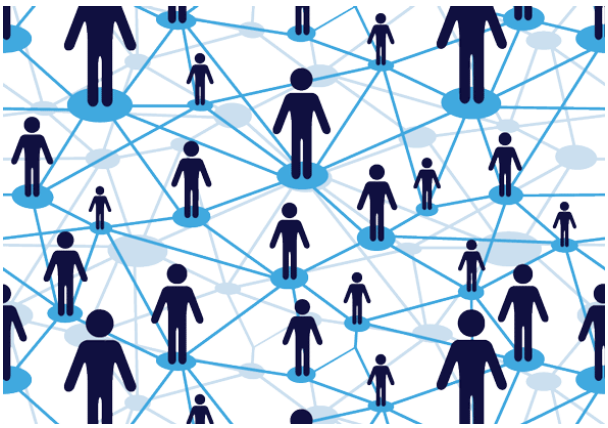
Office of Strategic National Alien Planning



StreamConnect
(SC)

Product Requirements and Design Document

October 16, 2018

0.1 Executive Summary

The StreamConnect(SC) is a file sharing service that can be used to share files between individuals in the organization. SC does not require purchasing of new file sharing servers and therefore is a very cost effective and reliable file sharing solution.

0.2 Changelog

20182809-0 *Beau* Wrote initial ideas for SC

20180910-1 *Pietro* Formatting and proof-reading

20180911-2 *Beau* Updated Executive Summary and Bussiness Requirements

20181001-3 *Pietro* Updated misc, format and OS

20181001-4 *Pietro* Updated future phases of security and security phase

20181001-5 *Beau* Layed out the security triad

1 Business Requirements

1.1 Overview

This project will create a peer to peer StreamConnectfile sharing network that an employees of OSNAP can use to exchange files. This system will not rely on a central server, and thus will be resilient to failures of individual file sharing nodes. Stream connect can be viewed as an intermediate or backup solution for file sharing until the purchase of file sharing server is feasible. SCensures our organization will always have a pipeline to share files.

1.2 Requirements List

1.2.1 Operating System

SC will have a framework that is accessible by any operating system that can run python applets.

1.2.2 Protocols

1. Only internet protocol (IP) networks need be supported
2. TCP on arbitrary high ports[1] must be supported
3. BitTorrent protocol must be understood by machines connected to the network.

1.2.3 Interface

1. A command line interface will be provided
2. Users will be able to upload (seed) files using their work computers.
3. Users will be able to download (leech) files using their work computers.
4. A GUI implementation of the SC user endpoints is an endeavour that will be considered during future implementations.

1.2.4 Miscellaneous

1. This project leaves many security questions and considerations open regarding the download access and permissions granted to various users.
2. Encryption of uploaded files using user keys could be an appropriate solution to this problem, but specifics of security implementations will be considered later.

[1] Ports over 1024

3. These work computers have semi-consistent uptime throughout a normal business day. This means that the computers will be on throughout the day with brief pauses and breaks during the day (lunch breaks, coffee breaks, etc).

4. While OSNAP does have international offices across all of the major time zones, this project will be limited to the United States branch of our business.

5. For this project, downloading of files will only need to be available during business hours of the day.

6. Users must be able to see all available documents for download on the network.

7. As soon as a user uploads a file, that file should be visible as an available file for download on the network.

2 Technical Requirements

2.1 Overview

SC uses a simple client/server architecture to accomplish the business needs. On one of the hosts to be checked, the server component will listen for new connections using a specific transport protocol and port. On the other host, the client component will attempt to make a connection to the server. If a successful connection occurs, the client will exit with a 0 exit code. If the connection is unsuccessful, the client will exit with a 1 exit code.

2.2 Requirements List

2.2.1 Implementation Language

SC is to be implemented in all three standard OSNAP implementation languages. This technical requirement was added so that the SC codebase can serve as example code for developer training.

1. C implementation
2. JAVA implementation
3. Python implementation
4. Clients and servers will interoperate between implementation languages

2.2.2 Protocol

1. IP networking using sockets
2. TCP support
3. UDP support
4. Unknown/unsupported messages will be ignored by the client and server
5. Unknown/unsupported messages in the TCP stream should cause the server to close the connection
6. Unknown/unsupported messages in the TCP stream should cause the client to exit with a 1 exit code

2.2.3 Interface

1. For any run, the settings will be passed in as arguments
2. Client Arguments
   (a) The first argument will be the transport, 'tcp' or 'udp'
   (b) The second argument will be the IP address to connect to
   (c) The third argument will be the port number
3. Server Arguments
   (a) The first argument will be the transport, 'tcp' or 'udp'
   (b) The second argument will be the IP address to bind to
   (c) The third argument will be the port number
4. Failure to provide the required arguments will cause a help message to be printed

2.2.4 Miscellaneous

1. The client will timeout after 5 seconds
2. Unacknowledged UDP will packets will be resent after 1 second

3 Security Plan

SC is a file sharing system with a particular set of priorities regarding availability, confidentiality and integrity as follows:

3.0.1 Confidentiality

1. While we can imagine situations where documents being uploaded to the network are not appropriate for the entirety of the company to view, we believe that this problem could be solved throughout this phase by having those individuals encrypt their messages on an ad-hoc basis.

2. As a result, this project will differ encryption of uploaded files to a later stage of our development (refer to future phases details below).

3.0.2 Integrity

1. It is necessary that there will be some check and measure to ensure that the files that an individual uploads are the same as the downloaded version.

2. In order to ensure that the files maintain their integrity, hashes of the uploaded files will be contained in the torrent files so that users who are downloading files can check that their download is the same as the version that was initially uploaded.

3. Furthermore, the projects implementation on top of the TCP protocol gives a certain degree of confidence and assurance that the information being sent over the network will arrive in the order that they were sent in and that all packets will arrive to their final destination.

3.0.3 Availability

1. It will be necessary that a significantly large number of users on the network are seeding files.

2. This is necessary because in order for a file to be available for download, a significant number of individuals on the network must have a copy of the file and be seeding the file in order for others to download it.

3.0.4 Future Phases

1. For future phases of the project, it is possible that the confidentiality of the data packets sent through the network could make use of a potential encryption system.

2. However, this not being the primary concern and nature of the networks current implementation.

3. Additionally, the use of digital signatures may pose some advantages to StreamConnect, but are not of immediate concern for our system.

4 Design

4.1 Design Overview

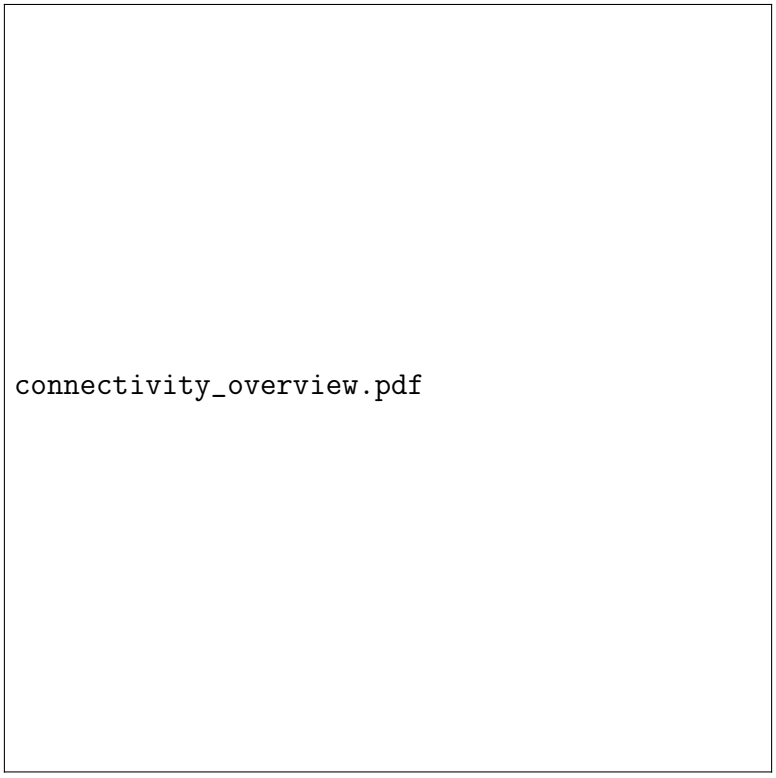


Figure 4.1: Diagram of our basic Peer To Peer architecture.

SC operates as a peer to peer application. The peers will have a simple configuration where they are all fully connected, as such the simplest Peer-To-Peer architecture is implemented (Figure 4.1).

4.2 Component Architecture

The peers can be thought of as nodes in a graph which would represent our file sharing network.

4.2.1 Peer Process

Our peer nodes will be able to initialize a shared directory by giving an IP address and Port that other peers can connect to. The init function will proceed to create a config meta-file which is used to keep track of the nodes in the network. Then, peers connected to the network, sharing their initialized directory. Lastly, the function to connect through TCP and share files with one another, in order to have a fully working system. 4.2.

Figure 4.2: Flowchart for the client component.

4.2.2 Configuration File Module

The configuration file is what will be used to update all the connections of the peers that are available in the network. This file is a simple abstraction of the .torrent files which can serve as trackers in other Peer-To-Peer networks. Our file will list the peers that are available in the network, their respective ports and addresses. 4.3.

4.3 Protocol Detail

Our peer to peer network is built to support as many messages as possible. Therefore this interactions are used to join the existing network, list the files available in the network and request the files. Utilizing TCP we transfer data in between nodes, utilizing very specific byte structures.

4.3.1 TCP Messages

TCP is a session based network protocol. A TCP connection presents as a stream of bytes, so partial messages or multiple messages from a single connection might be read during a single read() call. On the other hand, TCP provides reliability and in-order delivery.

Figure 4.4 shows the message traffic in the ideal case. Since TCP is secure and reliable, all of the network functionalities are further subdivided into a request and response protocol for each peer to create a connection.

TCP should be able to pass any kind of data, however, we have limited the messages to 8 bits for our peer interconectivity.

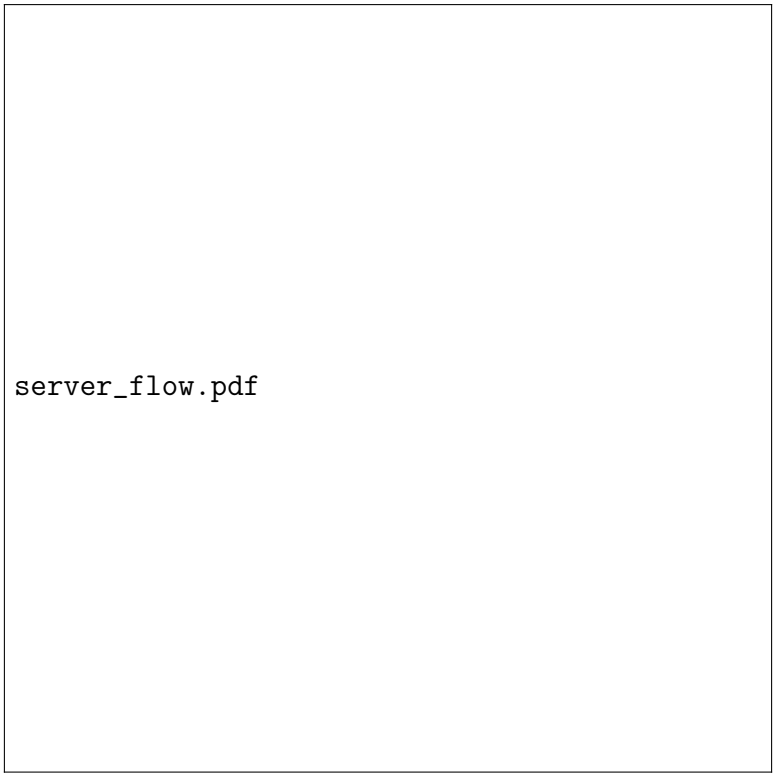


Figure 4.3: Explaining how the metafile works

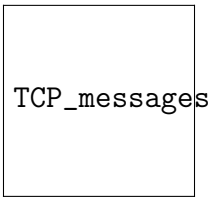


Figure 4.4: Diagram of the expected messaging traffic for TCP.

"JOIN
"

Sends a message of exactly 4 bytes attempting to connect to another peer in the network, and then awaits for a specific reply.

"ASK JOIN "

This 7 bytes message is padded such that a peer that listens the JOIN message will reply with size of the file alongside the file a peer wishes to share.

4.3.2 "LIST
"

Sends a 8 byte message which utilizes our fully connected node traversal algorithm to list all of the fiels available from the particularly requested peer.

4.3.3 "ACKLIST "

Returns the size of the files and lists all of the files contained in the shared directory.

4.3.4 "REQ FILE "

Requests the Size of the File Name and a String containing the respective file name.

4.3.5 "ACK REQ "

Returns the Size of the File and the respective file requested.

5 Scaling Model

SC implements a peer-to-peer architecture in where currently adding more peers will simply make the graph more challenging to traverse, therefore our implementation will have a maximum number of peers in its network.

6 Test Plan

Scripts to assist in the testing are provided, scc_server.sh and scc_client.sh.

6.1 Local Test

In this test, the client and server are run on the same host and connect via the loopback interface. The steps of the test should be tried for each supported implementation language and transport protocol.

6.1.1 Open Terminals

Open two terminal windows, on the same host, and change to the build directory in both windows.

6.1.2 Server Start

In terminal window B, start the server with the following command:

```
scc_server.sh c tcp 127.0.0.1 7000
```

6.1.3 Client Start

In terminal window A, start the server with the following command:

```
scc_client.sh c tcp 127.0.0.1 7000
```

6.1.4 Verify

ssc_client.sh will display a message indicating success or failure.

6.2 Network Test

In this test, the client is run on host A and the server is run on host B. This test is not scripted since the IP address cannot be known in advance. The steps of the test should be tried for each supported implementation language and transport protocol.

6.2.1 Open Terminals

Open two terminal windows, on different hosts, and change to the build directory in both windows.

6.2.2 Determine Server Address

In terminal window B, use ifconfig to determine the host IP address. That IP address will be referred to as ¡hostB¿ in the later steps.

6.2.3 Server Start

In terminal window B, start the server with the following command:

```
scc_server.sh c tcp <hostB> 7000
```

6.2.4 Client Start

In terminal window A, start the server with the following command:

```
scc_client.sh c tcp <hostB> 7000
```

6.2.5 Verify

ssc_client.sh will display a message indicating success or failure.