

Maraton

Genomics Distribution Computing System

October 22, 2015

YHGenomics

Contents

[1. Third-part Library 2](#_Toc433902754)

[1.1. Libuv 2](#_Toc433902755)

[1.1.1. Description 2](#_Toc433902756)

[1.1.2. Usage 2](#_Toc433902757)

[1.1.3. API 2](#_Toc433902758)

[1.2. Json 3](#_Toc433902759)

[2. UML Diagram 3](#_Toc433902760)

[3. Interface Layout 4](#_Toc433902761)

[3.1. Singleton<T> 4](#_Toc433902762)

[3.1.1. static instance() 4](#_Toc433902763)

[3.2. Manager<T> 4](#_Toc433902764)

[3.2.1. void push(T\* instance) 4](#_Toc433902765)

[3.2.2. bool pop(T\* instance) 4](#_Toc433902766)

[4. Protocol Layout 4](#_Toc433902767)

[5. Utility 4](#_Toc433902768)

[5.1. CircleBuffer 5](#_Toc433902769)

[5.1.1. push(const char\* data, int len) 5](#_Toc433902770)

[5.1.2. char\* pop(int len) 5](#_Toc433902771)

[5.2. Buffer 5](#_Toc433902778)

[5.2.1. Buffer(const Buffer& buffer) 5](#_Toc433902779)

[5.2.2. Buffer(const Buffer&& buffer) 5](#_Toc433902780)

[5.2.3. Buffer& operator=(Buffer& buffer) 5](#_Toc433902781)

[5.2.4. Buffer& operator=(Buffer&& buffer) 5](#_Toc433902782)

[5.2.5. raw(const char\* data,int len) 5](#_Toc433902783)

[5.2.6. char\* raw() 6](#_Toc433902784)

[5.2.7. int length() 6](#_Toc433902785)

[5.3. Logger 6](#_Toc433902786)

[5.3.1. static void sys(const char\* fmt,…) 6](#_Toc433902787)

[5.3.2. static void error(const char\* fmd,…) 6](#_Toc433902788)

[5.4. Zlib 6](#_Toc433902789)

[5.4.1. Buffer compress(const char\* data,int len) 6](#_Toc433902790)

[5.4.2. Buffer uncompress(const char\* data , int len) 6](#_Toc433902791)

[5.5. UUID 7](#_Toc433902792)

[5.5.1. static string create() 7](#_Toc433902793)

[6. Network Layout 7](#_Toc433902794)

[6.1. Network Layout 7](#_Toc433902795)

[6.1.1. UVSockService 7](#_Toc433902796)

[6.1.2. SessionManager<TT> 8](#_Toc433902797)

[6.1.3. Session 9](#_Toc433902798)

[6.1.4. HTTPSession 9](#_Toc433902799)

[6.1.5. ClusterSession 9](#_Toc433902800)

[6.1.6. ExecutorSession 10](#_Toc433902801)

[6.1.7. MasterSession 10](#_Toc433902802)

[6.2. HTTP Protocol Layout 10](#_Toc433902803)

[6.2.1. HTTPHandler 10](#_Toc433902804)

[6.2.2. HTTPRouter 11](#_Toc433902805)

[6.2.3. HTTPRequest 11](#_Toc433902806)

[6.2.4. HTTPResponse 12](#_Toc433902807)

[7. Master Layout 13](#_Toc433902808)

[7.1. Executor 13](#_Toc433902809)

[7.2. ExecutorManager 13](#_Toc433902810)

[8. Executor Layout 13](#_Toc433902811)

[8.1. Master Session 13](#_Toc433902812)

# Third-part Library

## Libuv

http://libuv.org/

### Description

libuv is a multi-platform support library with a focus on asynchronous I/O

### Usage

### API

#### uv\_tcp\_init(uv\_loop\_t\* loop,uv\_tcp\_t\* tcp)

Initial a TCP socket

#### uv\_ip4\_addr(char\* ip,sockaddr\_in\* sockaddr)

Initial a ipv4 address

#### uv\_tcp\_bind(uv\_tcp\_t\* handle,const sockaddr\* sock,uint flag)

Bind an address to a sock

#### uv\_tcp\_connect(uv\_connect\_t\* conn,uv\_tcp\_t\* tcp,sockaddr\* sockaddr,uv\_connect\_cb cb)

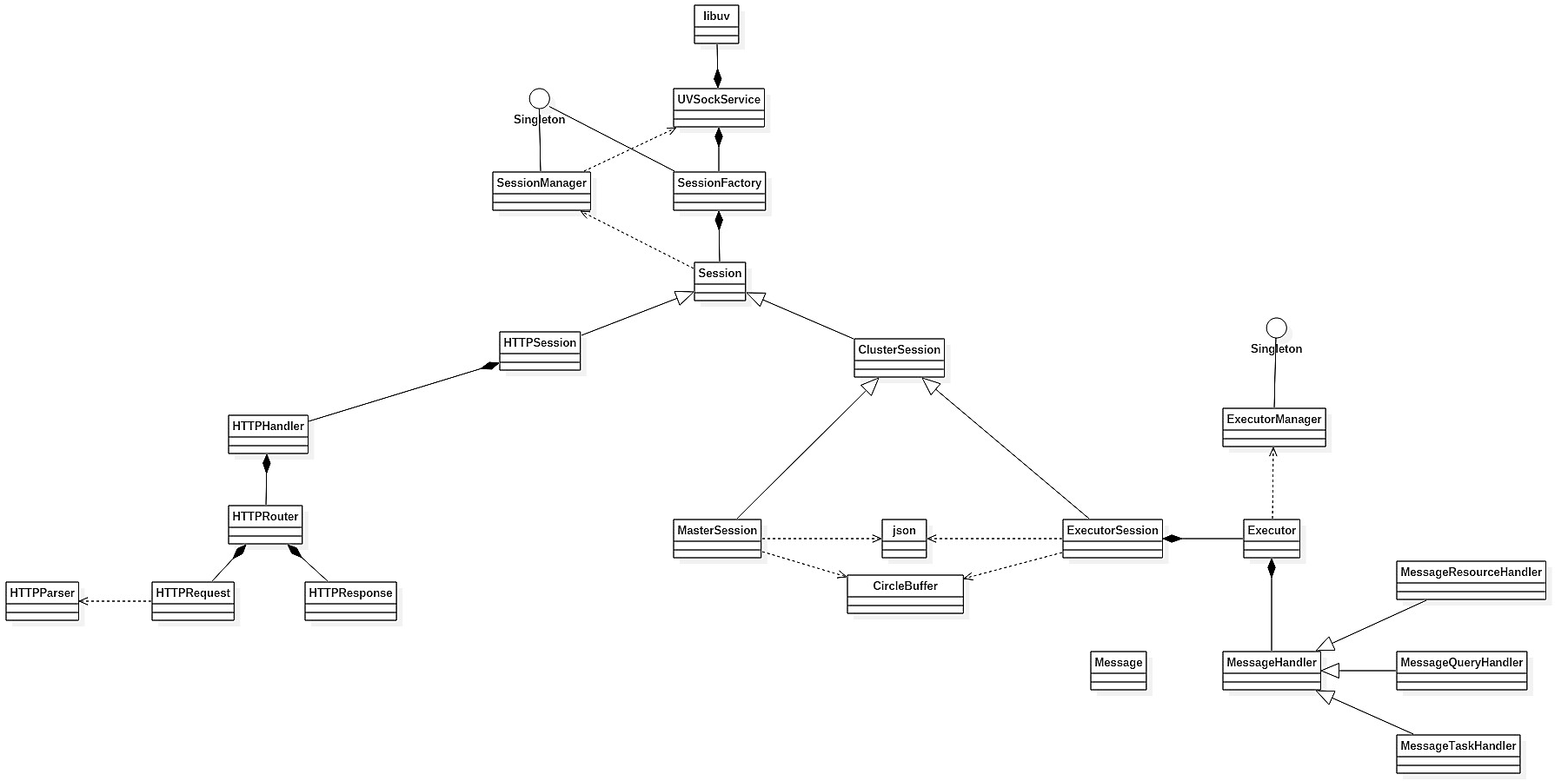
Connect to a remote TCP server

## Json

Checkout http://nlohmann.github.io/json/

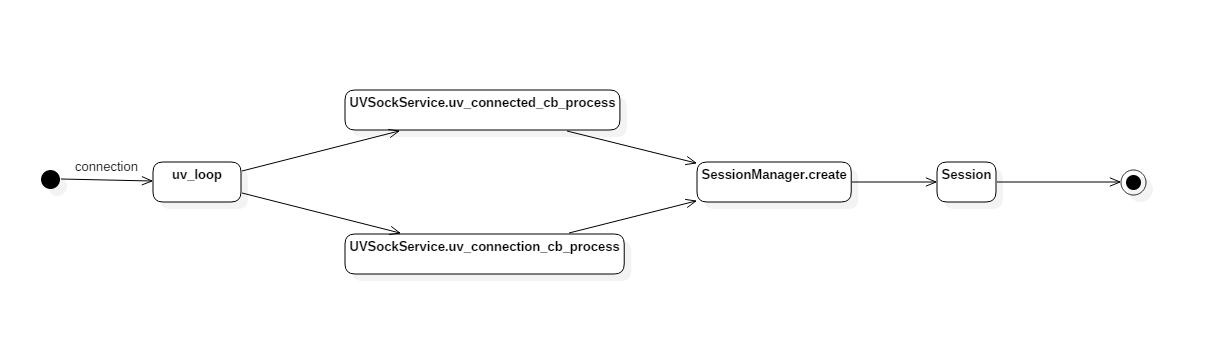
# UML Diagram

## Class Relation

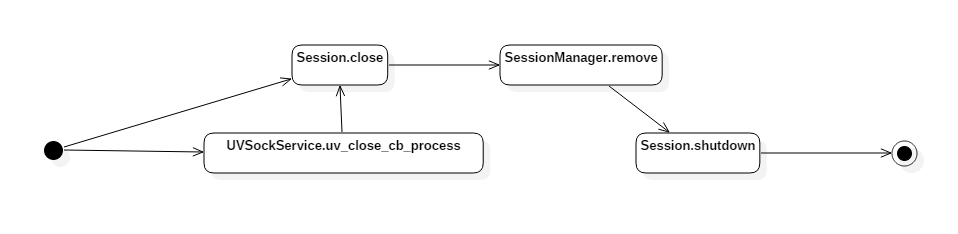


## Data Flow Diagram

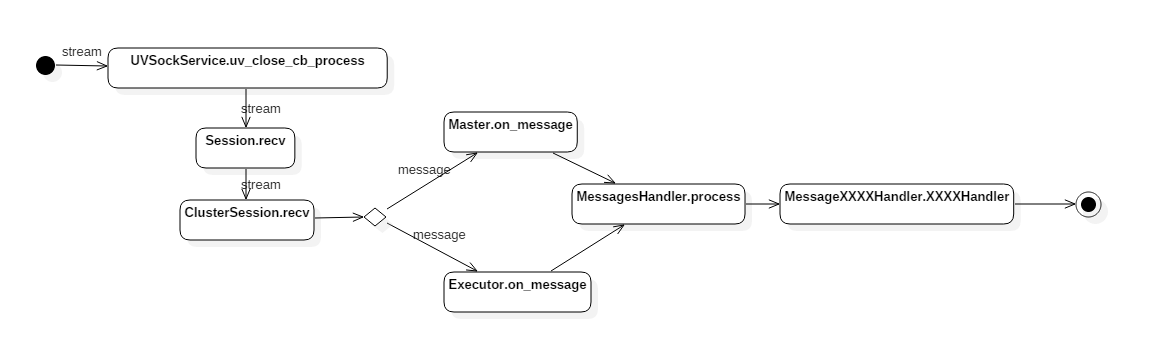
### Connection Flow



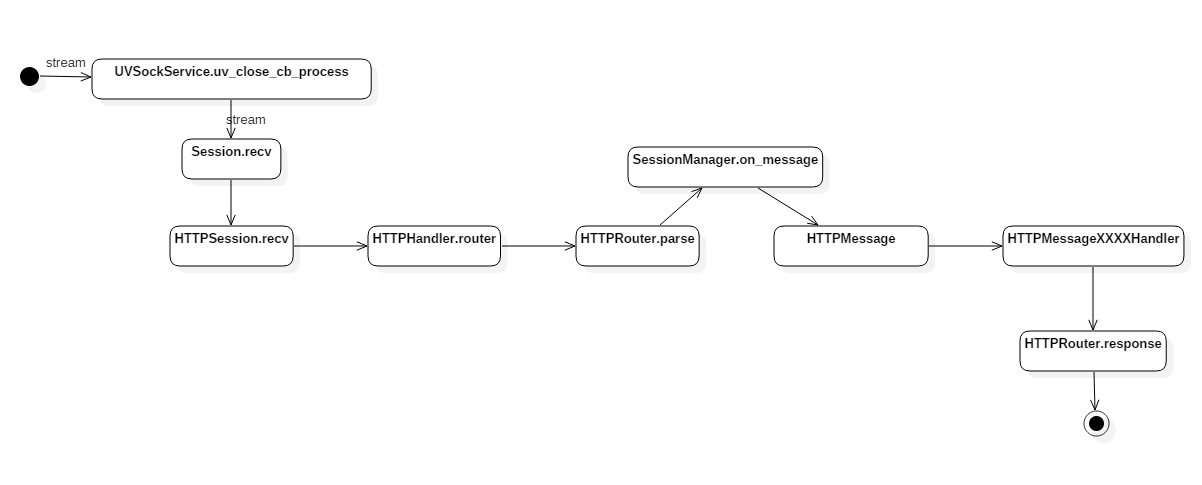
### Session Close Flow



### Cluster Session Data Flow



### HTTP Session Data Flow



# Interface Layout

## Singleton<T>

Template class for making a singleton class.

Children classes must declare friend Singleton<T> in your private region.

### static instance()

Return the instance of class

## Manager<T>

Provide manager function

### void push(T\* instance)

Add a new element to the set

### bool pop(T\* instance)

Remove na element in the set

PS: This function will not free the memory

# Protocol Layout

# Utility

## CircleBuffer

Provide a circle-buffer for saving memory.

### push(const char\* data, int len)

Add a data to the buffer

### char\* pop(int len)

Remove and return the data with len length



## Buffer

Provide a memory data struct with data pointer and data length

### Buffer(const Buffer& buffer)

Copy construction

### Buffer(const Buffer&& buffer)

Move construction

### Buffer& operator=(Buffer& buffer)

Override operator

### Buffer& operator=(Buffer&& buffer)

Override operator

### raw(const char\* data,int len)

Setter.

Set Buffer data and length

This function will copy the data and not release the input data.

### char\* raw()

Getter.

Return the pointer of the data

### int length()

Getter

Return the data length

## Logger

Provide a logger function

### static void sys(const char\* fmt,…)

Output the string in sys-format

### static void error(const char\* fmd,…)

Output the string in error-format

## Zlib

Provide a crypt, decrypt, compress and decompress functions

### Buffer compress(const char\* data,int len)

Compress the data.

This functions will copy data into Buffer and not release the input data

### Buffer uncompress(const char\* data , int len)

Compress the data.

This functions will copy data into Buffer and not release the input data

## UUID

Make a UUID

### static string create()

Return a UUID string

# Network Layout

## Network Layout

### UVSockService

Main network class. Provide the basic function of network.

Receiving data

Connect to server

Accept client

Release Session

#### bool listen( std::string ip, int port );

Make socket in server mode, that will accept income connection

#### bool connect( std::string ip, int port );

Make socket in client mode, that will connect to the target server

#### void run();

Run the main loop

This function will block the thread, any logic, register must be set before

### SessionManager<TT>

Session manager, will manage all the session, splits sessions by class type (TT)

Provide functions to add, remove, find, callback

#### Parent Class

* Manager<T>
* Singleton<SessionManager<T>>

#### Custom Type

* std::function<void( TT\* )> callback\_t

#### TT\* create(uv\_tcp\_t\*)

Create a session by passing uv\_tcp\_t to it

#### bool remove(TT\* t)

Remove the target session

#### TT\* find( TT\* t)

Find the target session by it instance.

If there is not such session it will return nullptr, otherwise return the instance.

#### void on\_create(callback cb)

Callback when a new session was created

#### void on\_close(callback cb)

Callback when a session was closed

### Session

Basic session class provide a way to use it

#### void close()

Close the session

This function will not effect immediately until next loop.

#### void send(const char\* data,int len)

Send the data to socket

This function will not effect immediately until next loop.

### HTTPSession

HTTP protocol session, created by port 80.

#### Parent Class

* Session

#### HTTPHandler\* handler();

Return the handler pointer

### ClusterSession

Basic class for cluster node.

Parse the network stream.

Verify the data

#### Parent Class

* Session

#### virtual send(Message\* message)

Send the message to the remote

#### virtual on\_message( callback )

Callback when a message is arrived

#### virtual on\_close( callback )

Callback when the session was closed

### ExecutorSession

Provide functions to communicate with the target executor.

This class is only used for master mode.

#### Parent Class

* ClusterSession

### MasterSession

Provide functions to communicate with the master

This class is only used for executor mode.

#### Parent Class

* ClusterSession

## HTTP Protocol Layout

### HTTPHandler

Provide HTTP formatting function.

### HTTPRouter

Provide usages for routing the http request

#### void parse( const char\* data, int len)

parse data and dispatch the request

#### response( callback )

Tell the class how to send data to client.

Callback when the class needs sending data to the client.

#### get(string path, callback)

Register a Get request on [path].

Callback when a GET request with the path arrived

Start with “/“

#### post(string path, callback)

Register a Post request on [path]

Callback when a POST request with the path arrived

### HTTPRequest

Manage the request, reads the HTTP header and body

#### string method()

Return the HTTP request mothod

#### string path()

Return the HTTP request path

#### map<string,string> header()

Return the HTTP header configuration

#### string body()

Return the HTTP body data

#### bool parse( const char data, size\_t len )

Parse the data as an available HTTP request

If the request is a HTTP then return true, otherwise return false

### HTTPResponse

Provide functions to make response easier

#### void status(int status)

Set the response status

#### int status()

Return the status

#### void header(string name, string value)

Set the header

#### void content(string content)

Set the response content

#### int length()

Return the length of response content

#### Buffer bytes()

Return the Buffer for entire response

#### clear()

Reset the response. Cleanup

# Executor Layout

## Executor

## ExecutorManager

# Master Layout

## Master Session