Modul 63211 – Verteilte Systeme
Einsendeaufgaben 2 – WS 2025/2026
FernUniversität Hagen, Lehrgebiet Kooperative Systeme

Aufgabe 2.1: RPC **5 + 5 + 5 Punkte**

Ein Client führt *RPCs* auf einem Einprozessorsystem aus, angesprochene Server verfügen aber über genügend Prozessorkapazitäten. Setzen wir nun folgende Zeitanforderungen voraus, um Gesamtzeiten zu berechnen:

Bei *jedem* RPC benötigt der Client zunächst 5 Millisekunden, um den Server zu lokalisieren, auf dem der Aufruf ausgeführt werden soll. *Marshaling* und *Unmarshaling* kosten jeweils 0,5 Millisekunden, sowohl auf dem Client als auch auf dem Server. Der Server braucht 10 Millisekunden für die Bearbeitung einer Anfrage. Die lokalen Betriebssysteme beim Client und beim Server benötigen jeweils 0,5 Millisekunden Rechenzeit, um eine der Operationen `send` und `receive` auszuführen. Die *Übertragung* einer Nachricht zwischen den Client und Server dauert 3 Millisekunden. Andere Zeiten werden zur Vereinfachung ignoriert.

1. Wie viel Zeit benötigt die komplette Bearbeitung eines einzelnen RPCs?
2. Wie viel Zeit vergeht, wenn zwei RPCs vorliegen und der Client insgesamt nur einen Thread zur Bearbeitung verwendet?
3. Wie viel Zeit vergeht, wenn zwei RPCs vorliegen und der Client mehrere Threads dafür einsetzt? (Tipp: erst nach Aufruf von `send`() blockiert der erste Thread, und der zweite Thread fängt an zu arbeiten.)

Aufgabe 2.2: Message-Queuing System **6 + 5 Punkte**

1. Worum geht es bei der *persistenten* und *transienten* Kommunikation? Worum geht es bei der *synchronen* und *asynchronen* Kommunikation?
2. Um persistente und asynchrone Kommunikation zu implementieren, wird ein *message-queuing-System* benötigt. Welche Probleme muss ein message-queuing-System lösen?

Aufgabe 2.3: Publish/Subscribe **15 Punkte**

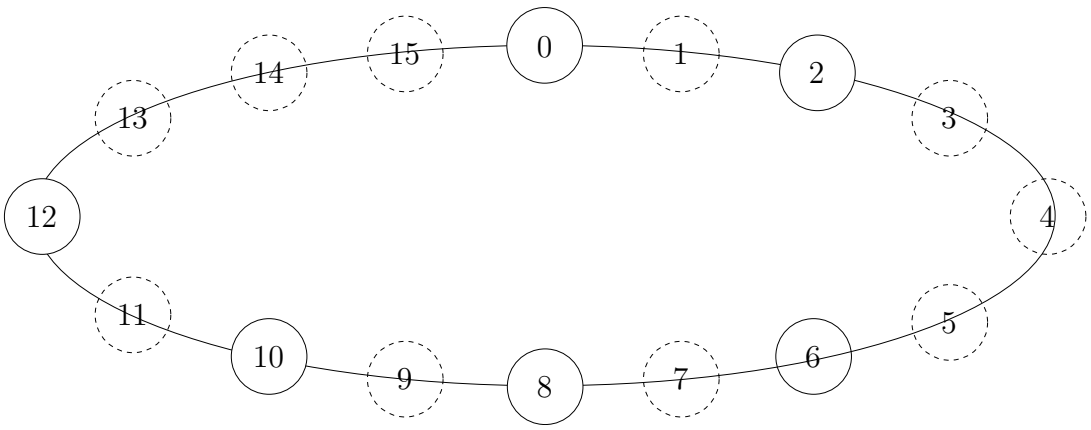
Betrachten Sie das Python-Programmbeispiel aus dem Buch in Fig. 4.22 auf Seite 216 (Distributed Systems, 4th edition, Version 4.02X, 2024) zum Thema Publish/Subscribe. Bringen Sie das Programm auf Ihrem Computer zum Laufen, der Subscriber (Client) soll einfach immer wieder nach der Zeit fragen.

Zum Testen starten Sie einen Publisher und mehrere Subscriber gleichzeitig.

Aufgabe 2.4: Peer-to-Peer-Systemen 6 + 12 + 3 Punkte

Betrachten Sie das *Chord-System* aus 16 Knoten in der Abbildung, wobei die Knoten mit den Bezeichnern $0, 2, 6, 8, 10$ und 12 aktiv sind und durch weiß gefüllte Kreise mit duchgezogenem Rand dargestellt werden.

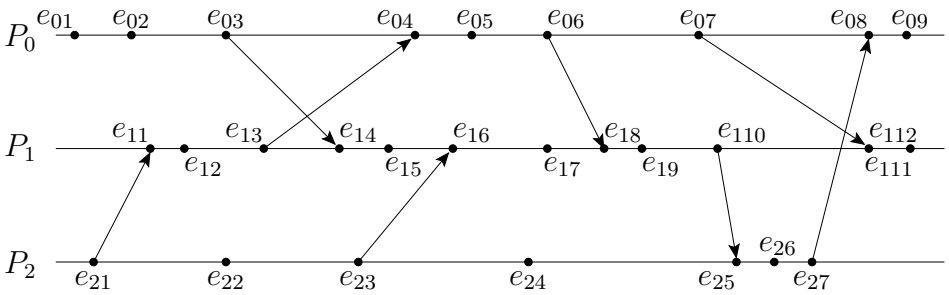
1. Welche *Nachfolger* (*successor*) haben die Schlüssel $0, 2, 4, 6, 11$ und 15?
2. Erstellen Sie die Finger-Tabellen, die für die Suche nach dem Nachfolger von Schlüssel $k = 9$ mit Startknoten 2 benötigt werden.
3. Welchen Pfad ausgehend von Knoten 2 nimmt die Auflösung von Schlüssel $k = 9$?



Aufgabe 2.5: Logische Uhren 19 + 19 Punkte

Betrachten wir das System von drei Prozessen P_0, P_1, und P_2 in der Abbildung unten, die jeweils über eine eigene Lamport-Uhr verfügen. Beim Prozess P_i haben sich die Events e_{ij} für $0 \leq i \leq 2$ und $1 \leq j \leq 12$ ereignet, wobei ein Pfeil von e_{ij} nach e_{kl} das Absenden einer Nachricht von Prozess P_i und das Empfangen beim Prozess P_k bedeutet. Nach jedem Empfangsereignis e_{kl} gibt es ein Auslieferungsereignis $e_{k(l+1)}$. Wir nehmen an, dass alle Uhren mit dem Wert 0 initialisiert sind.

1. Geben Sie jedem Ereignis einen Zeitstempel nach dem Algorithmus von Raynal und Singhal.
2. Geben Sie für jedes Ereignis einen Vektorzeitstempel an.



Ereignisse bei den Prozessen P_0, P_1, P_2.