TEA
Ein fehlertolerantes dynamisches zeitgesteuertes Protokoll

Jens Chr. Lisner

Inhalt

- Grundlagen
- Das TEA Protokoll
  - Idee
  - Architektur
  - Aufbau und Arbeitsweise des Protokolls
  - Sonstiges
  - Modellbasierte Analyse
- Zusammenfassung und Ausblick

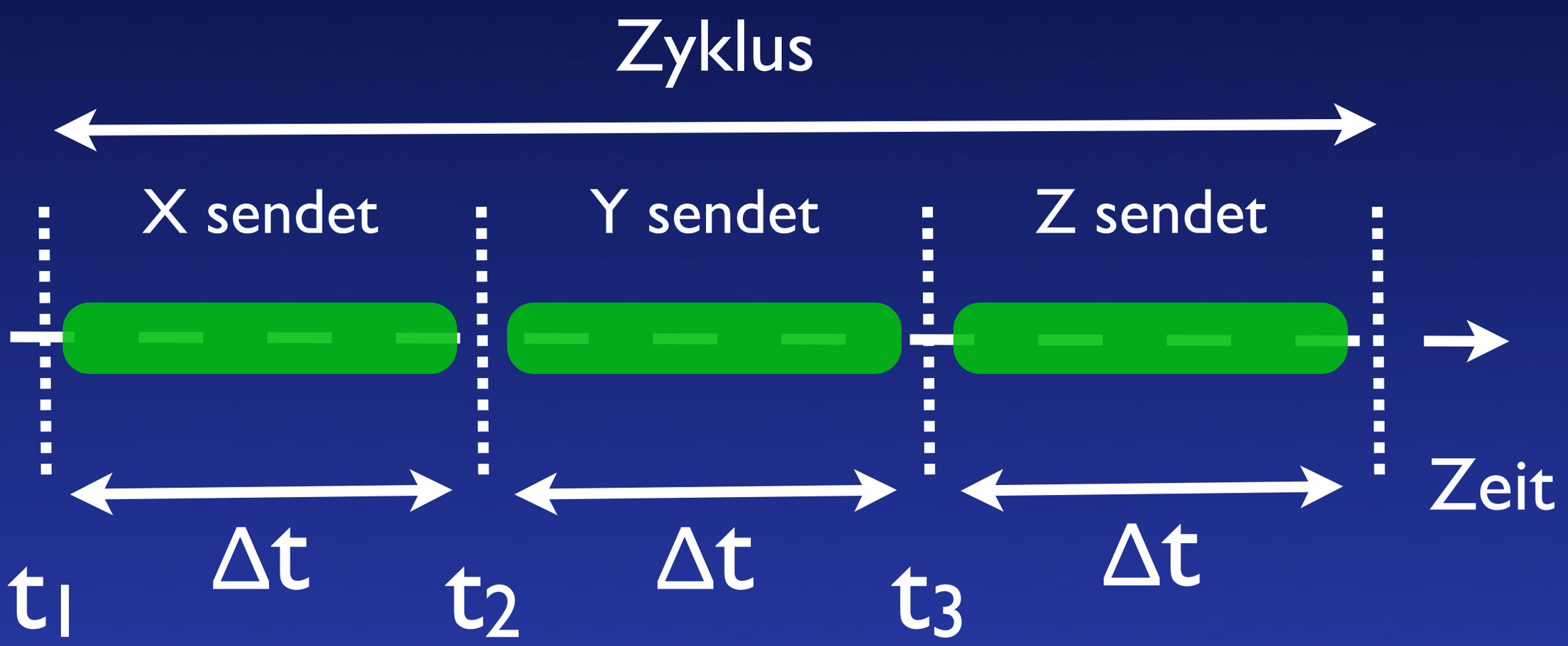
Feldbusprotokolle

- in Automation, Automotive, Avionics und Raumfahrt
- Beispiele: CAN, SAFEbus, FlexRay, byteflight, TT-CAN, ARINC 429, TTP/C
- Einsatz in kritischen Anwendungen
  ⇒ Fehlertoleranzanforderung
- Zeitgesteuerte Protokolle besonders geeignet

Fehlertoleranz

- Fehlertolerantes System
  - fehlertolerante HW-Architektur
  - fehlertolerantes Protokoll
    (hier: ISO/OSI Layer 2 und 3)
    - Zeitgesteuerte Arbitrierung (z.B. statisches TDMA, Minislotting)
    - Encoding (z.B. Prüfsummen, CRC)
- Ziel: Fehlerfreie Komponenten sind immer in der Lage zu kommunizieren.

Arbitrierung (statisches TDMA)

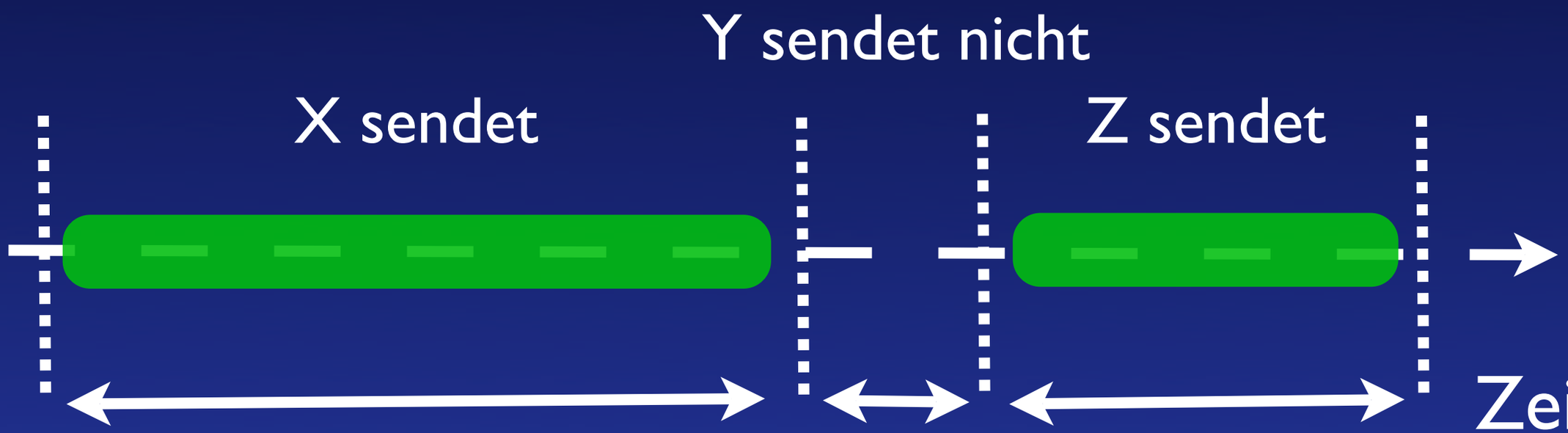


Jeder Sender muß senden!

Arbitrierung (statisches TDMA)

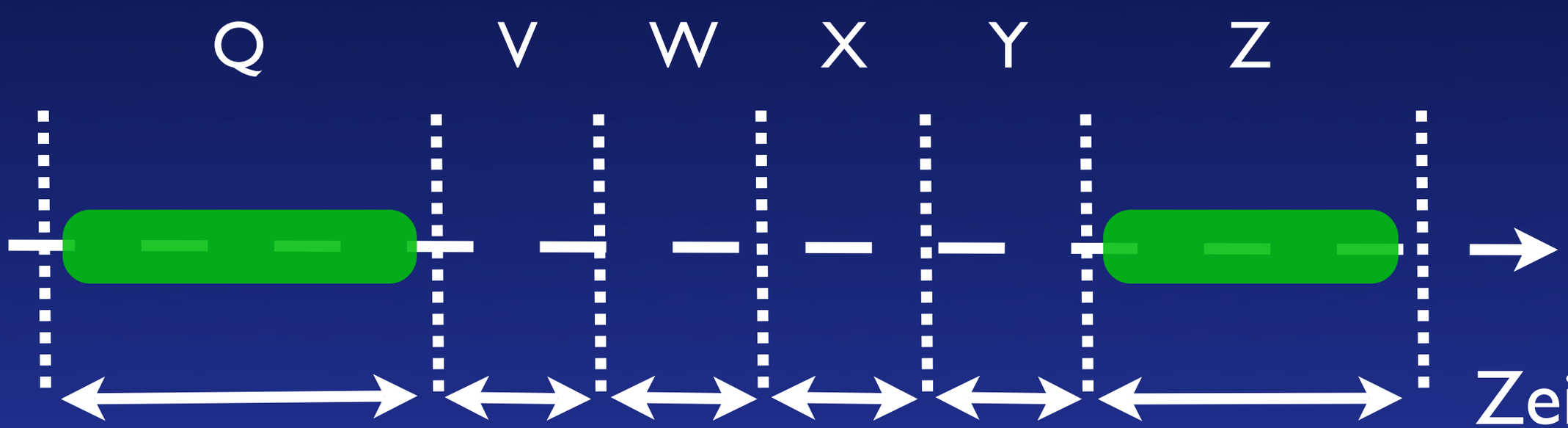
- Vorteile:
  - Deterministisch
  - Senden in eigenem Slot immer möglich
- Nachteile:
  - Sendereihenfolge statisch
  - Nachrichtenlänge statisch
  - Nachrichten müssen gesendet werden, auch wenn keine Daten zu übertragen sind
- Beispiele: TTP/C, FlexRay (statisches Segment)

Arbitrierung (Minislotting)



Δt_X Δt_{min} Δt_Z

Arbitrierung (Minislotting)



Δt_Q Δt_{min} Δt_{min} Δt_{min} Δt_{min} Δt_Z

„Verlorene“ Bandbreite: 4 Δt_{min}

Arbitrierung (Minislotting)

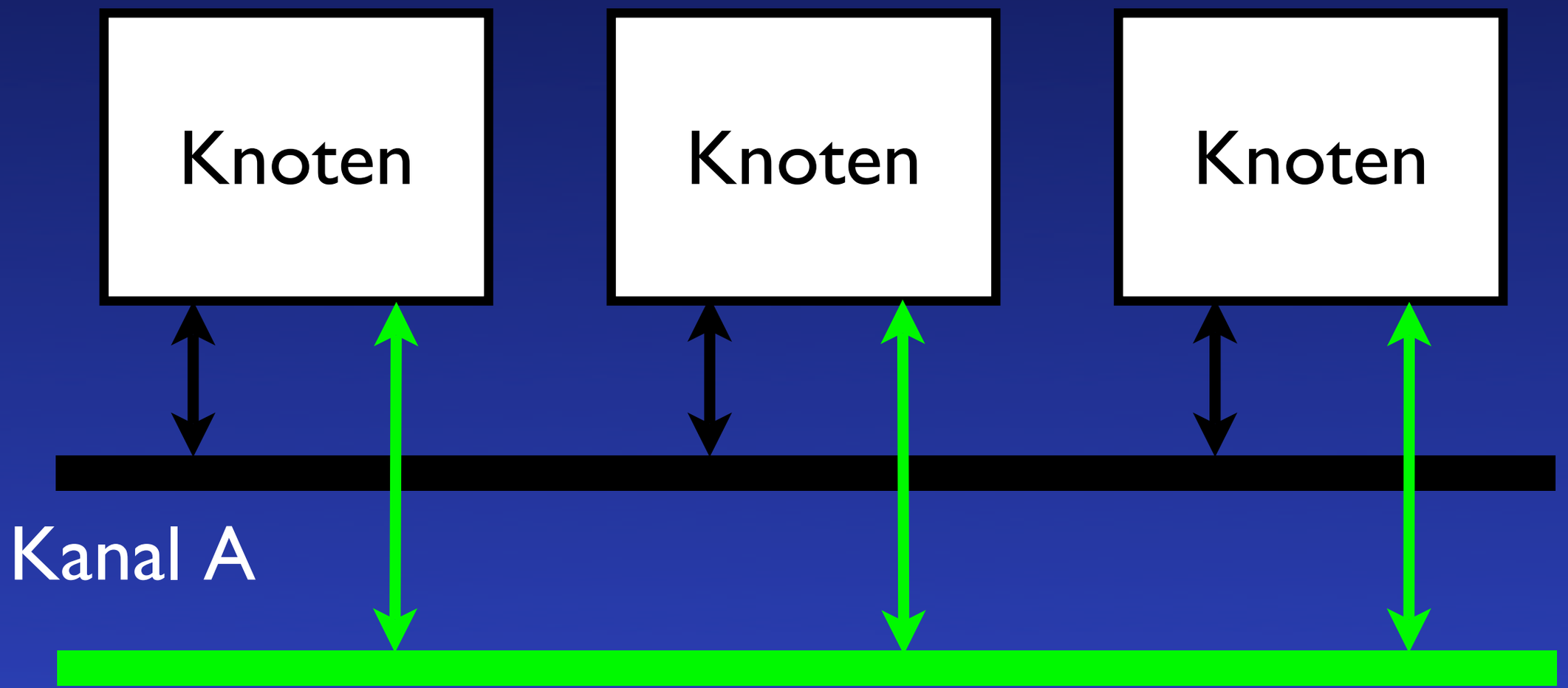
- Vorteile:
  - Reihenfolge der Zugriffsrechte bekannt
  - Nachrichtenlänge dynamisch
- Nachteile:
  - Sendezeitpunkt und -länge unbekannt
  - ggf. hoher Bandbreitenverlust
- Beispiele: byteflight, ARINC 459

Architektur: Kanal

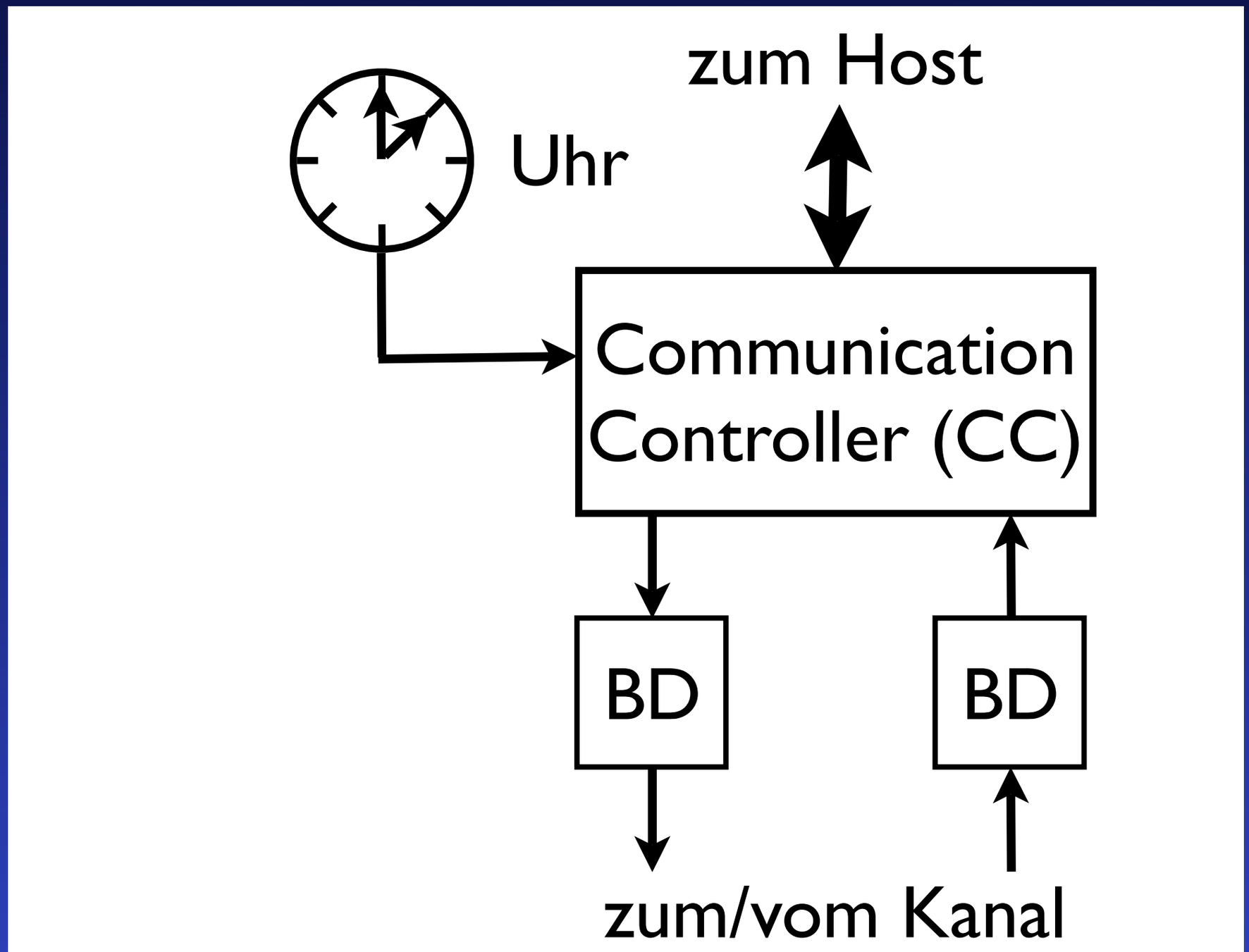
- Topologie: Bus, Stern, gemischt
- Kanalfehler:
  - Nachrichtenverlust
  - Nachrichtenverfälschung/-zerstörung (erkennbar mit hinreichend guten Prüfsummenverfahren)
  - byzantinischer Kanalfehler:
    Wenn K die Menge der Knoten ist, dann empfangen
    $E \subset K$ die Nachricht fehlerfrei, und
    $K - E$ die Nachricht fehlerhaft

Architektur: Kanal

Maßnahme gegen Kanalfehler:
Nachrichten werden auf zwei Kanälen übertragen



Architektur: Knoten



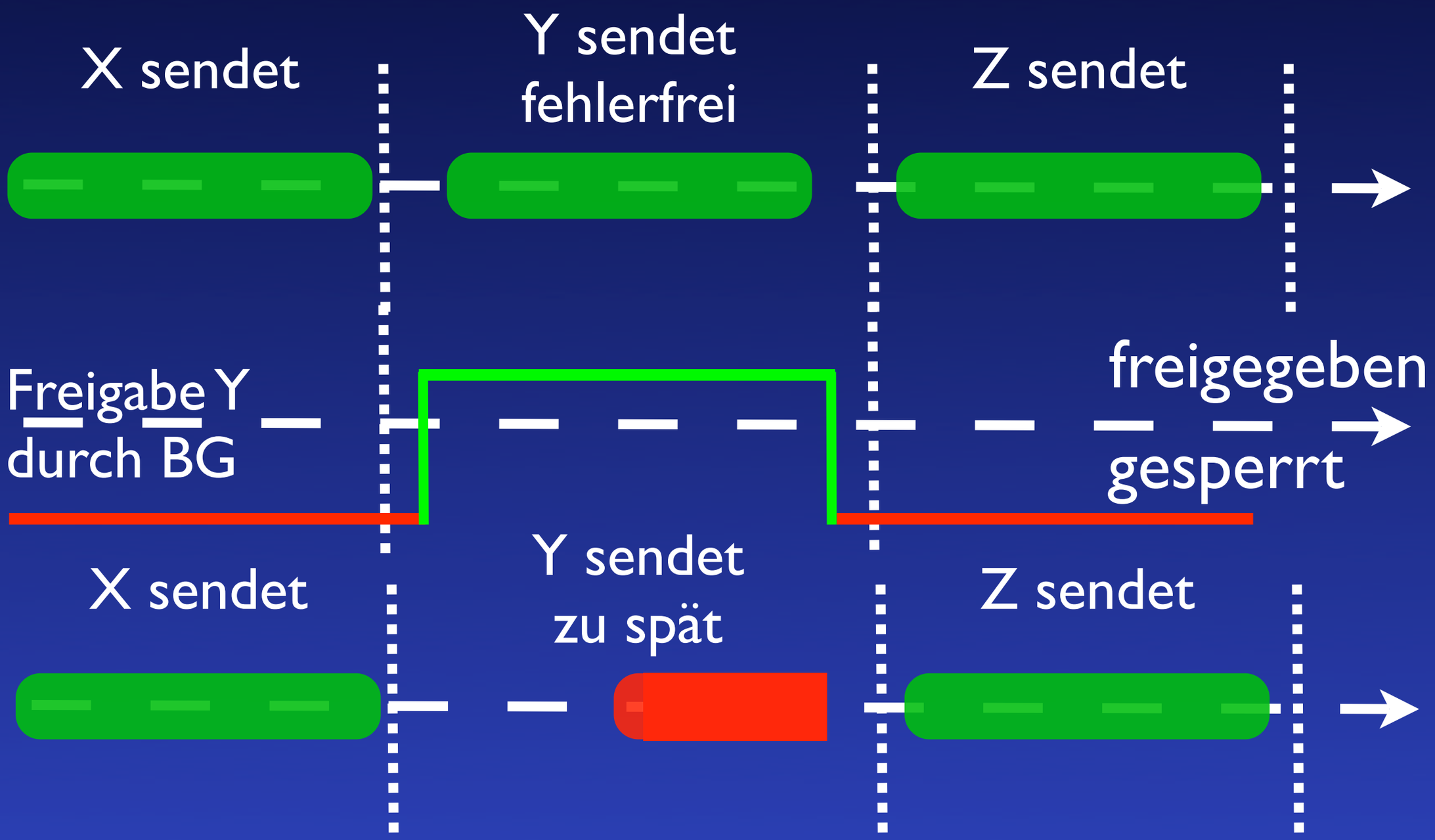
Architektur: Controller

- Aufgaben (z.B.):
  - Senden/Empfangen von Nachrichten
  - Kodieren/Dekodieren von Nachrichten
  - Scheduling
  - Uhrensynchronisation
  - Interface zum Host

Architektur: Controller

- Mögliche Fehler:
  - Senden falsch codierter Nachrichten
  - Senden von Nachrichten mit „falschem“ Inhalt (i.A. nicht erkennbar)
  - Nachrichtenausfall
  - Nachricht zum falschen Zeitpunkt gesendet
    ⇒ Kollisionen möglich

Architektur: Knoten



Nachricht wird abgeschnitten ⇒ keine Kollisionen

Architektur: Guardian

- Dezentrale Bus Guardians
  - unabhängig von Controller (eigene Uhr, eigene Synchronisation) ⇒ teuer
  - abhängig vom Controller ⇒ billig, aber nicht fehlertolerant
- Zentraler Bus Guardian
  - An jedem Eingang eines Sternkopplers ein Bus Guardian
  - Eigener Controller zur Steuerung
  - ⇒ nur Stern-Topologie

Das TEA-Protokoll

- Neues Zeitgesteuertes Protokoll mit statischen und dynamischen Anteilen
- Fehlertolerant (Doppelfehler)
- Bandbreitenverlust minimal
- Nachrichten dynamischer Länge
- Verschiedene Schedulingalgorithmen
- Formale Beschreibung
  - Fehlerfortpflanzung
  - Struktur/Timing
- Korrektheit Beweisbar

Das TEA-Protokoll

Fehlerannahme

- Knotenfehler
  (Ausfallfehler, fehlerhaftes Encoding, fehlerhafter Nachrichteninhalt, Timingfehler, nicht byzantinisch)
- Kanalfehler
  (Ausfallfehler, „Message corruption," byzantinisch)
- Knoten & Kanalfehler (Doppelfehler)
  (alle Kombinationen)

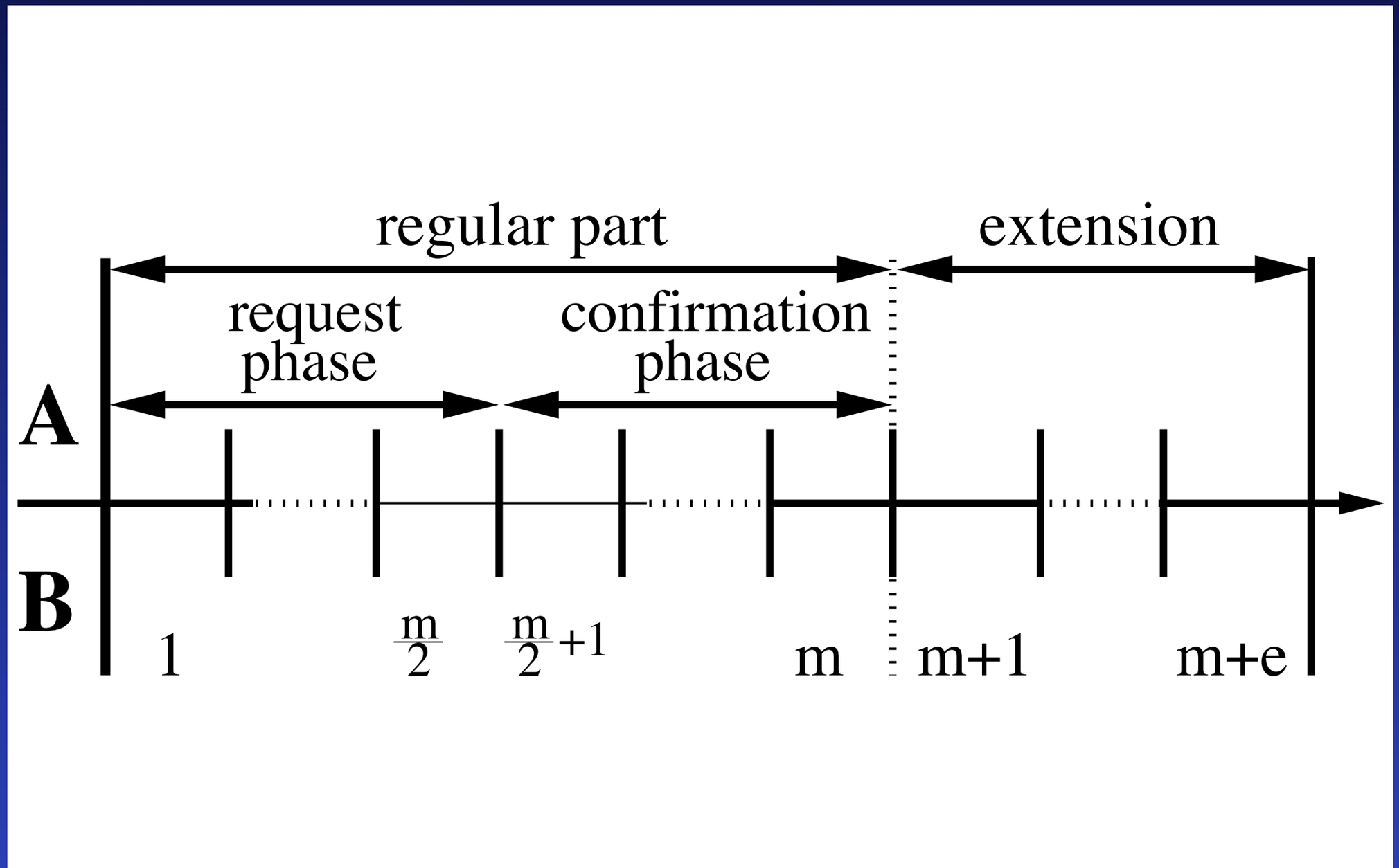
Das TEA-Protokoll

Idee

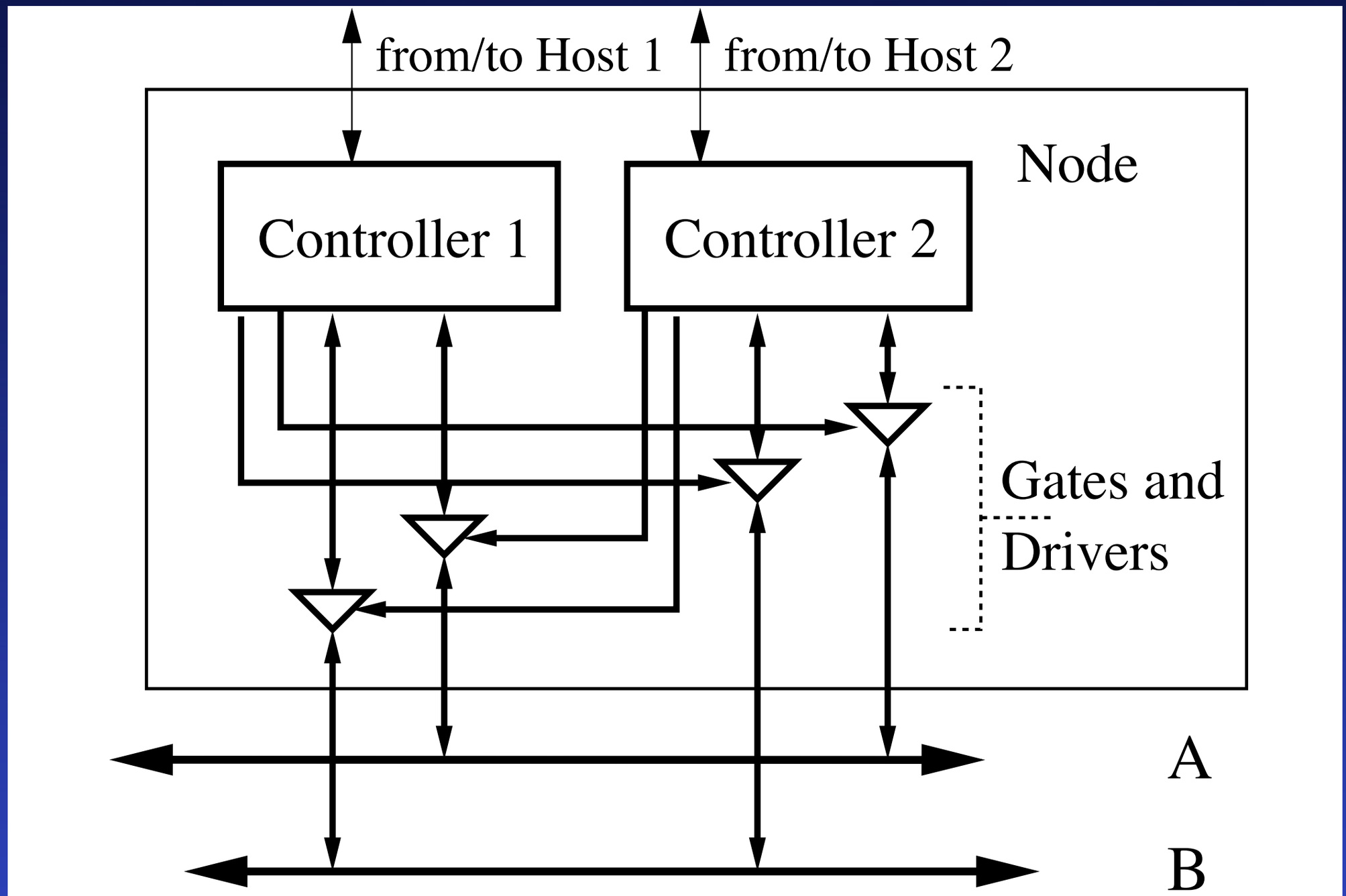
- (Zeitgesteuerter) Zyklus wird geteilt
  - Regular Part (statisch)
  - Extension Part (dynamisch)
- Jeder Sender kann Slot für Extension Part anfordern
- Schedule wird unmittelbar vor Extension Part festgelegt (Agreement Algorithm)
- Zwei Controller überwachen sich gegenseitig

Time-triggered Efficiency by Agreement

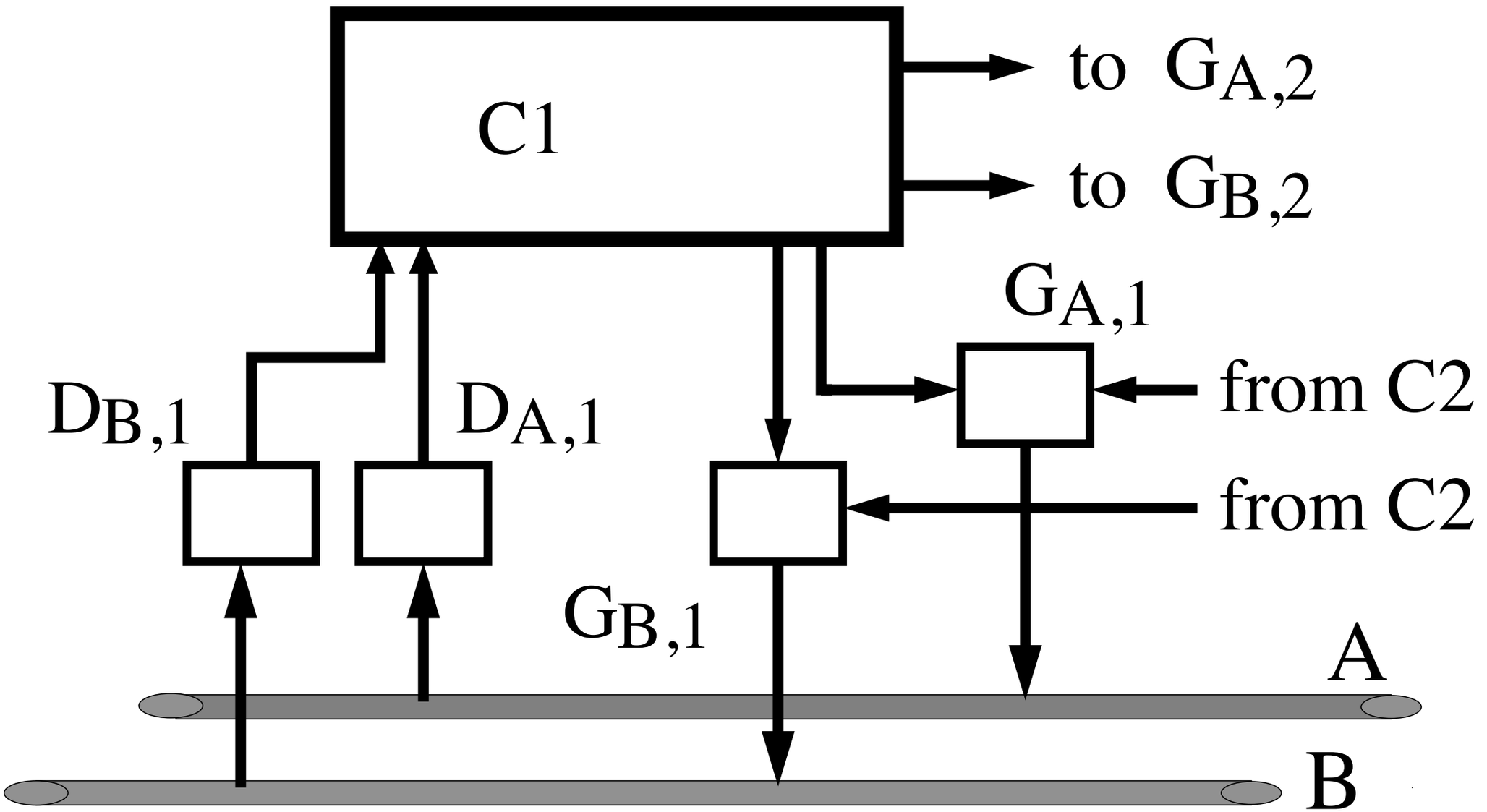
Das TEA-Protokoll



TEA Architektur: Knoten



TEA Architektur: Knoten

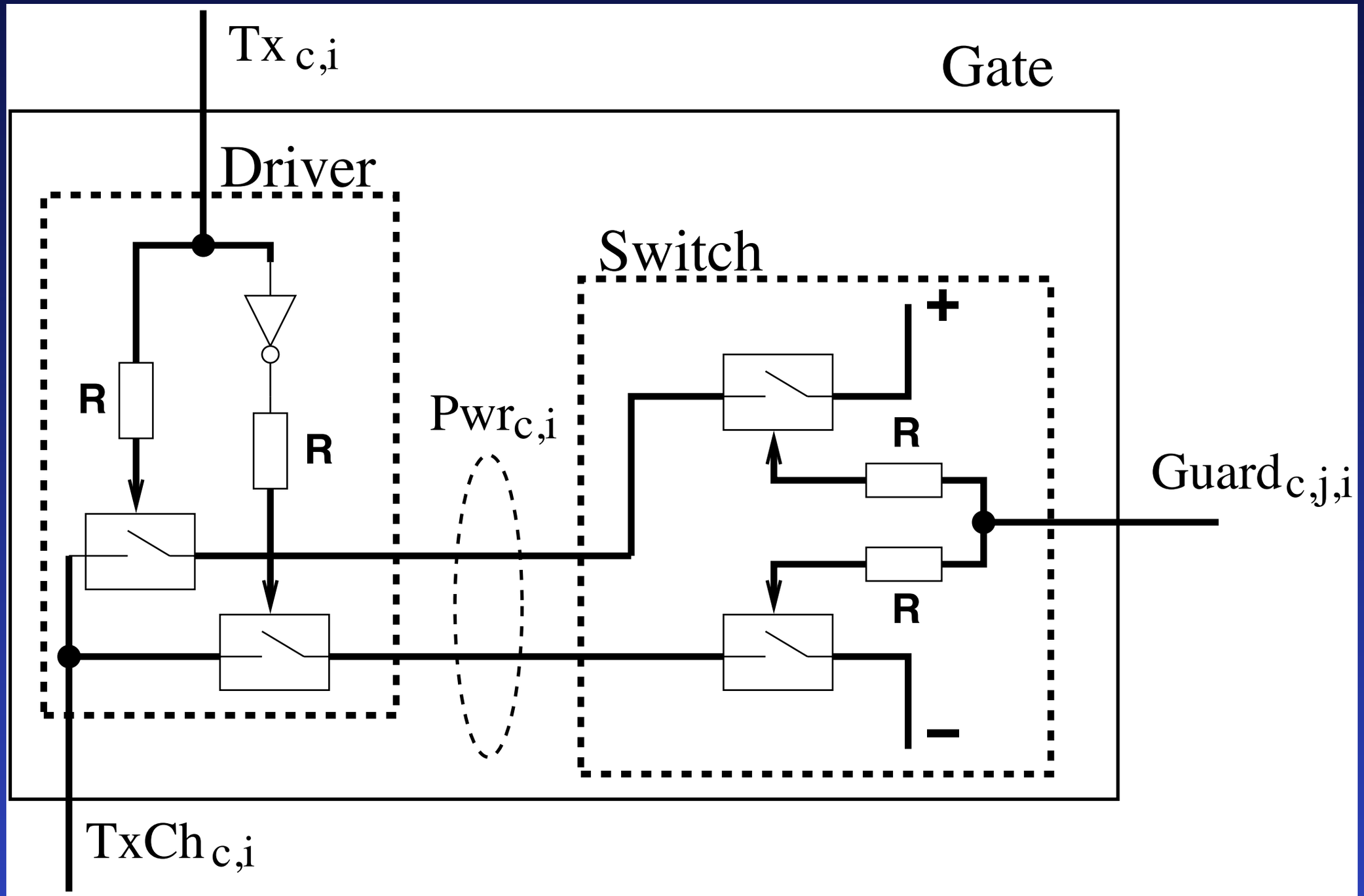


TEA Architektur: Knoten

- Dezentral
- Jede Topologie möglich
- Vollzieht Schedule für Extension Part nach
- Unabhängig

- Nachteil:
  Fehlerhafter Controller kann Nachbarn am senden hindern.

Gate



TEA: Regular Part

- Statisches TDMA
- Aufgeteilt in
  - Request Phase
  - Confirmation Phase

- Beide Phasen haben gleiche Länge
- Ergebnis: Schedule für Extension Part
- Jeder Sender muß mind. einmal in jeder Phase senden

TEA: Regular Part (Scheduling)

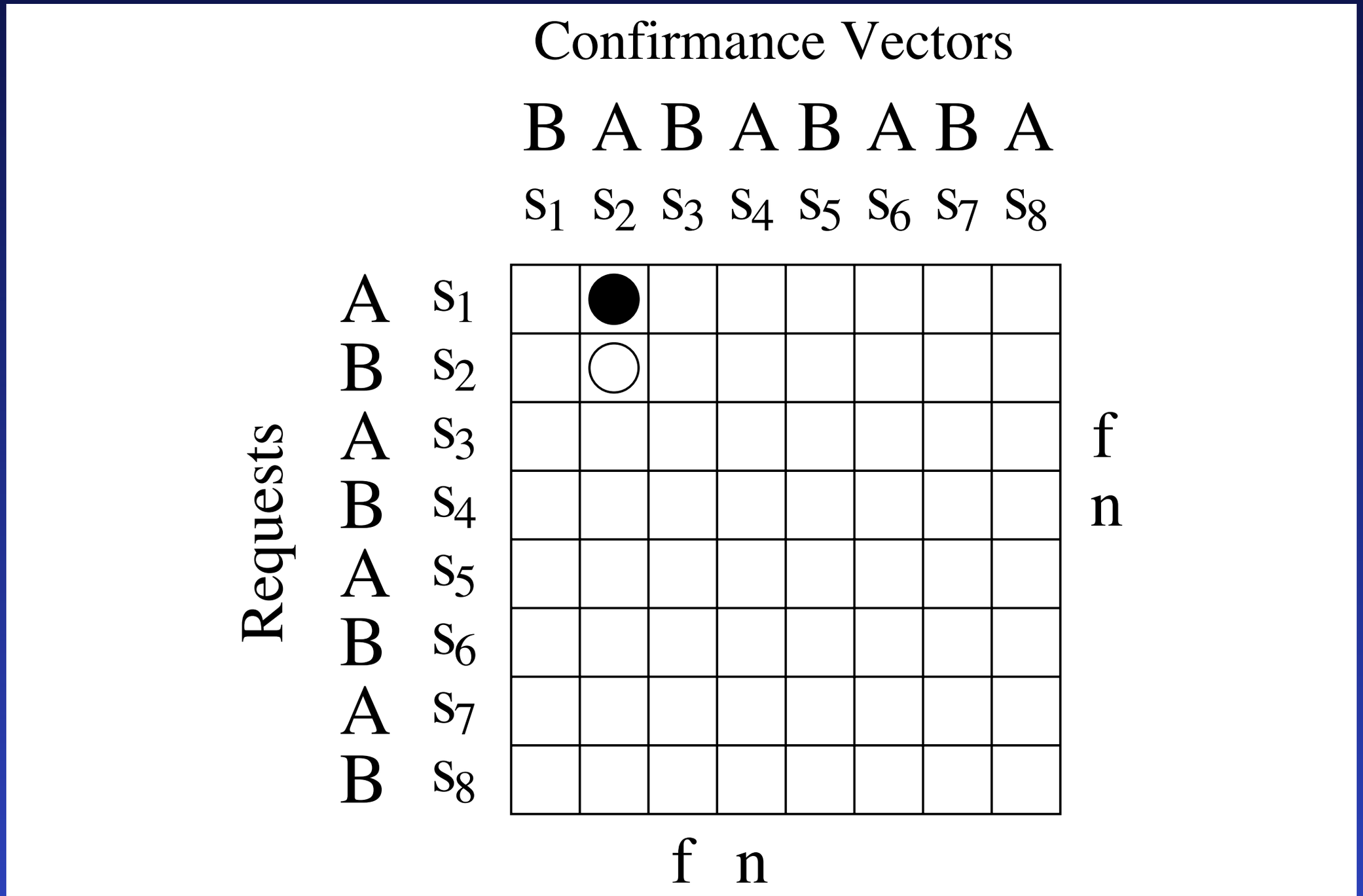
Request Phase	Slot 1	Slot 2	Slot 3	Slot 4
Kanal A	S1	S3	S5	S7
Kanal B	S2	S4	S6	S8

Confirm. Phase	Slot 5	Slot 6	Slot 7	Slot 8
Kanal A	S2	S4	S6	S8
Kanal B	S1	S3	S5	S7

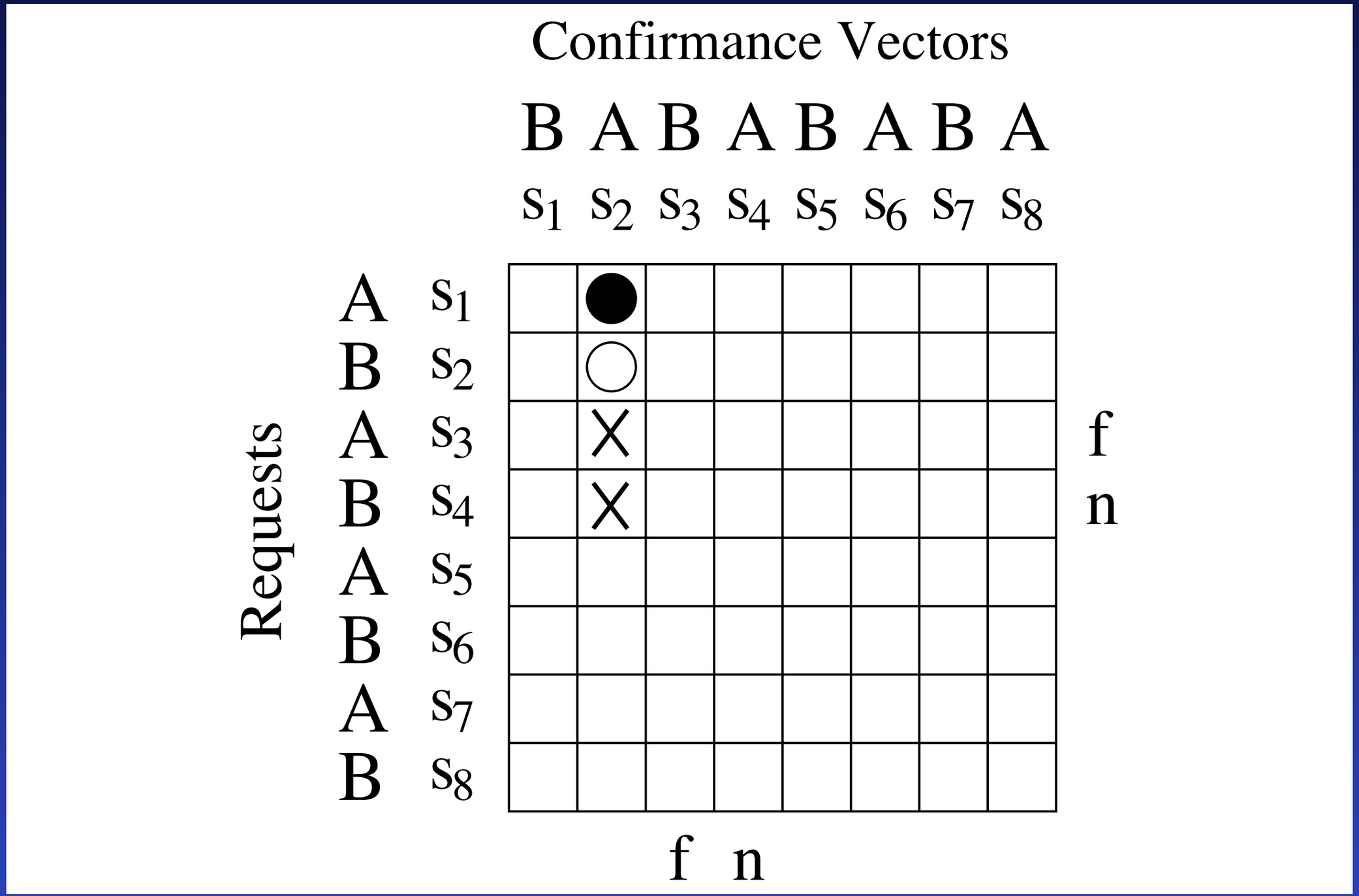
TEA: Request Phase

- Request Phase
  - Jede Nachricht enthält ein „request bit.“
  - Requests werden gesammelt
- Confirmation Phase
  - Jede Nachricht enthält einen „confirmation vector“
  - 3 mögliche Werte: request, no request, unknown (im Fehlerfall)
  - „request matrix“ wird aufgebaut

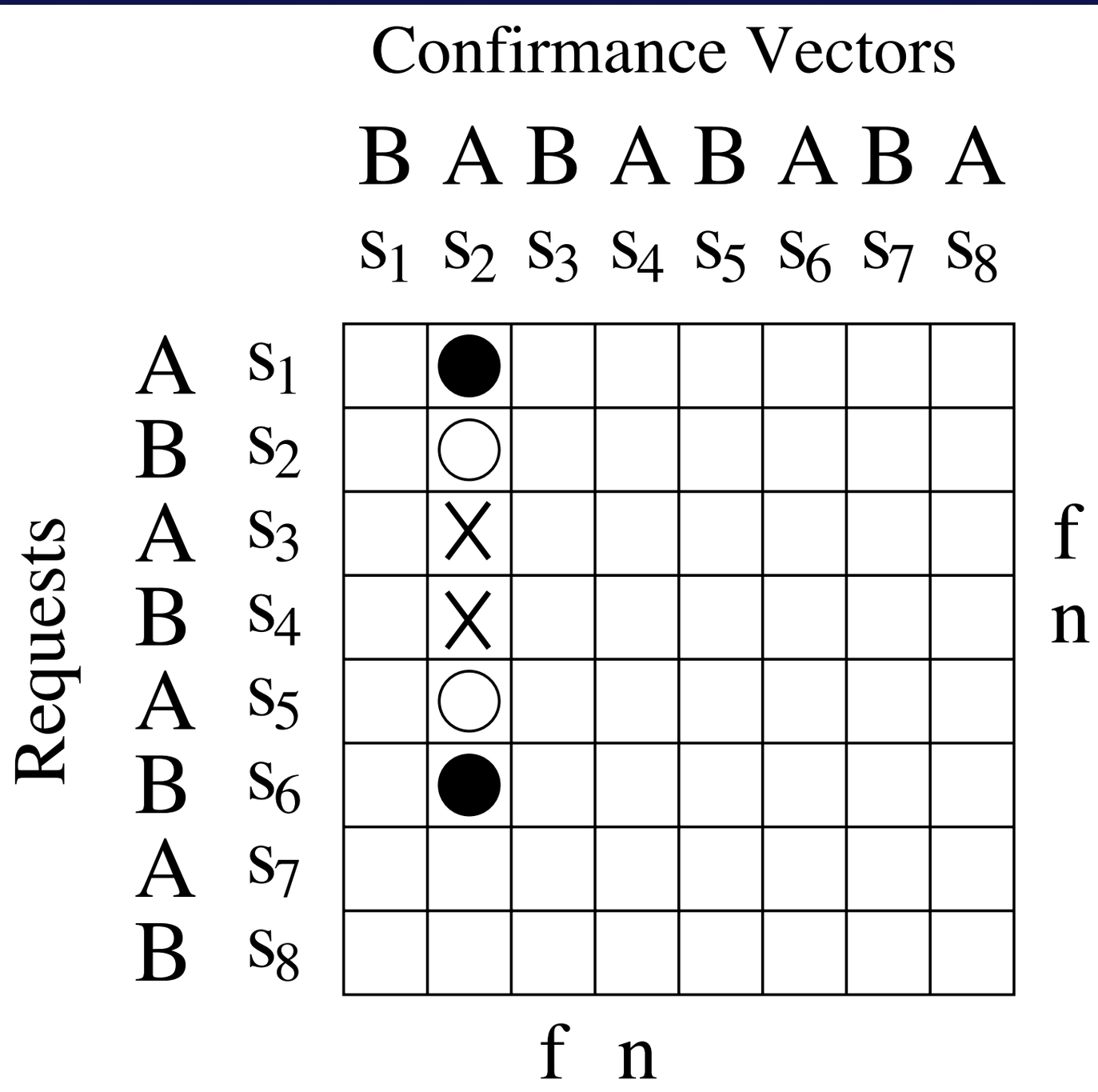
TEA: Request Phase (Slot 1)



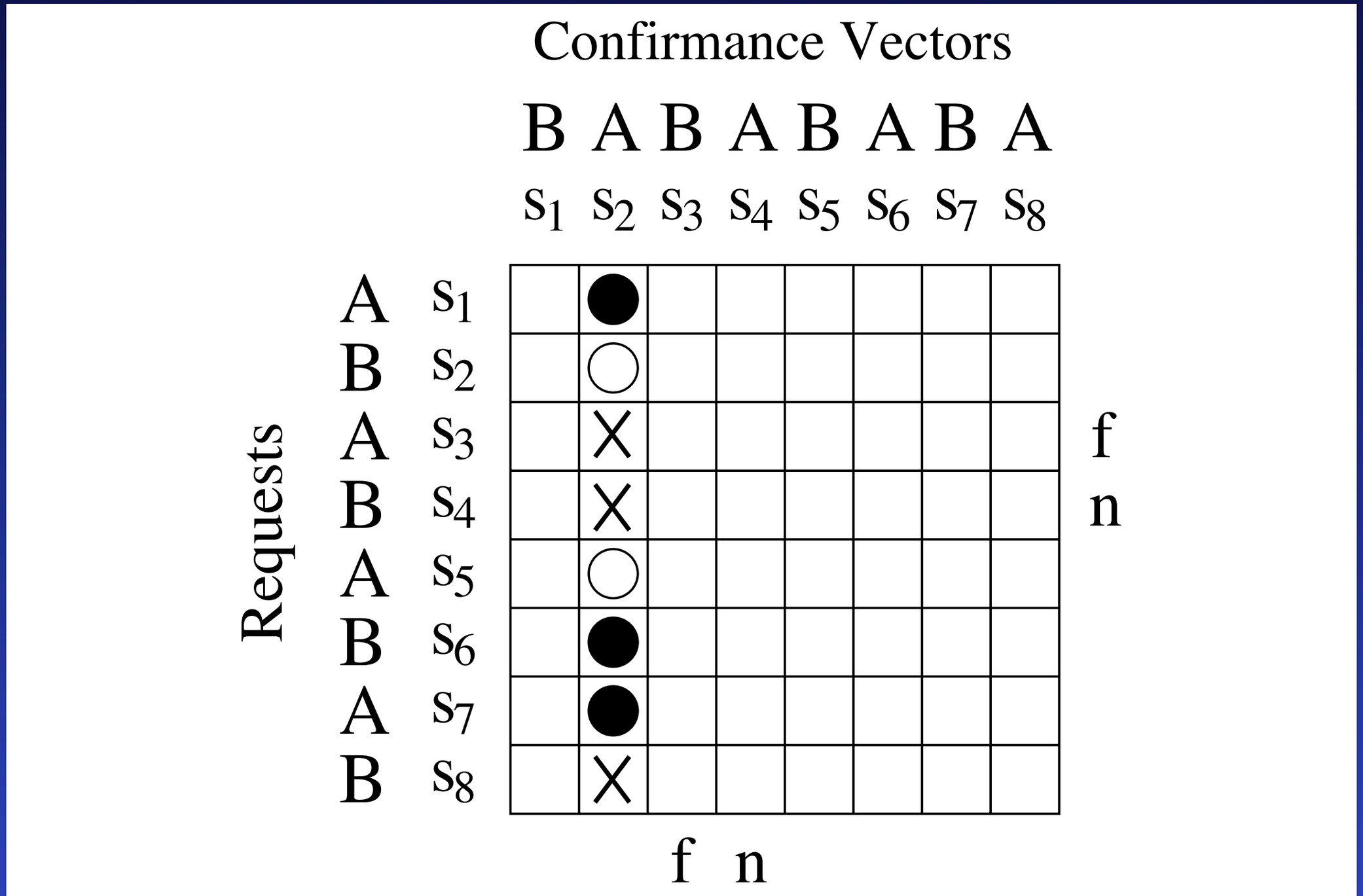
TEA: Request Phase (Slot 2)



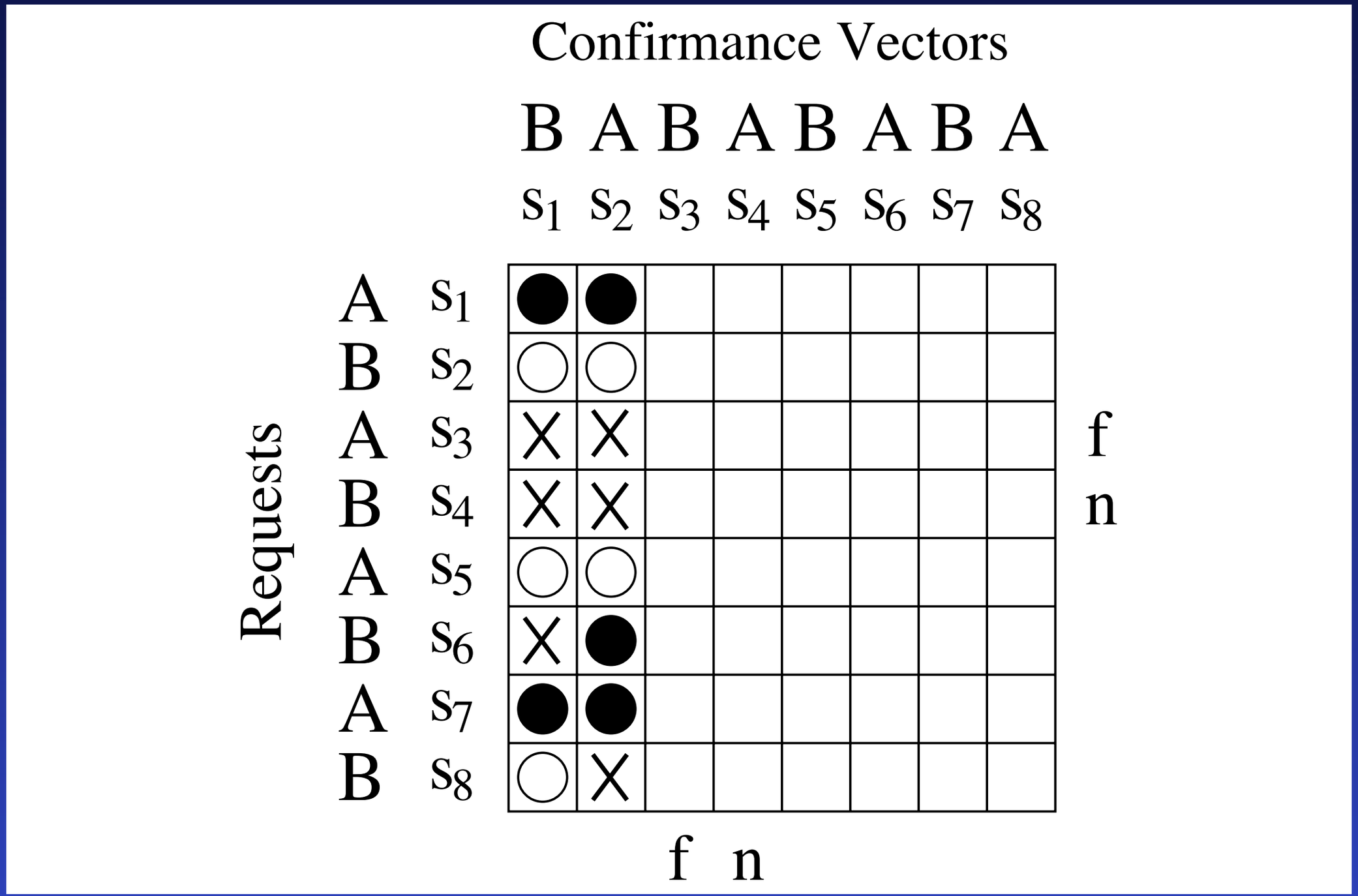
TEA: Request Phase (Slot 3)



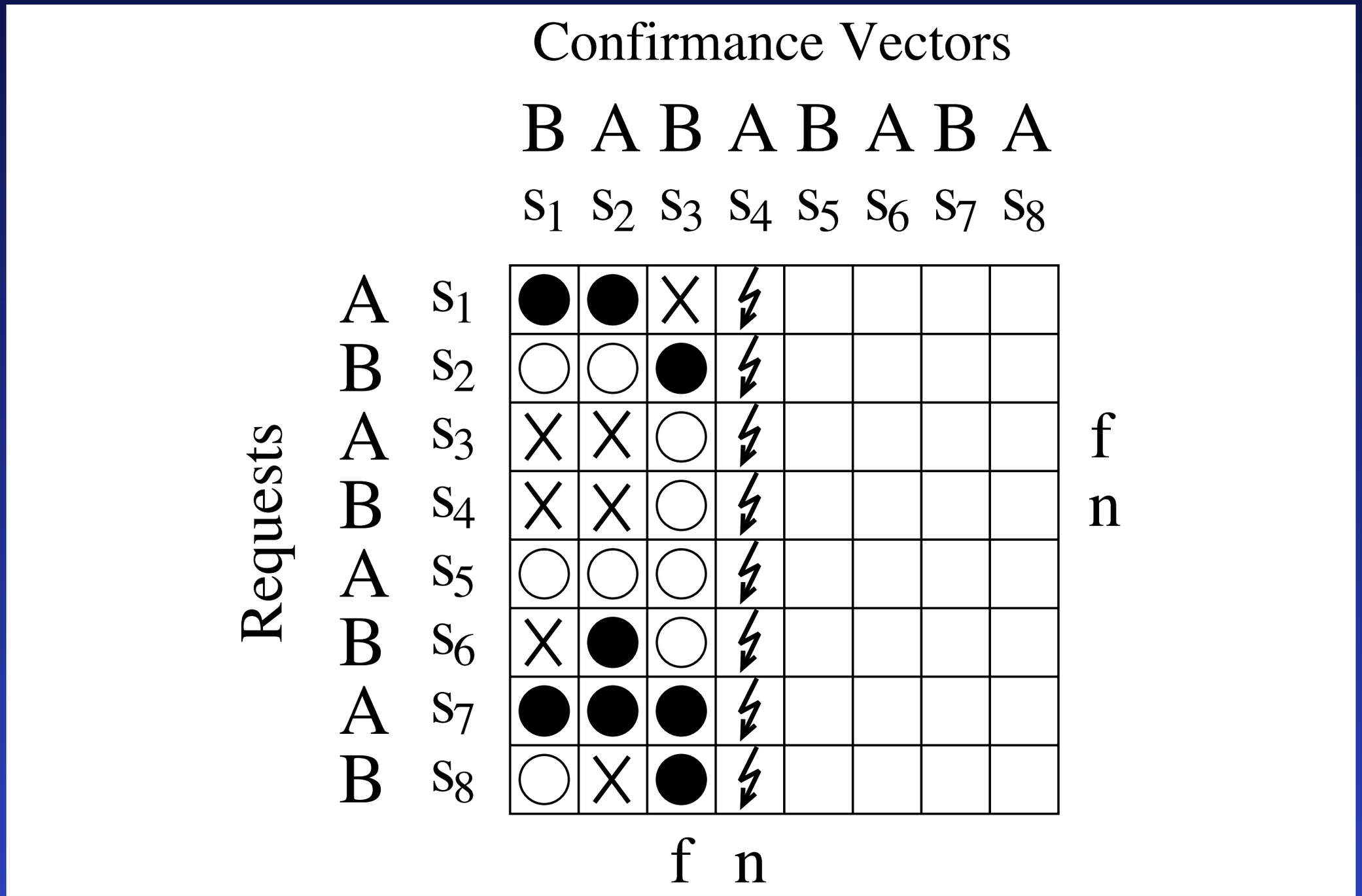
TEA: Request Phase (Slot 4)



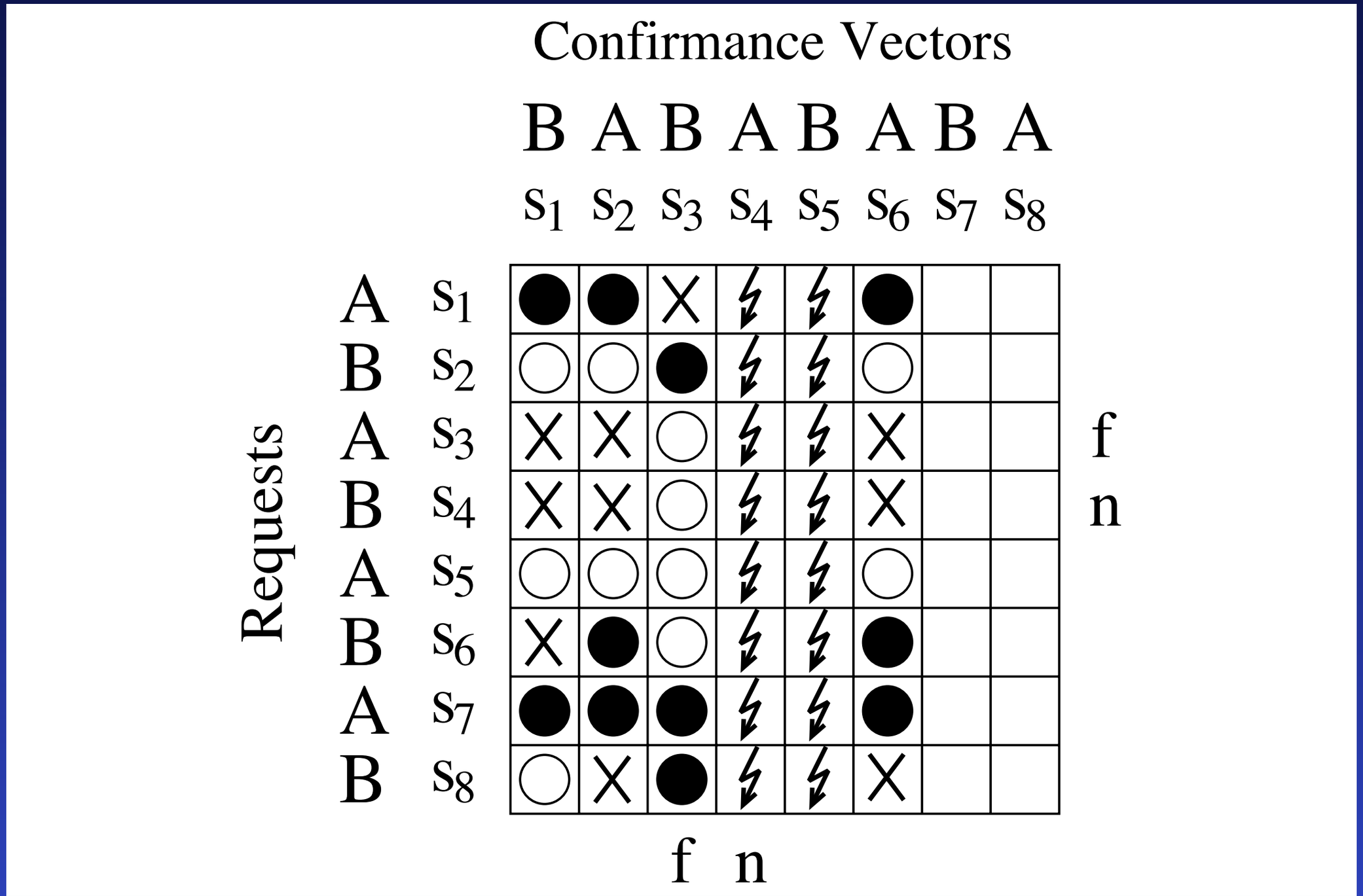
TEA: Confirmation Phase (Slot 5)



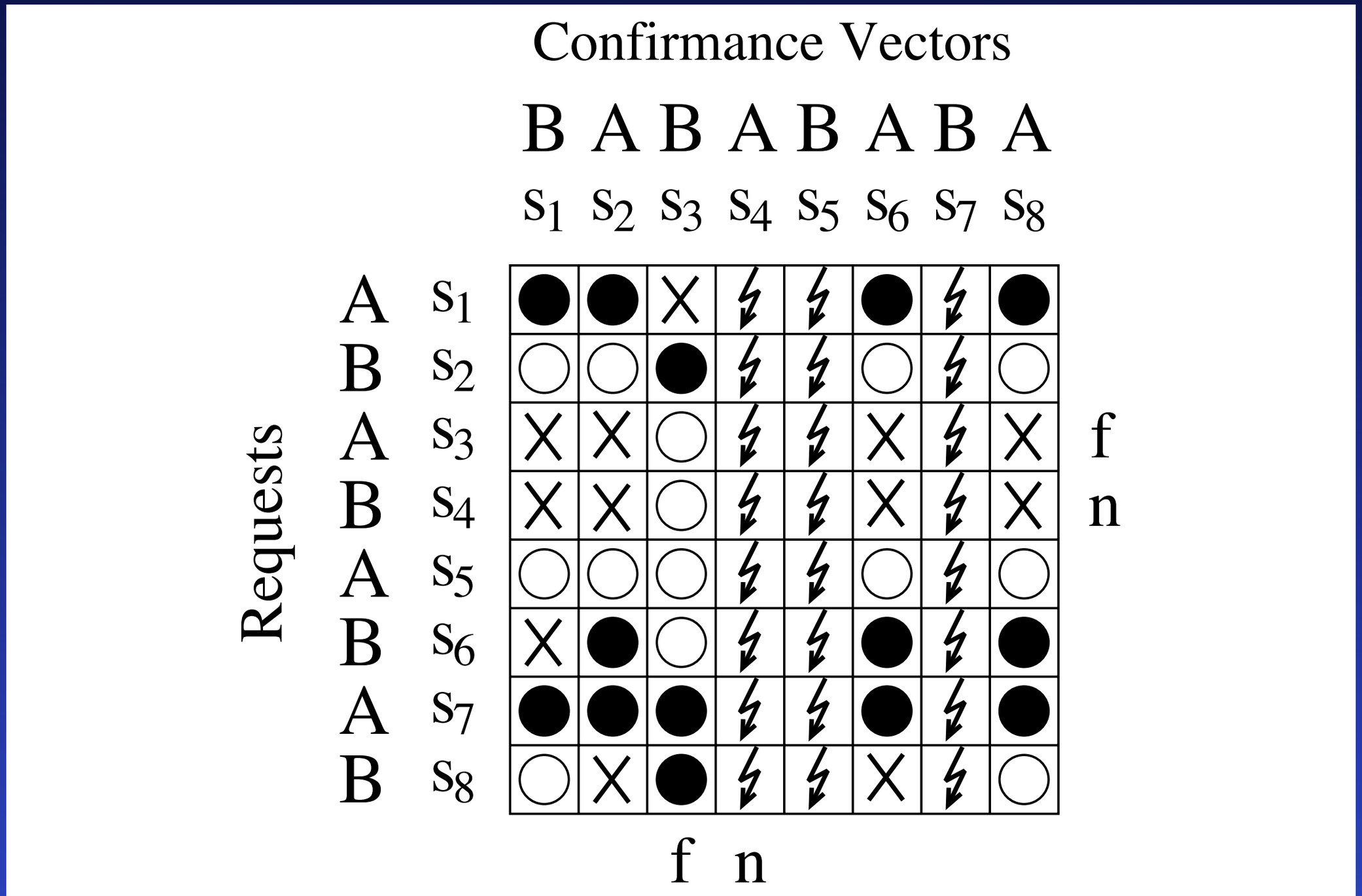
TEA: Confirmation Phase (Slot 6)



TEA: Confirmation Phase (Slot 7)



TEA: Confirmation Phase (Slot 8)



TEA: Agreement Algorithm

- Ergebnisvektor wird gebildet
- Default-Wert falls Wert nicht eindeutig
- Korrektheit formal bewiesen
- Effizient: Übereinstimmung wird innerhalb eines Zyklus erreicht
- Toleriert Doppelfehler

Sonstiges

- Scheduling im Extension Part
  - FIFO
  - Priority
  - FIFO-first
  - Priority first
- Slots dynamischer Länge
- Fehlertolerant!

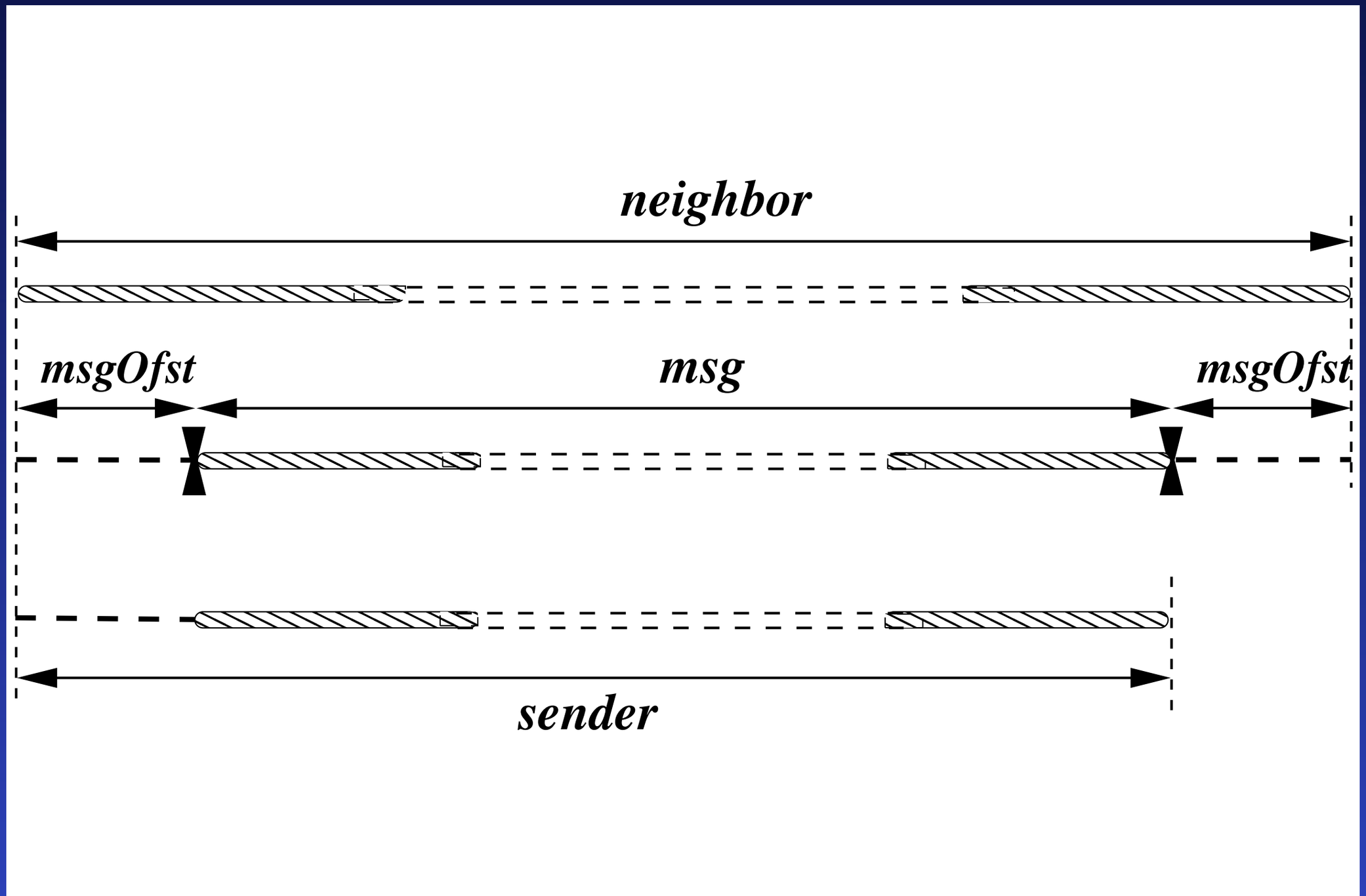
Network Behavior Functions

- Methode zur formalen Beschreibung der Fehler-fortpflanzung in einem System
- System dargestellt als gerichteter Graph
- Berücksichtigt:
  - Verhalten einer Komponente
    - Verhalten an den Eingängen in die Komponente
  - Verhalten ist Zeitabhängig
  - Automatisierbar

Timed Grammars

- Protokollablauf ist aus Aktionen aufgebaut
- Verschiedene Aktionen können verknüpft werden (Sequenz, Parallel, ...)
- Struktur des Systems: Darstellung als Grammatik
- Zu einer Aktion gehören Offset und Dauer
- Die Länge von Offset und Dauer sind „ungenau“ (z.B. Gangunterschied von Uhren!)
- Zu jedem Operator gibt es eine Rechenvorschrift
- Timings von Aktionen lassen sich automatisiert berechnen

Timed Grammars



Zusammenfassung

- ✓ Zeitgesteuertes Protokoll mit statischen und dynamischen Anteilen
- ✓ Fehlertolerant (Doppelfehler)
- ✓ Bandbreitenverlust minimal
- ✓ Nachrichten dynamischer Länge
- ✓ Verschiedene Schedulingalgorithmen
- ✓ Formale Beschreibung
  - Network Behavior Functions
  - Timed Grammars
- ✓ Korrektheit bewiesen

Zusammenfassung

TEA:
Erstes zeitgesteuertes, voll fehlertolerantes Protokoll mit dynamischen Anteilen!

Ausblick

- Weitere Operationsmodi (z.B. Startup)
- Aufbau in Hardware
- Weiterentwicklung formaler Methoden
- Erweiterung der Fehlerannahme
  (z.B. byzantinische Controllerfehler)
- Anfragen mehrerer Slots

Fragen ?

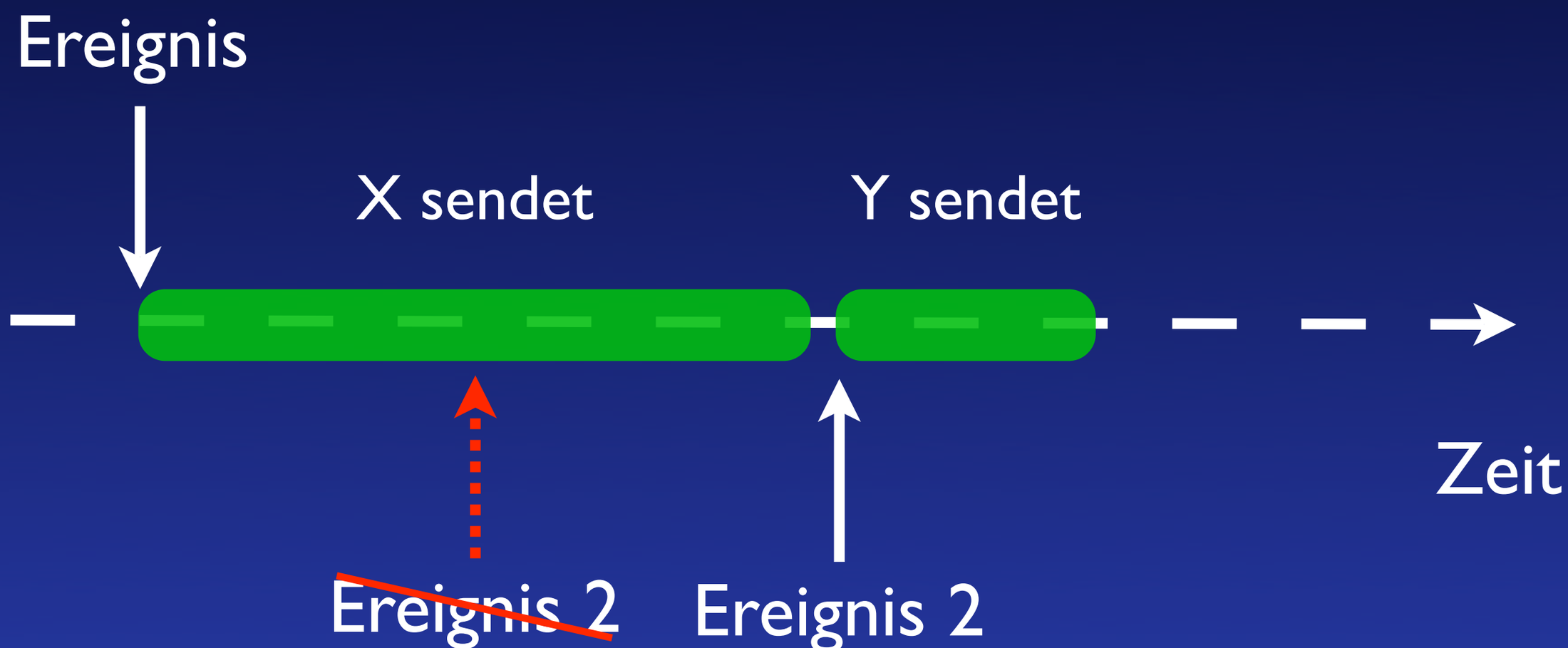
Echtzeitprotokolle

- in Automation, Automotive, Avionics und Raumfahrt
- Echtzeiteigenschaft:
  Jede Aktion muß bis zu einem bestimmten Zeitpunkt abgeschlossen sein.
- Einsatz in kritischen Anwendungen
  ⇒ Fehlertoleranzanforderung

Echtzeitprotokolle

- nicht fehlertolerant:
  - CAN (Automatisierung, Automotive)
  - byteflight (Automotive)
  - FlexRay (Automotive - mit dyn. Segment)
- fehlertolerant:
  - TTP/C (Automotive, Avionics)
  - FlexRay (Automotive - ohne dyn. Segment)

Arbitrierung (Ereignisgesteuert)



Beispiel: CAN

Arbitrierung (Ereignisgesteuert)

- Vorteile:
  - Zeitunabhängig
  - Kanalzugriff falls notwendig
  - Nachrichtenlänge dynamisch
- Nachteile:
  - Sendezeitpunkt unbekannt
  - Kanalzugriff nicht immer sofort möglich
- Beispiel: CAN

Fehlertoleranz

- Komponenten:
  - Kanal
  - Knoten (Controller, Uhren, ...)
- Typische Fehler:
  - Nachrichtenkollision
  - Nachrichtenverlust
  - Verfälschung der Nachricht
  - Senden der Nachricht zum falschen Zeitpunkt

TEA: Request Phase

- Jeder Slot wird durch zwei Sender belegt (Kanal A und B)
- Nachbarn senden auf verschieden Kanälen
- Jede Nachricht enthält ein „request bit.“
- Jeder Empfänger sammelt die Requests der anderen Controller

TEA: Confirmation Phase

- Zuordnung zu Kanälen wird vertauscht.
- Jede Nachricht enthält einen „confirmation vector“
- 3 mögliche Werte pro Sender
  - request
  - no request
  - unknown (im Fehlerfall)

TEA Architektur: Gate

- Zerfällt in ...
  - Output Driver
  - Switch
- Switch kann die Spannungsversorgung von Output Driver unterbrechen
- Switch wird vom Nachbarcontroller gesteuert
- Beide Teile gehören in verschiedene Fehlerbereiche

Architektur: Controller

- Maßnahmen gegen Fehler:
  - Nachrichtenverfälschung: Prüfsummen/Encoding
  - Kollisionsvermeidung:
    - dezentrale Bus Guardians
    - zentrale Bus Guardians
  - Doppelcontrollerarchitektur (später)

TEA Architektur

- Doppelkanal
- Stern, Bus oder gemischte Topologie
- Doppelcontroller
  - Jeweils zwei unabhängige Controller pro Knoten
  - Ein Knoten für mehrere Hosts
  - Gate für Zugriffskontrolle

TEA: Agreement Algorithm

Voraussetzungen:

- Mind. 8 Controller

Ziel:

- Bitvector (request, kein request)
- Bitvector gleich für jeden Controller

Timed Grammars

- Protokollablauf ist aus Aktionen aufgebaut
- Aktionen werden aus anderen Aktionen produziert
  - Zyklus besteht aus Regular und Extension Part
  - Regular Part besteht aus Phasen
  - Phase besteht aus Slots
  - etc.
- Verschiedene Aktionen können verknüpft werden (Sequenz, Parallel, ...)

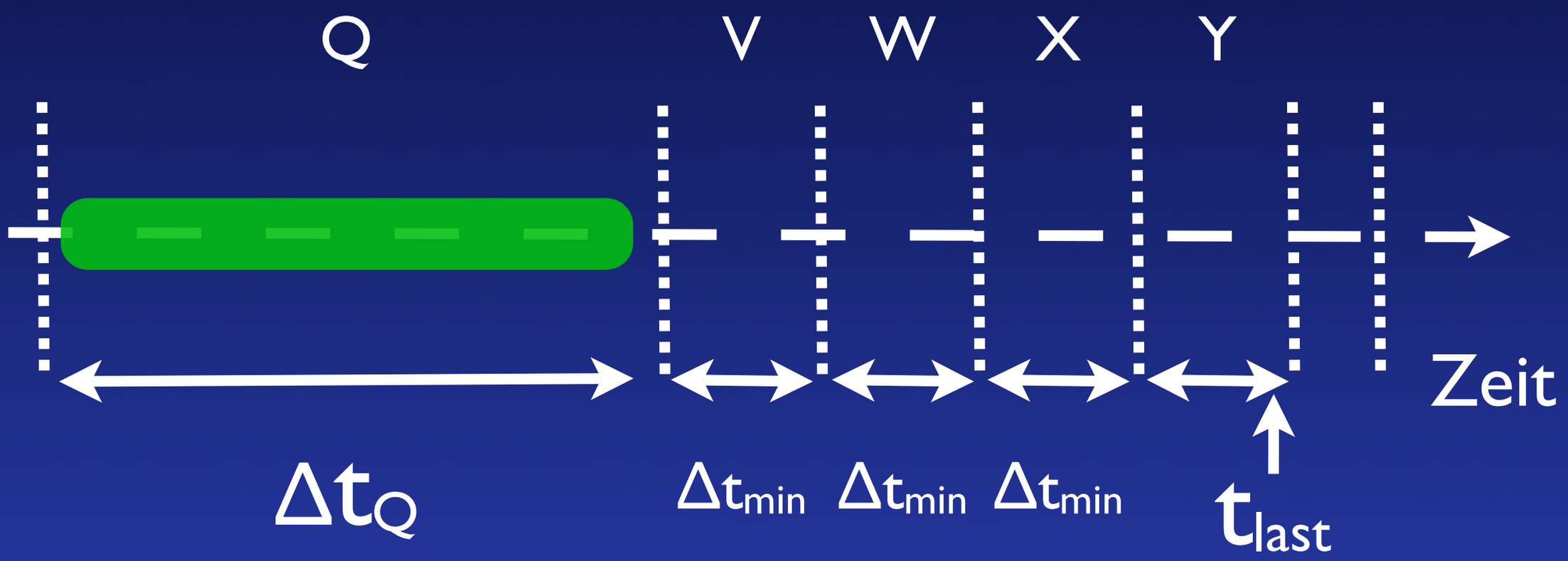
Timed Grammars

- Struktur des Systems: Darstellung als Grammatik
- Hierarchisch
- Mehrere Produktionen pro Aktion möglich
- Zu einer Aktion gehören Offset und Dauer
- Die Länge von Offset und Dauer sind „ungenau“ (z.B. Gangunterschied von Uhren!)
- Zu jedem Operator gibt es eine Rechenvorschrift
- Timings von Aktionen lassen sich automatisiert berechnen

Sonstiges: Slots dynamischer Länge

- Slots dynamischer Länge im Extension Part möglich
- Länge wird aus Längenfeld im Header abgeleitet
- Fehlertolerant
- Verifiziert durch Zustandsraumexploration

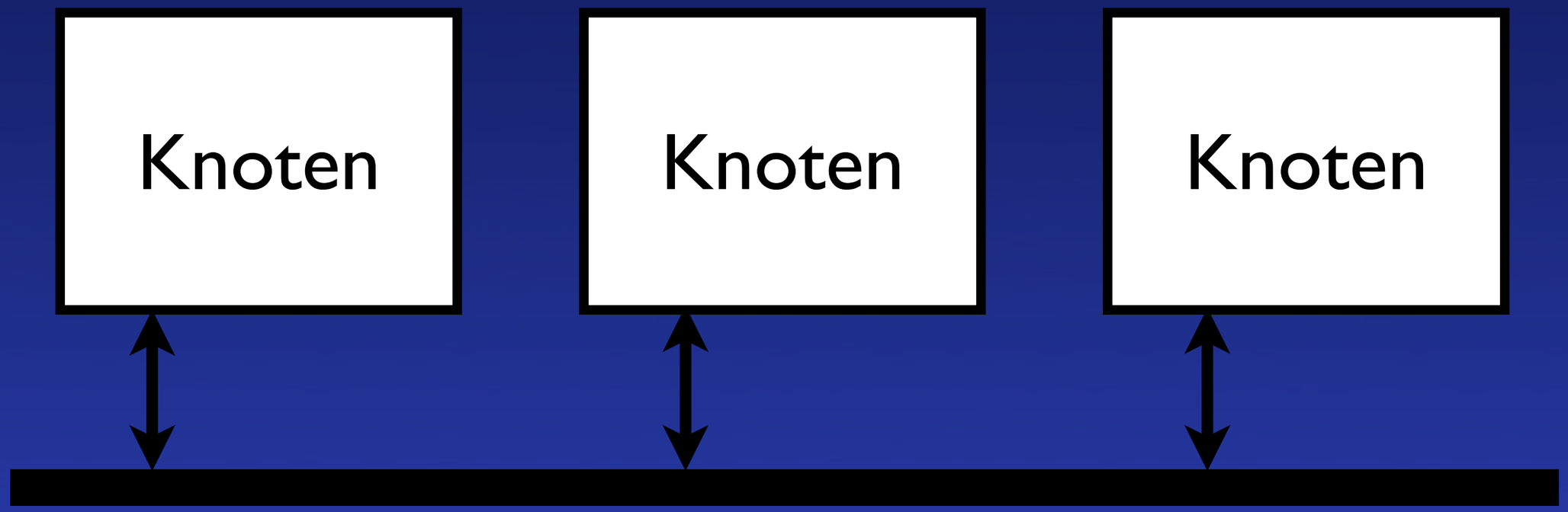
Arbitrierung (Minislotting)



Z kann nicht mehr senden!

Architektur: Kanal

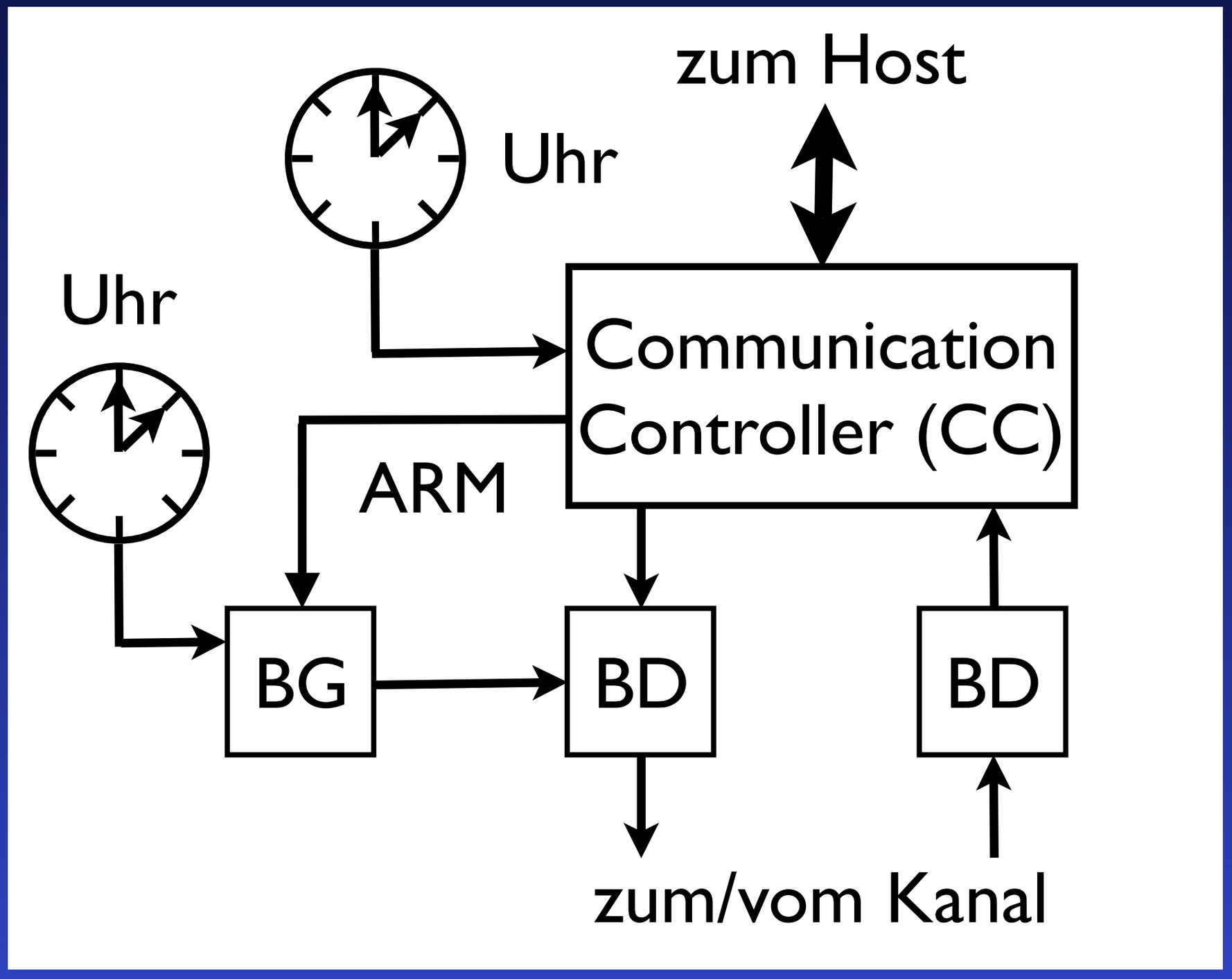
Topologie: Bus, Stern, gemischt



Architektur: Uhr

- Taktgenerator (Quarz)
- Spezifizierte max. erlaubte Abweichung von Nominalfrequenz
  ⇒ Uhrensynchronisation erforderlich
- Uhrensynchronisation im Controller
- Uhren als Teil des Controllers

Architektur: Knoten



Architektur: Driver

- Verbindung zwischen Kanal und Controller
- Zweimal pro Kanal (Hin- und Rückrichtung)
- Mögliche Fehler:
  - Nachrichtenverlust
  - Nachrichtenverfälschung