Data Networks 2011
Programming Lab

The goal of this programming lab is to allow you to apply what you learned about computer networks in the first three parts of this course in a realistic setting.

Working in Groups

You will work on the lab assignments in groups of three students. Your tutor will make sure that every member of the group participates actively in solving the assignments. Exchanging ideas between groups is definitely allowed; exchanging code or complete solutions is definitely not allowed and will be treated as an attempt of deception.

The Simulation Environment

To provide you with a realistic network programming experience without having to actually build large networks consisting of many physical computers to test your code on, we use the *cnet* network simulator, available at

`http://www.csse.uwa.edu.au/cnet/`

cnet has been developed at the University of Western Australia, runs on almost any Unix-like system (including Linux and Max OS X, but **not** Windows), is easy to install and use (including a nice graphical interface) and is well-documented on its website.

If you have any problems with or questions regarding *cnet* that are not covered on its website, please ask your tutor or any other member of the Data Networks team, but **do not contact the authors of *cnet*** with questions regarding your programming lab assignment!

cnet simulates a physical layer and an application layer. It will be your task in this lab to program the intermediate layers in order to provide a certain service to the application layer. Yet, you are not required to re-implement the entire Internet, since we are in a particularly local situation…

The Setting

In an unfortunate accident and for no adequately explored reason, you and your Data Networks Programming Lab team have time-travelled to a Saarland before the time of computer networks. Equipped with the knowledge from your last computer science course, you decide to make a fortune connecting the people of Saarland with a fast, efficient, and dependable network.

Step-by-step, you build an infrastructure of point-to-point wide-area links between the major and, later, minor settlements in the area. You leave the dirty job of providing local connections between the different users – companies, research institutions, public offices, and private homes – and your "access hubs" to other local entrepreneurs and simply call them "application layer". To you, all they do is provide pairs of a destination hub address and a chunk of data – a "message" – which you guarantee to deliver to the specified destination as fast as possible, without loss or corruption, and, for a given combination of source and destination hub, in the order that you received them.

First Milestone

Estimated fraction of total work: < 5 %
Topology files: `saarnet-1.txt`

In order to become profitable as quickly as possible, you first connect the cities of Saarbrücken and Homburg, expecting in particular the developing Saarland University to generate significant traffic and thus revenue. Your first connection is rather slow at 56 kbps, but comes with the advantage of being extremely reliable: Your telecommunications equipment manufacturer guarantees that this connection will never lose, corrupt or reorder any data.

Your tasks

- Download and install *cnet*. This includes reading the "Downloading and installing cnet" page on the *cnet* website. Before compiling *cnet*, modify the following values in `preferences.h`:
  - Line 208: Set GCC_PEDANTIC_WANTED to `true`
  - Line 250: Set MAY_TRUNCATE_FRAMES to `true`
- Read the documentation on the *cnet* website and become familiar with the tool, its simulation model and its interfaces.
- Write the code for the intermediate layers to power your access hubs in Saarbrücken and Homburg. Saarbrücken has lots of data to send, so you should maximise the throughput on the link.

How to complete this milestone

1. Write the necessary code and test it. Running `cnet saarnet-1.txt` must be sufficient to simulate your solution in *cnet*.
2. Arrange a date for the discussion of your solution with your tutor (**no later than 22 July**). Send him a `.zip` file that contains your solution. The file must be named `GroupX-M1.zip` (where X is your group's number), and your solution's base file must be contained in the file's topmost folder (i.e., `milestone1.c` must not be inside any subdirectories). Files that do not conform to this layout will be rejected by your tutor.
3. On the arranged date, your group meets with your tutor to discuss your solution and answer the tutor's questions about the problem and your solution. If your solution works correctly and your tutor is convinced that every team member understands the solution and has actively participated in its creation, you will have completed this milestone.

Hints

- Reading "a walk through some introductory protocols" on the *cnet* website may be very helpful.
- Use the CHECK(…) macro extensively. We only consider implementations correct that do not cause failures when CHECK is used for every call to *cnet* application layer API functions.
- Do not modify the topology files. We use the original files to test your implementation.

Second Milestone

Estimated fraction of total work: < 15 %
Topology files: `saarnet-2.txt`

It quickly becomes apparent that the capacity of your link between Saarbrücken and Homburg is not sufficient, so you upgrade to new technology that provides a bandwidth of a stunning 1 Mbps – with one drawback: The link's MTU is extremely low at 96 bytes.

You also realise that you will soon want to expand your network to other places within Saarland, and as you know from your CS studies, a good design makes a good implementation significantly easier. Therefore, you also start designing your expanded network now.

Your tasks

- Adapt your code from the previous milestone to work with the new link – again, maximising throughput.
- Based on the requirements of the third and final milestone, create a preliminary design of the layers and protocols you will use to complete that milestone. Think about the services your layers provide and, for the protocols, include brief descriptions of their header formats.

How to complete this milestone

1. Write the necessary code and test it. Running `cnet saarnet-2.txt` must be sufficient to simulate your solution in *cnet*.
2. Arrange a date for the discussion of your solution with your tutor (**no later than 19 September**). Send him/her a `.zip` file that contains your solution as well as a `.pdf` or `.txt` file that contains your layering and protocol design. The `.pdf` or `.txt` file must be named `GroupX-M2.pdf`/ `GroupX-M2.txt`, and the `.zip` file must be named `GroupX-M2.zip` (where X is your group's number), and your solution's base file must be contained in the file's topmost folder (i.e., `milestone2.c` must not be inside any subdirectories). Submissions that do not conform to these requirements will be rejected by your tutor.
3. On the arranged date, your group meets with your tutor to discuss your solution as for the first milestone.

Hints

- Find out what the maximum size of a message generated by *cnet*'s application layer is. What is the overhead in terms of header information added by your protocols?
- Have another look at the lecture slides about protocol design before you start working on the design document.
- Re-read the first page of this assignment if you are unsure what service you need to provide to your customers (the "application layer").

Third Milestone

Estimated fraction of total work: > 80 %
Topology files: `saarnet-3.txt` and all others released in the CMS

It's now time to expand your network beyond the single Saarbrücken-Homburg connection. You first hook up Saarlouis, which makes the access hub in Saarbrücken your first router, and then extend to other large and small places with demand for network connectivity in Saarland. In order to keep costs under control and because the 1 Mbps link with 96 byte MTU was really annoying, you use a different technology for your expansions, which is significantly cheaper, but also has the tendency to lose and corrupt messages from time to time. You also use links of varying capacities to reduce costs when connecting smaller places at the borders of your network.

Your tasks

- Write the code for the intermediate layers to power your network.
- Keep your design document in sync with your implementation.

How to complete this milestone

1. Write the necessary code and test it. Running `cnet` *<topology file>* must be sufficient to simulate your solution in *cnet*.
2. Arrange a date for the discussion of your solution with your tutor (**no later than 26 September**). Send him/her a `.zip` file that contains your solution as well as an updated `.pdf` or `.txt` file that shows the layering and protocols you actually implemented. The `.pdf` or `.txt` file must be named `GroupX-M3.pdf`/ `GroupX-M3.txt`, and the `.zip` file must be named `GroupX-M3.zip` (where X is your group's number), and your solution's base file must be contained in the file's topmost folder (i.e., `milestone3.c` must not be inside any subdirectories). Submissions that do not conform to these requirements will be rejected by your tutor.
3. On the arranged date, your group meets with your tutor to discuss your solution as for the previous milestones.

Hints

- We will simulate your solution on some additional, completely different topology files. Do not hardcode any decisions for the given topology files.
- In all the topology files we provide, hosts do not fail. You can assume that this holds for any additional topology files that we use for testing as well.

Getting a bonus

If you complete the programming lab, you may get a bonus in the range $[0,1]$ on your final grade number for the course depending on the following qualities of your solution for the third milestone:

- Performance
  - Up to 7 points depending on the performance of your implementation in terms of end-to-end throughput, latency and overhead.
- Design
  - 1 point for a high-quality layering and protocol design: Are the services well-defined, are concerns separated, is duplication of functionality kept to a minimum, …?
  - 1 point if you respected rule 10 of the ten rules ("commandments") of protocol design.
  - 1 point if your initial design held up well during the implementation phase (requires early completion of the second milestone).
- Code quality
  - 1 point for well-structured, well-documented and readable code.
  - 1 extra point if the code quality is exceptionally good.

Your bonus will be calculated as the number of points you achieved according to this list divided by the maximum number of points achievable.