SOCKET++

A SIMPLE SOCKET LIBRARY IN C++

Ankit Gupta
Garima Agarwal
Swati Singh

Want to Build a Server?? Things to Take Care Of

create socket, bind, listen, accept

Error Handling

make it thread safe

Event Handling

Multithreading

EXISTING LIBRARIES

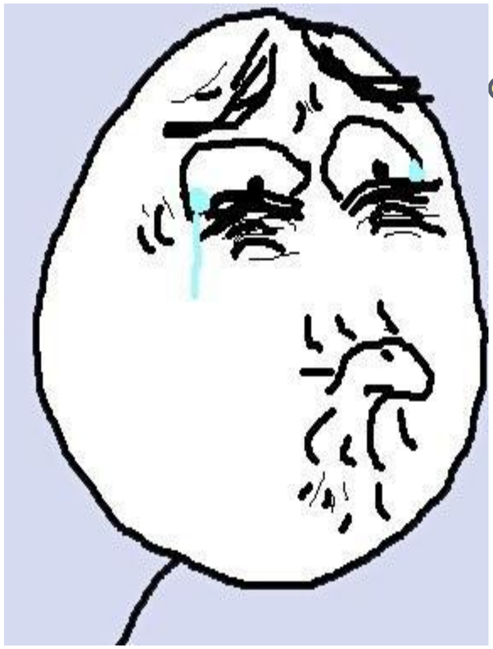
ACE

Practical Socket C++

Netlink Socket C++

Boost

Giallo C++ Network Library



EVEN MORE COMPLICATED

An Easy Interface for Socket Programming in C++

1) Easy and Convenient to Use

2) Error Handling Taken Care Of

3) Minimal Number of Lines of Code

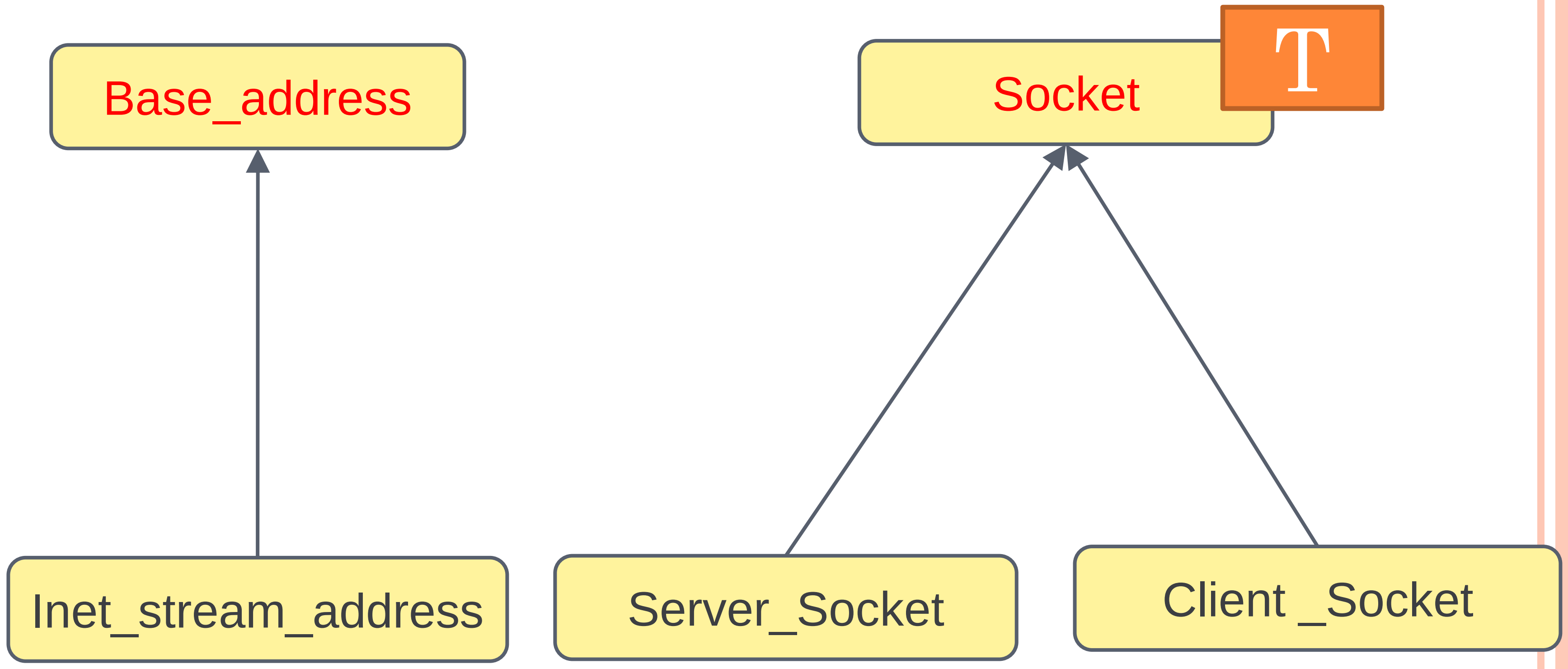
4) Provides Event Handling Interface

5) Extensible Design

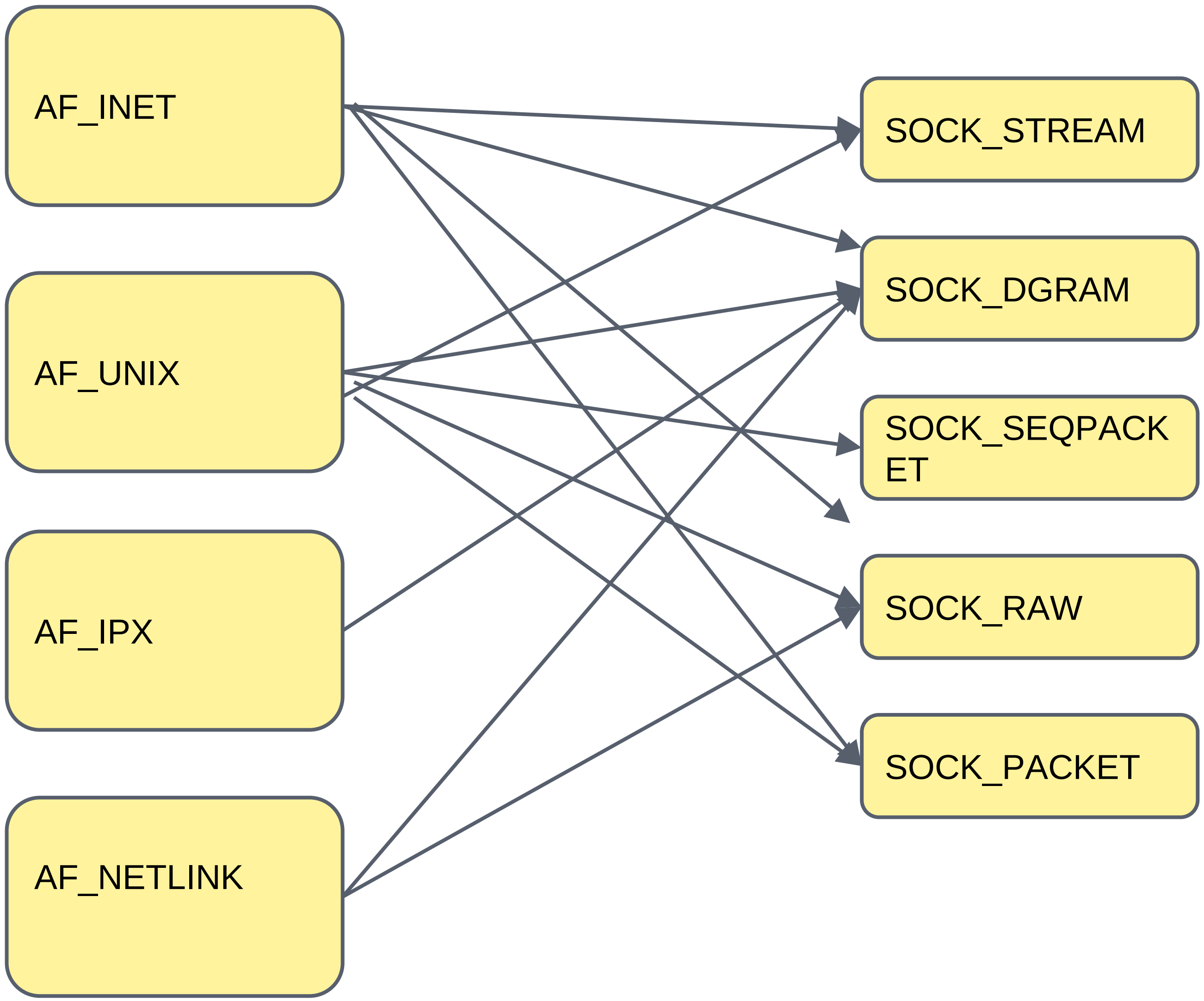
DESIGN DECISIONS

- Easier way of creating commonly used sockets.
- Separate Socket and Address class, instead of combining socket and address into one class only.
- Separate Interface for Synchronous and Asynchronous Server : As the way the Servers handle read/write differs considerably, we decided to go for separate interfaces for both the classes instead of a generic one.
  Single Event handler class for handling events for all the classes (Client/ Server) which would be linked to the Client/Server class.
- Separate Event Handler class that can be used for all Async Servers/Clients. Event handling Is implemented using a Reactor Design pattern

Socket and Address Class



Allowable Combinations



Creating a TCP Server Socket

```
int main ()
{
 int port = 80;
 int max_listen = 10;
 string ipaddress = “localhost”;
 server_sock_stream serversock(port,ipaddress,max_listen);
 string port2 = “80”;
 server_sock_stream serversock2(port2,ipaddress,max_listen);
}
```

INVARIANTS :

1) A TCP Server Socket must be in listening state after its constructed

Creating a TCP Client Socket

```
int main ()
{
 int port = 80;
 string ipaddress = “localhost”;
 client_sock_stream clientsock(port,ipaddress);
 string port2 = “80”;
 client_sock_stream clientsock2(port2,ipaddress);
}
```

INVARIANTS :

1) A TCP Client Socket must be in connected state after its constructor.

INVARIANTS

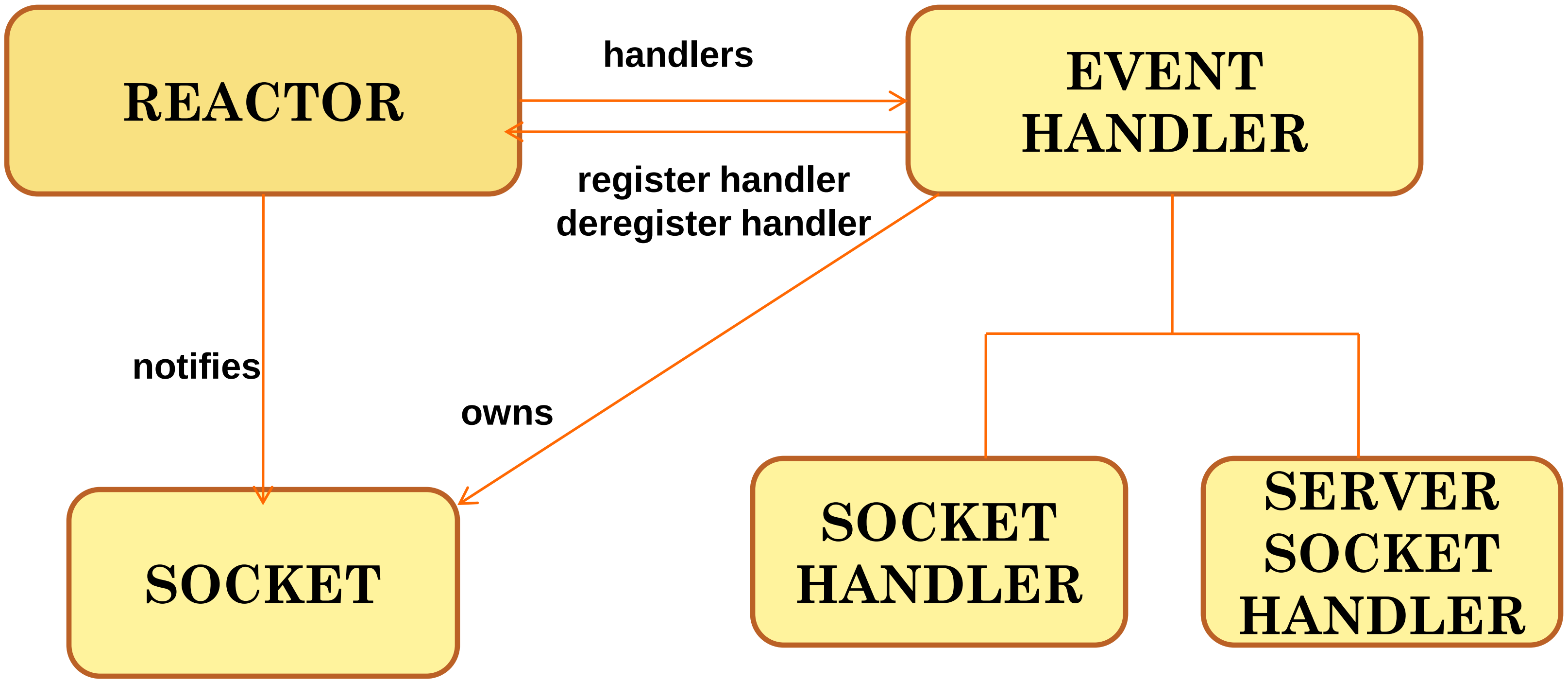
Socket invariants:

- We should have a valid socket, after the constructor of socket class.
- Socket Should be Closed when it goes out of Scope.(Close socket in Destructor)

Server invariants:

- When the server is constructed, it should listen on the port specified to it or an exception would be thrown.
- Number of accepted clients should always be less than the maximum value defined.

Event Handling Using Reactor Pattern



ASYNCHRONOUS TCP SERVER

```
class handle: public socket_handler<T>
{
   public:
     handle(unique_ptr<Socket<T>> sock): socket_handler<T>(std::move(sock)){};
     int handle_read()  {
     socket_handler<T>::handle_read();
                write(this->read_data);
     }
     void write(string data)   {
                this->write_data = data;
     socket_handler<T>::write();
     }
};
int main()
{
     Reactor *rec = Reactor::get_instance();
     async_server<inet_stream_addr, handle, server_socket_handler> server(8080);
     rec->Run();
     return 0;
}
```

ASYNCHRONOUS TCP CLIENT

```
class handle: public client_socket_handler<T>
{     public:
      handle(unique_ptr<client_sock_stream> sock): client_socket_handler<T>(std::move(sock)){};
      int handle_read()  {
                  client_socket_handler<T>::handle_read();
                  std::cout<<this->read_data;
                  write("hello");    }
      void write(string data) {
                  this->write_data = data;
                  client_socket_handler<T>::write();
      }
};
int main()
{     try{
      Reactor *rec = Reactor::get_instance();
      async_client<inet_stream_addr, handle> client(80, "localhost");
      client.write("hello");
      rec->Run();
      }
      catch (sock_error& serr)  {
      std::cout<<"Got error"<<serr.what();
      }
      return 0;
}
```

ERROR HANDLING

- Two Exception classes are used:
  - addr_error
    - For Invalid address, proper error should be thrown.
  - Sock_error
    - If socket cant be created
    - The port to which socket try to listen is busy.
    - If other end closes the connection.
    - If client is unable to connect to the server.

FUTURE WORK

- **SOCKET++ V 1.2**
  - Support for UDP
  - Synchronous Multi-threaded Client and Server
  - Documentation using Doxygen

- Extensive Testing using http://sockettest.sourceforge.net/

- Load Testing

ACKNOWLEDGEMENT

- Professor Bjarne Stroustrup,CSE Department ,Texas A&M University for teaching us Wonderful Course, Design Using C++.
- Andrew Nathan Sutton,Postdoctoral Researcher, Texas A&M Univeristy for helping us in various design decisions.

- www.stroustrup.com
- www.stackoverflow.com
- Beej guide: http://beej.us/guide/bgnet/
- Boost asio :http://www.boost.org/doc/libs/1_52_0/doc/html/boost_asio.html
- Ace Socket library: http://www.cs.wustl.edu/~schmidt/ACE-overview.html
- Poco C++ Library: http://pocoproject.org/

Further Reading

Code repository –

- https://github.com/ankitgupta29/socket

Thanks
Questions ?