UNIVERSITÄT SIEGEN

Fakultät IV
Betriebssysteme und verteilte Systeme
Prof. Dr. rer. nat. Roland Wismüller

Aufgabenblatt 1
Musterlösung

Vorlesung Verteilte Systeme
Wintersemester 2024/25

Aufgabe 1: Heterogenität und Offenheit

a) Da beide Rechner an das Internet angeschlossen sind, können wir annehmen, dass die Internetprotokolle die Unterschiede der Netze behandeln.

- Weil die Rechner **unterschiedliche Hardware** haben, müssen die Unterschiede der Darstellung der Datenelemente in den Anfrage- und Antwort-Nachrichten von Clients zu Server betrachtet werden. Ein allgemeiner Standard wird für jede Art eines Datenelements definiert, das zwischen den Server und den Clients übertragen werden muss.
- Da die Rechner **unterschiedliche Betriebssysteme** ausführen, müssen wir mit unterschiedlichen Operationen zurechtkommen, um Nachrichten zu senden und zu empfangen oder Aufrufe auszudrücken. Deshalb wird auf der Java/C++ Ebene eine einheitliche Methode verwendet, die auf die konkrete Operation des jeweiligen Betriebssystems übersetzt wird.
- Wir haben zwei **unterschiedliche Programmiersprachen** C++ und Java. Diese benutzen unterschiedliche Darstellungen für Datenstrukturen wie Zeichenketten, Arrays und Objekte. Für jede Art Datenstruktur, die zwischen dem Server und den Clients übertragen werden muss, wird deshalb ein einheitlicher Standard für die Datenstruktur selbst und eine entsprechende Übersetzung in die Datenstruktur der jeweiligen Sprache definiert.
- Die Implementierungen können von **unterschiedlichen Entwicklern (Herstellern)** sein, z.B. einer für C++ und ein anderer für Java. Sie müssen über die allgemeinen erwähnten Standards einig sein und die Standards dokumentieren.

b) Auch um neue Dienste in einem vorhandenen offenen System hinzuzufügen, müssen die Standards, die in dem Punkt **a)** erwähnt wurden, vereinbart worden sein. Kurz wiederholt:

- das verteilte System verwendet einen allgemeinen Satz Kommunikationsprotokolle (vermutlich Internet Protokolle)
- es verwendet einen definierten Standard für das Darstellen der Datenelemente.
- es verwendet einen allgemeinen Standard für die Message Passing Operationen.
- Verwendung einen sprachneutralen Standard für das Darstellen der Datenstrukturen.

Aber für ein offenes verteiltes System müssen die Standards vereinbart und dokumentiert worden sein, **bevor** die Implementierung bereitgestellt wird. D.h. die Implementierungen müssen jenen Standards entsprechen. Zusätzlich muss die Schnittstelle zum Dienst veröffentlicht werden, damit, wenn er dem System hinzugefügt wird, sowohl vorhandene als auch neue Clients ihn verwenden können. Die Veröffentlichung (Publikation) der Standards erlaubt, daß Teile des Systems von unterschiedlichen Softwarehersteller implementiert werden und später ohne Probleme zusammenarbeiten.

c) Jede Anfrage, die eine geschützte Methode aufruft, muss die Identität des Benutzers beinhalten, der die Abfrage gestellt hat. Die Probleme dabei sind:

- die Identitäten der Benutzer zu definieren. Anwendung dieser Identitäten (IDs) in der Liste der Benutzer, die berechtigt sind, die geschützten Methoden auzufrufen, und in den Anfrage-Nachrichten.
- Sicherstellen, dass die gelieferte Identität vom richtigen Benutzer kommt und nicht von irgendeinem anderen, der vortäuscht, dieser Benutzer zu sein.

- verhindern, dass andere Benutzer die Anfrage-Nachricht eines berechtigten Benutzers wiederholen oder verfälschen.

Zusätzlich:
Die Information, die als Resultat einer geschützten Methode zurückgeliefert wird, muss verschlüsselt werden, falls sie von nichtautorisierten Benutzern abgefangen werden kann.

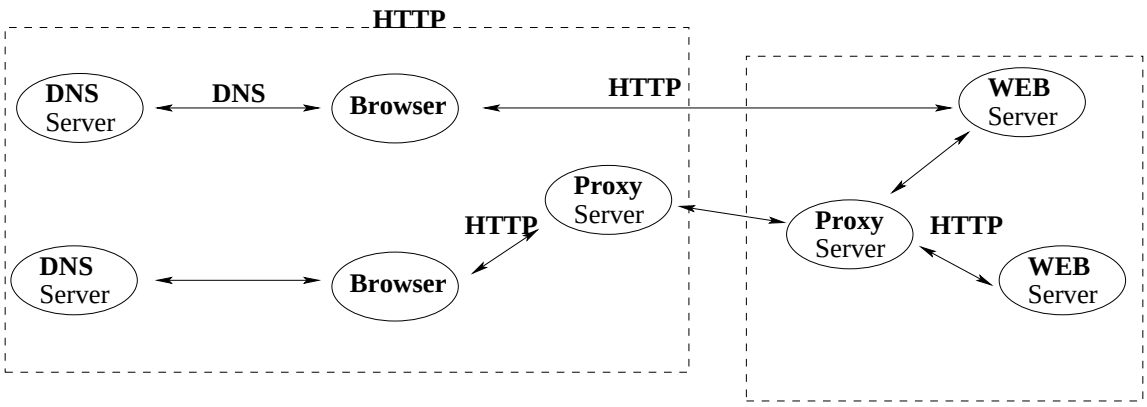
Aufgabe 2: Migration von Ressourcen

Eine Migration von Ressourcen wird u.a. durchgeführt, um

- Kommunikationverzögerungen zu verringern (Platzierung von Objekten auf einem Server in dem lokalen Netz, von dem aus sie am häufigsten benutzt werden),
- die Rechenlast und/oder Speicherauslastung zwischen unterschiedlichen Server auszugleichen.

Aufgabe 3: Architektur bekannter Client/Server-Anwendungen

WEB:

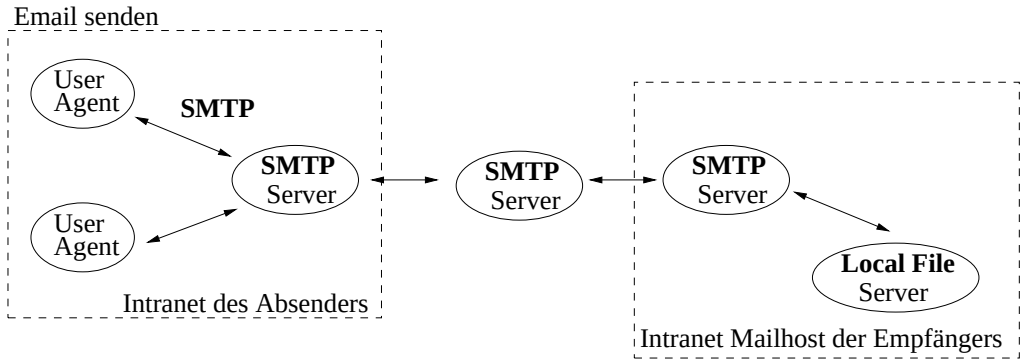


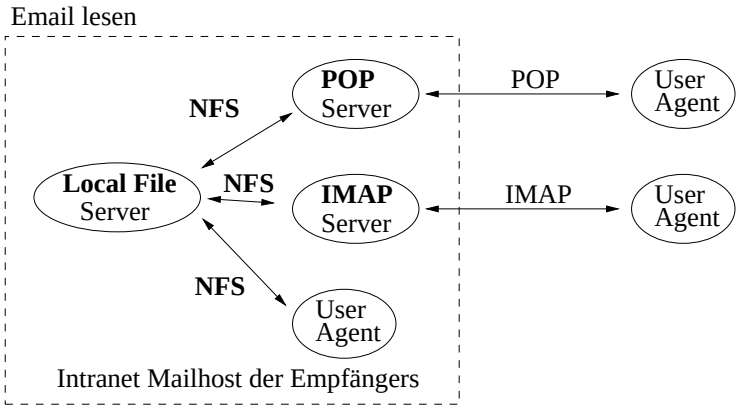
Browser sind Clients der Domain-Name-Server (DNS) und der Web-Server. Manche Intranets sind so konfiguriert, dass ein Proxy-Server dazwischengeschaltet ist. Proxy-Server verringern Netzverzögerungen und Netzverkehr, wenn sie sich am gleichen Ort wie der Client befinden. Wenn sie am gleichen Ort wie der Server sind, bilden sie eine Sicherheits-Kontrollstelle und können die Last auf dem Server verringern. Die Verantwortlichkeit für die Datenkonsistens liegt bei den Proxy-Servern – sie prüfen periodisch die Modifikationszeit der Seiten beim Ursprungs-Web-Server.

Webseiten werden im lokale Filesystem des Servers gespeichert. Die gesamte Information im Web ist daher auf viele Web-Server verteilt. Replikation ist kein Teil des Web-Protokolls, aber eine häufig besuchte Web-Site kann mehrere Server mit identischen Kopien des Dateisystems versorgen. HTTP-Anfragen können zwischen diesen Servern aufgeteilt werden (z.B. über DNS Lastverteilung). Zusätzlich unterstützen Proxy-Server die Replikation durch Caching zuletzt benutzter Seiten, ebenso wie auch Web-Browser.

DNS-Server können auch in alle Anwendungs-Architekturen, die unten beschrieben sind, miteinbezogen werden, sind aber zur Vereinfachung der Diskussion ausgelassen.

Email:





Nachrichten senden: Der User Agent (das Anwender-Programm zum Versenden von Emails) ist ein Client eines lokalen SMTP-Servers und übergibt jede abgehende Nachricht dem SMTP-Server für die Auslieferung. Der lokale SMTP-Server verwendet Email-Routingtabellen, um einen Weg für jede Nachricht festzustellen und leitet dann auf dem gewählten Weg die Nachricht an den folgenden SMTP-Server weiter. Jeder SMTP-Server verarbeitet und schickt jede ankommende Nachricht weiter, bis der Domain-Name in der Nachrichtadresse der lokalen Domain entspricht. Im diesem Fall versucht er, die Nachricht an lokale Empfänger auszuliefern, indem er sie in eine Mailbox-Datei auf einer lokalen Festplatte oder einem File-Server speichert.

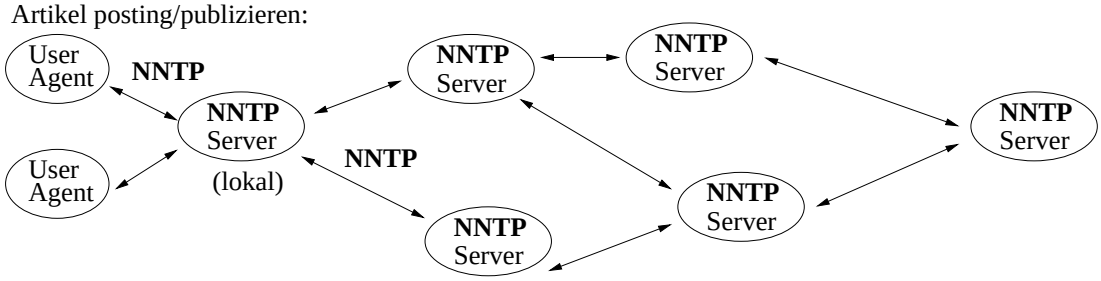
SMTP-Server müssen nicht notwendigerweise Email-Routingtabellen für alle Ziele haben. Stattdessen geben sie Nachrichten für unbekannte Ziele an einen anderen Server weiter, der wahrscheinlich eine passende Routingtabelle besitzt.

Nachrichten werden nur an ihrem Ziel (permanent) gespeichert. D.h., der Email-Dienst basiert nur auf Partitionierung, auch wenn eine Nachricht mit mehreren Empfängern auf mehreren Zierechnern repliziert wird.

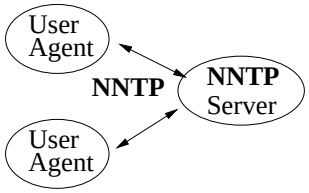
Nachrichten empfangen: Der User Agent (der Mail-Reader des Benutzers) ist entweder ein Client des lokalen File-Servers oder ein Client eines Email-Zustellungs-Servers wie ein POP- oder IMAP-Server. Im ersten Fall liest der User Agent die Nachrichten direkt von der Mailbox-Datei. Im letzten Fall bittet der User Agent um Informationen über den Inhalt der Mailbox-Datei des Benutzers von einem POP- oder IMAP-Server und empfängt die Nachrichten von diesem Server für die Darstellung zum Benutzer. POP und IMAP sind Protokolle, die spezifisch entworfen sind, um den Emailzugang über WAN und langsame Netzverbindungen zu unterstützen.

News:

Artikel posting/publizieren:



Artikel browsing/lesen:



Posten von News-Beiträgen: Der User Agent (das Anwender-Programm zum Erstellen und Versenden von News) ist ein Client eines lokalen NNTP-Servers und übergibt jeden abgehenden Artikel zum NNTP-Server für die Auslieferung. Jedem Artikel wird ein eindeutiger Bezeichner zugewiesen. Jeder NNTP Server hält eine Liste anderer NNTP-Server, für die er ein News-Erzeuger ist – sie sind registriert, um Artikel von ihm zu empfangen. Periodisch kontaktiert er alle registrierten Server, liefert alle neuen Artikel an sie und verlangt alle, die sie schon haben, aber er noch nicht (der eindeutige Bezeichner hilft festzustellen, welche das sind). Damit werden die Artikel einer Newsgroup auf allen Servern repliziert, die sie benötigen.

Lesen von News-Beiträgen: Der User Agent (das Newsleseprogramm des Benutzers) ist ein Client eines lokalen NNTP-Servers. Der User Agent bittet um Updates für alle Newsgroups, für die der Benutzer sich registriert hat, und stellt die News dar.

Aufgabe 4: Sitzungsverfolgung

Sitzungsverfolgung mit Cookies:

Ein Cookie besteht aus einem Namen, einem Wert und einem oder mehreren Attributen. Ein Attribut kann z.B. EXPIRES=dateValue, Version oder SECURE sein

Die SitzungsID wird innerhalb eines Cookies über den Browser oder ein Skript an den Client übergeben, damit der Server ihn erkennen kann und an ihn angepasste Informationen aufbereiten kann. Bei jedem erneuten Besuch werden die Textinformationen aus dem Cookie an den Server gesendet. Die Cookies können in den Headern von Anfragen und Antworten oder durch ein Skript wie z.B. JavaScript oder innerhalb von Java-Applets übertragen werden und bleiben immer lokal auf dem Client gespeichert.

Mit Hilfe von Cookies ist eine eindeutige Erkennung eines Clients möglich, damit können also Benutzerprofile innerhalb einer Webseite erstellt werden, die missbrauchlich auch mit personenbezogenen Daten verknüpft werden könnten. Um das weitere Surfverhalten eines Nutzers teilweise nachzuvollziehen, werden Third-Party-Cookies benötigt, die sich meist in Werbebannern oder anderen Medieninhalten einer Webseite befinden. Wenn der Nutzer eine der zugehörigen Webseiten besucht hat, können die Informationen verknüpft werden.

Der Nutzer hat aber die Möglichkeit, Cookies zu erlauben oder zu sperren, wobei dort noch feingranularer unterschieden werden kann. Bei öffenltichen Rechnern sollte dringend darauf geachtet werden, dass Cookies nicht dauerhaft gespeichert werden.

Sitzungsverfolgung mit URL-Rewriting:

Von URL-Rewriting spricht man u.a., wenn die Sitzungs-ID an das Ende einer URL angehängt wird, um damit die Sitzung eines Clients nachzuvollziehen, was browserunabhängig und eher simple umsetzbar ist. Problematisch ist es hier ein wenig auf Seiten des Servers, da hier die Kennungen zu den Sitzungen zugeordnet werden müssen.

Für den Nutzer ist die Erweiterung der URL sichtbar und birgt den Nachteil, dass diese URL nach einer Weile nicht mehr erreichbar ist, da diese sich auf eine bestimmte Session bezieht.

Des weiteren stellt sich ggf. die Länge der URL als problematisch dar, da die Browser, HTTP-Protokolle, Suchmaschinenindizierung usw. nur mit einer maximalen Länge von 255 Zeichen pro URL umgehen können.

Aufgabe 5: Antwortzeit von Client-Anwendungen

Die Antwortzeit von Client-Anwendungen wird von folgenden Faktoren beeinflusst:

a) die Auslastung des Servers,

b) die Latenz beim Austausch der Anfrage- und Antwortnachrichten (wegen der BS-Schichten und Middleware-Software im Client und Server),

c) der Last im Netz.

Caching hilft bei allen oben genannten Problemen (insbesondere Client-Caches). Proxy-Server Caches helfen bei a). Replikation des Dienstes hilft auch bei a). Die Verwendung von leichgewichtigen Kommunikationsprotokollen hilft bei b).

Aufgabe 6: Server-Implementierung

a) Im single-threaded Fall dauert ein Cache-Treffer 15 und ein Cache-Fehler 90 Millisekunden. Der gewichtete Mittelwert ist 2/3*15 + 1/3*90. Die durchschnittliche Anforderung dauert also 40 Millisekunden und der Server kann 25 pro Sekunde verarbeiten. Für einen Multithreaded-Server überlappt sich das Warten auf die Festplatte, somit dauert jede Anforderung 15 Millisekunden und der Server kann 66 2/3 Anforderungen pro Sekunde verarbeiten.

b) Ja, aus zwei Gründen: Erstens, Threads brauchen Speicher für die Einrichtung ihres eigenen, privaten Stacks. Demzufolge können sehr viele Threads so viel Speicher belegen, dass der Server nicht mehr korrekt arbeitet. Ein weiterer, ernsthafter Grund ist, dass unabhängige Threads für ein Betriebssystem auf chaotische Weise ausgeführt werden. In einem virtuellen Speichersystem kann es schwierig sein, eine relativ stabile Arbeitsmenge zu erstellen, was zu vielen Seitenfehlern und damit vielen Ein- und Ausgaben führt. Die Verwendung vieler Threads kann also zu einer Leistungsverschlechterung führen, welche aus dem Seiten-Thrashing entsteht.