TCP Project README/Performance Analysis

Author: Shan Lu (slu5), Jingyiping Zhang (jzhang12)

For better reading experience on this document, please click here to open a rich-formatted PDF file.

Data Structures

- TCP

```
type TCP struct {
    connections map[string]*TransControlBlock // key:localAddr.String()+remoteAddr
.String()
    socketMap   map[int]*TransControlBlock
    verbose     bool
    lock        sync.RWMutex
    ran         *rand.Rand
    network     *NetworkLayer
}
```

Above is the global TCP structure, which contains 2 hashmaps mapping socket number and socket address pair to the corresponding `TransControlBlock`, which stores the specific attributes regarding this connection. Specifically, if a socket is built with address:port pair of (192.168.0.2:4323, 192.168.0.4:5000), and this socket is assigned with socket number 3, then both 3 and string "192.168.0.2:4323192.168.0.4:5000" can get access to the corresponding Transmission Control Block within constant time. This "redundant" design is mainly for time efficiency purpose.

Other attributes include a "verbose" option, which allows user to see what happened in this connection. Since map in Go is not thread-safe, we have to lock TCP when we update `connections` and `socketMap`. Finally, `network` is a pointer pointing to the network layer. Whenever a TCP packet has been wrapped, it will be passed through `network`'s `onRecvTCPData` method.

- Transmission Control Block

```
type TransControlBlock struct {
    sockfd          int
    localAddr       *net.TCPAddr
    remoteAddr      *net.TCPAddr
    state           StateTCP
    iotype          SocketIOType
    send_unack      uint32
    send_next       uint32
    send_window     uint16
    recv_next       uint32
    recv_window     uint16
    read_buf        *CircularBuffer
    write_buf       *CircularBuffer
    retransmit      bool
    dropRate        int
    retransmitQueue *Queue
    lock            sync.RWMutex
}
```

- Circular Buffer

```
type CircularBuffer struct {
    data               []byte
    size               int
    appReadPtr         int
    window_left        int
    window_right       int
    numRead            int64
    numWritten         int64
    initSeqNo          uint32
    lock               sync.Mutex
    lastGetTime        time.Time
    unorderedPacketMap map[uint32][]byte
}
```

- Queue