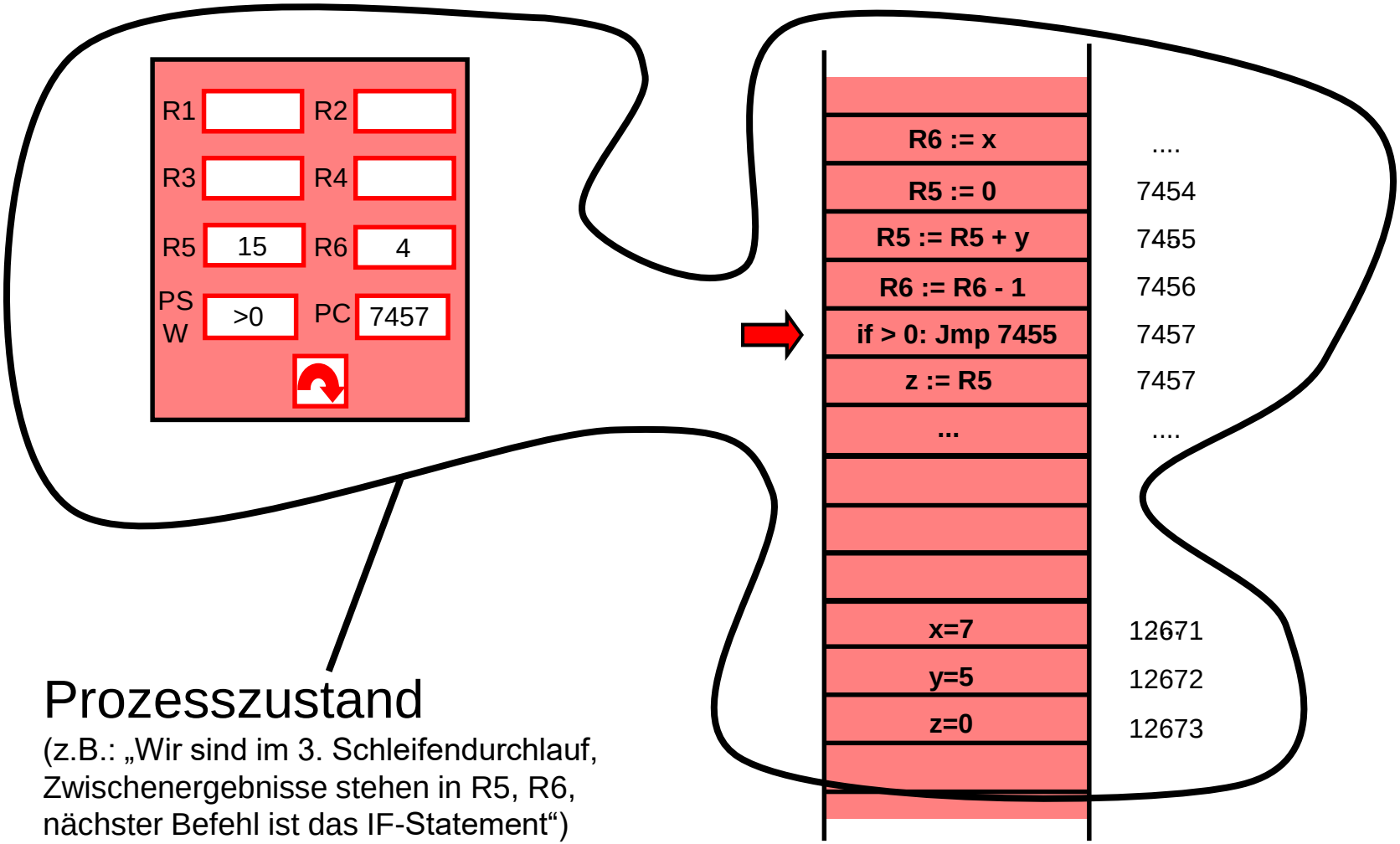
Betriebssysteme
3. Prozesse

Tobias Lauer

Prozesse – eine Einführung

- Zentrales Konzept von Betriebssystemen
- Prozess = aktive Ablaufinstanz des Rechenvorgangs
- Prozess = „Programm in Ausführung“
- Was gehört zum Prozess?
  - Programmcode (in Maschinensprache)
  - Speicherbereiche für das Programm
  - weitere Betriebsmittel (z.B. Dateien, E/A-Geräte, etc.)
- Wie kann man beschreiben, in welchem Zustand ein Prozess ist?
  - Was ist der nächste Maschinenbefehl, der abgearbeitet wird
  - Was ist der Inhalt der Speicherelemente, die dem Prozess gehören
  - Was ist der Zustand der E/A-Geräte und anderen Betriebsmittel

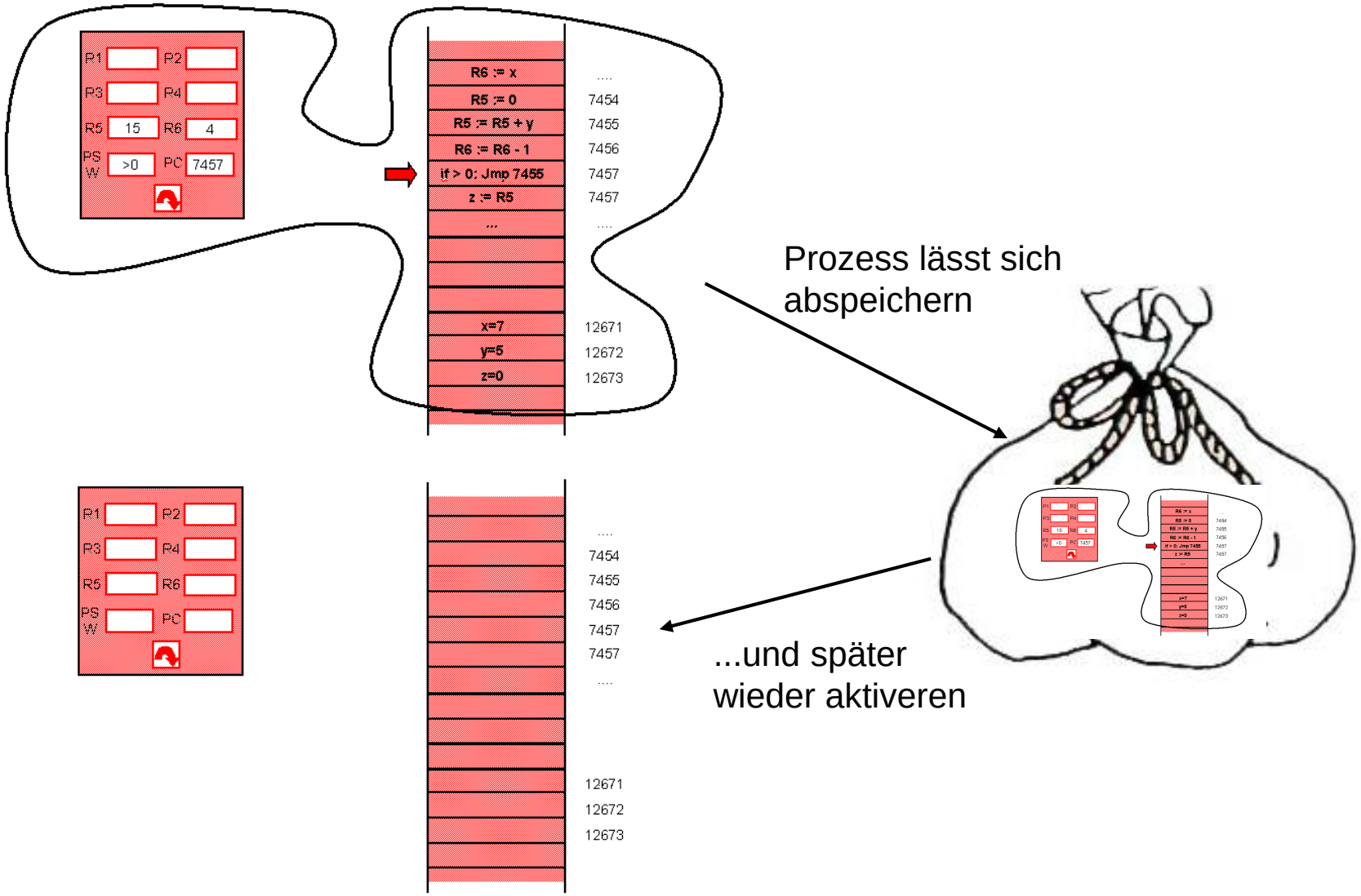
Das Beispielprogramm läuft als Prozess



Prozesszustand

(z.B.: „Wir sind im 3. Schleifendurchlauf, Zwischenergebnisse stehen in R5, R6, nächster Befehl ist das IF-Statement“)

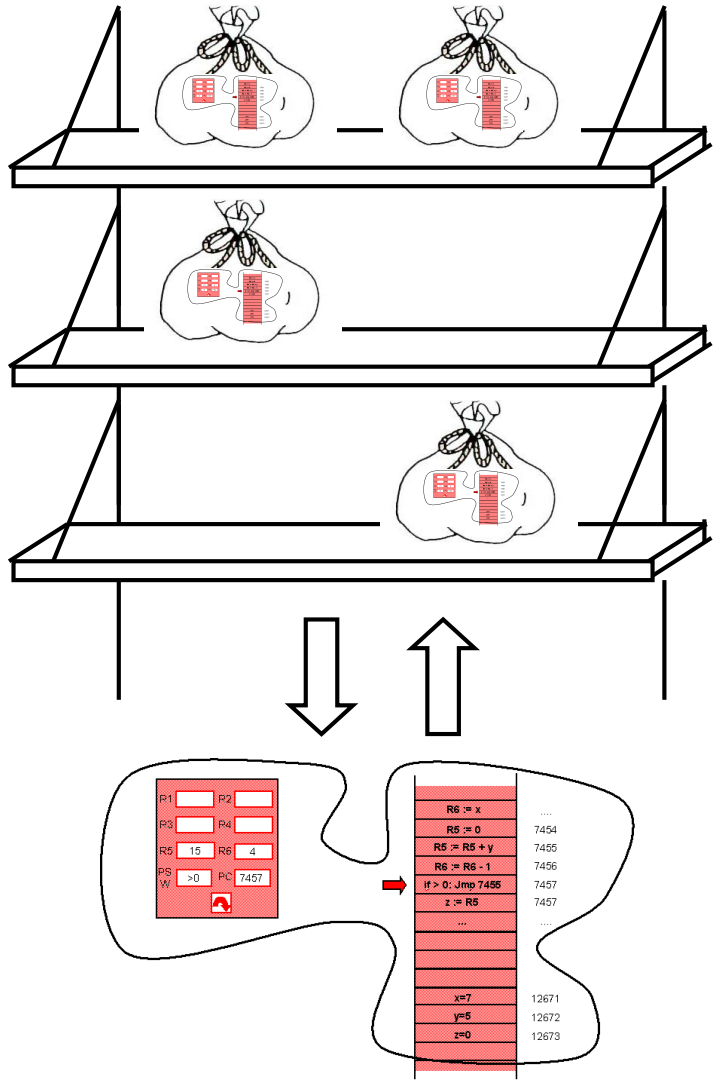
Das Beispielprogramm läuft als Prozess



Viele Prozesse auf einem Rechner

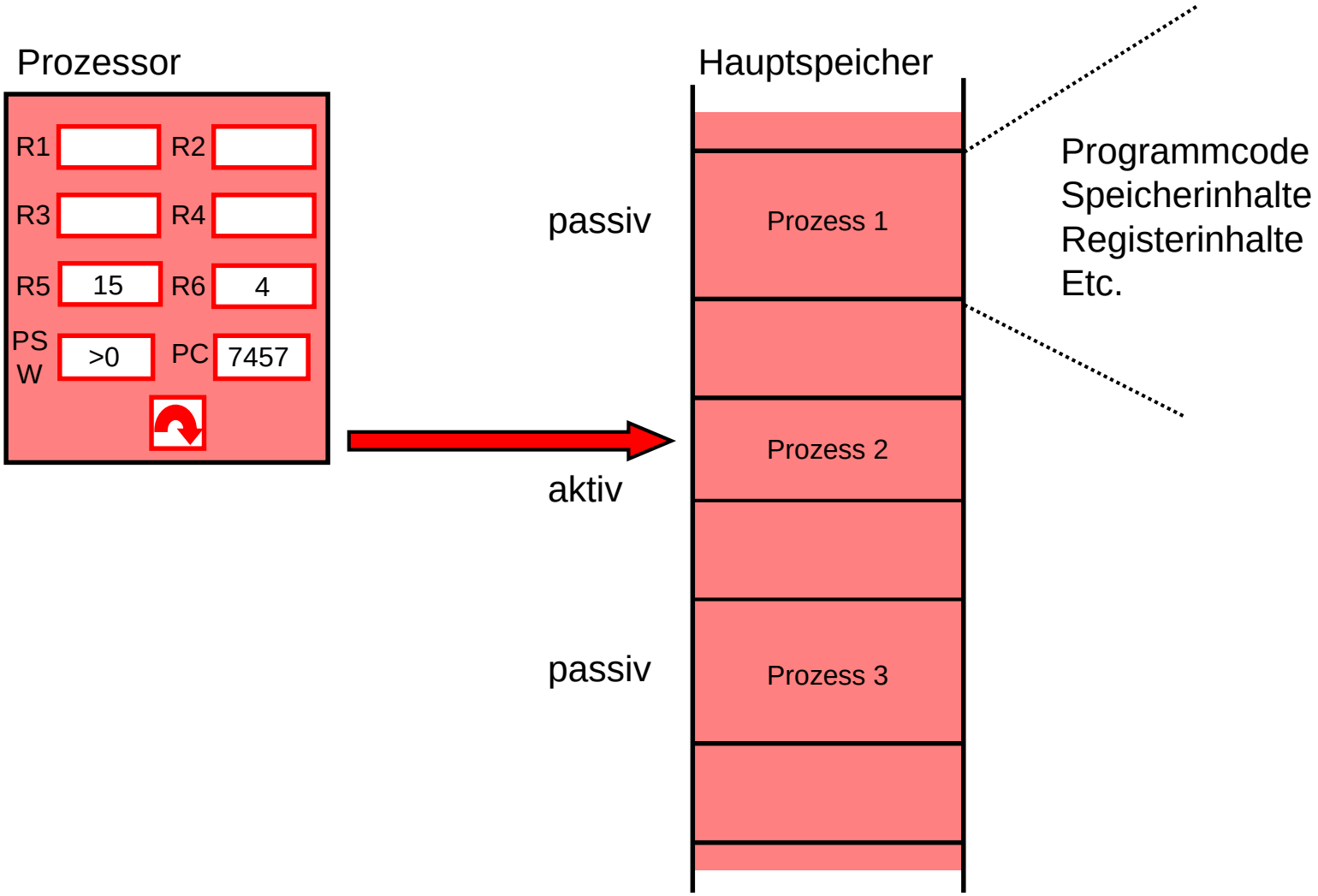
- Zu jedem Zeitpunkt sind auf einem Rechner viele Prozesse (= ausführende Programme) vorhanden
  - Viele passiv (schlafend, „im Regal“)
  - Einer* aktiv (rechnend)
- Beispiel: MS PowerPoint aktiv, Drucker-prozess wartend, Email wartend, etc.
- Betriebssystem verantwortlich für Prozessverwaltung

⇨ Zuteilung des Prozessors* an Prozesse



* Wir nehmen hier an, dass der Rechner nur einen Single-Core-Prozessor hat.

Prozesszustände werden selbst im Speicher abgelegt



Prozesskontrollblöcke

- beinhalteten alle wesentlichen Attribute eines Prozesses
  - Prozess-ID (eindeutiger Identifikator, „key“)
  - Registerinhalte (inkl. PSW, Kellerzeiger, etc.)
  - Letzter Programmzähler
  - Zeiger auf Code und Daten
  - Priorität
  - ggf. Referenz auf Vaterprozess
  - Statistische/Accounting-Daten (CPU-Zeit, Speicherverbrauch, ...)
  - ggf. Timeout-Tabelle
  - Attribute der Dateiverwaltung (Rechte, offene Datei-Zeiger, etc.)
  - ...

- Prozesskontrollblöcke sind im geschützten Datenbereich des Betriebssystems abgelegt

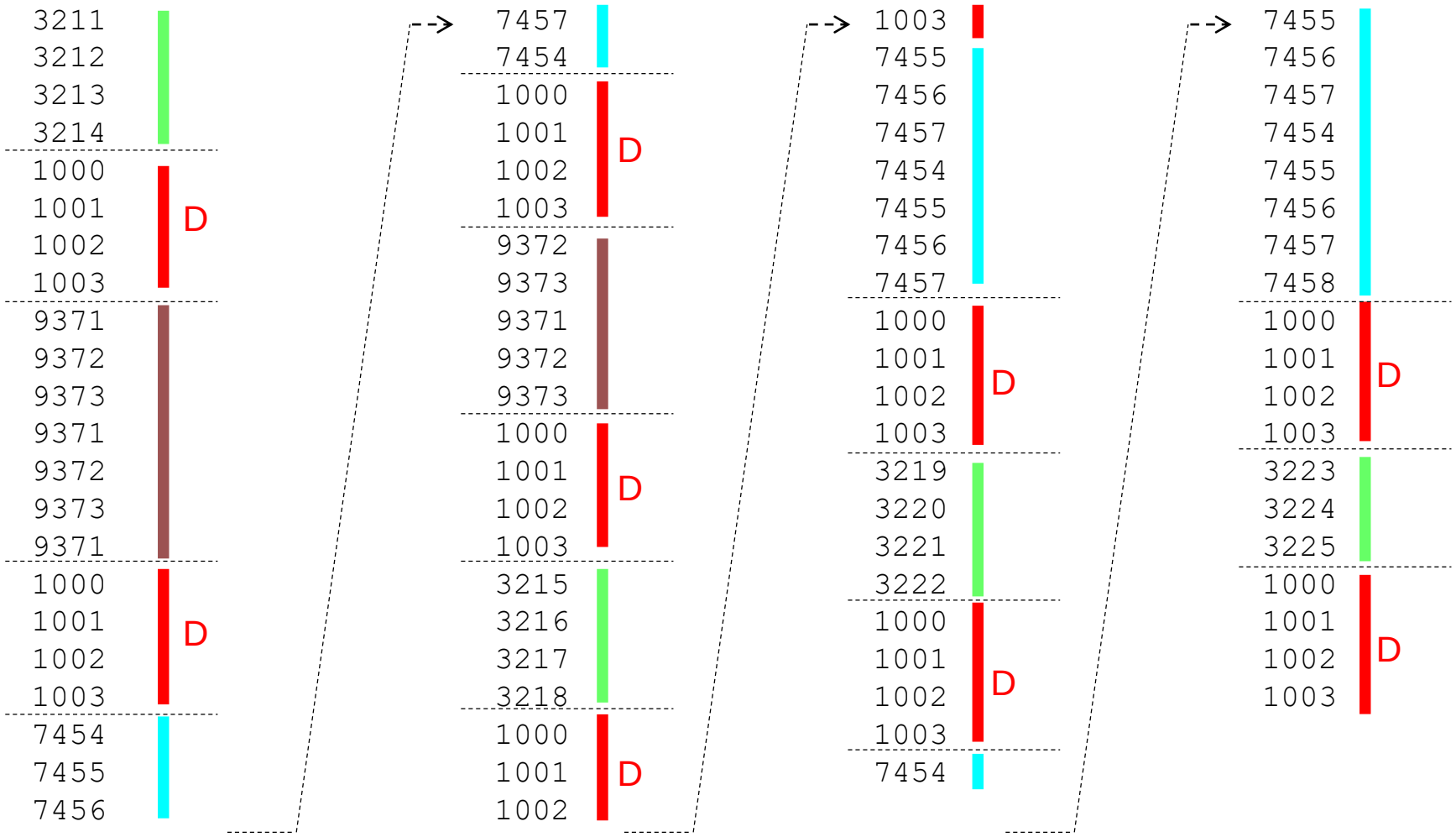
Prozesswechsel etwas detaillierter

1. Prozess 1 sei aktiv
2. Unterbrechung durchs Betriebssystem („System-Interrupt“); ab jetzt wird spezieller Betriebssystemcode ausgeführt: „Dispatcher“
3. Speichere aktuellen Programmzähler von Prozess 1 in Hauptspeicher
4. Speichere alle Datenregister, inklusive Prozessstatuswort von Prozess 1 in den Hauptspeicher
5. Lese Registerinhalte von Prozess 2 aus dem Hauptspeicher in den Prozessor
6. Lade Programmzähler von Prozess 2 in Prozessor
7. Prozessor führt Prozess 2 an der richtigen Stelle fort (implizit wird Ausführung des Betriebssystemcodes beendet)

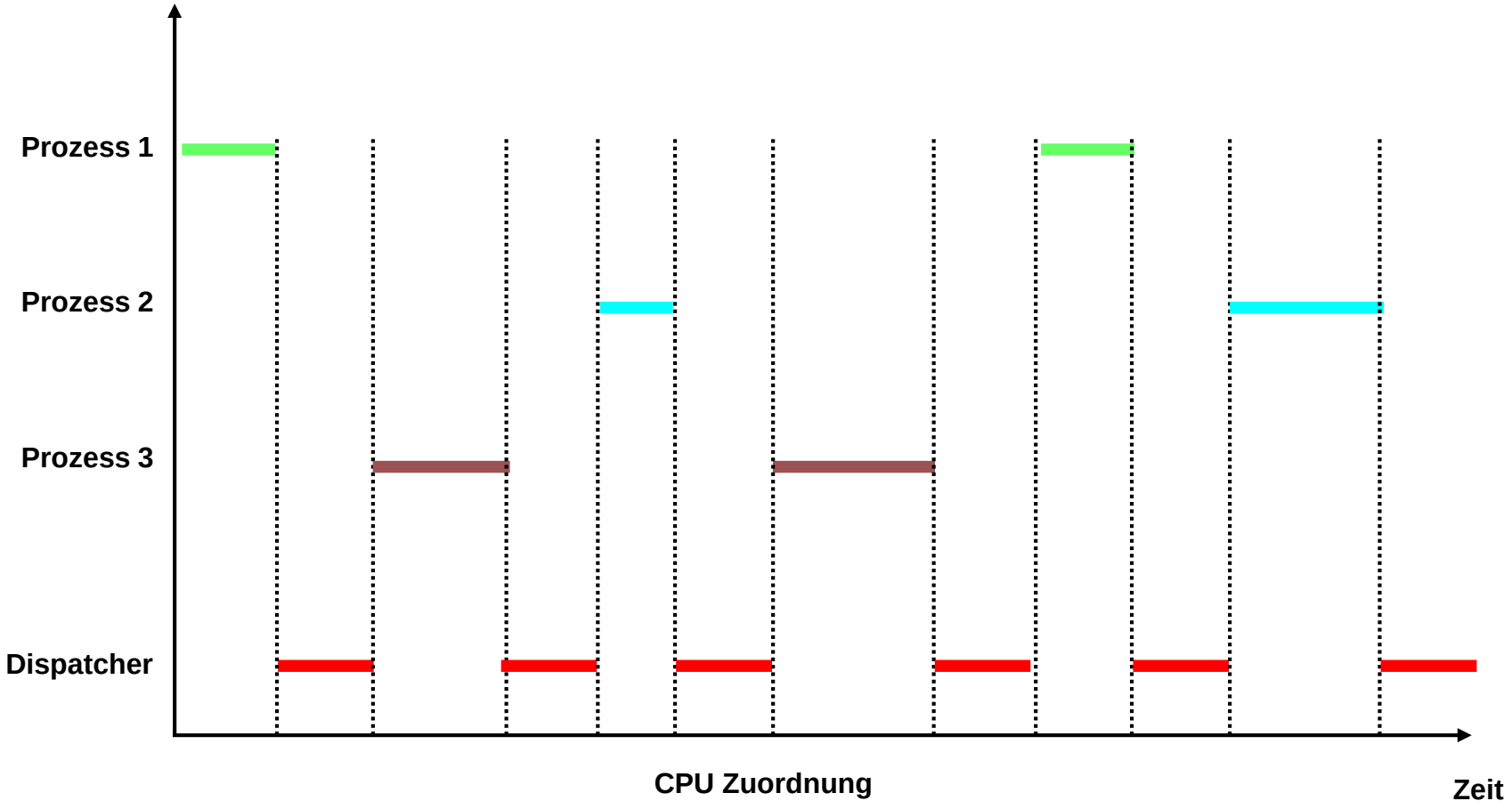
Beispiel mit 3 Prozessen: Programmzählerabfolge

Adresse	Prozess1	Prozess2	Prozess3
1000	3211	7454	9371
	3212	7455	9372
	3213	7456	9373
	3214	7457	9371
3200	3215	7454	9372
	3216	7455	9373
	3217	7456	9371
	3218	7457	9372
	3219	7454	9373
	3220	7455	9371
7450	3221	7456	9372
	3222	7457	9373
	3223	7454	
	3224	7455	
	3225	7456	
9370		7457	
		7454	
		7455	
		7456	
		7457	
		7458	

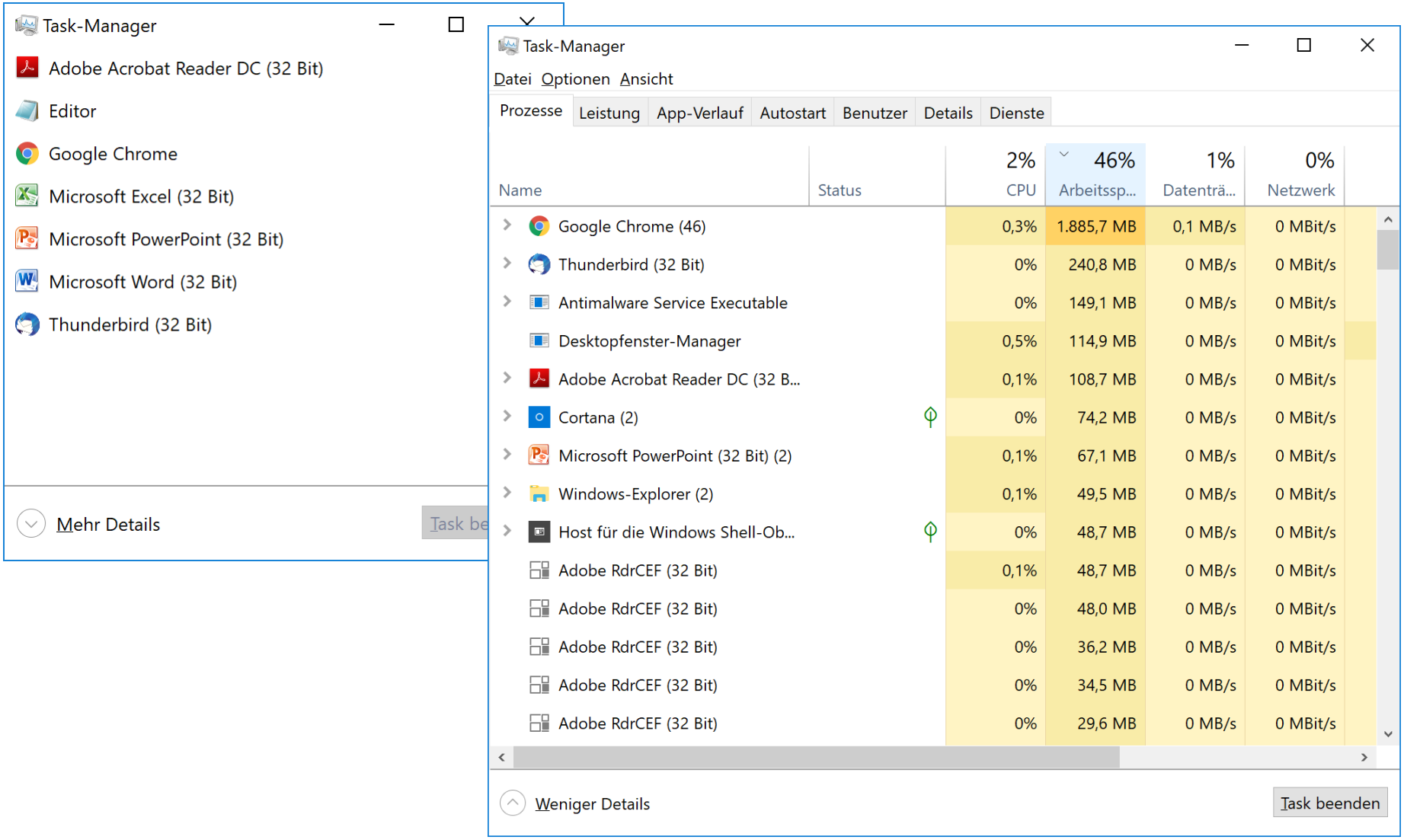
Beispiel Prozesswechsel



Beispiel Prozesswechsel (Zeitdiagramm)

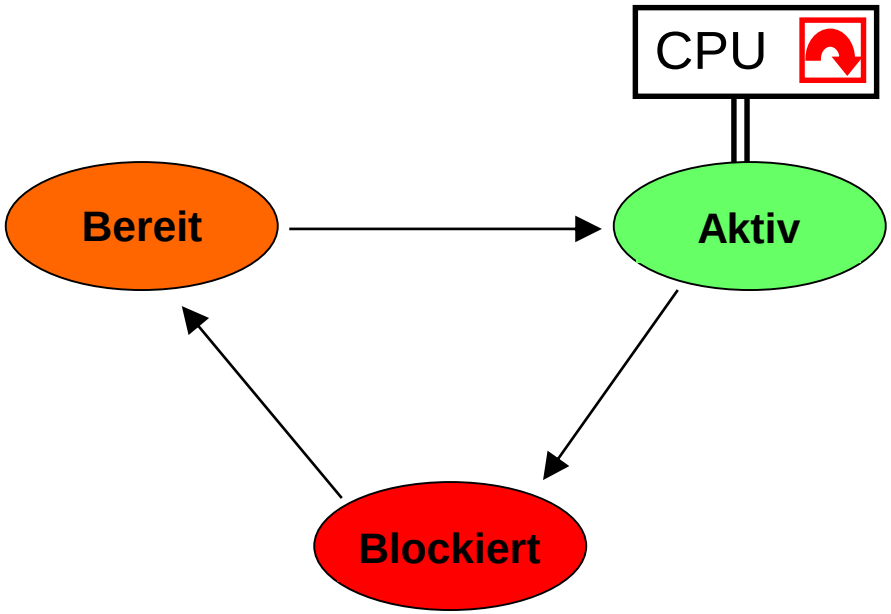


Beispiel Windows 10 Task Manager



Prozesszustände - 1

- jeder Prozess wechselt zwischen 3 Hauptzuständen:



Frage: Kann es auch mehrere Prozesse im Zustand aktiv geben?

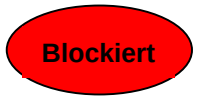
- „Ampel“-Schaltung

Prozesszustände – 2

- Prozesszustand

  - Prozess ist der CPU zugeordnet
  - Befehle des Prozesses werden abgearbeitet
  - Prozess benutzt CPU, Speicher, etc.

- Prozesszustand



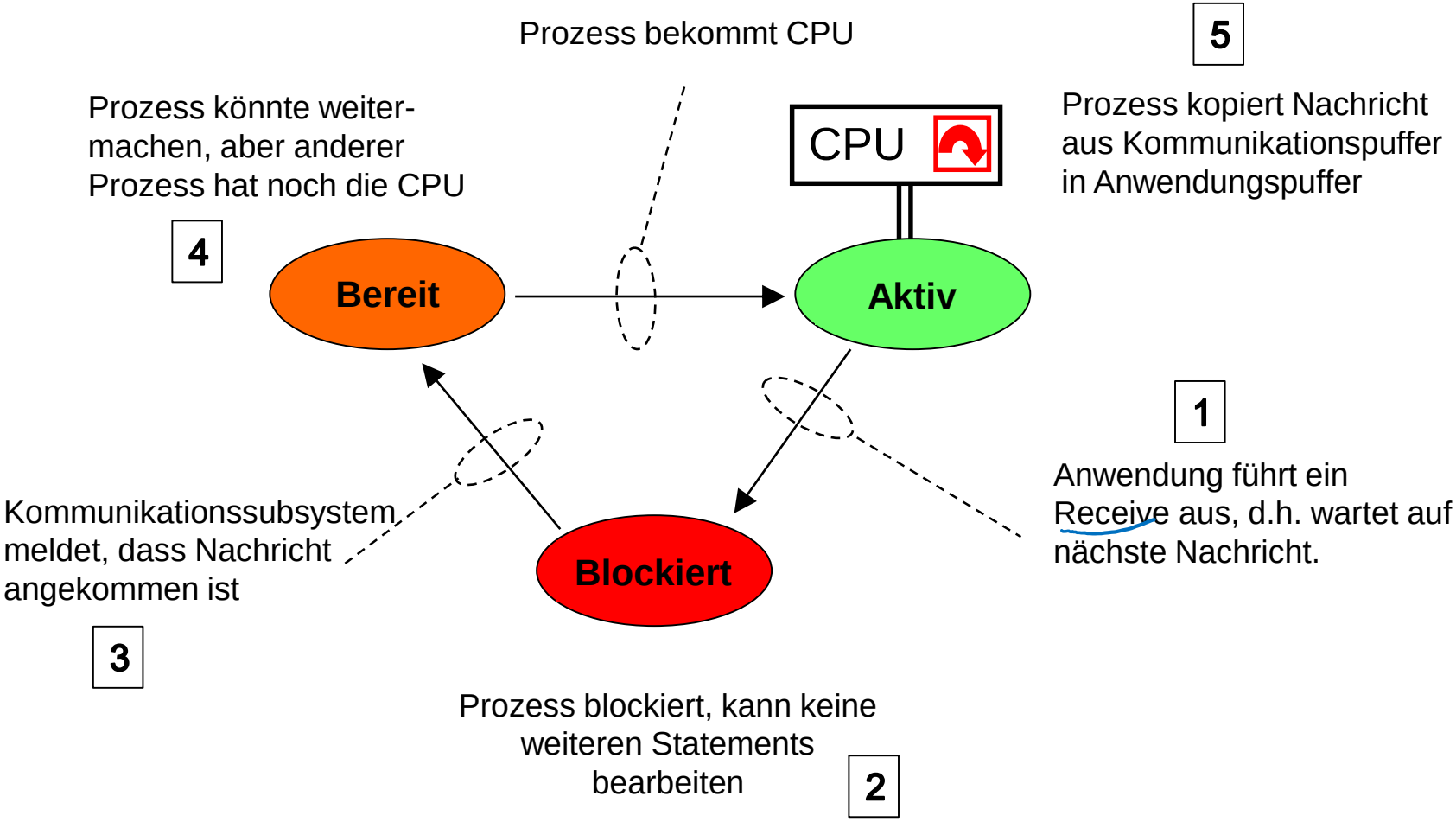
  - CPU ist gerade von anderem Prozess belegt
  - Prozess wartet auf externes Ereignis, ohne welches er nicht weitermachen kann, z.B. auf
    - Beendigung eines Empfangsvorgangs von einem Netzwerk
    - Beendigung eines Schreibvorgangs auf Festplatte

- Prozesszustand

  - CPU ist gerade von anderem Prozess belegt
  - Prozess könnte im Prinzip weitermachen (Kein externes Ereignis, auf das er noch warten müsste)
  - Prozess wartet auf CPU

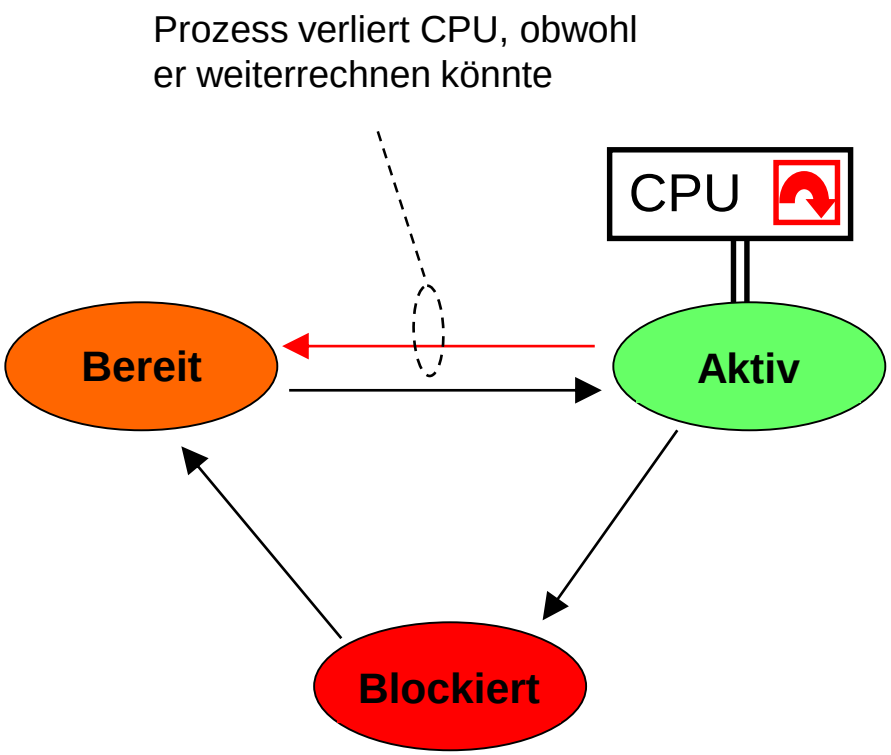
Prozesszustände - 3

- Beispiel für Zustandswechsel: Kommunizierender Prozess



Prozesszustände - 4

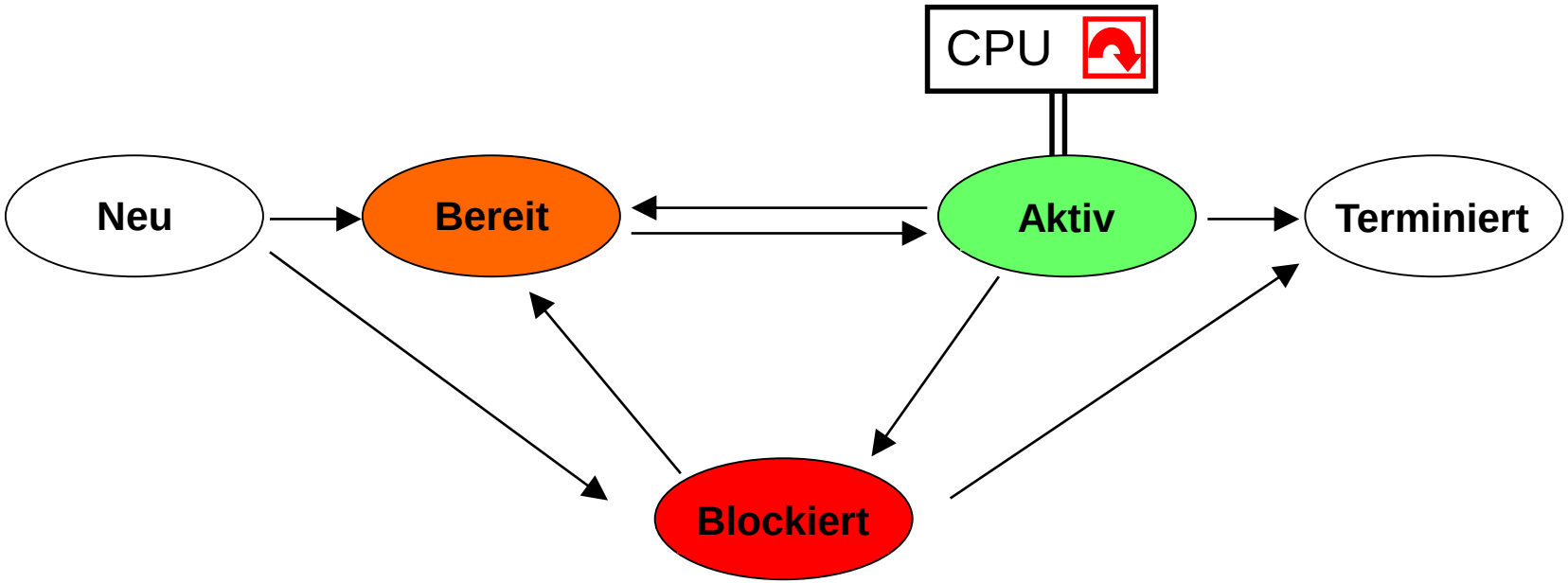
- Noch ein Zustandswechsel: „Verdrängung“ („Preemption“)



Frage: Warum ist es sinnvoll, Prozesse zu verdrängen?

Prozesszustände - 5

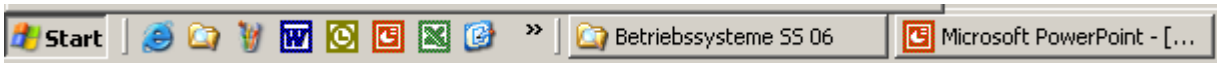
- Weitere (weniger wichtige) Prozesszustände:



- Prozesserzeugung: Neuer Prozess wird etabliert
  (mit Code, Daten, etc.)
- Prozessterminierung: Alle nicht-persistenten Zustandselemente
  des Prozesses werden gelöscht

Prozesserzeugung

- Bestimmte Systemprozesse werden bereits zum Boot-Zeitpunkt gestartet (Hintergrundprozesse, „daemons“)
  → Beispiel: Desktop-Prozess, Disk I/O, Spool-Server, etc.
- Prozesse können interaktiv durch User erzeugt werden:



Häufig: 1 Prozess = 1 Fenster (nicht immer!)
Interaktion (z.B. Click) = Kommunikation mit Prozess

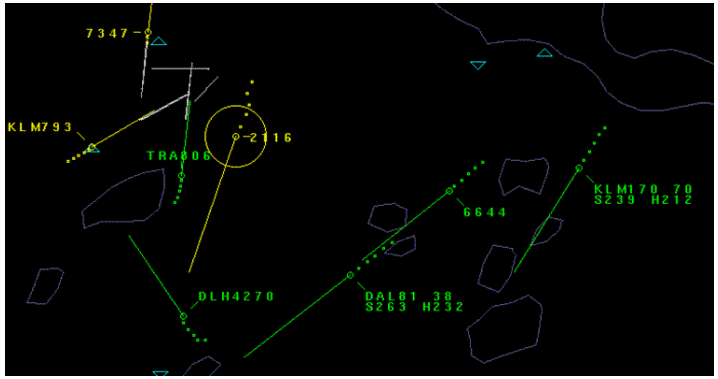
- Prozesse können durch andere Prozesse erzeugt werden
  - Eltern/Kind (oder: Vater/Sohn) Prozesse
  - UNIX: „fork“, Windows: „CreateProcess“
  - Aufbau von „Prozesshierarchien“, „Prozessfamilien“ (Windows: „Process tree“)
  - Gemeinsame Erledigung einer übergeordneten Aufgabe

„Prozessfamilien“ - Beispiel (Flugsicherung)

EUROCONTROL Tracking System „ARTAS“ (ATM SuRveillance Tracker and Server)*

Radaranlagen

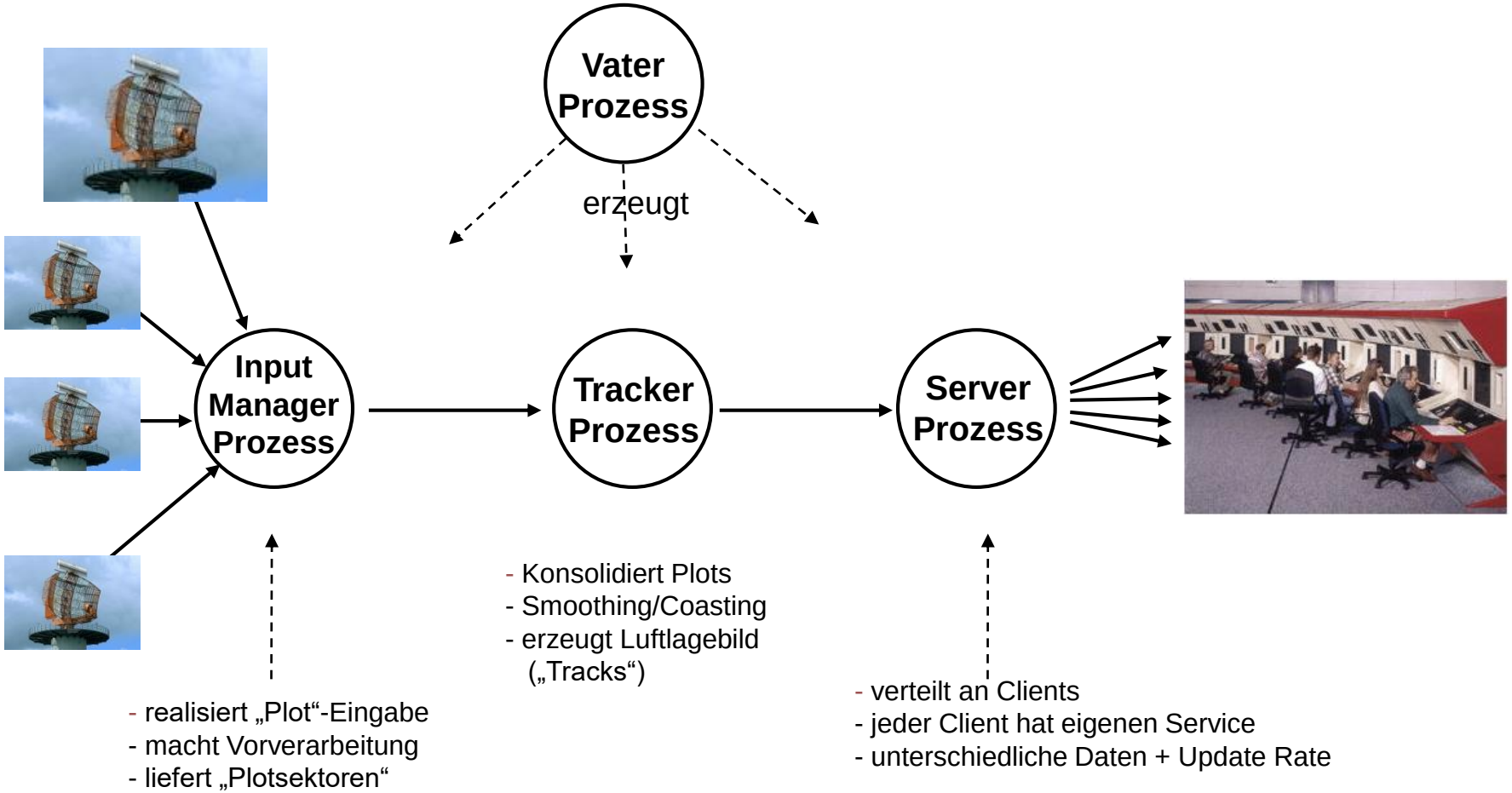
Flugbahn-verfolgungs-system



* im Einsatz in ca. 20 europäischen Ländern

Prozessfamilien - Beispiel Flugsicherung

- ARTAS Inside: Pipeline von Prozessen (Producer-Consumer)



Prozess Pipelining

- Jeder Prozess ist für eine eigene Teilaufgabe zuständig
  → Modularisierung, „Divide & Conquer“
- Klare Schnittstellen sorgen für bessere Testbarkeit, Fehlerabgrenzung
- Erhöhte Effizienz: Während einzelne Prozesse auf E/A warten, können andere Prozesse „den Ball weitergeben“
- Bessere Echtzeiteigenschaften: Unterschiedliche Prozesse können jeweils eigene Zeitbedingungen erfüllen

Betriebssystemunterstützung für Pipelining:

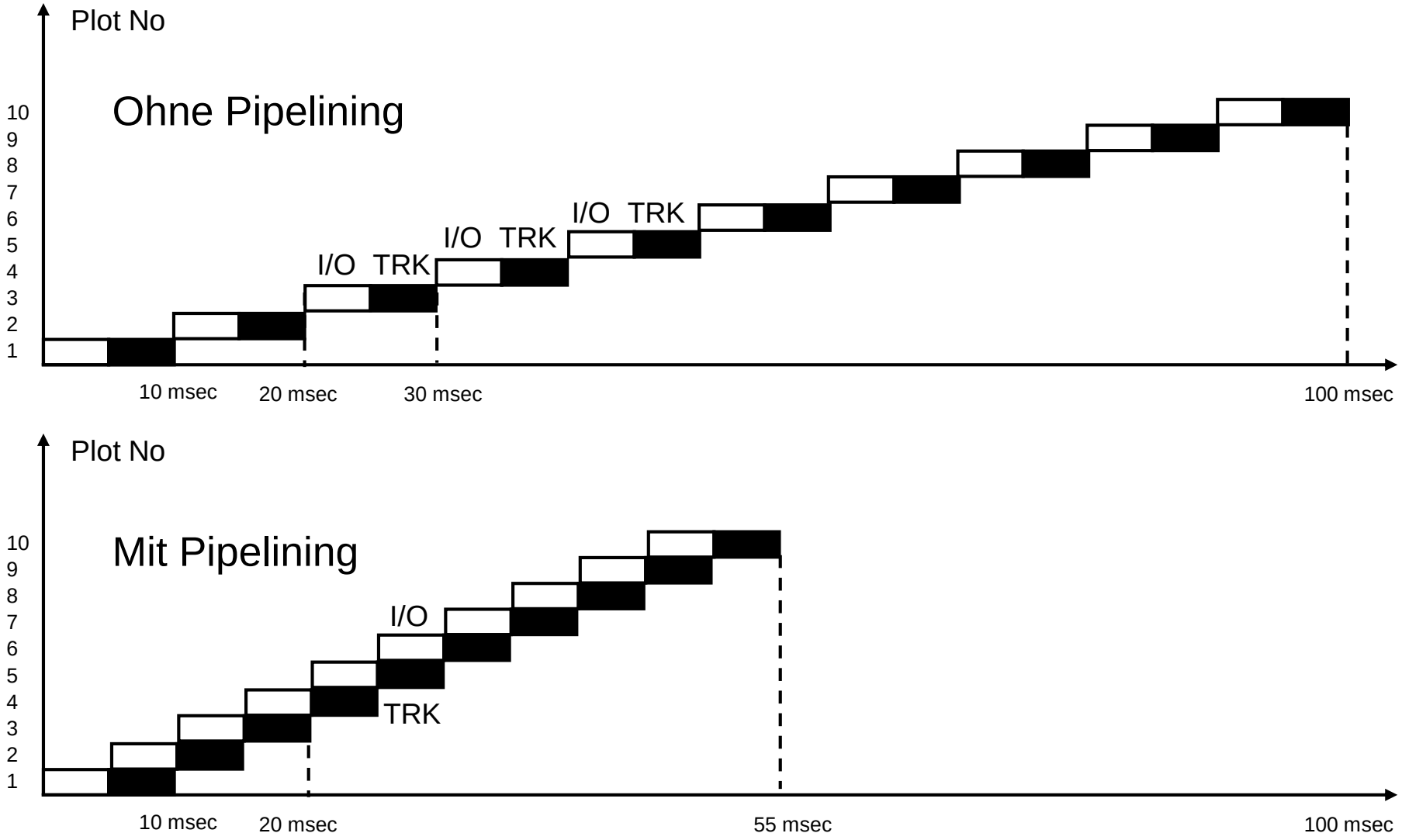
1. Erzeugung mehrerer Prozesse (Prozessfamilie)
2. Zeitgleiche Ausführung der Prozesse
3. Kommunikation von Prozessen (später!)

Beispielrechnung zum Prozess-Pipelining

- Input Manager Prozess ist über einen ISDN-Kanal (64 kbps) mit einem Radar verbunden.
- Jeder Plot (Radar-Zielmeldung) ist ca. 40 Byte lang
- Der Empfang der Daten erfolgt über eine E/A-Karte;
  Annahme: der Input-Manager stößt die E/A an, wartet auf eine neue Nachricht, dekodiert sie (kurze Aktion) und gibt sie an den Tracker weiter
- Der Tracker berechnet aus einer Zielmeldung innerhalb von 5 ms ein neues Luftlagebild (real geht‘s schneller..)

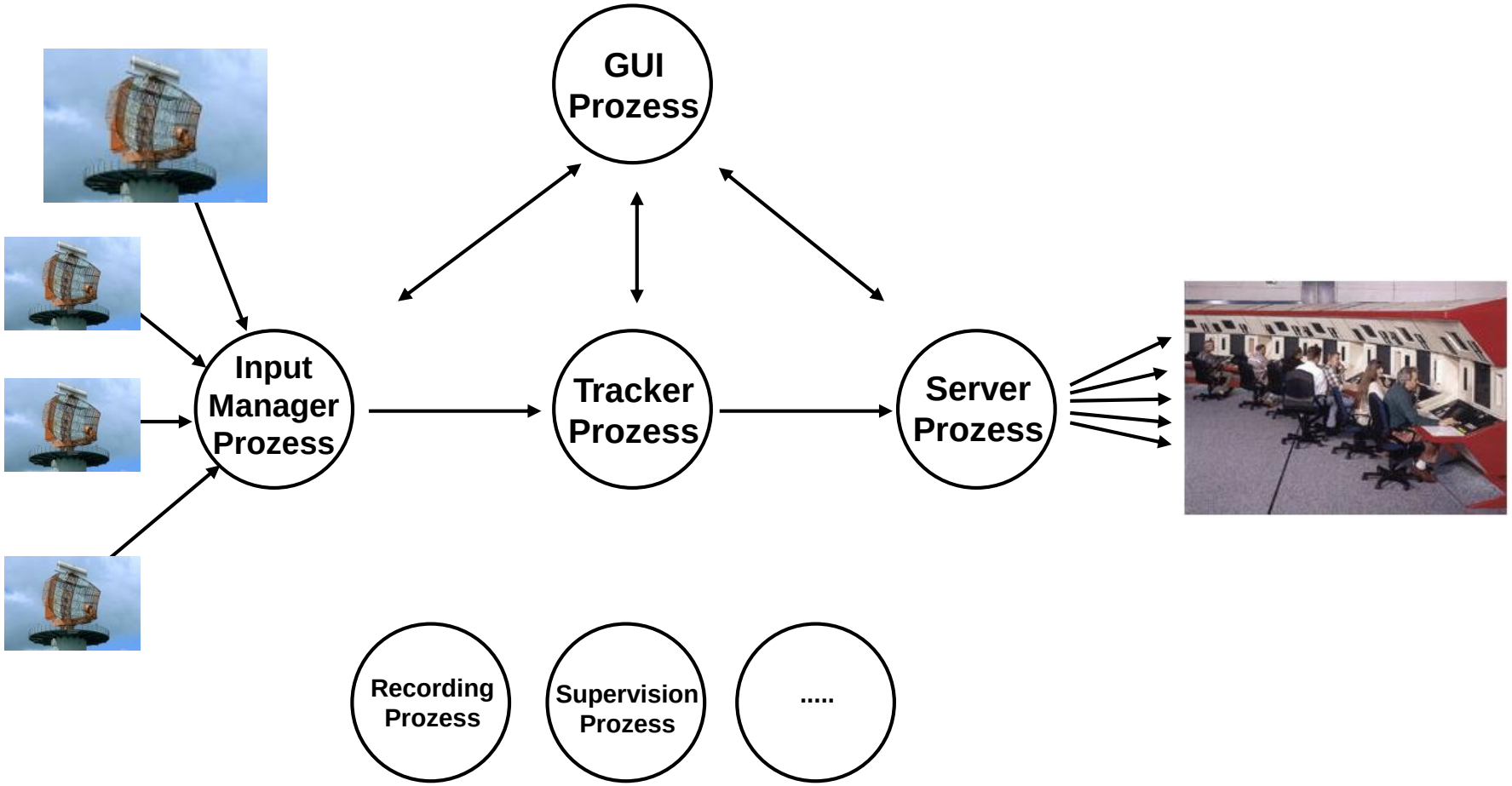
Wie groß ist der mögliche Durchsatz (Plots/sec) mit nur einem Prozess, bzw. mit dem dargestellten Prozess-Pipelining?

Beispielrechnung zum Prozesspipelining



Prozesshierarchien - Beispiel Flugsicherung

- **Das Bild vervollständigt:**



Prozessterminierung

- Reguläre Terminierung, nach Ende des Programms
  - Unix: „exit“, Windows: „ExitProcess“
  - Freigabe aller Betriebsmittel
  - auch im Fall von „kontrollierten Fehlern“ (z.B. Parameterfehler, die das Programm selbst erkennt)

- Fehler im Programm
  - Beispiele: DIV/ZERO, Float-Überlauf, zu großer Array-Index, falscher Pointer, etc.
  - Einfache Fehler kann Programm selbst behandeln (→ „Exception Handler“)
  - Schwerwiegende Fehler übernimmt das Betriebssystem und terminiert den Prozess

- Terminierung von außen
  - anderer Prozess (z.B. Vater) terminiert den Prozess
  - Unix: „Kill“, Windows: „TerminateProcess“