Inhaltsverzeichnis

1 Allgemeines

1.1 Definition: Embedded System

- Rechner ist in einem technischen Kontext integriert/eingebettet
- Wird eingesetzt um: Steuerung, Regelung, Überwachung eines technischen Prozesses umzusetzen
- nur für eine spezifische Aufgabe entwickelt (evtl. Echtzeitanforderung)
- Aspekte:
  - Resourcenverbrauch
  - Kombination von Hard und Software
  - Kosteneffizienz bei der Herstellung

1.2 Interaktion mit der Umwelt

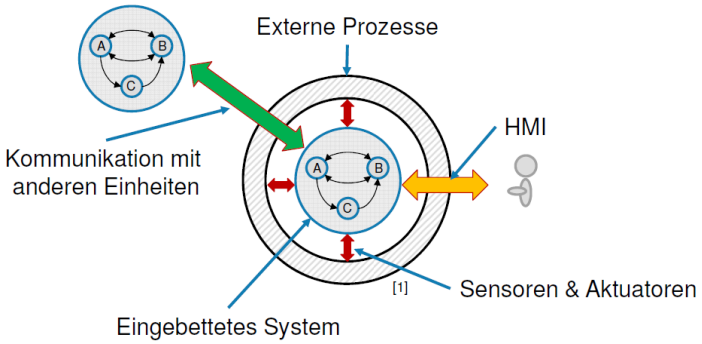

Abbildung 1: Eingebettetes System: Integration und Interaktion mit der Umwelt

1.3 Definition: Prozess

Gesamtheit von aufeinander einwirkenden Vorgängen in einem System, durch die Materie, Energie oder Information umgeformt, transportiert oder gespeichert wird

- Technischer Prozess
  Prozess in dem die Zustandsgrößen durch technische Hilfsmittel festgestellt und beeinflusst werden (Sensoren/Aktuatoren)
- Rechenprozess
  - Umformen, Transportieren, Speichern von Information
  - Berechnen von Ausgabewerten aus Eingabewerten
  - Beschrieben durch ein Programm
- Kognitiver Prozess

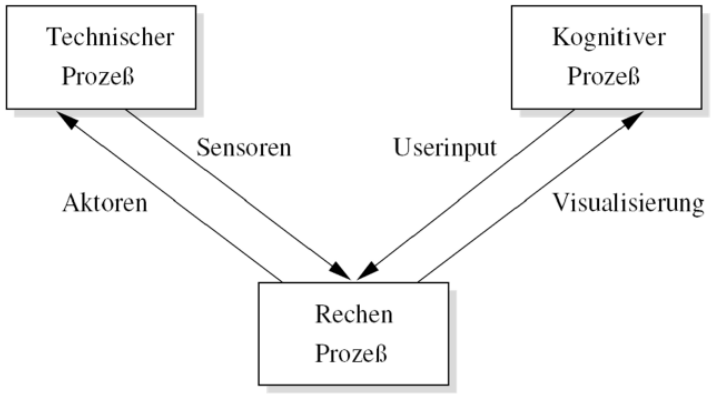

Abbildung 2: Zusammenwirken der Prozesse

1.4 Steuerung vs Regelung

Bei Regelung werden Sensorwerte verwendet für Steuerung.

1.5 Klassifizierung: Technischer Prozess

- Fließprozess (kontinuierlich/dynamisch) ⇒ Regler
- Folgeprozess (diskrete Informationselemente) ⇒ State Machine
- Stückprozess (Transport, Laden, Fertigen) ⇒ Datenbank

2 Host-Target-Entwicklung

Entwicklungsumbegung und Zielsystem sind unterschiedlich (Zielsystem nicht leistungsfähig genug)

- gegenteil ist Self-Hosted-Development
- Benötigt
  - Cross-Compiler
  - Remote-Debugger
  - Target-Libraries
  - Target-Betriebssoftware

3 Echtzeitbetrieb

3.1 Kriterien

- Pünktlichkeit/Rechtzeitigkeit (timeliness)
- Sicherheit (Safety) (Fail-safe/Fail-Operational; Automotive, Luft/Raumfahrt)
- Verfügbarkeit
- Deterministisch (in gegebenem Zustand liefert das System bei gegebener Eingabe immer dieselbe Ausgabe)

3.2 Klassifizierung: Echtzeitbedingung

- Hart
  muss rechtzeitig/pünktlich ausgeführt werden
- Weich
  darf Toleranz nicht überschreiten
- Fest
  sollte rechtzeitig/pünktlich ausgeführt werden, sonst nutzlos

3.3 Laufzeit-Modell

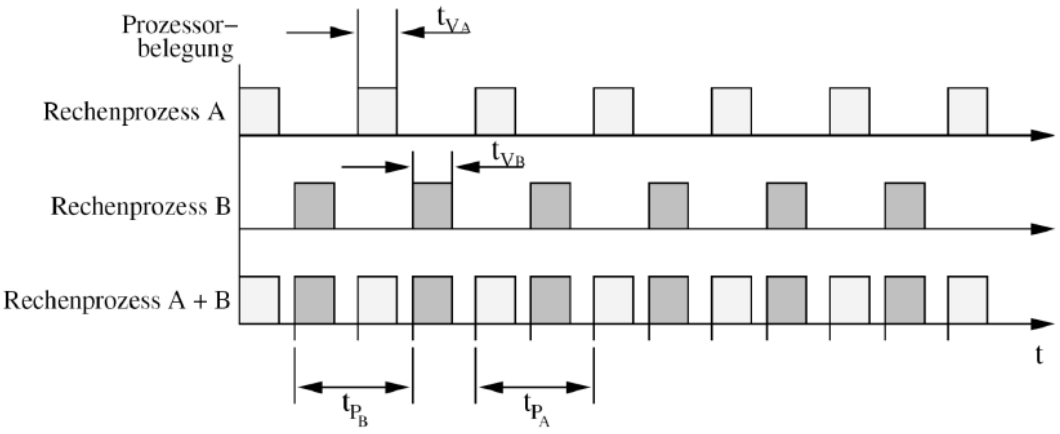

Abbildung 3: Prozessauslastung

- Verarbeitungszeit: t_v
- Prozesszeit: t_p
- Auslastung: $\rho = \frac{t_v}{t_p}$
- Gesamtauslastung: $\rho_{ges} = \sum_{i=0}^{n} \frac{t_{v_i}}{t_{p_i}}$ (muss ≤ 1 sein)

3.4 Pünklich vs Rechtzeitig

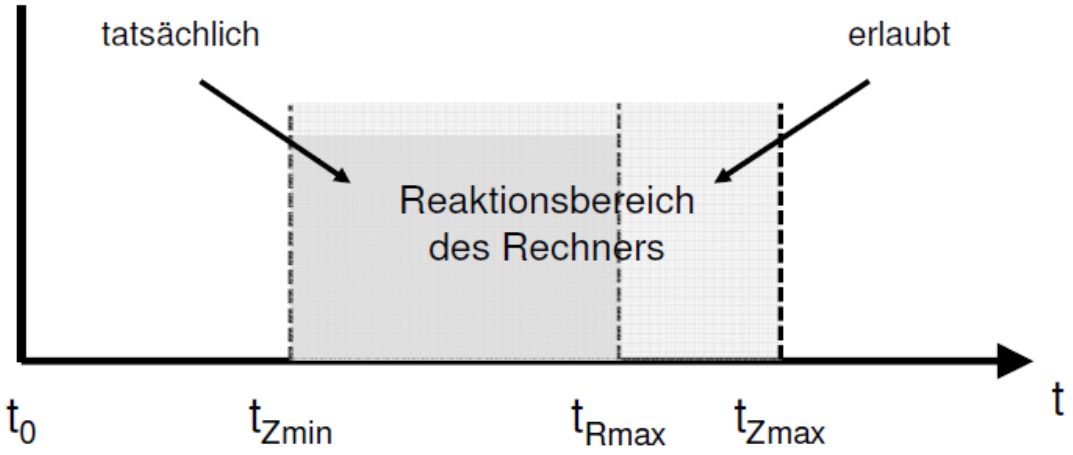

Abbildung 4: Echtzeitbedingung: Pünktlichkeit

- Wartezeit: t_w
- Reaktionszeit: $t_R = t_v + t_w$
- Zulässigkeit: $t_{Z_{min}}$, $t_{Z_{max}}$
  - Pünktlich: $t_{Z_{min}} \leq t_{R_{min}} \leq t_R \leq t_{R_{max}} \leq t_{Z_{max}}$
  - Rechtzeitig: $t_R \leq t_{R_{max}} \leq t_{Z_{max}}$

4 Echtzeitbetriebssystem

4.1 Aufgaben

- Ausführung der Benutzerprogramme
- Verteilung der Betriebsmittel
- evtl. Steuerung, Überwachung
- Standartisierter Zugriff (virtuelle Maschine)

4.2 Anforderungen: Prozessmanagement

- Zeitverhalten
  - Schnelligkeit
  - Bei einem RTOS insbesondere die Realisierung kurzer Antwortzeiten
  - Zeitlicher Determinismus (z.B. Speicherverwaltung/Garbage Collection ist
- Geringer Ressourcenverbrauch
  - Hauptspeicher
  - Prozessorzeit
- Zuverlässigkeit und Stabilität
  - Programmfehler dürfen Betriebssystem und andere Programme nicht beeinflussen
- Sicherheit
  - Dateischutz, Zugangsschutz
- Portabilität, Flexibilität und Kompatibilität
  - Erweiterbarkeit von Systemen
  - Einhalten von Standards (z.B. POSIX)
  - Möglichkeit, für andere BS geschriebene Programme zu portieren
- Skalierbarkeit
  - Hinzunehmen oder Weglassen von BS-Komponenten ermöglichen
  - Geringer (Programm-, Daten-)Speicherbedarf („Footprint“)
  - Komfort und umfassende Funktionalität bei großen Anwendungen

4.3 Determinismus

- Scheduling
- IPC und Synchronisation
- Speichermanagement: kein Swapping oder Garbage Colleciton

4.4 Unterbrechbarkeit

- Software Interrupt (Betriebssystem-Dienst Anfordern: Systemcall)
- Hardware Interrupt (Hardware fordert Dienste)
- ⇒ Präemtives Scheduling ⇒ TCB

4.4.1 Task Controll Block

- Priorität
- Maschinenzustand (Register, Stack, ...)
- Task-Zustand (+Bedingungen, auf die die Task wartet)
- Zeit-Quantum („erzeugte Prozessorlast"; Summe, in letzter Zeiteinheit)
- Verwaltungsdaten für Betriebsmittel (Filedeskriptoren,...)
- Speicherabbildungstabellen virtueller Speicher (Prozessadressraum) -> realer Speicher (code, data, stack)

4.4.2 Lightweight Processes

Minimierung des Kontextwechsels (nur Stack und PC werden separiert)

4.5 Speicherverwaltung

4.5.1 MMU

- Speicherschutz
- Adressumsetzung ⇒ Shared Libraries
- mit Hardware realisiert

4.5.2 Keine MMU

- Alle Programme zur Link-Zeit bekannt ⇒ zuordnen von Adressbereichen und Sprungadressen
- Loader: Ersetzt Adressen zur Lade-Zeit eines Programms
- Position Independent Code

4.6 IO-System

(**Unix**: Geräte werden im Dateisystem abgebildet)
⇒ Zugriffswunsch bei OS anmelden → Zugriffsrechte → Filedescriptor

4.6.1 Kernelmode

- Einheitlicher zugriff
- Kapselung
- Sicherheit

4.6.2 Systemcall

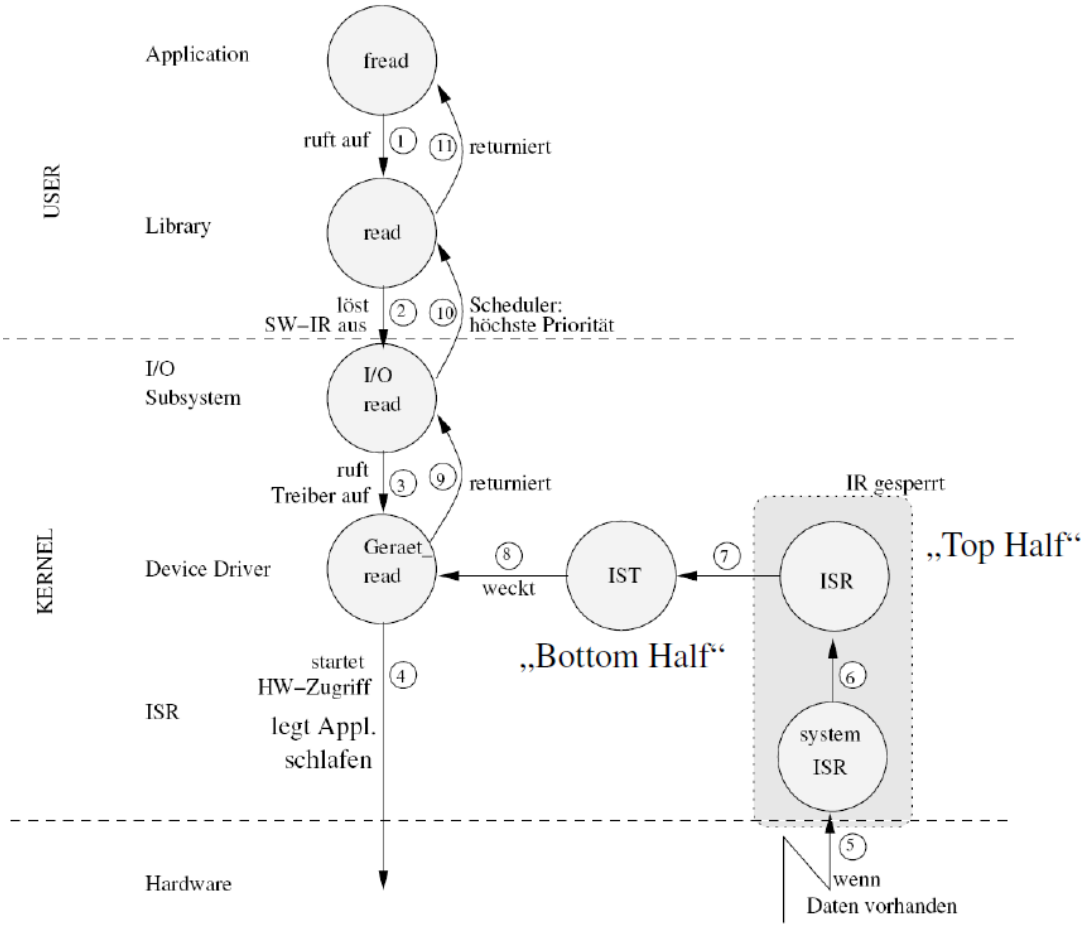

Abbildung 5: Ablauf: Systemcall

4.6.3 Latenzzeiten

4.6.4 Dateizugriff

- Synchronous IO (Implizites Warten)
  Unblock wenn:
  - Antwort des Geräts
  - Fehlerzustand

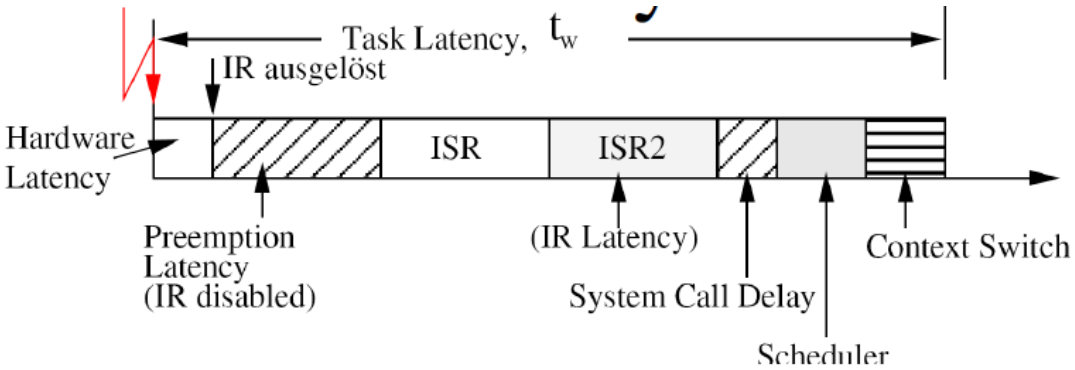

Abbildung 6: IO Latenzzeiten

  - Signal an Task

- Asynchronous IO (Explitizes Warten)
  kein blocking, sondern:
  - callback
  - polling
  - Signal an Task
- Asynchronous: Probleme
  - Parallelität von Task aus garantieren
  - Applikationsende ⇒ OS-Verwaltungsaufwand

5 Zeitdienste

5.1 Absolutzeit

- Beispiele
  - Datum, Uhrzeit
  - Zeitstempel
- Anwendung
  - Zuordnung von Zeitstempel zu Messergebnis
  - Geschwindigkeit/Durchsatz durch Differenzzeitmessung

5.2 Relativzeit

- Beispiele
  - Verzögerung/Warten von Zeitabständen
  - zyklische Ausführung
  - Watchdog (z.B. Totmanschalter)
- Anwendung
  - Systemzeit in Ticks
  - Bus-Controller
  - Vorwerts/Rückwerts-Zähler
  - single shot/ repetitiv

6 Scheduling

6.1 Statisch/Dynamisch

- Statisch
  - "Fahrplan" im vorhinein festlegen
  - Einsatz: Zyklische, Sicherheitskritische Anwendungen min fixen Zeitpunkten
  - Realzeitnachweis möglich
- Dynamisch
  - Reihenfolge wird zur Laufzeit entschieden
  - Rechenprozesse müssen präemptiv sein

6.2 Bewertungskriterien

- Gerechtigkeit
- Effizienz (möglichst gute Auslastung)
- Durchlaufzeit (Prozesse so schnell wie möglich abgeschlossen)
- Durchsatz (so viele Jobs wie möglich)
- Antwortzeit (Reaktion)
- Determinismus (Berechenbarkeit)

6.2.1 Wichtig bei Echtzeit-BS

- Antwortzeit
- Determinismus

6.3 Scheduling Strategien

- FIFO/FCFS (Warteschlange, Prozesse laufen bis zu Ende oder zur Blockierung)
- Round Robin (wie FIFO bloß mit Timeslicing)
  - Prioritätsgesteuert
  - Deadline-Scheduling (EDF oder LLF)
  - Sporadic-Scheduling

6.4 Deadline-Scheduling

- EDF: earliest deadline first
- LLF: Least Laxity First

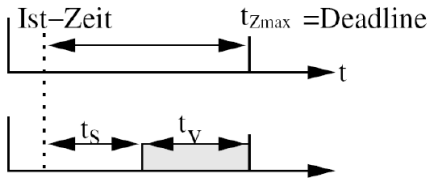

Abbildung 7: Deadline-Scheduling

6.5 Prioritätsgesteuert

- Für Realzeitsysteme geeignet
- Je kürzer die Prozesszeit t_P, desto höher die Priorität
- I/O bekommt höhere Priorität

6.6 Sporadic Scheduling

- Auslastung $< 100\%$: Task erhält zusätzliche Rechenzeit
- Auslastung $= 100\%$: Task verletzt Rechtzeitigkeitsbedingung
- Replenishment Period T (Prozesszeit)
- Initial budget C (Verarbeitungszeit pro Takt)
- Priority N
- Low Priority L
- Maximal number of pending replenishments

7 Echtzeitnachweis

7.1 Methoden (skizziert)

- relevante Kenndaten des technischen Prozesses
  - Anzahl der unterschiedlichen Anforderungen
  - Minimale Prozesszeit für jede Anforderung
  - Minimal zulässige Reaktionszeit $t_{Z_{min}}$ für jede Anforderung
  - Maximal zulässige Reaktionszeit $t_{Z_{max}}$ für jede Anforderung
  - Abhängigkeiten zwischen den Ereignissen
- maximale Verarbeitungszeit $t_{V_{max}}$ (WCET) für jede Anforderung identifizieren
- Auslastungsbedingung überprüfen
- Rechtzeitigkeitsbedingung verifizieren (Bestimmung von $t_{R_{min}}$ und $t_{R_{max}}$)

7.2 WCET

- Prozesszeiten möglichst kurz
- Verarbeitungszeiten möglichst lang
- alle sonstigen Ereignisse im System, die im aktuellen Prozesszustand möglich sind, treten gleichzeitig (t = 0) auf

7.2.1 Messen

+ Sprachunabhängig
+ meist einfach realisierbar

- Aussagekraft der Messungen abhängig von vielen Randbedingungen (Prozesszustand, Cache, Schleifen, Verzweigungen)
- Theoretisch sämtliche Kombinationen aus Inputdaten erforderlich (Test-Überdeckung)
- Produktiver Code auf Zielplattform oder Simulator erforderlich
- Messumgebung/Testrahmen erforderlich
- Instrumentierung modifiziert den Code

7.2.2 Analyse

Aus Quell- oder Zielcode wird ein Strukturgraph des Codestücks erstellt Mit Beschreibung der Zielhardware wird der längste Pfad durch den Graphen gesucht

7.3 BCET

Messung wie bei WCET, aber min geringster Systemlast und kürzestem Pfad

7.4 maximale Reaktionszeit

Vorraussetzungen

- prioritätengesteuertes Scheduling (Rate Monotonic)
- Alle Ergebnisse sind Unabhängig voneinander

7.4.1 Grafisches Verfahren

- Schnittpunkt mit X-Achse entspricht Reaktionszeit des niedrigpriorsten Prozesses
- Für den nächstniedrigprioren Prozess: X-Achse um t_V des vorhergehenden Prozesses nach oben verschieben

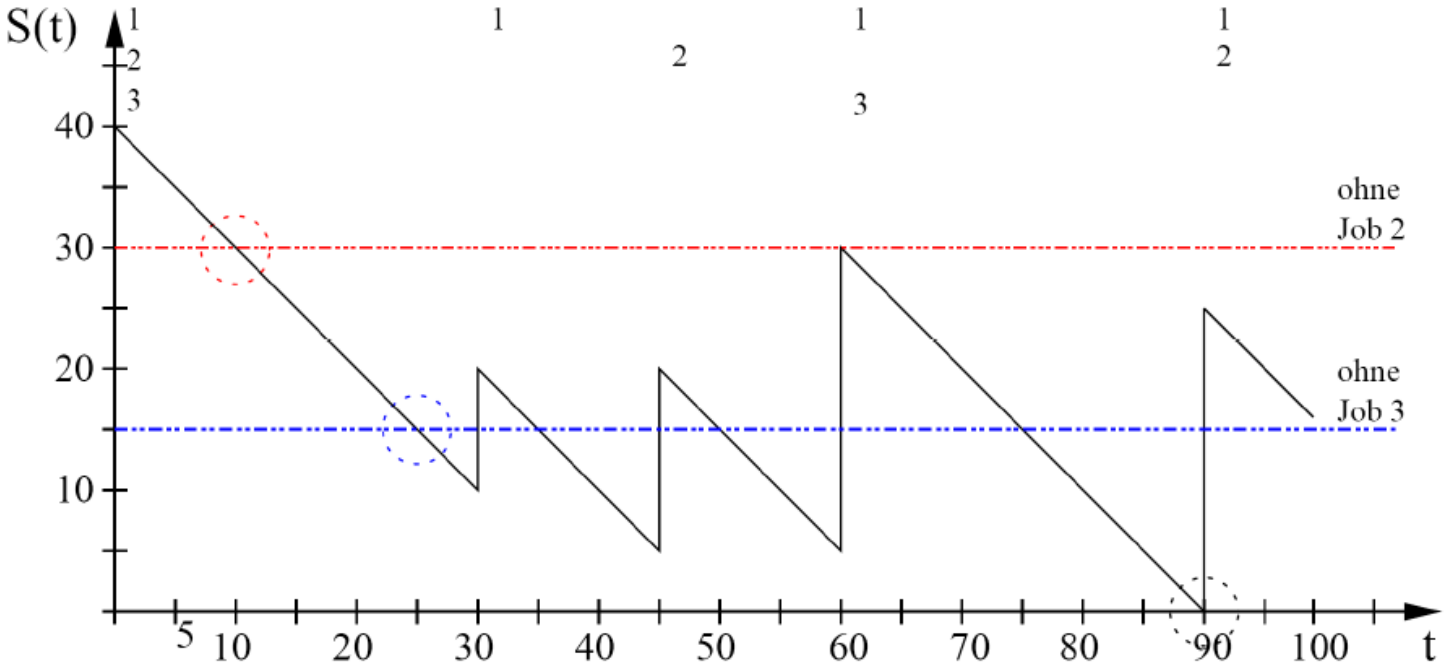

Abbildung 8: Grafisches Verfahren

7.5 Schedulingbedingung

Vorraussetzungen

- Scheduling mit statischen Prioritäten (Rate Monotonic...)
- Zyklische Tasks ohne Abhängigkeiten

$$\mu = \sum_{k=1}^{i} \frac{t_{V_k}}{min(t_{Z_{max}}, t_{P_k})} \leq i \cdot (2^{\frac{1}{i}} - 1)$$

i	μ in %
1	100
2	82.8
3	78
4	75.7
5	74.3
10	71.8
$i \to \infty$	69.3