GIT, SFML, AND NETWORKING

CREATED BY KYLE PARKER

MONDAY, APRIL 7, 2025

GIT & GITHUB

GIT (DEFINITIONS)

- Version Control System (VCS)
  - Enables parallel versions
  - Provides audit trail
  - Enables collaboration
- Distributed system
  - Every developer has a copy of the repo on their local machine
  - Full offline capability

GIT (DEFINITIONS)

- Remote
  - Version of repository hosted on a server
- Local
  - Version of repository on your machine
- Push
  - Action of uploading local changes to the remote
- Pull
  - Action of pulling remote changes onto the local machine

GIT (DEFINITIONS)

- Fetch
  - Check on the remote if there are changes which need to be pulled
- Commit
  - A snapshot in of the repo
- Repository
  - Storage space for changes and files
- Delta
  - A change that is made

GIT

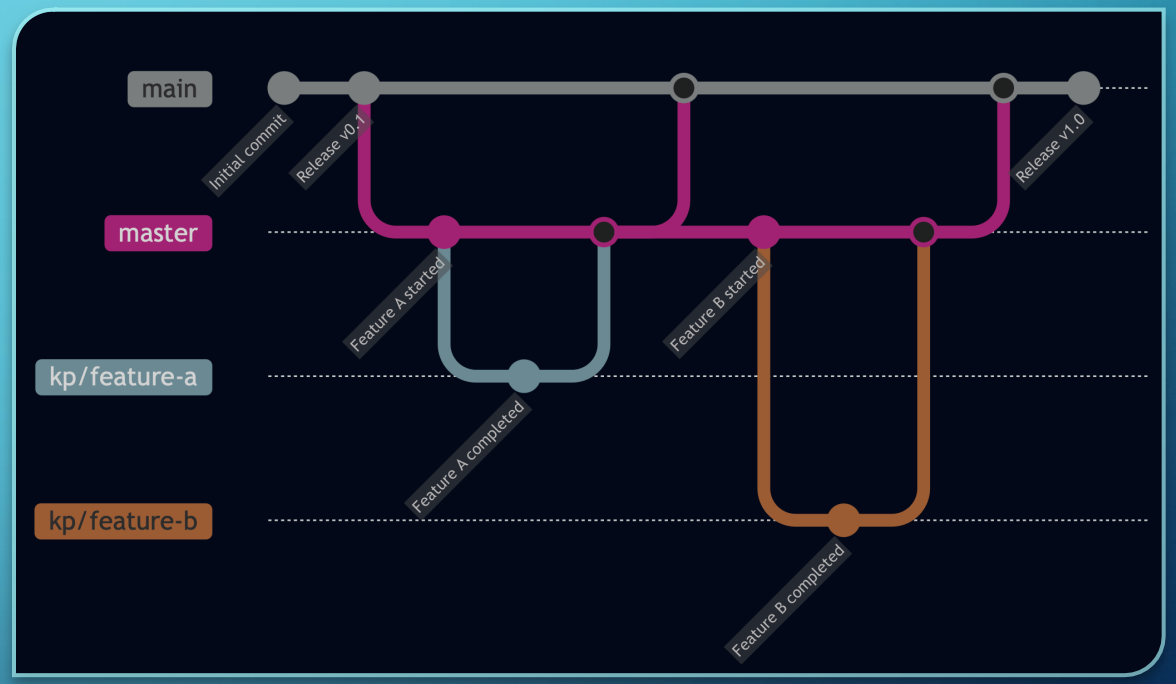
- When changes occur, only the changes are saved
  - Duplicate files are not maintained in the repo
- Not real-time (although there are some prototypes for GitHub)
- Communication is *very* important to ensure there are no conflicts

GIT (BRANCHES)

- Isolates changes from the main codebase
- Allows for parallel development
- Track progress of issues easier

GIT (BRANCH SUGGESTIONS)

- **Main is your default**
- **Create master to work from**
- If you want to use branching, use master as the base
- **Merge master into main after adding features**
- Branching is not required, but I strongly suggest using main and master for easy restoration



WHEN TO COMMIT

- After a new class is added w/ skeleton code
- After a function is updated or written
- After a refactor
- After MINOR changes are made
  - If you only wait for major changes, it will be hard to revert
- As you use git, this will become second nature

GIT (PROCESS)

1. Make changes to a file
2. Commit
3. Fetch
4. Pull (if needed)
5. Collision resolution (if needed)
6. Push

DOWNLOAD GIT CLI AND/OR GIT GUI

- Git CLI
  - Windows – https://git-scm.com/downloads/win **Download and run installer**
  - macOS – https://brew.sh `brew install git`
  - Linux – (may be already installed) `sudo apt update && sudo apt install git`
- GitHub Desktop
  - https://desktop.github.com/download/
- [Alternative] SourceTree (More Advanced) `brew install sourcetree`
  - https://www.sourcetreeapp.com/

SFML

DOWNLOAD SFML CMAKE PROJECT

- https://tinyurl.com/yc7j7dpm

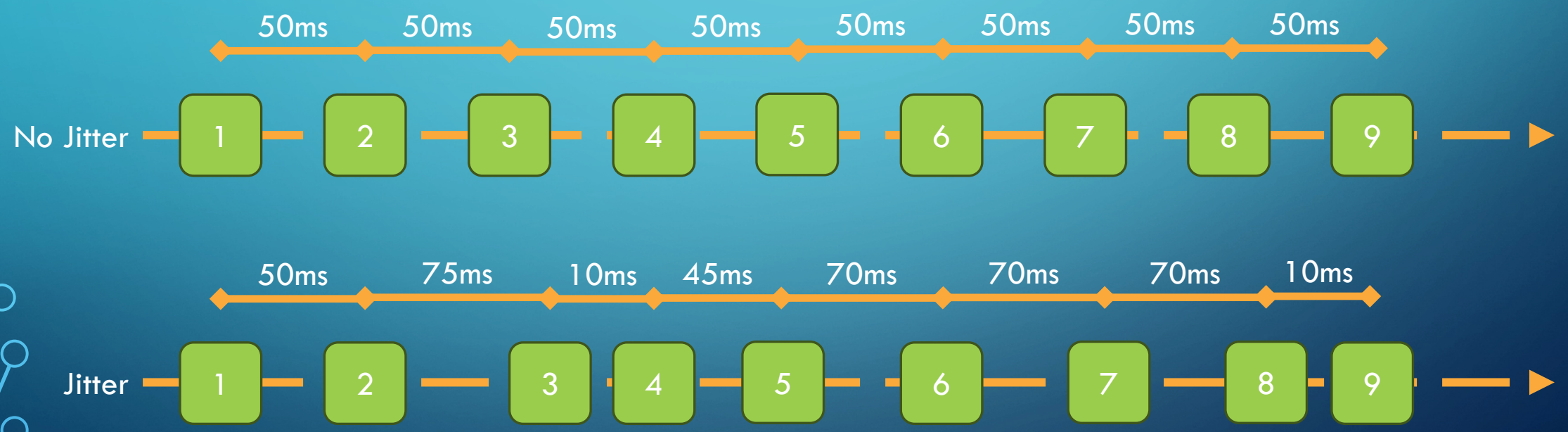
NETWORKING

TERMINOLOGY (HIGH LEVEL)

- Packet
  - A unit of data which is transmitted from point a to b.
- LAN
  - Local Area Network – home, school, office.
- WAN
  - Wide Area Network – contains multiple LANs such as the Internet.
- Latency
  - The time it takes for a packet to be delivered. Typically measured in milliseconds (ms).

TERMINOLOGY (HIGH LEVEL)

- Jitter
  - The difference in arrival times of packets.

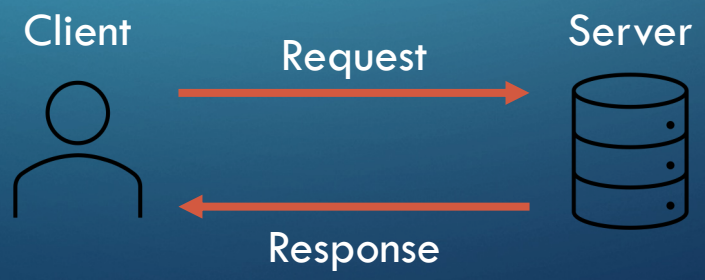


TERMINOLOGY (HIGH LEVEL)

- DHCP – Dynamic Host Configuration Protocol
  - The server which gives you an IP Address.
- TCP – Transmission Control Protocol
  - Ensures ordered and error-checked delivery of packets.
  - Good for scenarios such as game play.
- UDP – User Datagram Protocol
  - Does **not** ensure delivery of packets nor is the order guaranteed.
  - Good for scenarios such as video.

TERMINOLOGY (HIGH LEVEL)

- VPN – Virtual Private Network
  - Creates a secure connection over the Internet. Allows you to act as if you were connected directly to a private network.
- Server
  - A resource which accepts connections. Serves responses from a request.
- Client
  - A resource which connects to a server.



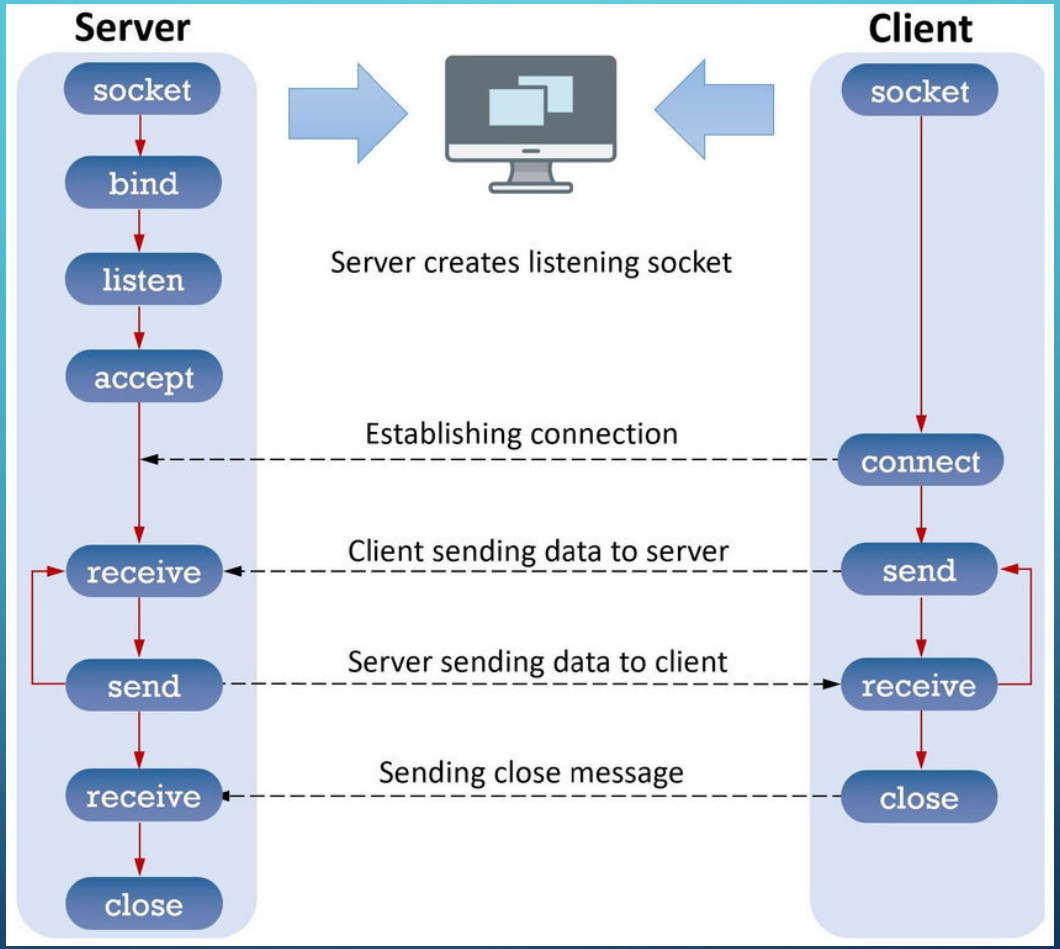
RESTRICTIONS WITH CONNECTING TO ANOTHER COMPUTER

- You may need a VPN if you are not all on the same network
- Sometimes networks will block certain connections to local machines
  - I am not aware of any restrictions on campus at WSU
- Firewalls on your device preventing incoming and/or outgoing connections to devices on the same network

HOW TO TEST TX AND RX

- macOS/Linux/Windows (Terminal or CMD Prompt)
  - `ping <IP>` **Example:** `ping apple.com`
  - `telnet <IP>` **Example:** `telnet 192.168.1.2`
    - May be unavailable for macOS
  - `nc <IP> <PORT>` **Example:** `nc 192.168.1.2 23`
    - Only for macOS and Linux by default
    - `nc -l -p <PORT>` **Example:** `nc -l -p 23`
      - Listen on port 23. *Note: -p is unnecessary on some machines*
    - **Important: You may have to press return twice when using nc to send commands**
  - `netstat`
    - Display current network connections (can tell you if the client/server is connected)
  - `nmap <OPTIONS> <IP>` **Example:** `nmap -A 192.168.1.2`
    - This is a great security tool, **use it responsibly!**

TCP CLIENT-SERVER SOCKET FLOW



NETWORKING – SFML SPECIFICS

SHOULD WE USE UDP OR TCP?

SHOULD WE USE UDP OR TCP?

- TCP because it ensures the data is sent and received
  - UDP should work, but could silently fail so we prefer TCP

```
// Server side

sf::TcpListener l;

l.listen(PORT_NUM);

l.accept(sf::TcpSocket);
```

```
// Client Side

sf::TcpSocket.receive(sf
::Packet);
```

WHAT IS A THREAD?

- A way of executing code “concurrently”
- Too many threads can degrade program performance
  - Frequent context switches slow down the app
- A thread is not a separate core (depends on your scheduler)
- Concurrency as used above does not mean at the same exact time
  - It refers to the perception they are running at the same time
- If threads have short tasks,

WHEN TO USE THREADS

- Audio
- Socket connections
- Anything you use which is considered “blocking”
  - Blocking means the thread will wait for the function to finish
    - If you have animation and sound, it will play the sound, then animate

HOW TO USE THREADS

```
// Important Note: No args can be passed
// You need to use a lambda function
sf::Thread threadName(&nameOfFunc);
threadName.launch(); // Do not forget this line
threadName.join();
// In std::thread, you can add args as below
std::thread threadName(&nameOfFunc, args…);
threadName.join();
```

DANGERS OF THREADS

- Race conditions
  - Occurs when two statements try to read or write to the same variable at the same time
  - Use mutex to prevent races; it locks variables so only one thread can use it
- Dead locks
  - Occurs when thread A holds lock 1 and waits for lock 2 while thread B holds lock 2 and waits for lock 1
- Starvation
  - A thread uses most of the CPU time and does not give up CPU time for others
- Some functions may not be thread safe
  - Think: Does this function or method use anything outside of this scope (member(s) of a class, global variables, etc.)

STD::OPTIONAL

```
// Suppose we are in some class
std::optional<sf::Packet> incomingPacket;
std::optional<sf::Packet> outgoingPacket;
// Check for value
if (incomingPacket.has_value());
// Get value (Must check for value first)
incomingPacket.value();
// Set empty value
incomingPacket = std::nullopt;
```

CREATE A POLLING LOOP (INEFFICIENT)

```
while (true) {
        if (this->incomingPacket.has_value()) {
                handleIncomingPacket();
        }
        if (this->outgoingPacket.has_value()) {
                sendOutgoingPacket();
        }
}
```