Die offenen Themen finden Sie ab Seite 2. Hier einige einleitende Überlegungen.

Der Studienschwerpunkt mobile Anwendungen ist Teil einer Hochschule der **angewandten Wissenschaften**. Wir arbeiten an der Lösung von Problemen, die für uns (ja, ich meine Sie :)) sichtbar sind, oft aber nicht im Massenmarkt. Alle Innovationen fangen so an. Leider sind spektakuläre Ideen anfangs nicht von dummen zu unterscheiden. Das ist Teil des Spiels. Als Ingenieur und angewandter Wissenschaftler interessieren mich Probleme für die es nicht die offensichtliche Lösung gibt. Mich interessiert primär nicht, ob es einen Markt für eine Lösung gibt. Wir wissen nicht, ob wir gerade die nächste Smartphone-Revolution oder das nächsten Cargolifter-Projekt aufsetzen. Im Nachhinein ist man klug. Eins aber weiß ich genau: Der sichere Weg, keine Innovationen zu finden besteht darin, es so zu machen wie es alle machen. Deshalb machen wir, was wir spannend finden.

Ich betreue **Bachelor-, Masterarbeiten, Independent Coursework und Forschungsprojekte**. Die folgenden Themen lassen sich in allem Modulen einsetzen – man muss sie manchmal ein wenig adaptieren. Die folgenden Projekte beschäftigen mich derzeit sehr.

Inhaltsverzeichnis

Projekt Shark/ASAP

Shark/ASAP ist ein Open-Source Developer-Framework. Damit lassen sich mobile dezentrale Anwendungen implementieren, die natürlich über das Internet nutzen **aber nicht müssen**. Die übergroße Mehrheit der mobilen Anwendungen greift auf einen Server zu und braucht dazu Internet-Access. Praktisch alle Peers von P2P Anwendungen laufen im Internet. Das ist alles in Ordnung wenn es um E-Commerce Anwendungen und digitale Währungen.

Das ist nicht mehr so *natürlich*, wenn es um Messenger-Dienste geht. Etwas dezentraleres als die menschliche Kommunikation kann man sich kaum denken. Viele Menschen reden, geben Infos weiter etc. pp. Das läuft seit einer Millionen Jahren stabil ohne Cloudservice und das soll auch so bleiben. Ich erwarte von einem Messenger, dass er immer funktioniert und dass er ein Kommunikationsweg nutzt, den ich in Ordnung finde und der gerade verfügbar ist. Das ist es was wir bauen. Machen Sie mit. Hier ein paar Themenvorschläge. Sprechen Sie mich gern an. Infos finden sich auch in der Mediathek.

Prof. Dr.-Ing. Thomas Schwotzer (thomas.schwotzer@htw-berlin.de)
HTW Berlin, WH C 616.

Generell kann man die Arbeiten in zwei Klassen teilen: Sie arbeiten an a) Anwendungen für Shark/ASAP oder ermöglichen b) die Kommunikation über weitere Netzwerke.

Offene Themen – Anwendungsentwicklung

Shark ist ein Developer-Framework. Man kann also Anwendungen bauen, die in andere Anwendungen integriert werden können. Mit SharkComponent existiert dazu ein Komponentenmodel. Kein Grund davor zurück zu schrecken. Es ist nicht komplex. Es ist aber die Basis auf der wir Anwendungen, konkreter: Komponenten implementieren. Das würden Sie in den folgenden Arbeiten auch tun.

Verteilte automatisierte Tests SharkMessenger

Es existiert ein Command Line Interface Implementierung für den SharkMessenger. In einem Video zeige ich dessen Nutzung. Ein CLI reagiert auf Kommandoeingaben, die dann ausgeführt werden. Diese Kommandos müssen nicht zwingend über die Tastatur von einem Menschen eingegeben werden. Sie könnten aber auch über ein File oder sogar über das Netzwerk empfangen werden.

Wir brauchen die Möglichkeit funktionale Tests und Lasttests in realen Umgebungen durchzuführen. Es gilt also ein System zu schaffen mit dem man Szenarien beschreiben kann. Ein Szenario beschreibt welche Kommandos wann abgearbeitet werden sollen. Die Logfiles müssen an einen zentrale Stelle geschickt werden.

Das könnte tatsächlich als Shark-Anwendung implementiert werden. Peer werden instanziiert, verbinden sich mit einem vorher definierten Peer. Von dort kommen die Kommandos, dorthin gehen die Logs. Das System würde also die Kommandos erhalten, dann alle Verbindungen beenden, die Kommandos abarbeiten, danach die Verbindung wieder aufbauen und die Ergebnisse senden.

Im Idealfall erlaubt das System, dass sich Peer im Ablauf koordinieren, das also eine Menge von Kommandos erst abgearbeitet wird, wenn ein anderes Peer eine Reihe von Kommandos erledigte.

Integration von Pretty Good Privacy in SharkPKI

Basierend auf Shark/ASAP wurde eine eigene SharkPKI implementiert, die auch stabil arbeitet. Vor allem Open-Source Projekte sollten natürlich offen sein für andere offene und freie Projekte.

Es liegt auf der Hand, dass man Pretty Good Privacy (PGP) in Shark nutzt. Die Idee von PGP ist einfach wie effektiv. Menschen erzeugen Keypaare und können diese auch zertifizieren. Oft aber werden die Keypaare lediglich auf zentrale Server gestellt. Von dort werden sie bezogen und im besten Fall werden die Fingerprints verglichen bevor sie genutzt werden.

Die SharkPKI arbeitet mit RSA Schlüsseln. Damit lassen sich natürlich auch PGP Key nutzen. Ziel der Arbeit ist es, ein Konzept zu entwickeln wie PGP mit der SharkPKI zusammen arbeiten kann. Prototypisch soll gezeigt werden, dass das auch programmierbar ist. Ein vollständig lauffähiger Prototyp wäre nett, aber erwartet wird nur eine Teilimplementierung.

SharkMessenger Android GUI

Der SharkMessenger ist DER Showcase für das Projekt. Die grundsätzliche Idee von ASAP ist die Verteilung von Nachrichten über Ad-hoc Netzwerke nach Prinzipien von Gossip- und opportunistischen Protokollen. Der Messenger funktioniert nunmehr recht stabil. Unit- und eine wachsende Anzahl von Lasttests beweisen das. Und wir haben eine erstes Command Line Interface, siehe auch das Video.

Es gibt eine rudimentäre GUI im Projekt SN2. Die Software ist nun stabil genug, um an einem Release zu bauen, der erste Schritt ist eine Alpha-Version. Das ist der Job. Es gibt bereits ein Programmiergerüst. Sie werfen alles weg, fangen von vorn an oder arbeiten sich ein und machen es besser.

Ihr Ziel ist die Erstellung eines funktionierenden Prototyps inklusive von End-to-End Tests auf Android.

Das Projekt eignet sich besonders für **Projekt im Bachelor und Master** und insbesondere auf für **Independend Coursework im AI-Master**.

SharkMessenger und ASAPHub Web GUI

Neben einem Android GUI wäre auch ein HTML GUI für den Messenger sehr sinnvoll. Es wäre aber auch ein HTML GUI für den ASAPHub sinnvoll. Der Hub läuft als Prozess und erlaubt ein Encounter von ASAPPeers über das Internet.

Wir wollen Transparenz. Also wäre es gut, alle Informationen strukturiert zu produzieren, die der Hub sammelt und als HTML zur Verfügung zu stellen.

Das Projekt eignet sich besonders für **Projekt im Bachelor und Master** und insbesondere auf für **Independend Coursework im AI-Master**.

Hub Beschreibungen verteilen

Ein Hub ist ein Prozess. Er ist über einen TCP-Port erreichbar. ASAPPeers können sich mit einem Hub verbinden, eine Liste verbundener Peers erhalten und sich mit diesen über den Hub verbinden.

Wie aber bekommen die ASAPPeer Informationen über die IP-Adresse und die Portnummer der Hubs und wann die Hubs verfügbar sind? Die Hubs werden nicht als permanent laufende Teile einer stabilen Infrastruktur gesehen, sondern als dynamische Komponenten.

SharkPeers bieten bereits die Möglichkeit, Hub Descriptions zu speichern. Es bieten sich drei Ansätze an:

1. **Shark-App**. Es wird eine SharkComponent implementiert. Die Hub-Beschreibungen werden über das Shark-System verteilt.
2. Die Informationen werden über eine **öffentliche Blockchain** verteilt. Das ist technisch zwar kaum notwendig, weil kein Konsens nötig ist, sondern nur die Informationen verteilt werden sollen. Aber man kann ein wenig mit Smart-Contracts arbeiten. Und es entstehen immer neue Ideen, wenn man zwei Technologien zusammen bringt.
3. Am spannendsten aber wäre es imho, wenn man die ASAPHubs also Peers im Sinne einer Blockchain versteht. Viele BC-Implementierungen nutzen die Bibliotheken **libp2p** und **discv**[4|5]. Das sind vereinfachten Implementierungen einer DHT basierend auf Kademlia. Kurz: Sie bieten die Chance, Informationen über Peers (hier den ASAPHubs) in einem dezentralen System zu verteilen. Man würde eine Menge über die technische Basis von Blockchains lernen. Es gäbe jede Menge Stoff zum Nachdenken.

Sharing-App

Wir haben eine PKI. Wir haben ein System mit dem Nachrichten ausgetauscht werden können. Wir haben alles, was man für eine App braucht, mit der man Sharing organisieren kann – nur eben ohne jeden Server.

Sie implementieren eine SharkComponent. Diese muss offenbar erlauben, dass man Entitäten benennen kann, die man teilen will (halten Sie das sehr allgemein, es ist technisch auch irrelevant, ob man ein Fahrrad oder einen Gemeinschaftsraum teilen will). Dann können Peers offenbar sagen, wann sie die Entität haben wollen. Ein Prozess muss ablaufen, der die Reservierung für einen Zeitraum vornimmt. Da es ein pur dezentrales System ist, kann es Konflikte geben. Diese gilt es zu lösen. Das ist der Kern der Arbeit.

Man kann eine Liste der Reservierungszeiten der Entitäten finden.

Ja, richtig, das klingt nach Konsensbildung. Das klingt nach Blockchain light und genau das ist es auch.

Verteilte Daten synchronisieren

Zentralisierte Systeme sind auch deshalb so einfach, weil das Problem verteilter Datenhaltung nicht auftritt. In ASAP/Shark tritt das Problem nicht auf, wenn Peers Daten schreiben, die andere nur lesen können – wie in der PKI. Zertifikate werden erzeugt und verteilt. Das macht keine Probleme.

Die Sharing-App, s.o. wird aber das Problem haben, kann es ggf. aber ignorieren. Es ist aber sehr sinnvoll, dass einmal allgemein anzugehen.

Das Problem ist schnell beschrieben: Mehrere Peers können ein zentrales Datum ändern. In der Sharing App ist das z.B. die Entscheidung, wer eine Ressource reservieren kann. Es gibt offenbar eine Wettlaufbedingung und es gibt offenbar inkonsistente Daten, wenn diese nicht gelöst wird – im

schlimmsten Fall ist mehr als ein Peer der Überzeugung, eine Ressource nutzen zu können; oder alle Peers sind der Meinung, die Ressource ist gebucht, aber tatsächlich wird sie nicht genutzt.

Das Problem ist bekannt. Eine Lösung nennt sich optimistisches Schreiben. Ein Peer änderte Daten und geht optimistisch davon aus, dass das schon gehen wird. Die Einschätzung kann sich ändern.

In jedem Fall muss ein Abstimmungsprozess definiert und implementiert werden. Es wird vorgeschlagen die ExtraData zu nutzen. Das ist eine kleine Datenstruktur, die Key-Value-Paare speichern kann. Jedes ASAP und SharkPeer hat solche eine Struktur. Shark-Apps können sie auch nutzen.

Ziel ist es nun, einen Wrapper zu implementieren, der optimistisch eine Änderung in den Daten vollzieht, diese aber an eine bekannte Gruppe von SharkPeer bekannt gibt. Diese übernehmen die Änderung, oder aber erkennen Konflikte. Eine Basis der Konfliktlösung ist immer die Reihenfolge. Wir können nicht von einer gemeinsamen Zeit ausgehen. Dank der Era haben wir in ASAP eine Möglichkeit Ereignisse, die auf unterschiedlichen Peers stattfanden, in eine zeitliche Reihenfolge zu bringen.

Der Kern der Arbeit besteht in der Beschreibung des Synchronisationsprozesses. Der wird am Ende nicht sehr komplex sein. Die Implementierung wird daher auch nicht sehr groß sein. Der intellektuelle Aufwand steht am Anfang in der Beschreibung des Problems und der klaren Formulierung der Lösungen.

Perfekte Abschlussarbeit.

Offene Themen – Netzwerkebene

ASAP ist ein Routingprotokoll mit denen SharkPeers Daten austauschen und weiter leiten. ASAP ist das IP von Shark wenn man so will. ASAP benötigt lediglich eine Punkt-zu-Punkt-Verbindung zwischen zwei Geräten. Das kann eine z.B. Bluetooth-Verbindung in einem Ad-hoc Netzwerk sein, eine Long Range Wifi Verbindung aber auch eine TCP-Verbindung. Mehr Details: https://github.com/SharedKnowledge/ASAPJava/wiki/IntroConnections

Large-Scale-Ad-hoc-Network (LSAN)

Bluetooth ermöglicht mobile Ad-hoc Netzwerke (MANETs). Bluetooth ermöglicht die Etablierung einer Picozelle in der bis zu acht Geräte Daten austauschen können. Oft beinhalten Picozellen lediglich zwei Geräte, wie z.B. ein Headset und einen Rechner. Die Reichweite von BT liegt bei ca. 10 Metern.

Der Bluetooth-Standard definiert ebenfalls Scatternets. BT Geräte können innerhalb von zwei Picozellen gleichzeitig sein. Ein solches Gerät kann daher als Router dienen, der Daten einer Picozelle in einer andere übertragt. Das ist Routing.

Leider aber bieten BT-APIs diese Fähigkeit selten an, vermutlich unterstützt die Hardware diese Fähigkeit nicht immer. Das ist schade. Aber das kann auch programmiertechnisch gelöst werden und darum geht es in dieser Arbeit. Hier ist die Idee:

Ein BT-Gerät kann in seiner Umgebung nach Geräten suchen und eine maximale Anzahl von Verbindungen aufbauen. Diese maximale Zahl gilt es auf unterschiedlicher Hardware einmal zu testen (in Dokumentationen steht vieles, was nicht immer stimmt). Im realen Einsatz sollte auch

noch die Chance bestehen z.B. ein BT-Headset zu betreiben. Ein Gerät kann als eine variable Anzahl von Verbindungen zu anderen Geräten aufbauen – nennen wir diese Zahl *n*.

Der Aufbau solcher Verbindungen wird von keiner zentralen Stelle koordiniert. Angenommen die Geräte A und B kommen in Empfangsreichweite und haben noch keine Verbindung aufgebaut. Hier nur eine Skizze eines Algorithmus für ein Gerät z.B. A:

1. Sind bereits n Verbindungen aufgebaut?
Nein: weiter mit 2.
Ja: Ist B bereits von A aus über einen anderen Pfad erreichbar, d.h. gibt es eine Gerät Y für den dieser Pfad existiert: A-Y-...-B ? Nein: Abbruch.
Ja: Trenne die Verbindung zu Y und verbinde direkt mit B. (Ziel: Geräte mögen sich mit Geräten verbinden, die in physischer Nähe (Sendereichweite) befinden. Verbindungen über mehrere Hops sollen gelöscht werden. Das führt in der Tendenz zu einer geographische Ausbreitung des Netzes und einer Reduktion der Vernetzung innerhalb eines Gebietes. Wir decken eine wachsende Fläche ab.)

2. Gibt es einen Pfad A-...Z-B?
Nein: Verbinde A mit B.
Ja: Fordere Z auf, die Verbindung mit B zu trennen. Baue eine Verbindung zu B auf. (Ziel: Sie Punkt 1.)

In jedem Fall: Teile neue Verbindung alle erreichbaren Geräten mit.

Weiter mit 3:

3. Finde Geräte X, die vor dem Aufbau einer Verbindung sowohl von B als auch A erreichbar waren. Welche Weg ist länger? A-X oder B-X. Lösche die Verbindung des längsten Weges. Wenn beide gleich lang sind entscheidet der Zufall, z.B. mittels der MAC-Adresse.

Dieser Algorithmus dient dazu, einen zyklenfreien Graphen aufzubauen. Prinzipiell ist die Größe eines solchen BT-basierten Netzwerkes unbegrenzt. Das ist auch nur eine Skizze eines Algorithmus – einer erste Idee, die auch verworfen werden kann.

Als Anwendungsszenario wird die Verknüpfung aller BT-Geräte von Studierenden in einem Seminarraum oder von allen Menschen innerhalb eines S-Bahnwagens gesehen.

Der Algorithmus ist unabhängig von Bluetooth, ließe sich also auch mit Wifi-Direct implementieren. In jedem Fall soll eine prototypische Implementierung erfolgen, die unabhängig vom genutzten Layer2 Protokoll ist. (Hier schlägt die Stunde vom Interfaces und ein paar kleinen Pattern. Software-Engineering ist gut!)

Diese Arbeit arbeitet dem Projekt ASAP/Shark lediglich zu, ist aber inhaltlich völlig unabhängig davon. Ein Einarbeiten in das Thema ist nicht notwendig. Alle ASAP-Apps würde aber sofort mit diesem BT-Ring arbeiten können.

Noch anonymere Kommunikation über Tor

Wenn man schon an einem sicheren dezentralen System baut, sollte man auch über Tor kommunizieren können (https://www.torproject.org/). Können wir leider bisher nicht. Tor hat zum Ziel Empfänger und Sender IP basiert übertragener Daten zu anonymisieren. Es existieren APIs für

TOR, z.B. Stem (https://stem.torproject.org/api.html), Thaliproject (https://github.com/thaliproject/Tor_Onion_Proxy_Library).

Die Aufgabe ist ebenso schnell formuliert wie sie Tiefe hat: Suchen und testen Sie existierenden APIs auf Anwendbarkeit. Es soll mit möglichst vielen APIs gezeigt werden, ob und wie stabil eine Verbindung über das TOR-Netzwerk zwischen zwei Entitäten aufgebaut werden kann. Es sollen Überlegungen zur Messung der Performance angestellt werden. Sie bauen eine Software, die eine Punkt-zu-Punkt-Verbindung zwischen zwei Geräten herstellen kann und zeigen das Senden und Empfanger beliebiger Daten funktioniert. Eine Integration in das Shark/ASAP Projekt ist nicht notwendig.

Wifi

Wir brauchen Wifi als Möglichkeit eines ASAP-Encounters. Das ASAPAndroid Projekt sammelt die Implementierungen der Layer2 Protokolle.

Das Projekt gibt auch einen Rahmen. Es beschreibt wann die Suche nach Wifi Geräten gestartet werden sollen. Sobald eine Punkt-zu-Punkt-Verbindung aufgebaut werden kann, muss gefragt werden, ob die Verbindung aufgebaut werden soll. Ist das der Fall, soll eine Verbindung aufgebaut werden. Am Ende muss ein IO-Streampaar übergeben werden, darüber wird dann ein ASAP-Encounter durchgeführt. Das ist also nicht mehr Zeil der Arbeit.

Sie implementieren also die Etablierung einer Wifi-Verbindung idealerweise unter Nutzung von Servicebeschreibungen, die wie gerade beschrieben mit dem Framework interoperiert und zeigen durch Tests, dass das stabil funktioniert.

Das ist eine sehr klar abgegrenzte Abschlussarbeit.