sehr erfolgreich. Sie hat naturgemäß einen gravierenden Nachteil: sie erfordert die Installation und Pflege eines Clients. Dies kann je nach Gast-Betriebssystem, Typ des Clients und Art der Kommunikation zum Server zu erheblichen Aufwendungen im Betrieb führen. So kann man klar feststellen, dass die Betriebskosten von Client-Server-Lösungen deutlich höher sind als die Betriebskosten zentralistischer Lösungen (speziell wenn man wirklich alle Aufwendungen auf der Client-Seite erfasst. Sie kennen die typischen Probleme: neue Windows-Version erfordert zuerst den umfangreichen Test auf Kompatibilität, dann gibt es Widersprüche im Bedarf verschiedenerer Clients, dann gibt es Probleme mit Treibern usw).

Damit kommen wir zu einer ganz zentralen These: Client-Server-Architekturen sind tot. Sie sind nicht mehr erforderlich, da wir durch Mikroservice-Architekturen und Parallelisieren auf der Server-Seite eine quasi unendliche Kapazität haben. Das mag nicht für alle Anwendungen gelten, es gibt sicher Ausnahmen, aber es gilt für mehr als 90% aller Anwendungen.

Logischerweise wird man dann auch gar nicht erst versuchen eine Client-Server-Anwendung in die Cloud zu verlagern. Auch wären die Verluste durch die Verzögerung in der Kommunikation zwischen Client und Server so hoch, dass mit hoher Wahrscheinlichkeit keine akzeptable Anwendungs-Performance erreicht werden kann.

Damit ist aber auch klar: die Diskussion über die Cloud setzt an der völlig falschen Stelle an. In Wirklichkeit geht es um die Anwendungs-Architektur der Zukunft. Es geht um die komplette Ablösung aller bestehenden Alt-Anwendungen durch moderne Software-Architekturen. Die Frage, ob diese dann in der Cloud laufen oder nicht ist dabei gar nicht so spannend wie diese zentrale Frage.

Unabhängig von bestehenden Anwendungen gibt es dabei einen zunehmenden Bedarf für neue Anwendungen, die spezielle Dienste für Kunden bereitstellen. Zum Beispiel wird die Zukunft der Banken und Versicherungen durch diese neuen Anwendungen geprägt sein. Die Zeit von Internet-Banking aus der Steinzeit nähert sich dem Ende. Wir sprechen in Zukunft über hochintelligente Finanz-Assistenten, die die gesamte Spannbreite von Kreditaufnahme bis hin zur Anlage von Geld abdecken werden. Dies wird eine sehr dynamische und agile Entwicklung voraussetzen. Die große

Schwerpunktthema

Container-Networking

Fortsetzung von Seite 1

Markus Schaub ist seit 2009 Leiter von ComConsult-Study.tv. Er verfügt über umfangreiche Berufserfahrung in den Bereichen Netzwerken und VoIP und ist seit mehr als 13 Jahren bei ComConsult beschäftigt. Seine Schwerpunkte liegen im Netzwerk-Design, IP-Infrastrukturdiensten und SIP, zu denen er viele Vorträge auf Kongressen hielt, erfolgreich Seminare durchführte und zahlreiche Veröffentlichungen schrieb.

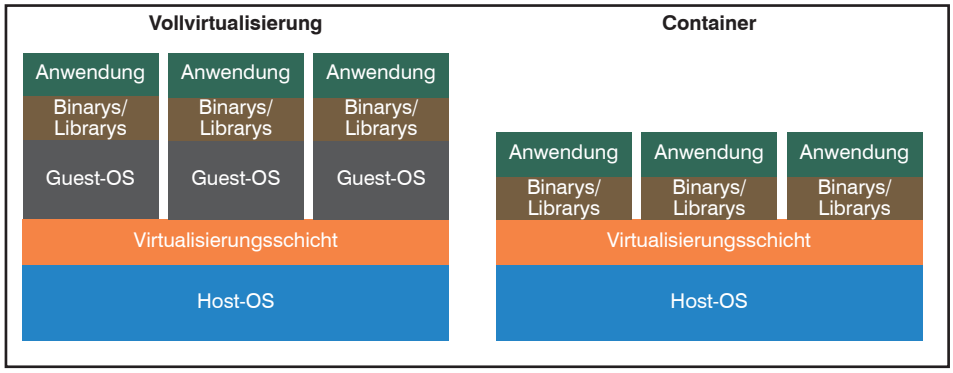


Abbildung 1: Vergleich virtuelle Maschine und Container

Viele betrachten Container deshalb als eine Art von lightweighted VMs, abgespeckten Virtuellen Maschinen. Das ist allerdings der falsche Ansatz. Denn dann stößt man auf eine ganze Reihe von unerträglichen Unzulänglichkeiten, wie die Abhängigkeit vom Betriebssystem des Hosts, die fehlenden virtuellen Netzwerkkarten, ein Fehlen von Funktionen wie vMotion und vieles mehr. Container erscheinen bei diesem Vergleich wie ein Rückschritt von einer ausgereiften Technologie zurück zu Virtualisierung aus der Steinzeit.

Man versteht Container besser, wenn man sie nicht als eine Form abgespeckter VMs betrachtet, sondern als eine neue Art der Softwareverteilung. Musste man bislang immer darauf achten, ob ein System die benötigte Umgebung für eine Anwendung bereitstellt, so bringen Container ihre Wohlfühlumgebung bereits mit. Da Container in sich gekapselt sind, braucht man auch bei der Installation nicht darauf achten, ob ggf. andere Anwendungen auf einem System laufen, die inkompatible Libraries benötigen.

Um ein Beispiel zu bringen: ein Entwickler programmiert ein neues Modul für eine Webseite. Da das Modul von Grund auf neu entwickelt wird, setzt er dabei auf PHP 7.0. Allerdings werden auch andere Module von der Webanwendung genutzt, die älteren Datums sind und deshalb maximal mit PHP 5.x zurechtkommen. Nach der Entwicklung muss also ein Web-Server gesucht werden, der bereits PHP 7.0 hat. Diese Information muss zwischen Entwicklern und Betreibern ausgetauscht und sauber dokumentiert werden.

Bei Containern ist das nicht nötig: jeder Container bringt seine Abhängigkeiten mit und läuft in seiner eigenen Laufzeitumgebung. Abbildung 2 zeigt schematisch das Innenleben eines Containers.

Laufen mehrere Container auf einem Host-System, so sind sie – und hier klappt der Vergleich mit Virtuellen Ma-

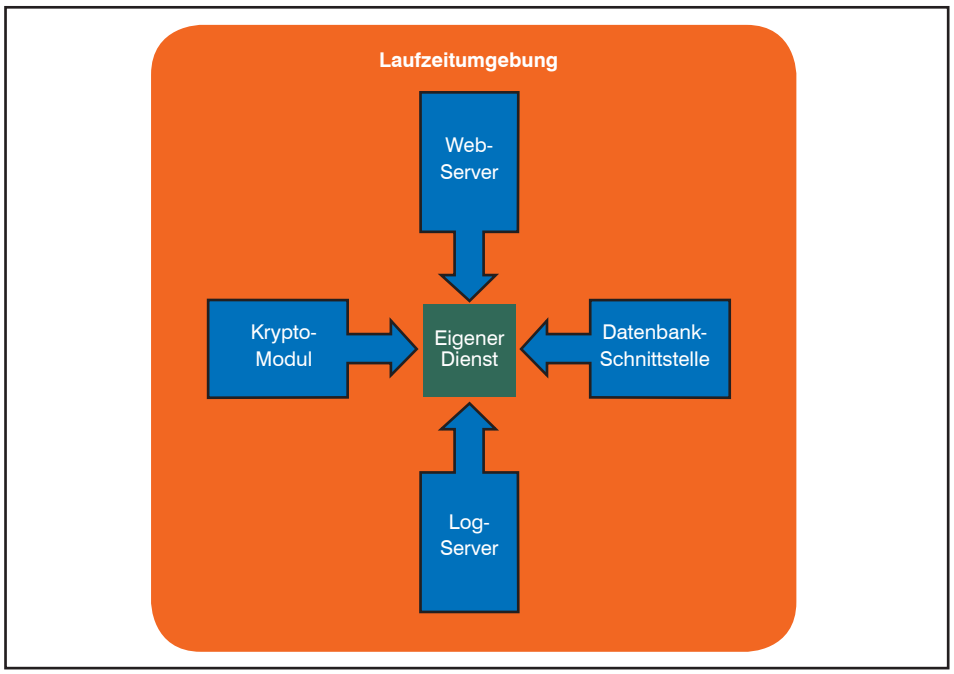


Abbildung 2: Innenleben eines Containers

schinen – gegeneinander abgeschirmt. D.h. im obigen Beispiel kann ein Container als Library PHP 5.x ein anderer PHP 7.x als Library mitbringen. Innerhalb des Containers können auch beide Webserver Port 80 nutzen. Problematisch wird letzteres erst, wenn man Ports für die externe Kommunikation, also außerhalb der Laufzeitumgebung des Containers freigeben möchte. Dazu später mehr.
Halten wir fest:

Eine Kernkomponente eines Containers ist es, einen Dienst inkl. seiner Abhängigkeiten und seiner Laufzeitumgebung isoliert zu betreiben.

Das alleine ist weder sonderlich neu noch ein Alleinstellungsmerkmal dieser vergleichsweise neuen Technologie. Es muss also noch mehr dahinter stecken. Dazu reicht es nicht einen laufenden Container zu betrachten, sondern vielmehr muss man verstehen, wie Container gebaut, verteilt und betrieben werden. Damit wären wir bei der Analogie des Kuchens.

Betrachtet man den Container als fertigen Kuchen, fehlen einige Dinge, die man benötigt, einen Kuchen zu backen: das Rezept, die Zutaten und die Geschäfte, wo man letztere erwerben kann.
Wenn man in dem Bild bleibt, so gibt es das alles auch für Container:

- Das Rezept ist die Anleitung, wie ein Container gebaut werden soll. Dieses Rezept heißt bei Docker bspw. Dockerfile.

- Die Zutaten sind die Libraries, Programme und Dienste, die alle zusammengefasst den Teig ergeben. Diesen Teig könnte man als Image bezeichnen. Der Teig muss noch gebacken werden, dem Image fehlt noch die Laufzeitumgebung.

- Die Geschäfte heißen Hub, dort kann man die notwendigen Zutaten beziehen, die im Rezept stehen.

- Fügt man dem Teig noch Hitze hinzu, hat man den Kuchen. Analog: das Image zzgl. einer Laufzeitumgebung ergibt einen Container.

Schauen wir uns die einzelnen Schritte im Detail an:

Der Dockerfile

Der Dockerfile ist eine Textdatei, die in der Programmiersprache Go geschrieben wird.

Der Dockerfile ist eine Anleitung, das Rezept, wie ein Image erstellt werden soll. Das Image, der Teig, ist die Vorstufe zum Container, zum Image später mehr.

Abbildung 3 zeigt beispielhaft einen solchen Dockerfile. Images können auf anderen Images beruhen, so wie man Fertigprodukte im Supermarkt kaufen kann, bspw. Blätterteig. Im Beispiel wird ein MySQL Server aufgesetzt. Die Basis „FROM“ ist eine bereits existente Ubuntu-Umgebung. Wichtig ist, nicht Ubuntu als komplettes Betriebssystem, sondern die Standard-Ubuntu-Umgebung ohne Kernel, denn der wird ja später vom Hosts-System bereitgestellt.

Als nächstes wird derjenige angegeben, der sich um dieses Image kümmert, der MAINTAINER.

Dann folgen eine Reihe von Befehlen, die die MySQL-Umgebung so aufsetzen wie der Maintainer es braucht: mysql wird installiert, konfiguriert, User werden angelegt, der Speicherort wird festgelegt, ein TCP-Port für die externe Kommunikation wird auf dem Host-System geöffnet etc.

```
FROM ubuntu:latest
MAINTAINER Markus Schaub

# Get MySQLRUN apt-get update
RUN apt-get install -y mysql-server-5.5

# Open MySQL to a world
RUN sed -i -e"s/^bind-addresss*=s*127.0.0.1/bind-address = 0.0.0.0/" /etc/mysql/my.cnf
# Script to pass MySQL
ADD startup.sh /
RUN chmod 755 /*.sh
#define access data
ENV DB_USER tagesschaub
ENV DB_PASSWORD shadow
ENV DB_NAME wp
ENV VOLUME_HOME "/var/lib/mysql"
# expose MySQL port to a network
EXPOSE 3306
# open to mount
VOLUME ["/var/lib/mysql", "/var/log/mysql"]
# start container
CMD ["/bin/bash", "/startup.sh"]
```

Abbildung 3: Beispiel eines Dockerfiles

Wer schon mal auf einem Linux-System Programme installiert hat, findet sich schnell zurecht, denn die Befehle entsprechen denen, die zu dem Linux-System gehören, hier eben Ubuntu. Denen werden nur entsprechende Go-Befehle vorangestellt. Hinzu kommen noch einige leicht zu erlernende Befehle, für die Definition der Schnittstellen zwischen dem Host und dem späteren Container, wie PORT (offener TCP-Port) oder VOLUME (Speicherplatz außerhalb der Laufzeitumgebung).

Hat man den Dockerfile fertig geschrieben, kann man das Image erstellen.

Das Image

Mittels des „build“ Befehles wird das Image erzeugt, indem die Befehle im File abgearbeitet werden. Im Beispiel aus Abbildung 3 geht die Containersoftware, bspw. Docker, nun hin und lädt zunächst vom Hub das aktuelle („latest“) Ubuntu-Image herunter. Danach werden die restlichen Befehle ausgeführt, also MySQL wird geladen und im Image installiert, die User erzeugt, etc.

In der Kuchenanalogie: es wird eingekauft und der Teig wird angesetzt. Und so ist das Resultat des Build-Prozesses auch kein Kuchen, also kein Container. Vielmehr wird eine Datei erzeugt, in der nun alles vorhanden ist, was der Container später benötigt, außer der Laufzeitumgebung, also beispielsweise temporären Verzeichnissen und natürlich dem Kernel selbst.

Der Container

Um nun vom Teig zum Kuchen zu kommen, muss man ihn noch backen. Das geschieht mit dem „run“ Befehl. Damit startet man ein Image und es wird zum Container. D.h. die Laufzeitumgebung wird erzeugt, und, wenn gewünscht, werden Schnittstellen bereitgestellt. Typisch sind Netzwerkschnittstellen und Speicher außerhalb des Containers.

Letzteres ist wichtig, denn wenn man einen Container löscht, so verschwindet auch seine Laufzeitumgebung und damit alle Daten, die nur im Container vorliegen. Hat man jedoch ein Programm „containerisiert“, das Daten erzeugt, die auch außerhalb des Containers verfügbar sein sollen, so muss man diese irgendwohin schreiben. Oder aber der Dienst im Container soll Dateien bearbeiten, die bereits vorhanden sind, dann muss dieser Speicherort natürlich gemountet werden.

WLAN

Was bringt der neue WLAN-Standard?

Fortsetzung von Seite 1

Dr.-Ing. Joachim Wetzlar ist seit mehr denn 20 Jahren Senior Consultant der ComConsult Beratung und Planung GmbH und leitet dort das Competence Center „Data Center“. Er verfügt über einen erheblichen Erfahrungsschatz im praktischen Umgang mit Netzkomponenten und Serversystemen. Seine tiefen Detailkenntnisse der Kommunikations-Protokolle und entsprechender Messtechnik haben ihn in den zurückliegenden Jahren zahlreiche komplexe Fehlersituationen erfolgreich lösen lassen. Neben seiner Tätigkeit als Trouble-Shooter führt Herr Dr. Wetzlar als Projektleiter und Senior Consultant regelmäßig Netzwerk- WLAN- und RZ-Redesigns durch. Besucher von Seminaren und Kongressen schätzen ihn als kompetenten und lebendigen Referenten mit hohem Praxisbezug.

Der neue Standard wurde am 14. Dezember 2016 veröffentlicht. Er umfasst nun 3.237 Seiten. Glaubt man der Inhaltsangabe, wurden verschiedene Berichtigungen eingebracht und einiges klarer dargestellt. Es wurden verschiedene nicht näher spezifizierte Erweiterungen des Medienzugangsverfahrens (MAC Layer) und der Übertragungsverfahren (PHY Layer) eingebracht. Insbesondere aber hat man die „Amendments“ 1 bis 5 in den Standard eingearbeitet.

Was sind „Amendments“? Zu Deutsch Änderungen oder Berichtigungen. In der Tat werden viele der Neuerungen, die von den Arbeitsgruppen des IEEE – den Task Groups – eingebracht werden, als Änderungen des bestehenden Standards formuliert. Die entsprechenden Dokumente enthalten zusätzliche Kapitel und darüber hinaus Änderungen an bestehenden Abschnitten. Diese Änderungen werden tatsächlich so dargestellt, als hätte man im Textprogramm die Überarbeitungsmarkierungen aktiviert. Man findet durchgestrichene Passagen und hinzugefügte.

Somit ist also das Zusammenstellen eines neuen IEEE-Standards eher eine redaktionelle Tätigkeit als eine inhaltliche. Die folgenden fünf Amendments aus den Jahren 2012 und 2013 wurden jetzt eingearbeitet:

- Amendment 1: Prioritization of Management Frames (IEEE 802.11ae) stellt sicher, dass die Anmeldung am WLAN, das Handover und weitere Vorgänge mit höherer Betriebssicherheit als bisher erfolgen können. Hierfür wird Quality of Service (QoS) für die entsprechenden Management Frames eingeführt.

- Amendment 2: MAC Enhancements for Robust Audio Video Streaming (IEEE 802.11aa). Dieses Dokument führt spezielle Multicast-Modi ein, um die Übertragung von Videos an mobile Endgeräte zu verbessern. Ein Element ist der „Trick“, Multicasts mehrfach auszusenden, um die Wahrscheinlichkeit für einen fehlerfreien Empfang zu erhöhen. Hierüber haben wir im Januar 2015 an dieser Stelle berichtet.

- Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band (IEEE 802.11ad). Hierbei handelt es sich um das WLAN im Millimeterwellenbereich, über das wir an dieser Stelle bereits mehrfach berichtet haben.

- Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz (IEEE 802.11ac). Damit ist die inzwischen allseits bekannte und verfügbare Technik der hohen Bitraten im 5-GHz-Band gemeint.

- Amendment 5: Television White Spaces (TVWS) Operation (IEEE 802.11af). Ähnlich wie Wi-Fi HaLow (IEEE 802.11ah) handelt es sich um eine Technik für Frequenzen unterhalb von 1 GHz. Es werden schmalere Bandbreiten und entsprechend geringere Bitraten spezifiziert.

Aus Sicht eines Betreibers eines Enterprise WLAN bringt der Standard also tatsächlich nichts neues. Insbesondere Very High Throughput (VHT) von 11ac ist ja bereits verfügbar. Und der Rest ist nicht wirklich relevant. Viel interessanter ist, an welchen Amendments das IEEE derzeit arbeitet. In der Folge möchte ich auf die Arbeit der Task Groups 11ax und 11ay eingehen. Und ich wage einen Ausblick auf die Beziehung zwischen WLAN und Mobilfunk.

IEEE 802.11ax: High Efficiency WLAN (HEW)

HEW wird vom IEEE als der Nachfolger für IEEE 802.11n und 11ac gesehen. Man erwartet, dass der Standard (bzw. das Amendment) in 2019 veröffentlicht wird. Derzeit arbeitet man an der ersten Version, dem Draft 1.0, der leider noch nicht verfügbar ist. Daher habe ich verschiedene andere Quellen zu Rate gezogen, insbesondere Artikel von Personen bzw. Unternehmen, die in irgendeiner Form an der Entwicklung des Standards beteiligt sind. Auf Basis dieser Informationen lässt sich inzwischen ein ziemlich klares Bild der neuen Techniken zeichnen, die mit HEW ins WLAN einziehen werden.

„Efficiency“, also Effizienz ist es, was die Entwickler des Standards erreichen wollen. Effizienz bedeutet, dass vor allem aus der Sicht des Anwenders die Effizienz größer werden soll. Sie können sich vorstellen, dass Bitrate alleine wahrscheinlich nicht der Schlüssel zum Erfolg ist. Darüber hinaus wissen Sie, dass die Möglichkeiten mit dem VHT WLAN (IEEE 802.11ac) bereits ziemlich weit ausgereizt wurden. In früheren Artikeln habe ich gezeigt, dass man mit VHT nicht besonders weit kommt. Ein Access Point pro Büro ist die Richtschnur.

Und nun stellen Sie sich ein Fußballstadion mit einer fünfstelligen Anzahl von Zuschauern vor! Fast alle haben ein Smartphone und möchten darüber während des Spiels online und in Echtzeit informiert werden. Wofür man das braucht, kann ich mir nicht wirklich vorstellen, aber das tut hier nichts zur Sache. Jedenfalls muss das WLAN dergestalt sein, dass der einzelne Anwender eine brauchbare Reaktionszeit beim Zugriff auf Inhalte erlebt.

Ein wesentliches Problem bei allen bisherigen WLAN-Varianten ist, dass Stationen sich das Medium „Luft“ teilen. Es muss also nacheinander gesendet werden. Im besagten Stadion, wo Anwender so dicht stehen, dass sie sich gegenseitig berühren, bleibt für den einzelnen nicht viel Bi-

trate übrig. Eigentlich stören sich alle Endgeräte und die Access Points nur gegenseitig. Man braucht also Techniken, um mehr Stationen ungestört parallel nebeneinander betreiben zu können. 3 Kanäle im 2,4-GHz-Band und 16 auf 5 GHz reichen dazu ganz sicher nicht.

Multi-User MIMO

Erste Ansätze für Parallelisierung gab es bereits im VHT WLAN. Das Schlüsselwort lautet „Multi-User". Multi-User MIMO ermöglicht es einem Access Point (auf wundersame Weise) gleichzeitig Daten an mehrere Stationen zu senden. Ein Access Point, der MIMO mit beispielsweise vier Spatial Streams unterstützt, kann diese vier Streams auf zwei Stationen aufteilen. Darüber hinaus kann er die Streams in unterschiedliche Richtungen aussenden, um jede der Stationen optimal zu erreichen. IEEE 802.11ac nennt das „Downlink Multi-User MIMO (DL-MU-MIMO) Beamforming". Das Verfahren wird bekanntlich ab Produkten mit 11ac „Wave 2" unterstützt.

Grundsätzlich ist das eine gute Idee. Denn wahrscheinlich unterstützen die wenigsten WLAN-Stationen so viele Spatial Streams wie der Access Point. Das gilt vor allem im Stadion. Smartphones bieten zum einen zu wenig Platz, um viele Antennen unterbringen zu können. Zum anderen sparen weniger Sender und geringere Bitraten Strom, denn Batteriekapazität ist bei Smartphones ein teures Gut.

Der Downlink, also die Richtung vom Access Point zur Station, ist natürlich nur die „halbe Miete". IEEE 802.11ax wird zusätzlich MIMO für den Uplink ermöglichen. Damit die Stationen gleichzeitig senden können, müssen sie vom Access Point koordiniert werden. Zu diesem Zweck sendet der Access Point so genannte Trigger Frames an die Stationen, die daraufhin mit den jeweiligen Daten antworten (vgl. Abbildung 1).

Modifiziertes OFDM

Bekanntlich wird bereits seit fast 20 Jahren im WLAN das Verfahren „Orthogonal Frequency-Division Multiplexing" (OFDM) eingesetzt. Hierbei werden die Bits nicht nacheinander auf einen einzigen Träger aufmoduliert sondern das Signal besteht aus mehreren „Unterträgern", die parallel mit Bits moduliert werden. Das Verfahren entspricht im Prinzip der Parallelschnittstelle, die Sie von Ihren alten PCs kennen (der 25polige Subminiatur-D-Stecker). Dort gab es 8 Datenleitungen, über die sich gleichzeitig ein ganzes Byte übertragen ließ.

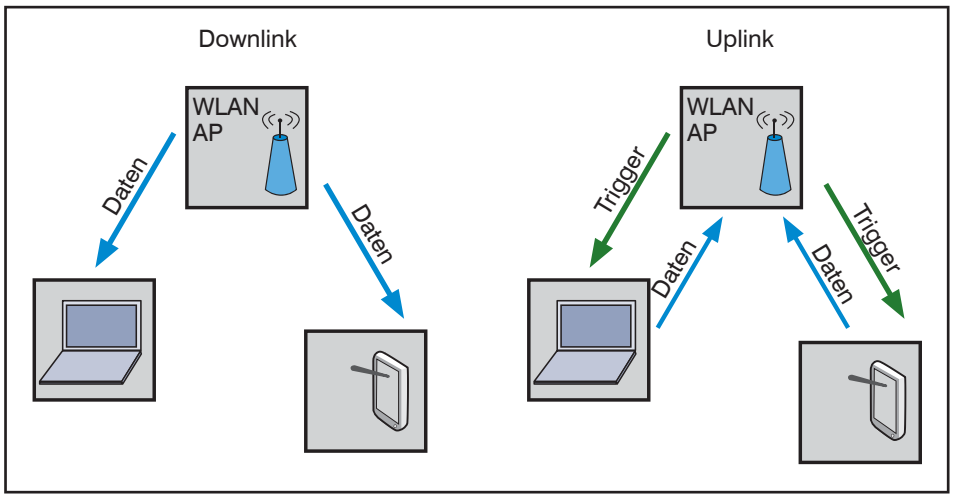


Abbildung 1: MU-MIMO im Downlink und Uplink

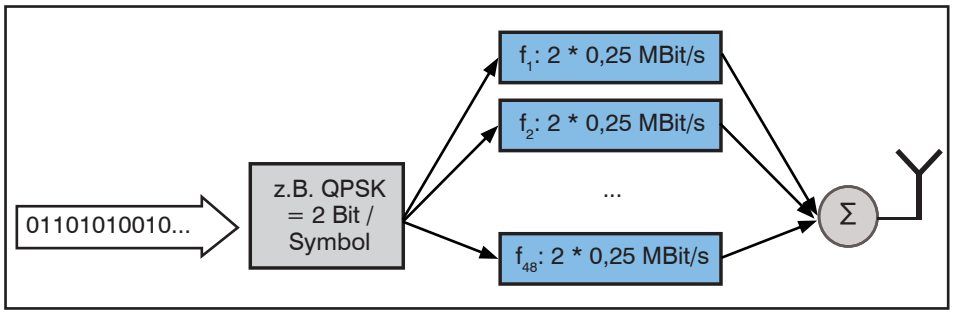


Abbildung 2: Zum Prinzip von OFDM

Bereits in IEEE 802.11a wurde OFDM mit 48 Unterträgern realisiert. Die Gruppe aus Bits, die sich mittels OFDM gleichzeitig übertragen lässt, nennt man „Symbol". Bei 11a besteht ein Symbol somit aus Vielfachen von 48 Bits, je nach Modulation des Unterträgers. Abbildung 2 illustriert dieses Konzept. Setzt man eine zweiwertige Modulation ein (binäre Phasenmodulation, BPSK), sind es genau 48. Setzt man eine vierwertige ein (quaternäre Phasenmodulation, QPSK), sind es 96 und so fort. Bekanntlich hat man die Wertigkeit der Modulation im Laufe der Zeit immer weiter gesteigert. Bei IEEE 802.11a und 11n war das Maximum die 64wertige Quadratur-Amplitudenmodulation (64-QAM). Mit 11ac kam eine 256wertige hinzu und 11ax wird sogar die 1024-QAM unterstützen.

Bei allen OFDM-Varianten ist bisher eines gleich geblieben: Der Abstand der Unterträger. Er beträgt genau 312,5 kHz. Daraus ergibt sich eine Symboldauer von 3,2 μs. Tatsächlich wird diese Zeit nicht vollständig für die Datenübertragung ausgenutzt sondern es gibt zwischen den Symbolen eine kurze Pause, den so genannten Schutzabstand (bzw. Guard Interval). Ursprünglich beträgt der Schutzabstand 0,8 μs, wodurch sich die nutzbare Symbolrate zu 250.000 pro Sekunde ergibt (der in Abbildung 2 dargestellte Wert). Mit 11n hat man einen kürzeren Schutzabstand eingeführt (0,4 μs), und man konnte dadurch die nutzbare Symbolrate um 20% erhöhen.

Wozu ist der Schutzabstand gut? In der Tat steckt im Schutzabstand der bahnbrechende Vorteil des OFDM gegenüber anderen Modulationstechniken. Funksignale gelangen meist auf mehreren Wegen zum Empfänger, da sie zwischenzeitlich an allerlei leitenden Elementen reflektiert werden. die Wege sind unterschiedlich lang, die reflektierten Signale erreichen also den Empfänger zu unterschiedlichen Zeiten. Der Schutzabstand sorgt dafür, dass sich zwei aufeinanderfolgende Symbole trotz Mehrwegeempfang nicht gegenseitig überlagern und stören. 0,4 μs entsprechen einem Weg von ca. 120 Metern. In normalen WLAN-Umgebungen sind Wegeunterschiede deutlich kürzer. OFDM ist also gegenüber Mehrwegeempfang sehr robust. Zum Vergleich: In WLAN gemäß IEEE 802.11b mit 11 Mbit/s beträgt die Symboldauer nur 0,25 μs, weniger als ein Zehntel der OFDM-Symboldauer. Trotz ihrer geringen Datenrate sind diese WLANs wesentlich anfälliger auf Mehrwegeempfang.

Betrachten wir noch einmal das Fußball-