IIA2017: Industrial IT

**Assignment 1: Data Communication**

Isak Skeie, 245362

Contents

Contents 2

1 Introduction 3

1.1 System Information 3

2 Network information 7

2.1 Theory Description 7

2.2 Network info 7

2.3 Wireless 9

3 Network testing 12

3.1 Ping Application 13

3.2 Pseudocode for average ping response 14

3.3 Pseudocode for checking all nodes in network 15

4 Conclusion 17

5 References 18

Appendices 19

# Introduction

As stated in the assignment specification. The purpose is to get experience and become familiar with I/O devices, setup of network devices, and to test the network on a personal computer. This is done by looking through device settings on windows, experimenting with the ping utility, as well as performing tasks surrounding the OSI model. The tasks are performed with a ThinkPad P15 laptop, supplied by work. During the assignment the laptop have been used at different locations. This gives inconsistent answers to the network and device tasks. As the laptop connects to different networks, and in some cases with a docking station.

## System Information

Figure 1 shows the Interface protocol NVM being used between the storage device and the motherboard. Figure 2 shows the us the size of the storage device being 475.69 GB, with 33% of this being available.

Graphical user interface, text

Description automatically generated

Figure 1: Device Manager; interface between motherboard and storage device

Graphical user interface, text

Description automatically generated

Figure 2: Disk Management

An extensive list of serial ports is shown in figure 3. The large number of serial ports are due to the dock the laptop is connected to at work. Here two additional screens are connected, which have two additional USB-ports each.

Figure 3 shows a list of Bluetooth devices that’s been connected to the laptop.

Graphical user interface, text

Description automatically generated with medium confidence

Figure 3: Overview of serial ports

Graphical user interface, application

Description automatically generated

Figure 4: List of Bluetooth devices

# Network information

<Start each chapter with a short introduction about what the chapter is about.>

## Theory Description

The ping-test utility is used for network testing and diagnostics. It uses the Internet Control Message protocol in the network layer to send Ip-Packets with an echo request. The echo request makes sure several packets are sent back and forth. This makes it possible to confirm if a host in a network is active, the speed/response -time of data travel, and the packet loss in the connection. [1] [2]

## Network info

Figure 5 shows a list of network devices from Device Manager. The large number of network devices are due to the docking station at work.

Figure 6 shows a PowerShell window displaying the output from the command *ipconfig -all*. Theres several physical addresses, one for each network device, with the one connected to the local network being the one of interest. At the time of sending the ipconfig command the laptop was connected to the local network through Ethernet adapter Ethernet 3, which has a MAC-Adress of 08-3A-88-5C-D8-2A.

Because of the IP-Address we can assume the connection is using the TCP/IP protocol. The first four bits in the IP-address tells us it’s a class B address, with the subnet-mask being 255.255.255.0. The ipconfig /all command also gives us the IP-address for the DHCP-server (Dynamic Host Configuration protocol), this implies that the address assigned to the ethernet adapter is dynamic.

Other than TCP/IP, other protocols on the computer are UDP for real time data transfer, POP for receiving emails, smpt sending emails, http/https for web use, FTP. [3] [4]

From figure 6, the information provided by ipconfig /all in PowerShell, under Ethernet adapter Ethernet 3 the IP6-adress is listed as Link-Local IPv6 address with the address being fe80::c4e5:e05f:7479:e7e2%21. This tells us that the computer supports IPV6.

Graphical user interface, text

Description automatically generated

Figure 5: List of network devices from Device Manager

Text

Description automatically generated with medium confidence

Figure 6: Local network information in powershell

## Wireless

In figure 1.2 from the assignment paper. The first option seems to be the ideal one, with the wireless signal being the strongest, and having the fewest connections.

From Figure 1.2, its unclear if a password is necessary. This is due to the Encryption being unknown for the network.

In figure 1.2 from the assignment paper, there are no other networks shown because the network module being used by the computer does not extend past the adjacent layer of the OSI-Model. With the network-module being layer 2 (Data link layer), and the access points being layer 3 (Network layer).

Assume owners, location, security. In the list in Figure 1.3 in the assignment paper. We can assume Thamm is the name of the owner of the network, as it is a computer-to-computer connection. Due to the bad signal strength, we can assume there’s a significant distance between the nodes, we can also see that the connection is unsecured which is unwise, especially to a computer. The network CM have security, but not specified which type. The signal strength is very good, indicating the access point is close. AB Stargate1 have the same properties as CM, with a lower signal strength. AB ext. and AB Stargate have the same properties as CM and AB Stargate1 but with a bad signal quality, indicating they are further away. AB Skywalker have a medium to low signal strength, and WPA security. WPA is better than the security of the other options, something like WPA2 should have been employed. [5]

In figure 7 the command *netsh wlan show network mode-ssid* is used to list the available wireless networks in PowerShell.

A picture containing text

Description automatically generated

Figure 7: List of available wireless connections

# Network testing

By first pinging Aftenposten.no, we it has a good connection with a short response time of 2ms. Trying Alibaba.com (assuming the server is further away from this computers location). There’s also a good connection, but with a significant higher response time. The results can be seen in figure 8.

Text

Description automatically generated

Figure 8: Ping test applied to two different websites

By using *ping Alibaba.com -t,* the ping command is send continuously until stopped. When opening WireShark and filter for ICMP, its possible to view the ping requests. In the header of each packet, the destination IP-address for Alibaba.com is visible, being 47.246.137.166. When clicking on a packet its possible to view details for that specific package.

Application, table

Description automatically generated

Figure 9: Monitoring Ping with WireShark

## Ping Application

By adding a website and a personal data-string, its possible to apply the ping application and use it for testing, the website used for this program was Alibaba.com, and the personal data-string being ”sitterpaatogettilbergenpaaenfred”. The results gets displayed in a console, as seen in figure 10. By monitoring the network at the same time with WireShark, its possible to pick up the ping package and view its content. As seen in figure 11.

Text

Description automatically generated

Figure 10: Result from the ping application with a induvidual host, and datapacket

Graphical user interface, text, application, email

Description automatically generated

Figure 11: Packet information from ping viewed in WireShark

## Pseudocode for average ping response

The pseudocode below shows how you could calculate an average of several responses. This is done using the source code for the Ping Test Utility as a class.

|  |
| --- |
| using System;  namespace PingAverageResponse  {  class Program  {  static void Main(string[] args)  {    Initialize host-string;  Initialize data-string;  Initialize sum-int;  Initialize average-double;  Initialize responses-int;  Declares array of classes with PingPackage;  for( Iterates through the number of responses)  {  Initializes classes for pingRespons;  Adds the response from class PingResponse to sum;    }  Calculates average: sum divided by (number of responses minus packages not received);    Write average to console;  }  }  } |

## Pseudocode for checking all nodes in network

As with the pseudocode for the Average ping response, the source code for Ping Response utility code is used to check whether it’s a valid IP-address or not. The source code for this console application can be found under Appendix B.

|  |
| --- |
| using System;  using System.Net.NetworkInformation;  using System.Net;  using System.Collections.Generic;  using PingUtility;  namespace NodeOverView  {    public class Program  {  static void Main(string[] args)  {  Initilize data-string;  Initialize ip-address string;  Initialized empty ip search string;  Create bool for ping test result;  Create Lost of strings for storing the available nodes;    Create new instance of ping testing class;  Write header for console, declaring the found nodes;  for(Goes through all the available network address within the subnet)  {  Creates ip from ip-adress and iterative number;  Send Data and ip to ping test instance;    if(Ping Test is succesful)  {  Add ip to List;  Write ip to console;  }  }  }  } |

# Conclusion

The tasks performed throughout this assignment have highlighted the communication capabilities of a regular laptop, as well as going into the fundamentals of networks and aspects of the OSI-model. By gathering network information and testing networks, you get insight into the different aspects of layer 2, and 3 of the OSI-Model. This is knowledge that’s fundamental for an automation engineer to grasp. The importance and use of this knowledge is shown in the tasks with the ping applications. Exploring relevant custom tools an engineer might need, as well as the occurrence of vulnerabilities in communication systems.

Network tools like the pingAppConsole, WireShark, ping and ipconfig are valuable and crucial for an engineer. For someone to be able to use these tools, its important to understand the fundamental aspect of communication and networking, as well networking theory surrounding them.

# References

|  |  |
| --- | --- |
| [1] | Cisco, «Understanding the Ping and Traceroute commands,» 03 02 2022. [Internett]. Available: https://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/ios-software-releases-121-mainline/12778-ping-traceroute.html. |
| [2] | A. Zola, «Ping,» 3 2 2022. [Internett]. Available: https://www.techtarget.com/searchnetworking/definition/ping. |
| [3] | w3schools, «Types of network protocols and their use,» 2 2 2022. [Internett]. Available: https://www.w3schools.in/types-of-network-protocols-and-their-uses/. |
| [4] | P. Predamkar, «Types of networking protocols,» 2 2 2022. [Internett]. Available: https://www.educba.com/types-of-networking-protocols/. |
| [5] | pandasecurity, «wpa-vs-wpa2,» pandasecurity, 20 1 2022. [Internett]. Available: https://www.pandasecurity.com/en/mediacenter/security/wpa-vs-wpa2/. [Funnet 3 2 2022]. |

Appendices

Appendix A <Ping source code based on

Microsoft MSDN1 information.>

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Threading;  using System.Net;  using System.Net.NetworkInformation;  //  namespace PingUtility  {  class Program  {  /// <summary>  /// //////////////////////////////////////////////////////////////////////////////  static void Main(string[] args)  ///  /// Purpose: the main function in the application handling the ping()communication  ///  /// Version: 1.0: 8-JAN-17: NOS  /// </summary>  {  string host, data;  byte[] buffer;  int timeout;  Ping pingSender = new Ping();  PingOptions options = new PingOptions();  // Use the default Ttl value which is 128,  // but change the fragmentation behavior.  options.DontFragment = true;  // Create a buffer of 32 bytes of data to be transmitted.  data = "sitterpaatogettilbergenpaaenfred";  buffer = Encoding.ASCII.GetBytes(data);  timeout = 120;  // Name or address of node to access  host = "www.alibaba.com";  PingReply reply = pingSender.Send(host, timeout, buffer, options);  if (reply.Status == IPStatus.Success)  {  Console.WriteLine(" Ping communication status for {0}:", host);  Console.WriteLine(" ------------------------------------------");  Console.WriteLine(" Address: {0}", reply.Address.ToString());  Console.WriteLine(" RoundTrip time (mSec): {0}", reply.RoundtripTime);  Console.WriteLine(" Time to live: {0}", reply.Options.Ttl);  Console.WriteLine(" Dont fragment: {0}", reply.Options.DontFragment);  Console.WriteLine(" Buffer size: {0}", reply.Buffer.Length);  Console.WriteLine(" ------------------------------------------");  }  else  {  Console.WriteLine(" Error connecting to network address/name {0}", host);  }  Console.WriteLine(" Press CR or Enter to Quit the application");  Console.ReadLine();  }  }  } |

Appendix B <Code for Node finder >

|  |
| --- |
| using System;  using System.Net.NetworkInformation;  using System.Net;  using System.Collections.Generic;  using PingUtility;  namespace NodeOverView  {    public class Program  {    static void Main(string[] args)  {  /\* GetIp used to recieve ip-adress of the wifi-interface  GetIp();  \*/  string data = "sitterpaatogettilbergenpaaenfred";  string ip = "10.0.0.";  string ipSearch = "";  bool PingTest;  List<string> Nodes = new List<string>();  PingPackage ping = new PingPackage();  Console.WriteLine("The nodes at the local network are:");  for(int i = 0; i<254; i++)  {  ipSearch = ip + i.ToString();  PingTest = ping.Ping(data, ipSearch);    if(PingTest)  {  Nodes.Add(ipSearch);  Console.WriteLine(ipSearch);  }  }  }  static public string GetIp()  {  string hostName = Dns.GetHostName(); // Retrive the Name of HOST  string myIP = Dns.GetHostByName(hostName).AddressList[6].ToString();    Console.WriteLine("My IP Address is :" + myIP);    Console.ReadKey();  return myIP;  }  }  } |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Threading;  using System.Net;  using System.Net.NetworkInformation;  namespace PingUtility  {  public class PingPackage  {  byte[] buffer;  int timeout;  private string data;  private string host;  public int response;  public static int conErrors = 0;  Ping pingSender = new Ping();  PingOptions options = new PingOptions();    //options.DontFragment = true;  public bool Ping(string dataSet, string hostSet)  {  data = dataSet;  host = hostSet;    buffer = Encoding.ASCII.GetBytes(data);  timeout = 120;  PingReply reply = pingSender.Send(host, timeout, buffer, options);  if (reply.Status == IPStatus.Success)  {    return true;  }  else  {  return false;  }  }  }  } |