1. VL01: Welche Aufgaben hat ein Betriebssystem?
   - Abstraktion der Hardware, so dass sie leichter benutzbar gemacht wird (für Anwendungsprogramme)
   - Verwaltung der Hardware-Ressourcen, gerechte Verteilung, Schutz vor Überbeanspruchung (Vermeidung von Konflikten, Fairness, Schutz voreinander (Prozesse))
2. VL02: Was sind Caches und warum sind sie notwendig? Warum funktionieren sie überhaupt?
   - Puffer-Speicher - um Daten, die bereits einmal vorlagen, beim nächsten Zugriff schneller lesen zu können
   - Weil man häufig auf die selben Daten von langsameren Speichermedien (im Vgl. zum Cache) mehrmals hintereinander zugreift (spacial locality)
3. VL07/F35: Was sind die condition variables, und wie verwendet man die POSIX-Funktion *wait(condition, mut)?
   - *Die Zustandsvariablen (conditions variables) erlauben effizientes "Warten auf eine Bedingung"*
   - *Zutreffen der Bedingung wird hier durch den Wert einer binären Variable dargestellt*
   - *Alternative zu komplizierteren Verfahren*
     - *Periodisch den Wert eines Mutex testen (z.B. mit trylock()), der eine Bedingung repräsentiert*
     - *Semaphore nutzen* (siehe Codeblock-Abhägnigkeit mit Semaphoren)

     (a) Thread A blockiert Mutex mut und betritt seine kritische Region (mit mutex_lock (mut))
     (b) Wenn eine Bedingung nicht erfüllt ist, ruft A wait(condition, mut) auf. Dabei wirt Mutex mut automatisch freigegeben, und Thread A geht schlafen
     (c) Thread B erlangt Mutex, erfüllt irgendwann die Bedingung. Dann weckt A via signal(condition) auf
     (d) Thread A wacht automatisch auf, und Mutex mut wird automatisch blockiert
     (e) Thread A beendet die kritische Region, gibt mut frei
4. VL07/F41: Wie funktioniert Swaping, und wozu ist es da?
   - *Roll out, roll in - Prizip: Prozesse mit niedriger Priorität werden ausgelagert (swapped out), Prozesse mit höherer hereingeholt*
   - *Swapping: man lagert das gesamte Speicherabbild eines Prozesses auf die FP aus, um RAM zu schonen*
5. VL08/F12: Welche zwei Herausforderungen des Speichermanagements gibt es?

- *Der Code muss ausführen, egal wo die Startadresse (von Code+Daten) liegt: Problem der Adressenrelokation. Das ist relativ leicht zu lösen.*
- *Adressenbereich muss zusammenhängend sein: Problem der zusammenhängenden Adressen. Die Lösung ist sehr aufwändig, und bedarf Hardware-Unterstützung.*

6. Was ist die externe Fragmentierung, was ist die interne Fragmentierung?
   (a) Löcher im Speicher, die durch ein und auslagern der einzelnen Prozesse entstehen (bei variable-partition scheme).
   (b) Löcher im Speicher, die in den fixen Speicher-Slots entstehen (bei fixed-sized partitions).

7. Was ist die MMU? Was ist/sind logischer Adressraum/logische Adressen versus physischer Adressraum/physische Adressen?
   - *Memory-Management-Unit: Zentrales Element der Lösung ist ein Memory-Management Unit - eine Hardwareeinheit (oft im Prozessor), die die Adressen des Programms auf die Adressen des RAMs übersetzt*
   - *MMU übersetzt den logischen Adressraum auf den unabhängigen physischen Adressraum*
   - *Logischer Adressraum besteht aus logischen oder virtuellen Adressen*
   - *Der Prozess und die CPU üsehen nur diese logische Adressen*
   - *Werte in den CPU-Registern sind alle logisch*
   - *PA besteht aus den physischen Adressen*
   - *Adressen, die von dem Speicher(bus) gesehen wird, d.h. echte oder Hardware-Adressen*
   - *Adressen, die die RAM-Bausteine tatsächlich sehen*

8. Wie funktioniert eine einfache MMU mit Basis- und Limitregister? Welche der beiden Herausforderungen des Speichermanagements kann diese lösen?
   - Prozess will auf logische Adresse X zugreifen:
     (a) Zunächst wird folgende Überprüfung angestellt: Ist X kleiner als Limitregister (ebenfalls logische Adresse). Falls nein, würe das eine Verletzung der Speichergrenze des Prozesses.
     (b) Falls ja, wird nun mittels X + Basisregister die physische Adresse auf die X verweisen soll berechnet.
   - *Problem P1 gelöst!*; Also die Herausforderung der Adressenrelokation ist gelöst.

9. Geben Sie eine Definition des Pagings an. **Welche Einschränkung hat dadurch die Übersetzung der Adressen, damit es umsetzbar ist?**
   - *Methode der Arbeitsspeicherverwaltung durch Übersetzung der Adressen zur Laufzeit*
   - *Dies geschieht mit einer Adress-Übersetzungstabelle (Seitentabelle)*
   - *Blöcke der logischen Adressen (Pages) werden in Blöcke der physischen Adressen (Frames) übersetzt*

- *Dazu wird eine von der CPU erzeugte Adresse (logische Adresse) konzeptionell unterteilt in:*
  - *Seitennummer bzw. Seitenindex (page number) p: Obere Bits der logischen Adresse*
  - *Seitenoffset (page offset) d: Untere Bits der log. Adresse*

10. Wie wird eine logische Adresse aufgeteilt, und was ist die Seitennummer bzw. der Seitenindex, und der Seitenoffset?

    - *Dazu wird eine von der CPU erzeugte Adresse (logische Adresse) konzeptionell unterteilt in:*
      - *Seitennummer bzw. Seitenindex (page number) p: Obere Bits der logischen Adresse*
      - *Seitenoffset (page offset) d: Untere Bits der log. Adresse*
    - *Bei Adressraum der Größe 2^m und 2^n als Größe einer Seite haben wir: Logische Adresse = m Bits; Seitennummmer p = m - n Bits; Seitenoffset d = n*

11. Was ist eine Seitenrahmennummer?

    - Nummer der Kachel die im physischen Speicher liegt.
    - *Die Adressübersetzung funktioniert durch das Ersetzen (in der Adresse) der Seitennummer p durch eine (eigentlich beliebige) Seitenrahmennummer f (d.h. obere Bits einer physischen Adresse). Die n Bits des Seitenoffsets d werden direkt übernommen*

12. Wie funktioniert die Adressenübersetzung auf der konzeptionellen Ebene, d.h. noch ohne eine Seitentabelle? Mit anderen Worten, welche Teile der logischen Adresse werden übersetzt und wodurch (hierbei soll man Begriffe wie Seitennummer, Seitenrahmennummer usw. verwenden)?

    - *Die Adressübersetzung funktioniert durch das Ersetzen (in der Adresse) der Seitennummer p durch eine (eigentlich beliebige) Seitenrahmennummer f (d.h. obere Bits einer physischen Adresse). Die n Bits des Seitenoffsets d werden direkt übernommen*

13. Wie funktioniert das Prinzip der Adressenübersetzung durch eine (direkte) Seitentabelle (ein Diagramm als Antwort)? Was ist der Index, was ist der Inhalt eines Eintrags in der Seitentabelle?

    - Index (key) = p
    - Inhalt (value) = f

14. Wie hilft das Paging, die externe Fragmentierung zu beseitigen?

    - *Um ein Prozess der Größe n Seiten auszuführen:*
      - *Finde n freie Seitenrahmen*
      - *Weise sie dem Prozess via Seitentabelle als Speicher zu*
    - *Der Clou: die Frames müssen nicht zusammenhängend sein, sondern können ggf. in vielen verschiedenen Bereichen des physischen Speichers liegen!*

- Da jeder Seitenrahmen gleich grösss ist, gibt es keine Speicherlöcken mehr

15. Warum muss jeder Prozess eine eigene (direkte) Seitentabelle haben?

    - Weil Prozess A und Prozess B z.B. beide die logische Adresse 00-00 haben. Ihre Seitennummern sind also gleich und sie würden auf die selbe Seitenrahmennummer zugreifen.

16. Was ist ein TLB, und warum ist es sehr wichtig und sehr schnell?

    - *Problem: Jeder Speicherzugriff muss in zwei Zugriffe übersetzt werden: A. Erhöhte Zugriffszeit auf den Speicher B. Hoher Speicherbedarf einer Seitentabelle (1) Lesen aus der Seitentabelle und (2) Eigentliches Lesen/Schreiben auf den Speicher*
    - **Vermutlich ist er sehr wichtig, weil sonst der Speicherzugriff mit einer naiven Lösung viel zu langsam wäre**
    - *Man verwendet einen Cache für die Einträge der Seitentabelle (spezielle Hardware in MMU), Den Translation Look-Aside Buffer (TLB)*
    - *Die Lösung nutzt spacial locality aus: Sehr wenige Seiten werden sehr häufig verwendet*

17. VL09/F12: Wie berechnet man die effektive Zugriffszeit?

    - **Fazit Formel: EAT = E[X] = (1+T)h + (2+T)(1-h) = 2 + T - h**

18. VL09/F14: Was ist die maximale Grösse einer direkten Seitentabelle? Welche Daten muss man zur Berechnung haben?

    - Grösse des maximal adressierbaren Speichers (entnimmt man dem Prozessorarchitektur, z.B. 32-Bit Architektur)
    - Seitengrösse, Grösse des logischen Speichers, Grösse eines Eintrages in der Seitentabelle
    - Berechnung: Speichergrösse/Seitengrösse = Anzahl der Einträge. Anzahl der Einträge mal Grösse eines Eintrages = Grösse der Seitentabelle.
    - **Grösse des logischen Adressraums** 2^m

19. Was ist das Konzept einer invertierten Seitentabelle, und welche Daten enthält diese?

    - Man betrachtet nur die Seitennummern (logischer Adressraum) die tatsächlich vom Prozess verwendet werden.
    - Alle Rahmennummern in Spalte f sind vorhanden und werden aufsteigen sortiert (indiziert von 0 bis Ende des RAM).

20. VL09/F20: Braucht man eine einzige invertierte Seitentabelle, oder eine pro Prozess?

    - *Es reicht eine einzige, wenn wir zu jeder Zeile (= log. Seite in Verwendung) auch die Prozessnummer abspeichern.*

21. Wie viel Speicherplatz braucht eine invertierte Seitentabelle, und welche Daten benötigt man zur Berechnung? Diese Frage kann auch in der Variante gestellt werden: Welchen Anteil des RAM braucht eine invertierte Seitentabelle?

    - Parameter zur Berechnung:
        - Grösse der PID
        - Rahmen
          Seiten-Grösse

22. VL10/F14: Was ist die virtuelle Speicherverwaltung, und was ist die Abgrenzung zum Paging?

    - *Verbindung von Paging und dem Auslagern einzelner Seiten (bzw. Seitenrahmen) auf die Festplatte*

23. VL10/F:Was ist ein Seitenfehler, und wie merkt die Hardware bzw. das Betriebssystem, dass man einen erzeugen sollte?

    - Bei dem Zugriff auf eine Seite wird zunächst geprüft:
        - Ist v/i-Bit == valid? Dann normale Adressübersetzung wie bei Paging
        - Ist v/i-Bit == invalid? Dann hat man einen sog. Seitenfehler (page fault)
    - 1. BS schaut in einer Hilfstabelle T nach
    - Ist das eine ungültige Speicherreferenz?
    - Ja, leite Fehlerbehandlung ein (Abbruch der Adressübersetzung)
    - Nein, alles OK, nur die Seite ist ausgelagert auf die FP

24. VL10/F21: Wie berechnet man die effektive Zugriffszeit unter Berücksichtigung der Seitenfehler?

    - Sei p die page-fault-rate, d.h. Anteil der Seitenfehler pro Speicherzugriff, also $0 < p < 1$.
    - $p = 0$, keine Seitenfehler
    - $p = 1$, nur Seitenfehler
    - Dauer des Speicherzugriffs = d
    - Dauer der Fehlerbehandlung = e
    - Formel ist dann: $(1 - p) * d + p * e$

25. VL10/F49 Was ist das Demand Paging? In welchem Zusammenhang benutzt man Copy-On-Write?

    - Einlagern auf Anforderung
    - VL11/F3: Ein Prozess startet ohne geladene Seiten im Speicher und lagert jede Seite beim 1. Zugriff ein.
    - Annahme: Seiten haben stets eine Kopie auf der Disk
    - Vorteile:

  - Weniger Ein-/Ausgabeoperationen
  - Weniger Speicher wird gebraucht
  - Mehr Prozesse/Benutzer
- Copy-on-Write wird bei der Prozesserzeugung verwendet.
- Bei Copy-on-Write teilen sich nach fork() die Eltern- und Kindprozesse zunächst den selben Speicher.
- Nur wenn einer der Beteiligten eine Seite modifiziert wird diese kopiert
- Erlaubt es, den fork() Befehl sehr schnell auszuführen. Kein unnötiges Kopieren, wenn anschließend exec() ausgeführt wird.

26. VL10/F25,26: Was ist der prinzipielle Ablauf der Seitenersetzung?

    (a) Finde einen Seitenrahmen f, der im Speicher ist, aber nicht viel gebraucht wird - das Opfer (victim)

    (b) Kopiere alten Inhalt von f auf die Festplatte und BS ersetze durch neuen

    (c) F26: Lagere das Opfer auf die Festplatte aus

    (d) Ändere v/i-Bit zu invalid

    (e) Bringe die benötigte Seite in den Speicher

    (f) Aktualisiere den Eintrag für die benötigte Seite in der Seitentabelle (d.h. schreibe Rahmennr. und setze auf valid)

    (g) Starte den CPU- Befehl neu (nicht gezeigt)

27. Was sind die M-Bit und R-Bit? Wann und wo werden sie gesetzt?

    - Jeder Eintrag der Seitentabelle (direkte oder hierarchische) hat u.a. ein M-Bit und ein R-Bit
    - Der M-Bit, bzw. Modify Bit wird automatisch bei einem Schreibzugriff auf die Seite gesetzt
    - Der R-Bit (Referenz-Bit) wird gesetzt, wenn eine Seite gelesen wurde

28. Was ist die Bellady-Anomalie? (Eine Erklärung als Diagramm wird bevorzugt).

    - Eigentlich sollte die Seitenfehlerrate sinkem, je mehr Seitenrahmen (RAM) zur Verfügung stehen.
    - Aber: Bei schlechten Algorithmen kann sie sogar mit mehr Rahmen steigen => die Belady-Anomalie

29. Was ist das Working set und wie definiert man diesen für die technische Nutzung?

    - F3: Programme weisen oft die Lokalitätseigenschaft auf (locality of reference), Sie beschränken Zugriffe (in jeder Phase der Ausführung) auf einen relativ kleinen Teil ihrer Seiten
    - F3: Die Menge von Seiten, in jeder solchen Phase benutzt wird, heißt der Arbeitsbereich (working set, WS)

- F8: Def.: Menge der Seiten, die in den letzten Δ Speicherzugriffen benutzt wurden.

30. Wie kann man Working Set tatsächlich effizient annähern?

    - Wir nähern Δ durch die Ausführungszeit eines Prozesses an
    - Alle T Sekunden (Epoche) wird ein Interrupt alle R-Bits aufzeichnen und anschließend löschen
    - Eine Seite ist in dem WS, wenn sie innerhalb der letzen k Epochen ($k * T$ entspricht Δ) benutzt wurde.

31. Was ist die Segmentierung und was muss der Programmierer dabei beachten?

    - Um Programme und Daten in unabhängige logische Adressräume aufzuspalten
    - Um gemeinsame Nutzung des Speichers und Schutz zu unterstützen

32. VL11/F34: Welche Systemaufrufe / Bibliotheksfunktionen sind notwendig, um eine Datei (in C) zu kopieren?

    (a) open(path, readmode) infile
    (b) creat(path, writemode) outfile
    (c) read(infile, bufferadr, buffersize) infile
    (d) write(outfile, bufferadress, readcount) outfile
    (e) close(infile) infile
    (f) close(infile) outfile

33. VL12/F8: Wie unterscheiden sich harte und symbolische Links? Erläutern Sie das an einem Diagramm.

    - Harter Link: Alle Verzeichniseinträge auf Daten sind identisch, erst wenn der letzte Verzeichniseintrag gelöscht wird, sind Daten weg. Man kann nicht zwischen den harten links unterscheiden.
    - Ein Verweis auf einen Verzeichniseintrag. Interpretation durch das Betriebssystem nötig.

34. Wie unterscheiden sich die CHS-Adressierung und die logische Adressierung der Blöcke einer Festplatte?

    - CHS-Adressierung (Physische Adresse): Tripel (Zylinder, Kopf, Sektor)
    - Logische Adresse: Alle CHS werden durchgehend indiziert (durchgezählt)

35. Was sind die gravierenden Nachteile einer zusammenhängenden Belegung (bei der Speicherung von Dateien auf einer Festplatte)?

    - Nachteile: unvermeidbare Fragmentierung; Dateilänge muss bei Erstellung festgelegt werden

36. F20: Wie funktionieren bei der Speicherung von Dateien auf einer Festplatte a. Belegung durch verkettete Listen und b. Die Verbesserung durch FAT?

    - Allgemeines Prinzip der verketteten Liste: HEAD-Pointer -> Block 0 -> Block 1 -> Block 2
    - Verbesserung durch FAT (File-Allocation-Table):
        - Zeiger auf den jeweils nächsten Block wird nicht mehr in Dateiblöcken, sondern in einer Tabelle gespeichert
        - Auf Disk: reservierter Bereich, nach dem booten in RAM geholt
        - Vorteil: Tabelle kann leicht durchsucht werden um einen freien Block zu finden. Auch Seek-Operationen sind viel schneller
        - Nachteil: Braucht viel RAM

37. F22: Was ist das Konzept der I-Nodes? Welche Probleme des FAT- Ansatzes löst es?

    - Jede Datei erhält eine spezielle Tabelle mit der Liste der logischen Indizes ihrer Datenblöcke
    - D. h. jede Datei hat nun 3 Teile:
        (a) Verzeichniseintrag, zeigt auf I-Node
        (b) I-Node
        (c) Die eigentlichen Datenblöcke
    - Vorteile:
        - Weniger RAM Verbrauch
        - Platz auf Disk der I-Node ist proportional zur Gesamtlänge der Datei
        - Platz im RAM ist proportional zur Anzahl der geöffneten Dateien

38. F:25 Wie speichert man mit den I-Nodes die Adressen der Blöcke bei sehr großen Dateien? Erläutern Sie das in Detail und mit einem Diagramm.

    - Lösung: Die letzten Einträge im I-Node indexieren sog. indirekte Blöcke, die ihrerseits Zeiger auf weitere indirekte Blöcke oder die Datenblöcke enthalten

39. In einem Dateisystem mit I-Nodes ist die maximale Länge einer Datei begrenzt. Wie können Sie die Obergrenze berechnen, und welche Angaben brauchen Sie dazu?

    - Blockgröße: I-Node Blöcke und Datenblöcke sind gleich groß
    - Zeigergröße (Größe eines I-Node Eintrags)
    - Wie viel Zeiger verweisen auf die nächste Stufe

40. VL13/F13 Welche zwei Faktoren beeinflussen die Zeit bis zum Lesen des 1. Bytes bei einer rotierenden Festplatte?

    - Zugriffszeit: Zeit der Kopfbewegung

- Drehlatenz: Zeit der Drehung bis der Kopf über dem richtigen Sektor ist

41. Wie funktionieren folgende Algorithmen zum Umordnen der Kopfbewegungen: SSTF, SCAN, C-SCAN?

    - SSTF Shortest Seek Time First:
        - Als nüchste Anfrage wird diejenige ausgewählt, die die kürzeste Zugriffszeit (bezüglich der aktuellen Position) hat
        - Anfragen werden ggf. nie ausgeführt - starvation (Verhungern)
    - SCAN:
        - Der Arm geht von einem Ende der Scheibe (z.B. außen) und geht bis zum anderen Ende (innen), und dann zurück
        - Dabei werden alle Anfragen auf dem Weg bearbeitet
        - Auch der Fahrstuhl-Algorithmus genannt
        - Problem: Neue Anfragen am aktuellen Ende kommen schnell dran, Anfragen am anderen Ende warten
    - C-SCAN:
        - Der Arm geht von einem Ende der Scheibe (z.B. außen) und geht bis zum anderen Ende (hier: innen)
        - Unterwegs werden alle Anfragen auf dem Weg bearbeitet
        - Unterschied zu SCAN: Dann aber führt der Arm an den Start zurück und beginnt von vorne
    - C-LOOK:
        - Hier führt der Arm nicht ans Ende der Disk, sondern nur bis zur Spur mit der weitesten Anfrage

42. Wie funktioniert RAID 0 und RAID 1, und was ist der Zweck von jedem dieser beiden Systeme?

    - RAID0 - Striping ohne Redundanz:
        - Die FP werden in zusammenhängende Blöcke gleicher Größe aufgeteilt, und die Dateien dadurch (implizit) in Streifen zerlegt (striping)
        - Die Daten können so parallel gelesen / geschrieben werden, wie in einem Reißverschlussverfahren
        - Es gibt aber keine Redundanz!
        - Beim Defekt einer FP sind ggf. alle Daten weg!
    - RAID1 - Mirroring:
        - Ein Verbund von mind. 2 FP für höhere Verlässlichkeit
        - Ein RAID 1 speichert auf allen Festplatten die gleichen Daten (Spiegelung)
        - Die Kapazität des Arrays ist hierbei höchstens so groß wie die der kleinsten beteiligten FP
        - Mirroring: alle FP-Scheiben am gleichen Controller
        - Duplexing: separate, selbständige FP

43. VL13/F25: Wie berechnet man die Ausfallwahrscheinlichkeit von RAID 0, und die von RAID 1, und welche Parameter sind für eine Berechnung notwendig?

    - Annahme: Eine FP fällt in 3 Jahren mit Wahrscheinlichkeit (W-keit) von 0.05 aus
    - Weiterer Parameter n = Wie viele FP befinden sich im RAID-Verbund
    - RAID1: Wahrscheinlichkeit wird mit n potenziert also 0.05^n
    - RAID0: P(mind. 1 FP fällt aus) = 1??$P(keine FP flltaus)$ = 1??"$(1-0.05)^2 = 1-(1-2*0.05+0.0025) = 0.1$??"$0.0025 = 0.0975$

44. VL13/F32: Wie funktioniert RAID 4 und welches Problem hat dieses System?

    - Aufeinanderfolgende Blöcke von Daten (z.B. 512-Bytes- Blöcke) werden auf verschiedene FP geschrieben
    - Eine zusätzliche FP speichert die Paritätsinformation
    - Wenn eine FP ausfüllt, können die n-1 Daten-FP + die Paritäts-FP zur Rekonstruktion benutzt werden
    - Schreibvorgänge sind langsamer- warum? Wenn A1 und B2 geschrieben werden, muss Disk 3 (mit den Paritätsinfos) zwei mal schreiben
    - Problem: Bei RAID 4 wird die P.-FP übermäßig benutzt und füllt schneller aus

45. VL13/F32: Wie funktioniert RAID 5, und warum vermeidet es das Problem von RAID 4?

    - Wie RAID 4 werden aufeinanderfolgende Blöcke auf verschiedene FP geschrieben
    - ABER: die Paritätsinformationen werden gleichmäßig auf allen FP verteilt
    - Z.B. bei k FP wird die Paritätsinformation für Block n auf der FP mit Index (n mod k) gespeichert
    - Warum dieser Unterschied zu RAID 4?
    - So werden alle FP gleichmäßig benutzt; das Schreiben ist schneller

46. VL14/F16: Was ist ein Deadlock? Hier ist die genaue Definition gefragt.

    - Eine Gruppe von Prozessen befindet sich in einem Deadlock-Zustand, wenn jeder Prozess aus der Gruppe auf ein Ereignis wartet, das nur ein anderer Prozess aus der Gruppe auslösen kann

47. Erläutern Sie die vier Voraussetzungen für die Entstehung eines Ressourcen-Deadlocks.

    - Wechselseitiger Ausschluss (mutual exclusion): Jede Ressource ist entweder verfügbar oder genau einem Prozess zugeordnet
    - Hold-and-Wait: Prozesse, die schon Ressourcen reserviert haben, können noch weitere Ressourcen anfordern

- Ununterbrechbarkeit (no preemption): Ressourcen, die einem Prozess bewilligt wurden, können diesem nicht gewaltsam wieder entzogen werden; der Prozess muss sie explizit freigeben
- Zyklische Wartebedingung (circular wait): Es muss eine zyklische Kette von Prozessen geben, von denen jeder auf eine Ressource wartet, die dem nächsten Prozess in der Kette gehört, d.h. in P 0 , P 1 , P 2 , ..., P n : P 0 wartet auf P 1 , P 1 auf P 2 , ..., P n auf P 0
- Gut ist: die Bedingungen sind notwendig aber nicht hinreichend und müssen gleichzeitig erfüllt sein!

48. VL14/F19: Wie können Sie die aktuelle Situation der Belegung der Ressourcen und Nachfragen nach ihnen grafisch darstellen? Wie heißt der zugehörige Graph?

    - Ressourcen-Belegungs-Graph
    - Prozesse = Knoten = P
    - Ressourcen = Knoten = R
    - Kante (P, R) = Prozess verlangt Ressource, wartet darauf
    - Kante (R, P) = Ressource wurde dem Prozess zugeordnet

49. F21: Falls es pro Ressourcentyp nur eine einzige Instanz einer Ressource gibt, wie äußert sich die Zyklische Wartebedingung in dem Graphen?

    - Der Graph bildet einen gerichteten Zyklus

50. F23: Falls wir mehrere Instanzen pro Ressourcentyp haben, können wir anhand des oben genannten Graphen leicht erkennen, ob ein Deadlock vorliegt, oder nicht? Warum?

    - Wenn kein Zyklus haben wir keinen Deadlock
    - Wenn ein Zyklus vorliegt, muss nicht zwingend ein Deadlock vorliegen

51. Zu welchem Zweck dient der Bankieralgorithmus und wann sollen wir diesen einsetzen? Wann funktioniert etwas Einfacheres zum gleichen Zweck?

    - Der Erkennung von Deadlocks bei mehreren Instanzen pro Ressource
    - Bei einer Instanz pro Ressource kann eine Tiefensuche durchgeführt werden um Zyklen zu erkennen (Deadlock)

52. VL15/F17,18: Was ist die zentrale Idee bei der Verhinderung von Deadlocks?

    - Meistens werden die Ressourcen nicht auf einmal, sondern nach und nach zugeteilt
    - Das BS kann entscheiden, ob die nächste Ressource zugeteilt wird oder doch nicht
    - Das BS sollte das nur tun, wenn es nicht gefährlich ist
    - Gibt es einen Algorithmus, der das zuverlässig entscheiden kann (und somit Deadlocks verhindert)?

- Ja, aber nur wenn bestimmte Informationen im Voraus zur Verfügung stehen
- U.a.: Maximale Anzahl der Ressourceninstanzen von jedem Typ, die ein Prozess reservieren wird

53. VL15/FWelche zwei Typen von Prozessen in Hinblick auf die CPU-Verwendung unterscheidet man?

    - Rechenintensiv (compute-bound) bzw. CPU-intensiv (CPU-bound)- sie haben lange CPU-Burst-Phasen
    - I/O-intensiv (I/O-bound) - Diese müssen i.A. nicht länger auf I/O-warten, sondern haben einfach kürzere CPU-Burst-Phasen

54. VL15/F34: Erläutern Sie die folgenden Scheduling- Strategien: FCFS, SJF, Shortest Remaining Time First.

    - First-Come-First-Served (FCFS)
        - Einfachste Lösung: Der als erster anfragende Prozess wird als erster bis zum Ende ausgeführt
        - Implementierung über eine First-In-First-Out Schlange
        - Nachteil: Lange durchschnittliche Wartezeit
        - Convoy effect: Viele I/O-intensive Prozesse werden durch einen einzigen CPU-intensiven Prozess verzögert
    - Shortest-Job-First (SJF)
        - Annahme: Wir kennen die Länge des nächsten CPU-Bursts eines jeden Prozesses
        - Der Scheduler wählt den Prozess mit der kleinsten (angenommenen) Berechnungszeit
        - Das größte Problem von SJF ist die mangelnde Kenntnis der Länge der CPU-Burst-Zeit
        - Man kann aber die BZL aus vorherigen Verhalten abschätzen - viele Verfahren sind bekannt
    - Shortest Remaining Time First
        - Eine Variation von SJF, bei der die Unterbrechbarkeit angenommen wird
        - Sei P ein (neuer) Prozess im Zustand Ready, und seine erwartete CPU-Burst-Zeit sei t b
        - Die Zeit t b wird verglichen mit: Der (erwarteten) CPU-Burst-Zeit von jedem anderen Prozess im Zustand Ready und (NEU) der restlichen CPU-Burst-Zeit des laufenden Prozesses
        - Ist t b die kleinere Zeit, wird der aktuelle Prozess unterbrochen und P ausgeführt
        - Prozesse müssen unterbrechbar sein
        - Auch hier ist das Problem der Vorhersagen vorhanden

55. Für welche Arten von Systemen/Anwendungsszenarien werden diese drei Scheduling-Strategien verwendet?

- Stapelverarbeitungssystemen

56. Welche drei Typen von Anwendungsszenarien unterscheidet man? Nennen Sie mind. je zwei Verfahren für die besprochenen zwei Typen (d.h. ohne Echtzeit)

    - Stapelverarbeitung
        - First-Come-First-Served (FCFS)
        - Shortest-Job-First (SJF)
        - Shortest Remaining Time First
    - Interaktivität
        - Round-Robin-Scheduling
        - (Prioritätsscheduling)
        - multilevel queue scheduling
        - Lotterie-Scheduling
    - Echtzeit

57. VL15/F42: Wie funktioniert das Round-Robin-Scheduling, und welche Größe kann man dabei festlegen?

    - Jedem Prozess wird eine maximale Zeitspannezugewiesen, in dem er am Stück ausführen darf - sein Quantum
    - Prozess, der gerade ausgeführt hat, kommt ans Ende einer Warteschlange (FIFO)
    - Wenn ein Prozess am Ende seines Quantums immer noch läuft, wird die CPU unterbrochen und an den nächsten Prozess in der Schlange abgegeben
    - Falls CPU-Burst kürzer als Quantum ist, wird die Kontrolle über CPU an den nächsten Prozess abgeben
    - Die Unterbrechbarkeit ist notwendig

58. Beschreiben Sie das Konzept des Prioritätsscheduling. Welche Ihnen bereits bekannte Technik ist ein Spezialfall davon?

    - Der Prozess mit der höchsten Priorität darf als erstes rechnen
    - Shortest Remaining Time First: Der Prozess mit der niedrigsten verbleibenden Rechenzeit hat die höchste Priorität

59. Erläutern Sie die Ideen des Lotterie-Schedulings.

    - Jeder Prozess bekommt Lotterielose für verschiedene Systemressourcen wie z.B. CPU
    - Bei CPU-Scheduling: Lotterie wird 50 Mal / Sek. abgehalten, jeder Gewinner darf 20 ms rechnen
    - Kooperierende Prozesse können ihre Lose tauschen

60. VL16/F29: Beschreiben Sie die Idee eines Puffer-Überlaufs-Angriffs. Erläutern Sie dabei, welche Daten auf dem Stack abgelegt werden.

- Wenn die Eingabe länger als der Puffer B ist, wird die Rücksprungadresse R überschrieben
- Eine geschickte Eingabe wird R so überschreiben, dass beim Rücksprung eigener Code ausgeführt wird
- Dieser Code wird bevorzugt innerhalb der Eingabe (bzw. B) liegen

61. N01/F3: Was bedeutet der Begriff Host, und wie wird das begründet?

    - Ein Host ist ein Endsystem, also ein Gerät am Rand des Netzwerkes
    - "to host" heißt bewirten, ein Host ist also ein Gastgeber für eine Netzwerkanwendung
    - siehe auch F17: Für uns nur Behälter für verteilte Anwendungen

62. N01/F25 Was ist das Prinzip der Leitungsvermittlung, und das der Paketvermittlung? Was wird in den heutigen Computernetzwerken verwendet?

    - Leitungsvermittlung: Die benötigten Ressource (wie Puffer, Schaltungen, Kanal) werden für die Dauer der Kommunikationssitzung zwischen diesen Endsystemen reserviert
    - Paketvermittlung: Die Ressourcen werden nicht reserviert. Die Nachrichten einer Sitzung verwenden diese Ressourcen nach Bedarf und m??ssen infolgedessen ggf. warten
    - Es wird Paketvermittlung verwendet

63. N01/F30: Was sind die wesentlichen Vorteile, und die wesentlichen Nachteile der Paketvermittlung?

    - Vorteile
        - Ressourcen werden nur nach Bedarf verwendet
        - Pakete werden mit der vollen Übertragungsgeschwindigkeit der Leitung übertragen
        - F28: Statistisches Multiplexing ist effizienter als Zeitmultiplexverfahren. Host A vs. Host B. Host A hat mehr Pakete im Routerbuffer, dann werden auch mehr Pakete von A verschickt.
        - F29: Mehr nutzer bei Paketvermittlung als bei Leitungsvermittlung
    - Nachteile
        - Kumulativer Bandbreitenbedarf kann die Kapazität überschreiten, keine Dienstgarantie für einen Benutzer
        - Pakete werden verzögert, insbesondere wenn viele senden. Schlecht für Echzeitdienste wie Audio/Video
        - Pakete gehen verloren, wenn Switch-Buffer überlaufen

64. Benennen Sie die möglichen Arten der Verzögerungen bei der Paketvermittlung.

    (a) Verarbeitungsverzögerung

        - Die Zeitdauer, welche zur Prüfung des Paket-Headers sowie zur Entscheidung über den weiteren Weg des Paketes benötigt wird

- Zeit, um nach Bitfehlern der Übertragung zu suchen

(b) Warteschlangenverzögerung
- Wartezeit im Puffer, bis das Paket über entsprechende Leitung versendet werden kann
- Hängt von der Länge der Schlange ab

(c) Übertragungsverzögerung
- R = Übertragungsgeschwindigkeit (link bandwidth) (in bps)
- L = Paketlänge (bits)
- Übertragungszeit = L/R

(d) Ausbreitungsverzögerung
- d = Länge der physischen Leitung
- s = Ausbreitungsgeschwindigkeit
- Ausbreitungsverzögerung = d/s

(e) Gesamtverzögerung pro Übertragungsknoten
- $d_{node} = d_{verarb} + d_{warte} + d_{ber} + d_{ausb}$

65. F34: Welche Art der Verzögerung hängt mit der Bandbreite einer Leitung zusammen, welche mit der Entfernung zwischen Endpunkten der Übertragung?
    - Bandbreite –> Übertragungsverzögerung
    - Entfernung –> Ausbreitungsverzögerung

66. N01/F37: Welche Informationen ermittelt das Dienstprogramm traceroute?
    - Traceroute erlaubt die Messung der Verzögerung von der Quelle bis zu jedem Router auf dem Weg zum Ziel
    - Für alle i = 0,...,255
        - Sende drei spezielle Pakete, die den Router i auf Pfad zu Ziel erreichen
        - Router i schickt die Pakete zum Sender (Quelle) zurück
        - Sender misst die Round-Trip-Time

67. N02/F13: Benennen Sie die fünf Schichten des Internet-Protokollstapels.
    (a) Anwendungsschicht / application-layer / HTTP, FTP, SMTP / Nachricht
    (b) Transportschicht / transport-layer / TCP, UDP / Segment
    (c) Netzwerkschicht / network-layer / IP, Routing-Protokolle / Datagramm
    (d) Sicherungsschicht / data-link-layer / PPP, Ethernet / Rahmen
    (e) Bitübertragungsschicht / physical-layer / Hängt vom Medium ab

68. F14: Durch welche Endpunkte der Kommunikation wird jeder der Schichten identifiziert? Geben Sie weiterhin für jede Schicht mindestens ein Beispielprotokoll an.

(a) Anwend.: Mehrere Programme, jedes kann aus einem oder mehreren Prozessen bestehen (z.B. Browser + Server)

(b) Transp.: Prozesse (bzw. zugehörige Sockets - API des BS auf einem oder mehreren Hosts)

(c) Netzw.: Start- und End-Hosts (es wird nicht zwischen den Prozessen auf einem Host unterschieden)

(d) Sicher.: Zwei direkt verbundene Geräte (Host, Router, Switch) an den beiden Enden einer Teilstrecke

(e) Bitüber.: Elektronik der Leitungen oder Glasfaser an den Enden einer Teilstrecke

69. F15: Welche dieser fünf Schichten implementieren jeweils: Hosts, Router, Switches?

    - Hosts: Alle 5 Schichten
    - Router: von Bitübertragungsschicht bis einschl. Netzwerkschicht
    - Switches: Bitübertragunsschicht und Sicherungsschicht

70. F25: Wenn zwei Prozesse miteinander kommunizieren, die Komponenten einer Netz-Anwendung sind, wie erkennt man unter diesen den Client, und wie den Server (hier ist die Funktion gemeint, und nicht die Anwendungsarchitektur)?

    - Der Prozess, der die Kommunikation eröffnet (also erstmals den anderen Prozess zu Beginn der Sitzung kontaktiert), wird als Client bezeichnet
    - Der Prozess, der darauf wartet, zu Beginn einer Sitzung angesprochen zu werden ist der Server

71. Was ist der Unterschied zwischen WWW und Internet, sowie zwischen HTTP und HTML (Definition reicht hier, herausstellen der Unterschiede macht zum Teil wenig Sinn)?

    - Das World Wide Web ist die bekannteste Netzwerkanwendung
    - WWW besteht aus: HTTP, HTML, URL
    - *The Internet is the global system of interconnected computer networks that use the Internet protocol suite (TCP/IP) to link devices worldwide. It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies.*
    - HTTP-Protokoll dient zur Übertragung von Daten, das sind häufig HTML-Dateien

72. F41: Welches Protokoll der Transportschicht benutzt HTTP? Was merkt sich der Server bzw. der Client zwischen den Anforderung/Antwort-Paaren?

    - HTTP verwendet TCP
    - HTTP ist zustandslos: Server behält keinen HTTP-basierten Zustand zwischen Anfragen

73. F42, F45: Geben Sie konzeptionell (grob) das Format einer HTTP-Anforderung und einer HTTP-Antwort an.

    - Request: Besteht aus einer Request-Zeile mit 3 Feldern. Methodenfeld, URL-Feld, HTTP-Versionsfeld.
    - Request: Danach kommen Headerzeilen, z.B. (Webserver-Adresse / Host, Browsertyp / User-agent, Verbindungstyp / Connection, bevorz. Sprache / Accept-language)
    - Request: Danach kommt der Entity-Body. Dieser ist bei GET-Methode leer, er wird nur bei der POST-Methode (Formulare) verwendet.
    - Response: Einleitende Statuszeile mit 3 Feldern. HTTP-Protokollversion, Statuscode, Statusnachricht.
    - Response: Headerzeilen
    - Response: Entity-Body (Daten)

74. F50: Was sind Cookies und welchem Zweck dienen sie? Erläutern Sie diese anhand eines Diagramms.

    - Sie erlauben dem Server, die Clients wiederzuerkennen

75. VLN03/F9: Warum gibt es Überhaupt UDP, obwohl es keinen zuverlässigen Datentransfer garantiert?

    - Schnell, Sprachübertragung, Videoübertragung, weniger Overhead als TCP.
    - F9: Kein Handshake nötig, kleinere Verzögerung
    - F9: Kein Verbindungszustand an den Enden, weniger Speicher nötig
    - Kleinerer Header
    - F9: Keine Überlastkontrolle: Man kann theoretisch so schnell senden wie gewünscht.

76. F10: Was sind die vier Felder in dem Header eines UDP-Pakets?

    - Quell-Port Nummer 2 Byte
    - Ziel-Port Nummer 2 Byte
    - Länge 2 Byte
    - Prüfsumme 2 Byte

77. F15: Was sind die Sockets und was sind die Ports?

    - Socket: Schnittstellen und Objekte des Betriebssystems
    - Sockets sind ähnlich zu Pipes
    - Jedem Socket wird ein Port zugeordnet. Das ist die ID des Sockets auf dem Host (16 Bit)

78. F16: Welche zwei Typen von Sockets kann ein Anwendungsprogrammierer benutzen? Welche Typen von Protokollen der Transportschicht verwendet jeder dieser Typen?

- Verbindungslose Sockets via UDP
- Verbindungsorientierte Sockets via TCP. Wir tun so, als ob wir eine Leitung hätten aber nur die Endpunkte wissen davon.
-

79. F17, F18: Beschreiben Sie konzeptionell (möglicherweise durch ein Diagramm), wie die Kommunikation zwischen Client und Server mit den UDP-Sockets funktioniert. Sie brauchen hier die Namen der API-Aufrufe nicht zu kennen.
    (a) Client: Erzeugt ein DatagramSocket();
    (b) Client: Holt sich IPAdresse mit getByName("servername")
    (c) Client: erzeugt DatagramPacket(sendData, sendData.length, IPAdress, 9876)...

80. F25: Was für eine Art Abstraktion liefert eine TCP-Verbindung? Welche Zusicherungen werden hier für den Anwendungsprogrammierer gemacht?
    - Leitungsvermittlung aus N01/N02 aufgreifen
    - Virtuelles Rohr
    - zuverlässiger Bytestrom-Dienst
    - Garantiert dass jedes vom Client versandte Byte in der ursprünglichen Reihenfolge den Server-Prozess erreicht.

81. F26: Beschreiben Sie konzeptionell, was aus der Sicht des Anwendungsprogrammierers auf der Serverseite bei einer TCP-Verbindung passiert. Insbesondere: welche Art von Sockets werden verwendet, und was es ihr Funktion?
    - Welcome Socket, wird vom Client angesprochen um eine Verbindung aufzubauen. Hat einen bekannten Port.
    - welcome socket erstellen
    - warte auf ankommende Verbindungsanfragen
    - TCP-Verbindungsaufbau
    - Daten annehmen von connectionSocket
    - Antwort verschicken über connectionSocket
    - Schließen von connection Socket
    - Connection Socket, wird nur für diese Verbindung erstellt

82. : Beschreiben Sie analog die Sicht des Anwenderprogrammierers auf der Clientseite (bei TCP). Wenn der Client mehrmals Daten an den Server sendet (über die gleiche Verbindung), wie macht der das, und welche Informationen muss er/muss er nicht jedes Mal mitgeben? Vergleichen Sie das auch mit UDP-Programmierung.
    - tcp ist bidirektional
    - nach erstellung von connection socket sind beide (Server und Client) gleichwertig

83. F37: Definieren Sie den Begriff Multiplexing und Demultiplexing.

    - Demult.: Ankommende Nachrichten müssen an den richtigen Socket gelangen.
    - Demult. beim Empfang: Abliefern der empfangenen Segmente an den richtigen Socket
    - Multiplexing beim Schreiben

84. F39: Wie funktioniert das Demultiplexing bei UDP? Insbesondere, welche Daten dienen als Kriterium für das Finden des richtigen Zielsockets?

    - Der Wert der Portnummer des Ziels entscheidet an welchen Socket das Segment geht.
    - und nat IP-Adresse

85. Wie funktioniert das Demultiplexing bei TCP? Auch hier, welche Daten denen zum Auffinden des korrekten Zielsockets?

    - Problemschilderung auf F42

86. N04/F3: Benennen Sie drei von den sechs grundlegenden Eigenschaften des TCP-Protokolls.

    (a) Point-to-Point: 1 Sender, 1 Empfänger
    (b) Zuverlässiger Datenstrom: Zustellung der Nachrichten "garantiert", es gibt keine "Grenzen"der Nachrichten
    (c) Pipelined: Daten kommen in der Reihenfolge des Sendens an (in der Anwendungsschicht)
    (d) Vollduplex: Bi-direktionaler Datenfluss
    (e) Verbindungsorientiert: Handshaking initialisiert den Zustand des Senders / Empfängers vor dem Datentransfer
    (f) Flusskontrolle: Der Sender wird den Empfänger nicht überfluten
    (g) Überlaststeuerung / Staukontrolle: Sender passt sich an die (aktuelle) Bandbreite der Leitung an

87. N04: Neben der Portnummern der Quelle und des Ziels, enthält der Header eines TCP Segmentes die Sequenz- Nummer und die Acknowledgement-Nummer. Erläutern Sie an einem Diagramm die Funktion dieser beiden Nummern (z.B. telnet-Szenario).

    - Sequence: Position des Pakets im Datenstrom, Index des ersten Bytes des Payloads des Pakets im Datenstrom.
    - Acknowledgment: Index des nächsten vom Empfänger erwarteten Bytes (in dem ÄntwortDatenstrom, d.h. B zu A)
    - Inkrement der Sequence# wird aus dem Payload berechnet
    - Seq#: Notwending, um die Pakete beim Empfänger in die richtige Reihenfolge zu bringen (und eig. eine allgemeine Indizierung der Pakete)

- ACK#: Zeigen dem Sender, dass Daten angekommen sind und ggf. welche nochmals geschickt werden müssen

88. N04/F10: Beschreiben Sie kurz die drei Phasen des 3-Wege-Handshake. Welche Bits werden wann gesetzt, und welche Rollen spielen dabei die Sequenz-Nummern und die Acknowledgement-Nummern?

    (a) Client sendet ein SYN-Segment: Seq# := zufällige Zahl = client_isn; SYN_Bit := 1; keine Daten

    (b) Server antwortet mit SYNACK: SYN_Bit := 1; ACK = client_isn + 1; Seq# := zufällige Zahl = server_isn; Server belegt Puffer

    (c) Client empfängt SYNACK und antwortet mit ACK-Segment: ACK := server_isn + 1;

89. N04/F19,20: Was ist die Flusssteuerung, und wie wird sie umgesetzt? Die Namen der Variablen sind nicht wichtig, nur das Konzept.

    - Wie schafft man es, dass der Sender dem Empfänger nicht mehr Daten schickt, als dieser gerade verarbeiten kann?
    - Verhindert das überfällen der Puffer, die die Hostst auf jeder Seite der Verbindung einrichten.
    - Die Anwendungsschicht holt Daten von diesem Puffer ab. Dies kann aber unregelmäßig geschehen, dann könnte der Pufferplatz ausgehen und Pakete würden verloren gehen.
    - Lösung: Die Hosts teilen sich gegenseitig mit wie viel Platz sie selbst noch im Puffer haben.
    - Im Header ist das Feld RcvWindow = Empfangsfenstergröße dafür vorgesehen
    - Ablauf
        (a) Host A schickt RcvWindow_A im Header mit
        (b) Host B sendet dann maximal RcvWindow_A Bytes an Host A

90. N04/F25: Wie funktioniert das Stop-and-Wait-Protokoll für die verlüssliche Nachrichtenzustellung?

    - Hauptidee: Bestätigung von Paketen
    - Fehlerfreie Übertragung:
        (a) Sender schickt ein Paket und startet einen Timer. SEQ_X kommt in den Header.
        (b) Empfänger antwortet mit einer Bestätigung ACK_X
        (c) Wenn Sender ACK_X empfängt schickt er SEQ_X+1
    - Fehlerhafte Übertragung:
        (a) Paketverlust:
            i. Das Paket SEQ_X geht verloren
            ii. Es kommt also kein ACK_X zurück
            iii. Timer läuft ab
            iv. Sender schickt SEQ_X erneut

(b) Verlust von ACK
   i. ACK_X geht verloren
   ii. Timeout
   iii. Sender schickt SEQ_X erneut
   iv. Empfänger erkennt Duplikat SEQ_X und sendet ACK_X erneut

(c) Verfrühtes Timeout / doppeltes Packet
   i. Verbindung zu langsam, verfrühter Timeout
   ii. Empfänger erkennt Duplikat SEQ_X und sendet ACK_X erneut
   iii. Warum? Weil Empfänger nicht weiß ob ACK_X verloren gegangen ist

91. N04/F30: Was ist der gravierende Nachteil dieses Protokolls, und mit welcher kleinen Änderung kann man diesen Nachteil beseitigen?

    - Sehr ineffizient und das auch im fehlerfreien Ablauf, da der Sender immer auf das ACK jedes Pakets wartet.
    - Der Sender sendet nur einen Bruchteil der Round-Trip-Time. Vor allem bei großer Ausbreitungsverzögerung sinkt dann der Durchsatz dramatisch.
    - Die Lösung: Pipelining, also Pakete gruppenweise rausfeuern
    - Sender wartet nur auf Bestätigung von n Paketen. n größer 1
    - Das erfordert die Pufferung der Pakete beim Sender und Empfänger
    - Pipelining erhöht die Auslastung um den Faktor n

92. N04/F34: Was ist das wesentliche Verhalten des Selektive Repeat-Protokolls? Benennen Sie die wichtigste Regel für den Empfänger, und die wichtigste Regel für den Sender.

    - Empfänger bestätigt jedes korrekt empfangene Paket individuell
    - Sender schickt erneut nur diese Pakete, für die k

93. N04/Eigentlich F37: Was ist das Sliding Window, und wann wird das linke Ende davon (mit kleinsten Sequenznummern) jeweils beim Empfänger und beim Sender verschoben?

    - Sendefenster: Sender nach Erhalt der nächsthöheren ACK
    - Empfänger: Reserviert Buffer für erwartete Pakete, Empfangsfenster
    - F37 nochmal aufarbeiten

94. Was ist das wesentliche Verhalten des Go-Back-N-Protokolls? Vergleichen Sie es kurz mit dem Selektive Repeat-Protokoll.

    - F 7 N05
    - nur die ACK für das letzte Paket das korrekt und in richtiger Reihenfolge empfangen wurde
    - Pakete außerhalb der Reihenfolge (aus der Zukunft) werden verworfen

95. Bei TCP-Protokoll: was ist die wichtigste Verhaltensregel des Empfängers?

    - Kumulative Bestätigung

96. N06/F7: Auf welche Art kann man innerhalb einer (großen) Institution eine Hierarchie von Subnetzen erzeugen? Welcher Teil der IP-Adresse spielt dabei eine Rolle, und wie wird dieser weiter unterteilt?

    - Die hostid (gesamte IP-Adresse abzgl. netid) wird weiter in subnetid und hostid (im Subnetz) unterteilt.

97. N06/F7: Eine Institution hat ca 2 hoch 16 (= 65536) IP-Adressen bekommen. Wenn man innerhalb dieser Institution 64 Subnetze erzeugen möchte, wie müsste man den Hostteil unterteilen, und wie viele IP-Adressen kann jedes Subnetz maximal haben?

    - netid = 16 bits
    - hostid = 16 bits
    - subnetid = 6 bits
    - hostid(nach subnetid) = 10 bits
    - also 2^{10} IP-Adressen pro Subnetz

98. F8: Was ist der Zweck einer Subnetzmaske? Geben Sie Beispiele für die beiden Schreibweisen zur Angabe dieser Maske an (d.h. Längenschreibweise und Maskenschreibweise).

    - Subnetzmaske (subnet mask) identifiziert, welcher Teil der Adresse uns zu dem Subnetz führt (Netid + Subnetz-Id), welcher (restliche) Teil zu Hostid gehört
    - Die eigene IP-Adresse in Verbindung mit der Subnetzmaske erlaubt Rückschlüsse darüber, wo sich eine andere IP-Adresse befindet:
    - Im selben Subnetz (also direkt erreichbar)
    - Im selben Netzwerk, aber in einem anderen Subnetz
    - In einem anderen Netzwerk

99. F10: Wie kann man mithilfe einer Subnetzmaske bestimmen, ob der Empfänger eines Pakets im gleichen Subnetz wie der Sender liegt?

    - Subnetz der eigenen IP-Adresse: 134.155.48.10 & 255.255.255.0 = 134.155.48.0
    - fremde Adr. A: 134.155.48.96 & 255.255.255.0 = 134.155.48.0 => gleiches Subnetz
    - fremde Adr. B: 134.155.55.96 & 255.255.255.0 = 134.155.55.0 => verschieden, anderes Subnetz
    - Zieladresse wird durch & mit der Subnetzmaske verknüpft
    - Ergebnis wird mit Eigener IP-Aresse (verknüpft mit Subnetzmaske)

100. F14: Was ist die Hauptbeschäftigung eines Routers? Was macht so ein Gerät in seiner Freizeit?

- Weiterleitung: Router liest die Zieladresse des Pakets ab
- Findet einen passenden Eintrag in der Weiterleitungstabelle (forwarding table)
- Leitet das Paket über den gefundenen Ausgang weiter
- Ein Router macht eine lokale Entscheidung: Paket an einen Ausgang schicken

101. F16: Die Einträge der Weiterleitungstabelle eines Routers codieren Bereiche von IP-Adressen. Wie sieht diese Kodierung in Detail aus? Verdeutlichen Sie sich an einem Beispiel, wie man mithilfe dieser Kodierung schnell feststellen kann, ob eine IP-Adresse in einem IP-Bereich (festgelegt durch einen Eintrag der Weiterleitungs-tabelle) liegt.

    - Man behält nur das gemeinsame Präfix aller IP-Adressen die zum selben Ausgang weitergeleitet werden
    - Wir kodieren die Ausgänge durch die Präfixe (das sind die k oberen Stellen) die in der unteren und oberen Adr.-Grenzen übereinstimmen
    - Router leitet N = 2^{32-k} verschiedene Adressen an diesen Ausgang
    - Longest Prefix Matching (LPM): Passen mehrere Präfixe zu einer Zieladresse, nehmen wir den Ausgang mit der längsten Prefix-Übereinstimmung

102. F22: Angenommen, ein Router leitet alle Pakete mit der Ziel IP-Adresse aus einem IP-Adressenbereich a.b.c.0 bis a.b.c.255 auf Ausgang 0. Können Sie erreichen, dass die Pakete mit den Ziel IP-Adressen aus dem Bereich a.b.c.0 bis a.b.c.7 auf Ausgang 1 (statt 0) ausgegeben werden? Falls ja, welche Regeln verwenden Sie hier? Verdeutlichen Sie sich das an einem konkreten Beispiel, insbesondere geben Sie die Einträge der Weiterleitungstabelle für diesen Router an.

    - a.b.c.00000 –> Ausgang 1
    - a.b.c. –> Ausgang 0

103. F23: Für Fortgeschrittene: Können Sie (zusätzlich zu der obigen Konfiguration) erreichen, dass Pakete mit den Ziel IP-Adressen im Bereich a.b.c.8 bis a.b.c.15 auf Ausgang 2 weitergeleitet werden? Geben Sie dazu den konkreten Eintrag in der Weiterleitungstabelle an.

    - a.b.c.00000 –> Ausgang 1
    - a.b.c.0000 –> Ausgang 2
    - a.b.c. –> Ausgang 0

104. F28: Beschreiben Sie kurz das Hauptkonzept und die vier Phasen des DHCP-Protokolls.

    - Host schickt die Nachricht DHCP discover (Broadcast)
        - UDP-Paket an Port 67
        - Broadcast-IP-Zieladresse 255.255.255.255, Quelladresse: 0-en
        - transaction ID: 654
    - DHCP-Server antwortet mit DHCP offer (Broadcast)

  - UDP-Paket an Port 68
  - yiaddr = (your IP address) = angebotene IP-Adresse wird mitgeschickt
  - Lifetime: 3600 secs
  - transaction ID: 654
- DHCP request
  - Host wählt einen DHCP-Server aus und schickt ihm die Nachricht DHCP request
  - Nicht ausgewählte Server interpretieren das als Ablehnung
  - transaction ID: 655
- DHCP ack
  - transaction ID: 655
- DHCP release

105. F32: Welchen Zweck hat die Source-NAT? Beschreiben Sie, was passiert, wenn ein Paket des lokalen Netzwerks den NAT-Router verlässt, und was passiert, wenn eine Antwort auf dieses Paket zurückkommt.

- Motivation: Lokales Netzwerk (LN) besitzt für die Außenwelt nur eine einzige IP-Adresse
- Kompensiert die Knappheit öffentlicher IPv4-Adressen
- Man kann die Adressen im LN ändern, ohne die Außenwelt informieren zu müssen
- Bietet zusätzliche Sicherheit, da die internen Eigenschaften / Hosts des LNs nach außen unsichtbar bleiben
- Alle Datagramme, die den NAT-Router verlassen, haben dieselbe Quell-IP-Adresse: 138.76.29.7
- Datagramme, die in diesem Netzwerk erzeugt wurden, haben 10.0.0/24 als Bereich der Quell-IP-Adressen
- Der Router ersetzt die QuellPort-Nummern (in den ausgehenden Paketen) mit verschiedenen Quell-Port-Nummer für A und für B; Router übersetzt damit die Antworten
- Prinzip: die Quell-Port# wird missbraucht, um nicht nur zwischen den Anwendungen, sondern auch zw. den Hosts (im lokalen Netzwerk) zu unterscheiden
- (Quell-IP-Adresse, Quell-Port #) <=> (öffentliche-IP-Adresse, neue Quell-Port#)

106. F38: Wie wird das ICMP-Protokoll (Internet Control Message Protocol) bei dem Tool traceroute verwendet?

- Quelle schickt eine Folge von UDP-Segmenten zum Ziel
- 1. hat TTL = 1
- 2. hat TTL = 2, usw.
- Zielport möglichst unbenutzt

- Wenn das n-te Datagramm beim n-ten Router ankommt:
- Router verwirft das Datagramm
- ... und sendet eine ICMP-Warnmeldung an die Quelle (Typ11, Code 0)
- Diese Warnmeldung enthält die IP-Adresse des Routers
- Wenn diese ICMP Nachricht bei der Quelle ankommt, kann diese aus dem laufenden Timer die RTT (Round Trip Time) zum n-ten Router ablesen
- Sobald das Quellsystem diese besondere ICMP-Nachricht erhält, weiß es, dass es keine weiteren Testpakete absenden muss
- Stop-Kriterium wird durch Firewalls geblockt (es kommt keine Antwort)

107. N07/F5, F6 : Wie modelliert man ein Netzwerk aus der Sicht der Routingalgorithmen? Welche drei Bestandteile hat so ein Modell?

    - Als Graph
    - Knoten sind Router
    - Kanten sind Leitungen
    - Gewichteter Graph
    - Kosten eines Pfades sind alle Kanten summiert

108. N07/F9: Geben Sie den Pseudocode für den Dijkstra-Algorithmus an, und erläutern Sie die Bedeutung der einzelnen verwendeten Datenstrukturen.

    - siehe F10

109. N07/F12 Wir haben bei unserer Variante des Algorithmus die Größe p(v) eingeführt, die den letzten Knoten vor dem Ziel v auf einem momentan kostengünstigsten Pfad von der Quelle u zu v an gibt. Wenn der Dijkstra-Algorithmus fertig ist (für die Quelle u), wie kann u seine eigene Weiterleitungstabelle aus p(v) berechnen?

    - Rückwärts gehen auf kürzestem Weg

110. F15: Geben Sie die Bellmann-Ford-Gleichung zusammen mit der Beschreibung der relevanten Größen an.

    - $d_x(y)$
    - $c(v, w)$
    - $d_x = min\{c(x, y) + d_v\}$

111. F17: Angenommen, der Knoten u kennt für jeden seinen Nachbarn v einen kostengünstigsten Pfad (und seine Kosten) von v zu dem Zielknoten z. Wie könnte u die Bellmann-fort-Gleichung verwenden, um die Kosten eines kostengünstigsten Pfads zum Zielknoten z zu berechnen, und welche Angaben muss u dazu noch kennen?

    - $c(u, v)$ für alle v in $neigh(u)$

112. F19: Beschreiben Sie das Verhalten jedes Knoten, welcher an dem Distanzvektor-Routing-Algorithmus (DV- Algorithmus) teilnimmt (das eventuell als Pseudocode).

    - Knoten v
    - while(TRUE):
        (a) warte auf eine Änderung von c(v, n), oder Distanzvektor-Aktualisierung
        (b) Berechne die neuen Schützungen im Distanzvektor
        (c) Falls der Distanzvektor zu irgendeinem Knoten verändert wurde, benachrichtige die Nachbarn

113. F27: Was sind die Endpunkte der Kommunikation in der Sicherungsschicht?

    - Knoten: Hosts, Router, Switches
    - Links = Kanten = Leitung
    - Zwei direkt verbundene Geräte

114. F7: Wie bestimmt man die MAC-Adresse eines Hosts aus seiner IP-Adresse? Wie funktioniert das entsprechende Protokoll?

    - ARP = Address Resolution Protocol

115. F24, F25: Beschreiben Sie kurz das Konzept eines Verfahrens, das Ethernet benutzt, um Fehlertoleranz trotz möglicher Kollisionen auf einem gemeinsamen Medium zu erreichen.

    - CSMA/CD-Algorithmus