Verteilte Systeme Laborbericht 1

Gruppe:
Alexander Bekker
Fabian Brenner

TCP:

Die Clients in unserem Simulator sind keine eigene Threads, das senden der Daten wird sequenziell vom Simulator ausgelöst.
Als Server wurde ein Multithreaded TCP Server verwendet, gemessen wurde die Zeit die vom starten des Threads bis zum schließen des Sockets vergeht.
Außerdem wurde getestet wie viele Clients der Server bearbeiten kann und wie lange der Server benötigt eine bestimmte Menge von Clients zu verarbeiten.

Wir fanden heraus das beim Betrieb von 6000 Clients einige Clients nicht mehr in der Lage waren zum Server zu verbinden.
Die Meldung lautete: "java.net.NoRouteToHostException: Die angeforderte Adresse kann nicht zugewiesen werden".
Der Fehler trat aber nur sporadisch auf und der Server war immer noch in der Lage die Verbindungen, die nicht betroffen waren, zu bearbeiten.

Als nächstes die Ergebnisse der Zeitmessung die, die "Lebenszeit" der Threads im Server bestimmte.
Gemessen wurde mit jeweils 100, 3000 und 6000 Clients, die Verbindungszeit wurde nur wenig von der Anzahl der Clients verändert.
Der Großteil der Verbindungen dauerte zwischen 1 und 5 ms, allerdings häuften sich bei steigender Clientzahl die Anzahl von Ausbrüchen einzelner Verbindungen in den 100-200ms Bereich.

Die Zeit zum bearbeiten von Clients wird an einer anderen Stelle dargestellt um sie mit UDP besser vergleichen zu können.

UDP:

Für UDP wurde ein möglichst einfaches Verfahren verwendet. Der Server besteht aus nur einem Thread der ankommende Pakete entgegennimmt überprüft ob diese die geforderten Randbedingungen erfüllen.

Die Clients senden ihr Paket und sonst nichts, es findet keine Wiederholung von verlorenen oder fehlerhaften Paketen statt. Wie bei TCP besteht der Simulator aus nur einem Thread der das Senden der Daten anstößt.
Beim Senden wurden 2 Verfahren verwendet. Beim ersten befahl der Simulator jedem Client (Auto) seine Daten an den Server zu schicken, nachdem der letzte gesendet hatte wurde der ganze Simulator für 5 Sekunden per sleep schlafen gelegt.
Das zweite Verfahren lief ähnlich nur mit dem Unterschied das zwischen jedem Senden der Simulator für 1 Sekunde schlafen gelegt wurde um dem Server Zeit zu geben das Paket zu verarbeiten.
Gemessen wurde die Anzahl der fehlerhaften und verlorenen Pakete, sowie die Zeit die Benötigt wurde um eine bestimmte Anzahl von Clients zu verarbeiten.

Es folgen die zwei Tabellen die, die Fehlerrate von beiden Verfahren mit steigender Clientzahl zeigen.

Verfahren 1:

Clients	Verlustrate	Gesendete Pakete	Fehlerhafte Pakete
1000	~ 50,00%	5000	8
3000	~ 66,00%	15000	55
5000	~ 66,00%	45000	420
8000	>75,00%	40000	412

Verfahren 2:

Clients	Verlustrate	Gesendete Pakete	Fehlerhafte Pakete
1000	<1%	3000	21
3000	<1%	9000	65
5000	<1%	15000	103
8000	<1%	16000	141
20000	<1%	40000	283

Nun folgt die Gegenüberstellung von TCP und UDP im Hinblick auf die Bearbeitungszeit der Clients. Bei UDP wurde das zweite Verfahren verwendet da beim ersten die Verlustrate untragbar war.

TCP:

Clients	Messung 1	Messung 2	Messung 3
5000	10318ms	9438ms	8512m
8000	16502ms	15288ms	14254ms

UDP:

Clients	Messung 1	Messung 2	Messung 3
5000	7918ms	8023ms	7953ms
8000	13099ms	12670ms	12651ms

Fazit:

Nach reichlicher Überlegung sind wir zum Schluss gekommen das wir für den weiteren Projektverlauf auf UDP mit dem Verfahren 2 setzten werden.
Die Vorteile liegen klar auf der Hand. UDP ist wie erwartet schneller als TCP und kann ein vielfaches der Clients verkraften.
Zwar ist bei UDP mit Verlusten zu rechnen, diese können wir in unserem Anwendungsfall aber vernachlässigen da die Meldung eines einzelnen Fahrzeuges zum erkennen eines Staus nicht ausschlaggebend ist.