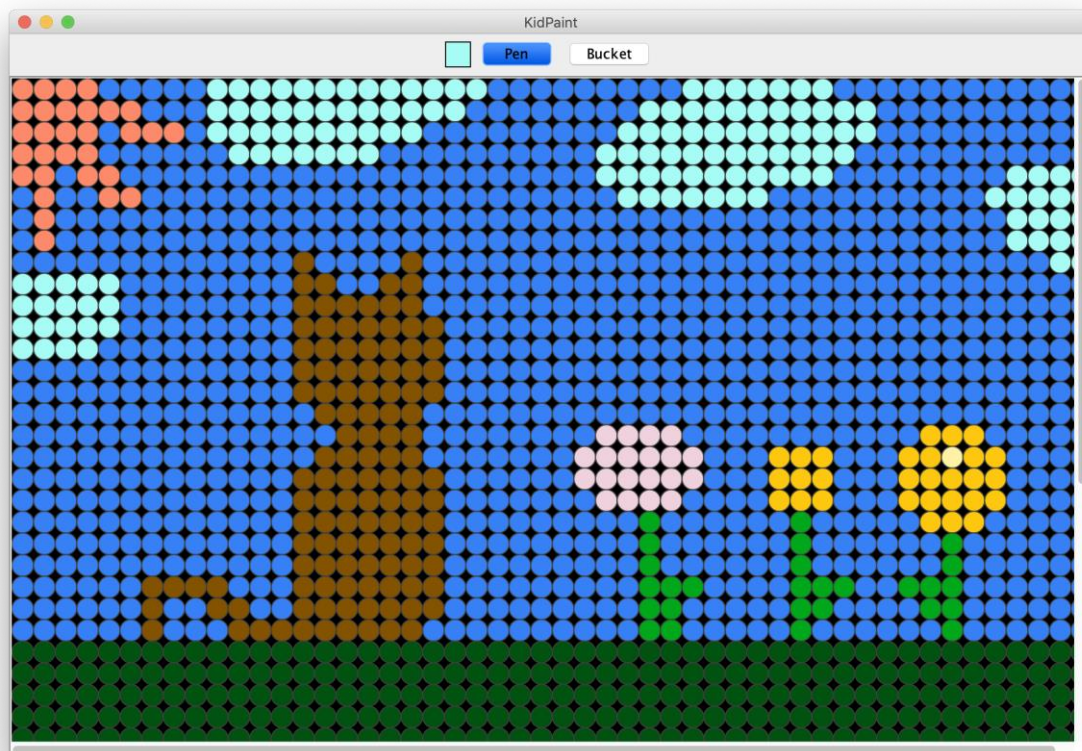


# COMP3015 Data Communications and Networking

## Project 2021-22

### Introduction

**KidPaint** is a paint app for kids. A kid can use a pen or bucket with different colors to draw and paint something on the sketchpad. The following is the layout of **KidPaint**:



### Problems

The vendor now wants to add some network features to **KidPaint**, so that multiple kids are able to draw on the same sketch at the same time through network connections. And, they can also communicate with each other. A message user interface with a key input event handler is ready in **KidPaint** but any network-related kinds of stuff are not yet implemented.

### Required Techniques

To complete the task, you are required to have the programming knowledge including basic Java UI, streaming, socket programming (TCP + UDP), multithreading, and basic OOP concepts.

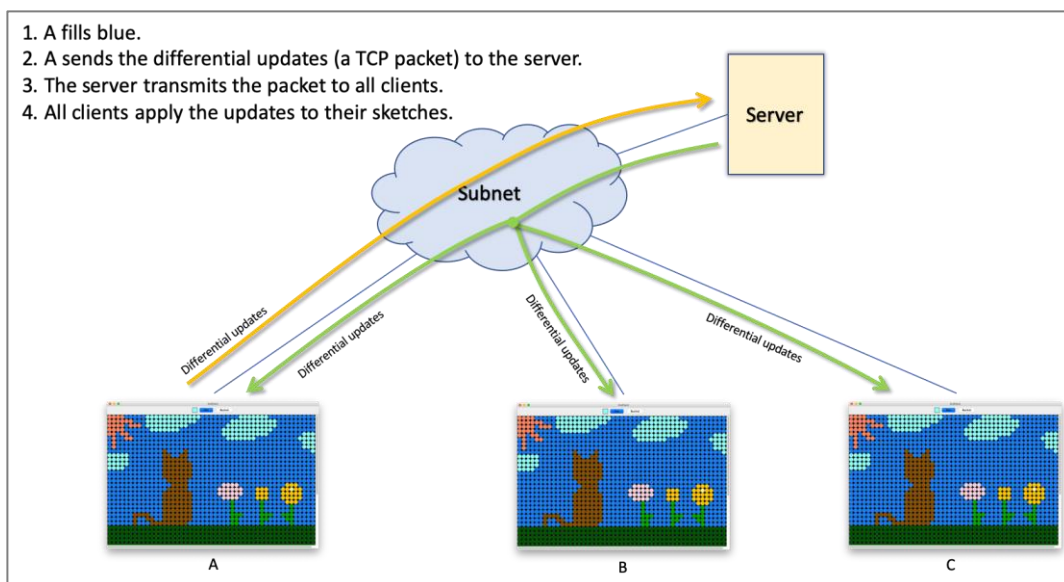
## Tasks

You have to perform the following:

1. Form a 2-member group.
2. Download the Java code of **KidPaint** from our course web page.
3. Add the network features to **KidPaint** for fulfilling one of the following approaches:
  - a. Basic client-server approach (max. 60 marks)
  - b. Advanced client-server approach (max. 70 marks)
  - c. Peer-to-peer approach (max. 80 marks)
4. Add additional features to **KidPaint**. (max. 20 marks)
5. Read the detailed requirements of each approach and explore the codes of **KidPaint** before starting the design and implementation.

### Basic client-server approach (max. 60 marks)

Your version of **KidPaint** should fulfil the following requirements:



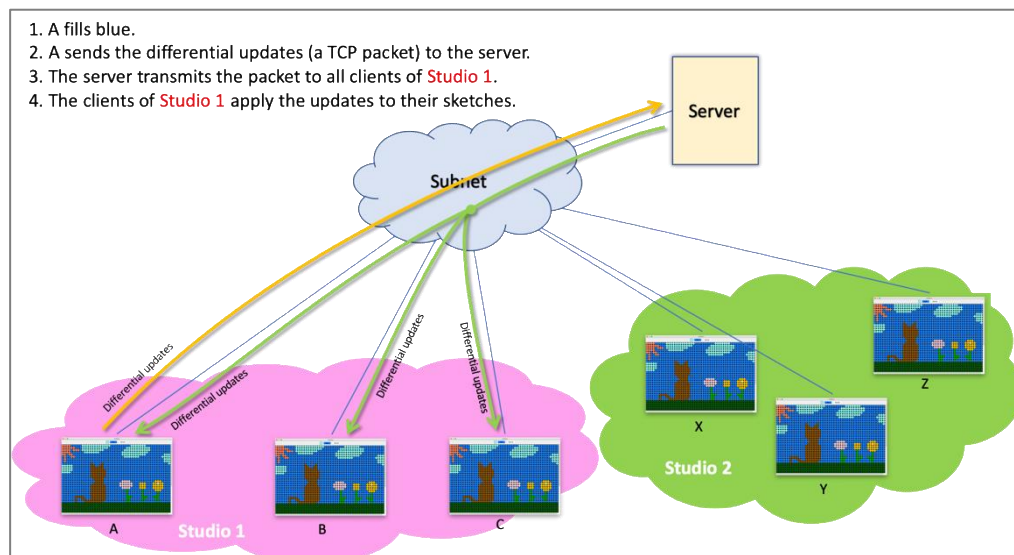
1. A server runs on the same subnet. The server-side program does not require any GUI.
2. The original **KidPaint** program must be run as a client-side program.
3. When **KidPaint** (client) has just been launched, it shows a GUI for inputting the user name. After inputting the name, the client broadcasts a request to the network using UDP.
4. When the server receives the request, it sends a UDP packet with its IP address and port number back to the client.
5. Once the client receives a reply from the server, the client establishes a TCP connection to the server and downloads the sketch data. The sketch will then be rendered on the sketchpad of the client.
6. The kid does not need to input anything related to the network setting including the IP address and port number of the server.

7. The client will send TCP packets with differential updates to the server if the kid drew on the sketchpad.
8. The client will receive TCP packets with differential updates from the server if other kids (other clients) drew on the sketchpads. Then, the client applies the updates to its sketch.
9. The client sends a TCP package with a message to the server if the kid typed a message in the message field and pressed ENTER.
10. The client will receive a TCP packet with a message from the server if one of the kids typed a message in the message field and pressed ENTER.
11. The received message with the sender's name will be displayed in the chat area immediately.
12. A new button should be added to the client-side program for saving the sketch data into a local file.
13. A new button should be added to the client-side program for loading the sketch data from a local file. The sketch data must be sent to the server, and the sketchpads of all connected clients must be updated then.
14. With this approach, all kids draw on the **SAME** sketch.

#### Advanced client-server approach (max. 70 marks)

Your version of **KidPaint** should fulfil the requirement point 1 to point 13 of the basic client-server approach and support multiple sketches. The following are the additional requirements:

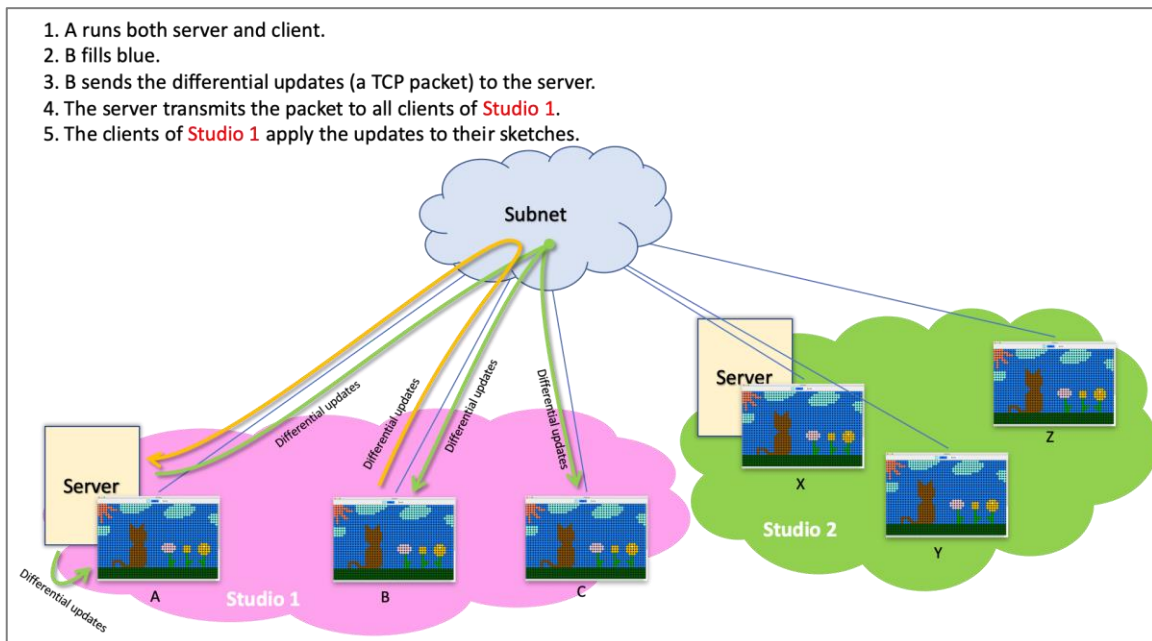
1. The server sends a list of studios to the client once a TCP connection established.
2. On the client, a GUI shows a list of studios.
3. The kid can either select one of the existing studios or create a new studio.
4. If the kid selects one of the existing studios, the client then downloads the sketch data of the selected studio and shows the sketch on the sketchpad.
5. If the kid creates a new studio, a GUI should show and let the kid decide the size of the sketch (width and height) for the new studio.
6. The sketch data and messages will be sent to the clients who are in the SAME studio only.



### Peer-to-peer approach (max. 80 marks)

Your version of **KidPaint** should fulfil the requirement point 5 to 14 from the basic client-server approach and the following additional requirements:

1. Combine the server-side code and client-side code into **KidPaint**. So, **KidPaint** can either run as a client or a server plus a client. If it runs as server plus client, the server-side code handles the data transmission among clients.
2. When **KidPaint** has just been launched, it prompts for inputting the user name. Then, it broadcasts a request to the network using UDP.
3. When a server receives the request, it replies with its studio name, IP address and port number back.
4. When **KidPaint** receives the replies (from multiple servers), it shows a list of studio names and allows the kid to select one of the studios or create a new studio.
5. If the kid selected one of the studios, **KidPaint** then runs as a client.
6. If the kid created a new studio, **KidPaint** then runs as a server plus client.
7. The sketch data (including differential updates and data loaded from the local file) and messages will be sent to the clients who are in the SAME studio only.



### Additional Features (max. 20 marks)

You need to add additional features to **KidPaint**. The marks are dependent on the difficulty level, functionalities, and completions of the additional features.

## Project Schedule

### **26 November 2021**

Submit your **Source Codes** through BUMoodle no later than **11:00 PM, 26 November 2021**

Submit your **Project Demonstration Video** no later than **11:00 PM, 26 November 2021**

The submission method will be announced later.