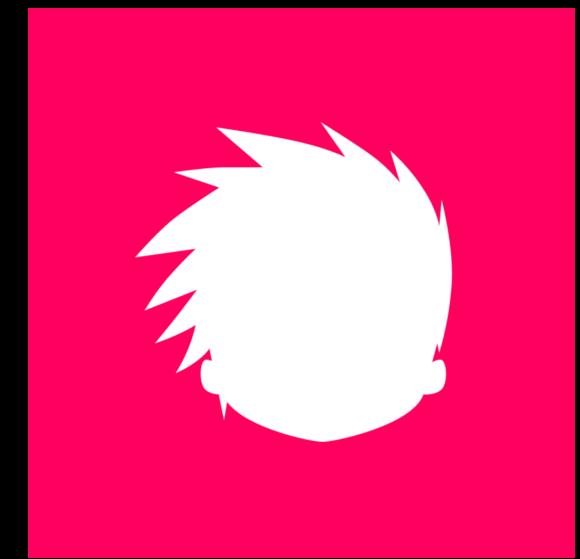
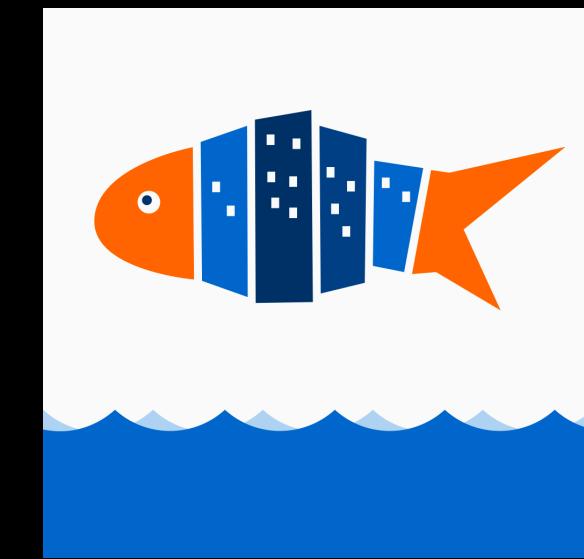


MULTIPEERCONNECTIVITY

Hello



NEARBY

**PROXIMITY, DEPENDS ON THE WIRELESS
TECHNOLOGY IN USE**

PEER

BUDDY

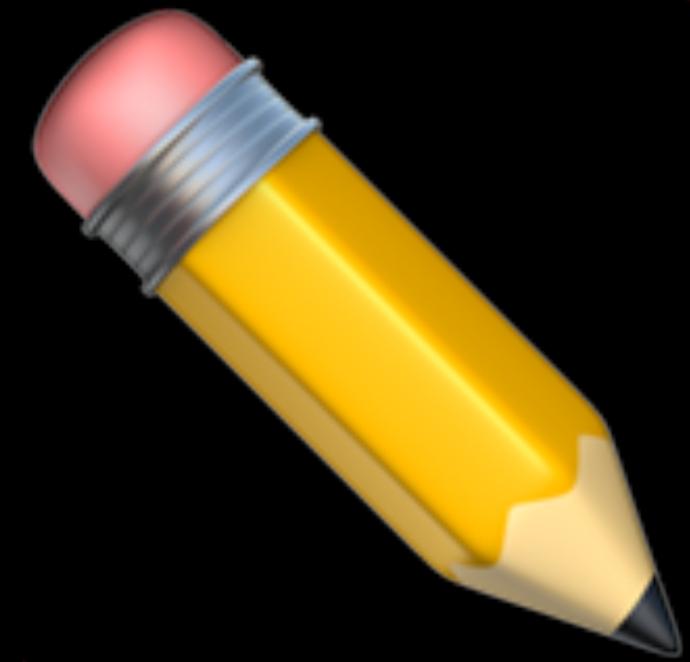
**BUDDY,
BUDGY
NEARBY DEVICE**

ADVERTISER

**THE DEVICE THAT OFFERS A
SERVICE OVER THE AIR**

BROWSER

THE DEVICE THAT'S LOOKING
FOR OTHER DEVICES OVER THE
AIR



PHASES

DISCOVERY

SESSION

TRANSMISSION

DATA

```
{  
  "hello": "world"  
}
```

```
let peers = ...
let mode = ... // (.reliable | .unreliable)

guard let message = "{\"hello\": \"world\"}.data(using: .utf8) else { return }

do {
    try session.send(data, toPeers: peers, with: mode)
} catch {
    // something went wrong
}
```

RELIABLE

UNRELIABLE

Data



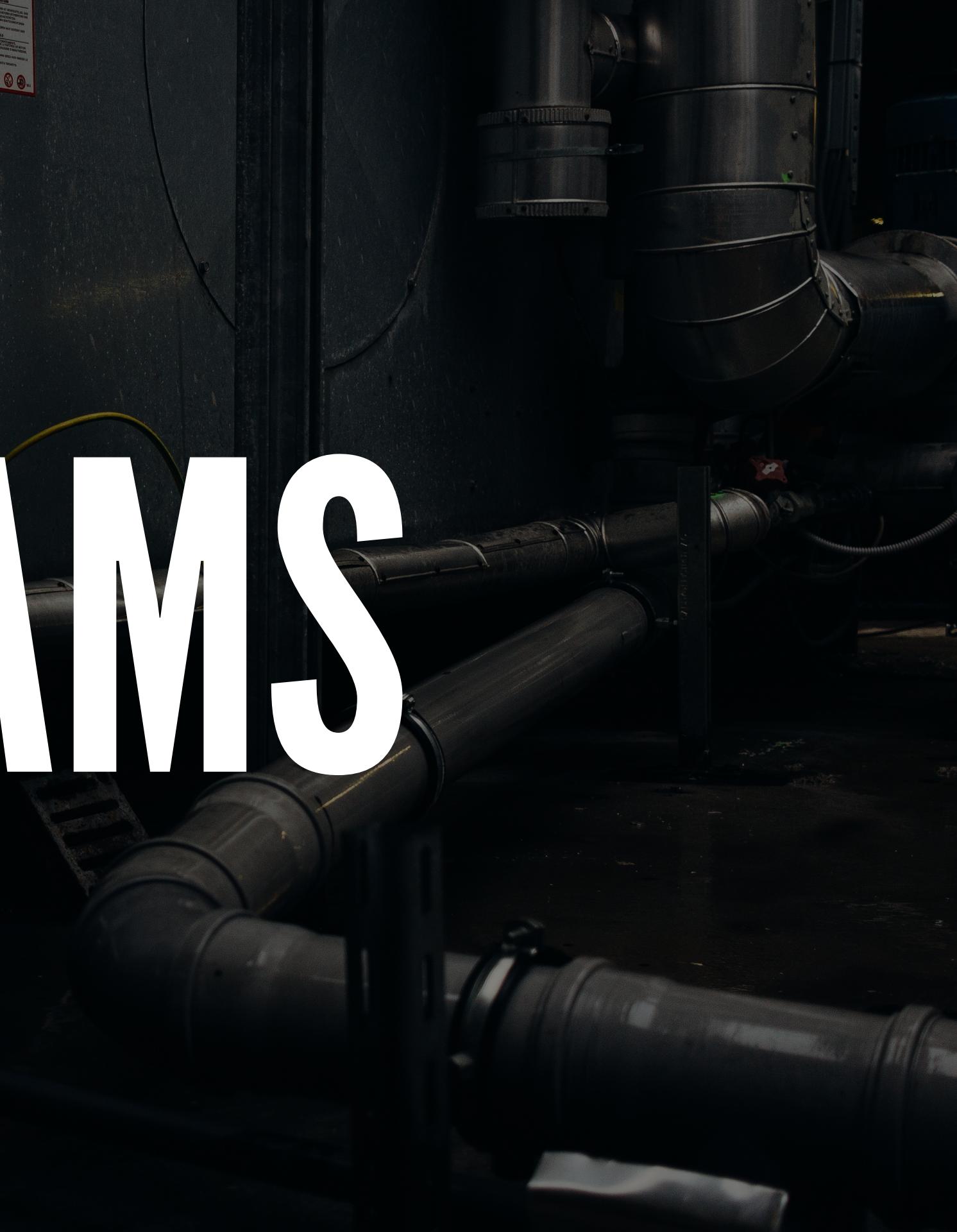
```
struct User: Codable {  
    let id: UUID  
    let name: String  
}
```

```
struct Message: Codable {  
    let sender: User  
    let contents: String  
}
```

```
let message = Message(  
    sender: User(id: UUID(), name: "Guilherme Rambo"),  
    contents: "Hello, peer!"  
)  
  
do {  
    let data = try PropertyListEncoder().encode(message)  
  
    try session.send(data, toPeers: peers, with: mode)  
} catch {  
    // something went wrong  
}
```



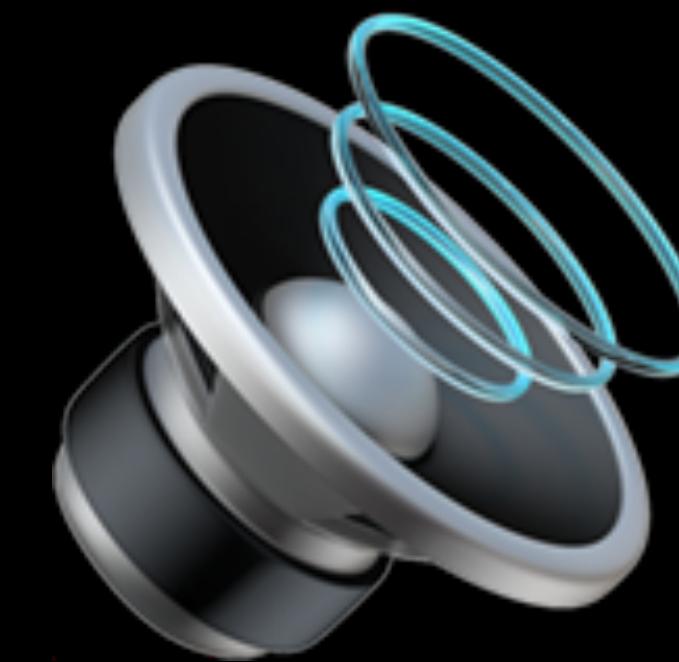
STREAMS



```
do {
    let outputStream = try session.startStream(withName: "Audio", toPeer: peer)
} catch {
    // something went wrong
}
```

```
let lengthWritten = outputStream.write(myAudioBuffer, maxLength: length)
```

```
func session(_ session: MCSession,  
            didReceive stream: InputStream,  
            withName streamName: String,  
            fromPeer peerID: MCPeerID)  
{  
    mySuperSpecialAudioQueue.async {  
        var lengthRead: Int = -1  
  
        repeat {  
            lengthRead = stream.read(&myAudioBuffer, maxLength: length)  
  
            // send to CoreAudio or something  
        } while lengthRead > 0  
    }  
}
```



RESOURCES

```
let progress = session.sendResource(at: fileURL,  
                                  withName: filename,  
                                  toPeer: peer,  
                                  withCompletionHandler: completion)  
  
let observer = progress?.observe(\.fractionCompleted) { _, change in  
    guard let fraction = change.newValue else { return }  
  
    print("Upload progress: %.2f%", fraction * 100)  
}
```

```
func session(_ session: MCSession,  
            didStartReceivingResourceWithName resourceName: String,  
            fromPeer peerID: MCPeerID,  
            with progress: Progress)  
{  
    myCurrentProgressObserver = progress.observe(\.fractionCompleted) { _, change in  
        guard let fraction = change.newValue else { return }  
  
        print("Download progress: %.2f%", fraction * 100)  
    }  
}
```

```
func session(_ session: MCSession,  
            didFinishReceivingResourceWithName resourceName: String,  
            fromPeer peerID: MCPeerID,  
            at localURL: URL?,  
           WithError error: Error?)  
{  
    if let error = error {  
        // something went wrong  
        return  
    }  
  
    myCurrentProgressObserver.invalidate()  
  
    guard let url = localURL else { return }  
  
    do {  
        try FileManager.default.moveItem(at: url, to: finalURLForReceivedFile)  
    } catch {  
        // something went wrong  
    }  
}
```



DEMO

THANK YOU

**TWITTER.COM/ INSIDE
STACKTRACEPODCAST.FM
DEVCHAT.TV/IPHREAKS**