# CPSC 559 – W2021 – Project Iteration 1

## Contents

## System Description

This semester, we'll create a peer-to-peer system. Each of you will write the code for your own peer process. A registry process, that will help you find some of the peer processes in the system, will be provided. There will be some differences between communicating with another peer process in the system and the registry process.

This first iteration will focus on the communication with the registry. Monitor the discussion board to get the IP address and port number for the registry. The default address is ~~136.159.5.27~~ 136.159.5.22 at port 55921. If this is unavailable for any reason, the registry will be moved to a different location. (It will help if your peer process takes the registry IP address and port number as a command line argument, in case the registry needs to be run at a different location.)

## Requirements for a Peer Process

You may use any programming language you wish to create your peer process. The instructor and TAs will indicate which languages they can provide support for. Make sure to choose one of those languages if you want support from the TA or instructor. (If you don't need any support in coding, feel free to use any language you wish. Do make sure you document your code carefully to improve the readability of your code, even for those that are not familiar with the language you choose.)

All communication between processes in the system will be using Unicode characters. **Make sure to convert numbers to strings (or a sequence of Unicode characters) before sending them over the network.**

Your peer process must keep track of the list of locations of peers in the system that it is aware of. For this iteration, it will only contain the initial list of peers that you'll receive from the registry. In the next iteration, you will communicate with peers to maintain and add to this list.

Each peer will be uniquely identified by their IP address and port number. No other information about peers needs to be stored at this time. Make sure that the list will not have any duplicates and carefully choose a data structure to ensure it can be used in a thread safe matter. (For future iteration, the list will likely be accessed by multiple threads.)

Your process should go through the following steps for this first iteration.

1. Connect with the registry to request an initial list of peer processes. See the section Registry Communication Protocol below for more detail on the communication with the registry. The Registry can ask for information before releasing a peer list.
2. Store this initial list of locations of peer processes.

You must be able to establish the initial connection with the registry and at minimum respond to the request for your code. If this does not work, **you will not be able to submit your code and you don't qualify for any marks for this first iteration.**

## Registry Communication Protocol

Use the TCP/IP communication protocol for communication with the Registry. Once the connection is established, the registry will send one of the following five requests. The registry can make these requests in any order.

### 1) Team Name Request

```
<team name request> ::= "get team name"<newline>

<team name response> ::= <team name><newline>

<newline> ::= '\n'
```

Explanation: once the connection is established, the registry sends the string 'get team name' followed by a new line character. The registry then waits to receive the name of the team which must be terminated by a new line character. (This means a new line character can't be part of the team name.)

### 2) Code Request

```
<code request> ::= "get code"<newline>

<code response> ::= <language><newline><code><newline><end_of_code><newline>

<language> ::= [a-zA-Z0-9]+

<code> ::= .*

<end_of_code> ::= "..."
```

Explanation: once the connection is established, the registry sends the string ‘get code’ followed by a new line character. The registry then waits to receive:

1. The name of the language that the code is written in followed by a new line character. The name of the language should be expressed as an alpha-numeric string without any punctuation or white space characters. (Use 1 or more a-z character, A-Z characters and 0-9 characters.)
2. The source that is used to run your peer process. If the code is over multiple files, send the content of each file, one after the other. Any Unicode characters can be used for the code. You can indicate using comments if code is in multiple files. (We won’t run the code however. It is used exclusively to grade the quality of the code.)
3. Once all the code is sent, indicate all code was send by sending three dots on their own line. (Your code itself should never have three dots on their own line.)

## 3) Receive Request

```
<receive request> ::=
                "receive peers"<newline><numOfPeers><newline><peers>

<numOfPeers> ::= <num>

<peers> ::= <peer> | <peer><peers>

<peer> ::= <ip><colon><port><newline>

<ip> ::= <num><dot><num><dot><num><dot><num>

<port> ::= <num>

<num> ::= [0-9]+

<dot> ::= '.'

<colon> ::= ':'
```

Explanation: once the connection is established, the registry sends the string ‘receive peers’ followed by a new line character. The registry does not expect any response for this request. Instead, it sends the number of peers that will be send (on its own line) followed by a list of peers in the form of IP addresses and port numbers. The IP address and port number are separated by a ~~comma~~colon. Each peer is on their own line.

For example, the request may contain the following:

*receive peers*
*2*
*136.159.5.27:41*
*136.99.21.5:567*

## 4) Report Request

```
<report request> ::= "get report"<newline>
<report response> ::=
      <numOfPeers><newline><peers><numOfSources><newline><sources>
<sources> := <source> | <source><sources>
<source> ::=
      <source location>~~<newline>~~<date><newline><numOfPeers><newline><peers>
<source location> ::= <peer>
<date> ::=
      <year><dash><month><dash><day><space><hour><colon><min><colon><sec>
<day> ::= <two digit num>
<month> ::= <two digit num>
<year> ::= <four digit num>
<hour> ::= <two digit num>
<min> ::= <two digit num>
<sec> ::= <two digit num>
<two digit num> ::= [0-9][0-9]
<four digit num> ::= [0-9][0-9][0-9][0-9]
<dash> ::= '-'
```

Explanation: once the connection is established, the registry sends the string 'get report' followed by a new line character. The registry then waits to receive your current list of peers followed by a report that indicates all sources of this peer list.

If no receive request has been received yet, your list of peers will be empty and the list of sources will be empty as well. If you have already got a receive request then the response should contain the list of peers you received from the registry and there is only one source of peers in your list of sources.

For example: if the 'get report' request is received after the peers in the 'receive peers' request example above, the following could be a response to a 'get report' request.

*2*
*136.159.5.27:41*
*136.99.21.5:567*
*1*
*136.159.5.27:55921*
*2021-01-25 15:18:23*
*2*

*136.159.5.27:41*
*136.99.21.5:567*

In this example, the first line indicates the length of the peer list. The second and third line contain the information about the peers in the list. The fourth line indicates there was one source that contributed to our list of peers. The fifth line gives the IP address and port of the single source. The sixth line the date that the source provided a list of peers. The remaining lines give the information received from the source. There is no ordering requirement for lists of peers.

### 5) Close Request

```
<close request> ::= "close"<newline>
```

Explanation: once the connection is established, the registry sends the string 'close' followed by a new line character. The registry does not expect any response. It will close the connection after sending the message.

## Testing Your Solution

Before the iteration due date, a Registry server will be running (mostly) continuously. By default, it will be running at ~~136.159.5.27~~136.159.5.22:55921. If that location is not available for any reason, the registry will be restarted at a different location and this location will be shared with the class via the discussion board. If the registry is not running at the default location or the location last communicated on the discussion board, post a note on the discussion board and I'll get the Registry restarted as quickly as possible.

This test registry will send requests in the following order:

1. 'get team name'
2. 'get code'
3. 'receive peers'
4. 'get report'
5. 'close'

The Registry that will be used to submit your solution will use a different sequence of requests when testing your code. Make sure that you are not dependent on that order of requests. There may also be duplicate requests.

For your convenience, the source code of this test registry implementation is available on D2L.

## Submission

This first iteration is due on Friday Jan 28 at 3pm. At this time, the location of the submission registry server will be released. Remember that the testing registry and submission registry will send the same request but NOT in the same order. Some requests may also be duplicated.

You are required to use your code to connect with this server any time between 3pm and 4pm on Friday Jan 28. (If you will not be able to access a computer and run your code in this timeframe, contact the instructor or TA. We can help you create a script that will automatically run your code on time.) Grading will be based on:

1. Your ability to connect with the submission registry server and your ability to send your source code when requested. **If this portion is not functioning, your submission will NOT be graded.**
2. The accuracy of the report(s) that your process sends to this registry sever.
3. The quality of the code. (Design, legibility and documentation)

## Collaboration Opportunities and Limitations

You may work alone or with one other student in the class. Each team must have a unique team name. Please contact the instructor with a preferred team name and the list of team members.

No other individuals should work directly on the code you write, except for the instructor and TAs. Do not send your entire code base to any individuals nor make it available in a public repository, a discussion board or any other forums.

You may use any other resources you find useful. You may ask questions in the course discussion boards and any public discussion boards. Do make sure you cite any sources you use and any suggestions you received on discussion boards and other forums. An example on citing code using code documentation can be found at: https://runestone.academy/runestone/books/published/StudentCSP/CSPCreativity/referencing.html Do make it clear exactly which of your submitted code the citation relates to.

If you find an algorithm at a source other than the materials provided for class, make sure you cite this as well in the code documentation.

If you are not sure if a certain level of help or collaboration from individuals outside the team is allowed, make sure to contact the instructor for clarification.