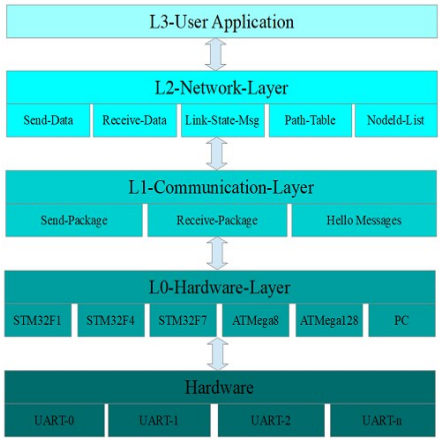
Projekt : X-NET

- Mikrocontroller-Netzwerk
- Open-Source
- Plattformübergreifend
- Selbstorganisierend

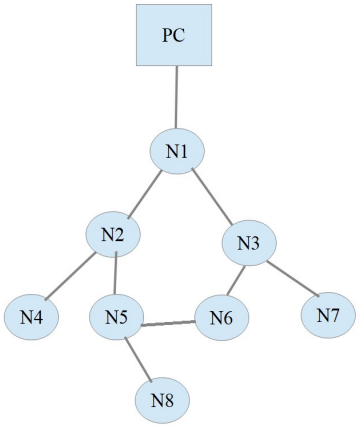
Inhaltsverzeichnis

1. Verbindungsschichten



- L3: User-Anwendung
- L2: Organisiert das Netzwerk
- L1: Handelt den Packet-Transfer
- L0: Übernimmt den Zugriff auf die Hardware
- Hardware: eigentliche Hardware-Plattform.

2. Netztopologie



(Beispiel eines Netzwerkes mit 8 Knoten und max. 3 Ports pro Knoten)

3. Layer-0 (Hardware-Layer)

- Besteht aus den Files: „xnet_l0.c“ und „xnet_l0.h“
- Dieser Layer spricht die Hardware-UART vom Mikrocontroller an
- Für eine Portierung von X-NET auf eine andere Plattform müssen diese beiden Files angepasst werden.
- Im Header-File wird die Spezifikation der Hardware eingetragen
  - Anzahl der Ports an diesem Knoten
  - Größe vom RX- und TX-Buffer der benutzen UART-Library
- Im C-File gibt es Funktionen, die für den Betrieb von XNET wichtig sind.
  - Init → hier werden alle internen UARTs initialisiert
  - Delay → hier wird ein delay von 1ms erzeugt
  - Send/Read_Uart → ruft die entsprechenden UART Funktionen auf.

4. Layer-1 (Communication-Layer)

- Besteht aus den Files: „xnet_l1.c“ und „xnet_l1.h“
- Dieser Layer handelt das versenden/empfangen von Packeten innerhalb vom X-NET
- an diesen Files keine Änderungen vornehmen !!

4.1. Beschreibung

- Jeder Knoten hat eine bestimmte Anzahl Ports (1..n).
- Jeder Port besitzt eine Hardware UART-Schnittstelle
- Jeder Port kann mit dem Ports eines anderen X-NET Knoten verbunden werden.

4.2. Verbindungsaufbau

- Layer-1 kennzeichnet zuerst alle Ports als „nicht verbunden“
- Layer-1 sendet zyklisch (alle 300ms) eine „Hello-Message“ über alle vorhandenen Ports.
- Falls das senden scheitert, bleibt der Port „nicht verbunden“
- Falls das senden zu einem anderen Knoten gelingt, wird der Port markiert mit „TX-OK“
- Zusätzlich prüft Layer-1 ständig alle vorhandenen Ports auf eingehende Packages
- Falls eine „Hello-Message“ an einem Port empfangen wird, wird der Port markiert mit „RX-OK“
- Sobald ein Port mit „TX-OK“ und „RX-OK“ markiert ist, gilt dieser Port als „verbunden“

4.3. Verbindungsunterbrechung

- Eine Unterbrechung wird durch die zyklischen „Hello-Messages“ erkannt.
- Falls das senden zu einem bereits als „verbunden“ markierten Port scheitert, wird der Port wieder als „nicht verbunden“ markiert.
- Falls der Empfang von einem Knoten scheitert, wird der entsprechende Port auch als „nicht verbunden“ markiert.

4.4. Verbindung mit Layer-2

- Bei einer Änderung vom Verbindungs-Status eines Ports „nicht verbunden“ ↔ „verbunden“ wird Layer-2 informiert.
- Falls ein Packet von einem anderen Knoten erfolgreich empfangen wurde, wird es in einen RX-Buffer eingetragen. Layer-2 kann diesen Buffer auslesen.
- Layer-2 trägt alle Packete zum senden in einen TX-Buffer ein. Layer-1 ließt diesen Buffer und sendet die Packete über den entsprechenden Port (bzw. mehrere Ports falls nötig) an den Nachbarknoten.

4.5. Reduzierung der Netzlast

- Um die Netzlast zu reduzieren, wird das senden der „Hello-Message“ einmalig ausgesetzt, sobald ein beliebiges Packet über diesen Port erfolgreich empfangen/gesendet wurde.
- Von zwei direkt verbundenen Knoten sendet somit immer nur einer die „Hello-Message“

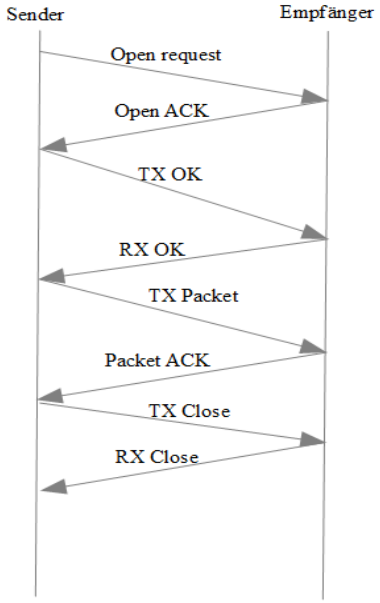
4.6. Packetaufbau

- Daten zwischen den Knoten werden als „Daten-Packete“ ausgetauscht
- Jedes Packet besteht aus 6...n Bytes
- Aufbau:

  - LEN = [1byte] Länge vom Packet in Bytes (Länge über alles inklusive LEN und CRC)
  - CMD = [1byte] Command-Identifier (definiert den Typ vom Packet)
  - ID = [1byte] NodeId (ID vom Knoten, der das Packet übertragen hat)
  - HOST = [1byte] NodeId (ID vom Knoten, der das Packet erstellt hat)
  - PAYLOAD = [0..n bytes] Zusatzdaten (optional)
  - CRC = [2 bytes] 16bit CRC (CRC16_CCIT_ZERO)

4.7. Packetversand

- Das senden/empfangen zwischen den Knoten geschieht mit einem Handshake am Start und Acknowledge am Ende der Übertragung.



- Der „open request“ wird mehrmals versucht, falls kein „open ACK“ empfangen wird, ist der Nachbarknoten nicht erreichbar → Port wird als „nicht verbunden“ markiert
- Falls ein Packet nicht mit „Packet ACK“ bestätigt wird, wird das senden bis zu 3mal wiederholt. Danach: → Port wird als „nicht verbunden“ markiert
- Der Layer-1 vom Empfänger prüft die CRC vom Packet und sendet „Packet NACK“ bei einem Fehler.
- Falls bei einem „open request“ ein „open request“ vom Empfänger empfangen wird → Kollision
  - beide Knoten warten eine unterschiedlich lange zeit und versuchen das senden nochmal.
  - Falls die Kollision nicht nach dem 3ten mal aufgelöst werden kann → Port wird als „nicht verbunden“ markiert
- Bei Sender und Empfänger gibt es an mehreren Stellen einen Timout falls die Gegenstelle nicht mehr reagiert.

5. Layer-2 (Network-Layer)

- Besteht aus den Files: „xnet_l2.c“ und „xnet_l2.h“
- Dieser Layer organisiert die Netzwerk-Topologie vom X-NET
- an diesen Files keine Änderungen vornehmen !!

5.1. Beschreibung

- Jeder Knoten besitzt einen eigenen „LinkState“
  (das ist der Status von jedem Port des Knotens)
- Jeder Knoten besitzt eine „NodeIdList“
  (das ist eine Liste aller im Netzwerk bekannten NodeID-Nummern der Knoten)
- Jeder Knoten besitzt eine „LinkStateTable“
  (das ist eine Liste aller LinkStates von allen Knoten des Netzwerks)
- Jeder Knoten besitzt einen „PathTable“
  (das ist eine Liste über welchen eigenen Port ein Knoten des Netzwerks zu erreichen ist)