Verteilte Systeme
Übung 1

Tillmann Faust

2.6.2024

Aufgabe 2

Zur Umsetzung der Aufgabe wurden neben dem Server-Rechner drei weitere verwendet, deren Standorte im selben Gebäude (und Netzwerk), in derselben Stadt, und die Stadt Neuwied (ca. 120km entfernt) waren. In beiden Experimenten hatte der Standort der Rechner daher keinen erheblichen Einfluss auf die Latenz, bzw. war der Einfluss der Latenz geringer als der der zusätzlich künstlich erzeugten Latenz durch die Variable SPIELER_LATENZ. Der größte Aufwand ergab sich durch das vorbereiten der Rechner auf die Umsetzung der Aufgabe, da auf zwei Rechnern Python noch installiert werden musste. Zusätzlich mussten auf allen Rechnern Packages installiert werden, welche das Projekt benötigte. Alle Rechner verwendeten dasselbe Betriebssystem, weshalb sich dort kein Mehraufwand ergab. Anschließend wurde per port-fowarding eine Verbindung zwischen Client-Rechnern, welche nicht bereits im selben Netzwerk waren und dem Server-Rechner hergestellt und die Client Anwendung gestartet. Da sich die Rechner im Besitz von anderen Personen befand und diese unterschiedlich große Hintergründe in der Informatik hatten lag der größte Aufwand in der Navigation durch die einzelnen Schritte zum erfolgreichen Starten der Anwendungen.

a)

Die erste Iteration der Anwendung ist, wie in der Aufgabenstellung beschrieben, eine simple TCP Server-Client-Anwendung in Python, in welcher die Start- und Stopp-Befehle vom Server an die Clients geschickt werden, welcher anschließend die vordefinierte Zeit wartet, "würfelt" und das Ergebnis, zusammen mit seinem Namen an den Server zurückschickt. Das Verhalten beim Testen der Anwendung außerhalb des localhost war - entsprechend den Erwartungen - dass spätestens nach der dritten Aufforderung zum Würfeln ein Client noch an einem vorherigen Wurf 'hing' und die Aufforderung zum erneuten Wurf nicht verarbeiten konnte. Das Resultat war, dass Würfe 'verloren gingen' und andere dafür falsch zugeordnet wurden.

b)

Die zweite Iteration war eine Abwandlung der ersten, mit zwei signifikanten Änderungen. Zum einen wurde eine Puffer-Liste 'event_queue' umgesetzt, in welcher alle Nachrichten vom Server gespeichert werden um dann vom Client nacheinander abgearbeitet zu werden. Der Grund für diese Struktur ist, dass in der Ersten Iteration trotz TCP Pakete 'verloren gegangen' sind, da der Server beim Senden der Stop-Nachricht wartet, bis der Client diese tatsächlich bekommt. Dadurch kommt beim Client statt 'START' oder 'STOP' 'STOP-START' an, da beide Nachrichten als Bytestrom direkt hintereinander gesendet und als eine vom Client interpretiert werden. Der zugehörige "START" befehl geht so verloren. Durch die event_queue wird dieses Problem umgangen.

Als zweites Wurde eine angepasste Lamport-Zeit umgesetzt. Der Server verwaltet einen counter LC, welcher bei jedem Sende-Event erhöht wird. Clients verwalten ebenfalls einen LC, allerdings wird dieser bei den Sende-Events nicht berücksichtigt. Stattdessen wird jede Nachricht vom Server mit der vom Server geschickten Zeit gespeichert und der Client erhöht beim Senden des Zugehörigen Würfelergebnisses diese Zahl um 1.

Während der Testreihe der Anwendung war das Ergebnis zuerst dasselbe wie bei Aufgabenteil (a) außer, dass der Server nun für jede Anfrage eine Würfelzahl erhält, welche allerdings weiterhin an falscher Stelle in der Liste gespeichert wird. Nach Durchlauf der Server-Anwendung wird die Liste der Ereignisse, welche alle Start- und Wurf-Events enthält, basierend auf der Zeit der Nachricht sortiert. Die so erhaltene Liste Enthält in logischer Reihenfolge alle Start-Events gefolgt von den Würfeln aller Clients zur entsprechenden Nachricht.

c)

Zum Umsetzen des dritten Aufgabenteils wurden über ein Shell-Script 20 Clients instantiiert und der Server anschließend gestartet. Es wurde ebenfalls versucht 50 Clients zu starten, allerdings kam es dann zur Fehlermeldung, dass der Versuch vom Server eine Nachricht an einen Client zu schicken, wegen einem nicht eindeutigen Port scheiterte. Das Ergebnis des Versuchs mit 20 Clients war, dass die Nachrichten von den Clients an den Server durcheinander und, wie im kleineren Versuch zuvor in Teil (b), oft ein oder zwei Würfelaufforderungen später ankamen. Die sortierte Liste der Ereignisse nach einem vollständigen Durchlauf hatte dennoch alle 20 Werte der Clients nach jeder Würfelaufforderung korrekt sortiert gespeichert. Im Falle von 20-50 verschiedenen Rechnern ist der Aufwand, selbst ohne die Konflikte, welche sich anfangs mit drei Rechnern ergaben, z.B. weil der Host auch Zugriff auf alle anderen Rechner, entweder physisch, oder remote, hat, enorm. Der Host muss sich bei jedem Rechner anmelden, die Datei auf den Rechner übertragen, die Verbindung zum Server-Rechner herstellen und die Anwendung starten. Angenommen der Host benötigt pro Rechner 2 Minuten, dann ergibt sich ein Gesamtaufwand von ca

40-100 Minuten oder ca. einer Stunde.