Netwerk latentie in communicatie tussen microcontrollerbord via TCP

Bryan Chung

November 2021

Abstract

Het doel van dit onderzoek is om te bepalen of de netwerk latentie van de communicatie tussen meerdere microcontrollerborden via TCP laag genoeg is voor praktische applicatie. Met andere woorden de netwerk vertraging is zo laag dat de communicatie via TCP tussen microcontrollerborden is zo snel, het kan worden gebruikt voor praktische applicatie bijvoorbeeld het besturen van een elektro-motortje. De gevonden resultaat van dit onderzoek is de netwerk latentie ligt onder de 100ms.

1 Introductie

Netwerk latentie is de tijd die nodig is voordat de gegevens die u aan het ene uiteinde van uw netwerk (node) invoert, aan het andere uiteinde verschijnen. In moderne robotica is een laag netwerk latentie belangrijk, want je kan in een groot afstand een robot besturen met een klein vertraging. Hierdoor kan bijvoorbeeld een chirurg in Rotterdam een patiënt in Nairobi realtime opereren via een chirurgische robot. Alles onder de 100ms telt als laag latentie[3]. Het doel van dit onderzoek is om te bepalen of de netwerk latentie tussen microcontrollerborden laag genoeg is voor praktische applicatie. Om dit te kunnen doen, werd er een experiment opgesteld met twee Arduino Uno en een Rasberry Pi, berichten werden naar elkaar gestuurd via het netwerk van de Rasberry Pi de tijdstip werd gemarkeerd van de Arduino's. Door het gemarkeerde zender tijdstip van de ontvanger aftrekken, krijg je de latentie.

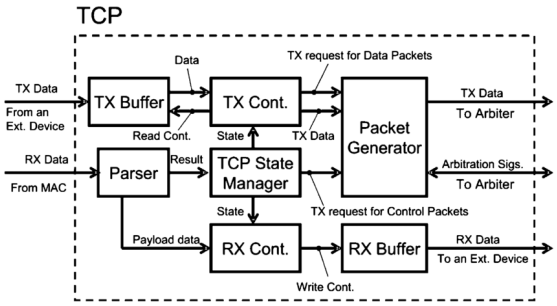
$$Latentie = Tijd_{ontvanger} - Tijd_{zender} \tag{1}$$

Door dit poging te herhalen wordt er een lijst van latentie opgesteld. Door de som van alle latentie in de lijst delen door het aantal pogingen krijg je de gemiddelde latentie.

$$Latentie_{gem} = Latentie_{som}/Poging_n \tag{2}$$

Hierdoor wordt de gemiddelde latentie berekend en een conclusie getrokken. De aanleiding om de TCP protocol te gebruiken is omdat de TCP een IP-netwerkprotocol voor stabiele, betrouwbare netwerkverbindingen is en geen verbindingsloos protocol zoals UDP en GRE. TCP heeft als kenmerken dat het in een

datastroom kan verzenden, waarbij de gegevens worden geleverd zoals ze ontvangen, en toekomstige communicatie fouten, zowel in de gegevens zelf als in de uitvoering van de gegevens kunnen worden opgevangen. Om een clientapplicatie te gebruiken die TCP als transmissieprotocol heeft gebruikt, geen rekening te houden met de onderliggende netwerkarchitectuur en toekomstige fouten in de communicatie. Hieronder is een schema van een TCP blok.



Dit is het hoofdblok van TCP en verwerkt TCP als server of client. Het parsercircuit verwerkt alleen pakketten die vooraf een TCP-tag en een bestemmingspoortnummer hebben[2]. Deze pakketten worden geanalyseerd. Circuits in dit blok, behalve de Parser, worden bestuurd door de manager met een TCP-status. Er wordt een gereduceerde TCP-statusmachine aangenomen. Gelijktijdig openen en sluiten wordt niet ondersteund.

2 Experiment

Doel van dit experiment is om te testen of de netwerk latentie laag genoeg is voor communicatie via TCP tussen microcontrollerborden. Om dit te kunnen doen wordt er een experiment protocol die bestaat uit vraagstelling, meetopstelling, te verrichten handelingen, vast te leggen data, analyse opgesteld. De reden waarom er een experiment protocol opgesteld is, omdat experimentele protocollen staan centraal voor reproduceerbaarheid [1].

2.1 Vraagstelling

Er werd een hoofdvraag opgesteld voor dit experiment:

- Wat is de netwerk latentie in communicatie via TCP tussen twee Arduino Uno ?

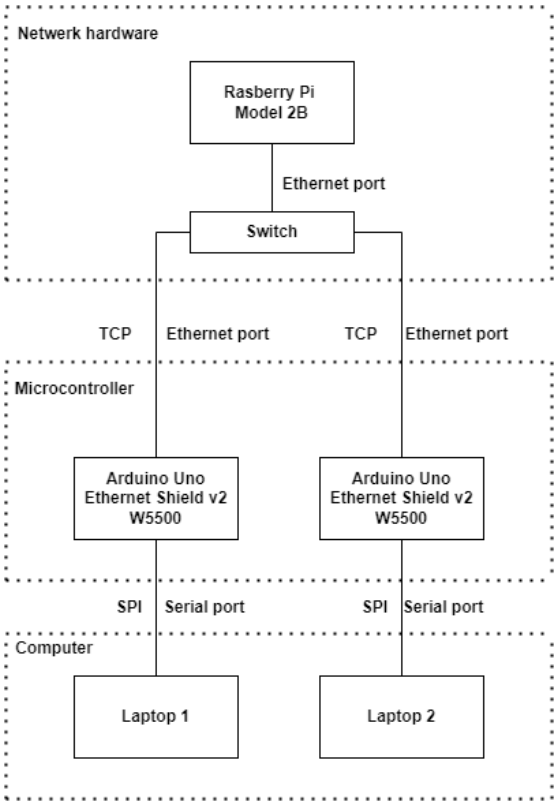
2.2 Meetopstelling

In dit experiment wordt er gebruik gemaakt van de volgende instrumenten:

- 1x Rasberry Pi model 2B
- 2x Arduino Uno

- 2x Arduino Ethernet shield 2
- 2x Laptop

De Raspberry Pi 2 Model B is de tweede generatie Raspberry Pi. Het is gebaseerd op de BCM2836 system-on-chip (SoC), die een quad-core ARM Cortex-A7-processor en een krachtige GPU bevat. De Arduino Uno is een microcontrollerbord gebaseerd op de ATmega328. Het heeft 20 digitale input/output-pinnen (waarvan 6 kunnen worden gebruikt als PWM-uitgangen en 6 kunnen worden gebruikt als analoge ingangen), een 16 MHz-resonator, een USB-aansluiting, een voedingsaansluiting, een in-circuit systeemprogrammering (ICSP) koptekst en een resetknop. Met het Arduino Ethernet Shield 2 kan een Arduinobord verbinding maken met internet. Het is gebaseerd op de Wiznet W5500 Ethernet-chip. De Wiznet W5500 biedt een netwerk (IP)-stack die zowel TCP als UDP aankan. Het ondersteunt maximaal acht gelijktijdige socketverbindingen. Als eerst moet je de Rasberry pi aansluiten op de switch. Zorg dat de Pi "Ethernet port sharing compatible" is. Dan moet je zorgen dat beide Arduino is aangesloten aan hun eigen laptop. Vervolgens sluit je beiden Arduino op de switch via de Ethernet port. Hieronder is een blokschema van de aanslutingen.



Voor de software van Arduino moet je Ethernet en SPI bibliotheek gebruiken. De SPI werd gebruikt om te communiceren met de Arduino via de serial port van

de laptop. SPI is een interfacebus die gewoonlijk wordt gebruikt om gegevens te verzenden tussen microcontrollers en kleine randapparatuur zoals schuifregisters, sensoren en SD-kaarten. Het maakt gebruik van afzonderlijke klok- en datalijnen, samen met een geselecteerde lijn om het apparaat te kiezen waarmee u wilt praten. De Ethernet bibliotheek is ontworpen om te werken met het Arduino Ethernet Shield, Arduino Ethernet Shield 2, Leonardo Ethernet en alle andere W5100/W5200/W5500-gebaseerde apparaten. Met deze bibliotheek kan een Arduino-bord verbinding maken met internet.

2.3 Te verrichten handelingen

Om het experiment te starten moet je de code op beide Arduino runnen. Hieronder zijn de codes van de ontvanger en de zender.

Ontvanger:

```
#include <SPI.h>
#include <Ethernet.h>

const int serverPort = 4080;
byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
EthernetServer TCPserver(serverPort);

void setup() {
  Serial.begin(9600);

  Serial.println("ARDUINO #2: TCP SERVER");

  if (Ethernet.begin(mac) == 0)
    Serial.println("Failed to configure Ethernet using DHCP");

  Serial.print("TCP Server IP address: ");
  Serial.println(Ethernet.localIP());
  Serial.println("-> Please update the serverAddress in Arduino #1 code");

  TCPserver.begin();
}

void loop() {
  // Wait for a TCP client from Arduino #1:
  EthernetClient client = TCPserver.available();

  if (client) {
    // Read the command from the TCP client:
    int value = client.read();
    Serial.print("- Received value: ");
```

```
    Serial.println(value);

    Ethernet.maintain();
  }

}
```

Zender:

```
#include <SPI.h>
#include <Ethernet.h>
#define value 1

const int serverPort = 4080;
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress serverAddress(192, 168, 137, 228);
EthernetClient TCPclient;

void setup() {
  Serial.begin(9600);
  Serial.println("ARDUINO #1: TCP CLIENT + A BUTTON/SWITCH");

  if (Ethernet.begin(mac) == 0)
    Serial.println("Failed to configure Ethernet using DHCP");

  if (TCPclient.connect(serverAddress, serverPort))
    Serial.println("Connected to TCP server");
  else
    Serial.println("Failed to connect to TCP server");

}

void loop() {

  if (!TCPclient.connected()) {
    Serial.println("Connection is disconnected");
    TCPclient.stop();
    // reconnect to TCP server (Arduino #2)
    if (TCPclient.connect(serverAddress, serverPort))
      Serial.println("Reconnected to TCP server");
    else
      Serial.println("Failed to reconnect to TCP server");
  }
  TCPclient.write(value);
  delay(5000);
}
```

De ontvanger ontvangt in een interval van 5 seconde een integer van de zender. Je noteer eerst de tijdstip op de serial monitor van het versturen van het integer. Vervolgens na 5 seconde komt de bericht bij de ontvanger, je noteer dan de tijdstip van het ontvangen bericht. Zo ga je verder door in een interval van 5 secondes totdat je 1 minuut bereikt, dus 13 berichten gestuurd. Je noteer alles in een tabel.

2.4 Vast te leggen data

De tijdstip wordt vastgelegd in de serial monitor van de Arduino. Vervolgens noteer je met hand de tijdstippen op een tabel. Om de latentie te berekenen gebruik je (1). Latentie noteer je in milliseconde(ms).

2.5 Analyse

Tabel:

Tijdstip zender	Tijdstip ontvanger	Latentie (ms)
16:05:17.567	16:05:17.661	94
16:05:22.567	16:05:22.597	30
16:05:27.567	16:05:27.613	46
16:05:32.567	16:05:32.650	83
16:05:37.567	16:05:37.609	42
16:05:42.567	16:05:42.653	86
16:05:47.567	16:05:47.663	96
16:05:52.567	16:05:52.628	61
16:05:57.567	16:05:57.590	23
16:06:02.567	16:06:02.608	41
16:06:07.567	16:06:07.606	39
16:06:12.567	16:06:12.637	70
16:06:17.567	16:06:17.589	22

Hierbij zijn de data van het experiment. Om de gemiddelde latentie te berekenen gebruiken we (2). Eerst ga je de som van alle latentie berekenen. $94 + 30 + 46 + 83 + 42 + 86 + 96 + 61 + 23 + 41 + 39 + 70 + 22 = 733$. Er zijn dus 13 berichten dus delen door 13. Dan wordt het

$$733/13.0 = 56.4$$

Dus gemiddelde netwerk latentie in dit experiment is rond 56ms.

3 Conclusie

Latentie heeft invloed op de algehele snelheid van uw verbinding, maar veel toepassingen hebben manieren gevonden om deze te omzeilen. Videodiensten zoals Netflix bufferen bijvoorbeeld de gegevens die ze downloaden. Dit betekent dat ze een voorsprong hebben op het afspelen en extra videogegevens opslaan,

zodat het klaar is voor gebruik zodra het nodig is. Zelfs als er een vertraging of een plotselinge fluctuatie in de stream is, kan de video nog steeds soepel worden afgespeeld zolang er nog informatie in de buffer zit. Andere soorten activiteiten zijn minder vergevingsgezind. Online games zijn afhankelijk van wederzijdse interactie tussen de speler en de server. Latentie vertraagt je reactietijd, ook wel lag genoemd. Lag kan uw acties traag of wankel maken. Als je latentie te hoog wordt, kun je helemaal uit het spel worden verwijderd.Het doel was dus om te onderzoeken of de netwerk latentie laag genoeg is voor praktische applicatie. Uit dit experiment ligt de netwerk latentie rond de 56 ms. Met 56 ms latentie zit het aardig goed voor praktische applicatie, je kan zelf soepel online gamen met anderen waar latentie veel belangrijker is.

References

[1] Marco Anisetti, Valerio Bellandi, Alberto Colombo, Marco Cremonini, Ernesto Damiani, Fulvio Frati, Joêl T Hounsou, and Davide Rebeccani. Learning computer networking on open paravirtual laboratories. *IEEE Transactions on Education*, 50(4):302–311, 2007.

[2] Tomohisa Uchida. Hardware-based tcp processor for gigabit ethernet. *Nuclear Science, IEEE Transactions on*, 55:1631 – 1637, 07 2008.

[3] Florian Voigtländer, Ali Ramadan, Joseph Eichinger, Claus Lenz, Dirk Pensky, and Alois Knoll. 5g for robotics: Ultra-low latency control of distributed robotic systems. In *2017 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, pages 69–72. IEEE, 2017.