Betrouwbare end-to-end communicatie

Patrick van Looy & Bram Leenders

9 mei 2014

1 Inleiding

Bij het opzetten van sensornetwerken, kan het zijn dat twee nodes niet in elkaars zendbereik vallen. In een dergelijk geval kan een tussenliggende node helpen door berichten door te sturen. Zo kunnen nodes als schakels in een ketting gebruikt worden om een groter bereik mogelijk te maken.

Een dergelijke constructie is een stuk gecompliceerder dan directe communicatie tussen twee nodes, omdat voor betrouwbare communicatie het zenden van ontvangstbevestigingen nodig is.

2 Probleemstelling

Om het probleem overzichtelijk te houden, is het aantal nodes in deze proef beperkt tot drie. Er is een zender, een ontvanger en een doorsturende node. Omdat het aantal nodes beperkt is, is er maar één mogelijk pad. Hierdoor kan het pad vooraf vastgelegd worden, dit heet ook wel statische routering.

De uiteindelijke opstelling moet de zender de garantie geven dat een bericht uiteindelijk ontvangen wordt door de ontvanger via de tussenliggende node.

3 Protocol

Om deze garantie te bieden, kan gebruik worden gemaakt van het alternating bit protocol. Dit protocol stuurt een nummer -een bit- mee met een pakket, wanneer de ontvanger het ontvangt stuurt deze bevestiging met datzelfde nummer terug. Indien de zender een bevestiging met het correcte (laatst verzonden) nummer ontvangt, hoogt deze een counter met een op en stuurt deze het volgende pakket.

Figuur 1 laat de mogelijke staten en transities van de verzender en ontvanger zien.

4 Methodologie

Onze implementatie gebruikt niet een enkele bit als identificatie van een pakket, maar een getal. Conceptueel verandert dit niets aan het protocol, maar het biedt wel de mogelijkheid voor later uitbreiding. Zo geldt niet langer de restrictie dat er slechts één oud pakket in het netwerk mag zitten. Onze implementatie werkt dus ook als oudere berichten (e.g. tien berichten terug) nog ronddolen in het netwerk.

Tevens stuurt de ontvanger een ACK terug die niet de identificatiecode bevat, maar de negatieve waarde hiervan. Hierdoor is het direct duidelijk of een pakket een bevestiging is (indien de code kleiner dan 0 is), of echte data bevat. Dit helpt om ook in grote netwerken ervoor te zorgen dat een zender niet zijn zelf verzonden bericht als ACK kan lezen.

5 Resultaten

De implementatie, waarvan de code is bijgevoegd in appendix A, is in verschillende vormen getest:

- De zender, repeater en ontvanger allemaal binnen ontvangstbereik van elkaar.
- De zender en repeater respectievelijk de repeater en ontvanger binnen ontvangstbereik, maar zender en ontvanger buiten ontvangstbereik.
- Alleen zender en ontvanger binnen ontvangstbereik; repeater buiten bereik.

Omdat er gebruik gemaakt is van statistische routering, is het geen optie voor de zender en ontvanger om direct te communiceren. In de eerste twee gevallen verloopt de communicatie via de repeater; in het derde geval kan geen verbinding worden opgezet omdat de repeater niet bereikbaar is.

6 Conclusie

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a diam lectus. Sed sit amet ipsum mauris. Maecenas congue ligula ac quam viverra nec consectetur ante hendrerit. Donec et mollis dolor. Praesent et diam eget libero egestas mattis sit amet vitae augue. Nam tincidunt congue enim, ut porta lorem lacinia consectetur. Donec ut libero sed arcu vehicula ultricies a non tortor. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean ut gravida lorem. Ut turpis felis, pulvinar a semper sed, adipiscing id dolor. Pellentesque auctor nisi id magna consequat sagittis. Curabitur dapibus enim sit amet elit pharetra tincidunt feugiat nisl imperdiet. Ut convallis libero in urna ultrices accumsan. Donec sed odio eros. Donec viverra mi quis quam pulvinar at malesuada arcu rhoncus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. In rutrum accumsan ultricies. Mauris vitae nisi at sem facilisis semper ac in est.

Vivamus fermentum semper porta. Nunc diam velit, adipiscing ut tristique vitae, sagittis vel odio. Maecenas convallis ullamcorper ultricies. Curabitur ornare, ligula semper consectetur sagittis, nisi diam iaculis velit, id fringilla sem nunc vel mi. Nam dictum, odio nec pretium volutpat, arcu ante placerat erat, non tristique elit urna et turpis. Quisque mi metus, ornare sit amet fermentum et, tincidunt et orci. Fusce eget orci a orci congue vestibulum. Ut dolor diam, elementum et vestibulum eu, porttitor vel elit. Curabitur venenatis pulvinar tellus gravida ornare. Sed et erat faucibus nunc euismod ultricies ut id justo. Nullam cursus suscipit nisi, et ultrices justo sodales nec. Fusce venenatis facilisis lectus ac semper. Aliquam at massa ipsum. Quisque bibendum purus convallis

nulla ultrices ultricies. Nullam aliquam, mi eu aliquam tincidunt, purus velit laoreet tortor, viverra pretium nisi quam vitae mi. Fusce vel volutpat elit. Nam sagittis nisi dui.

Suspendisse lectus leo, consectetur in tempor sit amet, placerat quis neque. Etiam luctus porttitor lorem, sed suscipit est rutrum non. Curabitur lobortis nisl a enim congue semper. Aenean commodo ultrices imperdiet. Vestibulum ut justo vel sapien venenatis tincidunt. Phasellus eget dolor sit amet ipsum dapibus condimentum vitae quis lectus. Aliquam ut massa in turpis dapibus convallis. Praesent elit lacus, vestibulum at malesuada et, ornare et est. Ut augue nunc, sodales ut euismod non, adipiscing vitae orci. Mauris ut placerat justo. Mauris in ultricies enim. Quisque nec est eleifend nulla ultrices egestas quis ut quam. Donec sollicitudin lectus a mauris pulvinar id aliquam urna cursus. Cras quis ligula sem, vel elementum mi. Phasellus non ullamcorper urna.

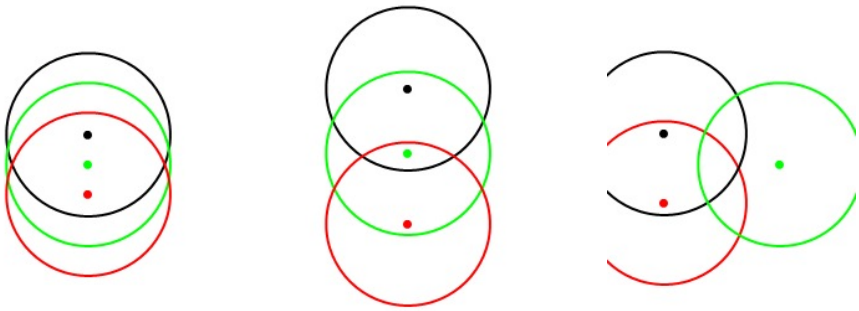
A Bijlage 1 - Code

```
/* Copyright (C) 2011 J. Coliz <maniacbug@ymail.com> */

#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"

RF24 radio(3, 9);

// sets the role of this unit in hardware.  Connect to GND to be
    the 'pong' receiver
// Leave open to be the 'ping' transmitter
const int role_pin_sender = 7;
const int role_repeat = 6;

// Radio pipe addresses for the 3 nodes to communicate.
const uint64_t pipes[3] = { 0x123456789aLL, 0x987654321bLL,
    0x384654f2cbLL };

const int sendValue = 170; // binary; 10101010
const int numberOfPackets = 1000;
const int RESETVAL = 42;

typedef enum { role_sender = 1, role_receiver = 2, role_repeater =
    3 } role_e;
// The debug-friendly names of those roles
const char* role_friendly_name[] = { "invalid", "Sender",
    "Receiver", "Repeater"};

// The role of the current running sketch
role_e role;

void setup(void) {
    // set up the role pin
    pinMode(role_pin_sender, INPUT);
    digitalWrite(role_pin_sender,HIGH);
    delay(20); // Just to get a solid reading on the role pin
    // read the address pin, establish our role
    if (!digitalRead(role_pin_sender)) {
        role = role_sender; // sender
    }
    else {
        // set up the role pin
        pinMode(role_repeat, INPUT);
        digitalWrite(role_repeat,HIGH);
        delay(20); // Just to get a solid reading on the role pin

        if (!digitalRead(role_repeat)) {
            role = role_repeater; // repeater
        }
        else {
            role = role_receiver; // receiver
        }
    }

    Serial.begin(57600);
    printf_begin();

    radio.begin();
    radio.setRetries(0,0);
```

```
    radio.setDataRate(RF24_250KBPS);
    radio.setPALevel(RF24_PA_MAX);
    radio.setChannel(0);

    // optionally, reduce the payload size.  seems to
    // improve reliability
    radio.setPayloadSize(8);

    if ( role == role_sender ) { // Sender
        radio.openWritingPipe(pipes[1]); // Write to repeater
        radio.openReadingPipe(1,pipes[0]); // Read from repeater
    }
    else if (role == role_repeater){ // Repeater
        radio.openReadingPipe(1,pipes[1]);
    }
    else { // Receiver
        radio.openWritingPipe(pipes[1]);
        radio.openReadingPipe(1,pipes[2]);
    }

    radio.startListening();
    radio.printDetails();
}

// Bevat het nummer dat nu gestuurd moet worden
int currentNumber = 1;

void resetRadioReads(void) {
    radio.stopListening();
    radio.openWritingPipe(0x111111110aLL);
}

void loop(void) {
    if (role == role_sender) {
        radio.stopListening();
        radio.openWritingPipe(pipes[1]);
        bool ok = radio.write( &currentNumber, sizeof(int) );
        resetRadioReads();
        radio.startListening();

        // Wait here until we get a response, or timeout (250ms)
        unsigned long started_waiting_at = millis();
        bool timeout = false;
        while ( ! radio.available() && ! timeout )
            if (millis() - started_waiting_at > 250 )
                timeout = true;

        // Describe the results
        if ( timeout ) {
          // Zend opnieuw!
          // currentNumber wordt opnieuw verzonden in de volgende
              iteratie van loop()
          printf("Timeout occurred at sender; no ACK received.
              Packet #: %i\n", currentNumber);
        }
        else {
            int receivedValue;
            radio.read( &receivedValue, sizeof(int) );

            // ACK our value! Increase our success counter.
            if(receivedValue == - currentNumber) {
```

```
                // Success!
                printf("ACK succesfully received for packet
                    #%i\n", currentNumber);
                currentNumber++;
            }
            else {
              // Received double ACK
              printf("Received double ACK. Packet #: %i\n",
                  currentNumber);
            }
        }
        delay(50);
    }

    if ( role == role_receiver ) {
        // if there is data ready
        if ( radio.available() ) {

            // Dump the payloads until we've gotten everything
            int v; bool done = false;
            while (!done) {
                done = radio.read( &v, sizeof(int) );
                delay(10);
            }

            printf("Received packet # %i\n", v);

            v = -v; // ACK waarde is negatief

            radio.stopListening();
            radio.openWritingPipe(pipes[1]);
            radio.write( &v, sizeof(int) );
            resetRadioReads();
            radio.startListening();
        }
    }
    if (role==role_repeater) {
        if ( radio.available() ) {
            int v; bool done = false;
            while (!done) {
                done = radio.read( &v, sizeof(int) );
                delay(10);
            }
            radio.stopListening();

            if (v > 0) { // Dit is het originele bericht; stuur
                naar receiver
                radio.openWritingPipe(pipes[2]);
            }
            else { // Dit is de ACK; stuur naar sender
                radio.openWritingPipe(pipes[0]);
            }

            radio.write( &v, sizeof(int) );
            resetRadioReads();
            radio.startListening();
        }
    }
}
```

Figuur 1: State-diagram Alternating Bit Protocol



Alles bereikbaar Repeater bereikbaar Repeater onbereikbaar

Figuur 2: Drie mogelijke scenario's; zwart is de verzender, groen is de repeater en rood is de ontvanger.