A1 – Covert Sockets

Ben Soer

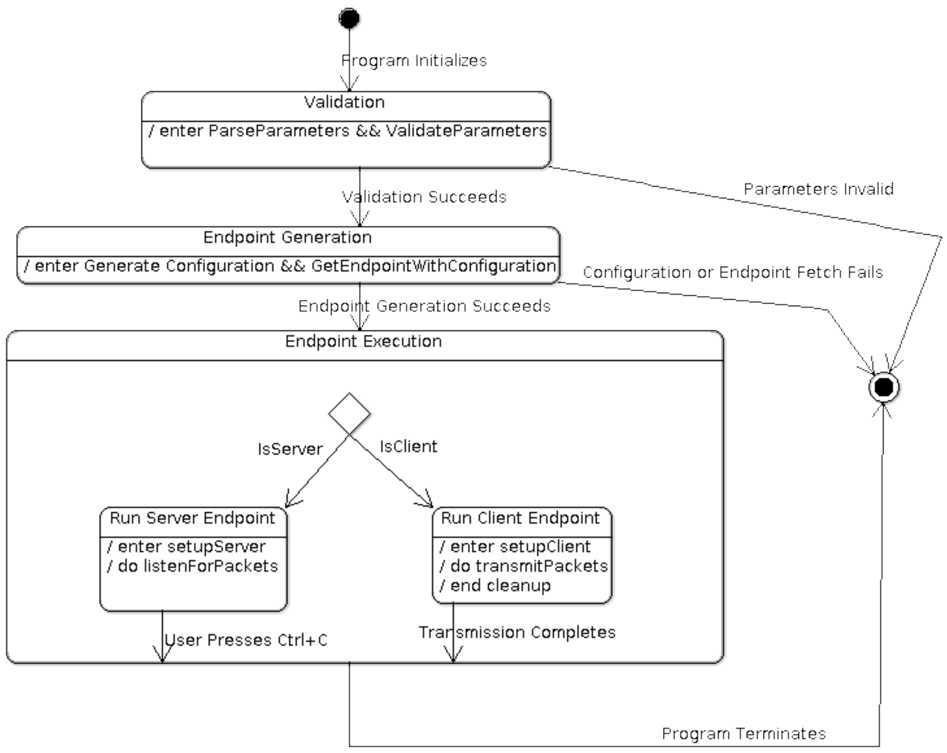
A00843110

Table of Contents

Finite State Machines

Covert Socket Application

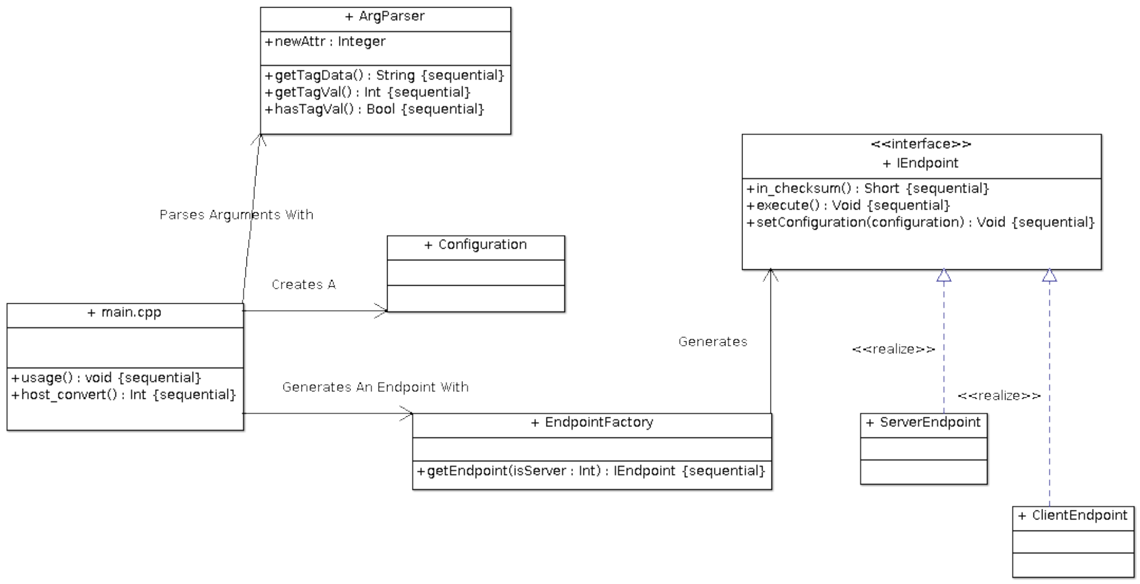
See images/CovertSocketStateMachine.jpg for original image



Class Diagram

Covert Socket Application

See imgs/CoverSocketClassDiagram.jpg for original image


+ ArgParser
+newAttr : Integer
+getTagData() : String {sequential}
+getTagVal() : Int {sequential}
+hasTagVal() : Bool {sequential}
<<interface>>
+ IEndpoint
+in_checksum() : Short {sequential}
+execute() : Void {sequential}
+setConfiguration(configuration) : Void {sequential}
Parses Arguments With
+ Configuration
Creates A
+ main.cpp
+usage() : void {sequential}
+host_convert() : Int {sequential}
Generates
<<realize>>
<<realize>>
Generates An Endpoint With
+ EndpointFactory
+getEndpoint(isServer : Int) : IEndpoint {sequential}
+ ServerEndpoint
+ ClientEndpoint

Pseudocode

main.cpp

main()

- Check Running as Admin/Root. Check Parameters passed. Call usage() if no parameters passed
- Create ArgParcer – start parsing arguments
- Run validation on arguments
- Create Configuration struct and store arguments into it
- Create EndpointFactory – get an Endpoint
  - If Error Getting Endpoint Abort
- Execute endpoint

- Cleanup and Terminate

usage(char *programname)

- Print program usage along with examples

host_convert(const char *hostname)

- Make DNS request to resolve hostname if name instead of IP is given
    - If Error Resolving. Abort. Else Return result adress

IEndpoint.cpp

in_chksum(unsigned short *ptr, int nbytes)

- Create a checksum based on ptr data and return its value

EndpointFactory.cpp

getEndpoint(int isServer)

- If Server == 0. This is a Client.
    - Validate configuration object for client setup
    - Determine and print mode being used
    - Create ClientEndpoint
    - Return ClientEndpoint object
- Else. This Is A Server
    - Validate configuration object for server setup
    - Determine decoding mode being used
    - Create ServerEndpoint
    - Return ServerEndpoint object

ServerEndpoint.cpp

setConfiguration(Configuration configuration)

- Set the configuration object to be used during execution of the server

execute()

- Initialize packet structures and variables
- Configure file writing for received and parsed packets
- Infinite While Loop:
    - receive packet
    - parse out appropriate header based on configuration object settings
    - print to screen
    - write to file

ClientEndpoint.cpp

setConfiguration(Configuration configuration)

- Set configuration object to be used during execution of client

execute()

- Initialize packet structures and variables
- Configure file reading for packets to be sent. If error opening file abort
- While there are characters to read in the file:
    - Build packet based on configuration object
    - Send packet
    - Print character that was sent