Assignment 1: Overlay Networks

Erik Henriksson, Christoph Burkhalter

October 5, 2013

1 PART A

1.1 DESIGN OVERVIEW

The overlay network contains two kind of nodes, a coordinator node that organizes the network and member nodes. Each node can play both roles, however there can only be one coordinator node at any given time.

MAINTENANCE CHANNEL. A node is characterized by its network address (ip and port) as well as it's id that was assigned by the coordinator. To maintain the network, the coordinator sends and receives TCP messages like join request or members list updates. Therefore, each node has a TCP socket where it listens for incoming TCP request. The TCP socket is configured with the network address of the corresponding node and will run in a separate thread. This is called the *maintenance channel*.

PING CHANNEL. Additionally, there is the *ping channel* to check the status of the connection to another node. This channel is based on the UDP protocol, every node listens for incoming ping request and responds to them. These channel cannot run on the same network address, so the port number is increase by one in the *ping channel*. The listener socket is also executed in a separate thread to ensure fast responses to ping requests.

MAIN FUNCTION The overall design builds on three parallel parts, the handling of the maintenance channel, the ping channel and thirdly the main function. The main function contains code for both coordinator and member and is executed periodically. It contains the *heartbeat* function of the coordinator to check the liveness of the members, the check of the

members if the coordinator is still alive and the measurement of the latency between the nodes of the network.

1.2 Bootstrap

At start-up time a node is neither the coordinator nor a member. After initializing variables and log files, the node will first start the UDP socket The new participant will first try to connect to a existing network, if this fails he will start it's own overlay network. However, a node has to have an initial list of possible members of the overlay network to detect if there is already an established network. Therefore, each node requires a file (*seeds.txt*) to exist in the current working directory.
The new node sends a join request on the *maintenance channel* to the first network address in the file. If there exist a node with this address and it happens to be the coordinator of the network, the new node is added as a member and gets the coordinator id and the list of members in network. If the node that receives the message happens to be a member of the network, it returns the address of coordinator. The new node can then immediately send a join request to the coordinator.
Failing this, the new node tries the next address in the list. If all requests fail and therefore it cannot join a network, the node selects itself as coordinator with id zero.
As last step of the bootstrap phase the node starts listening on the *maintenance channel*, either as coordinator or as member.

1.3 Coordinator

The coordinator has two main functionalities, he pings periodically all members to check their liveness and he listens for messages on the *maintenance channel* like a join request.

1.3.1 Heartbeat

The heartbeat function implements the periodically ping requests. If a node doesn't responds twice in a row, it is treated as dead. The coordinator removes this node from the members list, logs a FAIL event and distributes the updated members list to all members of the overlay network. Additionally, the coordinator tries to send a message (*kicked out*) to the failing node.

1.3.2 Join request

1.3.3 Leave request

1.4 Member

A member can join or leave the overlay network by sending the corresponding message on the *maintenance channel*. However, a member needs to detect when the coordinator fails, therefore it remembers the last ping request from the coordinator. This is implemented in the UDP server handler and the main function of a node. Whenever a ping request arrives that contains the id of the coordinator, the *last_ping* variable is update to the current time. Periodically, the member checks in the main function if the *last_ping* time is older than a

certain threshold. If this is the case the members tries to reconnect to the coordinator or, if this fails as well, re-elect a new coordinator.

2 Part B

2.1 Coordinator reelection

- First item in a list
  - First item in a list
    - First item in a list
    - Second item in a list
  - Second item in a list
- Second item in a list

2.2 Latency measurement

Every thirty seconds each node calculates the latency to every other node in the network. This is implemented in the *measure_latency* function by sending a ping message on the *ping channel* to another node. Using this latency, the new average latency is calculated and both values are logged to the *latency.log* file. Every minute, the nodes sends array of average latencies to the coordinator. (TODO!!!) The node creates a separate log entry for failed pings, o this will not affect the average latency.
The coordinator does exactly the same, he also uses the *maintenance channel* to send the average latencies to himself every minute. The average latencies that are sent to the coordinator are loggend to the *pings.log* file. This file is only created at the coordinator node.