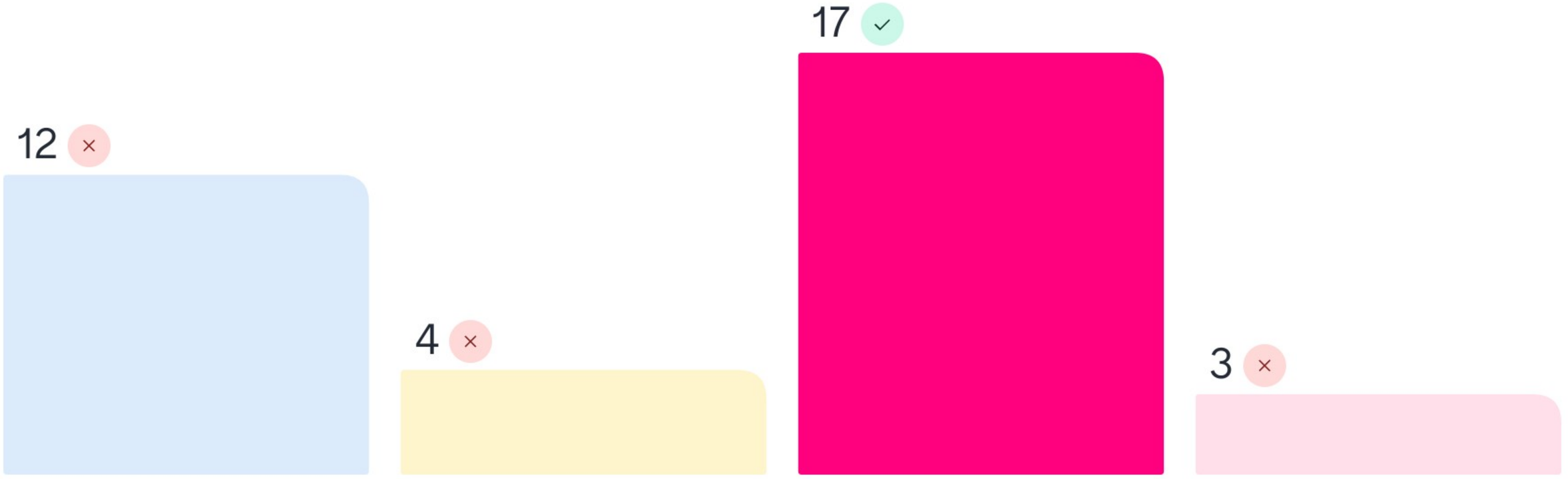
ENGF0034 Week 9

Part I

Introduction

- This week we focus on software specifications and their role in ensuring interoperability
- We discuss different types of specifications, including requirement specifications and formal specifications
- The emphasis is on specifications designed for interoperability, enabling different software components to communicate effectively
- We use the Pac-Man game to illustrate the key concepts

What is the primary purpose of a software specification?



1 22

What are the three main types of specifications?

Requirements specification, formal specification, protocol specification

12 Popular

talk tuah podcast? Those who know

7

protocol, formal, syntax

5

what's my tetris grade?

2

Can we instead specify why 9 am lectures exist

2

user, requirement, protocol

2

Network

1

Protocol

1

1 7

What are the three main types of specifications?

syntax

Imagine you're developing a multiplayer game. Describe a potential problem that could arise if there's no clear specification between devices

Crypto mining in the background 7 Popular	Martin benning 7 Popular	Parsing errors 5	desync 5
loss of data packets mid transmission 5	No cross platform compatibility 4	Lagging 4	No synchronisation between devices 4

Imagine you're developing a multiplayer game. Describe a potential problem that could arise if there's no clear specification between devices

managing microtransactions

3

no syntax to encode data

3

Chat are we cooked

2

connection doesn't work

2

loss of tranmission mid data packets

2

users keep losing and I stay on top

1

android users can't play it, womp womp

1

lag

1

12

Imagine you're developing a multiplayer game. Describe a potential problem that could arise if there's no clear specification between devices

Game crash

Why Specifications Matter

- Writing a specification is crucial for ensuring that different software components can work together seamlessly
- Without a clear specification, developers may create software that cannot communicate effectively, leading to compatibility issues
- A specification acts as a contract, outlining how different parts of a system should interact
- In the Pac-Man example, a specification dictates how two instances of the game, potentially written in different languages or running on different operating systems, can exchange information and function as a unified game

What comes to mind when you hear the term "interoperability"?

32 responses

Key Elements of Interoperability

- Achieving interoperability hinges on two main aspects: syntax and semantics
- **Syntax**: Deals with the encoding of information, specifying how data is represented for transmission (e.g., binary vs. text encoding, byte order for integers)
- **Semantics**: Focuses on the meaning of messages exchanged between software components

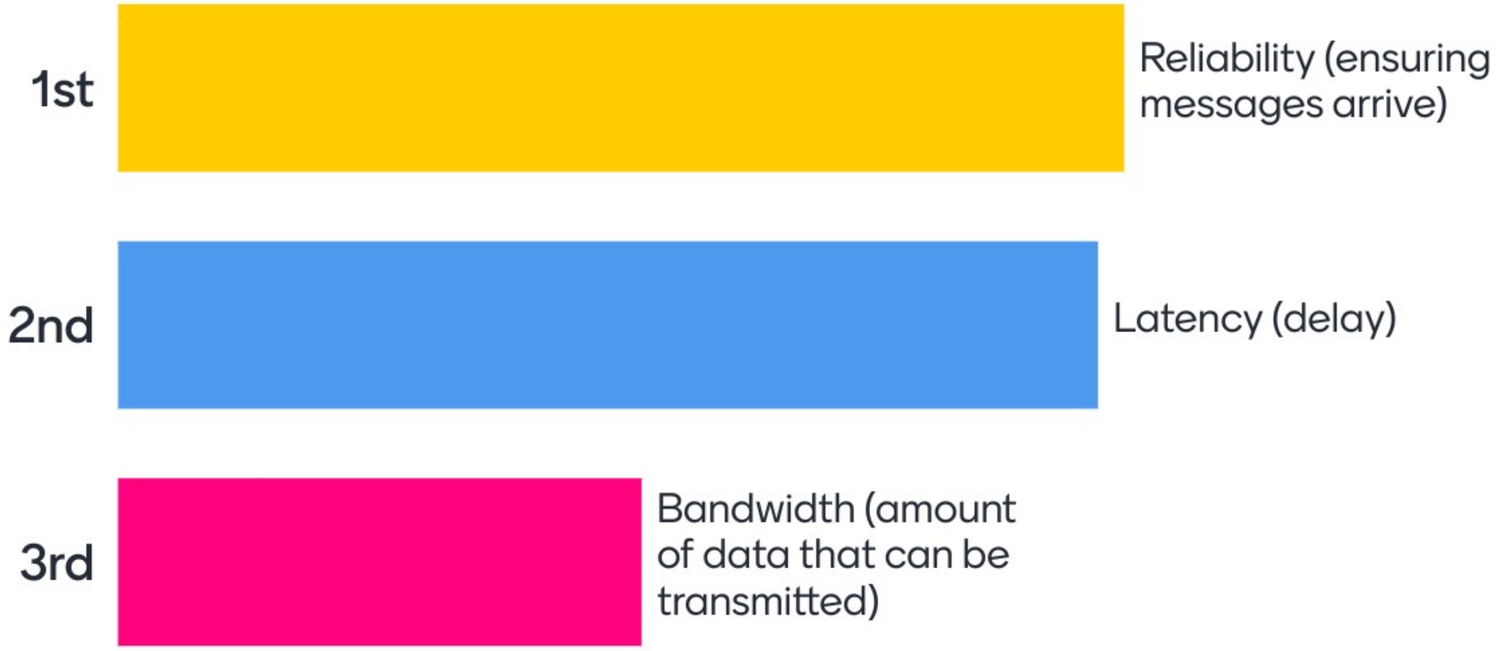
Semantics in Detail

- Understanding the semantics of a system involves defining:
- **When** messages are sent (e.g., triggering an update when Pac-Man changes direction or eats food)
- **What** information each message carries (e.g., Pac-Man's position, direction, speed)
- **How** the recipient should react to a message (e.g., updating the display based on Pac-Man's new position)

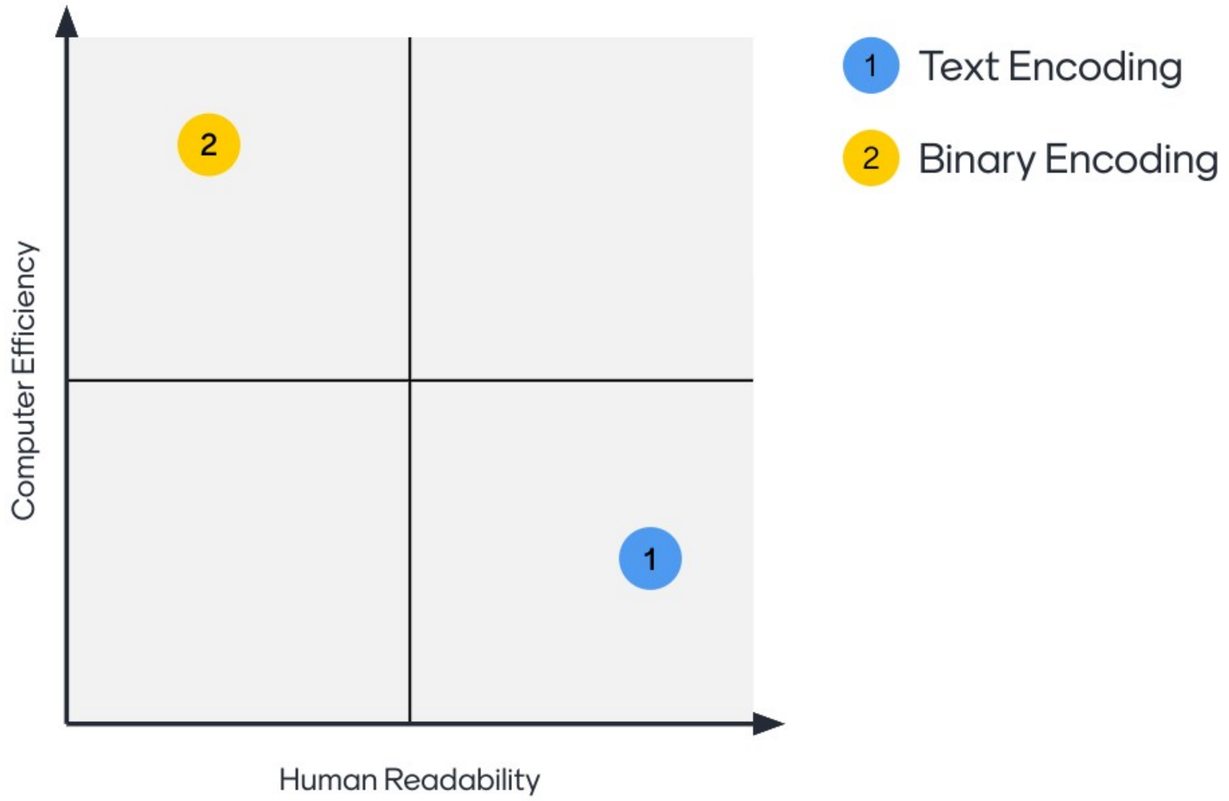
Handling Network Imperfections

- Network communication is rarely perfect, introducing challenges like message loss and delays
- The choice of network protocol (UDP or TCP) influences how these imperfections are addressed
- **UDP**: Offers speed but lacks reliability, meaning messages can be lost
- **TCP**: Ensures message delivery but can experience delays due to congestion control
- The specification must address how the sender and receiver handle lost messages (if using UDP) or delayed messages (if using TCP)

Rank the following network considerations from MOST to LEAST important for a real-time multiplayer game



Textual vs. Binary Encoding



Textual Encoding

- Human-readable, offering easier debugging and initial implementation
- Flexible, accommodating a wide range of data structures through a defined grammar
- Can be less efficient for computers to process compared to binary
- Requires parsing to extract information, adding complexity to the receiver

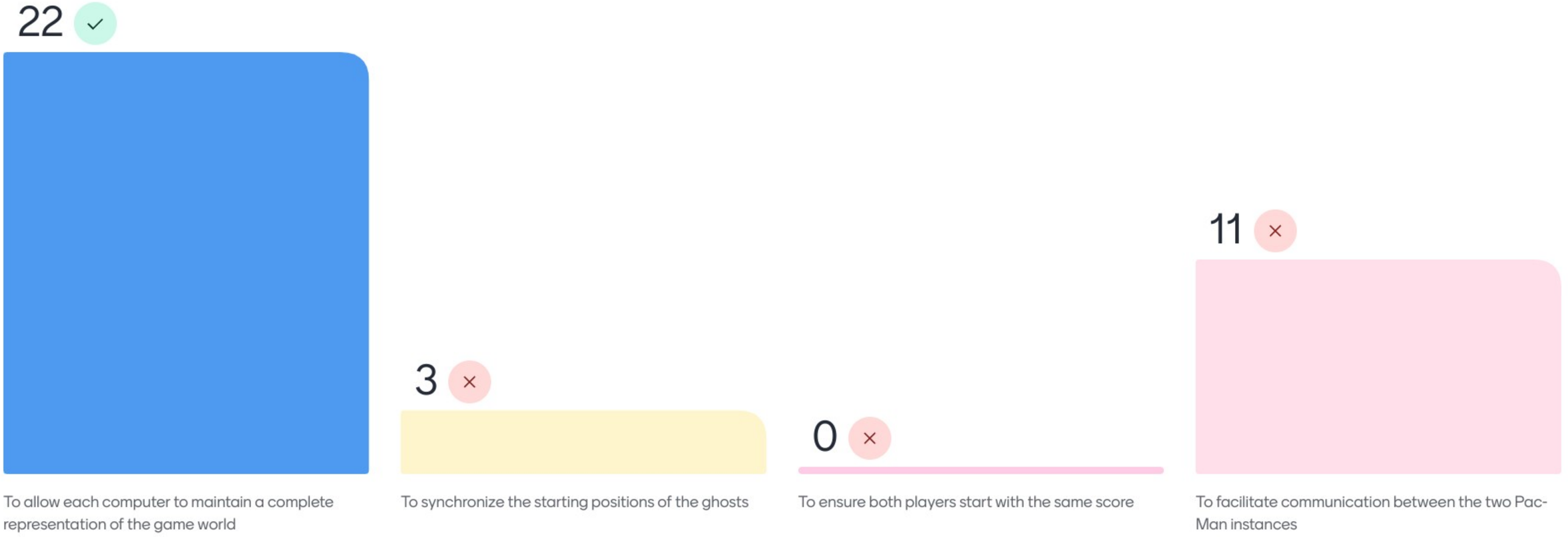
Binary Encoding

- Efficient and compact, minimising network overhead
- Easier for computers to process, potentially offering performance benefits
- Less human-readable, making debugging more challenging
- Limited flexibility, typically suited for fixed data structures

Pac-Man Protocol Analysis

- The existing Pac-Man game uses Python's Pickle for message encoding, but this approach has limitations
- **Downsides of Pickle:**
- Python-specific, hindering interoperability with games written in other languages
- Potential security risks as it allows loading arbitrary data structures, making it vulnerable to malicious input
- **The goal of the assignment is to design a better protocol that overcomes these limitations**

In the Pac-Man game, what is the primary purpose of exchanging maze layouts between two connected computers?



To allow each computer to maintain a complete representation of the game world

To synchronize the starting positions of the ghosts

To ensure both players start with the same score

To facilitate communication between the two Pac-Man instances

Exploring Pac-Man's Semantics

- The game involves two computers, each maintaining a model of the game world, including mazes, Pac-Man, and ghosts
- Upon connection, mazes are exchanged so each computer has a complete representation of the game environment
- As the game progresses, messages communicate updates about Pac-Man and ghost positions, food consumption, and other events
- Understanding the types of messages, their content, and when they are sent is crucial for designing the new protocol

Understanding Network Interactions

- Whether using a client-server or client-client-server setup, the underlying message exchange between Pac-Man instances remains the same
- The server in the client-client-server model acts as a simple relay, forwarding messages without modification
- This understanding simplifies protocol design, as the focus is on direct communication between game instances regardless of the network setup

Explain the role of a server in a client-client-server architecture, like the one optionally used in the Pac-Man game

Acts as a relay between the two clients

5 Popular

to relay the communication from one client to another

4

Relay innit

3

it's relay mate

3

Receives messages from Client A and forwards it onto Client B

2

acts as a intermediary between the two

2

Surely it's relay

2

acts as relay between clients, passing information along

1

Explain the role of a server in a client-client-server architecture, like the one optionally used in the Pac-Man game

Can't be relay

1

Relay message

Relay

Forwards the message without crashing the programme

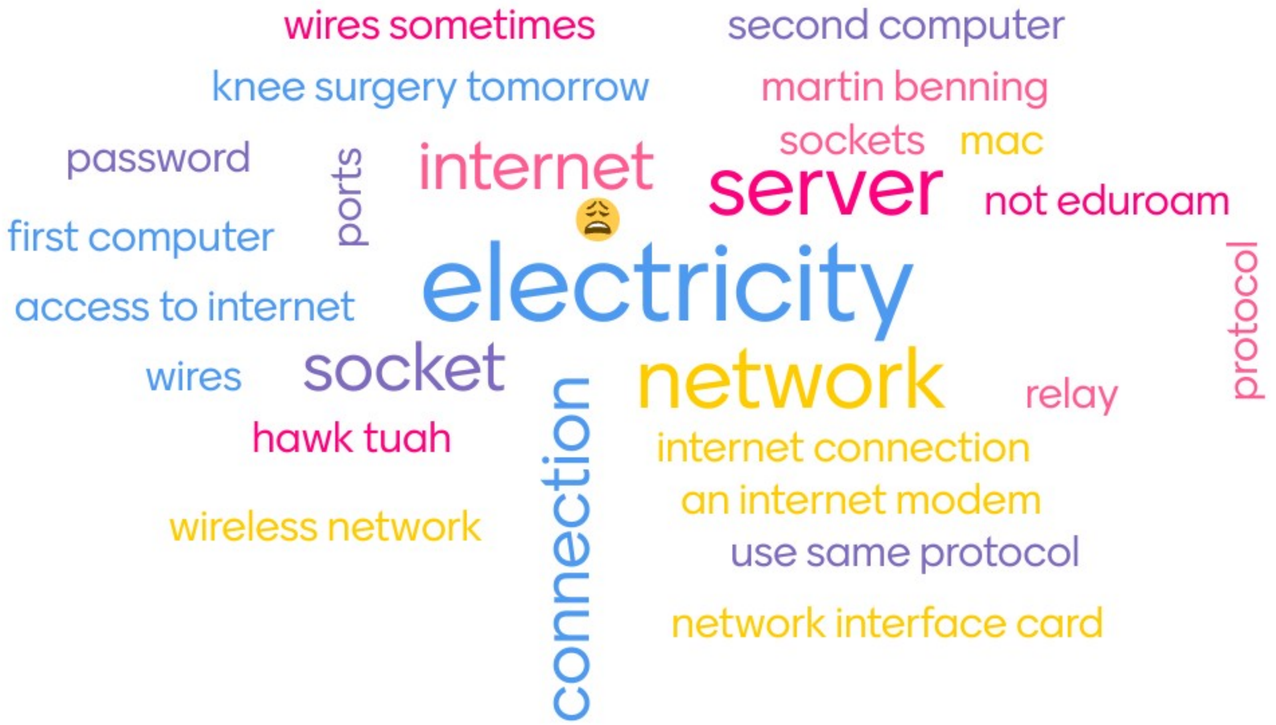
It allows the clients to interact even if one computer is behind a firewall

1

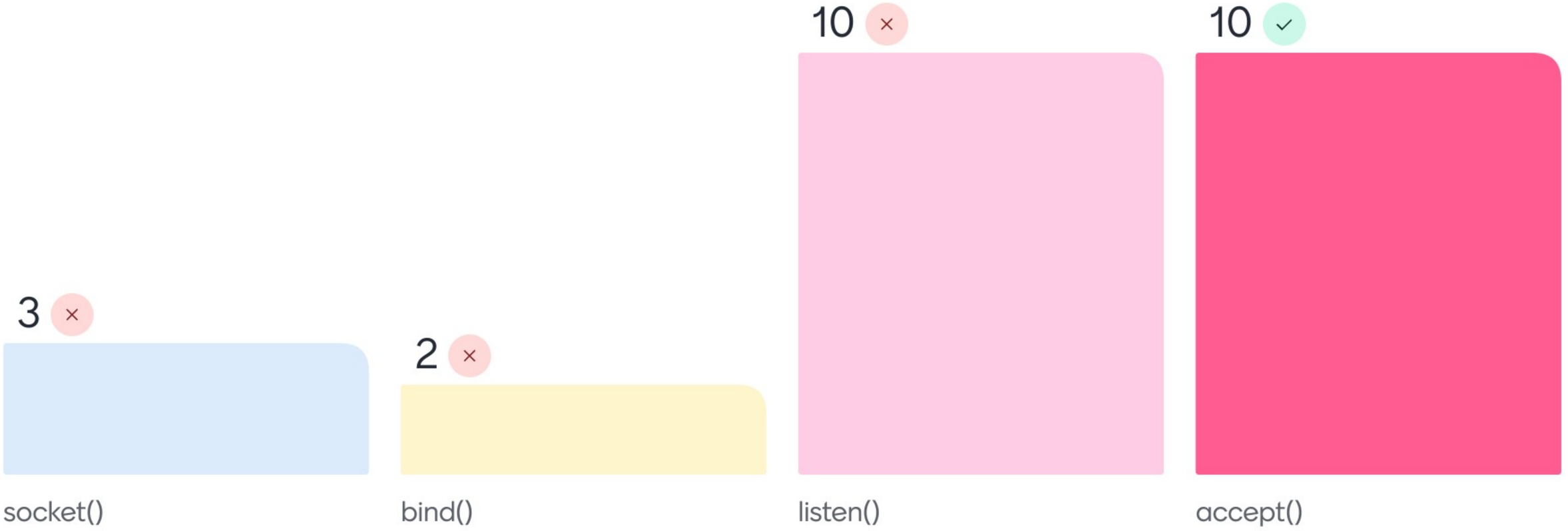
12

What are the essential components needed to establish a network connection between two computers?

38 responses



Which of the following is responsible for accepting an incoming connection on the server side?



Network Communication

- **Network Sockets**: The foundation of network communication is the socket, an endpoint that facilitates data exchange. Sockets are created using the **socket()** function in Python, specifying parameters like the address family (**AF_INET** for internet communication) and the socket type (**SOCK_STREAM** for the reliable TCP protocol)
- **Client and Server Roles**: A clear distinction is made between the roles of a client and a server in a network application. The server is set up to listen for incoming connections, while the client actively initiates a connection to a server
- **Binding to a Port**: The server uses the **bind()** function to associate its socket with a specific port number. This acts as the address on the server machine where the client can connect
- **Listening for Connections**: The server then uses **listen()** to actively listen on the specified port for any incoming connection requests from clients
- **Client Connection Initiation**: The client uses the **connect()** function to initiate a connection to the server, providing the server's IP address and the port number it is listening on
- **Accepting the Connection**: On the server side, the **accept()** function is used to accept an incoming client connection. This creates a new socket dedicated to handling the communication with that specific client

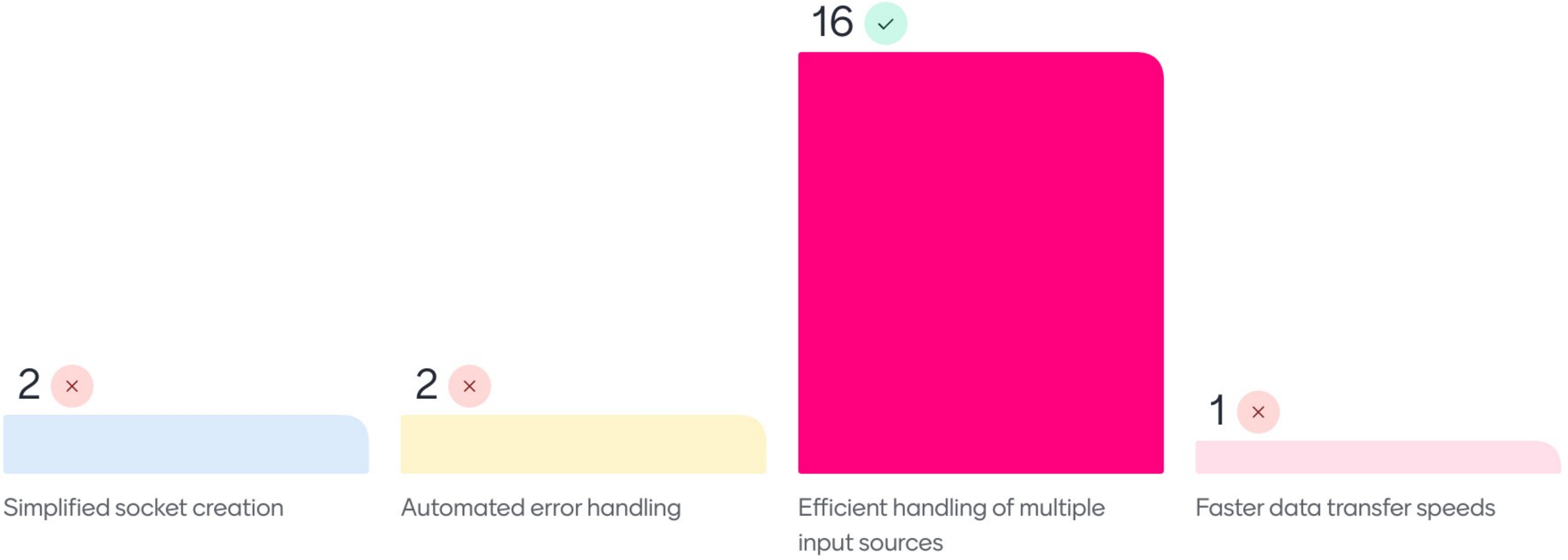
Data Flow and Ensuring Robustness

- **Data Exchange**: Once the connection is established, data can flow bi-directionally between the client and the server. The **send()** function transmits data from one end, while the **receive()** function retrieves data on the other
- **Text Encoding**: Python's default encoding is UTF-8, and this encoding is applied before sending and decoded after receiving to handle text correctly
- **Error Handling Essentials**: Video 14 emphasizes the need for robust error handling in network applications. Common errors that need to be addressed include:
- **ConnectionRefusedError** occurs when trying to connect to a server that is not listening
- **BrokenPipeError** signifies a disconnection from the other end
- **End-of-File (EOFError)** typically occurs when the user sends an end-of-file signal (like Ctrl+D)
- **Handling with try-except**: Python's try-except blocks are used to gracefully manage these exceptions, preventing the program from crashing and allowing for appropriate actions, such as reconnecting or informing the user

Addressing Blocking I/O

- **The Blocking I/O Dilemma**: Video 14 explains that functions like **input()** (for keyboard input) and **receive()** can block program execution
- This means the program pauses and waits until input is available.
- While this behavior is often desired, it can lead to unresponsiveness in interactive applications
- **Exploring Non-Blocking I/O**: One way to address blocking is to use non-blocking versions of these functions
- The source demonstrates this using **non_blocking_readline()**
- It is important to note that this function is not part of the Python standard library and you might need to obtain its implementation elsewhere if you wish to use it
- **Challenges with Non-Blocking I/O**: While non-blocking I/O prevents the program from hanging, it often introduces the need for "busy waiting"
- This involves repeatedly checking for input, leading to inefficient use of CPU resources

What is the primary advantage of using the **select()** function in network applications?



2 16

The select() Function for Efficiency

- **Introducing select()**: Python's **select()** function is a powerful tool to manage multiple input sources efficiently without the overhead of busy waiting
- **Monitoring for Readiness**: The **select()** function allows you to provide lists of file descriptors (representing sockets, files, etc.) that you want to monitor for specific events:
- **Reading**: Is data available to read from the descriptor?
- **Writing**: Is there space available to write data to the descriptor?
- **Exceptions**: Are there any exceptional conditions on the descriptor?
- **Sets of Ready Descriptors**: select() returns sets of descriptors that are ready for each of these operations
- **Conditional Execution**: The program can then check if the specific descriptors it cares about (the network socket and sys.stdin for keyboard input) are present in the returned sets
- This enables conditional execution – the program only attempts to read from descriptors that are actually ready, avoiding blocking and wasted CPU cycles

Applying select() in Chat Application

- **Integrating select()**: Video 14 provides concrete examples of how to integrate **select()** into the chat application's main loop
- The network socket and **sys.stdin** are added to the list of file descriptors monitored for reading
- **Checking for Input Readiness**: An **if** condition is used to check if sys.stdin is present in the set of descriptors ready for reading
- If so, it indicates keyboard input is available and **sys.stdin.readline()** is called to retrieve it
- **Checking for Network Data**: Similarly, another **if** condition checks if the network socket is in the ready-for-reading set
- If it is, the program calls **receive()** to retrieve the data from the network
- **Efficiency and Responsiveness**: This approach ensures that the program only attempts to read data when it is actually available, maximizing efficiency and responsiveness
- The program blocks only at the **select()** call, waiting for either keyboard input or network data

What is sys.stdin used for and how does it work?

As an input buffer that accepts user input from the console. The input is finally processed by the program when the buffer when it is cleared or flushed

8

Popular

Reads standard input from the keyboard

5

sys.stdin is a built-in Python object that provides a way to interact with the standard input stream, which is usually the keyboard. It allows you to read data from the user or from redirected input,

5

getting standard input from the system

4

Standard input innit

3

standard system input STRING mate

1

it takes in all the input from the keyboard until ctrl+d

1

takes all of the input some of the time

1

What is sys.stdin used for and how does it work?

Standard system input

Tkes some of the input all of the time

How exactly does the select function work? What are the three outputs and what are the three list arguments?

We have to write a 3000 word report 😭😭😭😭

7 Popular

Martin can we go home 👉👈🥺

6

mate, I'm so tired, I have no idea

4

Read, write, exceptions

4

read data, write data, and exceptions data

4

+1

3

👉👈

2

#include <iostream> #include <vector> #include <algorithm> int select(std::vector<int>& arr, int k) { std::nth_element(arr.begin(), arr.begin() + k, arr.end()); return arr[k]; }

1 7

How exactly does the select function work? What are the three outputs and what are the three list arguments?

select in STL/Algorithms Context (Median-of-Medians or nth Element) The select function in algorithms is often used to find the *k* k-th smallest (or largest) element in an unsorted array. It is commo

Please ask your questions

14 questions
9 upvotes