Announcements
Distributed Computing
Distributed Computing
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task.
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task.

- Computation is performed in parallel by many computers.
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task:

- Computation is performed in parallel by many computers.
- Information can be restricted to certain computers.
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task

- Computation is performed in parallel by many computers
- Information can be restricted to certain computers
- Redundancy and geographic diversity improve reliability
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task:

• Computation is performed in parallel by many computers
• Information can be restricted to certain computers
• Redundancy and geographic diversity improve reliability

Characteristics of distributed computing:
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task

• Computation is performed in parallel by many computers
• Information can be restricted to certain computers
• Redundancy and geographic diversity improve reliability

Characteristics of distributed computing:
• Computers are independent — they do not share memory
A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task

- Computation is performed in parallel by many computers
- Information can be restricted to certain computers
- Redundancy and geographic diversity improve reliability

Characteristics of distributed computing:

- Computers are independent — they do not share memory
- Coordination is enabled by messages passed across a network
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task

- Computation is performed in parallel by many computers
- Information can be restricted to certain computers
- Redundancy and geographic diversity improve reliability

Characteristics of distributed computing:

- Computers are independent — they do not share memory
- Coordination is enabled by messages passed across a network
- Individual programs have differentiating roles
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task

• Computation is performed in parallel by many computers
• Information can be restricted to certain computers
• Redundancy and geographic diversity improve reliability

Characteristics of distributed computing:
• Computers are independent – they do not share memory
• Coordination is enabled by messages passed across a network
• Individual programs have differentiating roles

Distributed computing for large-scale data processing:
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task

• Computation is performed in parallel by many computers
• Information can be restricted to certain computers
• Redundancy and geographic diversity improve reliability

Characteristics of distributed computing:
• Computers are independent — they do not share memory
• Coordination is enabled by messages passed across a network
• Individual programs have differentiating roles

Distributed computing for large-scale data processing:
• Databases respond to queries over a network
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task

• Computation is performed in parallel by many computers
• Information can be restricted to certain computers
• Redundancy and geographic diversity improve reliability

Characteristics of distributed computing:
• Computers are independent — they do not share memory
• Coordination is enabled by messages passed across a network
• Individual programs have differentiating roles

Distributed computing for large-scale data processing:
• Databases respond to queries over a network
• Data sets can be partitioned across multiple machines (next lecture)
Network Messages
Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network.
Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network.

Messages can serve many purposes:
Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network

Messages can serve many purposes:
- Send data to another computer
Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network

Messages can serve many purposes:
• Send data to another computer
• Request data from another computer
Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network.

Messages can serve many purposes:
- Send data to another computer
- Request data from another computer
- Instruct a program to call a function on some arguments
Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network

Messages can serve many purposes:
• Send data to another computer
• Request data from another computer
• Instruct a program to call a function on some arguments
• Transfer a program to be executed by another computer
Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network.

Messages can serve many purposes:
• Send data to another computer
• Request data from another computer
• Instruct a program to call a function on some arguments
• Transfer a program to be executed by another computer

Messages conform to a message protocol adopted by both the sender (to encode the message) & receiver (to interpret the message).
Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network.

Messages can serve many purposes:
• Send data to another computer
• Request data from another computer
• Instruct a program to call a function on some arguments
• Transfer a program to be executed by another computer

Messages conform to a message protocol adopted by both the sender (to encode the message) & receiver (to interpret the message).
• For example, bits at fixed positions may have fixed meanings.
Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network

Messages can serve many purposes:
• Send data to another computer
• Request data from another computer
• Instruct a program to call a function on some arguments
• Transfer a program to be executed by another computer

Messages conform to a message protocol adopted by both the sender (to encode the message) & receiver (to interpret the message)
• For example, bits at fixed positions may have fixed meanings
• Components of a message may be separated by delimiters
Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network

Messages can serve many purposes:

• Send data to another computer
• Request data from another computer
• Instruct a program to call a function on some arguments
• Transfer a program to be executed by another computer

Messages conform to a message protocol adopted by both the sender (to encode the message) & receiver (to interpret the message)

• For example, bits at fixed positions may have fixed meanings
• Components of a message may be separated by delimiters
• Protocols are designed to be implemented by many different programming languages on many different types of machines
Internet Protocol
The Internet Protocol
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks.
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks

- Networks are inherently unreliable at any point
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks

• Networks are inherently unreliable at any point
• The structure of a network is dynamic, not fixed
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks
• Networks are inherently unreliable at any point
• The structure of a network is dynamic, not fixed
• No system exists to monitor or track communications
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks

- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Octet</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Offsets</td>
<td>Octet</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Octet</td>
<td>Bit</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IPv4 Header Format

- Version
- IP Header Length
- Differentiated Services Code Point (DSCP)
- Explicit Congestion Notification (ECN)
- Total Length
- Identification
- Flags
- Fragment Offset
- Time To Live
- Protocol
- Header Checksum
- Source IP Address
- Destination IP Address
- Options (if IHL > 5)
The Internet Protocol (IP) specifies how to transfer packets of data among networks

- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

### IPv4 Header Format

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Octet</th>
<th>IPv4</th>
<th>IPv4 Header Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Version</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>IHL</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>64</td>
<td>DSCP</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>32</td>
<td>ECN</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>96</td>
<td>Total Length</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>128</td>
<td>Identification</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>160</td>
<td>Time To Live</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td></td>
<td>Source IP Address</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>256</td>
<td>Destination IP Address</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>28</td>
<td>Protocol</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>27</td>
<td>Header Checksum</td>
</tr>
<tr>
<td></td>
<td>44</td>
<td>26</td>
<td>Flags</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>25</td>
<td>Fragment Offset</td>
</tr>
<tr>
<td></td>
<td>52</td>
<td>24</td>
<td>Options (if IHL &gt; 5)</td>
</tr>
</tbody>
</table>

http://en.wikipedia.org/wiki/IPv4
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks:
- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

All machines know IPv4
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks

- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

All machines know IPv4

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Octet</th>
<th>IPv4</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet</td>
<td>Bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>96</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>160</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where to send the packet

http://en.wikipedia.org/wiki/IPv4
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks

- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

All machines know IPv4
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks

- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

All machines know IPv4

IPv4 Header Format

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Octet</th>
<th>IPv4</th>
<th>Octet</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>12345678910111213141516171819202122232425262728293031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Version</td>
<td>IHL</td>
<td>DSCP</td>
<td>ECN</td>
<td>Total Length</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>Identification</td>
<td>Flags</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>Time To Live</td>
<td>Protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>96</td>
<td>Source IP Address</td>
<td>Destination IP Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td>Options (if IHL &gt; 5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>160</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E.g., 192.168.1.1
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks

- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

All machines know IPv4

---

http://en.wikipedia.org/wiki/IPv4
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks
- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

All machines know IPv4

Max length: 216 = 65,536
E.g., 192.168.1.1

http://en.wikipedia.org/wiki/IPv4
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks

- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

---

**All machines know IPv4**

---

**Max length:**

\[ 2^{16} = 65,536 \]

E.g.,

192.168.1.1

---

**IPv4 Header Format**

- The packet knows its size
- Packets can't survive forever
- Where to send error reports
- Where to send the packet

---

http://en.wikipedia.org/wiki/IPv4
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks

- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Octet</th>
<th>IPv4 Header Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>Version IHL DSCP ECN</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>Identification Flags</td>
</tr>
<tr>
<td>12</td>
<td>96</td>
<td>Time To Live Protocol</td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td>Source IP Address Destination IP Address</td>
</tr>
<tr>
<td>20</td>
<td>160</td>
<td>Options (if IHL &gt; 5)</td>
</tr>
</tbody>
</table>

**IPv4**

- Max length: \(2^{16} = 65,536\)
- E.g., 192.168.1.1

- All machines know IPv4
- Decremented on forwarding

- Packets can't survive forever
- The packet knows its size
- Where to send error reports
- Where to send the packet

http://en.wikipedia.org/wiki/IPv4
The Internet Protocol

The Internet Protocol (IP) specifies how to transfer packets of data among networks

- Networks are inherently unreliable at any point
- The structure of a network is dynamic, not fixed
- No system exists to monitor or track communications

Packets are forwarded toward their destination on a best effort basis

Programs that use IP typically need a policy for handling lost packets

Max length: 216 = 65,536
E.g., 192.168.1.1
Transmission Control Protocol
Transmission Control Protocol
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:

- Packets are limited to 65,535 bytes each
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:

- Packets are limited to 65,535 bytes each
- Packets may arrive in a different order than they were sent
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:

- Packets are limited to 65,535 bytes each
- Packets may arrive in a different order than they were sent
- Packets may be duplicated or lost
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:

- Packets are limited to 65,535 bytes each
- Packets may arrive in a different order than they were sent
- Packets may be duplicated or lost

The Transmission Control Protocol (TCP) improves reliability:
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:

• Packets are limited to 65,535 bytes each
• Packets may arrive in a different order than they were sent
• Packets may be duplicated or lost

The Transmission Control Protocol (TCP) improves reliability:

• Ordered, reliable transmission of arbitrary byte streams
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:

• Packets are limited to 65,535 bytes each
• Packets may arrive in a different order than they were sent
• Packets may be duplicated or lost

The Transmission Control Protocol (TCP) improves reliability:

• Ordered, reliable transmission of arbitrary byte streams
• Implemented using the IP. Every TCP connection involves sending IP packets
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:
• Packets are limited to 65,535 bytes each
• Packets may arrive in a different order than they were sent
• Packets may be duplicated or lost

The Transmission Control Protocol (TCP) improves reliability:
• Ordered, reliable transmission of arbitrary byte streams
• Implemented using the IP. Every TCP connection involves sending IP packets
• Each packet in a TCP session has a sequence number:
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:
- Packets are limited to 65,535 bytes each
- Packets may arrive in a different order than they were sent
- Packets may be duplicated or lost

The Transmission Control Protocol (TCP) improves reliability:
- Ordered, reliable transmission of arbitrary byte streams
- Implemented using the IP. Every TCP connection involves sending IP packets
- Each packet in a TCP session has a sequence number:
  - The receiver can correctly order packets that arrive out of order
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:
• Packets are limited to 65,535 bytes each
• Packets may arrive in a different order than they were sent
• Packets may be duplicated or lost

The Transmission Control Protocol (TCP) improves reliability:
• Ordered, reliable transmission of arbitrary byte streams
• Implemented using the IP. Every TCP connection involves sending IP packets
• Each packet in a TCP session has a sequence number:
  • The receiver can correctly order packets that arrive out of order
  • The receiver can ignore duplicate packets
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:

- Packets are limited to 65,535 bytes each
- Packets may arrive in a different order than they were sent
- Packets may be duplicated or lost

The Transmission Control Protocol (TCP) improves reliability:

- Ordered, reliable transmission of arbitrary byte streams
- Implemented using the IP. Every TCP connection involves sending IP packets
- Each packet in a TCP session has a sequence number:
  - The receiver can correctly order packets that arrive out of order
  - The receiver can ignore duplicate packets
- All received packets are acknowledged; both parties know that transmission succeeded
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:

• Packets are limited to 65,535 bytes each
• Packets may arrive in a different order than they were sent
• Packets may be duplicated or lost

The Transmission Control Protocol (TCP) improves reliability:

• Ordered, reliable transmission of arbitrary byte streams
• Implemented using the IP. Every TCP connection involves sending IP packets
• Each packet in a TCP session has a sequence number:
  ▪ The receiver can correctly order packets that arrive out of order
  ▪ The receiver can ignore duplicate packets
• All received packets are acknowledged; both parties know that transmission succeeded
• Packets that aren't acknowledged are sent repeatedly
Transmission Control Protocol

The design of the Internet Protocol (IPv4) imposes constraints:

• Packets are limited to 65,535 bytes each
• Packets may arrive in a different order than they were sent
• Packets may be duplicated or lost

The Transmission Control Protocol (TCP) improves reliability:

• Ordered, reliable transmission of arbitrary byte streams
• Implemented using the IP. Every TCP connection involves sending IP packets
• Each packet in a TCP session has a sequence number:
  ▪ The receiver can correctly order packets that arrive out of order
  ▪ The receiver can ignore duplicate packets
• All received packets are acknowledged; both parties know that transmission succeeded
• Packets that aren't acknowledged are sent repeatedly

The socket module in Python implements the TCP
TCP Handshakes
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible.
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible.

"Can you hear me now?"  Let's design a handshake protocol.
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible

"Can you hear me now?"  

Let's design a handshake protocol

Handshake Goals:
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible.

"Can you hear me now?" Let's design a handshake protocol

Handshake Goals:

• Computer A knows that it can send data to and receive data from Computer B
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible.

"Can you hear me now?"  Let's design a handshake protocol

Handshake Goals:

• Computer A knows that it can send data to and receive data from Computer B
• Computer B knows that it can send data to and receive data from Computer A
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible.

"Can you hear me now?"  Let's design a handshake protocol

Handshake Goals:

• Computer A knows that it can send data to and receive data from Computer B
• Computer B knows that it can send data to and receive data from Computer A
• Lots of separate connections can exist without any confusion
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible.

"Can you hear me now?" *Let's design a handshake protocol*

Handshake Goals:
• Computer A knows that it can send data to and receive data from Computer B
• Computer B knows that it can send data to and receive data from Computer A
• Lots of separate connections can exist without any confusion
• The number of required messages is minimized
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible

"Can you hear me now?"  

Let's design a handshake protocol

Handshake Goals:

• Computer A knows that it can send data to and receive data from Computer B
• Computer B knows that it can send data to and receive data from Computer A
• Lots of separate connections can exist without any confusion
• The number of required messages is minimized

Communication Rules:
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible

"Can you hear me now?" Let's design a handshake protocol

Handshake Goals:
• Computer A knows that it can send data to and receive data from Computer B
• Computer B knows that it can send data to and receive data from Computer A
• Lots of separate connections can exist without any confusion
• The number of required messages is minimized

Communication Rules:
• Computer A can send an initial message to Computer B requesting a new connection
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible.

"Can you hear me now?"  

Let's design a handshake protocol

Handshake Goals:

• Computer A knows that it can send data to and receive data from Computer B
• Computer B knows that it can send data to and receive data from Computer A
• Lots of separate connections can exist without any confusion
• The number of required messages is minimized

Communication Rules:

• Computer A can send an initial message to Computer B requesting a new connection
• Computer B can respond to messages from Computer A
TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible.

"Can you hear me now?"  Let's design a handshake protocol

Handshake Goals:
- Computer A knows that it can send data to and receive data from Computer B
- Computer B knows that it can send data to and receive data from Computer A
- Lots of separate connections can exist without any confusion
- The number of required messages is minimized

Communication Rules:
- Computer A can send an initial message to Computer B requesting a new connection
- Computer B can respond to messages from Computer A
- Computer A can respond to messages from Computer B
Message Sequence of a TCP Connection
Message Sequence of a TCP Connection

Computer A
Message Sequence of a TCP Connection
Message Sequence of a TCP Connection

Computer A

Synchronization request

Computer B
Message Sequence of a TCP Connection

Computer A

Synchronization request

Acknowledgement & synchronization request

Computer B
Message Sequence of a TCP Connection

Synchronization request

Acknowledgement & synchronization request

Acknowledgement
Message Sequence of a TCP Connection

Computer A

Synchronization request

Acknowledgement & synchronization request

Acknowledgement

Computer B
Message Sequence of a TCP Connection

Computer A

Establishes packet numbering system

Synchronization request

Acknowledgement & synchronization request

Acknowledgement

Computer B
Message Sequence of a TCP Connection

Computer A

Establishes packet numbering system

Computer B

Synchronization request

 Acknowledgement 

& synchronization request

 Acknowledgement

 Data message from A to B

 Acknowledgement

 Data message from B to A

 Acknowledgement

..
Message Sequence of a TCP Connection

1. Synchronization request
2. Acknowledgement & synchronization request
3. Acknowledgement
4. Data message from A to B
5. Acknowledgement
6. Data message from B to A
7. Acknowledgement
8. Termination signal
Message Sequence of a TCP Connection

Establishes packet numbering system

Synchronization request

Acknowledgement & synchronization request

Acknowledgement

Data message from A to B

Acknowledgement

Data message from B to A

Acknowledgement

Termination signal

Acknowledgement & termination signal
Message Sequence of a TCP Connection

Computer A  Establishes packet numbering system  Computer B

Synchronization request

Acknowledgement & synchronization request

Acknowledgement

Data message from A to B

Acknowledgement

Data message from B to A

Acknowledgement

Termination signal

Acknowledgement & termination signal

Acknowledgement
Message Sequence of a TCP Connection

Computer A

Establishes packet numbering system

Computer B

Synchronization request

Acknowledgement & synchronization request

Acknowledgement

Data message from A to B

Acknowledgement

Data message from B to A

Acknowledgement

Termination signal

Acknowledgement & termination signal

Acknowledgement
Client/Server Architecture
The Client/Server Architecture
The Client/Server Architecture

One server provides information to multiple clients through request and response messages
The Client/Server Architecture

One server provides information to multiple clients through request and response messages

**Server role:** Respond to service requests with requested information
The Client/Server Architecture

One server provides information to multiple clients through request and response messages

**Server role:** Respond to service requests with requested information

**Client role:** Request information and make use of the response
**The Client/Server Architecture**

One server provides information to multiple clients through request and response messages.

**Server role:** Respond to service requests with requested information.

**Client role:** Request information and make use of the response.

**Abstraction:** The client knows what service a server provides, but not how it is provided.
The Client/Server Architecture

One server provides information to multiple clients through request and response messages

**Server role**: Respond to service requests with requested information

**Client role**: Request information and make use of the response

**Abstraction**: The client knows what service a server provides, but not how it is provided
Client/Server Example: The World Wide Web
Client/Server Example: The World Wide Web

The client is a web browser (e.g., Firefox):
Client/Server Example: The World Wide Web

The client is a web browser (e.g., Firefox):
* Request content for a location
Client/Server Example: The World Wide Web

The client is a web browser (e.g., Firefox):
• Request content for a location
• Interpret the content for the user
Client/Server Example: The World Wide Web

The client is a web browser (e.g., Firefox):

- Request content for a location
- Interpret the content for the user

The server is a web server:
Client/Server Example: The World Wide Web

The client is a web browser (e.g., Firefox):
• Request content for a location
• Interpret the content for the user

The server is a web server:
• Interpret requests and respond with content
Client/Server Example: The World Wide Web

The client is a web browser (e.g., Firefox):
• Request content for a location
• Interpret the content for the user

The server is a web server:
• Interpret requests and respond with content
Client/Server Example: The World Wide Web

The client is a web browser (e.g., Firefox):
• Request content for a location
• Interpret the content for the user

The server is a web server:
• Interpret requests and respond with content
Client/Server Example: The World Wide Web

The client is a web browser (e.g., Firefox):
- Request content for a location
- Interpret the content for the user

The server is a web server:
- Interpret requests and respond with content
Client/Server Example: The World Wide Web

The client is a web browser (e.g., Firefox):
• Request content for a location
• Interpret the content for the user

The server is a web server:
• Interpret requests and respond with content
The Hypertext Transfer Protocol
The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture.
The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture

The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture.

Uniform resource locator (URL)

The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture

Browser issues a GET request to a server at www.nytimes.com for the content (resource) at location "pages/todayspaper"
The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture.

Browser issues a GET request to a server at www.nytimes.com for the content (resource) at location "pages/todayspaper"

Server response contains more than just the resource itself:
The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture.

Browser issues a GET request to a server at www.nytimes.com for the content (resource) at location "pages/todayspaper"

Server response contains more than just the resource itself:
- Status code, e.g. 200 OK, 404 Not Found, 403 Forbidden, etc.
The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture.

Browser issues a GET request to a server at www.nytimes.com for the content (resource) at location "pages/todayspaper"

Server response contains more than just the resource itself:
• Status code, e.g. 200 OK, 404 Not Found, 403 Forbidden, etc.
• Date of response; type of server responding
The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture. Browser issues a GET request to a server at www.nytimes.com for the content (resource) at location "pages/todayspaper".

Server response contains more than just the resource itself:
• Status code, e.g. 200 OK, 404 Not Found, 403 Forbidden, etc.
• Date of response; type of server responding
• Last-modified time of the resource
The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture.

Browser issues a GET request to a server at www.nytimes.com for the content (resource) at location "pages/todayspaper"

Server response contains more than just the resource itself:
- Status code, e.g. 200 OK, 404 Not Found, 403 Forbidden, etc.
- Date of response; type of server responding
- Last-modified time of the resource
- Type of content and length of content
Properties of a Client/Server Architecture
Properties of a Client/Server Architecture

Benefits:
Properties of a Client/Server Architecture

Benefits:
- Creates a separation of concerns among components
Properties of a Client/Server Architecture

Benefits:

• Creates a separation of concerns among components
• Enforces an abstraction barrier between client and server
Properties of a Client/Server Architecture

Benefits:
• Creates a separation of concerns among components
• Enforces an abstraction barrier between client and server
• A centralized server can reuse computation across clients
Properties of a Client/Server Architecture

Benefits:

• Creates a separation of concerns among components
• Enforces an abstraction barrier between client and server
• A centralized server can reuse computation across clients

Liabilities:
Properties of a Client/Server Architecture

Benefits:
• Creates a separation of concerns among components
• Enforces an abstraction barrier between client and server
• A centralized server can reuse computation across clients

Liabilities:
• A single point of failure: the server
Properties of a Client/Server Architecture

Benefits:
• Creates a separation of concerns among components
• Enforces an abstraction barrier between client and server
• A centralized server can reuse computation across clients

Liabilities:
• A single point of failure: the server
• Computing resources become scarce when demand increases
Properties of a Client/Server Architecture

Benefits:
• Creates a separation of concerns among components
• Enforces an abstraction barrier between client and server
• A centralized server can reuse computation across clients

Liabilities:
• A single point of failure: the server
• Computing resources become scarce when demand increases

Common use cases:
Properties of a Client/Server Architecture

Benefits:
• Creates a separation of concerns among components
• Enforces an abstraction barrier between client and server
• A centralized server can reuse computation across clients

Liabilities:
• A single point of failure: the server
• Computing resources become scarce when demand increases

Common use cases:
• Databases – The database serves responses to query requests
Properties of a Client/Server Architecture

Benefits:
• Creates a separation of concerns among components
• Enforces an abstraction barrier between client and server
• A centralized server can reuse computation across clients

Liabilities:
• A single point of failure: the server
• Computing resources become scarce when demand increases

Common use cases:
• Databases — The database serves responses to query requests
• Open Graphics Library (OpenGL) — A graphics processing unit (GPU) serves images to a central processing unit (CPU)
Properties of a Client/Server Architecture

Benefits:
• Creates a separation of concerns among components
• Enforces an abstraction barrier between client and server
• A centralized server can reuse computation across clients

Liabilities:
• A single point of failure: the server
• Computing resources become scarce when demand increases

Common use cases:
• Databases – The database serves responses to query requests
• Open Graphics Library (OpenGL) – A graphics processing unit (GPU) serves images to a central processing unit (CPU)
• Internet file and resource transfer: HTTP, FTP, email, etc.
Peer-to-Peer Architecture
The Peer-to-Peer Architecture
The Peer-to-Peer Architecture

All participants in a distributed application contribute computational resources: processing, storage, and network capacity
The Peer-to-Peer Architecture

All participants in a distributed application contribute computational resources: processing, storage, and network capacity.

Messages are relayed through a network of participants.
The Peer-to-Peer Architecture

All participants in a distributed application contribute computational resources: processing, storage, and network capacity.

Messages are relayed through a network of participants.

Each participant has only partial knowledge of the network.
The Peer-to-Peer Architecture

All participants in a distributed application contribute computational resources: processing, storage, and network capacity.

Messages are relayed through a network of participants.

Each participant has only partial knowledge of the network.
Network Structure Concerns

Network Structure Concerns

Some data transfers on the Internet are faster than others
Network Structure Concerns

Some data transfers on the Internet are faster than others.

The time required to transfer a message through a peer-to-peer network depends on the route chosen.

![Peer-to-Peer Network Diagram](http://en.wikipedia.org/wiki/File:P2P-network.svg)
Network Structure Concerns

Some data transfers on the Internet are faster than others.

The time required to transfer a message through a peer-to-peer network depends on the route chosen.
Network Structure Concerns

Some data transfers on the Internet are faster than others

The time required to transfer a message through a peer-to-peer network depends on the route chosen
Network Structure Concerns

Some data transfers on the Internet are faster than others.

The time required to transfer a message through a peer-to-peer network depends on the route chosen.
Network Structure Concerns

Some data transfers on the Internet are faster than others

The time required to transfer a message through a peer-to-peer network depends on the route chosen

Network Structure Concerns

Some data transfers on the Internet are faster than others.

The time required to transfer a message through a peer-to-peer network depends on the route chosen.
Example: Skype
Example: Skype

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture
Example: Skype

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.
**Example: Skype**

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.

Conversations between two computers that cannot send messages to each other directly are relayed through supernodes.
Example: Skype

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.

Conversations between two computers that cannot send messages to each other directly are relayed through supernodes.

Any Skype client with its own IP address may be a supernode.
Example: Skype

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.

Conversations between two computers that cannot send messages to each other directly are relayed through supernodes.

Any Skype client with its own IP address may be a supernode.
Example: Skype

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.

Conversations between two computers that cannot send messages to each other directly are relayed through supernodes.

Any Skype client with its own IP address may be a supernode.
**Example: Skype**

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.

Conversations between two computers that cannot send messages to each other directly are relayed through supernodes.

Any Skype client with its own IP address may be a supernode.

Clients behind firewalls cannot communicate directly.
Example: Skype

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.

Conversations between two computers that cannot send messages to each other directly are relayed through supernodes.

Any Skype client with its own IP address may be a supernode.
**Example: Skype**

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.

Conversations between two computers that cannot send messages to each other directly are relayed through supernodes.

Any Skype client with its own IP address may be a supernode.

---

**Diagram:**
- **Clients** behind firewalls cannot communicate directly.
- **Client A** and **Client B** are connected indirectly through a supernode, **Client C**.
- **Client C** is a client not behind a firewall and can be used as a supernode.
Example: Skype

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.

Conversations between two computers that cannot send messages to each other directly are relayed through supernodes.

Any Skype client with its own IP address may be a supernode.

Clients behind firewalls cannot communicate directly.

A client not behind a firewall may be used as a supernode.
Example: Skype

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.

Conversations between two computers that cannot send messages to each other directly are relayed through supernodes.

Any Skype client with its own IP address may be a supernode.

![Diagram of Skype network](attachment:image.png)