PTCP / LNP

Verbindungsorientierte Kommunikation
zwischen RCX

Inhalt

- Idee
- Begriffe
- Pakettypen
- Datentypen
- Funktionen
- Interna
- Kommunikation (Beispiele)
- Ausblick

Idee

- Verbindungsorientiert kommunizieren
- Wissen, ob Daten wirklich ankommen
- Wissen, wer mit wem kommuniziert
- Medium möglichst effektiv nutzen

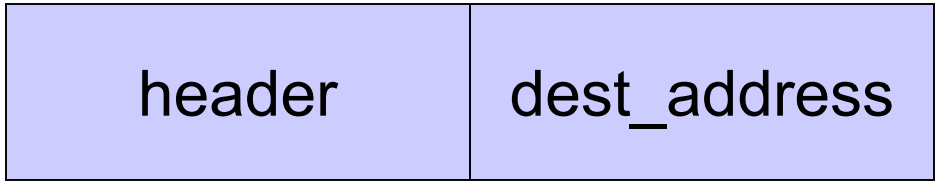
Begriffe

- Kanal
- Verbindung
- Datenpaket (I-Frame)
- U-Frame

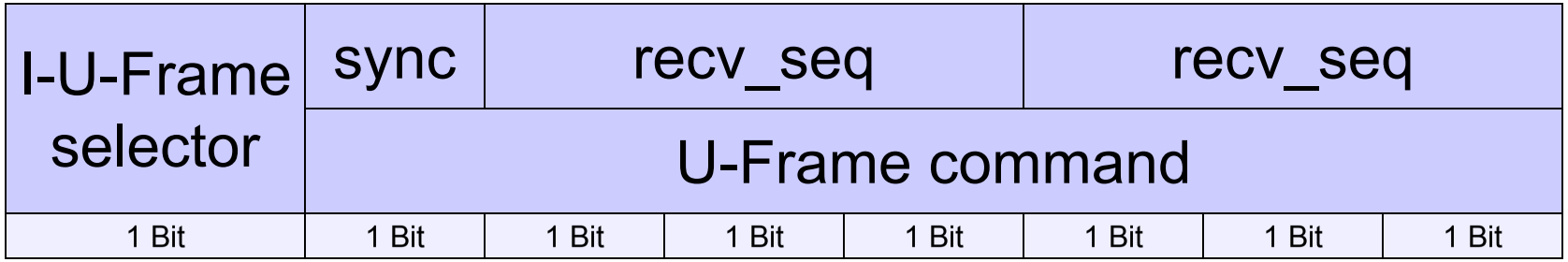
Pakettypen

- I-Frame

- U-Frame



- Header



Datentypen

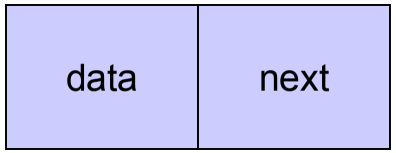
ptcp_channel_list_t

u_frame _msg	channel _flag	dest _address	my _address	send _seq	recv _seq	msg	error	next	prev

```
unsigned char u_frame_msg
unsigned char channel_flag
unsigned char dest_address
unsigned char my_address
unsigned char send_seq
unsigned char recv_seq
ptcp_message_list_t* message
void(*error_routine)(struct ptcp_connection_list_t_def *channel)
struct ptcp_connection_list_t_def *next
struct ptcp_connection_list_t_def *prev
```

Datentypen

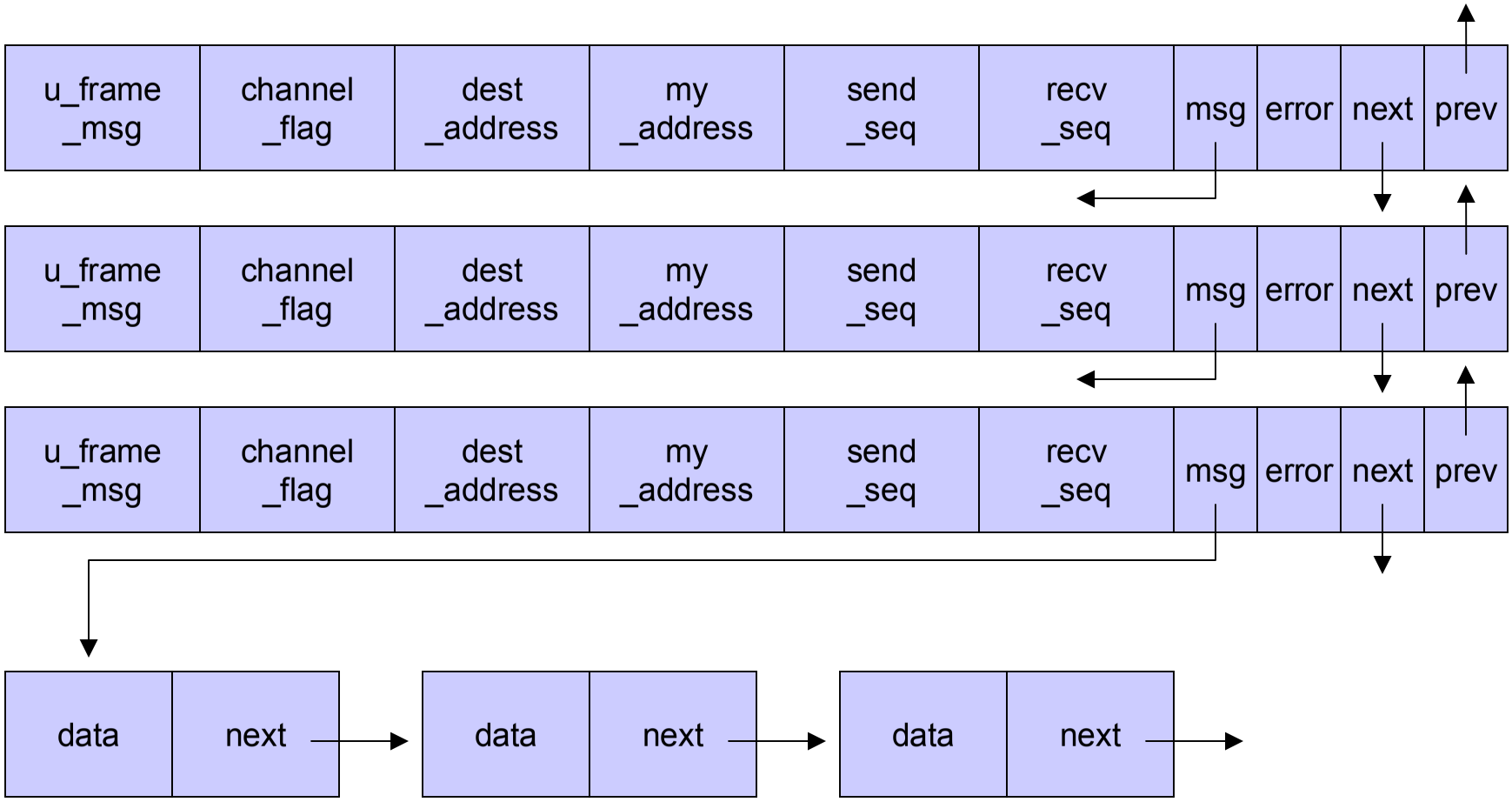
ptcp_message_list_t



unsigned char* data

struct ptcp_message_list_t_def* next

Datentypen


u_frame _msg
channel _flag
dest _address
my _address
send _seq
recv _seq
msg
error
next
prev
u_frame _msg
channel _flag
dest _address
my _address
send _seq
recv _seq
msg
error
next
prev
u_frame _msg
channel _flag
dest _address
my _address
send _seq
recv _seq
msg
error
next
prev
data
next
data
next
data
next

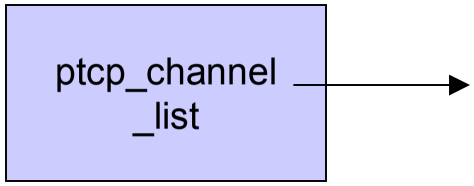
Funktionen

- PTCP einbinden
- Kanal initialisieren
- Verbindung aufbauen
- Daten senden
- Daten empfangen
- Bestätigung senden
- Verbindung abbauen
- Kanal löschen

Paket einbinden

#include <ptcp.h>

- globale Datenstruktur ptcp_channel_list wird angelegt



Kanal initialisieren

```
ptcp_channel_list_t* ptcp_channel (
        unsigned char dest_address,
        unsigned char my_port,
        void (*ptcp_error_routine)
           (ptcp_channel_list_t *channel)
        );
```

- startet ggf. ptcp_channel_hander_thread
- registriert ptcp_channel_handler
- legt Datenstrukturen an

Verbindung aufbauen

```
int ptcp_listen (
    ptcp_channel_list_t* channel
    );
```

- wartet auf Verbindungsaufbau der Gegenseite
- aktiviert Kanal nach erfolgreichem Aufbau

Verbindung aufbauen

```
int ptcp_connect (
    ptcp_channel_list_t *channel
    );
```

- sendet Anfrage zum Verbindungsaufbau an listener
- aktiviert Kanal nach erfolgreichem Aufbau

Daten senden

```
int ptcp_send (
    ptcp_channel_list_t *channel,
    const unsigned char *data
    );
```

- sendet Datenpuffer (max. 250 Bytes) über den Kanal
- wartet auf Empfangsbestätigung oder Timeout

Daten empfangen

```
unsigned char *ptcp_receive (
    ptcp_channel_list_t *channel
    );
```

- prüft message_list des Kanals auf Dateneingang
- liefert NULL, falls keine Daten vorhanden
- liefert sonst älteste Daten der Liste und löscht sie

Daten empfangen

```
unsigned char *ptcp_receive_wait (
    ptcp_channel_list_t *channel
    );
```

- wartet auf Dateneingang, falls message_list leer
- liefert älteste Daten der Liste und löscht sie

Bestätigung senden

```
int ptcp_send_ack (
    ptcp_channel_list_t *channel,
    unsigned char retries
    );
```

- sendet Empfangsbestätigung ohne auf den Timeout des PTCP Threads zu warten
- wartet ggf auf Rückantwort (bei retries > 1)
- dient zur Beschleunigung der Unidirektionalen Datenübertragung (Client-Server)

Verbindung abbauen

```
int ptcp_quit (
    ptcp_channel_list_t *channel
    );
```

- “beantragt Trennung der Verbindung”
- signalisiert dem Gegenüber die Bereitschaft zum Trennen der Verbindung
- danach kann der Sender von ptcp_quit keine Datenpakete mehr verschicken

Verbindung abbauen

```
int ptcp_disconnect (
    ptcp_channel_list_t *channel
    );
```

- folgt auf ptcp_quit
- Trennung der Verbindung nach Empfang eines ptcp_quit-Signals

Verbindung abbauen

```
void ptcp_connection_shutdown (
    ptcp_channel_list_t *channel
    );
```

- trennt die Verbindung unverzüglich
- wartet nicht auf eine Bestätigung der Gegenseite
- kann zu Datenverlust führen
- schnellste Methode

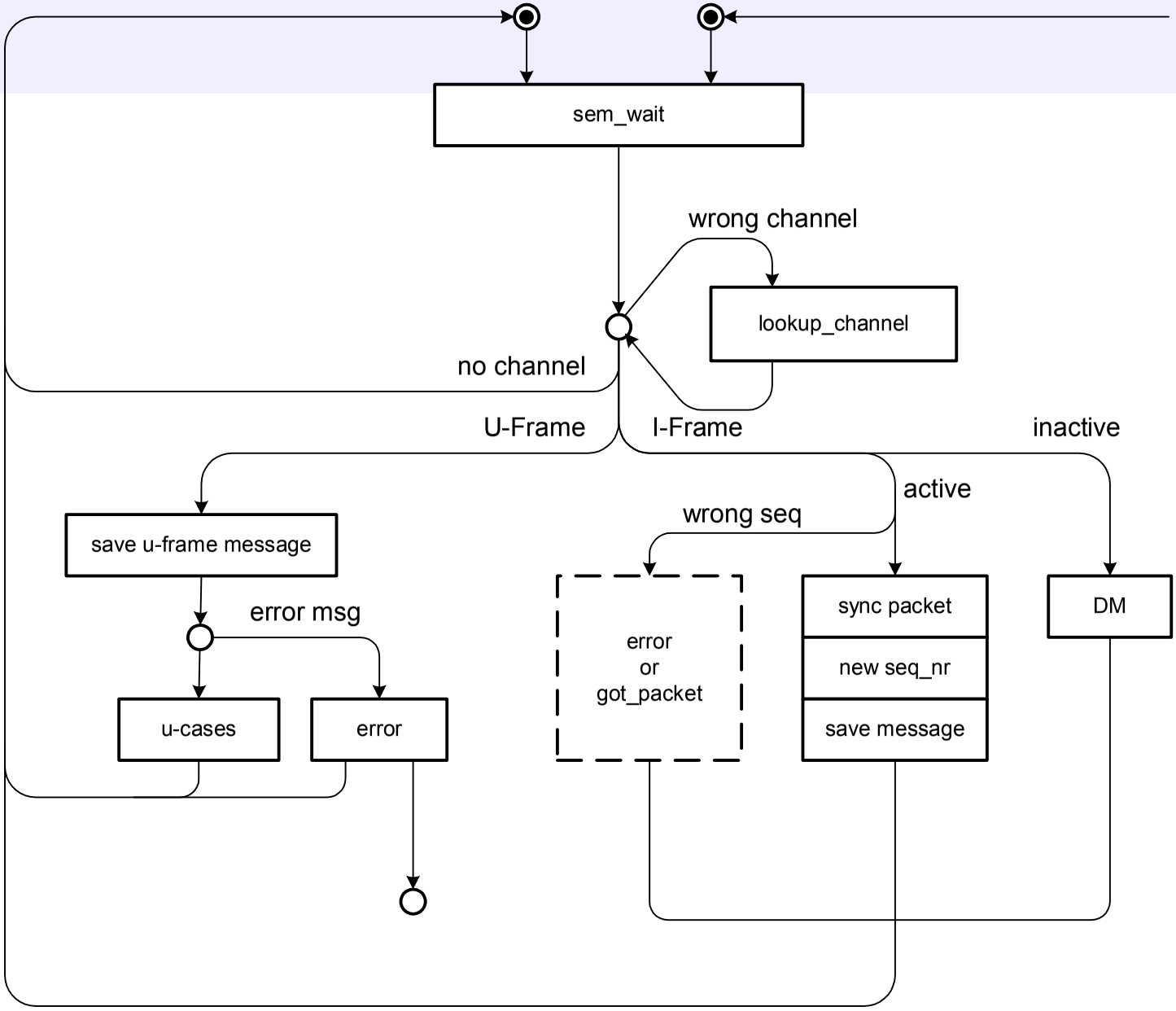
Kanal löschen

```
int ptcp_channel_close (
    ptcp_channel_list_t *channel
    );
```

- löscht Nachrichtenliste für den Kanal
- entfernt Kanal aus ptcp_channel_list
- löscht ggf. ptcp_channel_list
- stoppt ggf. den PTCP-Thread
- gibt Speicher frei

Interna

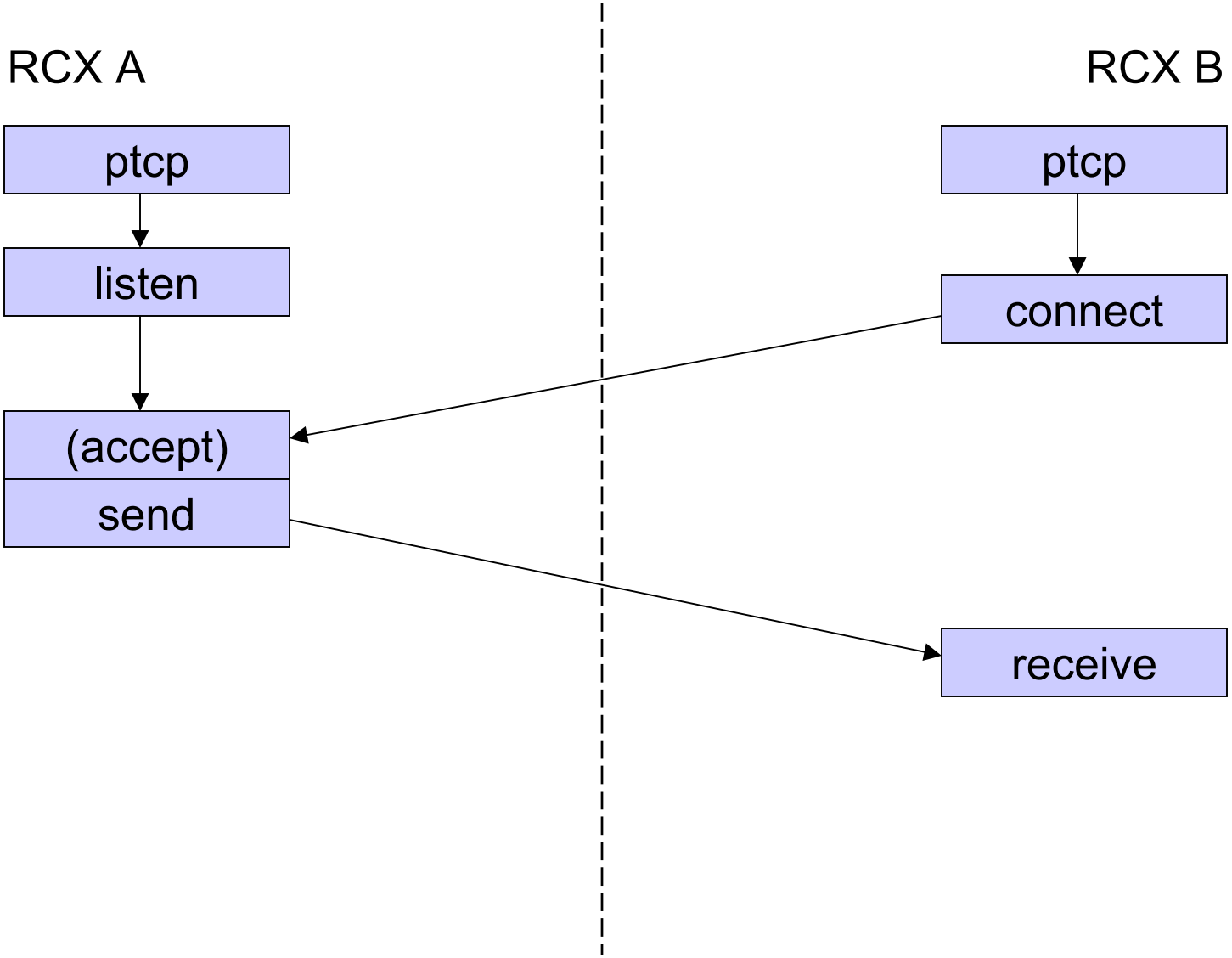
- Arbeitsweise des PTCP-Thread


sem_wait
wrong channel
lookup_channel
no channel
U-Frame
I-Frame
inactive
active
wrong seq
save u-frame message
error msg
u-cases
error
error
or
got_packet
sync packet
new seq_nr
save message
DM

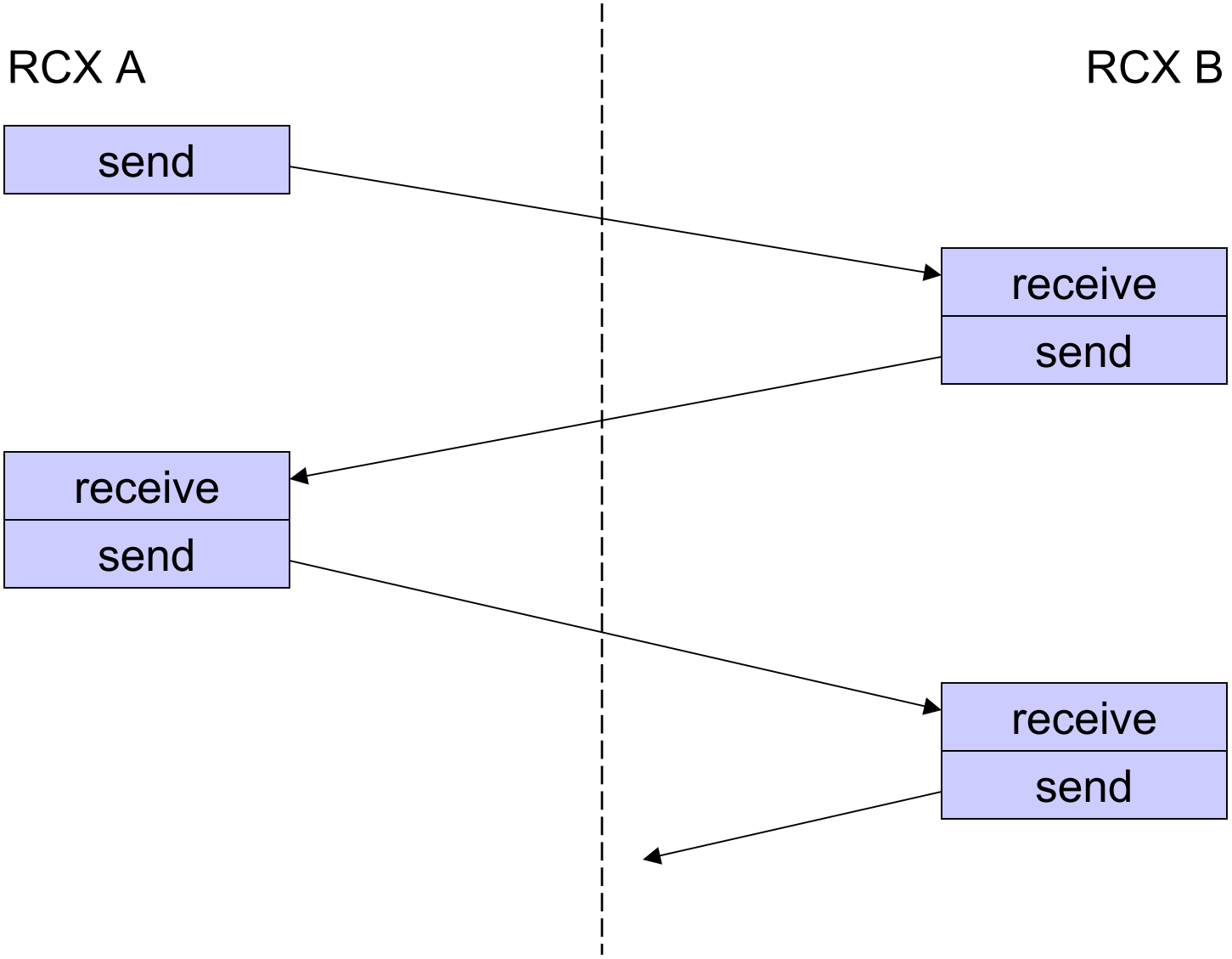
Kommunikation (Beispiele)

- Verbindungsaufbau
- gleichberechtigte Kommunikation
- Client – Server – Szenario
- sporadische Datenübertragung
- Fehlerszenarien
- Verbindungsabbau
  - kontrolliert (quit → disconnect)
  - kontrolliert (shutdown)
  - unkontrolliert

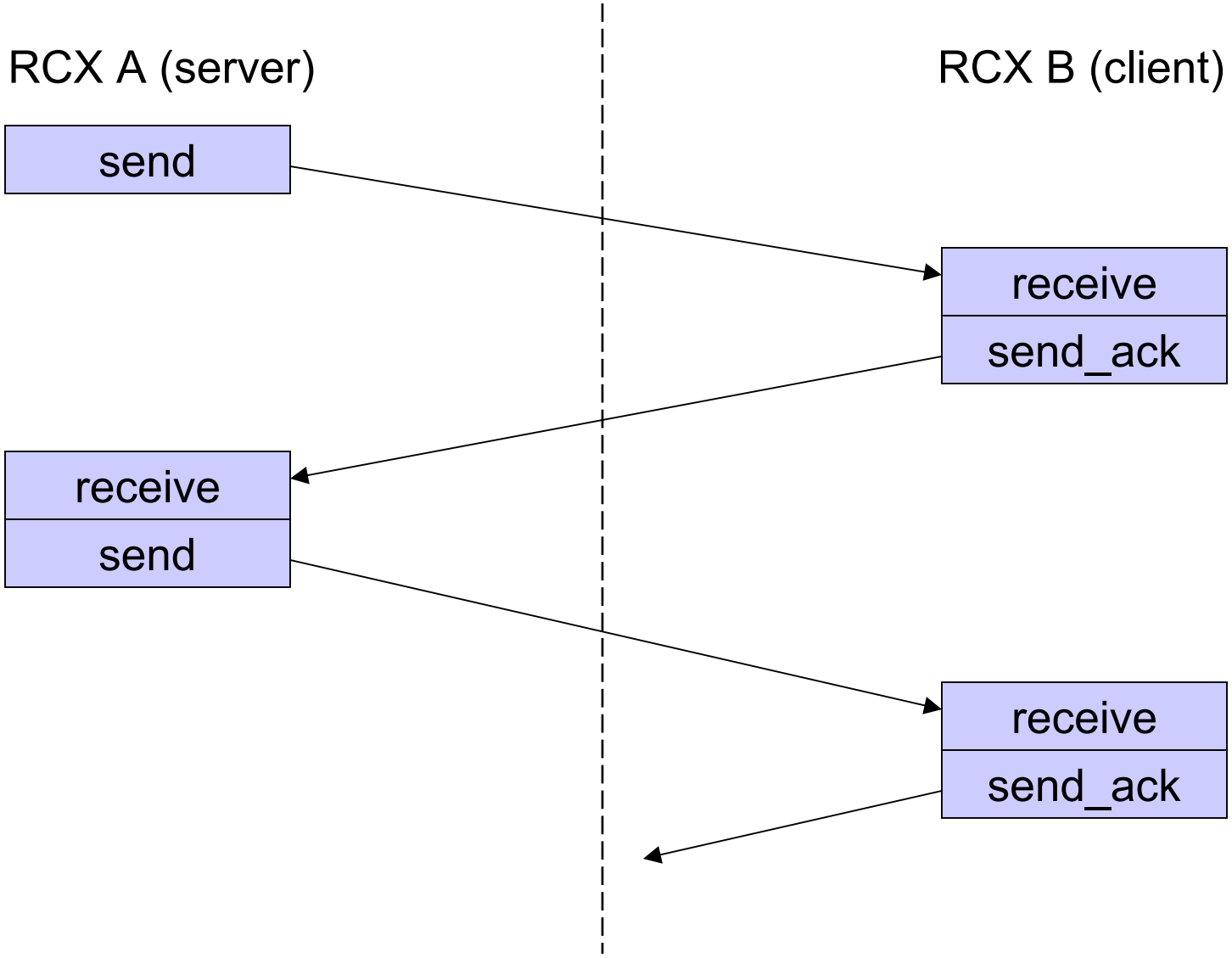
Verbindungsaufbau



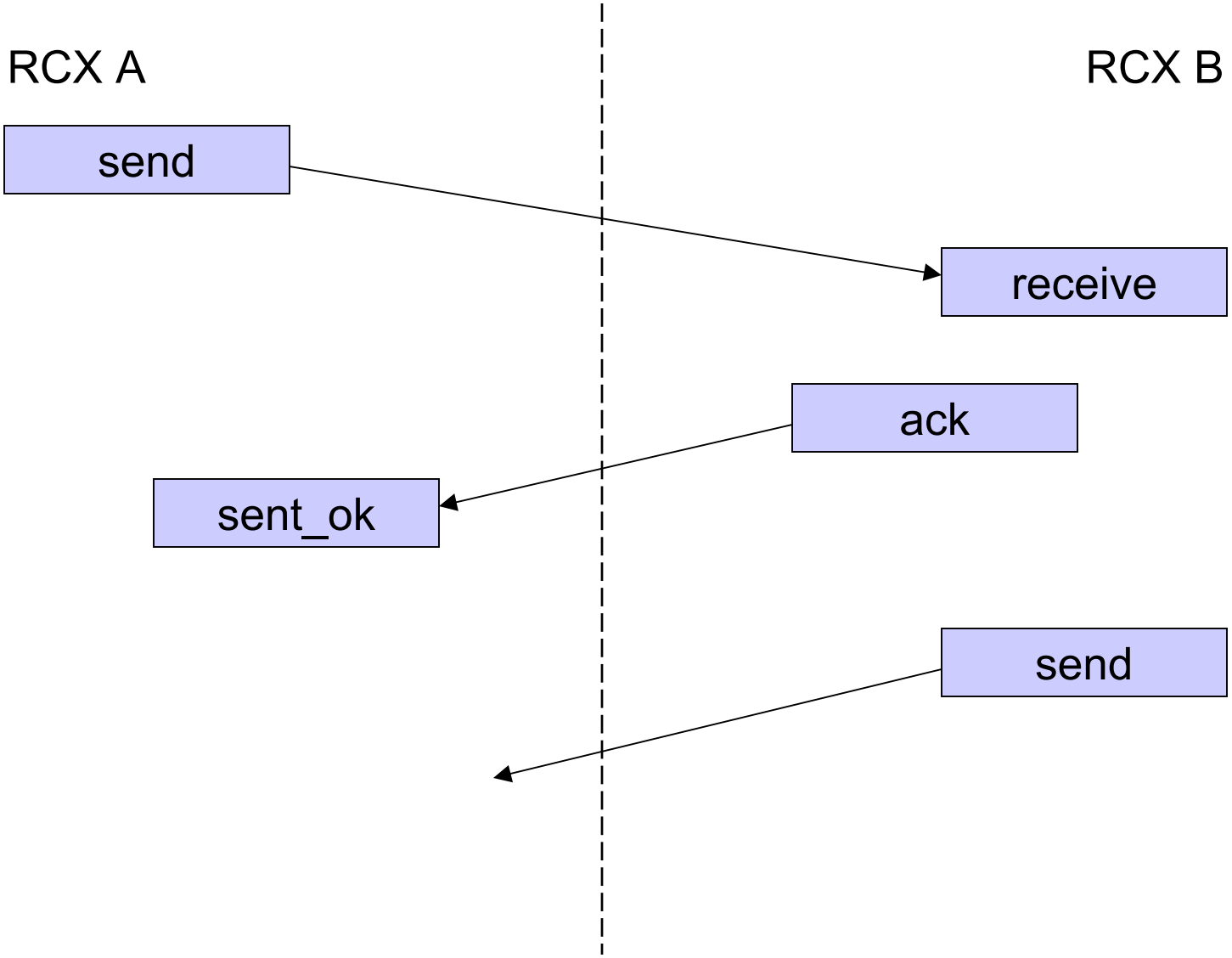
gleichberechtigte Kommunikation



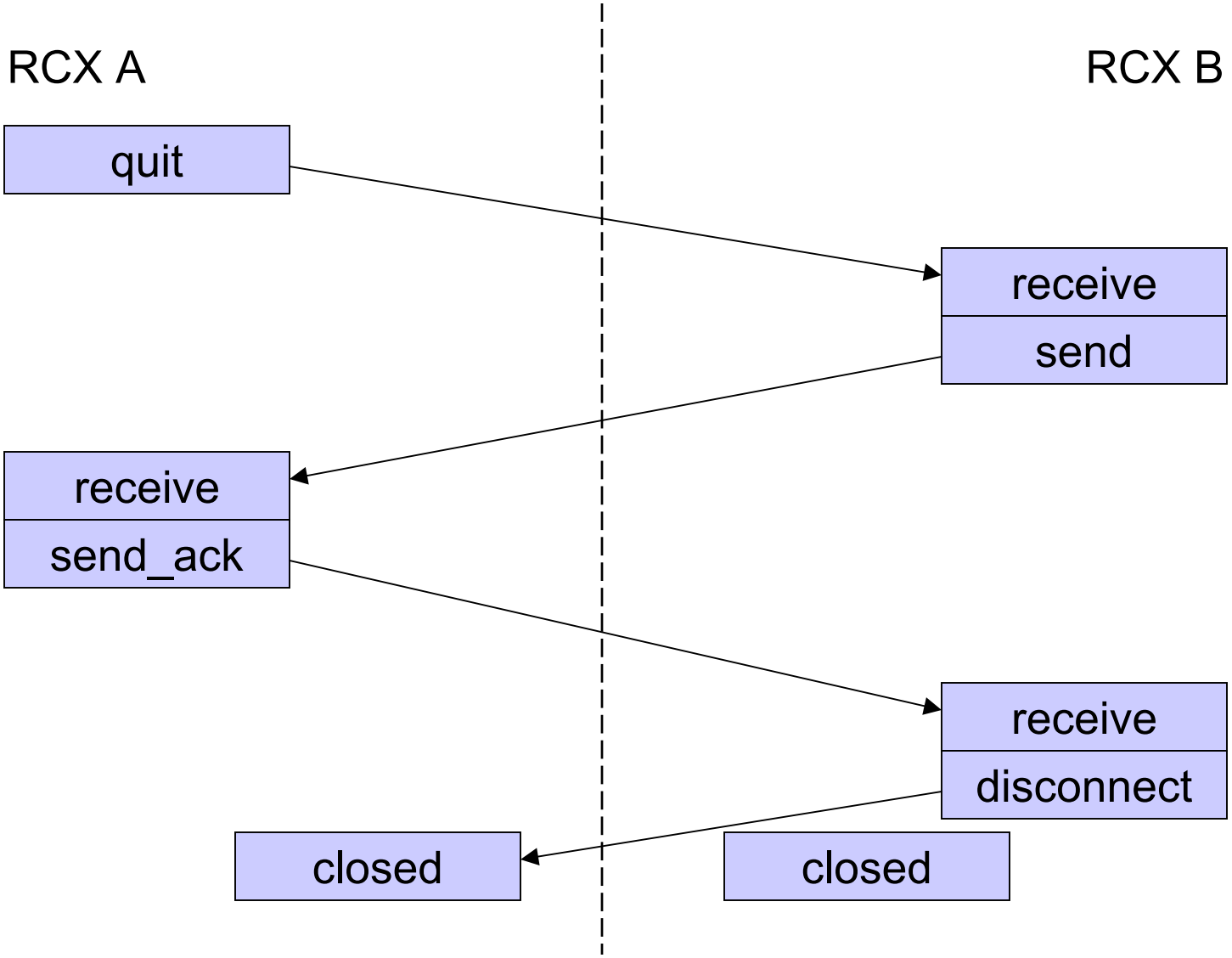
Client – Server – Szenario



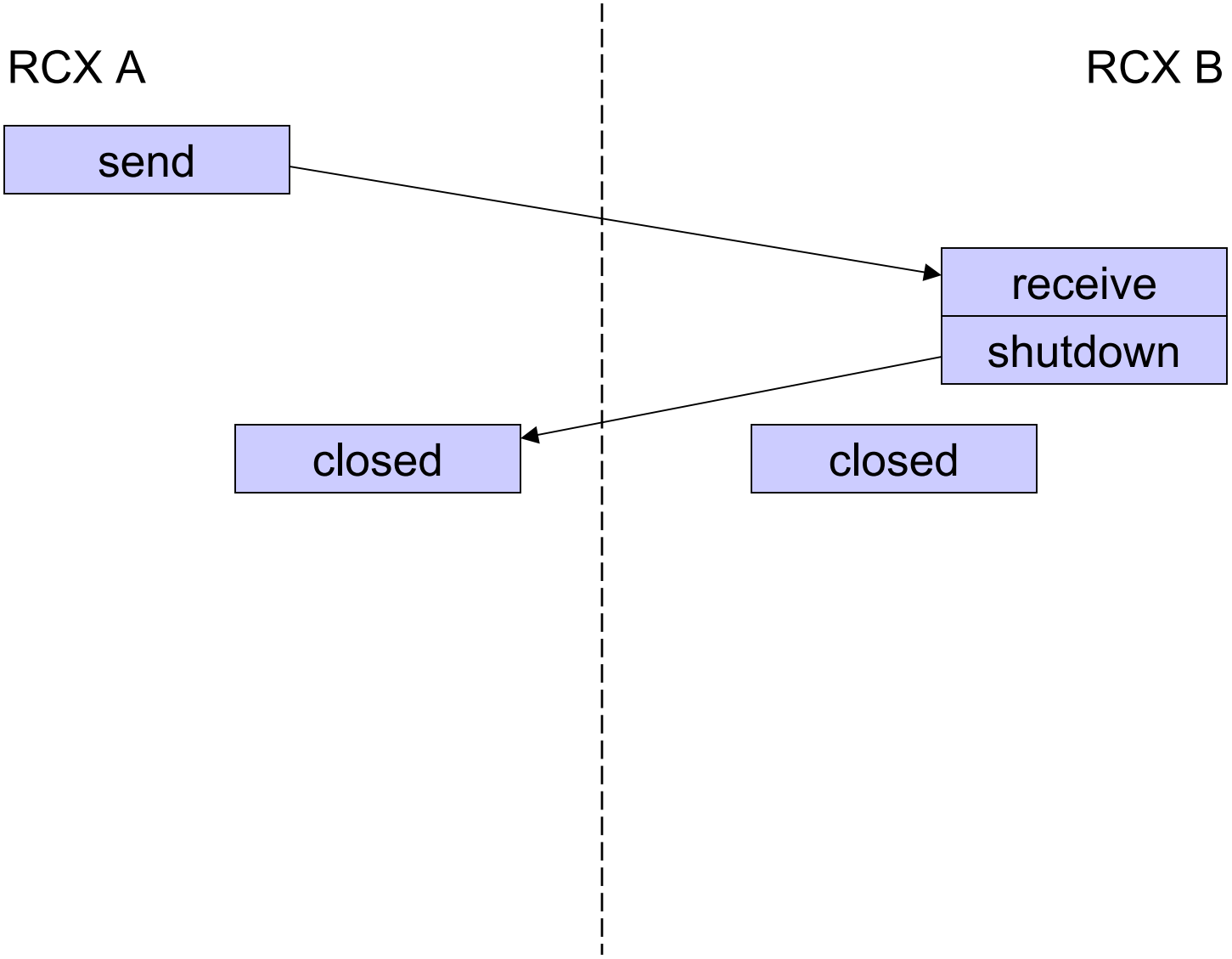
sporadische Datenübertragung



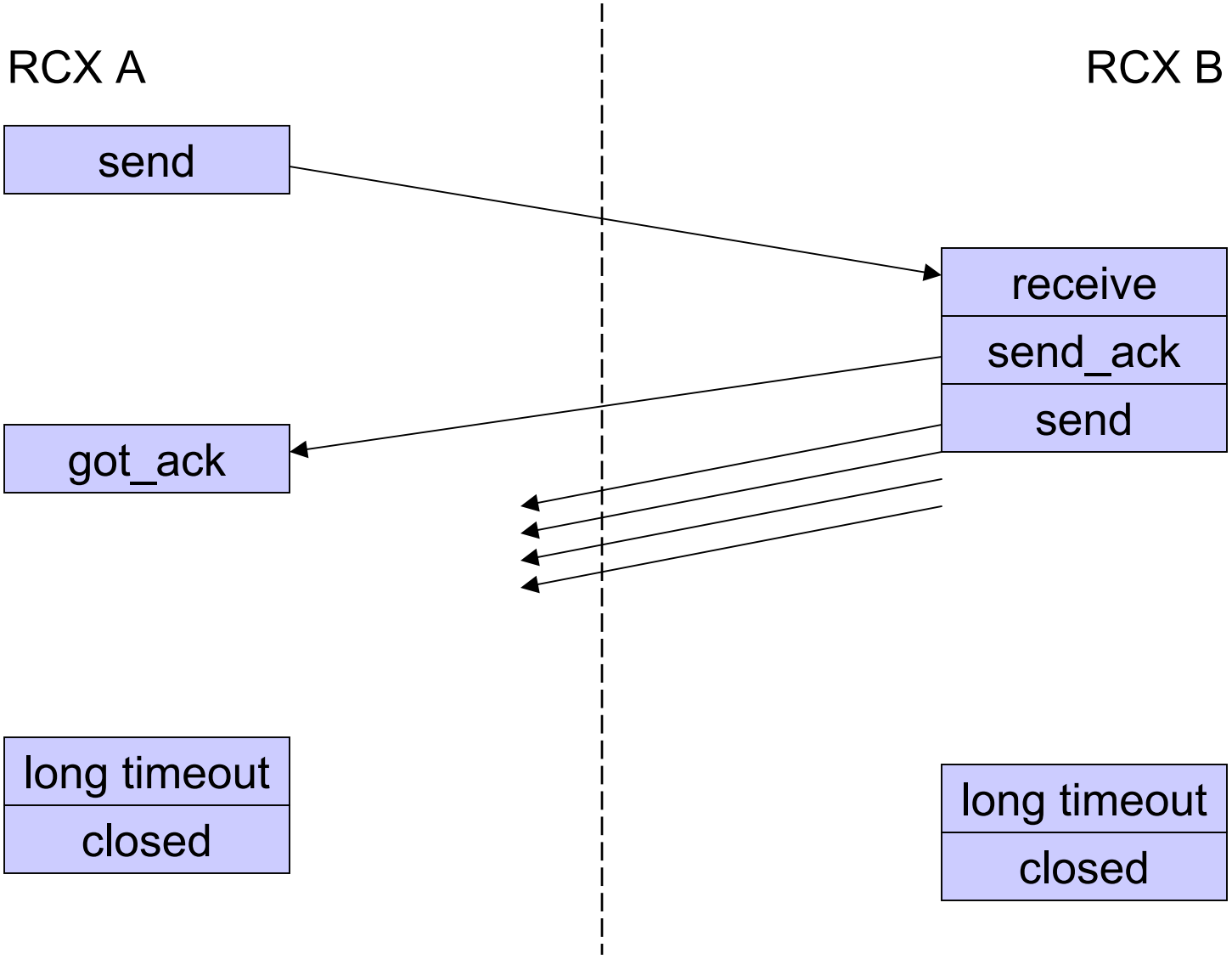
Verbindungsabbau: disconnect



Verbindungsabbau: shutdown



unkontrollierter Verbindungsabbau



Ausblick

- Verändern der aktuellen Funktionen zu den hier vorgestellten, Verringerung der Komplexität
- Überdenken der Errorbehandlung
- Verringerung des Speicherbedarfs
- Portieren des Protokolls auf PC
- Verwenden des LNP-Router für
  - Kommunikation mit PCs im Netz
  - Kommunikation zwischen RCX über andere Medien