IN2011 Computer Networks : Coursework Resit

July 7, 2025

This coursework assesses your ability to use what you have learnt about networking to go from a description of a protocol to functioning software that interacts with network services. This includes the following:

- Being able to read and understand the specification of a application layer protocol.
- Understanding the IP network layer and UDP transport layer.
- Using Java to implement the application layer protocol.
- Use of `wireshark` to understand and record network traffic.

Because this is a second year unit, we will *assume* that you have the following skills from the first year:

- Can write Java programs up to a few hundred lines long.
- Be able to test your software so that it is robust against a range of inputs and errors.
- Be able to build and run your software on a variety of different operating systems and configurations.
- Understand what a *tree* is an how *recursive* algorithms work.
- Can work with data that is represented in a binary format.

You will not get marks for these but you need to know them to do this coursework. If you do not feel confident about these please revise your notes from last year and talk to the staff running the support sessions or your personal tutor.

1 Tasks

- Read RFCs 1034 and 1035. Students may use other RFCs for guidance but are only required to understand and implement the protocol as described in RFC 1034 and 1035.
- Download the starting `git` repository, unzip it and carefully read the code.
- Complete `StubResolver.java` so that it can:
  - Send and receive UDP packets using a `java.net.DatagramSocket` and `java.net.DatagramPacket`.
  - Perform recursive resolution using the DNS server given during configuration. Do not use existing Java libraies to perform the resolution. Do not fix a particular DNS server.
  - Perform queries for `A`, `NS`, `MX`, `TXT`, `CNAME` resource records.
  - Handle the case when the domain name is found and when it is not.
- Complete `Resolver.java` so that it can:
  - Perform iterative resolution starting from the DNS server given during configuration. Do not use existing Java libraies to perform the resolution. Do not fix a particular DNS server.
  - Perform queries for `A`, `NS`, `MX`, `TXT`, `CNAME` resource records.
  - Handle challenging and error cases including unresponsive and backup servers, invalid or irrelevant responses, `CNAME` records found during resolution, `CNAME` loops, glue records, etc.
  - Handle the case when the domain name is found and when it is not.
- Complete `NameServer.java` so that it can:
  - Accept queries from multiple simultaneous clients via UDP. Do not use existing Java libraies to perform the resolution. Do not fix a particular DNS server.
  - Maintain a cache of live resource records and negative responses, including the correct TTL and removal of expired records.
  - Answer queries using the cache and iterative resolution.
  - Handle error conditions including invalid and malicious requests.
- Submit a zip file containing:
  1. Your version of the `git` repository including the three classes you have to implement. Do not include class or binary files. If you have create any other objects or project files that are needed to build and run your submission these must be included. It *must* be able to be built and run on the course virtual lab machine.

2. A `README.txt` file that gives:
   - All of the instructions needed to build and run your objects.
   - Which functionality you think is complete and working.
3. `pcap` files recorded with `wireshark` that showing each part of your submission working.

- The assessment of your objects will be automated. For this your classes will be used in a larger system. So it is *vital* that you:
  - Submit Java files that build and run on the course virtual lab machines.
  - Test your code so that it does not crash.
  - Do not alter any of the code marked as `DO NOT EDIT`.
  - Do not require any local user input, for example, do not use `System.in`.
  - Do not put the code into a `package`.
  - Do not "hard code" the addresses of any servers, including the global root DNS servers.

  This is your responsibility and you will loose marks if you do not do so!

2 Academic Conduct

- *This is individual coursework.* The code that you submit *must* be *entirely* your own work.
- You *must not* use generative AI tools.
- You *must not* share your code with other students.
- You *must not* make your code available to other students. This includes making it a publicly accessible git repository, posting it to Discord or sharing it on your WhatsApp group.

If you don't follow these rules it will be treated as academic misconduct and may result in a range of sanctions.

3 Mark Scheme

Submissions will be marked out of a total of 100. Marks will be awarded for the following:

StubResolver.java

- Send and receive UDP packets. (4)
- Send a valid DNS recursive query. (4)
- Correctly extract the answer from a DNS response. (4)
- Handle the required query types. (4)
- Handle successful and unsuccessful queries. (4)

Resolver.java

- Perform a successful iterative resolution for a variety of different domain names and query types. (12)
- Handle `CNAME` records during resolution including loops. (6)
- Make use of glue records during resolution. (6)
- Handle unresponsive name servers. (6)

NameServer.java

- Handle multiple simultaneous clients. (6)
- Cache queries and negative responses. (6)
- Correctly handle expired resource records. (6)
- Defend against malicious and spoofed responses. (12)

Wireshark Does your recording show all of the features working correctly? (6)

Robustness and Quality A final (14) marks will be awarded for code quality and robustness. This includes handling of error conditions, network faults and protocol errors.

4 Deadline

Friday August 8^{th} 2025 17:00

5 Hints

This section is not part of the coursework. You do not have to do any of the things described here. It gives some advice and ideas which might be useful.

- The tool `dig` we used during the lectures and practicals can perform recursive and iterative queries. You can use it to understand what your program needs to do. It is installed on the Azure virtual lab machines but is also available for most operating systems. A similar but older tool `nslookup` comes installed by default on many operating systems.
- Running `wireshark` during development will allow you to check the formatting of the packets that you send and receive. It will display the contents of the packet as hexadecimal, as ASCII and decoded. This may help you find which bits and bytes you need your program to use.
- There is some functionality which will be required in all three parts of the coursework. It is better to create methods or additional objects rather than copy and pasting code.
- The server you use for `StubResolver.java` must support recursive requests. The DNS server on most home routers will support this and you can use that. Alternatively you can use the Cloudflare public servers `1.1.1.1`.
- The server you use for `Resolver.java` should ideally be one of the root servers (`a.root-servers.net. 198.41.0.4`, `b.root-servers.net. 170.247.170.2`, ...).
- Some networks, including part of City's internal network, block the use of external DNS servers. If you are using one of these, you will have to use their internal DNS server `138.40.76.2`.
- There are some aspects of the protocol which are quite tricky. For example wildcards in domain names and requests for multiple record types. You have not been asked to implement these. If you want to you can but it is also fine to ignore these aspects of the protocol.
- You can also ignore parts of the protocol that are defined after RFC 1034 & 1035, for example EDNS, DNSSEC, TSIG, DNS Cookies, DoH, DoT, etc.
- As we discussed during the course, many operating systems restrict which user programs can use TCP and UDP ports under 1024. This means that you may need to run `NameServer.java` with increased privileges (such as using `sudo`) or run it on a higher, non-standard port. If you are using a non-standard port you may need to configure `Resolver.java`, `dig` or whatever tool you are using to test.

- `NameServer.java` is only required to be a caching and resolving name server. It does not need to be an authoritative name server for any zones. If you want you can implement this functionality and it may make it easier to test unusual, difficult and error situations.
- There are three testing programs which show how the interfaces will be used. You don't have to use these but they may make your testing easier. These are intended to help you start testing, they are not a complete or sufficient test suite on their own.
- If you have `NameServer.java` fully working, it should be possible to configure a web browser or other programs to use it as their DNS server. This can generate a lot of queries and will be a challenging test.