Aufgabenblatt 4[1]

IP, Routing und ARQ Programmierung

Ziele

- IP-Adressen vergeben und Subnetze bilden
- Routen konfigurieren und mit `mininet` überprüfen
- ARQ Protkolle kennenlernen und implementieren
- Eigenes Anwendungsprotokoll entwerfen und implementieren

Ergebnis

- Maximal 3-seitiges Praktikumsprotokoll (deutsch oder englisch), das die angeforderten Teilaufgaben umfasst und in geeigneter Form darstellt. Eine Auflistung von Stichpunkten ist nicht ausreichend. Es sollte mindestens ein UML-Diagramm pro Implementierungsaufgabe im Protokoll Verwendung finden. Das Protokoll muss von einer dritten Person ohne Vorlage der Aufgabe verstanden werden können.
- Bearbeiten Sie das Aufgabenblatt bereits vor Beginn des Praktikumstermins vollständig. Halten Sie das Protokoll und Ihren Lösungsansatz zum Abnahmegespräch bereit. Es ist erforderlich, dass Sie Ihren Bildschirm während des Abnahmegesprächs teilen. Überprüfen Sie die *Screen-Share*-Funktionalität vor Praktikumsbeginn in ihrem Team.
- Bei nicht ausreichender Leistung wird vor dem nächsten Praktikumstermin eine weitere Nacharbeitung zugelassen. Eine Nichterfüllung im angesetzten Zeitrahmen führt zum Ausschluss.
- Das Protokoll inkl. des Lösungsansatzes ist innerhalb einer Woche nach dem Praktikumstermin abzugeben. Details zur Abgabe finden Sie im MS Teams Raum *SoSe21-B-AI4-RN* im Kanal *Praktikum*.

A Pflichtteil

A.1 Routing

Sie verfügen über 2 Netze, die über einen Router miteinander verbunden sind. Netz 1 ist beschrieben mit 10.0.1.0/24 und Netz 2 mit 10.1.0.0 und Netzmaske 255.255.0.0. Bitte zeichnen Sie einen Netzplan mit geeigneten Symbolen. Stellen Sie für alle Teilnehmer sicher, das ein Ping gelingen kann. Entwickeln sie daher alle Routing-Tabellen und Netzkonfigurationen. Adressen und Routen dürfen frei gewählt werden, sind aber im Netzplan anzugeben.

- Geben Sie alle Routing Tabellen an.
- Geben sie jede Host Konfiguration an.
- Geben sie die Befehle an, die sie zur Anlage der Routen benötigen.
- Überprüfen Sie Ihre Konfiguration mit `mininet`[2].

Protokollieren Sie ihren Netzwerkentwurf und führen Sie ihre Experimente in `mininet` durch.

[1] basiert auf der Aufgabenstellung von Prof. Dr. Martin Becke

[2] siehe `http://mininet.org/`

A.2 Zuverlässiges File Transfer Protokoll über UDP

In dieser Aufgabe soll ein zuverlässiges File Transfer Protokoll über UDP in *C/C++*[3] implementiert werden.

A.2.1 Basic File Transfer Protocol

Ziel ist es einen zuverlässigen Dienst über ein unzuverlässiges Medium anzubieten. Zur Gewährleistung einer zuverlässigen Datenübertragung werden ARQ-Protokolle genutzt. In dieser Aufgabe soll das ARQ-Protokoll *Stop-And-Wait* (SAW) implementiert werden.

Das zu implementierende File Transfer Protokoll soll den Namen *Basic File Transfer* (BFT) tragen. Die Clientanwendung `bft.Client` soll Dateien auf den Server `bft.Server` in das Serververzeichnis hochladen können. Das Serververzeichnis wird über das Argument `DIRECTORY` (siehe Listing 2) angegeben. Ist das Serververzeichnis nicht vorhanden, soll dieses vom Server erzeugt werden. Das Herunterladen von Dateien vom Server oder eine Ressourcenauflistung muss das BFT-Protokoll nicht anbieten. Der `bft.Client` soll nach der erfolgreichen Übertragung der Datei terminieren. Der `bft.Server` soll in einer Dauerschleife auf Clientanfragen warten und kann durch das Signal `SIGINT` ordentlich terminiert werden. Der Server `bft.Server` darf nicht durch Client- oder Netzwerkfehler unerwartet terminieren und den Dienst quittieren. Bei `bft.Client` und `bft.Server` handelt es sich um zwei separate Programme, die über keinen gemeinsamen Speicher verfügen.

Der maximale Payload eines UDP Datagramms soll bei *512B* liegen. Es müssen nur Dateien bis zu einer maximalen Dateigröße von *4GiB* hochgeladen werden können. Der Name der zu übertragenden Datei soll dem Server mitgeteilt werden, damit dieser die übertragene Datei unter dem gleichen Namen abspeichern kann. Es müssen nur Dateinamen bis zu einer Länge von *16 Character* gesendet werden können. Das BFT-Protokoll soll binär kodiert (engl. *binary encoded*)[4] sein.

Listing 1 zeigt die Usage-Informationen des `bft.Client`s. Die Standardeinstellung für den *Retransmission Timeout* (RTO) soll auf 2 Sekunden gesetzt werden. Der *Retransmission Timeout* soll über den Schalter `-r` konfigurierbar sein. Ihre Anwendungen sollen die Anzahl der *Retransmissions*, *Duplicates* und die Dauer der Datenübertragung protokollieren. Listing 2 zeigt die Usage-Informationen des `bft.Server`s.

```
1 Usage: java bft.Client [OPTIONS...] SERVER PORT FILE
2 where
3   OPTIONS := { -d[ebug] | -h[elp] | -r retrans_timeout_ms }
```

Listing 1: `bft.Client` Usage Information

```
1 Usage: java bft.Server [OPTIONS...] PORT DIRECTORY
2 where
3   OPTIONS := { -d[ebug] | -h[elp] }
```

Listing 2: `bft.Server` Usage Information

Zudem soll über den Schalter `-d` bzw. `-debug` der Debug-Modus aktiviert werden. Ist der Debug-Modus aktiv sollen sinnvolle Debug-Nachrichten nach `STDERR` geschrieben werden. Der Schalter `-h` bzw. `-help` soll einen erweiterten Hilfetext ausgeben.

Das BFT-Protokoll muss nur eine Client-Verbindung zur Zeit bedienen können. Mehrere parallele Client-Verbindungen müssen nicht unterstützt werden. Die Implementierung einer Flusskontrolle (engl. *flow control*) oder Überlastkontrolle (engl. *congestion control*) ist ebenfalls nicht erforderlich.

[3] oder *Rust*

[4] Beispiel: JSON ist ein text-basiertes Datenformat. BSON ist die binär-kodierte Alternative zu JSON. (Für weitere Informationen siehe: `https://www.json.org/` und `https://bsonspec.org/`)

A.2.2 Evaluation

Um ein unzuverlässiges Medium zu simulieren soll `mininet` genutzt werden. Hierzu wird ein Experimentaufbau bereitgestellt, auf dem aufgebaut werden soll (siehe `bftp_experiment.zip`).

Demonstrieren Sie die zuverlässige Datenübertragung ihres ARQ-Protokolls, indem Sie die Testdateien über das verlustbehaftete Netzwerk übertragen. Hierzu führen Sie das bereitgestellte Experiment durch.

Führen Sie weitere Experimente mit unterschiedlichen RTOs auf dem Link aus. Wählen Sie eine geeignete Verteilung der Werte zwischen 1ms und 1024ms. Sie können hierzu auf dem mitgelieferten Skript aufbauen. Welche Auswirkung hat die RTO-Einstellung auf die Anzahl der *Retransmissions*, *Duplicates* und auf die Dauer der Datenübertragung? Welche Szenarien sind interessant?

Stellen Sie ihre Beobachtungen grafisch dar. Nutzen Sie das bzw. die Diagramme für Ihre Schlussfolgerungen.

A.2.3 Protokoll

- Protokollieren Sie in geeigneter Form den Aufbau des von Ihnen entwickelten BFT-Protokolls. Als Vorlage können Sie sich beispielsweise den RFC 768 anschauen.
- Protokollieren Sie in geeigneter Form die Implementierung des BFT-Protokolls, der `bft.Client`- und `bft.Server`-anwendung. Nutzen Sie hierzu mindestens ein Struktur- und ein Verhaltensdiagramm.
- Dokumentieren Sie Durchführung, Beobachtungen und Schlussfolgerungen Ihrer Experimente.

Tipps

- Diskutieren Sie in Ihrem Team oder gemeinsam mit anderen Teams über das Design der Header der Transportprotokolle. Welche Informationen sind für das BFT-Protokoll wichtig?
- Denken Sie bei der Implementierung Ihres BFT-Protokolls an den *Divide-and-Conquer*-Ansatz. Versuchen Sie kleinteilig vorzugehen und jeden Schritt zu testen bzw. mit Wireshark zu überprüfen, bevor es zum nächsten Implementierungsschritt übergeht.
- Für viele Experimente bietet es sich an, z.B. ein Skript zu schreiben, welches die Experimente automatisiert durchführt und ggf. direkt geeignete Visualisierungen generiert.
- Eine automatisierte Generierung von Diagrammen würde sich anbieten. Einige Anregungen sind: `matplotlib`[5] (python), `xchart`[6] (Java), `gnuplot`[7] oder `R`[8] . Ein Tabellenkalkulationsprogramm tut es im Notfall jedoch auch.

[5] `https://matplotlib.org/`
[6] `https://knowm.org/open-source/xchart/`
[7] `http://www.gnuplot.info/`
[8] `https://www.r-project.org/`