Lösungshinweise zur Klausur
Verteilte Systeme
am 05. März 2005

Aufgabe 1: **Kommunikation** **10 Punkte**

Ein Radiosender strahlt sein Programm für viele Empfänger aus. Er sollte sich darauf konzentrieren, die Sendungen an alle Empfänger abzuschicken, kann sich aber nicht darum kümmern, ob bei jedem einzelnen Empfänger alles korrekt ankommt. Der Empfänger wird die Daten normalerweise gar nicht abspeichern, sondern sofort über Lautsprecher ausgeben. Deshalb macht es wenig Sinn, ein verloren gegangenes Datenpaket später noch einmal zu senden, weil die Sendung auch beim Empfänger schon weitergelaufen ist.

Damit kommt nur ein verbindungsloses Protokoll wie UDP in Frage. Im Vergleich zum verbindungsorientierten TCP vermeidet der Sender damit den Zusatzaufwand (Overhead) für

- das Abspeichern von Informationen über den Zustand der Empfänger,
- die Übertragung von Paketen zur Kontrolle der Verbindung (Bestätigungen usw.),
- das wiederholte Übertragen von Daten, die nicht korrekt empfangen wurden,
- das Warten auf Empfänger, die nicht bereit sind.

Aufgabe 2: **Zeit** **15 Punkte**

1. Siehe Abbildung 1. Dort haben wir 1 als Inkrement beim Setzen der Empfangszeitpunkte benutzt.
2. - $b \longrightarrow d$, wegen der Nachricht M.
   - b und c sind nebenläufig, denn weder $b \longrightarrow c$ noch $c \longrightarrow b$ kann aufgrund des Nachrichtenflusses gelten.
   - $b \longrightarrow e$, wegen der Nachrichten M und N.
3. Nach Erweiterung der Zeitpunkte mit der Prozessnummer erhalten wir

   Ereignis a um $(15, 1)$
   b um $(15, 0)$
   c um $(30, 2)$
   d um $(24, 3)$
   e um $(54, 2)$

   Die Reihenfolge ist deswegen b, a, d, c, e.

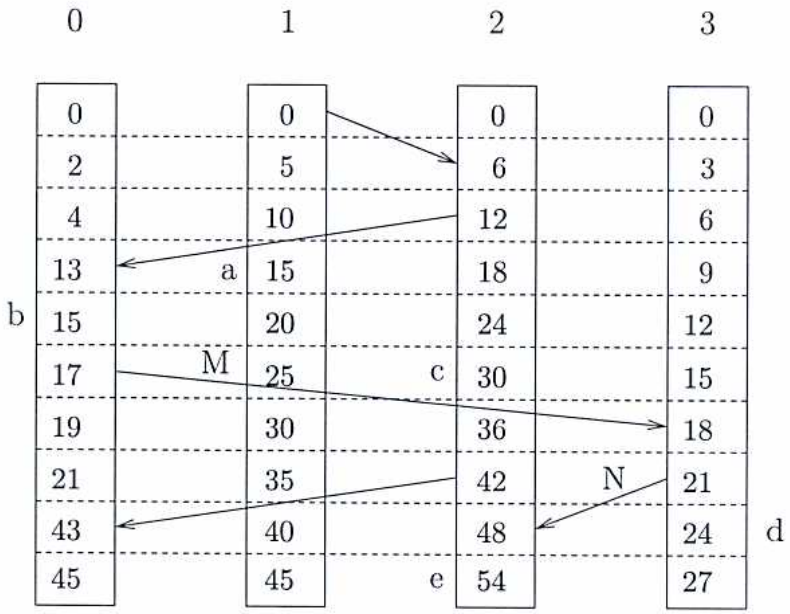

Abbildung 1: Nachrichtenaustausch zwischen den Prozessen 0, 1, 2, 3.

Aufgabe 3: **Client-Server/Threads** **16 Punkte**

1. Ohne Threads benötigt der Server für eine Anfrage durchschnittlich

$$\frac{1}{2} \cdot 500 + \frac{1}{2} \cdot (500 + 9000) = \frac{1}{2} \cdot 10^4 = 5000\,\mu\text{s},$$

   also kann er höchstens

$$\frac{10^6}{5000} = 200$$

   Anfragen innerhalb einer Sekunde ($10^6\,\mu$s) bearbeiten.

2. Wir zeigen, dass höchstens 19 Threads benötigt werden.

   Jeder Thread braucht $500\,\mu$s, um eine Seite für die Anfrage im Cache durchzusuchen. Wenn ein Zugriff auf eine Festplatte benötigt wird, dann ist er für $9000\,\mu\text{s} = 18 \cdot 500\,\mu$s blockiert. Während dieser Zeit kann ein weiterer Thread starten, um den Cache durchzusuchen. Im schlimmsten Fall braucht jeder weitere Thread einen Zugriff auf eine Festplatte. Das heißt, während ein Thread blockiert, werden in worst case insgesamt noch 18 weitere Threads benötigt, damit die CPU die ganze Zeit beschäftigt ist.

   Nach 9 ms wird ab jetzt jede $500\,\mu$s mindestens ein der 19 Threads mit seiner Bearbeitung fertig. Sie können für die weiteren Bearbeitungen der nächsten Anfragen eingesetzt werden. Die weiteren neuen Threads werden nicht mehr gebraucht.

Aufgabe 4: **Sicherheit** **13 Punkte**

Voraussetzung: Sowohl der Benutzer A wie der Host B besitzen ein Schlüsselpaar für ein asymmetrisches Verschlüsselungsverfahren. Jeder kennt den öffentlichen Schlüssel des anderen. Das bedeutet im allgemeinen, dass sich sowohl der Benutzer A für jeden Host, auf dem er einen Account hat, dessen öffentlichen Schlüssel merken muss, als auch der Host B

für jeden Benutzer. **Achtung:** Passwörter werden dabei überflüssig.

Protokoll:

1. Der Benutzer A erzeugt eine Zufallszahl x und schickt diese, zusammen mit seinem Benutzernamen an den Host B.

2. Host B signiert x mit seinem privaten Schlüssel und schickt das so verschlüsselte x zusammen mit einer Zufallszahl y (und ggf. der Angabe seines Namens) an den Benutzer A.

3. Benutzer A entschlüsselt x mit dem öffentlichem Schlüssel von Host B und vergleicht dieses x mit der von ihm erzeugten Zufallszahl. Stimmen beide überein, so verschlüsselt er y mit seinem privaten Schlüssel und schickt ihn an den Host B.

4. Host B sucht den privaten Schlüssel von Benutzer A heraus, entschlüsselt y und vergleicht mit der von ihm erzeugten Zufallszahl. Stimmen beide überein, so erhält der Benutzer A Zugang zum Rechner.

Die Verwendung von Zufallszahlen ist notwendig, damit nicht ein Angreifer C sich eine signierte Standardnachricht (z.B. `login`) aufheben kann, um so Zugang unter falschem Namen zu erlangen. Gleiches gilt für die andere Seite.

Aufgabe 5: **Kryptographie** **12 Punkte**

Der Klartext lautet: `fernuniklausur`.

Aufgabe 6: **Versionenverwaltung** **12 Punkte**

- `cvs checkout project`
  Alle Dateien des Projekts werden in den lokalen Arbeitsbereich geladen.

- `cvs add file`
  Die Datei `file` wird dem CVS-Repository hinzugefügt. Hierbei wird aber zunächst nur ein Eintrag erzeugt, der Dateiinhalt wird mit dem nächsten `cvs commit` übertragen.

- `cvs commit file`
  Die veränderte Datei `file` wird in das CVS-Repository übertragen und steht danach für alle Benutzer des Systems zur Verfügung. Man kann hier noch einen Kommentar zur Änderung hinterlassen.

- `cvs status file`
  Gibt die Antwort auf die Frage: Ist die lokale Kopie der Datei `file` noch up-to-date, wurde sie lokal oder von einem anderen Benutzer verändert (oder beides) oder wurde sie zunächst lokal hinzugefügt?

- `cvs update file`
  Aktualisiere die Datei `file` auf die neueste Version aus dem CVS-Repository. Bei einer auch lokal geänderten Datei wird ein *merge* stattfinden und gegebenenfalls ein *merge confict* resultieren, der vom Benutzer aufgelöst werden muss.

- `cvs log file`
  Listet alle bisherigen Versionen von Datei `file` mit Verursacher, Änderungsdatum und Kommentar auf.

Aufgabe 7: **Verteilte Dateisysteme** **10 Punkte**

Vergleich AFS mit NFS:

	NFS	AFS
Anzahl Workstations	klein (einige 10)	sehr groß (>1000)
Netzwerk-Belastung	groß	klein
Daten-Sicherheit und -Schutz	gering	sehr gut (Kerberos)
Datenzugriff/Caching	immer über das Netz	lokal
Interaktion Client-Server	sehr groß	gering
Administration	kompliziert	einfach
File-Zustand	nicht definiert	Unix-Semantik

Bei AFS wird durch die Unterteilung in Cluster, die Verlagerung vieler Aufgaben zu den Clients und das Zwischenspeichern ganzer Dateien eine sehr hohe Skalierbarkeit erreicht. Die Skalierbarkeit von AFS wird durch die zentralen Verwaltungsdatenbanken (Authentication Data Base: enthält Benutzernamen und Passwörter, Protection Data Base: enthält die Zugriffsberechtigungen, Volume Location Data Base: enthält Information über die Fileserver-Maschinen, auf denen sich die Volumes befinden, Backup Data Base: enthält Information über die von den Volumes angelegten Backup-Kopien) begrenzt.

Aufgabe 8: **CSCW** **12 Punkte**

- Bildung einer Gruppe, Einrichtung eines Arbeitsbereichs für die Gruppe, exklusiver Zugriff nur für Gruppenmitglieder

  Ja. Ein BSCW-Benutzer kann einen neuen Arbeitsbereich einrichten, er ist dann der Manager und kann Einladungs-E-Mails an ausgewählte Adressen verschicken. Nur die Gruppenmitglieder haben Zugriff zum Arbeitsbereich.

- Ablage von und Zugriff auf Dateien im Arbeitsbereich

  Ja. Jedes Gruppenmitglied kann Dateien in den Arbeitsbereich laden oder von dort holen.

- Reservierter, exklusiver Zugriff auf eine bestimmte Datei für ein Gruppenmitglied für eine bestimmte Zeit

  Nein, ein Gruppenmitglied kann eine Datei nicht für eine bestimmte Zeit vor dem Zugriff durch andere Gruppenmitglieder schützen. Man kann höchstens einen Kommentar hinterlassen und andere bitten, hier im Moment nichts zu ändern.

- Automatische Benachrichtigungen über Änderungen an den gemeinsamen Dateien

  Nein, BSCW verschickt keine automatischen Benachrichtigungen über Änderungen z. B. per E-Mail, die Benutzer müssten dies selbst veranlassen.

- Rückgriff auf ältere Versionen einer Datei

  Ja, die früheren Versionen einer Datei stehen zur Verfügung.

- Anzeige der Unterschiede von verschiedenen Versionen einer Datei

  Nein, ein Benutzer kann zwar mehrere Versionen einer Datei auf seinen Rechner holen, muss dann aber selbst sehen, wie er die Unterschiede anzeigen kann.