*Verteilte Systeme im Sommersemester 2023*

Luca Schöneberg, Matr. Nr. 979049

Henry Wulff, Matr. Nr. 1014530 Osnabrück, 19.3.2023

# Aufgabenblatt 1

## Aufgabe 1 - Beispiele für Verteilte Anwendungen

In der Vorlesung wurden Beispiele verteilter Anwendungen vorgestellt. Entscheiden Sie anhand der o.g. Indikatoren / Gegenindikatoren, ob eine Versionsverwaltung für Dateien als verteilte (siehe z.B. GIT) oder nicht verteilte Anwendung realisiert werden sollte.

|  |  |
| --- | --- |
| **Indikator** | **Bewertung** |
| (A1) gemeinsame Nutzung von Ressourcen | Wenn die Versionsverwaltung rein lokal verläuft, werden Ressourcen nicht verteilt verwendet oder geteilt. Zumindest nicht automatisch. |
| (A2) anbieten von offenen Schnittstellen | GIT könnte mehreren Nutzern Zugriff auf kooperative Versionen bieten. |
| (A3) parallele / nebenläufige Ausführung von Aktivitäten | Bei einem Verteilten System für Versionsverwaltung wird parallele oder nebenläufige Arbeit erst möglich. |
| (A4) dynamische Skalierung des Rechenbedarfs | Der Rechenbedarf ist bei Versionsverwaltungen zu vernachlässigen. |
| (A5) erhöhte Fehlertoleranz | Mit GIT würde sowohl eine lokale Version auf dem Rechner eines jeden Nutzers liegen, als auch ggf. mehrere Lösungen auf dem gegebenen Server.  Würde bei einer rein lokalen Lösung der lokale Rechner ausfallen, würde sämtlicher Fortschritt verloren gehen. |
| (N1) höhere Entwicklungskosten | Eine verteilte Lösung würde die Entwicklungskosten vermutlich senken, da Zusammenarbeit dadurch flexibler wird. Andere Kosten können hingegen steigen. |
| (N2) gefordertes Sicherheitsniveau ist schwerer zu erfüllen | Mehr verteilte Systeme benötigen mehr Systeme, dessen Sicherheitsstandards eingehalten werden müssen. |
| (N3) höhere Wartungsaufwände | Abhängig von der Implementation bringt die erhöhte Flexibilität auch mehr zu wartende Aspekte mit sich. |
| (N4) Verhalten nicht vorhersehbar | Eine kompliziertere Lösung bringt auch weniger Vorhersehbarkeit mit sich. |

Geben Sie anhand Analyse der o.g. Indikatoren jeweils eine Anwendung an, die Sie als verteilte Anwendung bzw. auf gar keinen Fall als solche realisieren würden.

* (A1) gemeinsame Nutzung von Ressourcen
  + Riesenprojekte, wie die Google Suche o.Ä.
* (A2) anbieten von offenen Schnittstellen
* (A3) parallele / nebenläufige Ausführung von Aktivitäten
  + Android Applikationen, die Prozessorkerne für unterschiedliche Dinge verwenden. UI-Thread, DB-Thread etc.
* (A4) dynamische Skalierung des Rechenbedarfs
  + Virtualisierungskluster (Hypervisor)
* (A5) erhöhte Fehlertoleranz
  + Services, wie online Videospiele. Wenn ein Server ausfällt, wird ein anderer verwendet.
* (N1) höhere Entwicklungskosten
  + Bei jedem winzigen Projekt, dessen Entwicklungsaufwand und benötigte Rechenleistung verschwindend gering sind, lohnt es sich nicht diese auf ein Verteiltes System zu implementieren.
* (N2) gefordertes Sicherheitsniveau ist schwerer zu erfüllen
  + Streng vertrauliche Daten sollten nicht verteilt werden, um dessen Unzugänglichkeit zu erhöhen.
* (N3) höhere Wartungsaufwände
  + Bei jedem winzigen Projekt, macht es keinen Sinn die Anwendung verteilt aufzubauen, da dies unnötig Arbeit u.A. im Wartungsaufwand mit sich bringt.
* (N4) Verhalten nicht vorhersehbar
  + Künstliche Intelligenz sollte ggf. nicht verteilt werden, um dessen Schnittstellen zur Außenwelt gering zu halten.

## Aufgabe 2 - Plattform-Unterstützung für Verteilte Systeme

1. Geben Sie an, mit welchen Befehlen inklusive der Parametrisierung Sie auf einem UNIX / Linux-System ermitteln können:
   1. ob ein Host erreichbar ist und mit welcher Verzögerung die Antwort auf eine Anfrage an den Host zu rechnen ist;

* Um zu überprüfen, ob ein Host erreichbar ist und welche Verzögerung bei der Antwort auf eine Anfrage an den Host zu erwarten ist, können Sie den Befehl ping verwenden. Beispiel: ping [www.google.com](http://www.google.com)
  1. welche TCP- und UDP-Sockets geöffnet sind und welche aktiven Server mit diesen verbunden sind:
* Um zu überprüfen, welche TCP- und UDP-Sockets geöffnet sind und welche aktiven Server damit verbunden sind, können Sie den Befehl netstat verwenden. Beispiel: netstat -a
  1. über welchen Netzwerk-Adapter der jeweilige Nachbarrechner im Pool erreicht werden kann und welche MAC-Adresse dieser Adapter besitzt;
* Um zu überprüfen, über welchen Netzwerk-Adapter der jeweilige Nachbarrechner im Pool erreicht werden kann und welche MAC-Adresse dieser Adapter besitzt, können Sie den Befehl arp verwenden. Beispiel: arp -a
  1. unter welcher IP-Adresse ihr Pool-Rechner gegeben durch den DNS-Namen (si002x-0xx-lin.res.hsos.de) hat und umgekehrt welcher DNS-Name mit der IP-Adresse verbunden ist.
* Um die IP-Adresse eines Hosts basierend auf seinem DNS-Namen zu ermitteln, können Sie den Befehl nslookup verwenden. Beispiel: nslookup si002x-0xx-lin.res.hsos.de. Umgekehrt, um den DNS-Namen zu einer IP-Adresse zu finden, können Sie den Befehl host verwenden. Beispiel: host 8.8.8.8.

Testen Sie die von Ihnen ermittelten Kommandos (Beschreibungen befinden sich im Skript zur Vorlesung; weitere Informationen können über die Manuale auf den Rechnern abgerufen werden) und protokollieren Sie die Ergebnisse. Zusatzaufgabe: Beantworten Sie dieselben Fragen für ein Windows (10, 11) Betriebssystem.

1. Ermitteln Sie, welche IP-Ports auf den Linux-Laborrechnern den folgenden Diensten zugeordnet sind:
   1. Ntp

* ntp (Network Time Protocol) verwendet in der Regel den Port 123/UDP. Dieser Port ist auch von der IANA für NTP festgelegt.
  1. https
* https (Hypertext Transfer Protocol Secure) verwendet in der Regel den Port 443/TCP. Dieser Port ist auch von der IANA für HTTPS festgelegt.
  1. Echo
* echo verwendet in der Regel den Port 7/TCP und 7/UDP. Dieser Port ist auch von der IANA für echo festgelegt.
  1. Ssh
* ssh (Secure Shell) verwendet in der Regel den Port 22/TCP. Dieser Port ist auch von der IANA für SSH festgelegt.
  1. ftps
* ftps (FTP Secure) kann entweder den Port 990/TCP (für die Steuerung) oder den Port 989/TCP (für die Datenübertragung) verwenden. Diese Ports sind jedoch nicht von der IANA für FTPS festgelegt.
  1. Sftp
* sftp (Secure File Transfer Protocol) verwendet in der Regel den Port 22/TCP. Dieser Port ist auch von der IANA für SFTP festgelegt.
  1. Kerberos
* Kerberos verwendet in der Regel den Port 88/TCP und 88/UDP. Diese Ports sind auch von der IANA für Kerberos festgelegt.
  1. Rsync
* rsync (Remote Sync) verwendet in der Regel den Port 873/TCP. Dieser Port ist jedoch nicht von der IANA für rsync festgelegt.

Berücksichtigen Sie nur die für das TCP- oder UDP-Protokoll relevanten Ports. Für die o.g. Anwendungen sind sogenannte Well Known Ports durch die Internet Assigned Numbers Authority (IANA, http://www.iana.org) festgelegt. Überprüfen Sie, ob die von Ihnen ermittelten Ports mit den durch IANA festgelegten übereinstimmen. Geben Sie auch in kurzen Worten an, welche Funktion die hinterlegten Dienste haben. Freiwillige Zusatzaufgabe: Beantworten Sie dieselben Fragen für ein Windows-Betriebssystem. Wo werden unter Windows die entsprechenden Informationen gespeichert?

## Aufgabe 3 - Entwicklungsumgebung für Verteilte Systeme

Betrachten Sie das im Dateibereich bereitgestellte Programm daytime.c. Dieses stellt eine minimale C-Implementierung des TCP/IP-Dienstes zur Ausgabe von Tageszeit / Datum dar.

1. Sehen Sie sich den Quellcode des Programms an und kompilieren Sie das Programm in der Praktikumsumgebung.

* Das Programm **daytime.c** implementiert den Daytime-Service, der das aktuelle Datum und die Uhrzeit als Antwort auf eine Anfrage zurückgibt.
* Die **main**-Funktion ist der Einstiegspunkt in das Programm und akzeptiert zwei optionale Argumente: eine IP-Adresse und eine Portnummer. Die IP-Adresse wird verwendet, um den Daytime-Service auf einer bestimmten Netzwerkschnittstelle zu starten. Wenn keine IP-Adresse angegeben ist, wird die Adresse 127.0.0.1 verwendet. Die Portnummer wird verwendet, um den Service auf einem bestimmten Port zu starten. Wenn keine Portnummer angegeben ist, wird die Portnummer 9013 verwendet.
* Das Programm erstellt einen Socket und bindet ihn an die angegebene Adresse und Portnummer. Anschließend wartet das Programm auf Verbindungsanfragen von Clients und gibt das aktuelle Datum und die aktuelle Uhrzeit als Antwort zurück.
* Das Programm erzeugt einen Zeitstempel mit **strftime** und gibt diesen an den Client zurück. Die **strftime**-Funktion nimmt das aktuelle Datum und die aktuelle Uhrzeit und formatiert diese gemäß dem angegebenen Format in einen String (**tstamp**). Der formatierte String wird dann an den Client zurückgegeben.
* Das Programm verwendet eine Endlosschleife, um ständig auf Verbindungsanfragen zu warten und die Anfragen zu bearbeiten. Wenn eine Verbindung hergestellt wird, wird der Zeitstempel an den Client zurückgegeben und die Verbindung wird geschlossen.
* Um das Programm in der Praktikumsumgebung zu kompilieren, kann der Befehl **gcc -o daytime daytime.c** in der Befehlszeile ausgeführt werden. Dabei wird eine ausführbare Datei **daytime** erstellt. Vor dem Kompilieren wird empfohlen, den Quellcode zu lesen, um das Verständnis des Programms zu verbessern. Der Code enthält Kommentare, die die verschiedenen Schritte im Programm erläutern und helfen, das Verständnis zu vertiefen.

1. Machen Sie sich mit der telnet-Anwendung vertraut und testen Sie damit Ihren Server. Das Argument für den Port muss an die entsprechende Einstellung des daytime Programms angepasst werden.

* Um den Daytime-Server mit der Telnet-Anwendung zu testen, haben wir uns mit der Anwendung vertraut gemacht. Dazu haben wir den Befehl telnet <server-ip> <port> in der Befehlszeile eingegeben und eine Verbindung zum Server auf localhost mit Port 9013 hergestellt. Der Server hat das aktuelle Datum und die aktuelle Uhrzeit als Antwort zurückgegeben.
* Da der Daytime-Server auch auf einem anderen Port als dem Standardport 9013 ausgeführt werden kann, haben wir den Server mit dem Befehl gcc -o daytime daytime.c localhost 13 gestartet, um ihn auf localhost mit Port 13 laufen zu lassen. Anschließend haben wir mit dem Befehl telnet localhost 13 eine Verbindung zum Server hergestellt und das aktuelle Datum und die Uhrzeit abgerufen.

1. Testen Sie die in dem Netzwerk zur Verfügung stehenden Server der anderen Teilnehmer.

Leider zuhause nicht möglich.

1. Im Internet stehen verschiedene Daytime-Server zur Verfügung. Machen Sie mindestens zwei Server ausfindig und rufen Sie diese per telnet auf. Dokumentieren Sie die von Ihnen verwendeten Aufrufe inkl. Parametrisierung.

* telnet time.nist.gov 13 => 60017 23-03-14 18:26:39 50 0 0 328.9 UTC(NIST) \*
* telnet time-b.nist.gov 13 => 60017 23-03-14 18:26:40 50 0 0 920.6 UTC(NIST) \*
* telnet utcnist2.colorado.edu 13 => 60017 23-03-14 18:26:43 50 0 0 510.5 UTC(NIST) \*

1. Sind Latenzen und Verarbeitungszeiten bei den Aufgaben 3. und 4. für eine Korrektur der Zeitstempel zu berücksichtigen (z.B. bei Anwendung von Cristian’s Algorithmus)?

* Latenzen und Verarbeitungszeiten sind bei der Anwendung des Cristian-Algorithmus für eine genaue Korrektur der Zeitstempel zu berücksichtigen. Der Cristian-Algorithmus basiert auf der Messung der Round-Trip-Zeit zwischen dem Client und dem Server. Dabei wird angenommen, dass die Round-Trip-Zeit symmetrisch ist, was jedoch nicht immer der Fall ist. Wenn die Round-Trip-Zeit asymmetrisch ist, kann die Zeitkorrektur ungenau sein und zu einer falschen Zeitangabe führen.

Zur Nutzung von telnet:

* Linux: Sie verlassen den Eingabemodus von telnet durch Drücken von STRG und ALT GR und 9, danach . Am Prompt telnet> geben Sie dann quit ein. Kopieren Sie (auch bei den weiteren Programmteilen) alle Ein- und Ausgaben ins Protokoll.
* Windows: Wenn Sie telnet unter Windows nutzen wollen, so müssen dies als bei den Einstellungen unter ‚Optionale Features‘ explizit freischalten.