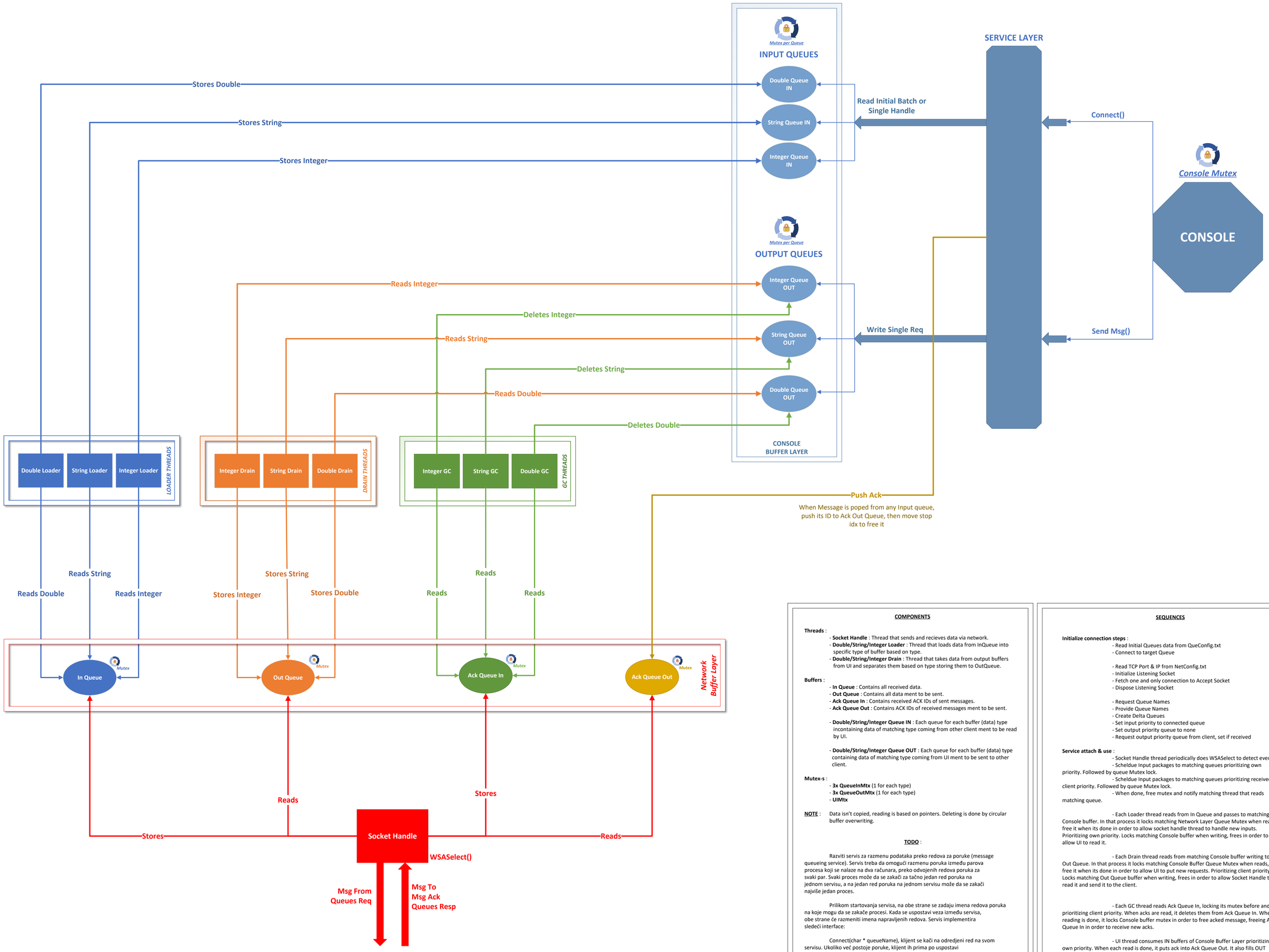

COMPONENTS

Threads :
- **Socket Handle** : Thread that sends and recieves data via network.
- **Double/String/Integer Loader** : Thread that loads data from InQueue into specific type of buffer based on type.
- **Double/String/Integer Drain** : Thread that takes data from output buffers from UI and separates them based on type storing them to OutQueue.

Buffers :
- **In Queue** : Contains all received data.
- **Out Queue** : Contains all data ment to be sent.
- **Ack Queue In** : Contains received ACK IDs of sent messages.
- **Ack Queue Out** : Contains ACK IDs of received messages ment to be sent.

- **Double/String/Integer Queue IN** : Each queue for each buffer (data) type incontaining data of matching type coming from other client ment to be read by UI.

- **Double/String/Integer Queue OUT** : Each queue for each buffer (data) type containing data of matching type coming from UI ment to be sent to other client.

Mutex-s :
- **3x QueueInMtx** (1 for each type)
- **3x QueueOutMtx** (1 for each type)
- **UIMtx**

NOTE : Data isn't copied, reading is based on pointers. Deleting is done by circular buffer overwriting.

TODO :

Razviti servis za razmenu podataka preko redova za poruke (message queueing service). Servis treba da omogući razmenu poruka između parova procesa koji se nalaze na dva računara, preko odvojenih redova poruka za svaki par. Svaki proces može da se zakači za tačno jedan red poruka na jednom servisu, a na jedan red poruka na jednom servisu može da se zakači najviše jedan proces.

Prilikom startovanja servisa, na obe strane se zadaju imena redova poruka na koje mogu da se zakače procesi. Kada se uspostavi veza između servisa, obe strane će razmeniti imena napravljenih redova. Servis implementira sledeći interface:

Connect(char * queueName), klijent se kači na odredjeni red na svom servisu. Ukoliko već postoje poruke, klijent ih prima po uspostavi konekcije. Ukoliko red queueName ne postoji, napraviti ga.

SendMessage(void* message, int messageSize), klijent šalje poruku drugom klijentu koji se zakačio na red sa istim imenom na drugom računaru. Servis treba da omogući pouzdan transport poruka. Poruka će biti sklonjena sa reda tek kada je isporučena odredištu. Ukoliko odredište ne postoji, sačuvati poruku dok se ono ne pojavi.

SEQUENCES

Initialize connection steps :
- Read Initial Queues data from QueConfig.txt
- Connect to target Queue

- Read TCP Port & IP from NetConfig.txt
- Initialize Listening Socket
- Fetch one and only connection to Accept Socket
- Dispose Listening Socket

- Request Queue Names
- Provide Queue Names
- Create Delta Queues
- Set input priority to connected queue
- Set output priority queue to none
- Request output priority queue from client, set if received

Service attach & use :
- Socket Handle thread periodically does WSASelect to detect events.
- Scheldue Input packages to matching queues prioritizing own priority. Followed by queue Mutex lock.
- Scheldue Input packages to matching queues prioritizing received client priority. Followed by queue Mutex lock.
- When done, free mutex and notify matching thread that reads matching queue.

- Each Loader thread reads from In Queue and passes to matching Console buffer. In that process it locks matching Network Layer Queue Mutex when reads, free it when its done in order to allow socket handle thread to handle new inputs. Prioritizing own priority. Locks matching Console buffer when writing, frees in order to allow UI to read it.

- Each Drain thread reads from matching Console buffer writing to Out Queue. In that process it locks matching Console Buffer Queue Mutex when reads, free it when its done in order to allow UI to put new requests. Prioritizing client priority. Locks matching Out Queue buffer when writing, frees in order to allow Socket Handle to read it and send it to the client.

- Each GC thread reads Ack Queue In, locking its mutex before and prioritizing client priority. When acks are read, it deletes them from Ack Queue In. When reading is done, it locks Console buffer mutex in order to free acked message, freeing Ack Queue In in order to receive new acks.

- UI thread consumes IN buffers of Console Buffer Layer prioritizing own priority. When each read is done, it puts ack into Ack Queue Out. It also fills OUT buffers of Console Buffer Layer with writing requests.

Queues as circular buffers :
- Push before start & Pop at stop

Uglješa Todorović PR87/2018
Nikola Oražanin PR6/2018