9. Mai 2017 Studie · Web & Cloud · Mobile Sicherheit · 19 Min. Lesezeit

# Migration von Edge Computing zu Hybrid-Edge-Computing-Verfahren

**Archiviert.** Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

**Quelle** https://technology2.a-sit.at/veroeffentlichungen/migration-von-edge-computing-zu-hybrid-edge-computing-verfahren/

Dieses Projekt zeigt, dass die Ansätze die von Edge Computing und Mobile Edge Computing verfolgt werden, noch nicht ausreichend sind und eine Erweiterung benötigen. Ergebnis ist ein neues Modell, Hybrid Mobile Edge Computing, das die Vorteile aus früheren Ansätzen sowie der Edge Computing Ansätze kombiniert und auf Auslagerungsframeworks abzielt.

Aktuell gibt es bereits mehrere Verfahren, um es mobilen Geräten zu ermöglichen, Aufgaben auf andere Rechner auszulagern. Die Ziele sind in diesem Kontext immer dieselben: Performance von mobilen Geräten verbessern und Energiebedarf reduzieren. In letzter Zeit hat sich außerdem herauskristallisiert, dass Rechenleistung näher zu den Benutzerinnen und Benutzern gebracht werden muss um bestmögliche Ergebnisse zu erzielen.

## 1. Einleitung

Internet-of-Things (IoT) ist ein aufstrebendes Forschungsgebiet und bezieht sich auf die Vernetzung von Klein- und Kleinstgeräten, sowie Sensoren. Das Ziel dieser Vernetzung ist es, einen erhöhten Nutzen aus den gesammelten Daten zu ziehen. Ein prominentes Beispiel für die Vernetzung sind beispielsweise *Connected Cars* (Vernetzung von Fahrzeugsensoren). Hierbei wird die Erfahrung der Lenkerin bzw. des Lenkers verbessert, durch die Einbeziehung von Daten von anderen, sich in der Umgebung befindlichen Fahrzeugen. Daten wie Wetter, Straßenbeschaffenheit, oder Verkehr können hier einbezogen werden, um genauere Verkehrsabschätzungen zu treffen, oder die Lenkerin bzw. den Lenker bereits frühzeitig über möglicherweise auftretende Gefahren zu warnen. *Smart Cities* sind ein weiteres Beispiel, die das IoT Konzept auf größerer Ebene verwenden. Hierbei werden beispielsweise Daten von Verkehrs-, Parkplatz-, oder Luftqualitätssensoren kombiniert, um den Verkehr in Stadtteilen besser zu lenken. Eine andere Möglichkeit ist beispielsweise Stromverbraucher den dezentral zur Verfügung stehenden Erzeugern gegenüberzustellen und diese je nach Bedarf ein- und auszuschalten. Ein klassischer Ansatz diese Geräte zu verbinden und die beschriebenen Visionen zu realisieren führt über den Einsatz von Cloud Computing. Hierbei werden die gesammelten Daten an die Rechenzentren übertragen, dort die Berechnungen

durchgeführt, und die entsprechenden Daten wieder zurückgesendet. Dieser Ansatz wird für viele Anwendungen ausreichend sein, allerdings entstehen immer mehr Latenzabhängige Anwendungen. Untersuchungen von Tomanek et al. [1] haben, durch kontinuierliche Messung der Latenz zu Cloud Services, ergeben, dass auch Cloud Services teilweise mit Latenzen jenseits der 100ms aufwarten können. Dies wird für Latenz-sensitive Anwendungen bereits zum Problem. Um dieses Problem zu umgehen, wurde das Fog- oder Edge-Computing Konzept [2] entwickelt. Hierbei wird die notwendige Rechenleistung in den Nahbereich der Benutzerinnen und Benutzer gebracht und zur Verfügung gestellt. Dies hat den Vorteil, dass die Last auf viele unterschiedliche Rechenknoten aufgeteilt wird, somit ausfallssicherer als der semizentrale Cloud Computing Ansatz ist und gleichzeitig nur kürzerer Übertragungswege überbrückt werden müssen. Somit wird die Latenz erheblich verbessert und die Geschwindigkeit durch direkte Verbindungen erhöht. Hiermit wird eine erhebliche Verbesserung der aktuellen Situation erreicht. Edge Computing dient hierbei nicht zwingend als kompletter Ersatz für Cloud Computing, sondern hat einen unterstützenden Charakter. Rechenleistung wird auf einer weiteren Ebene hinzugefügt. Wie eingangs erwähnt, bezieht sich der hier beschriebene Einsatz auf IoT Geräte, oder verallgemeinert, auf Anwendungen ohne interaktiv mit der Benutzerin bzw. mit dem Benutzer zu interagieren. Heutige mobile Geräte stehen aber vor einem sehr ähnlichen Problem, auch wenn dieses anderwärtig motiviert ist. Benutzerinnen und Benutzer möchten auf ihren mobilen Geräten immer aufwendigere und rechenintensivere Anwendungen ausführen, müssen dafür aber erhebliche Abstriche in der ihnen zur Verfügung stehenden Akkulaufzeit hinnehmen. Analysen des Akkuverbrauchs von mobilen Geräten haben gezeigt, dass neben der Anzeige die CPU der Hauptverbraucher in mobilen Geräten ist. Ein Ansatz der sich bereits bewährt hat ist, aufwendige Berechnungen an andere Geräte mit verfügbarer Rechenleistung auszulagern. Dies verlangt allerdings ein dynamisches Verhalten der Applikation, um auf die Gegebenheiten effizient eingehen zu können. Die Kombination dieser Verfahren mit Edge Computing oder ähnlichen Konzepten klingt in diesem Kontext vielversprechend. In diesem Projekt werden die Möglichkeiten ausgelotet, wie Edge Computing mit Auslagerungssystemen kombiniert werden kann, um das starre Verhalten bestehender Systeme zu umgehen und auf Rechenleistung in der Umgebung zurückzugreifen. Die Ergebnisse wurden in [3] veröffentlicht.

## 2. Analyse bestehender Verfahren

Ziel dieses Kapitels ist es, bestehende Verfahren anhand einer generalisierten Architektur zu untersuchen und Schwachstellen zu identifizieren, um flexiblere Systeme zu realisieren und sich dem Edge Computing Konzept anzunähern. Dieses Kapitel stellt hierbei einen ersten Schritt dar, in Kapitel 3 erfolgt ein Zwischenschritt, in dem ein auf mobile Geräte und auf speziell getrimmte Applikationen fokussiertes Edge Computing Modell eingeführt wird. In Kapitel 4 wird dann die nächste Entwicklungsstufe, als weiteren Beitrag dieses Berichts vorgestellt. Es ist nicht Ziel dieses Kapitels eine Aufzählung und detaillierte Beschreibung von bestehenden Frameworks bereitzustellen. Die Analyse erfolgt anhand eines abstrahierten Modells.

### 2.1. Aktuelle Ansätze

Mobile Cloud Computing (MCC) dient derzeit als Überbegriff für Systeme, die rechenintensive Aufgaben an externe Systeme auslagern. MCC beschränkt sich bei externer Rechenleistung hierbei auf von Cloud Computing bereitgestellte Dienste. Diese Dienste sind üblicherweise über das Internet erreichbar. Hierbei muss auf Modellebene bereits zwischen zwei unterschiedlichen Ansätzen unterschieden werden:

- Die in der Cloud bereitgestellten Dienste führen bereits die gesamte Programmlogik (oder starr vordefinierte Teile der Programmlogik) aus. Die mobilen Geräte dienen in diesem Szenario nur mehr als Eingabestationen, und ermöglichen damit die Miteinbeziehung von lokalen Daten des mobilen Gerätes wie GPS Koordinaten oder Benutzereingaben. Sind die Cloud Dienste also nicht verfügbar, ist auch die Funktionalität der Applikation, je nach Grad der Einbindung, sehr beschränkt.

- Ein erweitertes Modell, das ebenfalls in den MCC Bereich fällt, ermöglicht das dynamische Auslagern von Programmteilen an generischere Dienste in der Cloud. Dynamisch bedeutet in diesem Fall, dass Kontextbedingungen miteinbezogen werden. Es wird bei der Auslagerung also darauf Rücksicht genommen, dass durch die Auslagerung bessere Ergebnisse erzielt werden, unter Berücksichtigung der Werte wie Verzögerungszeit zum Cloud Dienst oder aktuelle verfügbare Bandbreite. Der zweite Ansatz ist der, auf den in diesem Bericht abgezielt wird. Hierbei werden im Wesentlichen zwei Verfahren eingesetzt um eine Remote Ausführung von einzelnen Programmteilen zu ermöglichen:

- Konzepte basierend auf virtuellen Maschinen zielen darauf ab, eine Umgebung bereitzustellen, die der des mobilen Gerätes sehr ähnlich ist. Dabei werden Faktoren berücksichtigt wie Architektur, CPU, Betriebssystem und Betriebssystemversion. Dies ermöglicht es Frameworks sogar Applikationen, die ursprünglich nicht für eine verteilte Ausführung entwickelt wurden, mit verschiedenen Verfahren trotzdem verteilt auszuführen. CloneCloud [4] beispielsweise analysiert Applikationen und ermittelt Applikaitonspartitionen, die sich dafür eignen, remote ausgeführt zu werden. An den entsprechenden Stellen im Binary werden dann spezielle Instruktionen eingeführt, die von einem modifizierten Betriebssystem als Auslagerungsinstruktionen erkannt werden. ThinkAir [5] und COMET [6] sind nur zwei Implementierungen, die ähnliche Ansätze verfolgen. Die Nachteile des VM Konzepts liegen auf der Hand. Einerseits müssen eine Vielzahl von virtuellen Maschinen verwaltet bzw. dynamisch erzeugt werden, um den unterschiedlichsten Anforderungen nachzukommen. Andererseits werden auf den mobilen Geräten teilweise modifizierte Betriebssystemversionen benötigt.

- Ein leichtgewichtigeres Auslagerungskonzept ist der Methoden-basierte Ansatz. Hierbei wird zwar Unterstützung des Entwicklers vorausgesetzt, der bereits auf Source-Code Level Applikationspartitionen vorgibt. Diese Programmteile können dann Frameworkabhängig ausgelagert werden. MAUI [7] beispielsweise implementiert ein System basierend auf der Platformunabhängigem .NET Umgebung. Dadurch kann eine Auslagerung auch Platformübergreifend erfolgen, ohne Rücksicht auf konkret verwendete Hardware und Architekturen.

## 2.2. Architektur

Ohne auf die Einzelheiten jeder spezifischen Implementierung einzugehen kann eine generelle Architektur abgeleitet werden, die einen Großteil der Aspekte von bestehenden Frameworks abdeckt. Die abgeleitete Architektur ist in Abbildung 1 dargestellt.

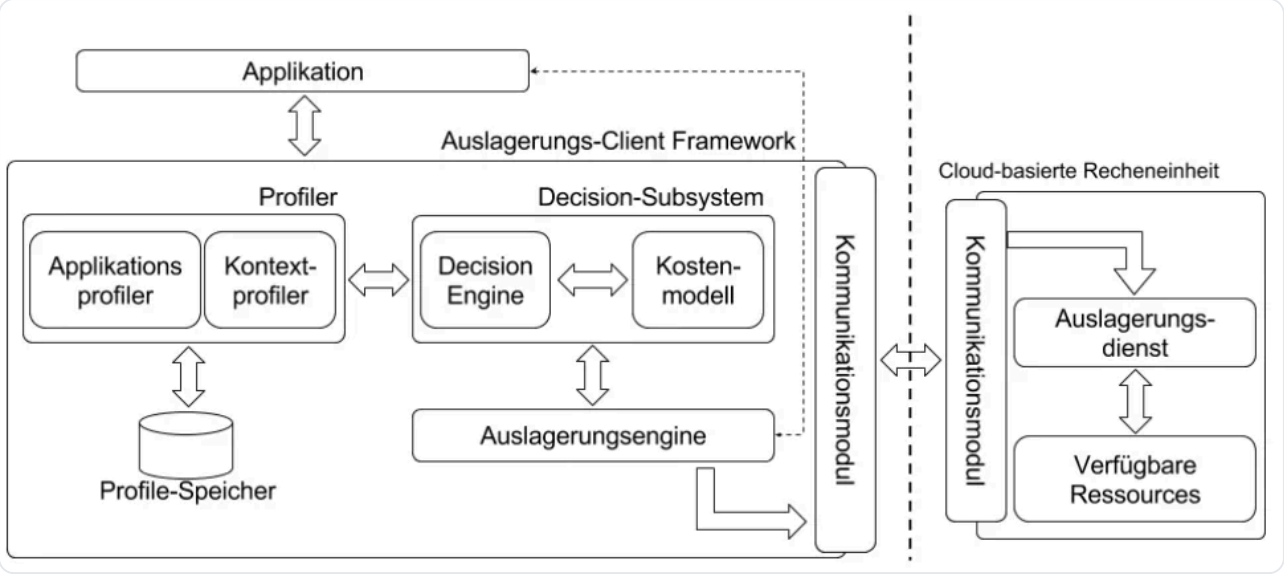



Abbildung 1: - Abstrahierte Architektur bestehender Frameworks Klar ersichtlich ist ein unterschiedlicher Detaillierungsgrad auf der Server- und Client-Seite. Dieser kommt daher, da bestehende Frameworks sehr auf die Clientseite fokussiert sind, etwa wie können Programmteile, die zur Auslagerung geeignet sind, identifiziert werden, oder wie die Auslagerung technisch durchgeführt werden kann. Die Serverseite wird nicht im Detail betrachtet und wird meist auf die einfache Ausführung reduziert. Durch eine fixe Zuordnung von einzelnen Cloud-basierten Recheneinheiten ist diese Betrachtung jedoch durchaus ausreichend. Generell unterscheiden Auslagerungssysteme zwischen vier Komponenten:

```
- Der _Profiler_ stellt Performancewerte für die unterschiedlichen Tasks bereit. Der _Applikationsprofiler_ zeichnet die Ausführungszeiten auf, und gibt Abschätzungen über die Ausführungszeit von Tasks ab. Der _Kontextprofiler_ hingegen liefert Performancewerte der Umgebung wie etwa die Latenz zum Remotesystem oder aktuell verfügbare Bandbreite. Die ermittelten Werte werden zwischengespeichert.

- Das _Decision-Subsystem_ ermittelt, unter Einbeziehung der Profilingwerte, ob sich die Auslagerung eines Tasks lohnt. Das Kostenmodell definiert hierbei, ob die Entscheidungen auf Leistung oder Energieverbrauch getrimmt werden sollen.

- Die _Auslagerungsengine_ ist anschließend für die technischen Aspekte der Auslagerung verantwortlich. Dies beinhaltet die notwendige Synchronisation des Applikationsstatus, sowie die Migration des Programmablaufs und Reintegration der Resultate.

- Das _Kommunikationsmodul_ ist die Schlüsselkomponente in der Kommunikation mit der Remoteseite.
```

## 2.3. Analyse

MCC legt den Fokus ausschließlich auf Cloud Computing, ohne andere Konzepte in Betracht zu ziehen. Für viele Anwendungen ist dies durchaus ausreichend, auch im Bereich der interaktiven Anwendungen. Fakt ist allerdings, dass Cloud Computing das Hauptaugenmerk nicht auf die Minimierung der Latenzen zu den Benutzerinnen und Benutzern legt, sondern auf die Maximierung der gebotenen Flexibilität und Bündelung

der verfügbaren Ressourcen. Vor Allem durch die Verwendung von Technologien wie UMTS und LTE ergibt sich in vielen Szenarien automatisch eine höhere Verzögerung. Generell kann also festgehalten werden, dass MCC zwar technologisch viele Fortschritte gebracht hat, diese Technologie alleine aber nicht zur Behebung des Performance- und Energieproblems bei mobilen Geräten führt. Großes Potential wird hierbei der Edge Computing und Mobile Edge Computing Technologie zugeschrieben, das in den folgenden Kapiteln im Detail herausgearbeitet wird.

## 3. Mobile Edge Computing

Edge Computing im Allgemeinen unterscheidet sich von Cloud Computing [8] in mehreren Aspekten:

```
- Verglichen mit Edge Computing, kann Cloud Computing als ein semizentraler Ansatz
angesehen werden. Es gibt also mehrere zentrale Stellen, die die angeforderte
Rechenleistung bereitstellen können. Bei Edge Computing hingegen wird die
Rechenleistung in die unmittelbare Umgebung des Benutzers gebracht.

- Die Zeitverzögerung in Cloud Computing Systemen wird durch die Anzahl der
benötigten Hops auf IP Ebene bestimmt. Die Anzahl der Hops wird maßgeblich durch
die Anbindung des Internetanbieters bestimmt, kann sich aber durchaus im
zweistelligen Bereich bewegen. In Edge Computing Szenarien ergeben sich durch die
unmittelbare Nähe der Recheneinheiten sehr geringe Verzögerungen und nur einige
wenige Hops. Diese Unterscheidungen können gleichzeitig auch als Hauptvorteile von
Edge Computing verstanden werden. Edge Computing per se bietet noch kein
Ausführungs- oder Programmiermodell, sondern liefert nur eine mögliche Art der
Organisation von Rechenleistung. Außerdem darf die Edge Computing Ebene nicht als
Ersatz für Cloud Computing verstanden werden, sondern dient als dynamische
Ergänzung der bereitgestellten Rechenleistung. Es ist klar zu erkennen, dass sich
das Edge Computing Modell optimal zur Anreicherung von mobilen Geräten eignet. Dies
wurde vom European Telecommunication Standard Institute (ETSI) aufgegriffen [9]-
[11] und in eine detaillierte Spezifikation zur Verwendung von Edge Computing in
mobilen Use Cases entwickelt. Mobile Edge Computing (MEC) sieht eine komplette
Integration in die Infrastrukturen von Telekommunikationsprovidern vor, ist also
auf die 4G und 5G Technologie zugeschnitten. Dies ist einerseits ein
einschränkender Ansatz, ermöglicht aber gleichzeitig eine detailliertere
Spezifikation der Abläufe. Ein Überblick über die MEC Architektur ist in Abbildung
2 ersichtlich. MEC forciert eine strikte Trennung zwischen Applikationen, die am
Endgerät des Benutzers, und Mobile Edge Applikationen (ME App), die auf der MEC
Infrastruktur ausgeführt werden. ME Applikationen sind lose mit den
Endbenutzerapplikationen verbunden. Die Endbenutzerapplikationen können Instanzen
von ME Applikationen anfordern, die dann in den Programmablauf eingebunden werden.
In der Spezifikation wird zwischen der Systemebene und der Hostebene unterschieden.
```

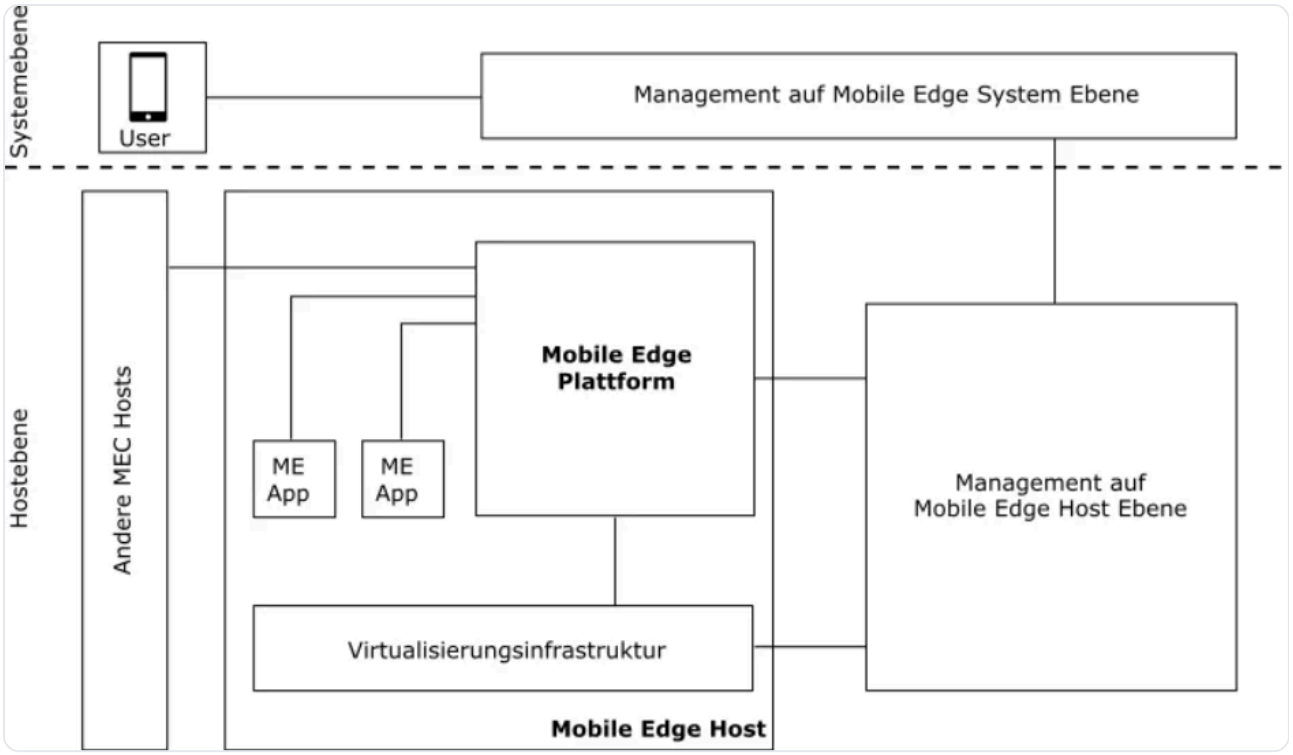



Abbildung 2: - Mobile Edge Computing Architekturübersicht Die Systemebene fungiert hierbei als Kontaktpunkt für Endbenutzerapplikationen. Die Hostebene hingegen fungiert als Backbone und kombiniert die verfügbaren Ressourcen. Hierbei wird zwischen den folgenden logischen Entitäten unterschieden:

```
- Der _Mobile Edge Host_ stellt Rechenleistung, Speicher, und Netzwerkressourcen
für _Mobile Edge Applikationen_ zur Verfügung. Es wird dazu auf
Virtualisierungstechnologien zurückgegriffen.

- Die _Mobile Edge Plattform_ Dient als Hostplattform für _Mobile Edge
Applikationen_ . Außerdem bietet die Plattform Services an, um andere Dienste zu
verwenden bzw. ausfindig zu machen.

- _Mobile Edge Applikationen_ werden auf der virtualisierten Infrastruktur
ausgeführt und können von Endbenutzerapplikationen die auf mobilen Geräten
ausgeführt werden verwendet werden.

- Die Managementkomponenten verwalten die zur Verfügung stehende virtualisierte
Hardware und instanziieren entsprechende virtuelle Maschinen die für die Ausführung
von Applikationen erforderlich sind. Damit einhergehend verwalten sie auch den
Lebenszyklus von _Mobile Edge Applikationen_ . MEC dient als Technologie die es
ermöglicht Use Cases umzusetzen die zwar Rechenleistung auf Cloud Computing Niveau
benötigen, aber striktere Anforderung an Latenzen stellen. Außerdem erlaubt MEC
Zugriff auf zusätzliche Kontextinformationen durch die enge Verstrickung mit der
Telekommunikationsinfrastruktur. Haupteinsatzgebiete dieser Technologie sind die
bereits erwähnten _Connected Cars_ um beispielsweise Gefahrenwarnungen besser zu
kommunizieren. Im Bereich der Videoanalyse bei Sicherheitskameras kann diese
Technologie verwendet werden, um eine ungenaue Analyse die direkt von Kameras
durchgeführt wird zu vermeiden, trotzdem aber nicht riesige Datenmengen zu
entfernten Cloud Providern hochgeladen werden müssen.
```

## 4. Hybrid Mobile Edge Computing

Die von MEC identifizierten Lösungsansätze können im mobilen und IoT Bereich viele Probleme lösen, sind aber in ihrem Einsatzbereich nach wie vor sehr eingeschränkt. Für Entwickler ist der MEC Ansatz aufwendig, da mehrere Applikationen parallel entwickelt werden müssen. Aus der Sicht von mobilen Geräten, mit dem Ziel die Leistungsfähigkeit der Geräte zu verbessern, durch Auslagern von rechenintensiven Aufgaben, ist der MEC Ansatz prinzipiell dem herkömmlichen Cloud Computing Ansatz sehr ähnlich. Der Unterschied besteht darin, dass MEC Applikationen dynamisch in der Umgebung des Benutzers instanziiert werden. Dies bedeutet aber gleichzeitig auch, dass mobile Applikationen im Allgemeinen nicht funktionsfähig sind, wenn keine kompatible MEC Infrastruktur in Reichweite ist. Für mobile Applikationen mit ähnliche Anforderungen wie Mobile Edge Computing Anwendungen, wie etwa geringe Latenzen aber trotzdem Zugriff auf eine Cloud Computing ähnliche Infrastruktur, werden also flexiblere Ansätze benötigt. Die Building Blocks hierfür sind an sich über die bereits bestehenden Auslagerungsframeworks und über das Mobile Edge Computing Konzept bereits vorhanden, benötigen aber entsprechende Anpassungen. Die folgenden Kapitel konzentrieren sich auf diese beiden Aspekte. In Kapitel 4.1 wird die Architektur von bestehenden Frameworks dahingehend angepasst um auch den Anforderungen von interaktiven mobilen Applikationen zu genügen. In Kapitel **Fehler! Verweisquelle konnte n icht gefunden werden.** wird dann ein mögliches Deploymentszenario illustriert um Lösungen in das Mobile Edge Computing Konzept einzubinden.

## 4.1. Architektur

Die Architekturüberlegungen basieren auf dem abstrakten Model aus Abbildung 1. Die Architektur wird sukzessive erweitert um die Anforderungen an Flexibilität und dynamischer Verwendbarkeit der verfügbaren Recheneinheiten zu erfüllen. Um die Übersicht zu gewährleisten, wird eine Trennung in Client und Server beibehalten, diese kann aber als rein strukturelle Trennung verstanden werden. Die durchgeführten Entwicklungen und Verbesserungen sind in Abbildung 3 ersichtlich. Die wohl auffälligste Änderung ist die Einführung eines Kommunikationsnetzwerkes. Dies ersetzt die statische Verbindung zu einzelnen Nodes. Ziel dieses Netzwerkes ist es mobile Geräte, die Unterstützung benötigen, mit Geräten, oder allgemeiner Nodes, die Unterstützung anbieten, zusammenzubringen. Dabei sollen Aspekte wie Verbindungsaufbau und Erreichbarkeit einzelner Nodes ausgeklammert werden. Prinzipiell zielt die Architektur hierbei nicht darauf ab, Implementierungen technologisch einzuschränken. Peer-to-Peer Netzwerke sind aber mit Sicherheit eine Möglichkeit dieses Netzwerk umzusetzen. Die konkrete Realisierung ist außerdem vom jeweiligen Deployment abhängig. Dadurch ergibt sich eine Struktur, in der mobile Geräte mit vielen potentiellen Nodes konfrontiert sind die ihre Ressourcen anbieten. Diese müssen von den mobilen Geräten kategorisiert bzw. aussortiert werden. Nicht alle Nodes sind für jedes mobile Gerät geeignet und können Anforderungen an Latenz und Bandbreite möglicherweise nicht erfüllen. Ein Hintergrundtask als fixer Bestandteil des Frameworks katalogisiert vorhandene Nodes und bewertet in einem ständigen Prozess deren Performancewerte. Diese Nodes werden in Evidenz gehalten um im Falle einer anstehenden Auslagerungsoperation bereit zu sein. Profiler sind unterstützende Komponente, die Performancewerte der Applikationsausführung sowie der Umgebung messen. Diese sind bereits integraler Bestandteil von bestehenden Frameworks, bieten dabei aber nur ein geringes Maß an Flexibilität. Oft werden Performanceprofile offline erstellt, fokussiert auf einzelne Systeme. Dieser Ansatz ist bei statischen Verbindungen zu Auslagerungsressourcen durchaus zielführend. Werden aber Knoten mit unterschiedlichsten Architekturen und Eigenschaften in das System miteinbezogen, haben unteranderem die folgenden Parameter Auswirkungen auf den Auslagerungsprozess:

```
- Architektur bzw. Type des Geräts

- Verfügbare CPU

- Verfügbarer Speicher

- Aktuelle Auslastung, evtl. durch andere Benutzer die dieselben Knoten verwenden

- Verfügbare Bandbreite

- Aktuelle Latenz Deshalb wird bereits auf Architekturebene ein dynamischerer
Ansatz forciert. Hierbei sollen diese Parameter zur Laufzeit ausgewertet werden.
Dieser Prozess kann natürlich durch offline gesammelte Werte ergänzt werden. Die
Decision Engine und der Entscheidungsprozess im Allgemeinen wurde bereits
detailliert von anderen Frameworks betrachtet und wird an dieser Stelle nicht neu
erfunden. Eine Ergänzung ist jedoch die Verankerung von mehreren, parallelen
Kostenmodellen. Kostenmodelle generell bestimmen das Ziel einer
Auslagerungsoperation. Zwei konträre Strategien sind hierbei _Leistungsoptimiert_
und _Energieoptimiert_ . Ein Kostenmodell verfolgt hierbei entweder das Ziel
maximale Leistung für den Benutzer zu erreichen, bzw. die Laufzeit des mobilen
Geräts maximal auszudehnen. Durch mehrere parallele Kostenmodelle kann der Einfluss
der einzelnen Kostenmodelle dynamisch, beispielsweise basierend auf dem aktuellen
Batteriestand, verändert werden. Durch die Einführung des Sicherheitssubsystems
werden zukünftige Erweiterungen ermöglicht die beispielsweise eine
Zugriffskontrolle bzw. eine Bewertung der Vertrauenswürdigkeit von remote Nodes
zulassen.
```

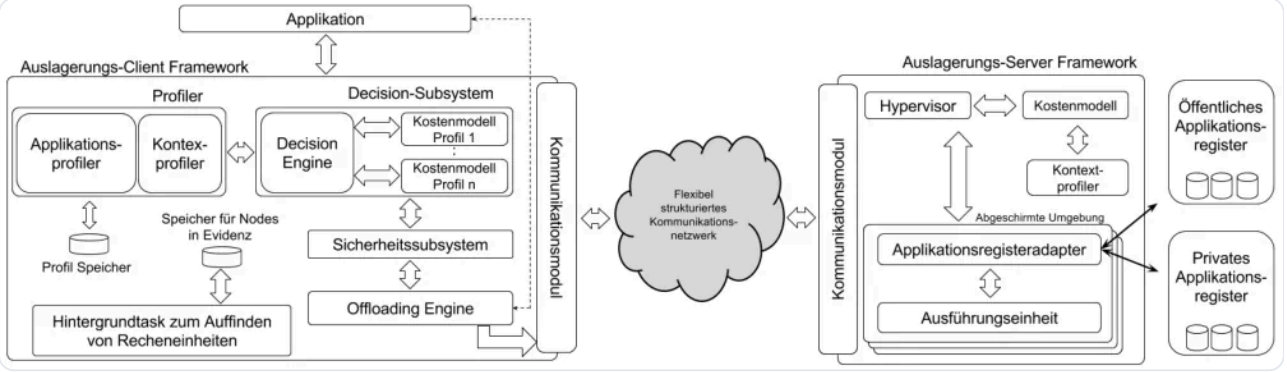



Abbildung 3: - Verbesserte und angepasste Architekturübersicht Serverseitig ergibt sich die Situation, dass bisherige Frameworks und Implementierungen diese nicht im Detail betrachtet haben. Durch die statische Verbindung der einzelnen Nodes war Task

und Ressourcenmanagement nicht notwendig. Dies muss nun nachgeholt werden. Es können die folgenden Anforderungen an die Serverumgebung definiert werden:

```
- Ausgeführte Tasks dürfen sich gegenseitig nicht beeinflussen können bzw.
gegenseitig auf ihre Daten zugreifen.

- Betreiber sollen die Kontrolle über ihre freigegebenen Ressourcen behalten und
diese entsprechend verwalten können

- Notwendige Applikationsteile müssen dynamisch nachgeladen werden können.
```

Daraus ergibt sich eine Serverseitige Architektur wie ebenfalls in Abbildung 3 dargestellt. Ein Hypervisor kontrolliert die Ausführung von angenommenen Tasks und führt diese in abgeschirmten Umgebungen (Sandboxing) aus. Der Mechanismus zum Abschirmen der Umgebung unterscheidet sich erheblich je nach verwendeter Umgebung und Architektur. Auf mobilen Geräten beispielsweise könnte auf die bereits vorhandene starke Abschirmung zwischen Prozessen zurückgegriffen werden. In Serverumgebungen bieten sich virtualisierte Umgebungen an. Generell soll dies auf der aktuellen Abstraktionsebene aber nicht festgesetzt werden. Der Hypervisor ist eng mit einem Kostenmodell und einem entsprechenden Kontextprofiler verbunden. Diese sollen eine faire Nutzung der bereitgestellten Ressourcen ermöglichen, bzw. soll es dem Betreiber ermöglichen seine Ressourcen nur bis zu einem gewissen Grad zu teilen. Applikationen bzw. relevante Applikationsteile werden über externe Repositories bereitgestellt. Diese Repositories können öffentlich und für jeden zugänglich sein, oder nur autorisierten Teilnehmern den Zugriff erlauben. Dieses Kapitel behandelt notwendige Architekturänderungen um die Flexibilität und Fähigkeiten der dynamischen Auslagerung zu verbessern, ist aber im Grunde nicht auf Edge- bzw. Mobile Edge Computing hin getrimmt. Im folgenden Kapitel wird der entwickelte Ansatz in die Domäne von Mobile Edge Computing transferiert und eröffnet damit ein neues Feld: Hybrid Mobile Edge Computing.

## 4.2. Hybrid Mobile Edge Computing

Mobile Edge Computing bietet, wie in Kapitel 3 festgestellt, die perfekten Voraussetzungen um mit dynamischen Mobile Cloud Computing Ansätzen kombiniert zu werden. Aktuelle Mobile Edge Computing Ansätze zielen stark auf IoT Geräte. Gerade Smartphones und Tablets würden aber anlassbasierte Möglichkeiten benötigen um die Performance der Geräte zu steigern bzw. die Batterielaufzeit zu verlängern. Mobile Edge Computing hat dazu bereits die notwendige Rechenleistung und ist eng integriert mit dem dichten Netz der Telekommunikationsanbieter. Rechenleistung, nahe am Benutzer, ist also bereits vorhanden wird aktuell aber nicht zu dem Grad verwendet wie es möglich und sinnvoll wäre. Im Wesentlichen wird in diesem Kapitel eine Umsetzung des flexiblen Kommunikationsnetzwerkes beschrieben, über das Auslagerungsframeworks in Mobile Edge Computing Infrastrukturen integriert werden können. Die abgeleiteten Architekturüberlegungen bleiben hierbei unverändert gültig. An früherer Stelle wurde bereits erwähnt, dass sich Peer-to-Peer Netzwerke dafür anbieten, dieses Netzwerk zu realisieren. In Peer-to-Peer Netzwerken kann jeder Teilnehmer als gleichberechtigt angesehen werden, wird also auch dafür verwendet, um Anfragen an die Zielgeräte weiterzuleiten. Speziell im Fall von mobilen Geräten, mit oftmals instabilen Verbindungen ist dies nicht ideal und würde die Performance des gesamten Netzwerkes negativ beeinflussen. Außerdem sind Nodes, die in der unmittelbaren Umgebung zum Benutzer bzw. zur Benutzerin stehen erheblich wichtiger und wertvoller, als entfernte Nodes. Entfernte Nodes sind eher nur in Ausnahmefällen relevant, wenn beispielsweise spezielle Hardwareanforderungen erfüllt werden müssen oder die Auslagerung nur auf bestimmten Nodes durchgeführt werden darf. Eine Verbindung aller Nodes ist also erforderlich, lokale Nodes sollten aber in gewisser Weise bevorzugt werden.

Um diese Anforderungen zu erfüllen, wird ein zweistufiger Ansatz verwendet, der sich optimal mit Mobile Edge Computing integriert. Die Eckpunkte von Mobile Edge Computing sind, dass pro Telekommunikationsbasisstation lokale Nodes verfügbar sind die im Mobile Edge Computing Szenario lokal zur Verfügung gestellt werden. Zwischen den Basisstationen wird ein Super-Node basiertes und performantes Peer-to-Peer Netzwerk realisiert. Verfügbare Nodes sowie mobile Geräte melden sich an der Super-Node der jeweiligen Basisstation an und sind damit quasi Teil des Peer-to-Peer Netzwerkes, ohne Netzwerkverwaltungsaufgaben übernehmen zu müssen. Eine Illustration dieses Deployments ist in Abbildung 4 ersichtlich. Über den Zusammenschluss aller Super-Nodes können trotzdem alle Nodes im gesamten System erreicht werden. Super-Nodes können aber die Ressourcenanfragen auf lokale Nodes optimieren. Darüber hinaus kann ein globaler Zugriffsknoten eingeführt werden, der es ermöglicht, eigene, exklusive Ressourcen in dieses System einzubinden, wie beispielsweise den eigenen Heimcomputer für Aufgaben auf vertraulichen Daten. Ziel ist es, dass mobile Geräte die Unterstützung anfordern, und Knoten die Unterstützung in Form von Rechenleistung anbieten immer direkt verbunden sind. Das Super-Node basierte Peer-to-Peer Netzwerk nimmt hierbei nur die Rolle eines Vermittlers ein und unterstützt beispielsweise beim Verbindungsaufbau nach dem ICE [12] Protokoll.

## 5. Fazit

Dieser Bericht widmet sich dem steigenden Leistungs- und Energiebedarf von aktuellen mobile Geräten. Teilaspekte dieses Problem wurden durch Auslagerungsframeworks mit dem Überbegriff Mobile Cloud Computing bereits gelöst. Auf technischer Ebene ist es also bereits klar, dass die Auslagerung von Rechenintensiven Programmteilen funktioniert und erhebliche Performance- sowie Energieverbrauchsverbesserungen mit sich bringt. Parallel wurde eine Spezifikation erarbeitet, die auf ein ähnliches Problem hinzielt, im Endeffekt aber nur eingeschränkt für den Einsatz auf mobilen Geräten geeignet ist. Durch die Einführung von Hybrid Mobile Edge Computing und die Erweiterung der Architektur bestehender Systeme wird der Einsatzbereich dieses Ansatzes ausgedehnt. Dies erweitert die Einsatzszenarien einerseits und erleichtert die Verwendung andererseits. Ein offener Punkt derzeit ist die Bewertung der Vertrauenswürdigkeit von Nodes im System. Sollen sensible Daten verarbeitet werden, muss sichergestellt werden, dass dies nur auf vertrauenswürdigen Knoten passiert.

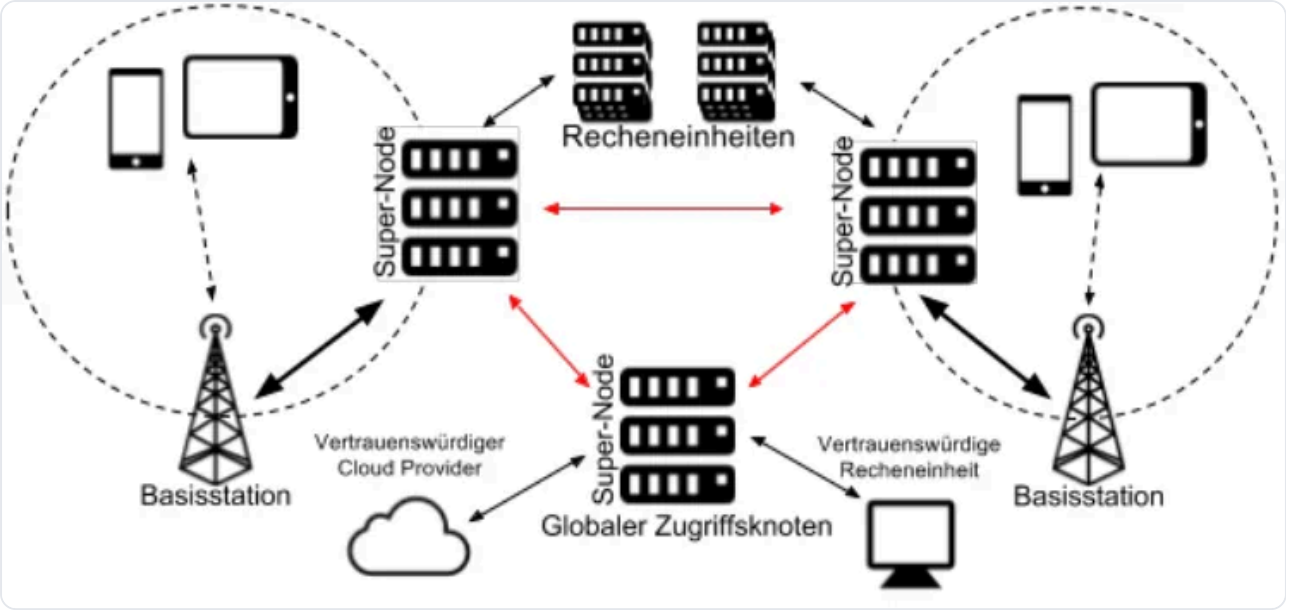



Abbildung 4: - Super-Node basiertes Deployment Im Allgemeinen erlaubt dieses Verfahren nicht nur die Integration mit Mobile Edge Computing Ressourcen, sondern ebenfalls mit beliebigen anderen und kann deshalb als Hybrid Mobile Edge Computing bezeichnet werden.

## 6. Literaturverzeichnis

[1] O. Tomanek, P. Mulinka, and L. Kencl, "Multidimensional cloud latency monitoring and evaluation," *Comput. Networks*, vol. 107, p., 2016.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," *Proc. first Ed. MCC Work. Mob. cloud Comput.*, pp. 13–16, 2012.

[3] A. Reiter, B. Prünster, and T. Zefferer, "Hybrid Mobile Edge Computing: Unleashing the Full Potential of Edge Computing in Mobile Device Use Cases," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2017.

[4] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic Execution Between Mobile Device and Cloud," in *Proceedings of the sixth conference on Computer systems*, 2011, pp. 301–314.

[5] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proceedings of 2012 IEEE INFOCOM*, 2012, pp. 945–953.

[6] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, "COMET : Code Offload by Migrating Execution Transparently," pp. 93–106.

[7] E. Cuervo *et al.*, "MAUI : Making Smartphones Last Longer with Code Offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, 2010, vol. 17, pp. 49–62.

[8] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. We, and L. Sun, "Fog Computing: Focusing on Mobile Users at the Edge," *eprint arXiv:1502.01815*, pp. 1–11, 2015.

[9] ETSI, "ETSI GS MEC 002 - Mobile Edge Computing (MEC); Technical Requirements," 2016.

[10] ETSI, “ETSI GS MEC 003 - Mobile Edge Computing (MEC); Framework and Reference Architecture,” 2016.

[11] ETSI, “ETSI GS MEC-IEG 004 - Mobile Edge Computing (MEC); Service Scenarios,” 2015.

[12] J. Rosenberg, “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols,” 2010.