Fragenkatalog Echtzeitsysteme

Sebastian Ederer & Michael Schneider – WS19/20

Was ist der Unterschied zwischen RTS und NRTS?

- Ein Real-Time-System ist ein System, das in der Werte- und Zeitdomäne korrekt ist.
- Ein Non-Real-Time-System ist nur in der Wertedomäne korrekt.

Erklären Sie den Begriff Accuracy und Precision. Geben Sie jeweils ein Beispiel an.

Accuracy

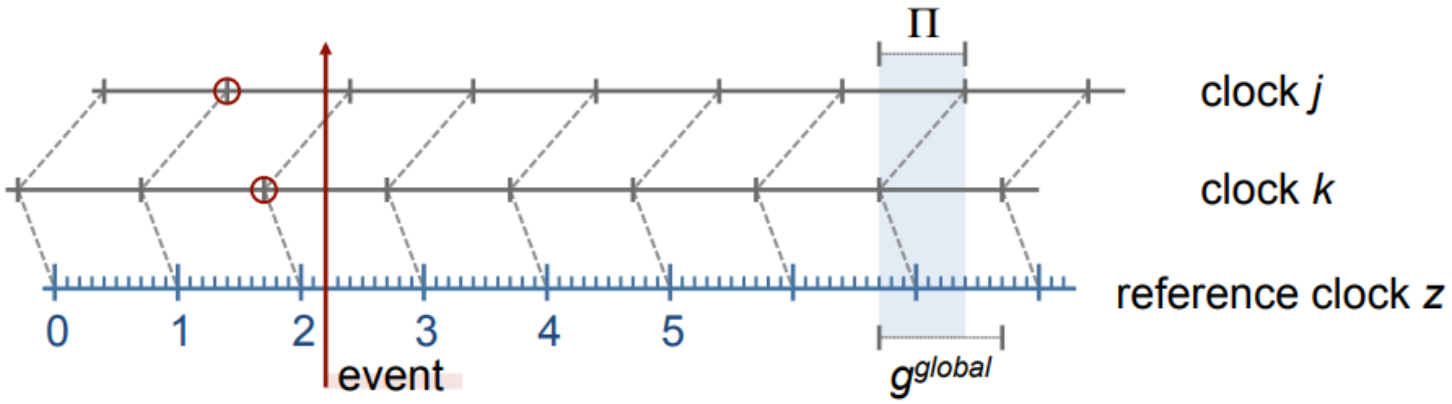
- Accuracy ist der Offset zwischen der Uhr k und einer Referenzuhr z beim Tick i.
- Beispiel: Eine Atomuhr, an welcher sich viel Uhren richten, ist 2 Sekunden vor meiner eigenen Uhr bei genau 12 Uhr. Damit ist der Offset der Accuracy zwischen meiner Uhr und der Atomuhr 2 Sekunden.

Precision

- Precision ist der Offset zwischen zwei Uhren j und k zum Tick i.
- Auch um 12 Uhr vergleiche ich meine Uhr mit der meines Kommilitonen. Dabei ergibt sich ein Unterschied von 5 Sekunden, bei welcher meine Uhr weiter vorne ist. Also ergibt sich eine Precision von 5 Sekunden.

Wie lautet die Reasonableness Condition für eine globale Zeitbasis? Erklären Sie diese.

- Die Globale Zeit t ist resonable, falls für alle lokalen Implementierungen gilt: $g^{global} > \Pi$
  - g^{global}: globale Granularität
  - Π: bekannte Precision



Wann ist eine globale Zeit reasonable? Warum ist das wichtig?

- Die Globale Zeit t ist resonable, wenn die Reasonableness Condition erfüllt ist.
- Das ist wichtig, da dadurch sichergestellt wird, dass
  - der Synchronisationsfehler kleiner als ein Macrogranule ist.
  - für jedes Event e gilt: $|t^j(e) - t^k(e)| \leq 1$

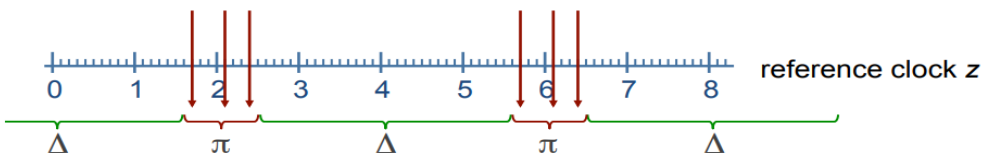
Wie lautet die Synchronisationsbedingung? Erklären Sie die Symbole.

Um die Uhren intern synchron mit einer Präzision von Π zu halten, muss die Synchronisationsbedingung folgendes enthalten: $\Phi + \Gamma \leq \Pi$

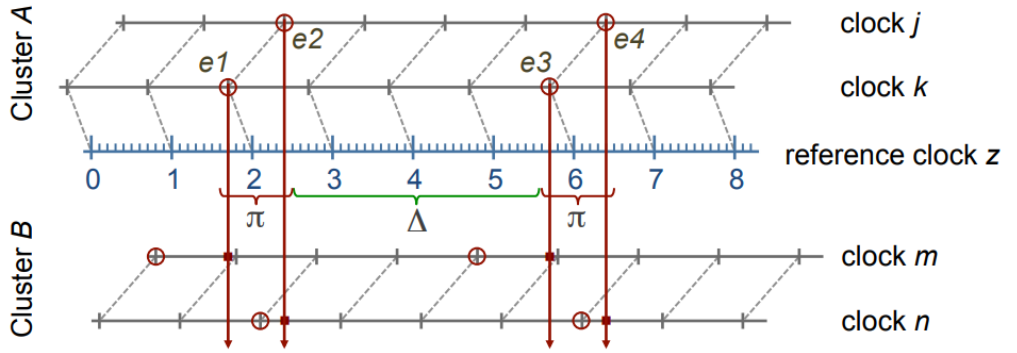
- Φ (Konvergenzfunktion):
  - Maximaler Offset nach der Synchronisation
  - Ist abhängig von dem Synchronisationsalgorithmus und dem Nachrichten-Latenz-Jitter ε – Die Differenz der Übertragungszeiten zwischen schnellster und langsamster Nachricht
  - $\varepsilon = d_{max} - d_{min}$
- Γ (Drift-Offset):
  - Divergenz von freilaufenden Uhren
  - $\Gamma = 2 \rho R_{int}$
  - R_{int} = Resynchronisationsintervall

Was versteht man unter einer Sparse Time Base? Wofür wird diese benötigt?

- Bei einer Sparse Time Base dürfen Events nur in dem Zeitintervall der Aktivität π auftreten, gefolgt von dem Intervall der Stille Δ



- Diese Zeitbasis wird für die Vereinbarung der Event-Order benötigt.
  - Angenommen es gibt 2 Rechencluster A und B
  - In jedem Cluster werden Uhren synchronisiert ($g = g^{global}$)
  - Keine Synchronisation zwischen A und B
  - Cluster A generiert Events, welche von B geordnet werden müssen:
    - B muss die Ordnung bzw. Gleichzeitigkeit von allen beobachteten Events bestimmen können.
  - Zeitbasis von A muss 1g/4g-sparse sein.
  - Eine 1g/3g-sparse Zeitbasis ist nicht ausreichend.



- e1, e2 werden im selben Aktivitätsintervall generiert: $t^n(e2) - t^m(e1) = 2$
- e2, e3 werden in verschiedenen Aktivitätsintervallen generiert: $t^m(e3) - t^n(e2) = 2$

Geben Sie die vier grundlegenden Grenzen der Zeitmessung in einem verteilten System mit globaler Zeitbasis der Granularität g an.

1) Wenn ein einzelnes Event von zwei Nodes beobachtet wird, darf der lokale Zeitstempel nur um einen Tick abweichen.
2) Duration Measurement: $d_{obs} - 2g^{global} < d^{z}_{true} < d_{obs} + 2g^{global}$
3) Die zeitliche Ordnung von zwei Events e_1 und e_2 kann von deren Zeitstempeln abgeleitet werden, wenn gilt: $|t^{j}(e_1) - t^{k}(e_2)| \geq 2$
4) Die zeitliche Ordnung von Events kann immer abgeleitet werden, falls das Event-Set $0/3g^{global}$-precedent ist.

Welche vier Anforderungen sind an eine globale Zeitbasis für eine verteiltes Echtzeitsystem zu stellen?

- Chronoskopisches Verhalten
- Bekannte Präzision Π
- Hohe Zuverlässigkeit
- Metrik einer physikalischen Sekunde

Was versteht man unter einer Dense Timebase? Welche Eigenschaften einer solchen Dense Timebase kann man aus den fundamentalen Grenzen der Zeitmessung (Fundamental Limits of Time Measurement) ableiten?

- Dense Timebase heißt, Ereignisse dürfen zu jedem Zeitpunkt auftreten
- Wenn ein Event von zwei Nodes beobachtet wird, so unterscheidet sich der Zeitstempel für das Event möglicherweise um einen Tick
- Duration Measurement: $d_{obs} - 2g^{global} < d^{z}{}_{true} < d_{obs} + 2g^{global}$
  - Dauer d eines Events is begrenzt bei +/- 2*g
- Die zeitliche Reihenfolge von 2 Events e_1 und e_2 kann von deren Zeitstempeln abgeleitet werden, wenn gilt: $\left|t^{j}(e_1) - t^{k}e_2\right| \geq 2$
  - in Worten: Die Differenz zwischen deren Zeitstempeln muss größer-gleich 2 sein.

Was versteht man unter dem Action Delay? Wie groß ist der Action Delay in einem verteilten Echtzeitsystem (a) ohne globale Zeitbasis bzw. (b) mit globaler Zeitbasis?

- Der Action Delay ist das Intervall zwischen dem Zeitpunkt wann eine Nachricht vom Sender gesendet wird und dem Zeitpunkt wann der Sender weiß, dass die Nachricht permanent ist.
- Ohne globale Zeitbasis:
  - Maximaler Action Delay: $d_{max} + \varepsilon = 2d_{max} - d_{min}$
- Mit globaler Zeitbasis (timestamped messages):
  - Action Delay: $d_{max} + 2g$

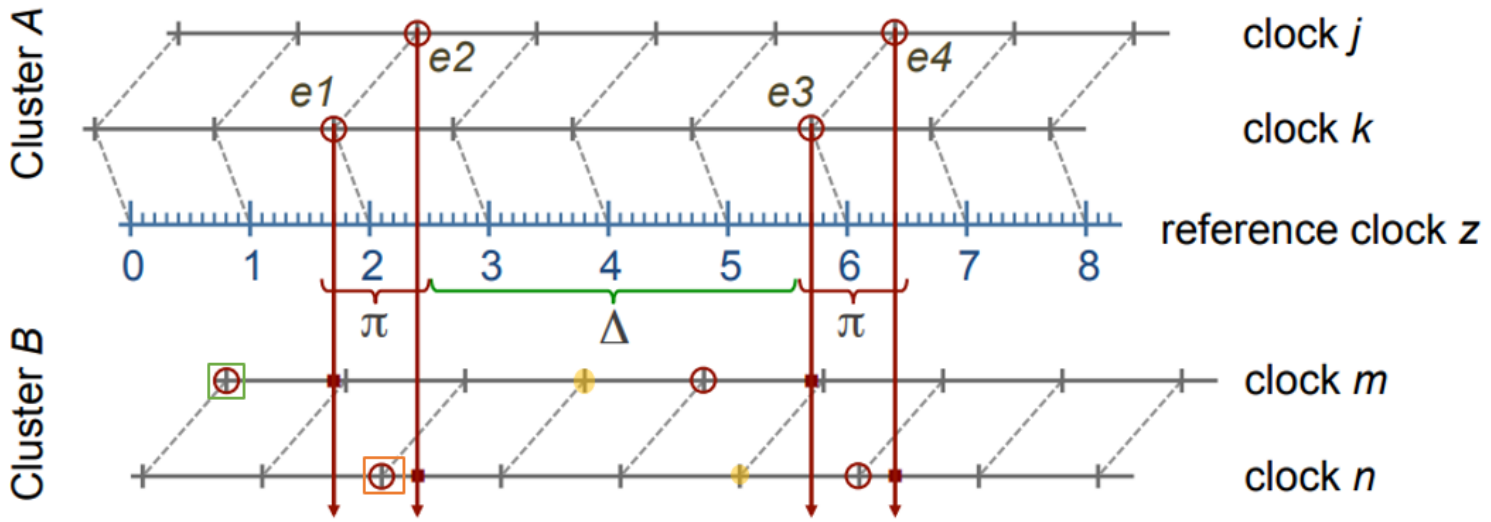
Zeigen Sie anhand eines Beispiels, dass zwei oder mehrere Cluster, die jeweils eine 1g/3g-sparse Timebase verwenden, nicht so kommunizieren können, dass die Ordnung aller Events auf der Empfängerseite immer eindeutig rekonstruiert werden kann.

Annahme: 2 Rechencluster A, B

- In jedem Cluster werden Uhren synchronisiert ($g = g^{global}$)
- Keine Synchronisation zwischen A und B
- Cluster A generiert Events, welche von B geordnet werden müssen:
  - B muss die Ordnung bzw. Gleichzeitigkeit von allen beobachteten Events bestimmen können.

➢ Zeitbasis von A muss 1g/4g-sparse sein. (1g = Aktivitätsintervall; 4g = Stilleintervall)

➢ Eine 1g/3g-sparse Zeitbasis ist nicht ausreichend, da die Zeitstempel im Cluster B alle den gleichen Abstand zum vorherigen haben. Dadurch lässt sich von B aus nicht mehr rekonstruieren, wann ein Aktivitätsintervall oder ein Stilleintervall aufgetreten ist.



e1, e2 werden im selben Aktivitätsintervall generiert: $t^n(e2) - t^m(e1) = 2$ (2 – 0 = 2)

e2, e3 werden in verschiedenen Aktivitätsintervallen generiert: $t^m(e3) - t^n(e2) = 2$ (6 – 4 = 2)

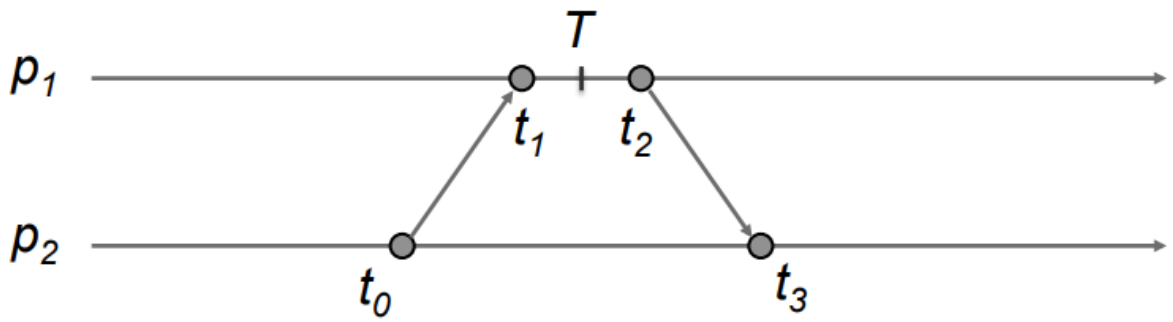
Welche Präzision ist bei der Uhrensynchronisation mittels Fault-Tolerant-Average-Algorithmus zu erreichen? Geben Sie die Formel an und beschreiben Sie diese.

$$\Pi\,(N, k, \varepsilon, \Gamma) = (\varepsilon + \Gamma)\,\frac{N - 2k}{N - 3k} = (\varepsilon + \Gamma)\,\mu\,(N, k)$$

- μ (N, k) ist der Byzantinische Fehlerterm
- (ε + Γ) ist der Drift-Offset mit dem Latenz-Jitter addiert.

Wie funktioniert Christians Algorithmus zur Uhrensynchronisation?

- Zeit-Request von p_2 zu p_1 zum Zeitpunkt t_0
- Antwort von p_1 kommt zum Zeitpunkt t_3 an: Enthält T (der Zeitpunkt an dem der Zeitstempel von p_1 in der Antwort festgeschrieben wird – bei t_2 wird dieser dann an p_2 geschickt.).
- p_2 berechnet die Round-Trip Time $d = t_3 - t_0$
- Uhrensynchronisation:
  - p_2 setzt lokale Zeit zu $T + d / 2$
  - Uhrensynchronisations-Fehler $\leq d / 2$



Was versteht man unter einer Lamport-Uhr? Erklären Sie deren Funktionsweise und beschreiben Sie, in welcher Hinsicht Vektor-Uhren den Lamport-Uhren überlegen sind.

- Eine Lamport-Uhr ist eine logische Uhr, die es erlaubt, den Events in einem verteilten System aufgrund ihrer Zeitstempel eine partielle kausale Ordnung zuzuweisen.

Funktionsweise:

- Die Zeitstempel sind wie bei einer logischen Uhr die Werte eines Counters, anstatt einer physikalischen Zeit.
- Dabei gibt es folgende Clock-Update-Rules zu beachten:
  - Im folgenden: C = Counter; p = Prozess
  - R1: p_i inkrementiert C_i für jedes lokale Event.
  - R2: Jede Nachricht enthält den Wert der Uhr des Senders (C_{msg})
  - R3: Wenn p_i eine Nachricht mit dem Timestamp C_{msg} empfängt:
    1. $C_i = \max(C_i, C_{msg})$
    2. $C_i = C_i + 1$;

➢ Lamport-Uhren erfüllen NICHT die starke Konsistenzbedingung für Uhren. Vektor-Uhren dahingegen schon.

Erklären Sie die Funktionsweise des Central-Master-Algorithmus zur Uhrensynchronisation. Welche Precision kann mit dem Algorithmus erreicht werden?

1. Der Master-Node sendet periodisch Synchronisationsnachrichten, welche die lokale Zeit beinhalten, an den Slave.
2. Der Slave stellt die lokale Uhr ein, indem er folgende Schritte ausführt:
    1. Aufzeichnen/Speichern der lokalen Ankunftszeit der Synchronisationsnachricht
    2. Berechnen der Differenz zwischen Master-Uhr und lokaler Uhr
    3. Korrigieren dieser Differenz, wegen der Latenz (lokaler Parameter)
    4. Korrigieren der lokalen Uhr

➢ Precision des Central-Master-Algorithmus: $\Pi_{central} = \varepsilon + \Gamma$

Erläutern Sie zwei Arten von Uhrensynchronisation. Wofür werden sie verwendet?

Externe Uhrensynchronisation

- Die Uhr gleicht sich mit einem externen System ab. Dieses System gibt eine Referenzzeit vor, welche angenommen wird ohne „Mitspracherecht“
- Hat Langzeitstabilität, jedoch möglicherweise niedrigere Verfügbarkeit
- Wird verwendet um verteilte Systeme mit einer Referenzuhr zu synchronisieren. (Zum Beispiel GPS)

Interne Uhrensynchronisation

- Alle Uhren eines geschlossenen Systems gleichen ihre Zeit untereinander ab. Danach gehen alle Uhren mit einer gewissen Precision gleich.
- Hohe Verfügbarkeit und gute Kurzzeitstabilität
- Wird verwendet um die Uhren von geschlossenen System zu synchronisieren, um eine bestimmte Präzisionsgrenze beizubehalten.

Wie genau können Intervallgrenzen mit globaler Zeit gemessen werden?

- $d_{obs} - 2g^{global} < d^{z}{}_{true} < d_{obs} + 2g^{global}$

➢ D.h., dass jede Grenze um $2g^{global}$ genau gemessen werden kann.

Wie definiert man im Kontext der Echtzeitsysteme eine Komponente? Wodurch ist diese Definition begründet?

- Eine Komponente besteht aus Hardware, Software und Status
- Eine Komponente ist ein Baustein eines größeren Systems
- Eine Komponente liefert einen klar definierten Service
- Die Integration einer Komponente erfordert nicht das Wissen, um das Interne der Komponente
- Eine Echtzeit-Komponente muss Zeit-Bewusst sein.

Was ist eine Observation (Beobachtung)?

- Sie erfasst Informationen über den Status einer RT-Entity.

  Observation = <Name, Time, Value>

  - Name: Name der RT-Entity
  - Time: Zeitpunkt in Echtzeit, wann die Beobachtung gemacht wurde
  - Value: Wert der RT-Entity

- Sie werden in/mit Nachrichten transportiert.

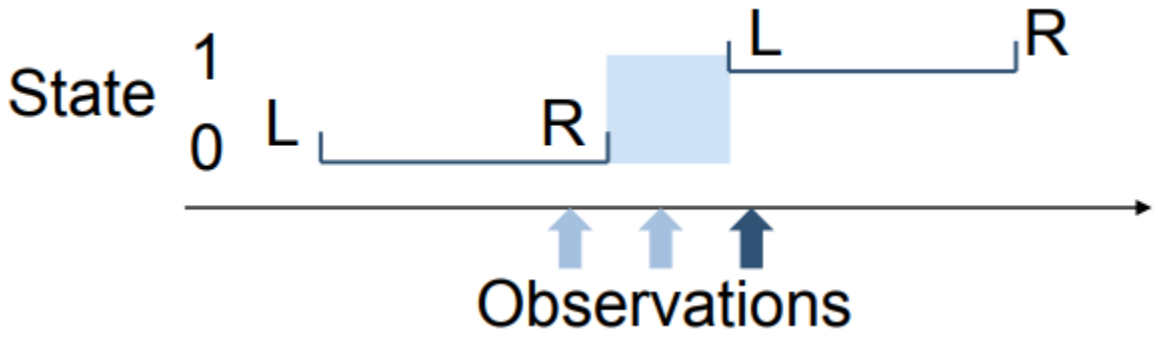
Was ist der Unterschied zwischen State- und Event-Observation?

Statusbeobachtung

- Wert beinhaltet den ganzen oder einen Teil des Status der RT-Entity
- Beobachtungszeitpunkt: Zeitpunkt, wann die RT-Entity beobachtet wurde.

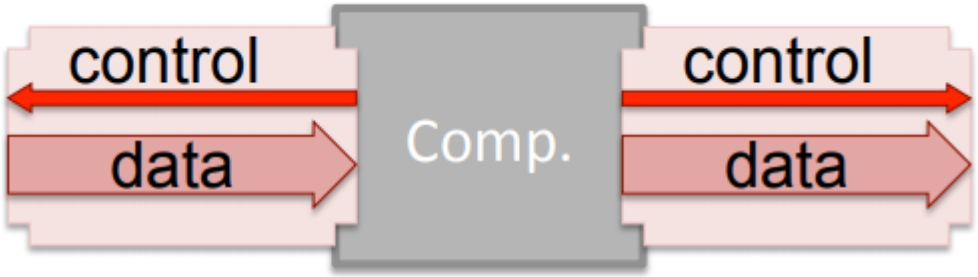
Eventbeobachtung

- Wert beschreibt die Differenz zwischen dem alten Status (dem zuletzt beobachteten Status) und dem neuen Status.
- Beobachtungszeitpunkt: Zeitpunkt des L-Events des neuen Status.



Was versteht man unter einem Temporal-Firewall-Interface?

- Ein Interface, welches die externe Kontroller einer Komponente verhindert.



component with two temporal-firewall interfaces

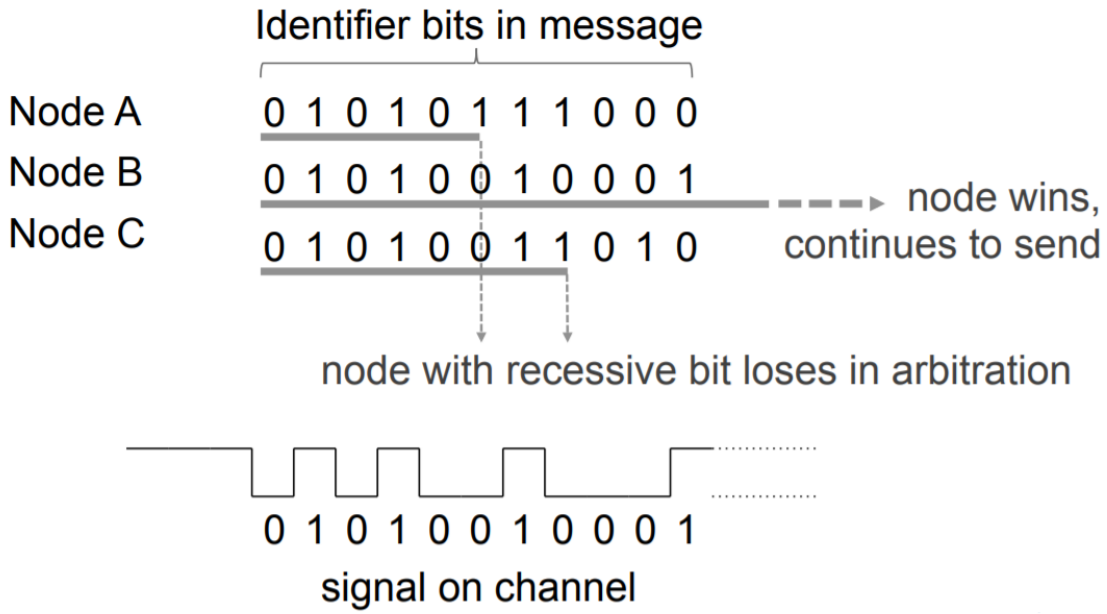
Was versteht man unter der Permanenz einer Nachricht? Wann ist eine Nachricht permanent?

- Eine Nachricht M_i wird permanent bei einem Objekt O, sobald alle Nachrichte - M_{i-1}, M_{i-2}, …, M_{i-n} – welche **vor** M_i an O gesendet wurden (zeitliche Ordnung) bei O angekommen sind.
  - Unter der Permanenz einer Nachricht M_i versteht man, dass alle Nachrichten, die (nach zeitlicher Ordnung) vor M_i an das Zielobjekt O geschickt wurden, bereits angekommen sind.

- NOTE: Aktionen, die auf nicht-permanente Nachrichten angewendet werden, können Fehler und Inkonsistenz verursachen.

Wie funktioniert die Arbitrierung in einem CAN-Netzwerk?

- Sie funktioniert mit CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance)
- Im CAN-Frame kommt zuerst ein SOF-Bit und dann direkt ein 11-Bit Identifier. Dieser Identifier wird benötigt für die Arbitrierung. Außerdem ist noch wichtig, dass standardmäßig die „0“ das dominante Bit ist und die „1“ das rezessive.



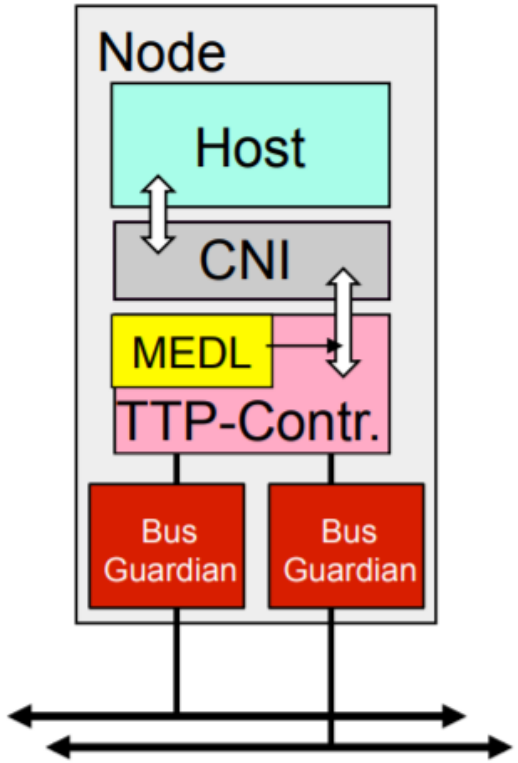
Kapitel 04_rt_communication

Was versteht man unter einem Bus Guardian für ein zeitgesteuertes Kommunikationsprotokoll? Wie funktioniert dieser?

Fehlerhafte Controller (o.ä.) könnten außerhalb ihres Timeslots senden. Dies würde den Kommunikationsablauf im System stören.

⇒Deshalb wurde das Konzept des Bus Guardian eingeführt. Dies ist ein unabhängiger Controller, welcher sich um den Buszugriff kümmert – er hat also eine Gate-Keeper-Funktion.

- Er weiß, wann ein Node senden darf.
- Er öffnet dementsprechend ein Gate zum Bus damit nur während des vorgesehenen Timeslots gesendet werden kann.

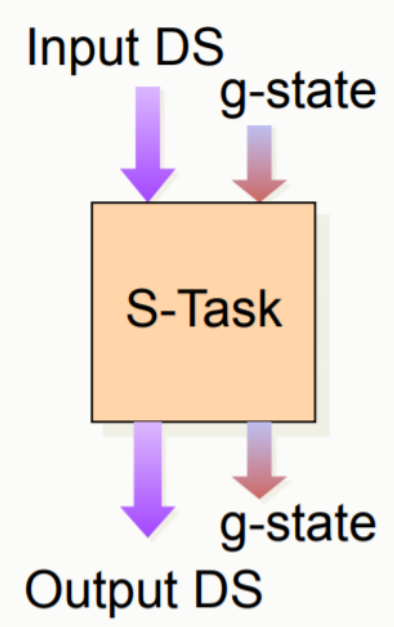


Bus Guardian bei FlexRay: https://elearning.vector.com/mod/page/view.php?id=239

Was versteht man unter einem Simple Task (S-Task) und was versteht man unter einem Complex Task (C-Task)?

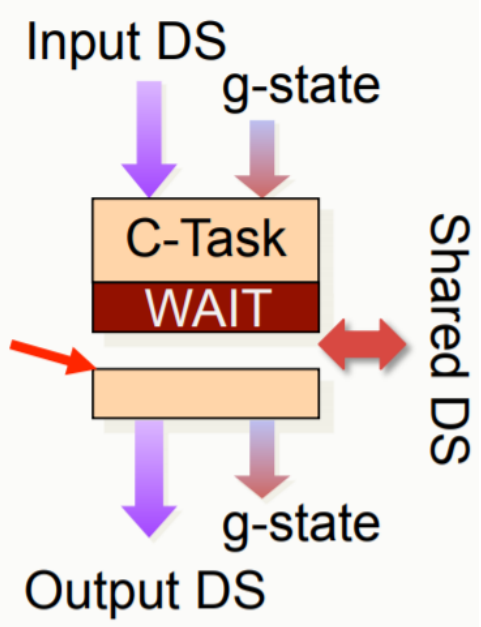
S-Task

- Wird von Anfang an bis zum Ende ohne jeglichen Delay ausgeführt, wenn die CPU zugeteilt wurde.
- Kein Blocking im Task-Body (keine Synchronisation, Kommunikation)
- Unabhängiger Fortschritt
- Inputs sind in der Input-Datenstruktur beim Task-Start verfügbar.
- Outputs sind in der Output-Datenstruktur zum Task-Ende bereit.
- API: Input DS, Output DS, g-state



C-Task

- Kann ein oder mehrere WAIT-Statements im Task-Body beinhalten.
- Mögliche Abhängigkeiten durch Synchronisation, Kommunikation
- Fortschritt hängt von anderen Tasks im Node oder der Umgebung ab.
- Das Timing des C-Tasks ist ein globales Problem.
- API: Input DS, Output DS, g-state, Shared DS



Welche Eigenschaften müssen beschrieben werden, um das Interface (Linking-Interface) zwischen zwei Subsystemen eines RTS vollständig zu charakterisieren?

1. In Messages
2. Out Messages
3. Temporal
4. Meaning

Statische und dynamische Attribute einer RT-Entity

- Statische Attribute
  - Name
  - Typ
  - Wertedomäne
  - Maximale Änderungsrate
- Dynamische Attribute
  - Ist-Wert zu einem bestimmten Zeitpunkt

Wann spricht man von einem temporally accurate RT-Image?

- Das RT-Image ist zum gegenwärtigen Zeitpunkt t_i zeitlich genau (temporally accurate), wenn gilt:
  - $\exists t_j \in RH_i : Value\ (RT\ image\ at\ t_i) = Value\ (RT\ entity\ at\ t_j)$
- In Worten: Das RT-Image ist zu einem Zeitpunkt t_i zeitlich genau, wenn der Zeitpunkt der Beobachtung und der Zeitpunkt der Verwendung nicht mehr als d_{acc} abweichen

Erklären Sie die Funktionsweise eines zeitgesteuerten Kommunikationsprotokolls. Welche Vorteile bietet die zeitgesteuerte gegenüber der ereignisgesteuerten Kommunikation?

- Fortschreiten von globaler Zeit steuert Protokoll. Der Zeitpunkt zu dem eine Nachricht gesendet wird, ist vorher bekannt für alle Empfänger
- Maximale Ausführungszeit ist ungefähr die gleiche wie die durchschnittliche Ausführungszeit. Deshalb geringer Lesefehler (Jitter)
- Fehlererkennung durch Empfänger, basierend auf vorherigem Wissen
- Das Protokoll ist unidirektional → sehr passend für Multicast Umgebung

- Vorteile:
  - Lesefehler sind verglichen zur durchschnittlichen Ausführungszeit geringer als bei ereignisgesteuertem
  - Besser geeignet für Multicast Umgebungen (ereignisgesteuert braucht Bestätigung bei Fehlererkennung → Kollisionen)

Charakterisieren Sie die Funktionsweise und Eigenschaften eines ereignisgesteuerten Kommunikationsprotokolls

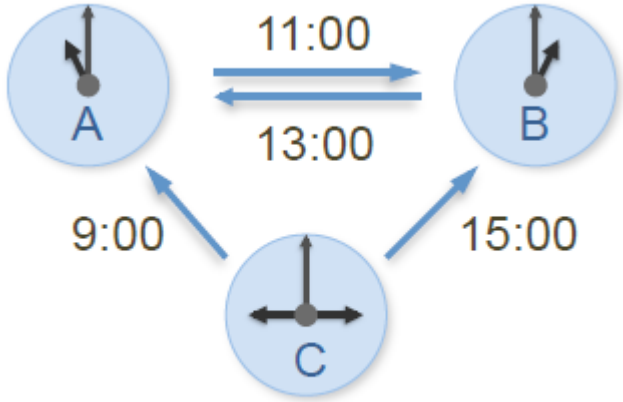
- Ein Event beim Sender löst eine Protokollausführung zu einem willkürlichen Zeitpunkt aus
- Die maximale Ausführungszeit und der Lesefehler des Protokolls sind groß im Vergleich zur durchschnittlichen Ausführungszeit
- Fehlererkennung ist beim Sender
- Fehlererkennung braucht Bestätigung. Dadurch entsteht Kollisionsverkehr in Multicastnetzwerken
- Keine temporäre Verkapselung
- Explizite Flusskontrolle um den Empfänger vor Informationsüberflutung zu schützen

In welchem Verhältnis, stehen kausale und temporale Ordnung von Events? Beschreiben Sie beides und geben Sie ein Beispiel an.

- Kausale Ordnung impliziert temporäre Ordnung
- Temporäre Ordnung ist notwendig aber nicht ausreichen für kausale Ordnung
- Temporäre Ordnung:
  - Abhängig vom Zeitstempel der physikalischen Zeit
- Kausale Ordnung:
  - Abhängig von den „kausalen Eigenschaften“ zwischen Events
  - Ist Ereignis a ein Grund von Ereignis b, so bringt eine kleine Variation in a eine kleine Variation in b mit sich.
- Beispiel:
  - Ereignis a: Telefon klingelt
  - Ereignis b: Jemand betritt den Raum
  - a vor b: kausale Abhängigkeit möglich (betritt Raum, weil das Telefon klingelt)
  - b vor a: kausale Abhängigkeit unwahrscheinlich (Telefon klingelt, weil man den Raum betritt? :D)

Illustrieren Sie anhand eines Beispiels, wie eine bösartige Uhr die Konvergenz der Uhrensynchronisation unterbinden kann. Wie nennt man solch eine bösartige Uhr? Wie viele Uhren braucht man zur Uhrensynchronisation, um auch dann noch Uhren synchronisieren zu können, wenn sich k bösartige Uhren im System befinden?

- So eine Uhr nennt man „malicious“ oder byzantinische Uhr
- Beispiel:
  - Uhr 3 ist fehlerhaft und sendet anstatt 12 Uhr einmal 9 und einmal 15 Uhr. Dies führt zu fehlerhaften Synchronisation.
- Um dieses Problem zu beheben muss die Anzahl von Uhren $N \geq 3k + 1$



two-faced malicious clock

Was versteht man unter TAI und UTC. Charakterisieren Sie diese Standards kurz.

TAI (International Atomic Time)

- Standard für physikalische Zeit
- Definiert die Epoche, den Ursprung der Zeimessung als 1.1.1958 um 00:00:00
- Chronoskopische Zeitskala, d. h. eine Zeitskala ohne Diskontinuitäten.

UTC (Universal Time Coordinated)

- Astronomischer Zeitstandard
- Basis für die Zeit der “Wanduhr”
- Dauer einer Sekunde stimmt mit der des TAI Standards überein
- Anzahl von Sekunden pro Stunde gelegentlich durch Einfügen einer Sprungsekunde in UTC manipuliert. Um Synchronität zwischen “Wanduhren” und der astronomischen Phänomenen wie Tag und Nacht zu garantieren

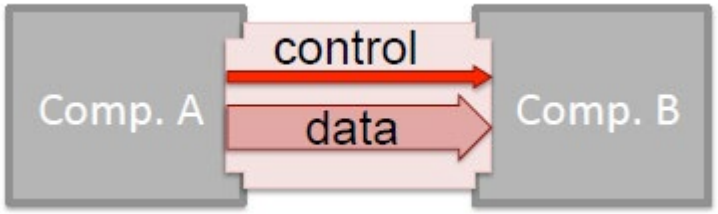
Welche Interfaces einer Echtzeitkomponente unterscheiden wir? Beschreiben Sie kurz die Aufgabe und Charakteristik jedes dieser Interfaces.

- Realtime Service (RS) oder Linking Interface (LIF):
  - In periodischen Kontrollanwendungen
  - RT Observation
  - Zeitsensitiv
- Diagnostic and Maintenance (DM) Interface – Technology Dependent (TDI)
  - Sporadischer Zugriff
  - Setzt Wissen über Internes eines Nodes voraus
  - Nicht Zeitsensitiv
- Configuration Planning (CP) Interface – Technology Independent (TII)
  - Sporadischer Zugriff
  - Notwendig um einen neuen Node in eine Konfiguration zu installieren
  - Nicht Zeitsensitiv
- Local Interface zur Komponentenumgebung

Was versteht man unter einem Information Push Interface bzw. einem Information Pull Interface?

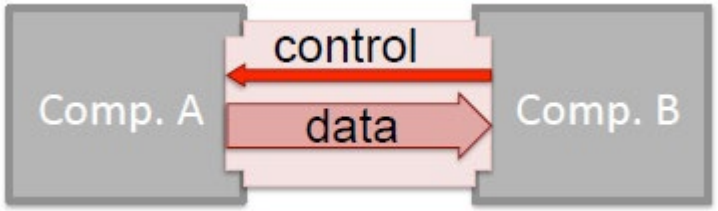
Information Push

- Informationen zum Konsumenten pushen (schieben)



Information Pull

- Konsument ruft Informationen ab wenn er sie benötigt



In welchen drei Phasen erfolgt die Synchronisation von Uhren mittels eines verteilten Uhrensynchronisationsverfahrens?

- Nodes tauschen Nachrichten aus und erhalten Informationen über globale-Zeitzähler von anderen Nodes
- Jeder Node analysiert gesammelte Informationen (Fehlererkennung) und führt die Konvergenzfunktion aus um einen korrekten Term für seinen lokalen globale-Zeitzähler zu erstellen
- Jeder Node stellt seinen lokalen Zeitzähler auf den korrigierten Term ein

Was versteht man unter einem H-State und einem G-State? Beschreiben Sie die Bedeutung dieser beiden Konzepte

H-State (History State)

- Umfasst alle notwendigen Daten um einen initialisierten Node oder Task (i-state) zu einem gegebenem Zeitpunkt zu starten
- Größe des h-state ändert sich mit der Zeit
- Relatives Minimum umgehend nach einer abgearbeiteten Berechnung
- Sollte zum Reintegrationspunkt möglichst gering sein

G-State

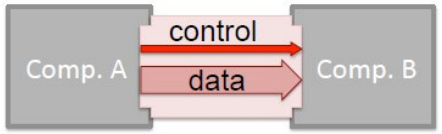
- Ist minimaler h-state eines Subsystems
- Tasks sind inaktiv und die Kanäle bereinigt
- Ist quasi der Reintegrationspunkt

Durch welche Eigenschaften ergänzen sich die interne und die externe Uhrensynchronisation?

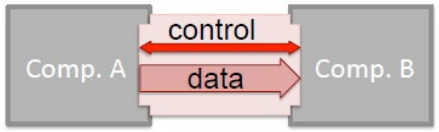
- Interne Uhrensynchronisation besitzt gute Kurzzeitstabilität und hohe Verfügbarkeit
- Externe Uhrensynchronisation besitzt gute Langzeitstabilität, ist dafür aber möglichweise weniger gut verfügbar
- Mögliche Ergänzung:
  - Gateway zu externer Referenzzeit = Rate-Master zu interner Synchronization

Wodurch sind die folgenden Interface-Typen charakterisiert: Elementary Interface, Composite Interface und Temporal Firewall Interface? Beschreiben Sie jeden Interface-Typ kurz

- Elementary Interface
  - Unidirektionaler Kontrollfluss (Schreiben zu dualported RAM)



- Composite Interface
  - Bidirektionaler Kontrollfluss (Queue von Ereignisnachrichten)



- Temporal Firewall Interface
  - Verhindert Kontrolle einer Komponente von außerhalb

Beschreiben Sie die Begriffe Elementary Interface und Composite Interface. Welche Rolle spielen diese Arten von Interfaces für Echtzeitsysteme?

Elementary Interface

- Unidirektionaler Kontrollfluss
  - Spielt eine Rolle wenn beispielsweise Daten schnell in eine Richtung übertragen werden müssen (Schreiben zu dualported RAM)

Composite Interface

- Bidirektionaler Kontrollfluss
  - Queue von Ereignisnachrichten

Wann spricht man von einem phasensensitiven, wann von einem phaseninsensitiven Real-Time Image?

Phasensensitives RT-Image

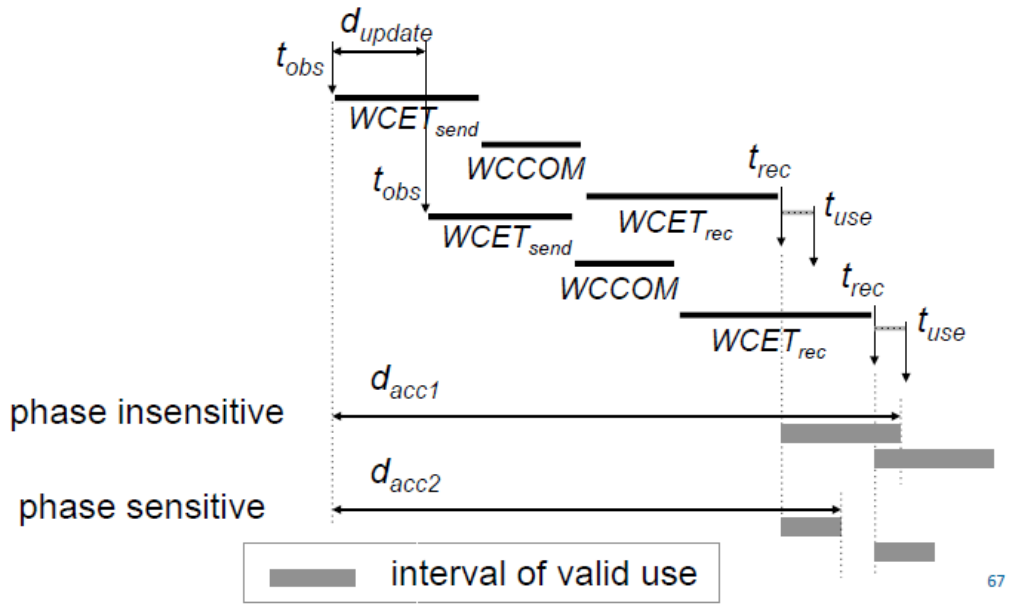
- Ein Image ist phasensensitiv, wenn:

$$d_{acc} \leq d_{update} + WCET_{send} + WCCOM + WCET_{rec}$$
$$\text{and } d_{acc} > WCET_{send} + WCCOM + WCET_{rec}$$

- WCETsend is sendingtask
- WCETrec is receivingtask
- WCCOM is Communication

Phaseninsensitives RT-Image

$$d_{acc} > d_{update} + WCET_{send} + WCCOM + WCET_{rec}$$

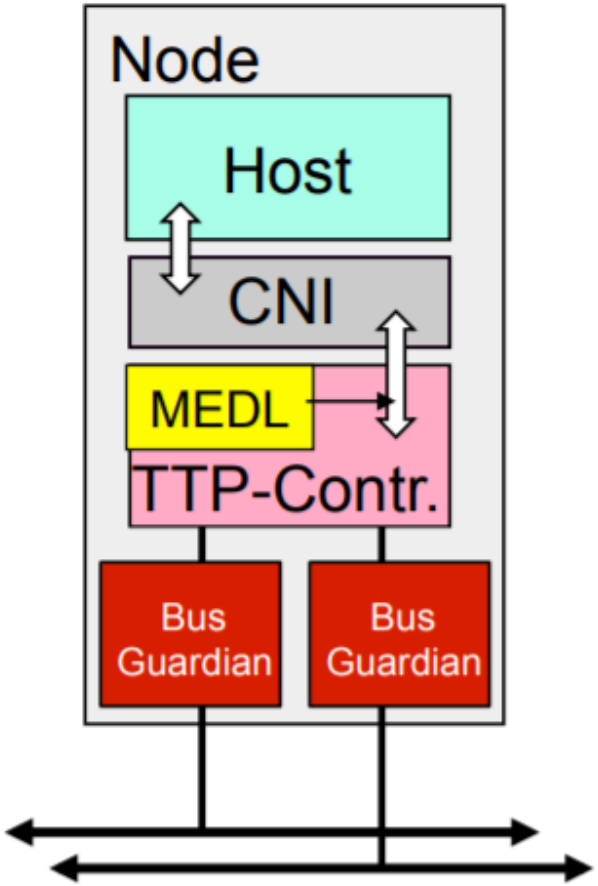


Was versteht man unter einem End-to-End Protocol? Worin liegt der Vorteil der Verwendung eines solchen Protokolls?

- Zeigt und kontrolliert den beabsichtigten Effekt einer Kommunikation am beabsichtigten Endpunkt
- Bietet eine höhere Fehlererkennungsabdeckung
  - Beispiel: eine Sensornachricht berichtet über Änderung und ist End-to-End-Bestätigt per Kommandomessage zum Flowaktuator

Skizzieren Sie einen Fail-Silent TTP Node und beschreiben Sie dessen Komponenten und deren Aufgaben kurz.

- Host:
  - Sendet Daten an CNI
  - Schreibt Kontrolldaten in das CNI
  - Lest erhaltene Daten/Statusinformationen vom CNI
- CNI:
  - Data-Sharing Interface zwischen Host und TTP-Controller
  - Fungiert als Firewall für Kontrollsignale
- MEDL (Message Descriptor List):
  - Speichert den Nachrichten-Schedule.
- TTP-Controller:
  - Schreibt erhaltene Daten in das CNI
  - Setzt Stati
  - Verschickt Nachrichten welche von CNI-Daten zusammengestellt wurden über das Netzwerk.
- Bus-Guardian:
  - Unabhängiger Controller, der den Buszugriff regelt.



Link: https://www.semanticscholar.org/paper/Avoiding-the-babbling-idiot-failure-in-a-system-Temple/ecec65dfcd073dad3df9b240824584d92594fbb9

Was versteht man unter der Idempotenz eines Sets von Nachrichten?

Ein Set an Nachrichten ist idempotent, wenn das Erhalten von mehreren Nachrichten dieses Sets den Gleichen Effekt hat, wie das Erhalten von einer Nachricht dieses Sets.

Beispiele:

- Duplizierte Statusnachrichten sind idempotent
- Duplizierte Eventnachrichte sind idempotent

➢ Idempotenz von redundanten Nachrichten vereinfacht das Design von fehlertoleranten Systemen.

Beispiel aus der Mathematik: Bezüglich der Multiplikation sind die Lösungen 0 und 1 der Gleichung x^{2}=x die einzigen idempotenten reellen Zahlen.
=> Egal wie oft man mit 1 multipliziert, es kommt immer 1 raus.

Charakterisieren Sie das Ressourcenmanagement in einem zeitgesteuerten Echtzeitbetriebssystem.

- In einem zeitgesteuerten OS gibt es kaum dynamisches Ressourcenmanagement.
  - Statische CPU-Zuweisung
  - Autonomes Speichermanagement. (Braucht kaum Aufmerksamkeit des OS)
  - Buffermanagement ist minimal → Keine Queues
  - Implizite, vorgeplante Synchronisation erfüllt alle Synchronisations-Anforderungen und Prioritätsbeschränkungen
    - S-Tasks only
  - Keine explizite Synchronisation
- OS wird simpel und kann formell analysiert werden.

Wie funktioniert der Priority Ceiling Algorithmus?

Jeder Prozess hat eine Standard-Priorität

1. Weise jeder Ressource eine Prioritätsgrenze zu. Die Prioritätsgrenze ist gleich der Priorität des Highest-Priority-Tasks, der die Ressource benutzt.
2. Zu jedem Zeitpunkt wird ein Task, der das Maximum der eigenen statischen Priorität und der Grenzwerte all seiner blockierten Ressourcen ist, mit einer dynamischen Priorität ausgeführt.

➢ Ein Task kann nur dann Ressourcen annehmen, wenn die Priorität dieses Tasks höher als die Prioritätsgrenzen aller Ressourcen, die von anderen Tasks blockiert werden, ist.

Was ist Priority Inheritance? Wofür wird es verwendet?

- Wenn ein Task mit niedriger Priorität einen oder mehrere Tasks mit höherer Priorität blockiert, nimmt er temporär die höchste Priorität der von ihm blockierten Taks an.
- Wird verwendet damit Tasks höherer Priorität schneller an die geforderten Ressourcen kommen.

Was ist Priority-Ceiling? Wofür wird es verwendet? Was ist der Unterschied zu Priority-Inheritance? Beschreiben Sie ein konkretes Szenario.

- Priority-Ceiling ist ein Protokoll zur Lösung des Priority-Inversion-Problems.
  - **Grobe Funktionsweise**:
  - Die Priorität eines Tasks ist seine eigene statische Priorität plus die Prioritäten aller von ihm blockierten Ressourcen.
  - Ein Task kann nur dann Ressourcen annehmen, wenn die Priorität dieses Tasks höher als die Priorität aller Ressourcen, die von anderen Tasks blockiert werden, ist.
- Es wird vom Scheduler benutzt, um Tasks zu priorisieren und somit festzulegen, in welcher Reihenfolge diese ausgeführt werden.
- Im Vergleich zu Priority-Inheritance, kann Priority-Ceiling Dead-Locks vermeiden.

Beispiel:

Gegebene Tasks: A(high); B(medium); C(low)

Gegebene Ressourcen: R1(high); R2(medium)

1. Task C nimmt sich R1
2. Task A will nun auch R1, kann aber nicht, da Task C R1 blockiert.
3. Task B nimmt sich R2. Da nun Task B eine höhere Priorität als Task C hat, bekommt er die Rechenzeit.

➢ Somit muss Task A, der mit der höchsten Priorität, auf alle anderen warten.

➢ Zur Lösung ist Priority-Ceiling da, der dem Task die Priorität der Ressourcen erben lässt.

Was ist der FTA und wie funktioniert er?

- Fault tolerant average (FTA) algorithm
- Kalkuliert die Differenz zwischen der lokalen und allen anderen Uhren im Netzwerk
- Sortiert die Differenzen
- Eliminiert immer den kleinsten und den größten Differenzwert k
- Korrektionsterm ist Durchschnitt der übrigen Differenzen N-2k

Was ist State-Estimation? Geben Sie ein Beispiel an. Beschreiben Sie einen möglichen Zusammenhang zur Uhrensynchronisation

- Schätzt den aktuellen Status der RT-Entity
- Wird periosich innerhalb des RT-Objekts berechnet
- Basierend auf dem Rechenmodell der Dynamik der RT-Entity
- **Mutmaßung:**
  - Für die Uhrensynchronisation kann eine Zeitabweichung abgeschätzt werden um sich eventuell wieder an die korrekte Zeit anzunähern.

Geben Sie ein Beispiel für einen Hidden Channel an. In welchem Zusammenhang steht dazu der Begriff Permanence?

- Hidden Channel ist zum Beispiel ein Kessel mit Ventil und einem Drucksensor. Betätigt man jetzt das Ventil, so ändert sich der Druck im Kessel und somit auch der Wert am Sensor, ohne dass man dem Sensor dies explizit mitgeteilt hat.
- Permanent sind die Nachrichten, wenn alle vorherig gesendeten Nachrichten auch bereits angekommen sind. Wären die Nachrichten dies nicht, so würden in diesem Fall Fehler auftreten.

Wie wird bei CAN der Zugriff auf das Medium geregelt? Wodurch ist die maximale Übertragungsrate bestimmt?

- Dominantes Bit auf der Leitung gewinnt immer
- Kleinster Identifier gewinnt immer, da die dominante 0 die Leitung nach unten zieht
- Maximale Übertragungsrate:
  - Abhängig von Ausbreitungsgeschwindigkeit der Bits am Bus und Länge der Leitung

Finde ich nicht in den Vorlesungen

Gegeben sind folgende unabhängige periodische Tasks, mit Deadline Intervall gleich Taskperiode: T1(4|8); T2(3|16); T3(1|4)

a) Ermitteln Sie, ob dieses Task-Set mit dem RMS schedulbar ist:

- $U = \frac{4}{8} + \frac{3}{16} + \frac{1}{4} = \frac{15}{16}$
- $U \leq 3\left(2^{\frac{1}{3}} - 1\right) = 0.78$
- $\frac{15}{16} \leq 0.78\ ist\ falsch$ → System ist nicht mit RMS schedulbar.

b) Zeigen Sie, wie der Prozess mit RMS geschedulet wird:

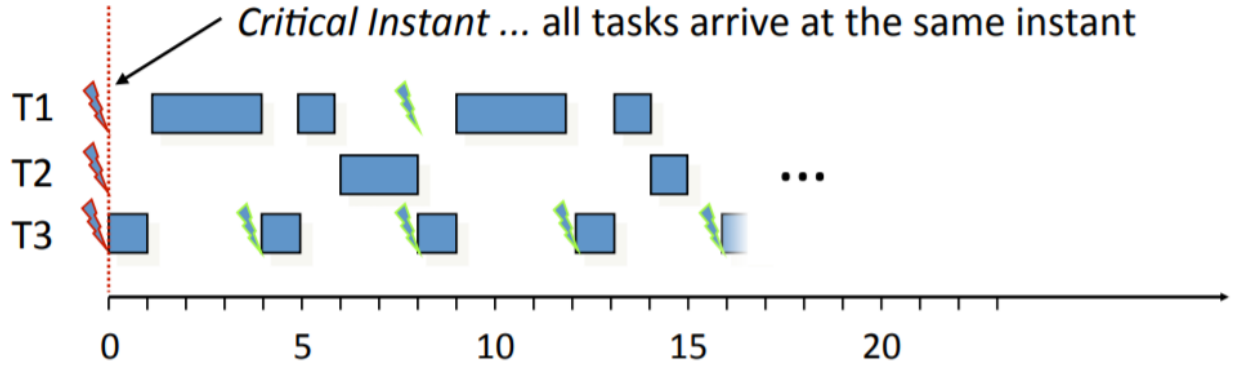


Bild stimmt mit Werten überein!

c) Führen Sie das Earliest Deadline First Scheduling Verfahren durch:

- $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$
- $U = \frac{15}{16} \leq 1$ ist wahr → System ist mit EDF schedulbar.

- Beim EDF-Algorithmus bekommt der Task mit der frühesten Deadline die höchste Priorität.

Erklärung: https://www.youtube.com/watch?v=ejPXTOcMRPA

Codieren Sie die Bitfolge 110001110101 mit dem MFM Code

- 1: Flankenwechsel bei Datenflanke
- 0 gefolgt von 0: Flankenwechsel bei Taktflanke
- 0 nach oder vor 1: Do nothing

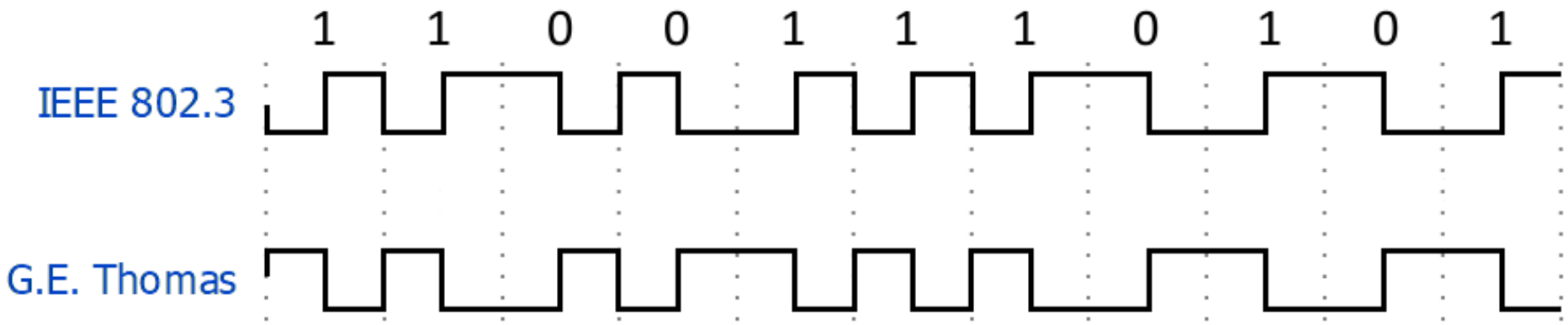
T = Taktflanke
D = Datenflanke

Finde ich nicht in den Vorlesungen

Codieren Sie die Bitfolge 11001110101 mit dem Manchester Code.

- IEEE 802.3 :
  - 0 – Signal geht von High zu Low
  - 1 – Signal geht von Low zu High
  - Wird meistens verwendet

- G.E Thomas:
  - 0 – Signal geht von Low zu High
  - 1 – Signal geht von High zu Low



Finde ich nicht in den Vorlesungen

Was versteht man unter dem byzantischen Fehler? In wieweit spielt dieser eine Rolle?

- Ein byzantischer Fehler bezeichnet ein System, welches sich beliebig falsch verhält.
  - Beispiel: Eine Uhr sendet einmal 10 und einmal 12 Uhr obwohl bei ihr 11 Uhr ist
- Es werden viel mehr Systeme benötigt um das System vor Schaden zu bewahren.
- Fehlerhafte Uhren können Schaden am System verursachen.

Zählen Sie drei wichtige Eigenschaften von Leitungscode auf.

- Robustheit
  - Was passiert bei Bitfehlern?
- Selbsttaktung
  - Kann das Taktsignal aus dem Signalverlauf rekonstruiert werden?
- Bandbreitenbedarf (Übertragungsgeschwindigkeit)
  - Wie viele Redundanzen durch Codierung?

- Synchronisation
  - Der Datenempfänger muss sich auf den Sender synchronisieren können, um die Daten korrekt auszuwerten.

Was ist Bitstuffing? Wofür wird es benötigt?

- Bitstuffing fügt nach gleichen Bits ein invertiertes Bit ein.
  - Bei Can wird nach fünf 0en eine 1 eingefügt
- Das verhindert, dass Sender und Empfänger asynchron laufen und der Empfänger mehr oder weniger gleiche Bits liest als der Sender gesendet hat.

Finde ich nicht in den Vorlesungen

Nennen Sie Timeouts des ARINC 629 Protokolls und beschreiben Sie deren Zweck.

ARINC ist ein Wartezimmer-, bzw. Minislottingprotokoll. Im ersten Intervall finden sich alle Tasks, die schreiben wollen, in einem "Warteraum". Im nachfolgenden Intervall (Epoche) schreiben alle wartenden Prozesse, bevor neue den Warteraum betreten können.

- Timing Intervall: Verhindert Monopolisierung des Buses, ist identisch für alle Nodes
- Terminal Gap: Kontrolliert Buszugriff, unterschiedlich für alle Nodes
- Synchronisation Gap: Kontrollierter Zugriff zum Warteraum, identisch für alle Nodes

Vorlesung 04_rt_communication

Wikipedia: https://de.wikipedia.org/wiki/ARINC_629

Wie funktioniert der EDF-Algorithmus, wie der Least Laxity Algorithmus? Welcher ist besser?

EDF (Earliest Deadline First) Alogithmus

- Die Deadlines bestimmen die Reihenfolge
- Der mit der frühesten Deadline wird als erstes ausgeführt

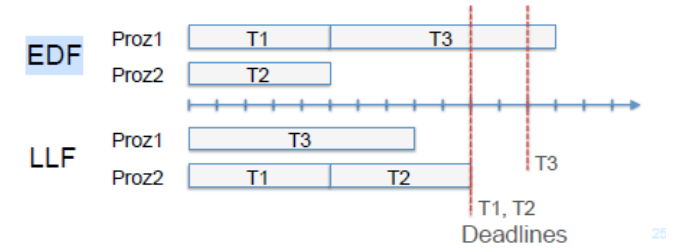
Least Laxity Algorithmus

- Laxity ist Differenz zwischen Deadline und verbleibende Berechnungszeit
- Der Task mit der kleinsten Laxity bekommt die höchste Priorität und ist der nächste, der ausgeführt wird
- Optimal für Uniprozessor Systeme

Scheinbar ist LLF besser als EDF

Task set

Task	Deadline	WCET
T1	10	5
T2	10	5
T3	12	8



Vorlesung 07_rt_sceduling

Welchen Schedulability-Algorithmus verwendet RMA? Erklären Sie die Variablen.

- Er verwendet RMS (Rate-Monotonic-Scheduling)
- Voraussetzung: Alle Tasks sind periodisch
- Der Task mit der kürzesten Periode bekommt die höchste Priorität.
- Von den Tasks, die bereit sind, wird der mit der höchsten Priorität als nächstes ausgeführt.
- Folgendes muss gelten, damit ein Task-Set mit dem RMS schedulbar ist :

$$\sum_{i=1}^{n} \frac{C_\mathrm{i}}{T_\mathrm{i}} \leq n(2^{\frac{1}{n}} - 1)$$

C_i: Ausführungszeit eines Tasks
T_i: Periodenlänge eines Tasks
n: Anzahl der Tasks

Finde ich nicht genau so in den Vorlesungen

Was ist der Unterschied zwischen Sampling und Polling?

Sampling

- Memory-Element liegt in der Sphere of Control des Sensors
- Im Fehlerzustand ist Sampling robuster, da wenn die Übertragungsleitung gestört wird, dies nur Einfluss auf den Zeitpunkt des Abtastens hat.
- Wenn der Controller neugestartet wird, bleiben beim Sampling die Werte im Memory-Element weiterhin vorhanden.

Polling

- Memory-Element liegt in der Sphere of Control des Controllers
- Beim Polling merkt sich das Memory-Element jeden Fehler auf der Übertragungsleitung. Das kann leichter zu fehlerhaften abgetasteten Werten führen.
- Wenn der Controller neugestartet wird, sind beim Polling alle Werte im Memory-Element gelöscht, da es in dessen SOC liegt.

Gegeben sind: Bandbreite 2Mbit/s, Kanal: 2km lang, Nachrichtenlänge: 320bit – Geben Sie die Grenzen der Protokolleffizienz an.

$$ProtocolEffeciency = \frac{MessageLength}{MessageLength + BitLength}$$

$$BitLength = PropagationDelay \cdot Bandwidth$$

$$PropagationDelay = \frac{CableLength}{SpeedThroughCable}$$

$$SpeedThroughCable = 200 \cdot 10^6 \frac{m}{s}$$

Durchschnittlicher SpeedThroughCable
(Ist entweder gegeben oder man nimmt diese Zahl)

$$PropagationDelay = \frac{2000m}{200 \cdot 10^6 \frac{m}{s}} = 10^{-5}s$$

$$BitLength = 10^{-5}s \cdot 2 \cdot 10^6 \frac{bit}{s} = 20bit$$

$$ProtocolEffeciency = \frac{320bit}{320bit + 20bit} = 0.94 = 94\%$$

Vorlesung 04_rt_comminication so halb

Erklären Sie die explizite und implizite Flusskontrolle. Welchen Einfluss haben diese auf das Echtzeitverhalten eines Systems?

Explizit:

- Sender kann immer senden, wenn der Kanal frei ist.
- Sender übermittelt eine Nachricht an den Empfänger und wartet auf Bestätigung.
  - Durch das Bestätigen wird ein Rückkanal notwendig
- Empfänger hat Erlaubnis den Sender zu bremsen
- Fehlererkennung beim Sender
- Fehlende Bestätigung bedeutet:
  - Nachrichtenverlust
  - Empfänger verspätetet sich
  - Empfänger ist gescheitert

➢ Kann bei großer Last Trashing verursachen, dies führt zu einem größeren Protokoll Latenz Jitter und ist daher für Echtzeitsysteme ungeeignet.

Implizit:

- Sender und Empfänger vereinbaren im Vornherein eine Übertragungsrate (Intervall in dem gesendet wird/werden muss)
  - Diese muss vom Empfänger machbar sein
- Fehlererkennung von Empfänger
  - Keine Nachrichten-Bestätigung
  - Unidirektionale Nutzung des Kanals
- Multicast ist sehr einfach realisierbar

➢ Wenn richtig geplant auch bei großer Last noch funktionsfähig. Kleiner Protokoll Latenz Jitter, daher für RTS geeignet.

Welches Media Access Control Verfahren wird bei CAN verwendet? Begründung.

- CAN verwendet CSMA/CA (CA = Collision Avoidance)
- Greifen mehrere Netzwerkstationen auf denselben Übertragungskanal zu, so werden durch die Arbitrierung Kollisionen verhindert, da immer der mit der höchsten Priorisierung gewinnt.
- Im Gegensatz zum CSMA/CD Verfahren werden somit wichtigere Nachrichten (höher Priorisierte) immer vor den Unwichtigeren (niedriger Priorisierten) Nachrichten gesendet.

Was ist ein Real Time Image? Welche Aussagen lass sich treffen?

- Das RT-Image ist eine Abbildung des Zustands einer RT-Entity.
- Gültig zu einem bestimmten Zeitpunkt, falls es eine genaue Darstellung der entsprechenden RT-Entity bzgl. Wert und Zeit ist.
- Nut gültig während eines spezifizierten Intervalls von Echtzeit
- Kann auf eine Beobachtung oder Zustandsschätzung (State-Estimation) basieren.
- Kann in einem Datenobjekt gespeichert werden. Entweder in einem Computer (einem RT-Objekt) oder außerhalb in einem Aktuator.

Finde ich nicht in den Vorlesungen

Was ist ein Agreement-Protokoll? Wofür ist das wichtig?

- Ein Agreement-Protokoll liefert einen Konsens über die Werte eines Beobachters und über die Zeit, wann der Beobachter unter eine Anzahl von fehlerfreien Mitgliedern eines Ensembles aufgetaucht ist.

- Falls eine RT-Entity von zwei (oder mehr) Nodes eines verteilten Systems beobachtet wird, könnte folgendes passieren:
  - Dasselbe Event kann einen anderen Zeitstempel erhalten.
  - Wenn man von einem analogen Sensor liest, werden viele dichte Mengen von Werten zu digitalen Werten gemapped (A/D-Wandler) – Diskretisierungsfehler entsteht.
- Wann immer dichte Mengen von Werten zu einer diskreten Darstellung gemapped werden (z.B. viele physikalische Volt-Messungen werden zu Metern gemapped), wird ein Agreement-Protokoll benötigt, um sich einig zu werden, welche Sensorwerte redundant sind.

Nennen Sie drei Beispiele für „a priori“ Wissen (vorheriges Wissen), welches für die Erfüllung von Echtzeitanforderungen verwendet werden kann.

- **Fehlererkennung**: Es ist vorher bekannt, wann ein Node eine Nachricht senden muss.
- **Flusskontrolle**: Es ist vorher bekannt, wie viele Nachrichten in einem Peak-Load Szenario ankommen werden (Ressourcenplanung)
- **Nachrichtenidentifikation**: Der Zeitpunkt der Nachrichtenübertragung identifiziert eine Nachricht (Reduzierung der Nachrichtengröße)

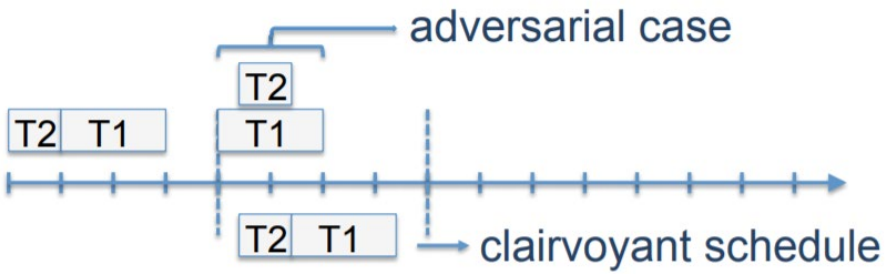
Weisen Sie nach, dass es keinen optimalen Online-Scheduler gibt.

- Wenn ein periodischer Task abgearbeitet wird, muss der sporadische Task warten bis er fertig ist.
- Da aber der sporadische Task eine Laxity von 0 hat, verpasst er seine Deadline.

Erklären Sie das Adversary Argument.

- Das „Adversary Argument“ sagt aus, dass es generell nicht möglich ist einen optimalen, komplett dynamischen, Online-Scheduler zu erzeugen, solange es gegenseitige Ausschlüsse zwischen periodischen und sporadischen Tasks gibt.

Task	Period / Deadl.	WCET	Type
T1	4 / 4	2	periodic
T2	4 / 1	1	sporadic



- T1 und T2 schließen sich gegenseitig aus.
- Obwohl es eine Lösung gibt, kann der Online-Scheduler diese nicht finden.