CNNS lab assign 4 的简短说明

Part 1. design decision about listening on a port

- 问题

```
a server must be listening on a port before a client can connect to it. However, a separate command to setup a server is not strictly required. You may arrange your server to listen on a default port number. However, you're encouraged to implement such a command. Please clearly document your design decision when you submit your assignment.
```

- decision

在本次 implement 中并没有增加一个新的命令行来监听初始化后监听的端口外的其他端口，在服务器初始化时默认采用接受到的命令行中不为 256 的端口。

TCPManage 中新建 socket，而不分配端口：

```
public TCPSock socket() {
    TCPSock sock = this.socks.newSocket();
    if (sock != null) {
        sock.setManager(this);
        TCPSockID sid = sock.getID();
        sid.setLocalAddr(this.addr);
    }
    return sock;
}

public TCPSock newSocket() {
    for(int i = 0; i < this.sz; ++i) {
        if (this.socks[i] == null) {
            this.socks[i] = new TCPSock();
            return this.socks[i];
        }
    }
    return null;
}
```

设置 server 端口相关代码：

```
public int getLocalPort() {
    return this.localPort;
}

public void setLocalPort(int localPort) {
    this.localPort = localPort;
}
// TCPSock.java
public int bind(int localPort) {
    if (this.id.getLocalPort() != 256) {
        return -1;
    } else if (localPort == 256) {
        return -1;
    } else {
        int localAddr = this.id.getLocalAddr();
```

```
            TCPSock sock = this.tcpMan.getLocalSock(localAddr, localPort);
            if (sock != null) {
                return -1;
            } else {
                this.id.setLocalPort(localPort);
                return 0;
            }
        }
    }
    将这些设置好端口的 socket 放进“线程池”（一个数组），需要时取用：
    public TCPSock getLocalSock(int localAddr, int localPort, TCPSock.State
state) {
        for(int i = 0; i < this.sz; ++i) {
            TCPSock sock = this.socks[i];
            if (sock != null) {
                TCPSockID id = sock.getID();
                if (id.getLocalAddr() == localAddr && id.getLocalPort() ==
localPort && (state == TCPSock.State.ANY || state == sock.getState())) {
                    return sock;
                }
            }
        }
        return null;
    }
```

Part 2: 所有包的通信建立操作

1. connect： 建立与另一个套接字的连接，处理 SYN 包的发送；通过设置 snd_base、snd_top、snd_next 等参数完成 sliding window 和 buffer 的初始化

```
public int connect(int destAddr, int destPort) {
        if (this.state != TCPSock.State.NEW) {
            return -1;
        } else if (this.id.getLocalPort() == 256) {
            return -1;
        } else {
            this.id.setRemoteAddr(destAddr);
            this.id.setRemotePort(destPort);
            this.snd_buf = new byte[16385];
            this.snd_initseq = this.snd_base = this.snd_top = this.snd_next =
this.tcpMan.initSeq(this);
            this.RTT_sample_seq = this.snd_base - 1;
            ++this.snd_top;
            this.state = TCPSock.State.SYN_SENT;
            this.sendSYN();
            return 0;
        }
    }
```

2. listen： 准备 accept

```
public int listen(int backlog) {
        if (this.state != TCPSock.State.NEW) {
            return -1;
        } else if (this.id.getLocalPort() == 256) {
            return -1;
        } else {
            this.pendingConnections = new LinkedList();
            this.backlog = backlog;
            this.state = TCPSock.State.LISTEN;
            return 0;
        }
    }
```

3. accept: 返回第一个等待连接的套接字

```
public TCPSock accept() {
        if (this.state != TCPSock.State.LISTEN) {
            return null;
        } else {
            return this.pendingConnections.isEmpty() ? null :
(TCPSock)this.pendingConnections.removeFirst();
        }
    }
```

Part 3: 对收到的包的处理和显示

- SYN：打印'S'；检查 sequence number；处理 sliding window、pendingConnection list；正式建立连接并发送 ACK
- FIN：检查是否正常接收并关闭连接
- ACK：检查 socket 状态并执行相应操作
- DATA：正常使用 sliding window 接受数据包