Project Orion

Jakob Klemm, Dominik Keller

Inhaltsverzeichnis

TODO: Abstract

Vorwort

Die schriftliche Komponente der Maturaarbeit `Projekt Orion` besteht aus drei verschiedenen Teilen. Da die behandelten Themen äusserst komplex und umfangreich sind, verlangen verschiedene Abschnitte der Arbeit verschiedenes Vorwissen und einen verschiedenen Zeitaufwand. Deswegen wurde die schriftliche Komponente in drei Subkomponenten aufgeteilt, wobei sie nach technischem Detailgrad sortiert sind. Wer nur ein oberflächliches Verständnis über die Arbeiten und eine Analyse des Umfelds will, ohne dabei zu technisch zu werden, muss nicht über den Umfang dieses Dokuments hinaus. Aber für vollständigen Einblick in die Errungenschaften und Konzepte muss mit einem grösseren Aufwand gerechnet werden.

- Schriftlicher Kommentar: In diesem Dokument hier findet sich eine klassische Besprechung der Arbeit. Begonnen mit einer Zielsetzung und Besprechung verschiedener Projekte, bis zur Analyse des Produkts und einem Ausblick in die Zukunft gibt dieses Dokument einen guten, aber oberflächlichen Einblick in das `Projekt Orion`. Natürlich wird besonders bei der Analyse der existierenden Projekten und Darlegung des Konzepts gewisses technisches Know-How benötigt, aber es wurde versucht, alle Fachbegriffe zu umschreiben oder zu erklären. Wer nur über die Vision und den aktuellen Stand wissen will muss nicht über dieses Dokument hinaus, aber verschiedene Konzepte und nahezu die gesamte technische Umsetzung befinden sind nicht in diesem Dokument.

- Dokumentation: In dieser alleinstehenden Dokumentation, welche im Detailgrad zwischen dem schriftlichen Kommentar und der Code-Dokumentation steht, werden die Konzepte und Ideen besprochen. Wer die Entstehung und aktuelle Form der Komponenten genauer verstehen will, oder wer von den umgesetzten Funktionen profitieren will, sollte die Dokumentation durcharbeiten. Das Dokument ist eher umfangreich, es kann aber auch gut als eine Art Nachschlagewerk verwendet werden.

- Code: Neben der Dokumentation des Projekts und der Konzepte, existiert eine weitere Form der Dokumentation. Nahezu jede Funktion, jedes Modul und jedes Objekt über die verschiedenen *Crates* sind dokumentiert. Diese Dokumentationen lassen sich nicht in einem klassisch strukturierten Dokument finden. Stattdessen ist die Code-Dokumentation online über automatisch generierte Rust-Dokumentation zu finden. Die Seiten mögen anfangs etwas unübersichtlich wirken, wer aber den Code von `Projekt Orion` verwenden will wird sich dort gut zurecht finden.

1 Vision

1. Oktober 1964 - UCLA an Stanford: LO

1964 rechnete niemand mit der fundamentalen Änderung unserer Existenz und Lebensweise, die mit dieser einfachen Nachricht in Bewegung gebracht wurde. Eigentlich hätte die erste Nachricht über das `ARPANET` im Jahre 1964 `LOGIN` heissen sollen, doch das Netzwerk stürzte nach nur zwei Buchstaben ab. Ob dies als schlechtes Omen für die Zukunft hätte gewertet werden sollen bleibt eine ungeklärte Frage. Aber das Internet ist hier und es ist so dominant wie noch nie zuvor. Jetzt ist es in der Verantwortung jeder neuen Generation auf diesem Planeten, mit den unglaublichen Möglichkeiten richtig umzugehen und die Vielzahl an bevorstehenden Katastrophen und Gefahren zu navigieren.

Ohne das Internet wäre die Welt wie wir sie kennen nicht möglich. Unsere Arbeit, Kommunikation und unser Entertainment sind nicht einfach nur abhängig von der enormen Interkonnektivität des Internets, ohne sie würden ganze Industrien und Bereiche unser Gesellschaft gar nicht erst existieren. Das Internet hatte einen selbstverstärkenden Effekt auf sein eigenes Wachstum. Der um ein Vielfaches schnellere Datenaustausch und die enorme Interkonnektivität führten dazu, dass jede neue Innovation und jede neue Plattform im Internet noch schneller noch mehr User erreichte und auf immer unvorstellbarere Grössen anwuchs.

Das ist ja grundsätzlich nichts Schlechtes. Das Internet hat eine unvorstellbare Menge an Vermögen, Geschwindigkeit und Bequemlichkeit für uns alle geschaffen und wir haben unsere Gesellschaftsordnung daran ausgerichtet. Aber man muss sich fragen, ob wir manche der Schritte nicht doch überstürzt haben. Im Namen des Wachstums und aus `FOMO` (*Fear Of Missing Out*) wurden Technologien für die Massen zugänglich, die eigentlich nie für solche Grössenordnungen entwickelt wurden. Denn sobald die immer höheren Erwartungen an teils unglaublich fragile Systeme nicht mehr erfüllt werden, kommt es schnell zur Katastrophe. Und durch unsere Abhängigkeit von diesen Systemen steht bei einem solchen Szenario nicht nur der Untergang einiger Produkte oder einzelner Firmen bevor, nein, es könnte zum Kollaps ganzer Länder oder Gesellschaften kommen.

Egal wie sicher und zuverlässig unsere *öffentliche* Infrastruktur auch scheinen mag, es lassen sich doch schnell Risse im System erkennen. Nicht nur

an der Oberfläche, sondern auch im Herzen unseres digitalen Leben gibt es Probleme. Oftmals handelt es sich dabei nicht um *Kleinigkeiten*, *Meinungsverschiedenheiten* oder *Kontroversen*, sondern um physikalische Grenzen, grundlegende Designfehler und das vielseitige Versagen der involvierten Parteien.

In den nächsten Kapiteln sollen einige dieser zentralen Probleme besprochen werden. Dabei soll versucht werden nicht nur die fehlerhaften Implementierungen zu erklären, sondern auch die dadurch entstandenen Probleme in Verbindung mit unseren täglichen Interaktionen und Verwendungen des Internets zu bringen. In einem nächsten Schritt soll dann eine Lösung besprochen werden: ein System, mit welchem sich möglichst viele der grössten Probleme lösen lassen, und welches tatsächlich praktischen Nutzen bietet.

1.1 Adressen

Das Internet erlaubt einfache, standardisierte Kommunikation zwischen Geräten aller Art. Egal welche Funktion oder Form sie auch haben mögen, es braucht nicht viel um ein Gerät mit dem Internet zu verbinden. Nebst den benötigten Protokollen, hauptsächlich `TCP` und `UDP` wird eine `IP-Addresse` als eindeutige Identifikation benötigt. Während vor dreisig Jahren wunderbare Systeme und Standards geschaffen wurden, welche seither die Welt grundlegend verändert haben, so gibt es doch einige fundamentale Probleme und Limitierungen, welche zu immer grösseren Problemen für unsere verbundene Welt werden.

1.1.1 IP-V4

In der Geschichte der Menschheit haben wir aus vielen verschiedenen Gründen Krieg geführt. Für Wasser, Nahrung, Öl, Frieden oder Freiheit in den Krieg zu ziehen, scheint zu einer fernen Welt zu gehören. Aber auch wenn diese grundlegenden Verlangen gedeckt sind, werden schon bald neue Nöte aufkommen. Während *Daten* oft als Gold des 21. Jahrhunderts bezeichnet werden, gibt es noch eine andere Ressource, deren Vorräte wir immer schneller erschöpfen.

$4'294'967'296$. So viele `IP-V4`-Adressen wird es jemals geben. `IP-V4`-Adressen werden für jedes Gerät benötigt, das im Internet kommunizieren will und dienen zur eindeutigen Identifizierung. Aktuell wird die vierte Version (`V4`) ver-

wendet. In einer Wirtschaft, in der unendliches Wachstum als letzte absolute Wahrheit geblieben ist, kann ein solch hartes Limit verheerende Folgen haben. Besonders wenn die limitierte Ressource so unendlich zentral für unser aller Leben ist, wie nichts anderes. Mit `IP-V6` wird zurzeit eine Alternative angeboten, die solche Limitierungen nicht hat. Aber der Wechsel ist eine freiwillige Entscheidung, für die nicht nur alle Betroffenen bereit sein müssen, sondern für die auch jede einzelne involvierte Komponente diese neue Technologie unterstützen müssen.

Für jeden einzelnen kann dies verschiedene Konsequenzen haben:

- Die Preise der Internetanbieter und Mobilfunkabonnemente wird wahrscheinlich langfristig steigen, sobald die ehöhten Kosten für neue Addressen bis zum Endnutzer durchsickern.
- Ein aufwengdiger technologischer Wandel wird langfristig von Nöten sein, welchen jeden einzelnen dazu zwingt auf neue Standards umzusteigen. Eine solche Umstellung wird den häufigen Problemen grossflächiger technischer Umstellungen nicht ausweichen können.

1.1.2 Routing

Freiheit und Unabhängigkeit sind menschlich. Es darf niemals bestraft werden, nach diesen fundamentalen Rechten zu streben. Und doch führt das egoistische Streben nach Freiheit zu Problemen, oftmals allerdings nicht für die nach Freiheit Strebenden.

Genau diese Situation findet man im aktuellen Konflikt um die Grösse von *Address-Abschnitten* vor. Um dieses Problem richtig zu verstehen, muss als erstes die Funktion der *Zentralrouter* und der globalen Netzwerkinfrastruktur erklärt werden:

Jedes Gerät im Internet ist über Kabel oder Funk mit jedem anderen Gerät verbunden. Da das Internet aus einer Vielzahl von Geräten besteht, wäre es unmöglich, diese direkt miteinander zu verbinden. Daher lässt sich das Internet besser als *umgekehrte Baum-Struktur* vorstellen:

- Ganz unten finden sich die Blätter, die Abschlusspunkte der Struktur. Sie stellen die *Endnutzergeräte* dar. Jeder Server, PC und jedes `iPhone`. Hier ist es auch wichtig festzustellen, dass es in dieser Ansicht

des Internets keine magische *Cloud* oder ferne Server und Rechenzentren gibt. Aus der Sicht des Netzwerks sind alle Endpunkte gleich, auch wenn manche für Konsumenten als *Server* gelten.

- Die Verzweigungen und Knotenpunkte über den Blättern, dort wo sich Äste aufteilen, stellen *Router* und Switches dar. Hier geht es allerdings nicht um Geräte, die sich in einem persönlichen Setup oder einem normalen Haushalt finden. Mit Switches sind die Knotenpunkte (`POP-Switches`) der Internet-Anbieter gemeint. Diese teilen eingehende Datenströme auf und leiten die richtigen Daten über die richtigen Leitungen.

- Ganz oben findet sich der Stamm. Während ein normaler Baum natürlich nur einen Stamm hat, finden sich in der Infrastruktur des Internets aus Zuverlässigkeitsgründen mehrere. Von diesen `Zentralroutern` gibt es weltweit nur eine Handvoll und sie sind der Grund für das Problem.

Die `Zentralrouter` kümmern sich nicht um einzelne Adressen, sondern um Abschnitte von Adressen, auch `Address Spaces` genannt. An den zentralen Knotenpunkten geht es also nicht um einzelne Server oder Geräte, zu dem etwas gesendet werden muss, stattdessen wird eher entschieden, ob gewisse Daten beispielsweise von Frankfurt aus nach Ost- oder Westeuropa geschickt werden müssen.

Im Laufe der Jahre wurden die grossen Abschnitte von Adressen aber immer weiter aufgeteilt. Internet-Anbieter und grosse Firmen können diese Abschnitte untereinander verkaufen und aufteilen. Und jede Firma will natürlich ihren eigenen Abschnitt, ihren eigenen `Address Space`. Für die Firmen hat dies viele Vorteile, beispielsweise müssen weniger Parteien beim Finden des korrekten Abschnitts involviert sein. Aber für die `Zentralrouter` bedeutet es eine immer grössere Datenbank an Zuweisungen. Dieses Problem geht so weit, dass die grossen *Routingtables* inzwischen das physikalische Limit erreichen, was ein einzelner Router verarbeiten kann.

1.2 Zentralisierung

Die Macht in den Händen einiger weniger Kapitalisten und internationaler Unternehmen ist unvorstellbar gross. Einige wenige CEO's, welche nie gewählt, überprüft oder zur Rede gestellt wurden, sind in voller Kontrolle

unserer Leben. Egal welcher politischen, wirtschaftlichen oder gesellschaftlichen Ideologie jemand auch folgt, eine solche Abhängigkeit wirft gewisse Fragen und Probleme auf.

Aber neben den ideologischen Fragen und Sicherheitsbedenken gibt es auch noch sehr praktische Probleme in der Art, wie moderne Internet-Dienste implementiert sind.

1.2.1 Datenschutz

Wenn man nicht für etwas zahlt, ist man das Produkt.

Nach dieser Idee ist man für ziemlich viele Firmen ein Produkt. Doch leider muss man realisieren, dass man selbst bei kostenpflichtigen Diensten als Produkt gesehen wird. Denn das Internet hat einen neuen Rohstoff zur Welt gebracht. Wer viele Daten über Menschen besitzt, bekommt binnen kürzester Zeit Macht.

In ihrer einfachsten Funktion werden Daten für personalisierte Werbung eingesetzt. Damit lassen sich Werbungen zielgerichtet an Konsumenten schicken und der Umsatz, sowohl für Firmen als auch für Anbieter, optimieren.

Werbung ist mächtig und hat einen grossen Einfluss auf den Markt. Aber damit lassen sich lediglich Konsumenten zu Käufen überzeugen oder davon abbringen. Wenn man dies mit dem tatsächlichen Potential in diesen Daten vergleicht, merkt man schnell, wie viel noch möglich ist. Denn die Daten die sich täglich über uns im Internet anhäufen, zeigen mehr als unser Kaufverhalten. Von Echtzeit-Positionsupdates, Anrufe und Suchanfragen bis hin zu privaten Chats und unseren tiefsten Geheimnissen, sind wir meist überraschend unvorsichtig im Umgang mit digitalen Werkzeugen.

Während man davon ausgehen muss, dass Firmen, deren Haupteinnahmequelle Werbungen ist, unsere Daten sammeln und verkaufen, gibt es eine Vielzahl an anderen Firmen, die ebenfalls unsere Daten sammeln, obwohl man von den meisten dieser Firmen noch nie gehört hat. Die Liste der potentiellen Mithörer bei unseren digitalen Unterhaltungen ist nahezu unendlich: Internet-Anbieter, DNS-Dienstleister, CDN-Anbieter, Ad-Insertion-Systeme, Analytics-Tools, Knotenpunkte & Datencenter, Browser, Betriebssysteme, ...

Aus dieser Tatsache heraus lassen sich zwei zentrale Probleme formulieren:

- Selbst für die einfachsten Anfragen im Internet sind wir von einer Vielzahl von Firmen und Systemen abhängig. Dieses Problem wird noch etwas genauer im Abschnitt Abhängigkeit besprochen.
- Wir haben weder ein Verständnis von den involvierten Parteien noch die Bereitschaft, Bequemlichkeit dafür aufzugeben.

1.2.2 Abhängigkeit

In einem fiktionalen Szenario[1] erklärt *Tom Scott* auf seinem YouTube-Kanal was passieren könnte, wenn eine einzelne Sicherheitsfunktion beim Internetgiganten `Google` fehlschlagen würde. In einem solchen Fall ist es natürlich logisch, dass es zu Problemen bei den verschiedensten `Google`-Diensten kommen würde. Aber schnell realisiert man, auf wie vielen Seiten Nutzer die *Sign-In with Google* Funktion benutzen. Und dann braucht es nur eine böswillige Person um den Administrator-Account anderer Dienste und Seiten zu öffnen, wodurch die Menge an Sicherheitsproblemen exponentiell steigt.

Aber es muss nicht immer etwas schief gehen, um die Probleme zu erkennen. Sei es politische Zensur, *Right to Repair* oder *Net Neutralität*, die grossen Fragen unserer digitalen Zeit sind so relevant wie noch nie.

Während die enorme Abhängigkeit als solche bereits eine Katastrophe am Horizont erkennen lässt, gibt es noch ein konkreteres Problem: Den Nutzern (*den Abhängigen*) ist ihre Abhängigkeit nicht bewusst. Wenn sie sich ihren Alltag ohne `Google` oder `Facebook` vorstellen, denken sich viele nicht viel darunter. Weniger *lustige Quizfragen* oder Bilder von Haustieren, aber was könnte den schon wirklich Schlimmes passieren?

Während es verständlich ist, dass das Benutzen von `Google` natürlich von `Google` abhängig ist, so versteht kaum jemand, wie viel unserer täglichen Aktivitäten von Diensten und Firmen abhängen, die selbst wieder von `Google` abhängig sind. Seien es die `Facebook`-Server, durch welche keine Whatsapp-Nachrichten geschickt werden konnten, oder die fehlerhafte Konfiguration bei `Google`, durch welche manche Kunden die Temperatur ihrer Wohnungen

[1] Tom Scott: Single Point of Failure `https://youtu.be/y4GB_NDU43Q`, heruntergeladen am 24.05.2020.

auf ihren Nest Geräten nicht mehr anpassen konnten[2], das Netz aus internen Verbindungen zwischen Firmen ist komplex und undurchschaubar, und nicht nur für die Entnutzer, da oftmals die Firmen selbst von kleinsten Problemen anderer Dieste überrascht werden können. Der wirtschaftliche Schaden solcher Ausfälle sind unvorstellbar, aber noch wichtiger muss die zerstörende Wirkung dieser auf unvorhergesehenen, entfernten Problemen auf millionen von Meschen bedacht werden.

1.3 Komplexität

In diesem Abschnitt soll noch kurz die unglaubliche Komplexität angesprochen werden, welche die häutige Web-Entwicklung mit sich bringt. Natürlich existieren automatisierte Dienste und Anbieter, die den Prozess vereinfachen. Wer aber Wert auf seine Privatsphäre und auf die Verwendung von open-source Software legt, muss sich um vieles selbst kümmern. Nicht nur die Auswahl an verschiedenen Programmen kann erschlagend wirken, sondern der Fakt, dass diese untereinander kompatibel sein müssen. Zwar reden wir oft von einem Webserver, allerdings sind es tatsächlich viele verschiedene Programme, die alle fehlerfrei miteinander interagieren müssen, um Resultate zu liefern. Dies kann den Einstieg schwerer machen, oder, in gefährlicheren Fällen kann es dazu führen, dass Sicherheit und Datenschutz aus Zeit- oder Komplexitätsgründen weggelassen oder vernachlässig werden.

Dabei geht es oben nur um *klassische* Webseiten oder Webserver. Die Welt der dezentralen Technologien ist im vergleich dazu wie der wilde Westen, ohne Standards, ohne Kompatiblität oder Regelungen. Dies führt dazu, dass es zwar für gewisse Anwendungen speziell entwickelte Netzwerke gibt, diese allerdings kaum allgeimen einsetzbar sind.

1.4 Präsentation

Ein weiteres Problem, dass es zu berücksichtigen gibt, ist die Frage, wie man die hier behandelten Probleme technisch nicht versierten Personen erklären kann. Tatsächlich sind sowohl die besprochenen Probleme, als auch deren Lösungsansätze nicht nur abstrakt, sondern dazu noch Teil einer kleinen Nische in der Welt der Informatik. Manche der obigen Probele wurden bereits von anderen Applikationen zumindest teilweise behandelt, diese haben aber

[2]Fastcompany: Google outage, heruntergeladen am 24.10.2021. `https://www.fastcompany.com/90358396/that-major-google-outage-meant-some-nest-users-couldnt-unlock-doors-or-use-the-ac`

oftmals das Problem, dass sie viel Fachwissen und Aufwand benötigen, um sie effizient und sicher einzusetzen.

2 Prozess

2.1 Modularität

Tatsächlich beschreibt dieses Dokument bereits den zweiten Versuch, die zweite Iteration eines dezentralen Datensystems. Da während dieser ersten Entwicklungsphase viele Lektionen gelernt wurden, ist es wichtig die Ideen und die Umsetzung genau zu analysieren. Zwar unterscheiden sich die Ziele und Methoden der beiden Ansätze stark, gewisse Konzepte und einige Programme lassen sich für die aktuelle Zielsetzung genau übernehmen.

Als erstes ist es wichtig, die Zielsetzung des Systems, welches hier einfach als "Modularer Ansatz" bezeichnet wird, zu verstehen und die damit entstandenen Probleme genau festzuhalten.

- Modularität
  Wie der Name bereits verrät, ging es in erster Linie um die Modularität. Ziel war also eine Methode zur standardisierten Kommunikation, durch welche dann beliebige Komponenten an ein grösseres System angeschlossen werden können. Mit einigen vorgegebenen Komponenten, die Funktionen wie das dezentrale Routing und lokales Routing abdecken, können Nutzer für ihre Anwendungszwecke passende Programme integrieren.

- Offenheit
  Sobald man den Nutzern die Möglichkeit geben will, das System selbst zu erweitern und zu bearbeiten, muss man quasi zwingend open-source Quellcode zur verfügung stellen.

Die grundlegende Idee war die selbe: *Die Entwicklung eines dezentralen vielseitig einsetzbaren Kommunikationsprotokoll.* Da allerdings keine einzelne Anwendung angestrebt wurde, ging es stattdessen um die Entwicklung eines vollständigen Ökosystems und allgemein einsetzbare Komponenten.

Im nächsten Abschnitt sollen einige dieser Komponenten und die Entscheidungen die zu ihnen geführt haben beschrieben werden. In einem weiteren Abschnitt sollen dann die Lektionen und Probleme dieses erster ersten Entwicklungsphase besprochen werden.

2.1.1 Shadow

Zwar übernahm die erste Implementierung des verteilten Nachrichtensystems, Codename `Shadow` weniger Funktionen als die aktuelle Umsetzung, für das System als ganzes war das Programm aber nicht weniger wichtig. Der Name lässt sich einfach erklären: für normale Nutzer sollte das interne Netzwerk niemals sichtbar sein und sie sollte nie direkt mit ihm interagieren müssen, es war also quasi *im Schatten*. Geschrieben in `Elixir` und mit einem `TCP`-Interface konnte `Shadow` sich mit anderen Instanzen verbinden und über eine rudimentäre Implementierung des `Kademlia`-Systems Nachrichten senden und weiterleiten. Um neue Verbindungen herzustellen wurde ein speziell Entwickeltes System mit so genannten *Member-Files* verwendet. Jedes Mitglied eines Netzwerks konnte eine solche Datei generieren, mit welchen dann beliebige andere Instanzen beitreten konnten.

Sobald eine Nachricht im System am Ziel angekommen war, wurde sie über einen `Unix-Socket` an den nächsten Komponenten im System, meistens also `Hunter` weitergegeben. Dies geschah nur, wenn das einheitlich verwaltete Registrierungssystem für Personen und Dienste, eine Teilfunktion von `Hunter`, ein treffendes Resultat lieferte. Ansonsten wurde der interne Routing-Table verwendet. Dieser bestand aus einer Reihe von Prozessen, welche selbst auch direkt die `TCP`-Verbindungen verwalteten.

2.1.2 Hunter

Während `Shadow` die Rolle des verteilten Routers übernimmt, ist `Hunter` der lokale Router. Es geht bei `Hunter` also nicht darum, Nachrichten an andere Mitglieder des Netzwerks zu senden, sondern sie an verschiedene Applikationen auf der gleichen Maschine zu senden. Jedes beliebige Programm, unabhängig von Programmiersprache & internen Strukturen müsste dann also nur das verhältnismässig Protokoll implementieren und wäre damit in der Lage, mit allen anderen Komponenten zu interagieren. Anders als `Shadow` wurde `Hunter` komplett in Rust entwickelt und liess sich in zwei zentrale Funktionen aufteilen:

- Zum einen diente das Programm als Schnittstelle zu einer einfachen *Datenbank*, in diesem Fall eine `JSON-Datei`. Dort wurden alle lokal aktiven Adressen und die dazugehörigen Applikationen gespeichert. Ein Nutzer der sich beispielsweise über einen Chat mit dem System verbindet wird dort mit seiner Adresse oder seinem Nutzernamen und dem Namen des Chats eingetragen. Wenn dann von einem beliebigen an-

deren Punkt im System eine Nachricht an diesen Nutzer kommt, wird der passende Dienst aus der Datenbank gelesen. All dies lief durch ein *Command Line Interface*, welches dann ins Dateisystem schreibt.

- Das eigentliche Senden und Weiterleiten der Nachrichten war nicht über ein kurzlebiges Programm möglich, da dafür längere Verbindungen existieren müssten. Dafür muss `Hunter` als erstes gestartet werden, wobei das Programm intern für jede Verbindung einen dedizierten Thread startet.

Diese klare Trennung der Aufgaben und starke Unabhängigkeit der einzelnen Komponenten erlaubt ein einheitliches Nachrichtenformat, da für die einzelnen Komponenten kein Verständnis über andere Komponenten oder die Verbindungen haben müssen.

2.1.3 NET-Script

Ein weitere zentrale Komponente des Systems ist eine eigene Programmiersprache, welche mit starker Integration in das restliche System das Entwickeln neuer Mechanismen und Komponenten das System offener machen sollte. Ein einfacher lisp-ähnlicher Syntax sollte das Entwickeln neuer Programme einfach und vielseitig einsetzbar machen.

2.1.4 Interface

TODO: Interface

2.1.5 Probleme

Die oben beschriebene Architektur hat viele verschiedene Vorteile, allerdings ist sie nicht ohne Probleme. Grundsätzlich geht es bei jedem Programm darum, Probleme zu lösen. Einer der zentraler Ideen war die Modularität, welche es Nutzern erlauben soll, die verschiedenen Komponenten des Systems einfach zu kombinieren. Und auch wenn dieses Ziel auf einer technischen Ebene erfüllt wurde, so ist die Umsetzung alles andere als *einfach*. Die Anzahl möglicher Fehlerquellen steigt mit jedem eingebundenen Komponenten exponentiell an, und wenn mindestens vier der Komponenten für selbst die einfachsten Demos benötigt werden, kann nahezu alles schiefgehen. Dazu kommt, dass viele Fehler nicht richtig isoliert und verarbeitet würden, weswegen sich die Probleme durch das System weiter verbreiten würden. Während die Umsetzung also ihre eigentlichen Ziele erfüllt hatte, war sie noch lange davon

entfernt, für tatsächliche Nutzer einsetzbar zu sein.

Trotzdem wurden die beschriebenen Komponenten vollständig entwickelt, getestet und vorgeführt. Zwar war es umständlich und nur bedingt praktisch einsetzbar, trotzdem war es aber eine technisch neuartige, funktionsfähige Lösung für komplexe und relevante Probleme. Nachdem die erste Entwicklungsphase erfolgreich abgeschlossen wurde, kam allerdings noch ein weiteres Problem auf, welches die folgenden Entscheidungen stark beeinflusst hat, und es ist ein Problem welches sich auf die grundlegende Natur der Informatik zurückführen lässt:
Anders als nahezu alle Studienrichtungen, Wissenschaften und Industrien, werden in der Informatik die gleichen Werkzeuge verwendet und entwickelt. Wer die Werkzeuge der Informatik verwenden kann, ist gleichzeitig in der Lage (zumindest bis zu einem gewissen Grad) neue Werkzeuge zu entwickeln. Diese Eigenschaft erlaubt schnelle Iterationen und viele, fortschrittliche Werkzeuge, so kommen gleichzeitig neue Probleme auf:

- Neue Methoden und Werkzeuge werden mit unglaublicher Geschwindigkeit entwickelt und verbreitet. Wer also nicht mit den neusten Trends mithält kann schnell abgehängt werden. Dies macht auch das Unterrichten besonders schwer.

- Natürlich werden die Werkzeuge meistens immer besser und schneller, allerdings kommt es oftmals auch zu einer Spezialisierung. Dies führt schnell zu immer spezifischeren, exotischeren Lösungen und unzähligen Unterbereichen und immer kleineren Gebieten. So beispielsweise der Begriff *dezentrale Datensysteme*, der zwar ein einzelnes Gebiet genau beschreibt, für Aussenstehende ist er aber mehrheitlich bedeutungslos und sorgt für mehr Verwirrung als Aufklärung.

- Die immer neuen Gebiete und Gruppen können auch schnell zu Elitismus führen, wodurch es für Anfänger schwer sein kann, Zugang zu finden.

Diese Eigenschaften, besonders bei unseren sehr neuartigen Ideen und Mechanismen, machten es unglaublich schwer, Aussenstehenden die Funktionen und Konzepte zu erklären. Ohne Vorkenntnisse über Netzwerke und Kommunikationssysteme war es nahezu unmöglich, auch nur die einfachsten Ideen zu erklären oder den Inhalt dieser Arbeit darzulegen. Und selbst mit grossem Vorwissen liessen sich nur die absoluten Grundlagen innerhalb absehbarer Zeit erklären, das Erklären der theoretischen und technischen Grundlagen

würde Stunden in Anspruch nehmen.

Da am Ende dieser Arbeit zwingend eine zeitlich begrenzte Präsentation vor einem technisch nicht versierten Publikum stehen würde, mussten nach dieser ersten Entwicklungsphase gewisse Aspekte grundlegend überarbeitet werden, diesmal mit einem besonderen Fokus auf die *präsentierbarkeit*.

2.2 Präsentation

Auch wenn von der ersten Entwicklungsphase viele Konzepte und sogar einige Umsetzungen übernommen werden konnten, so gab es grundlegende Probleme, welche nicht ignoriert werden konnten. Es wurde schnell klar, dass unabhängig aller technischer Fortschritte eine bessere Art der Präsentation gefunden werden musste. Dabei ist es wichtig, die technischen Neuerungen und Besonderheiten nicht zu vergessen. Die Umsetzung der ersten Entwicklungsphase, wie innovativ und attraktiv sie auch wirken mag, ist noch weit davon entfernt, von Endnutzern verwendet oder gar angepasst zu werden. Auch wenn manche der Ideen hier wieder aufgegriffen werden, musste doch ein grösserer Fokus auf die *Präsentierbarkeit* der Fortschritte gelegt werden. Daher wurde die Entscheidung getroffen, die Entwicklung in zwei Bereiche zu unterteilen:

- Ein möglichst vielseitig einsetzbares Nachrichtensystem basierend auf den bereits bekannten Prinzipien wird als Bibliothek für die Anwendungen sowie öffentlich angeboten. Entwickelt in Rust soll Geschwindigkeit und Sicherheit garantiert werden und möglichst viele Möglichkeiten bieten, in andere Applikationen eingebunden zu werden.
- Anwendungen:

3 Produkt

3.1 Actaeon

3.2 Orion

3.3 Anwendungen

4 Ausblick

4.1 Verifizierung

4.2 Balance

4.3 Anwendungen