Info 408
Distributed Applications Programming
Project
Networked media sensors simulator

The purpose of this project is to study the behaviour of a networked multimedia sensors. The objective is to model this kind of network, define protocols for the communication between its elements, and implement a simulation architecture of such a network.

You have till Monday, May 31, 2021 to form your team, whereupon it will be too late and your project will not be corrected (it will be graded 0). One of the members of the team should upload on Teams (in the assignment named "SubmitYourNames") the full names of the different members of the group.

The end date to submit the project is scheduled for June 25, 2021 as described at the end sections of this document. The defence will take place starting from 26/06 (the schedule will be sent later)

---

1. Sensors

The term sensor is a priori very large. It is an autonomous element capable of acquiring information.

A multimedia sensor can be a webcam, a camera or a microphone that captures images and / or sound.

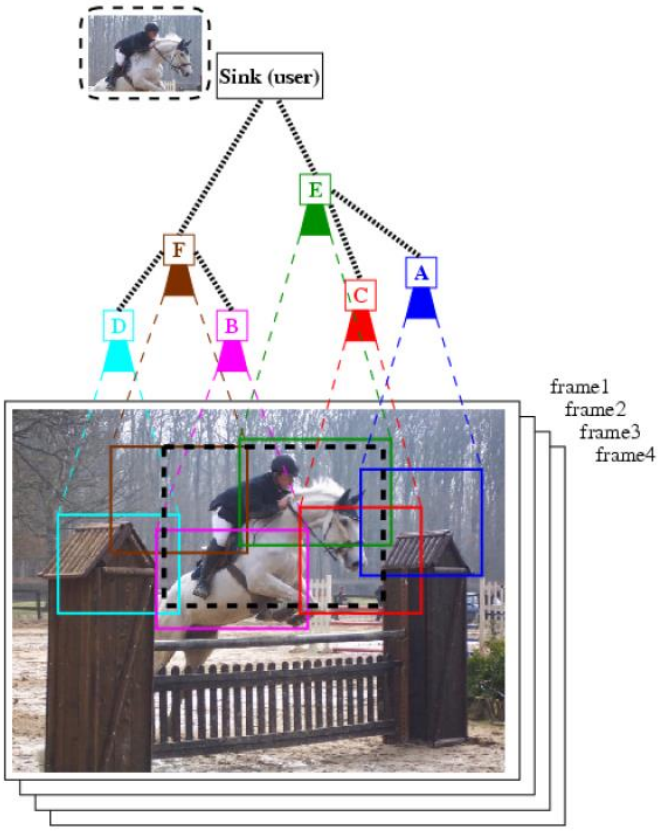
In this project, we consider a simple abstraction of a multimedia sensor as an application that has

a set of information representing an image or a sound, refreshed at a given time interval. For example, information can be images, with their spacial coordinates in a common referential (for example, the two opposite corners of the take) associated with a temporal information (the date of the take).

In real life, the images provided by the various sensors of the same scene would be with different

angles and distances. For simplicity, in the context of this project, we consider that all the sensors "look" at the same scene with the same angle and the same distance. However, each sensor can acquire only a part of the scene (the coverage zone of the sensor). Of course, it is possible that the acquired images by the different sensors overlap. The image shown on the next page, shows an example with 6 sensors (A, B, C, D, E and F) shown with their "coverage zone" of the scene in different colours.

Besides this ability to acquire images at regular time intervals, a sensor has the ability to communicate with other network elements. In the context of our simulation, a sensor is represented by a Java application run on a machine with an IP address and associated transport layer (TCP and UDP) in a LAN.



2. The network nodes

In real situations, the capacities of the sensors are highly variable in terms of acquisition, energy consumption and therefore autonomy, communication (baud rate), etc… In this project, we model these differences by several (at least two) levels of "third parties". Sensors of first tiers (base) are simply capable of transmitting (some or all) of the information they capture (A, B, C, D are examples in the figure). Sensors of higher tiers can additionally perform operations related to communication in the sensor network (route requests, responses, optimize or distribute the traffic …)

There is a particular node in this network called the sink, it is the node that initiates the queries and

receives the final resulted data. We can consider that he's the end user of the sensor network. Typically, the sink wishes to view images corresponding to a zone of the scene, and emits one or more queries in this regard.

3. A collaborative network

In general, each sensor sends regularly, information about its condition and its capabilities: if it is idle or not, what is the geographical area it covers, what is the area of the scene it covers, what is the frequency

to refresh its image, what is its availability (i.e. the load of work he is doing), what is his actual communication rate, etc … This information transmitted by the sensors are called descriptors.

Recipients of descriptors, as all the possible communications in a network of sensors, depend on the model of communication considered.

- In a hierarchical network, like the one shown in the figure, the communication takes place between parents and children. Sensors of first tiers (base, such as A, B, C or D) simply issue descriptors, receive and respond to requests. The sensors of higher tiers can also contact their parents or their children, but may also route information or attempt to optimize traffic, process images (data aggregation _ _ _), etc …
- In a mesh network, one can imagine that the lines of communication are not a tree but a graph. In the figure, for example, E and F may communicate with each other.

Beyond the simple exchange of descriptors, a sensor network must be able to satisfy one or more user requests. Various classical problems are then encountered:

- Can a given query be satisfied by a network in a given state (depending on the load of each node having to participate in the satisfaction of the request, the authorized ow, etc …)? This is a problem of ow admission.
- How can we optimize a query, especially in the case where multiple sensors can transmit the same required image zone? This is an optimization problem: cutting of an image and its reconstruction on an intermediate node or on a destination one…
- In the case of a mesh network, the existence of several routes between the points of image acquisition and the sink can help distribute the load over different communication links. This is a problem of load balancing.

One objective of this project is to perceive these issues and to test different approaches to solve them in a simulated environment.

4. The simulation

To simulate the sensor network and the different communication strategies, we would like to use different machines on which we will run Java applications representing sensors. Two "simulators" will be used.

- Data collector simulator. Rather than each application representing a sensor has a mechanism to capture the image and sound, these multimedia data will be provided directly to the application by a server. The same server can be used thus to provide the data needed for each sensor. In practice, each sensor represented by a Java application obtain data that is supposed to "capture" from the same data server.
- Configurator and supervisor simulator. Rather than having to manage, configure and supervise locally and independently each [application simulating a] sensor, these setup, configuration and supervising operations of the various sensors will be performed remotely by a supervisor. In practice, each sensor represented by a Java application communicates with the same supervisor server.

5. Required work

The required work consists of three applications and three different communication protocols.

1. Sensors. This Java application can actually be of different nature.

   a. a sensor of first tiers, the basic one;
   b. a sensor of higher tiers, with advanced features (we can a priori be limited to two or three tiers).
   c. a sink, requests issuer and responses recipient from the network (usually a sink is not necessarily a sensor but you can decide whether or not to make it a separate network element)

2. A supervisor server. This Java application must be able to record the network sensors that will be "started" in the form of a Java application on a machine. When registering, the supervisor server will be responsible to provide the sensor all of its parameters (coverage zone, capacity, communication links with its neighbours, etc _ _ _). In addition, during the life of the sensor, the supervisor server must be able to dynamically configure the state of the sensor (standby, wake, stop, restart, modify its surveillance area, the state of its communication links, etc _ _ _). It must also be able to collect various information from the sensor (status, current load, autonomy, etc _ _ _). The supervisor server is the only one who uses the network sensors simulator: the user of the simulator must be able to manipulate an interactive configuration interface (to configure or to supervise a particular sensor, to transmit a request by the sink or to retrieve the result _ _ _) or to "play" a predefined scenario described in advance in a form of a file.
3. A data server. This Java application simply provides each sensor with its concerned media stream (corresponding to its coverage zone, its capacity, its resolution, etc _ _ _).
4. A supervision protocol. This protocol should describe the format of the data that is exchanged between the supervisor server and sensors.
5. A data transmission protocol. It should describe the format of the data that is exchanged between the data server and the sensors.
6. A communication protocol. This should describe the format of data that is exchanged between the sensors in the network, whether to transmit descriptors, requests, responses, or optimize traffic, routing, etc _ _ _ The specificity of this protocol is that it must be kept at the sensor network level even if this latter is not simulated. In other words, in the absence of both data and supervisor servers, the communication protocol shall remain valid and allow running of a real sensor network (weather stations) offering the opportunity for a sink to send requests and receive its corresponding information.

The order in which applications and protocols are listed above is arbitrary. Nevertheless, it must be considered that the communication protocol and sensors (and sink) applications, that implement this protocol, are the 'basic' elements of the network.

In addition to the implementation of the communication protocol, and in order for the network simulator to be operational, the sensors should set up data transmission and supervision protocols, as well as setting up the respective servers.

6. Pre-rendering

The project is to be done by groups of 3 people.

For two weeks, you should think about the architecture of your applications and the different protocols used. Nothing prevents you from making small prototypes to validate the feasibility of this or that feature (it is strongly recommended), but you will not be asked to make code at this stage.

For Monday, June 7, 2021 at the latest, you must provide descriptions of the three protocols used, in RFC format (in English and in ASCII - text mode, between 10 pages for each protocol). Each RFC must clearly state the objective of the protocol, its operation principle and the format of the exchanged data (transport protocol, format, encoding, operation codes, etc...). You mustn't in any case make assumptions about how to achieve implementation of this protocol, in terms of programming language, or in terms of application architecture.

These documents (RFC files zipped into a single zip file RFC Name1Name2.zip, where the names are

those of the group members listed in alphabetical order) must be uploaded via the submission platform teams. The RFC file names must follow this convention, and each file must contain at its beginning the names of the authors:

- rfcData.txt for the data transmission protocol between the sensors and the data server.
- rfcSupervision.txt for the protocol governing the communication between the sensors and the supervisor server.
- rfcSensors.txt for the communication protocol between the sensors in the network.

7. Final report

During the next four weeks, you will make Java applications that meets the requirements. In terms of communication, these applications must follow the RFCs that you have set for the pre-rendering. In the case where the initially defined RFCs would not allow your applications to run and require revisions, you formalize the new revisions, and detail in an attached document all the modifications along with their justification.

For Friday, June 25, 2021 at the latest, you must provide your entire project as a zip archive (all rar, tar.gz, 7z and other _le types will not be accepted) containing:

- a src directory containing the sources of the project and any resources (images, sounds, etc, ...) to be copied along with classes;
- a docs directory containing a user manual (user.pdf) stating how to install, handle and use your applications as well as a manual that explains your architecture (dev.pdf). Both manuals must be in PDF format;
- a rfc directory containing the three original RFCs as they were delivered during the pre-rendering (in ASCII text format). In the case where changes have been made, you should provide also a modified RFC file version with the following naming convention XXX.revised.txt, and a document modifications.pdf to justify all the changes that you have been forced to make.
- executable jars: supervisor.jar for the supervisor server, dataserver.jar for the data server and sensor.jar for the sensors.

This zip archive must have the name Name1Name2Name3.zip, where the names are those of the group members listed in alphabetical order. Extracting the zip archive must create a directory containing all the requested elements. You should upload the zip file using the submission platform teams.

8. Defence

In your defence (which will take place end June 2021), you must use the zip archive uploaded via the submission platform and present a demonstration that you have prepared.

Date	Programme
31 May 2021	Form your team
07 June 2021	Submit RFC protocols
25 June 2021	Submit your project
26 June 2021 (To be confirmed later)	Project defence