Lecturer
Prof. Dr. Christian Tschudin (christian.tschudin@unibas.ch)

Tutors
Carlos Andrés Tejera (carlos.tejera@stud.unibas.ch)
Andreas Wassmer (andreas.wassmer@unibas.ch)

Ausgegeben am
Do 14. April 2022

Abgabe bis
Mi 27. April 2022
auf https://adam.unibas.ch

Modalities

For this exercise sheet, we expect you to hand-in written solutions and a partial presentation in a online meeting. Besides explanations and answers to questions, delivered as PDF, your hand-in should also contain the detailed documented source code. Furthermore, for execution tasks not required to be presented (noted in the questions) a list of all executed commands and generated outputs (as screenshots if relevant) must be provided.
The solutions can be handed-in in groups of two. The presentation will take place in the week of the deadline or the following depending on available time-slots. All hand-ins (more than a single PDF) should be compressed into a single zip-file.
Programming implementations can be executed as **C++** or **Python3**, as you prefer. The handed-in source code has to be well documented containing :

- the description of the idea and realisation
- methods/function call description, i.e. the target and call arguments
- points of complex / difficult implementation
- emphasis points as asked by the task description
- boundary conditions , limitations and in-/valid values
- makefile or instructions for compilation via README file

The OS target system for code implementations shall be a Linux distribution. Please, implement the code in library/module style (called for a second file with main() for C++ or by Python3 package standards[1]).

[1] `https://packaging.python.org/tutorials/packaging-projects`

Task 1 - Routing - Are We There Yet?! AWTY?! AWTY?! AWTY?! ... (9 Points)

In this exercise, we practically experiment with routing and forwarding. To this avail, we will consider a simplistic routing setup, where each nodes knows its connections and their speed absolutely. For simplicity, we do not consider external influence or impact on the connections other than one for simulation administrative purposes. The latter distributes the information trustworthy to all nodes, such that we can simulate different and changing scenarios. This setup also allows to observe local simplistic TCP communication that is asked for in other tasks. You can use it.

Task :

- For any choice you take, write it down and explain your choice.
- In the following, we just use (IP-)addresses as connection information, yet we silently assume port numbers to be a part of it (especially if nodes are running on the same machine (hence same IP) but different ports).
- Nodes must be distributed **across at least 3 machines** (2 VMs and 1 host)
- Although we suggest using TCP connections, you can also use UDP. But keep in mind, you then need to add a primitive level of reliable transport to it. Please **explain why** you use **TCP or UDP** for this exercise.
- Do not use multi-processing as it creats several potential conflicts. (E.g. serialise update accesses , order network update packages)

α) SubTask - Basic Setup (3 Points)

In this step, we build the essentials for simplistic routing network. Each peer, i.e. **node**, is a basic **TCP listener**. Received packages are evaluated and processed. All outgoing communication is a reaction to incoming packets.
Additionally, we design a central **command node**, which is only launched to **send TCP** messages. Its basic task is to send out the **network topology updates (NTU)**, i.e. the connections of each node and their RTTs, and **new messages**. The NTUs contain connections (or updates to them) of each node and their RTTs. New messages have an (arbitrarily) chosen origin, i.e. starting node, and an (arbitrarily) chosen target node (both delivered by the user, like the message itself).

Peer Node :

- Prints **message-content to terminal only at target node**, i.e. output for messages only addressed to processing node. For easier identification, the node adds its own name to the print-out.
- Messages addressed to another node are forwarded. The debug-info "forward" is printed to terminal.
- **Network update (NU)** messages are forwarded to all connections except in the case, that the update did not contain any better connection than already in this nodes forwarding table. (update process has an end)
- Think about connections with multiple optimal paths. How would you handle it, or is it necessary to handle it at all (implicitly handled)?
  Hint : Consider NU forwarding conditions.
- A **NU**, i.e. relay of own connectivity to other nodes, is **executed** by a node **when updating** its own connections and connection RTTs, or updating its routing information (gossip protocol). Keep in mind that many fast consecutive changes in a network topology, like artificially updating all nodes as we do here by the control node, can cause chaotic update behaviour. Counter measures have to be applied!
  Hint: We suggest using a "finale" keyword with the last node being informed (NTU) to initialise an NU outgoing from this node (update wave).
- **Each nodes** creates its **own forwarding table** for packages based on its connections and the NU messages.
- You can choose freely the **routing algorithm** to find to**calculate the forwarding table**. We suggest implementing the distant vector algorithm. Its by far the easiest to implement.
- For easy forwarding and routing, each node knows the name of all other network nodes. Assigning connections to node names contains the forwarding table.
- Each node only knows the (IP-)addresses of its own neighbours, i.e. connections.
- The message-content is limited to a simple text string.
- Only entire messages are forwarded here for simplification, not packet-wise like commonly done.
- The **own name** of each node is **given/assigned by the NTU** message. This allows default node programming and dynamic changes to the network.
- Each node only establishes a connection for the time to transmit a message. Afterwards, the connection is closed immediately again. (Otherwise, dead-lock possible during processing of NUs and blocking communication line though network is troublesome)

Control Node :

- It is a purely artificial node for simulation here. It is a pure TCP sender.
- This node only runs when called to send a message.
- It sends the NTU to all nodes.
- It initialises messages to be forwarded by the node network and received by a single node.
- It is the only node knowing all node (IP-)addresses, and hence, it can directly connect to all nodes.
- The node to introduce a message to the network can be chosen freely. Typically, this node is not the receiver, i.e. target, of the message.
- For simplicity, the network topology is read-in from a file in a simplistic raw format. In other words, it can be a list of node-names with IPs, and a list of node-names with connections to some other node-names, each with an integer number for its RTT. The exact format can be chosen freely.
- At any time, the central node can be called to transmit a NTU with a new file, to change the network topology to the one given by the new file.

Packages/Messages :

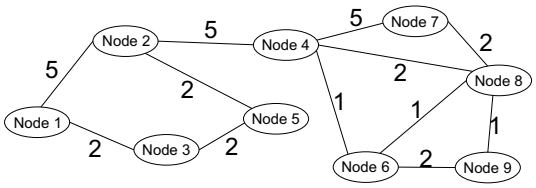
- The format of messages/packages can be chosen freely.
- Each simple message contains a target node-name and a message content (at least).
  Hint: For simplicity, we suggest a simple String-format that can be entered as argument to the control node binary.
- - Each NTU-package contains the (new) name of the receiver node, its connections, i.e. directly connected neighbours with their names and IP-addresses, and a list of all existing node-names in the network. Each package is designed for the receiving node specifically, and only contains the (minimal) necessary information.
  Hint: For easier processing, a sequence of messages each containing a part of the total NTU is acceptable as well.
- Each NTU does not need to contain updates for all nodes, with the exception of the first/initialising one, or when changing the number of existing nodes, node-names or IP-addresses.
- A NU is only sent to other nodes in the network. It contains the original sender (if necessary) and the (minimal) necessary information based on the routing algorithm.
  Hint: For the distance vector routing algorithm, that would be the (partial) forwarding table (node names + RTTs). (simple and compact)

Your task is to implement the peer/node and the control-node as described here.

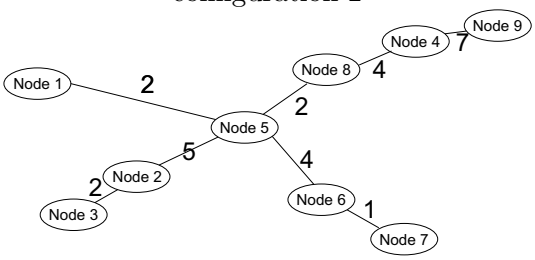
β) SubTask - Routing Test (3 Points)

You have the following 2 given configuration. Create files for NTUs, and run the required number of nodes. (Nodes can be copy-past run in default design if implemented as asked in α).)

configuration 1



configuration 2



Present your running routing network in your next exercise presentation running sequences similar to the following:

- initialise network with topology 1
- send messages starting at node 1 to node 9
- send messages starting at node 3 to node 7
- update topology with topology 2
- send messages starting at node 1 to node 9
- send messages starting at node 3 to node 7

γ) SubTask - Observation and Discussion (3 Point)

Show data, discuss your observations and explain :

(a) Use Wireshark or an equivalent on the machines running the nodes and the control node.

(b) Observe the TCP connections of each package type, identify the connection related packets (handshake, ACK) and the message related packets.

(c) Why using TCP connections and not UDP. Elaborate on necessary changes for switching (while keeping reliable transport), discuss potential problems and difficulties. Roughly, explain necessary changes to implementation keeping all discussed topics in mind.

δ) SubTask - Bonus - (0 Point, but a challenge)

(a) Implement a 2nd routing algorithm and present a second routing test with it.

(b) Allow dynamic updates : Instead of artificially print a network topology onto the nodes (via NTU), allow the network to grow node by node and connection by connection. The control node adds each connection separately allowing each node to register itself to the network and update the forwarding before the next connection. Expand the network update messages to allow registering new nodes (gossip protocol) and initialising a routing update.

(c) Configure and present the routing network with connection over the internet, i.e. connect nodes in one private LAN network to the nodes in another LAN network (e.g. LAN of your working group partner).
Hint: Potential solutions are via TCP port forwarding (Router configurations), you can read up on webpage1[2] for instance and a discussion of the potential risks on webpage2[3]. Another option is using VPN-like connection with Team-Viewer-VPN[4] for instance.

Task 2 - Wireshark - Snooping as a Profession (2 Points)

Using "Sniffer"-software like Wireshark, we investigate local web traffic even further.

α) SubTask - "I Spy with My Little Eye..."

To start easy, investigate the traffic from the previous implemented TCP routing. (If you had problems getting it running, you can use the Chat-Server from the previous exercise.)

- Observe the TCP connection initialisation and mark each step and identify it.
- Increase the transmitted message beyond the packet limit and identify its transfer. Can you read which packet contains which part?
- Bonus Task : Try to launch 2 or more messages with crossing paths, such that you can observe an intersection of connections. Can you observe repeated hand-shake trials?

[2] https://ncona.com/2013/02/making-your-local-server-accessible-from-anywhere/
[3] https://thetechnologyland.com/can-you-get-hacked-if-you-port-forward/
[4] https://community.teamviewer.com/English/kb/articles/6354-teamviewer-vpn

β) SubTask - "Big Brother is Watching You!"

In this section, we want to observe data transfers for their readability and focus on http.

The task of finding "http" queries becomes more and more difficult (as of 2022), so please excuse the language barrier : use the website http://so.news.cn/ (a chinese news agency) and query for the chinese translation of "Entwicklung Universität"[5]). The query will render about 2 results.
If you already switched to secure HTTPS DNS queries, temporarily change it back to a unsecure http one.
Hint : It also helps disablling as many other currently running web-services as much as possible. Running the browser and Wireshark inside a VM helps to reconfigure the web-services without crippling your actual OS.
Open your browser and observe your web connections with Wireshark.

(a) Start recording the packet exchange.

(b) Open the website.

(c) Enter the query.

(d) After the query-answer has finished loading, you can stop recording.

Task :

- Try to identify your DNS query to open the website
- Try to identify the http connection packets (categorize initialise, content).
- Try to identify the packets of your search query and try to raw-read their content.
- Elaborate and explain your findings. Also elaborate on switching to https and why.

[5]use online translater to get the chinese symbols for the query. (I simply could not get LaTeX to show them!)