

**Distributed Systems – Course Work**

**<Assembly Gods>**

Henri Koski, 2190426

Ville Välimaa, 1981254

Janne Määttä, 1983508

Nemanja Vukota, 2190646

Introduction

This project’s purpose is to create an application which utilizes paradigms of a distributed system. Project aims to familiarize students with the developing of a distributed system etc. Our application called “Tracertmap” in its simplicity aims to provide trace routes for different URL’s and IP’s. Basically user can input an URL or an IP into input field and press “Trace” button. Trace route would then be displayed on the map.

Distributed System by 8 principles

In this section we will explain our distributed system with the 8 principles mentioned in the lectures.

## Architecture

Our application has a rather simple architecture:

Master node

n 1 1 n

Browser

Basically browser sends an URL or an IP to the master. Master then sends the information to all the nodes we have. Nodes then start to trace route to the target location. They memorize all the location points from the route to the target. After reaching the target they then send the data back to the master which aggregates the data and transmits the data to the client browser which then draws the trace routes. Master itself also does the trace routing. Furthermore, each node is identical to each other but their state is one of two possible (“master”, “slave”).

## Processes

Each node has one process in where the node.js is running.

## Communication

Our nodes communicate over HTTP requests. Each slave node has a heartbeat to the master node as well.

## Naming

DNS name resolving between browser and master node. When slave node becomes a master node, it points to the DNS to identify itself.

## Synchronization

Master node upkeeps a list of all the IP:s that are subscribed to it as a slave. When a new slave is subscribed and accepted it also gets the list of IP’s excluding the IP of itself. All of the nodes know their own IP, IP of the master and IP’s of all the slave nodes.

## Fault Tolerance

In case that our master node dies for some reason any of our slave node can take its position. This is possible because each slave node has a heartbeat to master node. If three concurrent heartbeat pings to the master fails, a slave node attempts to take its position.

## Security

All the data between the nodes is crypted with AES256-CTR.

Extra Points

* Early bird
  + This one is obvious, returned before end of 30.03.2015
* Naming
  + We use DNS discovery
* Security
  + Communication between nodes in encrypted
* Fault Tolerancy
  + Any node can become master if the master dies
* Distributed Synchronization
  + Use of timestamp to synchronize states of the node
* Communcation
  + Slave nodes subscribe themselves to the master node

Reflection

*Lessons learned and biggest issues:*

Process of selecting the master node. What happens when the master dies or when the slave dies? All in all the distributed way of developing an application.

Instructions

Note that you need to have git, node.js and npm installed.

Commands:

git clone https://github.com/nemonanja/ballin-octo-computing-machine.git

cd (git repo)

npm install

sudo node server.js