Distributed Computing
Distributed Computing

A distributed computing application consists of multiple programs running on multiple computers that together coordinate to perform some task.
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.
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.
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 spread across multiple machines (Wednesday).
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 & the receiver to interpret it.
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 & the receiver to interpret it.
- 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 & the receiver to interpret it.
- 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 & the receiver to interpret it.

- 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 a variety of platforms.
The Internet Protocol
The Internet Protocol

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

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

- Networks are inherently unreliable at any point.
The Internet Protocol

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

- Networks are inherently unreliable at any point.
- The structure of a network is dynamic.
The Internet Protocol

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

• Networks are inherently unreliable at any point.
• The structure of a network is dynamic.
• No system exists to monitor or track communications.
The Internet Protocol

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

- Networks are inherently unreliable at any point.
- The structure of a network is dynamic.
- No system exists to monitor or track communications.

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

IPv4 Header Format

<table>
<thead>
<tr>
<th>Offsets</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>Version</td>
<td>DSCP</td>
<td>ECN</td>
<td>Total Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Identification</td>
<td>Flags</td>
<td>Fragment Offset</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time To Live</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Source IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Destination IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Options (if IHL &gt; 5)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Internet Protocol

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

- Networks are inherently unreliable at any point.
- The structure of a network is dynamic.
- 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></td>
<td>0</td>
<td>Version</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Identification</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Time To Live</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>Source IP Address</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>Destination IP Address</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>Options (if IHL &gt; 5)</td>
</tr>
</tbody>
</table>

http://en.wikipedia.org/wiki/IPv4
The Internet Protocol (IP) specifies how to transfer packets of data among different networks.

- Networks are inherently unreliable at any point.
- The structure of a network is dynamic.
- No system exists to monitor or track communications.
The Internet Protocol (IP) specifies how to transfer *packets* of data among different networks.

- Networks are inherently unreliable at any point.
- The structure of a network is dynamic.
- No system exists to monitor or track communications.

<table>
<thead>
<tr>
<th>Offsets (Octet)</th>
<th>Octet</th>
<th>IPv4</th>
<th>IPv4 Header Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>IPv4</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td></td>
<td>Identification</td>
</tr>
<tr>
<td>12</td>
<td>96</td>
<td></td>
<td>Time To Live</td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td></td>
<td>Source IP Address</td>
</tr>
<tr>
<td>20</td>
<td>160</td>
<td></td>
<td>Destination IP Address</td>
</tr>
<tr>
<td>24</td>
<td>256</td>
<td></td>
<td>Options (if IHL &gt; 5)</td>
</tr>
</tbody>
</table>

*Where to send the packet*

*Where to send error reports*
The Internet Protocol (IP) specifies how to transfer *packets* of data among different networks.

- Networks are inherently unreliable at any point.
- The structure of a network is dynamic.
- 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></td>
<td>IPv4</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Where to send error reports</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>Identification</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>Time To Live</td>
</tr>
<tr>
<td>12</td>
<td>96</td>
<td>Source IP Address</td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td>Destination IP Address</td>
</tr>
<tr>
<td>20</td>
<td>160</td>
<td>Options (if IHL &gt; 5)</td>
</tr>
</tbody>
</table>

The packet knows its size.

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

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

- Networks are inherently unreliable at any point.
- The structure of a network is dynamic.
- No system exists to monitor or track communications.

| Octet | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|-------|-----|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0     | 0   |   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 4     | 32  |   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 8     | 64  |   |   |   |   |   |   |   |   |   |   |    |    | Identification |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 12    | 96  |   |   |   |   |   |   |   |   |   |   |    |    | Protocol |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 128 |   |   |   |   |   |   |   |   |   |   |    |    | Source IP Address |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 20    | 160 |   |   |   |   |   |   |   |   |   |   |    |    | Destination IP Address |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

IPv4

The packet knows its size

Where to send error reports

Packets can't survive forever

Where to send the packet

IPv4 Header Format

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

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

- Networks are inherently unreliable at any point.
- The structure of a network is dynamic.
- No system exists to monitor or track communications.

Packets are forwarded toward their destination using simple rules on a best-effort basis.
Transmission Control Protocol
Transmission Control Protocol

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

The design of the Internet Protocol (IP) imposes constraints:
• Packets are limited to 65,535 bytes each.
Transmission Control Protocol

The design of the Internet Protocol (IP) 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 (IP) 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** (IP) 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** (IP) 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** (IP) 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.
Transmission Control Protocol

The design of the Internet Protocol (IP) 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.
• Correctly orders packets by including sequence numbers.
Transmission Control Protocol

The design of the Internet Protocol (IP) 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.
• Correctly orders packets by including sequence numbers.
• Removes duplicates; requests retransmission of lost packets.
Transmission Control Protocol

The design of the Internet Protocol (IP) 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.
• Correctly orders packets by including sequence numbers.
• Removes duplicates; requests retransmission of lost packets.

TCP connection initiates with a "handshake" procedure.
Transmission Control Protocol

The design of the **Internet Protocol** (IP) 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.
- Correctly orders packets by including sequence numbers.
- Removes duplicates; requests retransmission of lost packets.

TCP connection initiates with a "handshake" procedure.
- What's the minimum number of messages needed to prove to both computers that two-way communication is possible?

5
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

Computer B

Acknowledgement & synchronization request
Message Sequence of a TCP Connection

Computer A

Synchronization request

Acknowledgement & synchronization request

Acknowledgement

Computer B
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

Synchronization request

Acknowledgement & synchronization request

Acknowledgement

Data message from A to B

Data message from B to A

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

Data message from B to A

Termination signal
Message Sequence of a TCP Connection

Establishes packet numbering system

Synchronization request

Acknowledgement & synchronization request

Acknowledgement

Data message from A to B

Data message from B to A

Termination signal

Acknowledgement & termination signal
Message Sequence of a TCP Connection

Establishes packet numbering system

Synchronization request

Acknowledgement & synchronization request

Acknowledgement

Data message from A to B

Data message from B to A

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 ...

... Data message from B to A ...

Termination signal

Acknowledgement & termination signal

Acknowledgement
Client/Server Architecture
Client/Server Architecture

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

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

One server provides information to multiple clients through \textit{request} and \textit{response} messages.

\textbf{Server role}: Respond to service requests with requested information.

\textbf{Client role}: Request information and make use of the response.
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 from a location on behalf of the user.
Client/Server Example: The World Wide Web

The **client** is a web browser (e.g., Firefox):
- Request content from a location on behalf of the user.
- Display the content to the user.
Client/Server Example: The World Wide Web

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

- Request content from a location on behalf of the user.
- Display the content to the user.

The **server** is a web server (e.g., www.nytimes.com)
Client/Server Example: The World Wide Web

The **client** is a web browser (e.g., Firefox):
- Request content from a location on behalf of the user.
- Display the content to the user.

The **server** is a web server (e.g., [www.nytimes.com](http://www.nytimes.com))
- Respond with (perhaps personalized) content at that location.
Client/Server Example: The World Wide Web

The **client** is a web browser (e.g., Firefox):
- Request content from a location on behalf of the user.
- Display the content to the user.

The **server** is a web server (e.g., www.nytimes.com)
- Respond with (perhaps personalized) content at that location.
Client/Server Example: The World Wide Web

The **client** is a web browser (e.g., Firefox):
- Request content from a location on behalf of the user.
- Display the content to the user.

The **server** is a web server (e.g., [www.nytimes.com](http://www.nytimes.com))
- Respond with (perhaps personalized) content at that location.

![Diagram showing TCP Initialization Handshake and HTTP GET request of content between a web browser and a web server.](image)
Client/Server Example: The World Wide Web

The **client** is a web browser (e.g., Firefox):
- Request content from a location on behalf of the user.
- Display the content to the user.

The **server** is a web server (e.g., www.nytimes.com)
- Respond with (perhaps personalized) content at that location.

---

TCP Initialization Handshake

HTTP GET request of content

HTTP response with content
Client/Server Example: The World Wide Web

The **client** is a web browser (e.g., Firefox):
- Request content from a location on behalf of the user.
- Display the content to the user.

The **server** is a web server (e.g., www.nytimes.com)
- Respond with (perhaps personalized) content at that location.
Client/Server Example: The World Wide Web

The **client** is a web browser (e.g., Firefox):
- Request content from a location on behalf of the user.
- Display the content to the user.

The **server** is a web server (e.g., www.nytimes.com)
- Respond with (perhaps personalized) content at that location.

![Diagram of client/server interaction]

- **TCP Initialization Handshake**
- **HTTP GET request of content**
- **HTTP response with content**
- **Follow-up requests for auxiliary content**

Demo
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 www.nytimes.com for the content (resource) at location "pages/todayspaper".

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 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 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 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

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

Browser issues a GET request to 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 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
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 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

Demo
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 with increasing demand.
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 with increasing demand.

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 with increasing demand.

**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 with increasing demand.

**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 with increasing demand.

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).
• File and resource transfer: HTTP, FTP, email, etc.
Peer-to-Peer Architecture
Peer-to-Peer Architecture

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

All participants in a distributed application contribute computational resources: processing, storage, and network. Messages are relayed through a network of participants.
Peer-to-Peer Architecture

All participants in a distributed application contribute computational resources: processing, storage, and network. Messages are relayed through a network of participants. Each participant has only partial knowledge of the network.
Peer-to-Peer Architecture

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

Client A
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.

Client A  Client B
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.

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.
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

Client A

Client B

Client C