61A Lecture 35

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

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

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

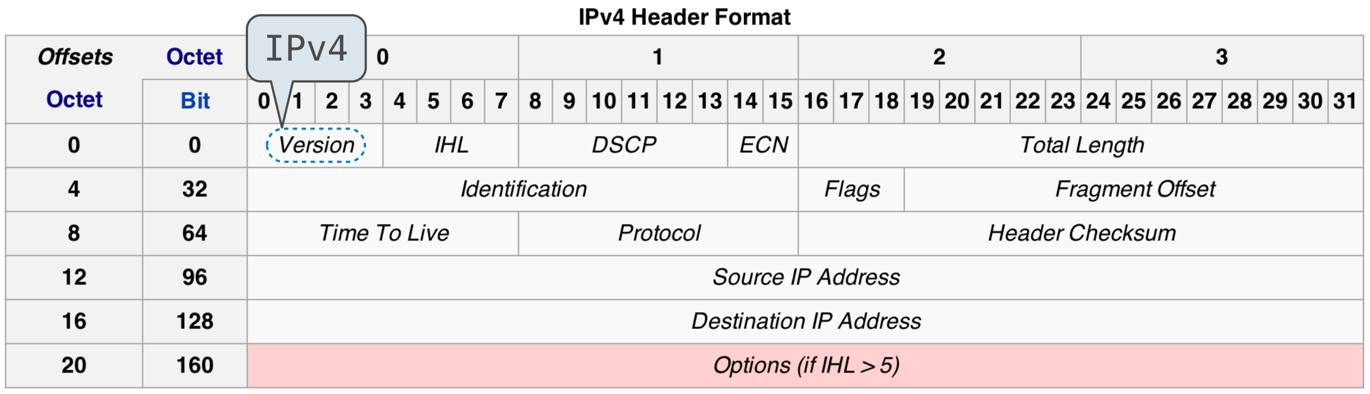
IPv4 Header Format

Offsets	Octet	0								1								2								3							
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	*Version*				*IHL*				*DSCP*						*ECN*		*Total Length*															
4	32	*Identification*																*Flags*			*Fragment Offset*												
8	64	*Time To Live*								*Protocol*								*Header Checksum*															
12	96	*Source IP Address*																															
16	128	*Destination IP Address*																															
20	160	*Options (if IHL > 5)*																															

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



IPv4 Header Format

Offsets	Octet	0								1								2								3							
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	*Version*				*IHL*				*DSCP*						*ECN*		*Total Length*															
4	32	*Identification*																*Flags*			*Fragment Offset*												
8	64	*Time To Live*								*Protocol*								*Header Checksum*															
12	96	*Source IP Address*																															
16	128	*Destination IP Address*																															
20	160	*Options (if IHL > 5)*																															

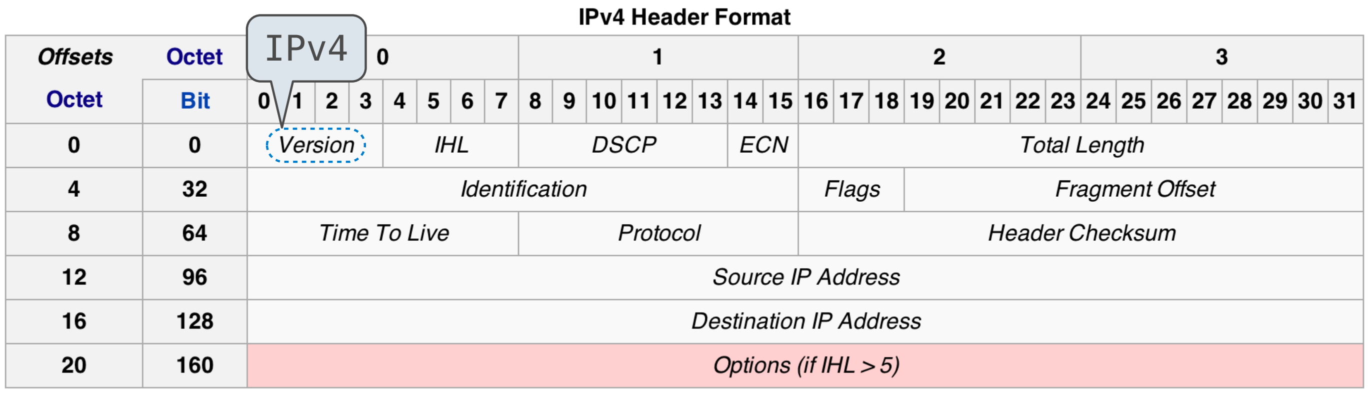
http://en.wikipedia.org/wiki/IPv4

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



IPv4 Header Format

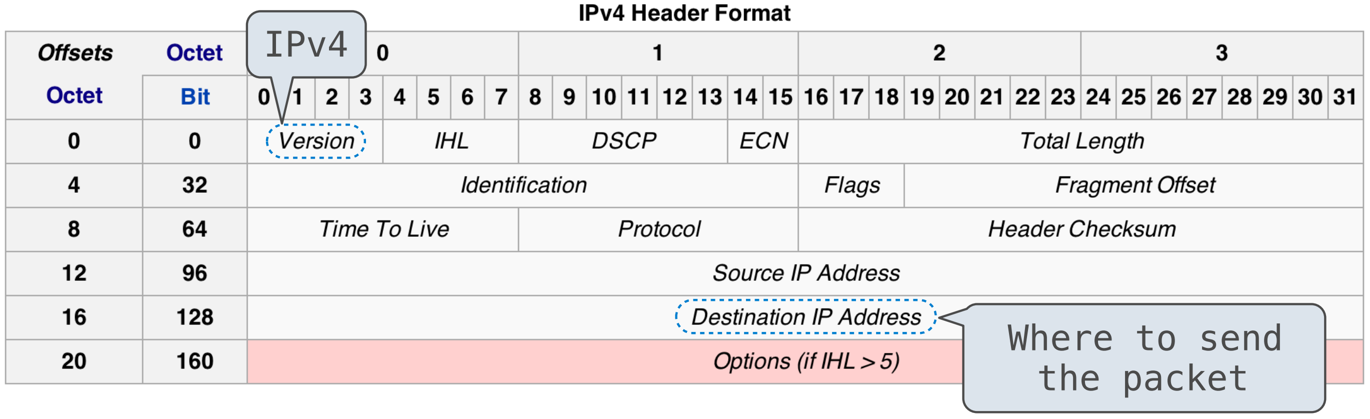
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Version*				*IHL*				*DSCP*						*ECN*		*Total Length*															
4	32	*Identification*																*Flags*			*Fragment Offset*												
8	64	*Time To Live*								*Protocol*								*Header Checksum*															
12	96	*Source IP Address*																															
16	128	*Destination IP Address*																															
20	160	*Options (if IHL > 5)*																															

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
Where to send the packet

IPv4 Header Format

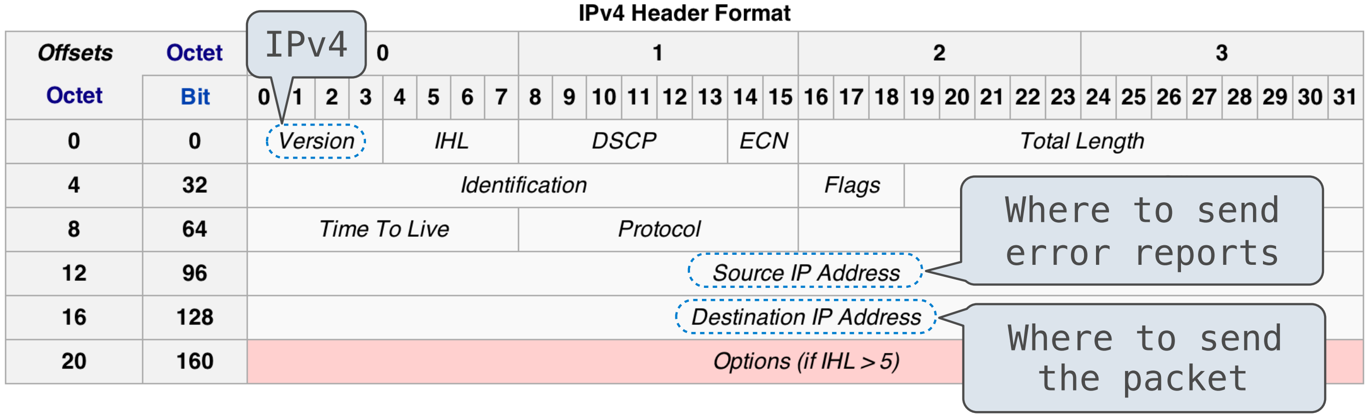
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset												
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

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
Where to send error reports
Where to send the packet

IPv4 Header Format

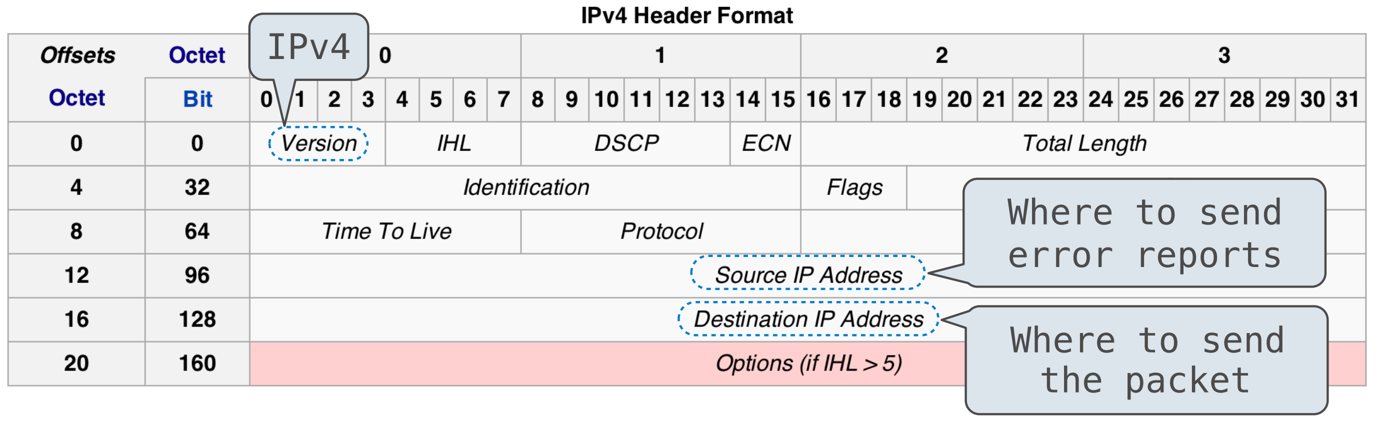
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Version*				*IHL*				*DSCP*						*ECN*		*Total Length*															
4	32	*Identification*																*Flags*															
8	64	*Time To Live*								*Protocol*																							
12	96	*Source IP Address*																															
16	128	*Destination IP Address*																															
20	160	*Options (if IHL > 5)*																															

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
Where to send error reports
Where to send the packet

IPv4 Header Format

Offsets	Octet	0								1								2								3							
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	*Version*				*IHL*				*DSCP*						*ECN*		*Total Length*															
4	32	*Identification*																*Flags*															
8	64	*Time To Live*								*Protocol*																							
12	96	*Source IP Address*																															
16	128	*Destination IP Address*																															
20	160	*Options (if IHL > 5)*																															

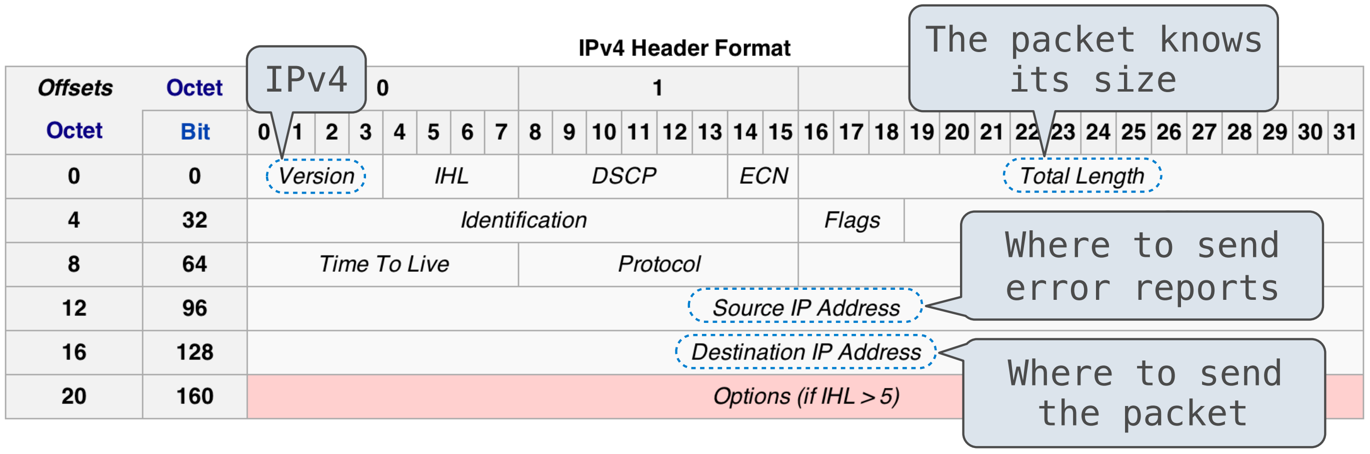
E.g., 192.168.1.1

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



IPv4 Header Format

Offsets	Octet	0								1																							
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	*Version*				*IHL*				*DSCP*						*ECN*		*Total Length*															
4	32	*Identification*																*Flags*															
8	64	*Time To Live*								*Protocol*																							
12	96	*Source IP Address*																															
16	128	*Destination IP Address*																															
20	160	*Options (if IHL > 5)*																															

IPv4
The packet knows its size
Where to send error reports
Where to send the packet

E.g., 192.168.1.1

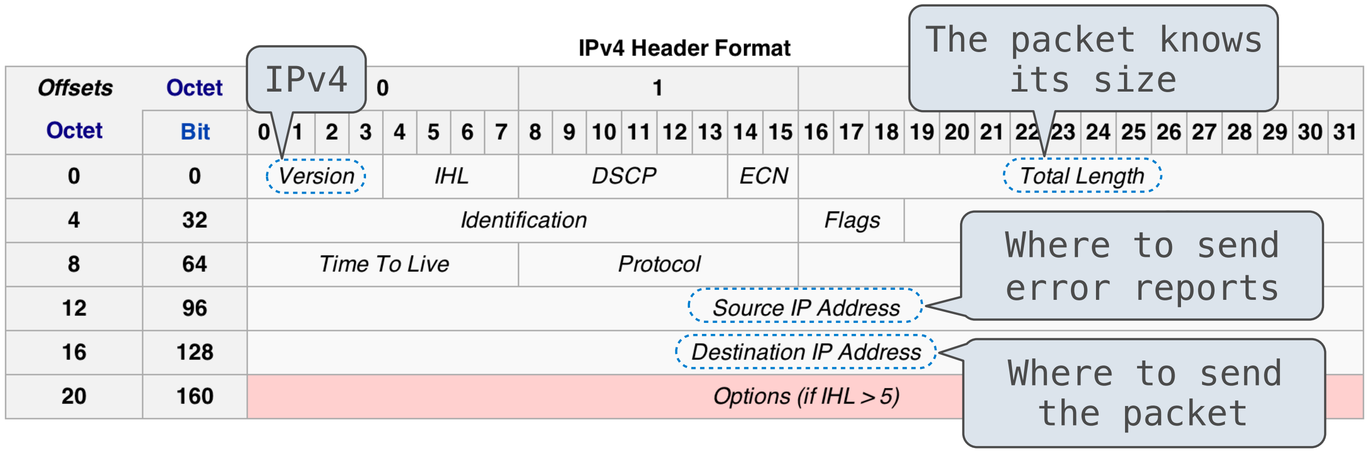
http://en.wikipedia.org/wiki/IPv4

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


IPv4
The packet knows its size
Where to send error reports
Where to send the packet

IPv4 Header Format

Offsets	Octet	0								1																							
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	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags															
8	64	Time To Live								Protocol																							
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

Max length: 216 = 65,536

E.g., 192.168.1.1

http://en.wikipedia.org/wiki/IPv4

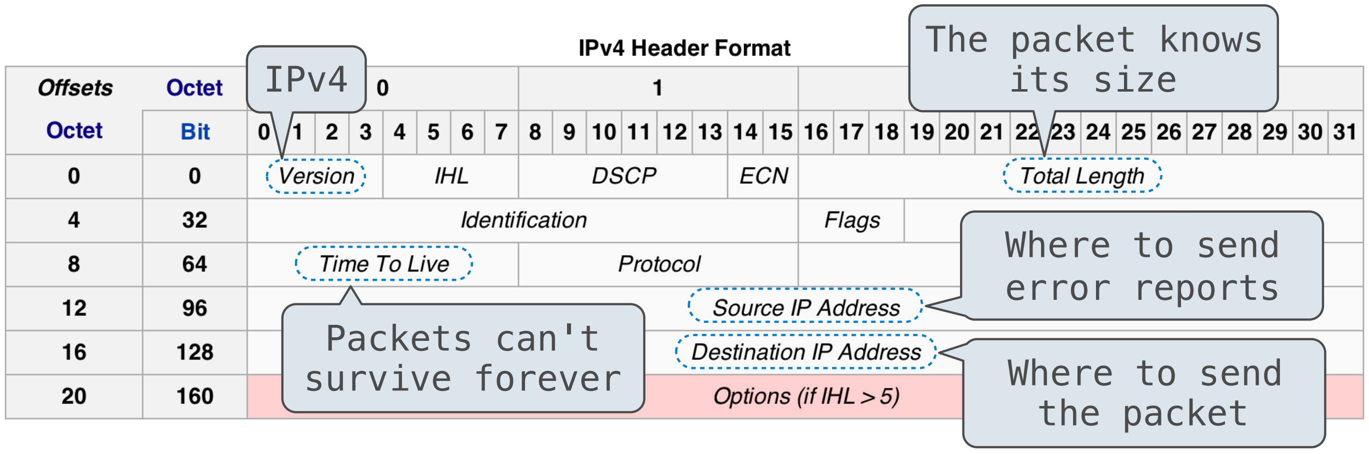
The Internet Protocol

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

IPv4 Header Format



IPv4
The packet knows its size
Where to send error reports
Packets can't survive forever
Where to send the packet

Offsets	Octet	0								1																							
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	*Version*				*IHL*				*DSCP*						*ECN*		*Total Length*															
4	32	*Identification*																*Flags*															
8	64	*Time To Live*								*Protocol*																							
12	96	*Source IP Address*																															
16	128	*Destination IP Address*																															
20	160	*Options (if IHL > 5)*																															

Max length: 216 = 65,536

E.g., 192.168.1.1

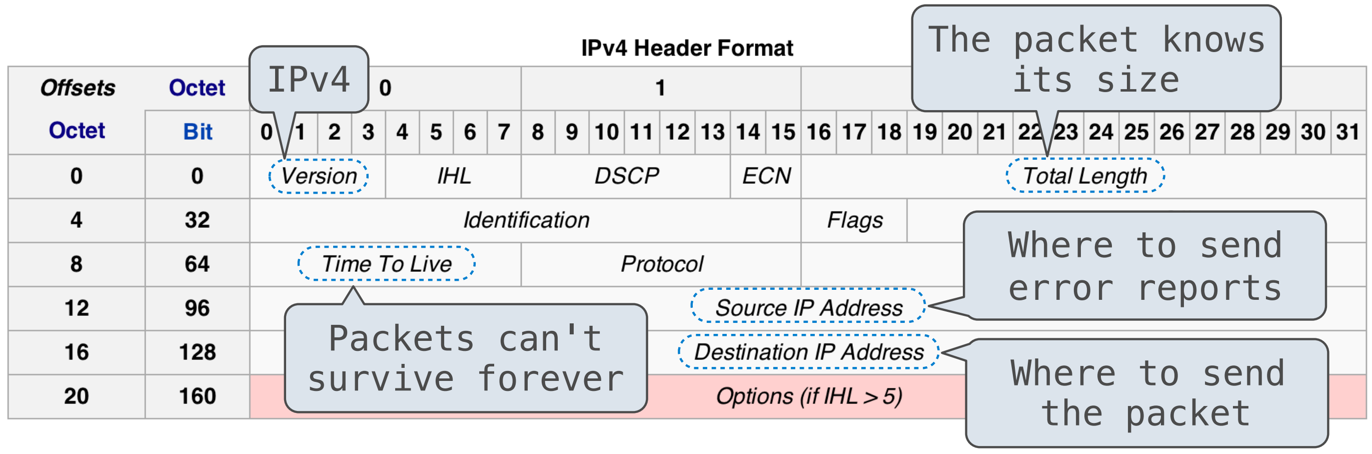
The Internet Protocol

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

Decremented on forwarding


IPv4
The packet knows its size
Where to send error reports
Packets can't survive forever
Where to send the packet

IPv4 Header Format

Offsets	Octet	0				1		2	3
Octet	Bit	0–3	4–7	8–13	14–15			16–23	24–31
0	0	*Version*	*IHL*	*DSCP*	*ECN*			*Total Length*	
4	32	*Identification*						*Flags*	
8	64	*Time To Live*		*Protocol*					
12	96	*Source IP Address*							
16	128	*Destination IP Address*							
20	160	*Options (if IHL > 5)*							

Max length: 216 = 65,536

E.g., 192.168.1.1

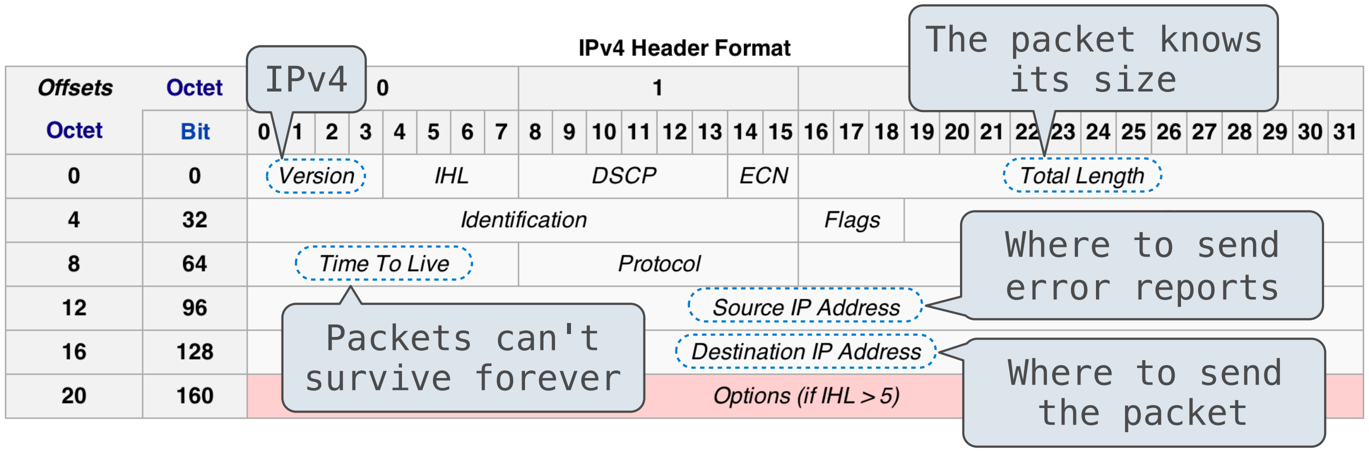
The Internet Protocol

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

Decremented on forwarding


IPv4
The packet knows its size
Packets can't survive forever
Where to send error reports
Where to send the packet

IPv4 Header Format

Offsets	Octet	0	1		
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	Version · IHL	DSCP · ECN	Total Length	
4	32	Identification		Flags	
8	64	Time To Live	Protocol		
12	96	Source IP Address			
16	128	Destination IP Address			
20	160	Options (if IHL > 5)			

Max length: 216 = 65,536

E.g., 192.168.1.1

Packets are forwarded toward their destination on a best effort basis

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

http://en.wikipedia.org/wiki/IPv4

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:

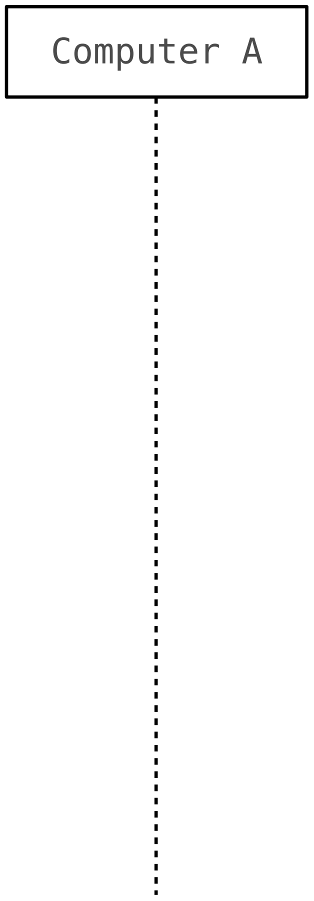
- Computer A knows that it can send data to and receive data from Computer B
- Computer B knows that it can send data to and receive data from Computer A
- Lots of separate connections can exist without any confusion
- The number of required messages is minimized

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

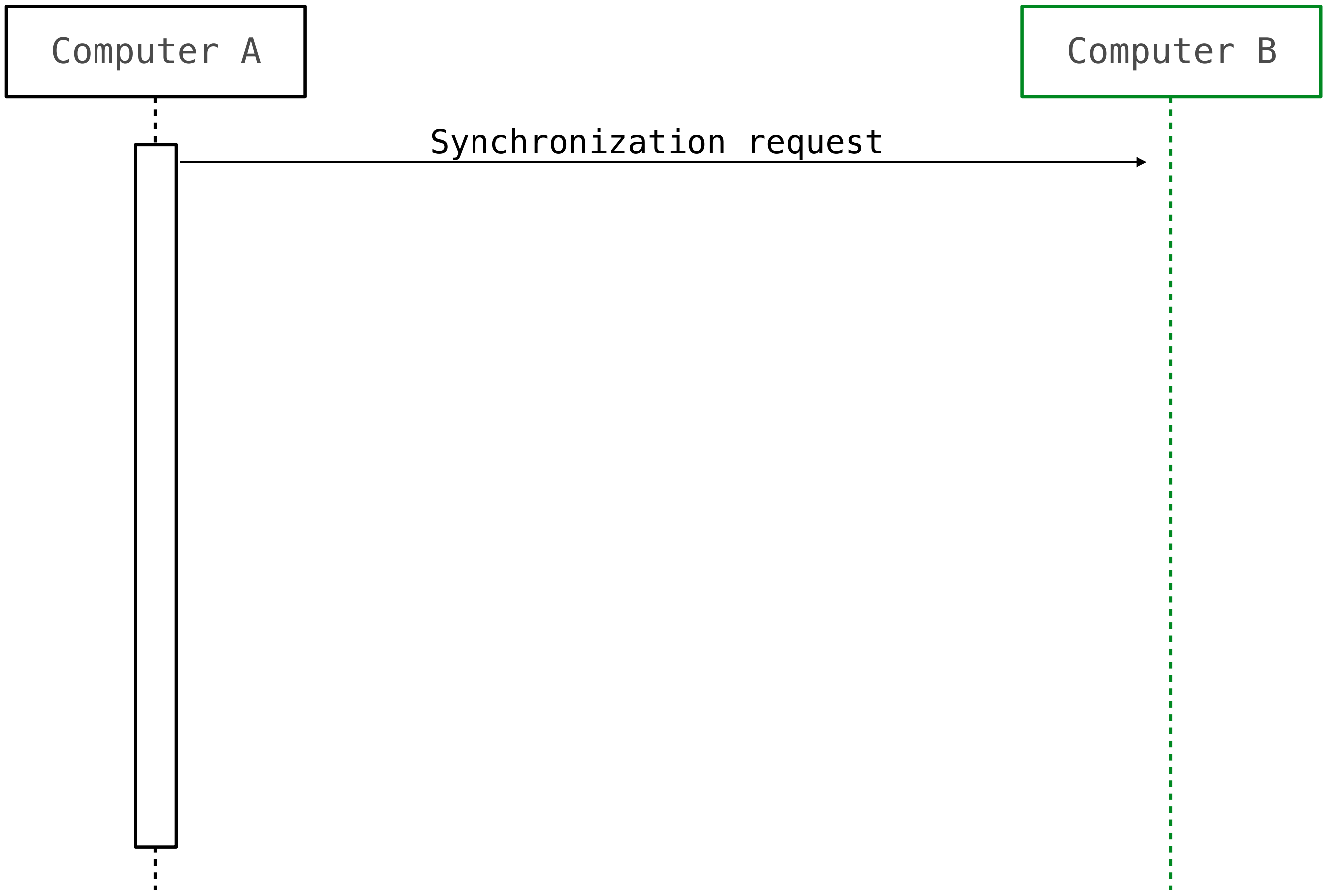
Message Sequence of a TCP Connection



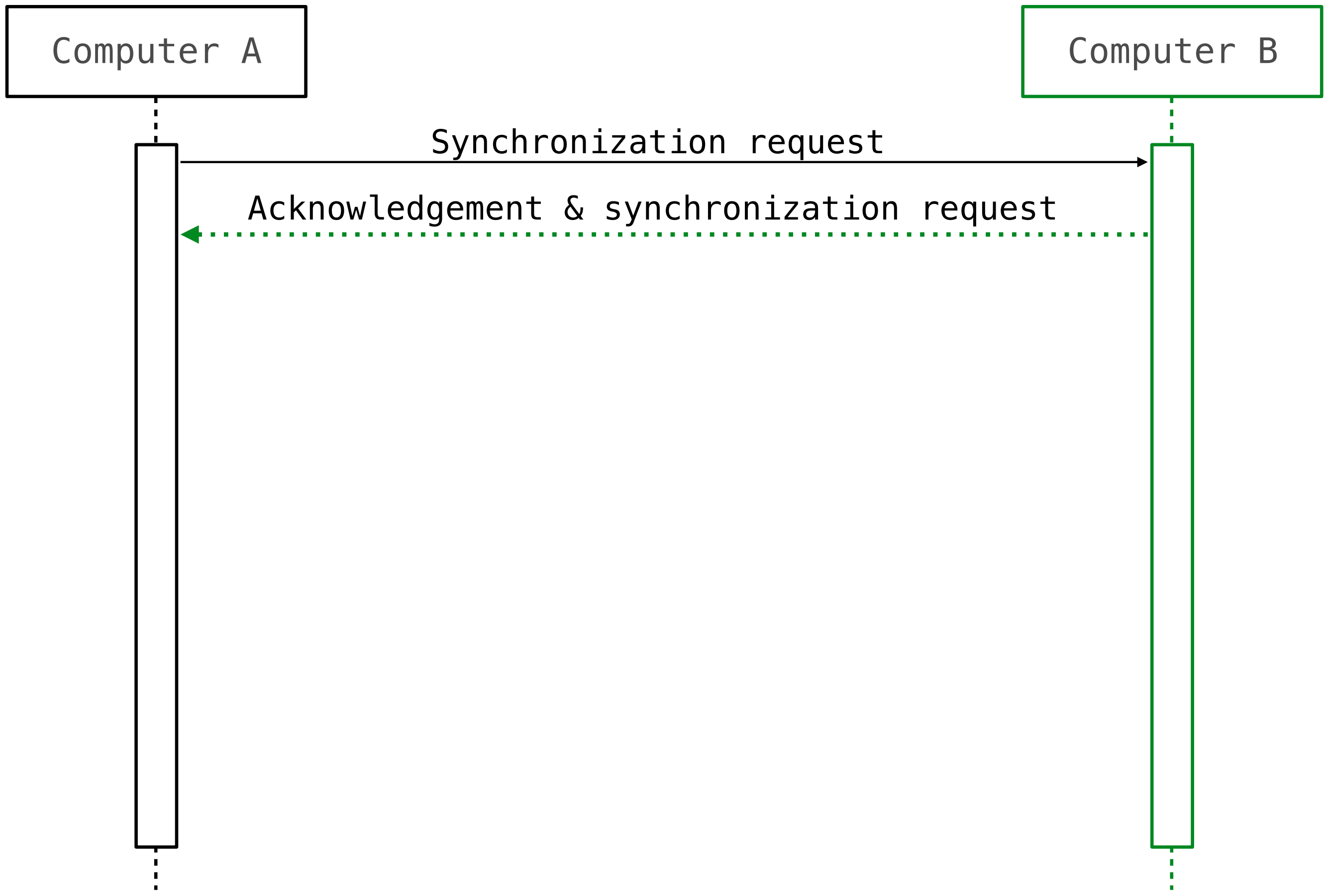
Message Sequence of a TCP Connection

Computer A
Computer B

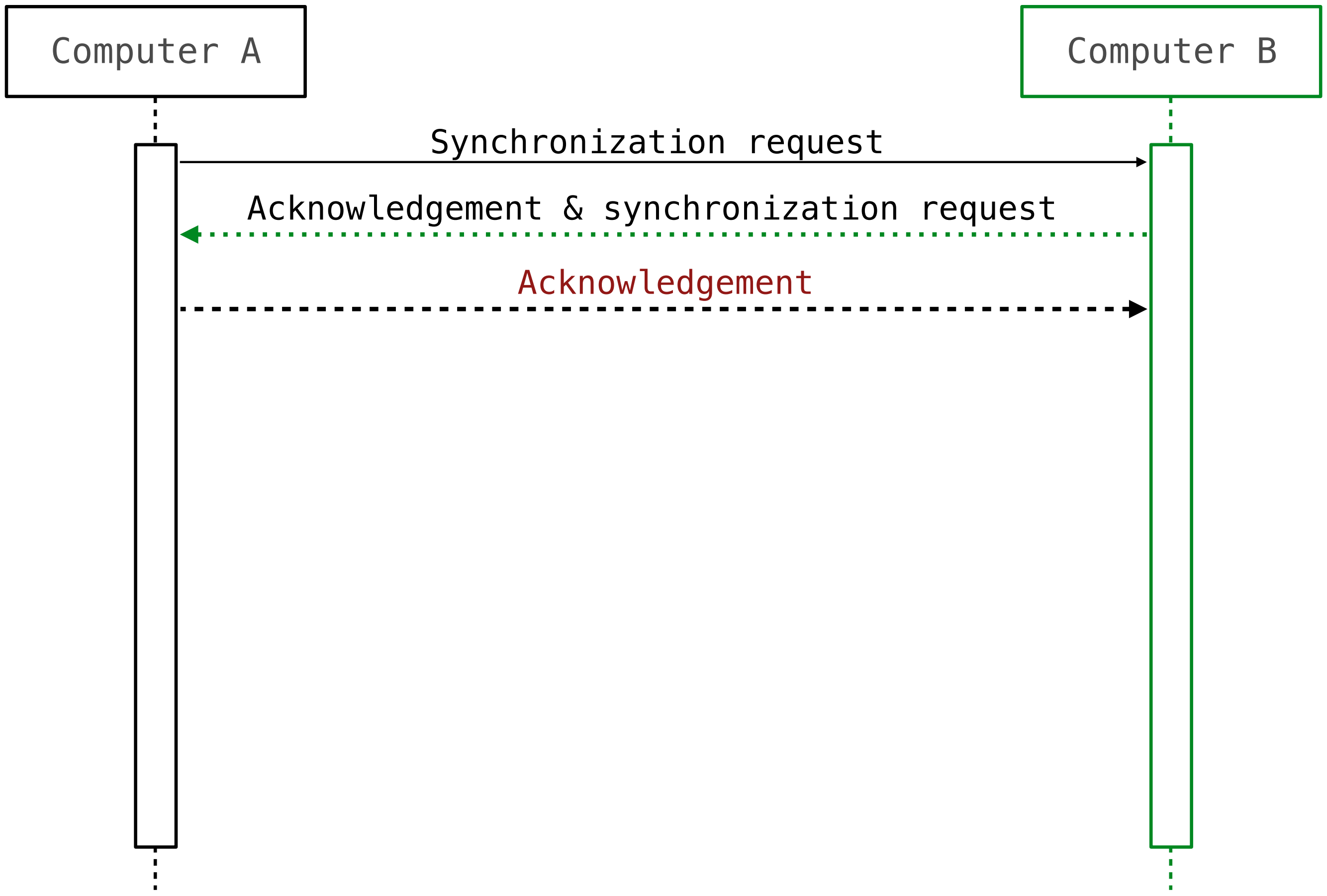
Message Sequence of a TCP Connection



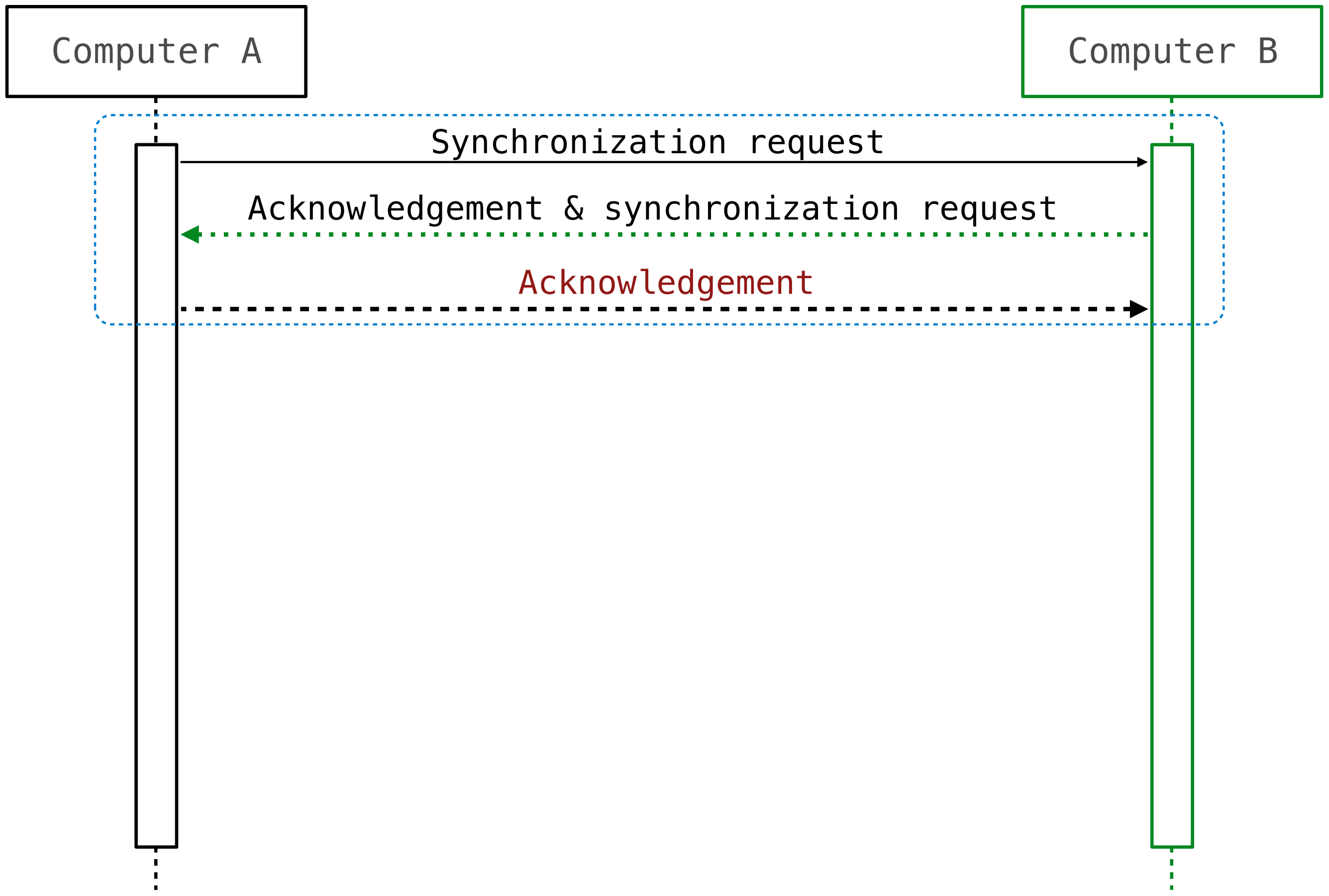
Message Sequence of a TCP Connection



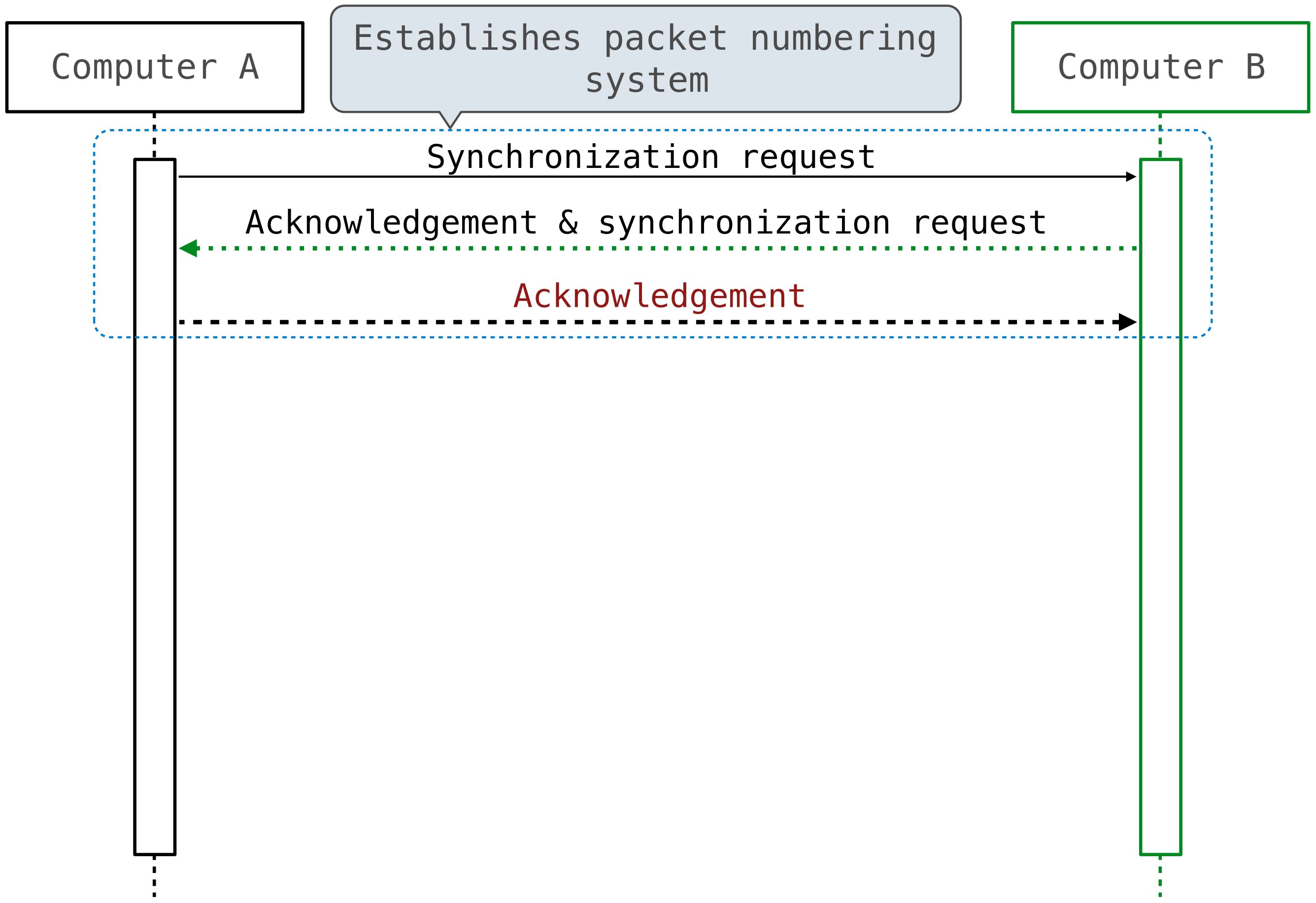
Message Sequence of a TCP Connection



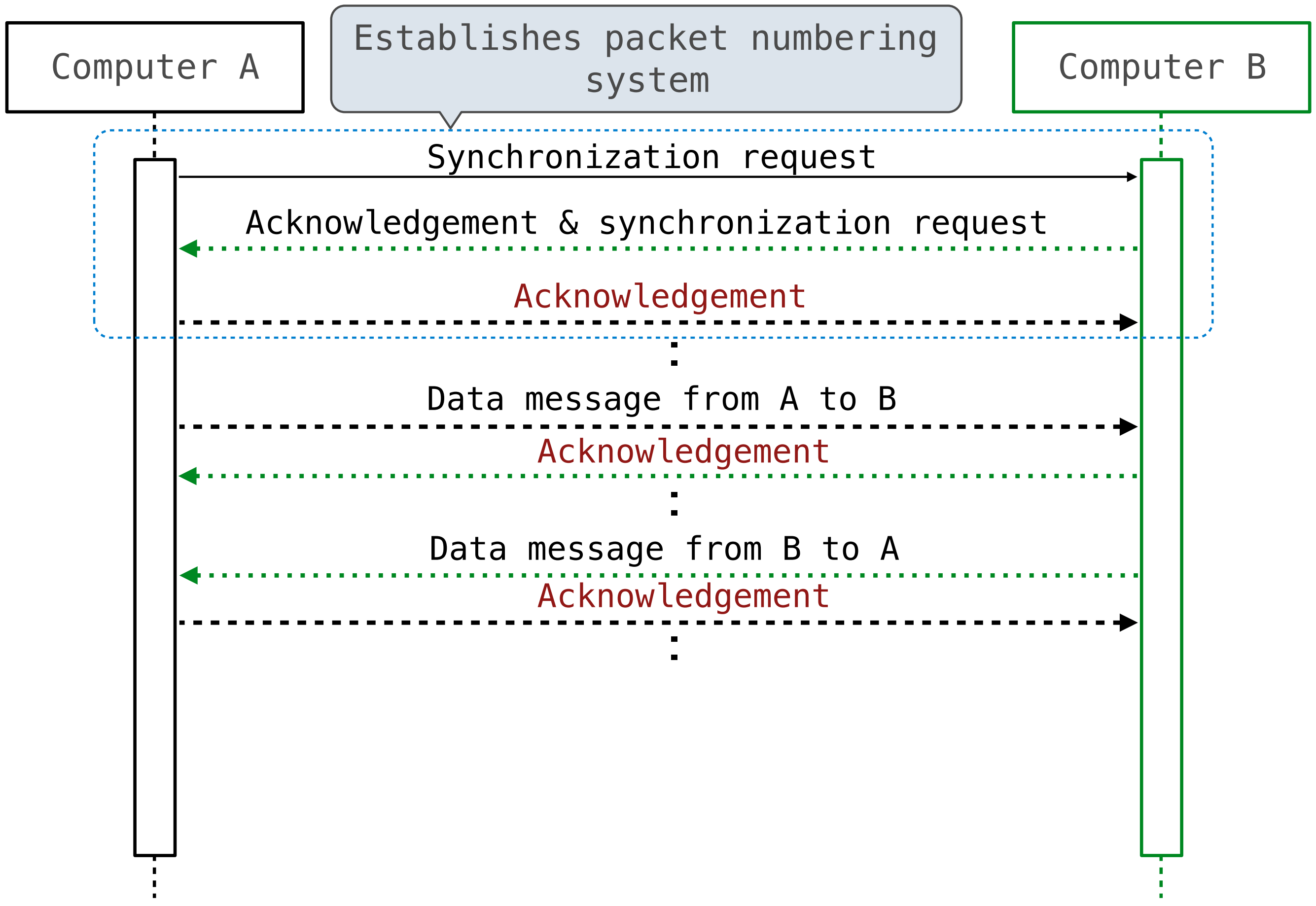
Message Sequence of a TCP Connection



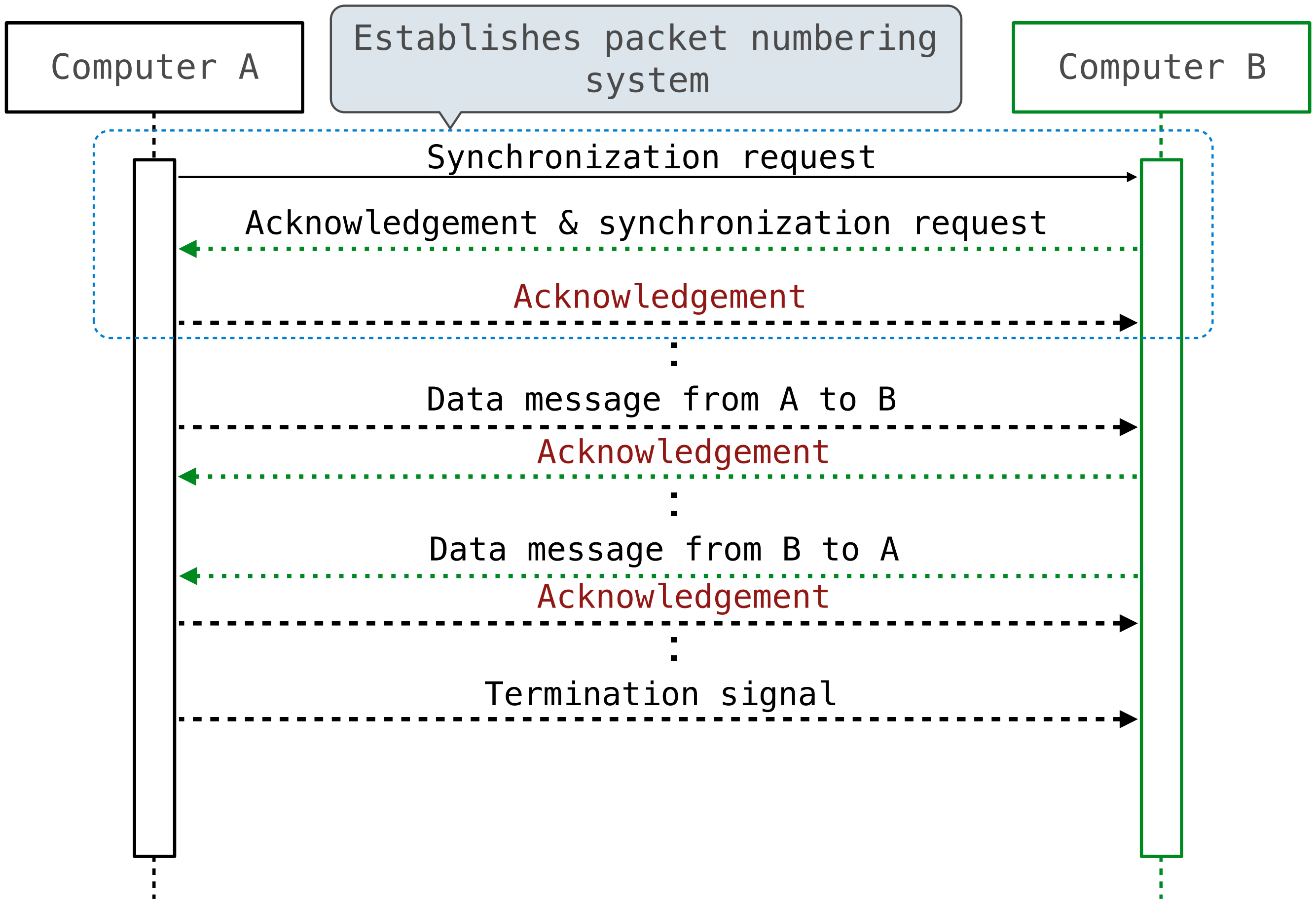
Message Sequence of a TCP Connection


Computer A
Establishes packet numbering system
Computer B
Synchronization request
Acknowledgement & synchronization request
Acknowledgement

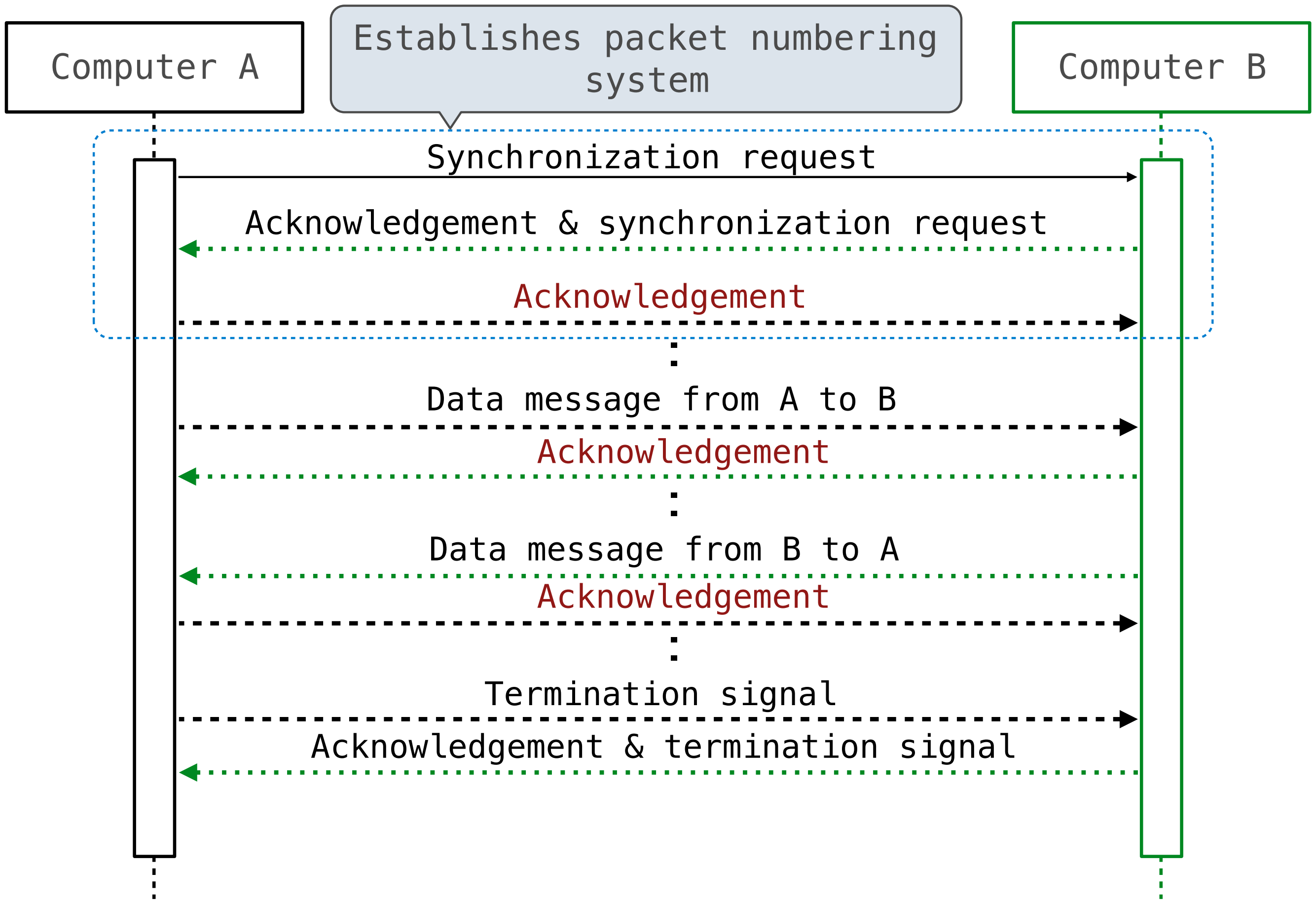
Message Sequence of a TCP Connection



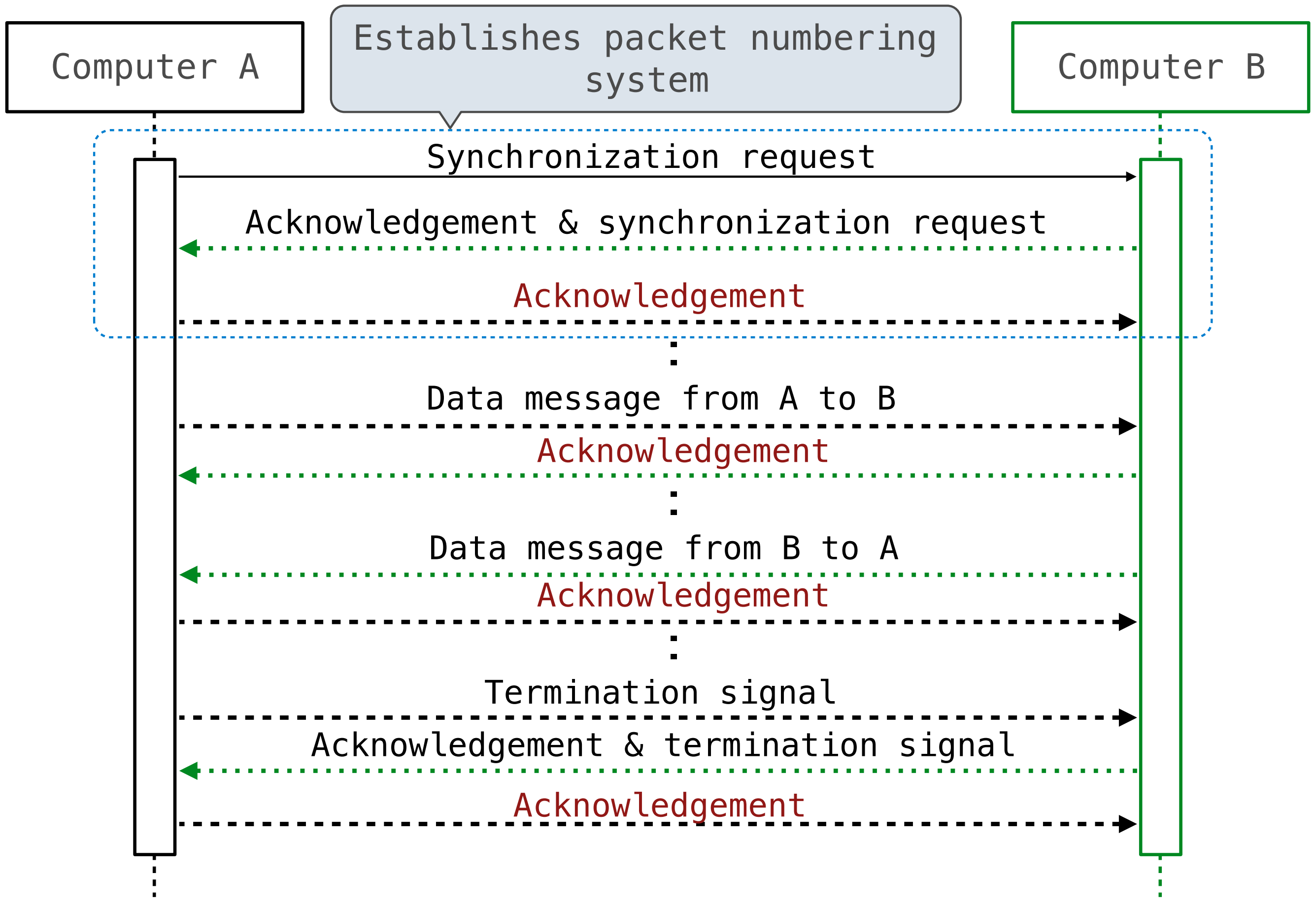
Message Sequence of a TCP Connection



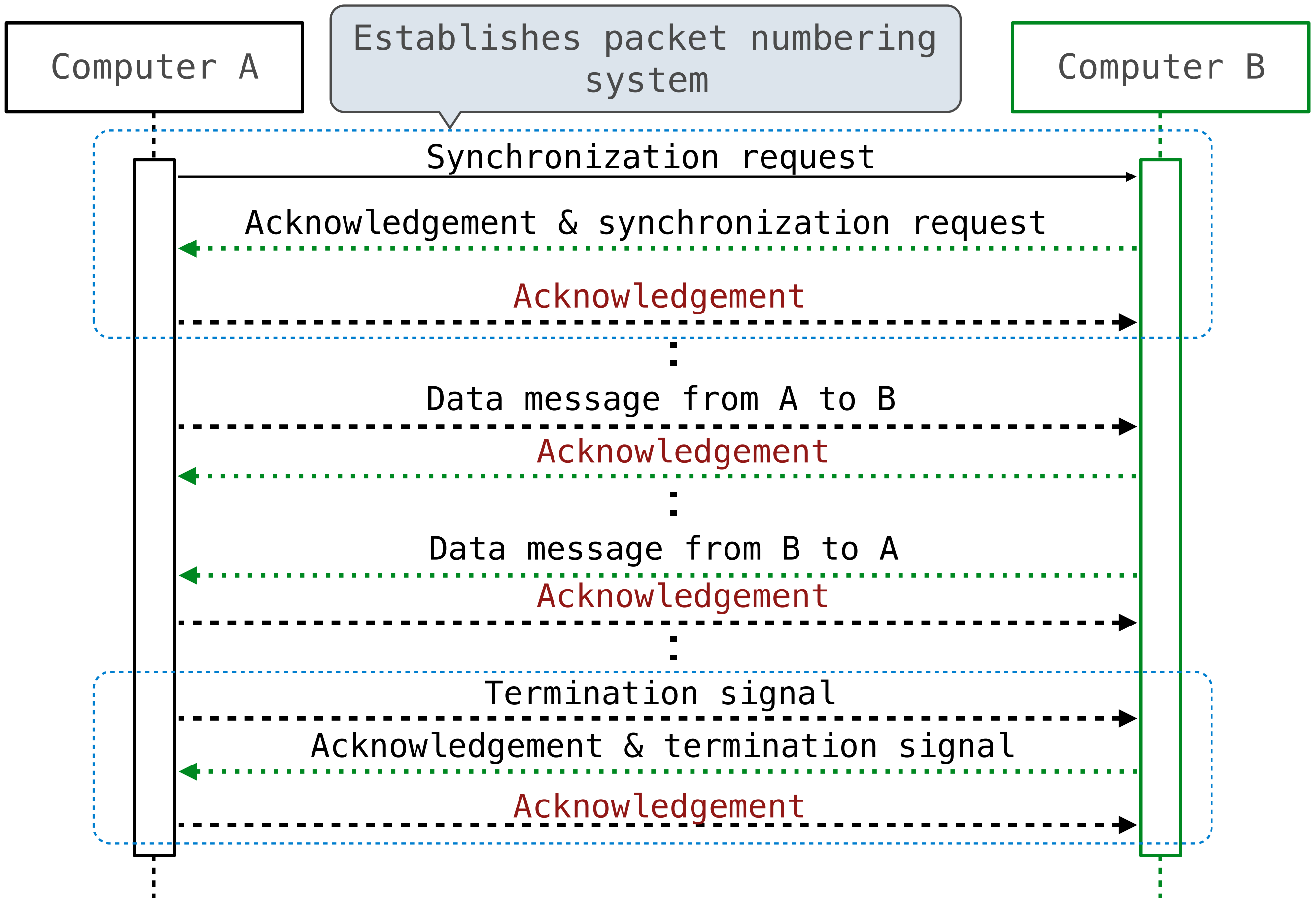
Message Sequence of a TCP Connection



Message Sequence of a TCP Connection



Message Sequence of a TCP Connection



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

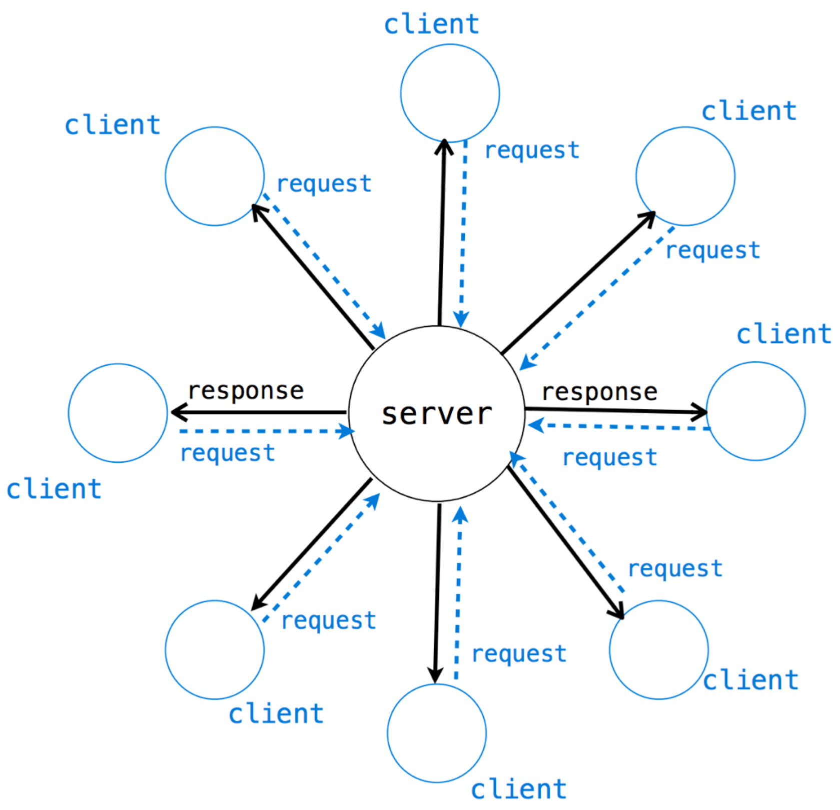
The Client/Server Architecture

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

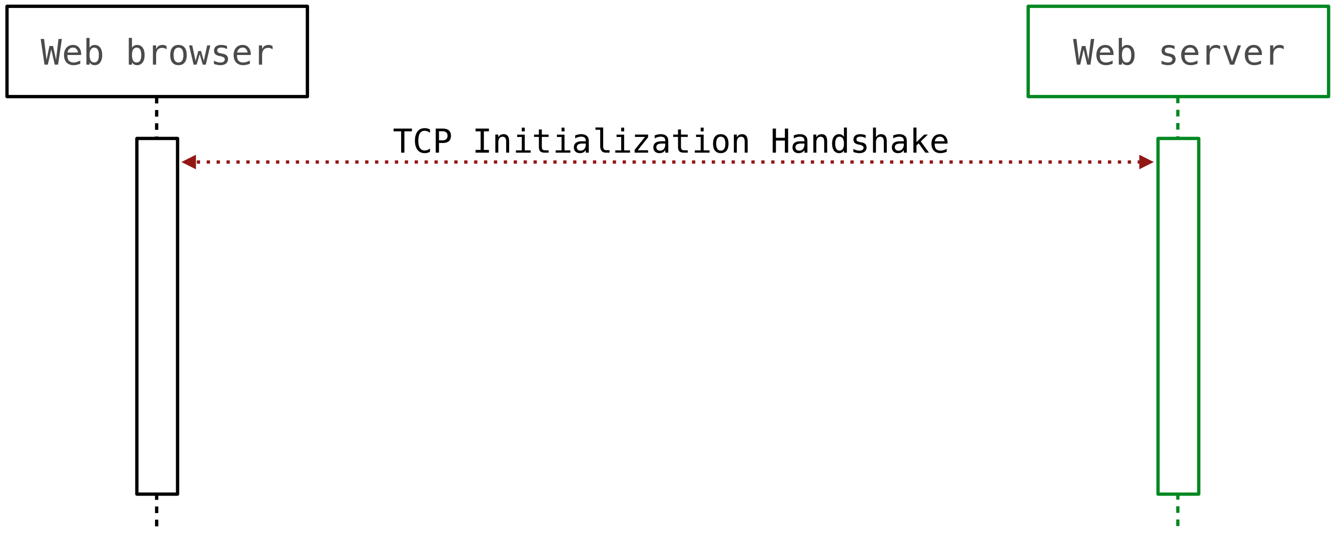
Client/Server Example: The World Wide Web

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



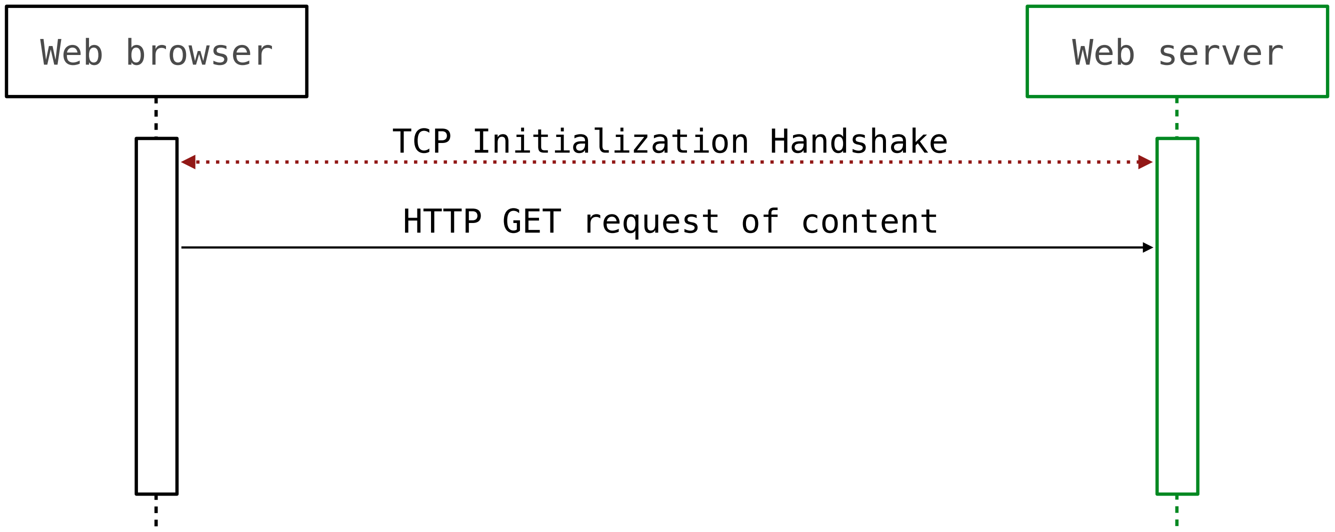
Client/Server Example: The World Wide Web

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



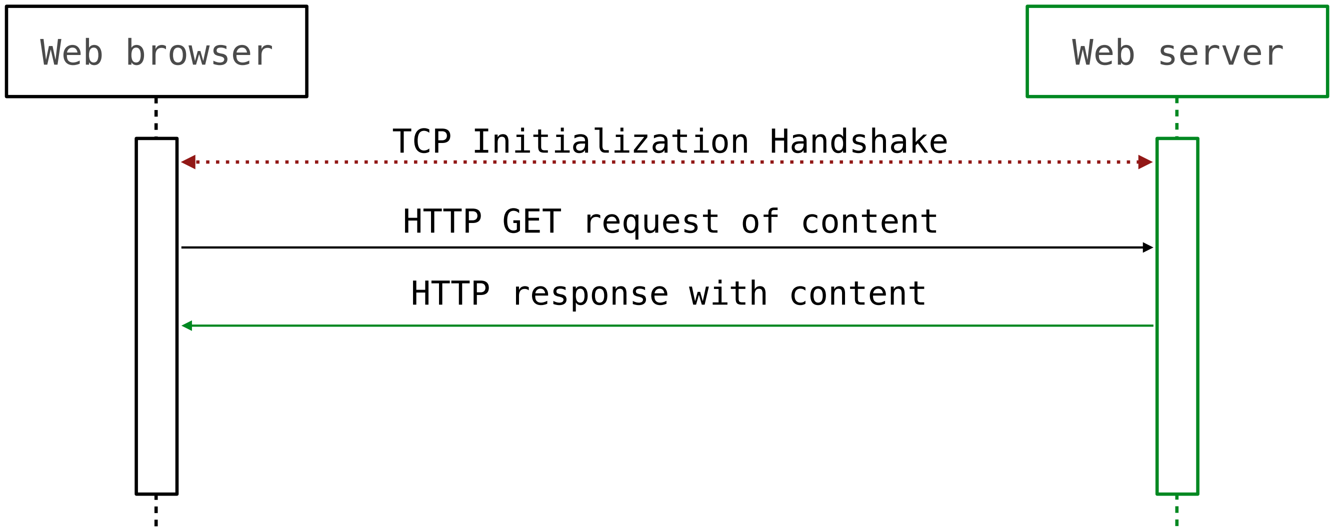
Client/Server Example: The World Wide Web

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


Web browser
Web server
TCP Initialization Handshake
HTTP GET request of content
HTTP response with content

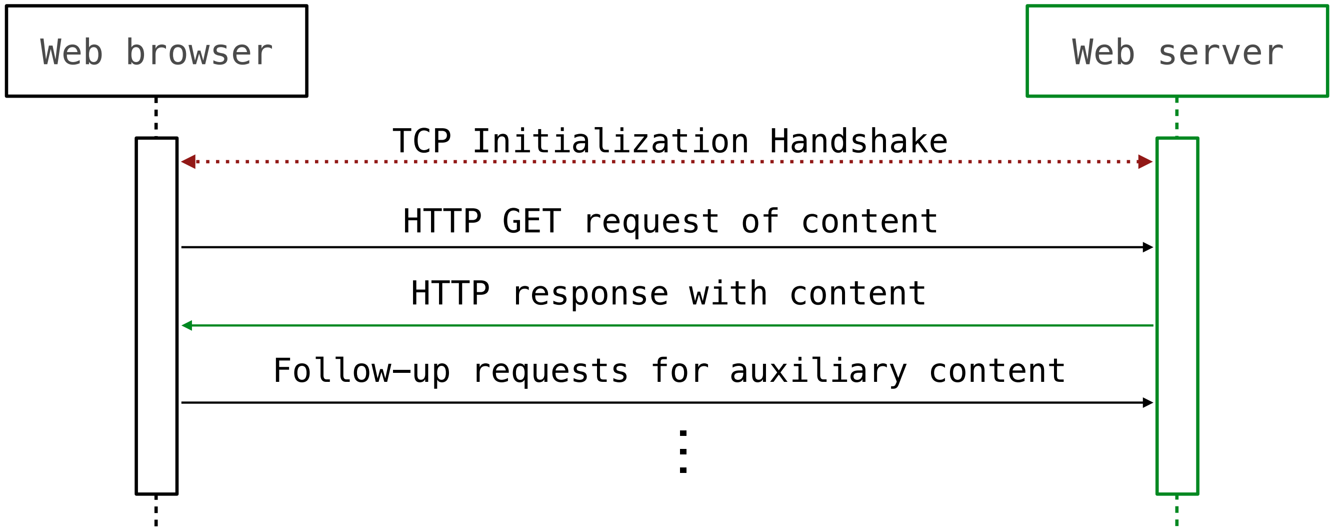
Client/Server Example: The World Wide Web

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

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

The server is a web server:

- Interpret requests and respond with content


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

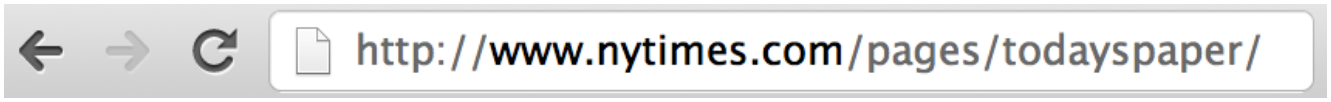
The Hypertext Transfer Protocol

The Hypertext Transfer Protocol

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

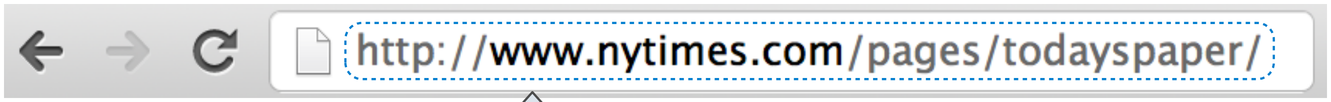
The Hypertext Transfer Protocol

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



The Hypertext Transfer Protocol

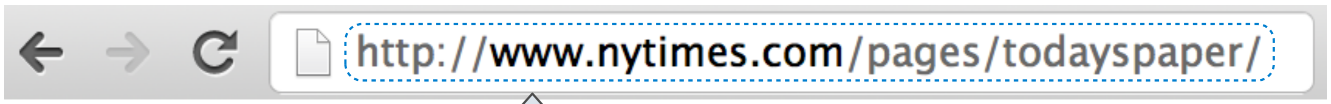
The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture



Uniform resource locator (URL)

The Hypertext Transfer Protocol

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

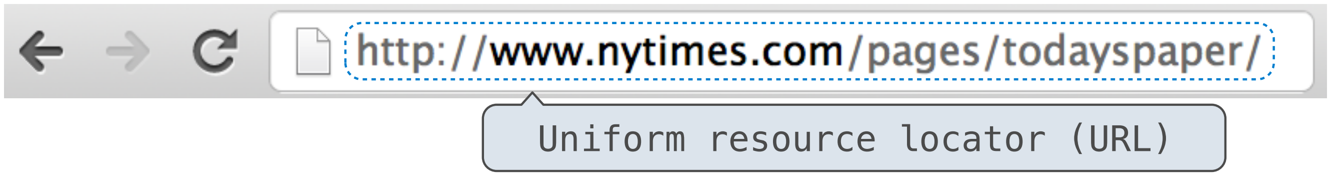


Uniform resource locator (URL)

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

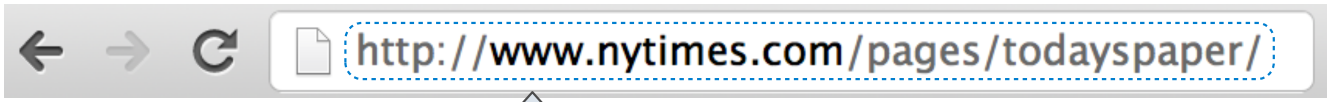

http://www.nytimes.com/pages/todayspaper/
Uniform resource locator (URL)

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



Uniform resource locator (URL)

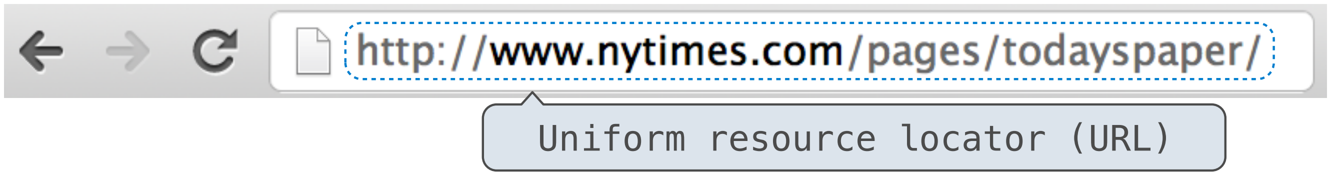
Browser issues a GET request to a server at www.nytimes.com for the content (resource) at location "pages/todayspaper"

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


http://www.nytimes.com/pages/todayspaper/
Uniform resource locator (URL)

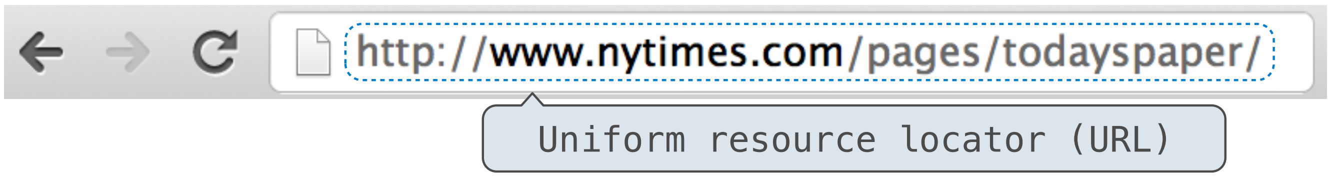
Browser issues a GET request to a server at www.nytimes.com for the content (resource) at location "pages/todayspaper"

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

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


http://www.nytimes.com/pages/todayspaper/
Uniform resource locator (URL)

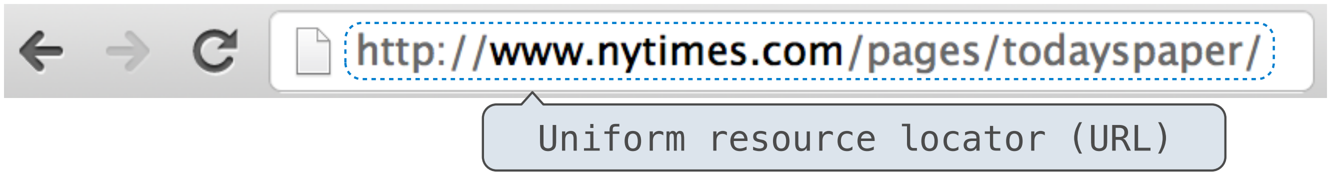
Browser issues a GET request to a server at www.nytimes.com for the content (resource) at location "pages/todayspaper"

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


http://www.nytimes.com/pages/todayspaper/
Uniform resource locator (URL)

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

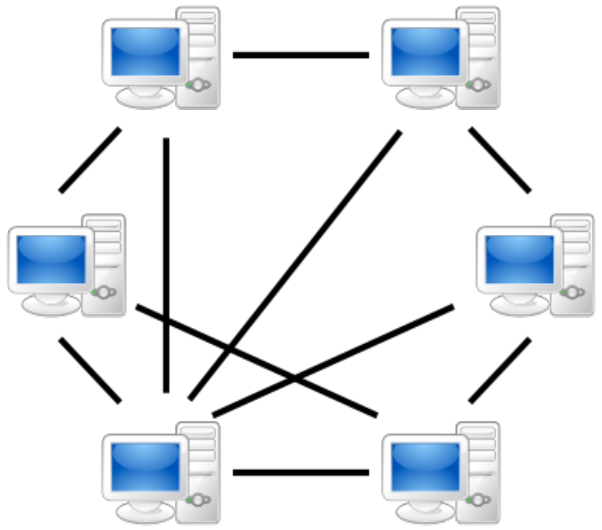
Each participant has only partial knowledge of the network

The Peer-to-Peer Architecture

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

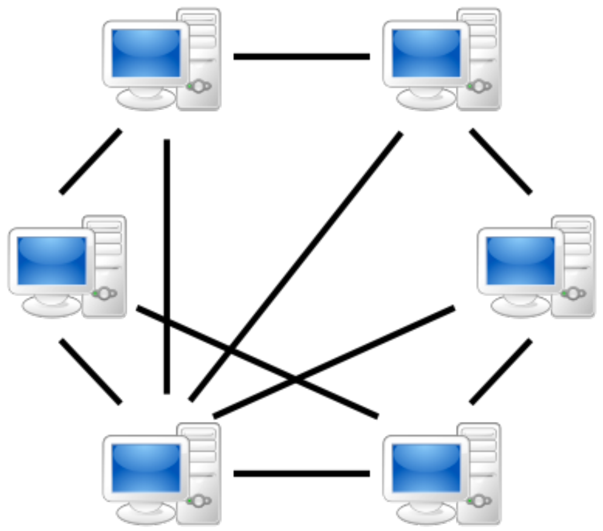
Messages are relayed through a network of participants

Each participant has only partial knowledge of the network



http://en.wikipedia.org/wiki/File:P2P-network.svg

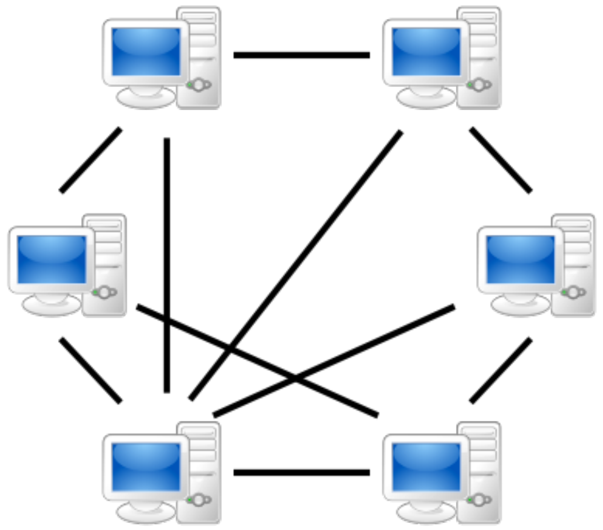
Network Structure Concerns



http://en.wikipedia.org/wiki/File:P2P-network.svg

Network Structure Concerns

Some data transfers on the Internet are faster than others

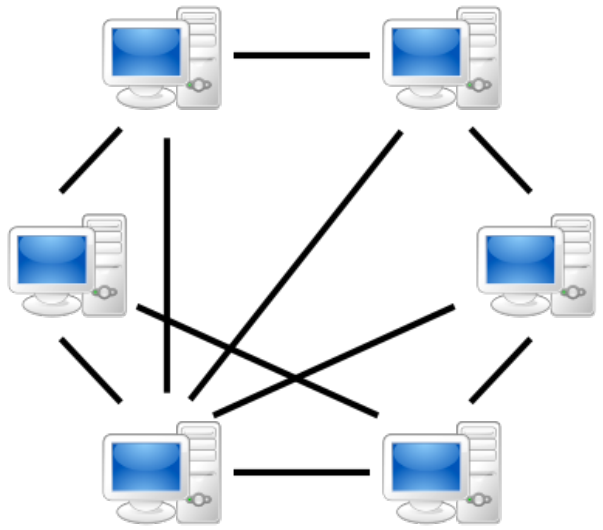


http://en.wikipedia.org/wiki/File:P2P-network.svg

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

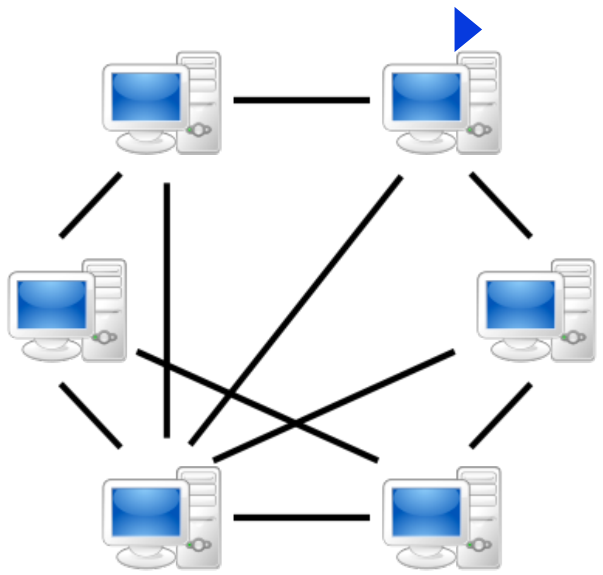


http://en.wikipedia.org/wiki/File:P2P-network.svg

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

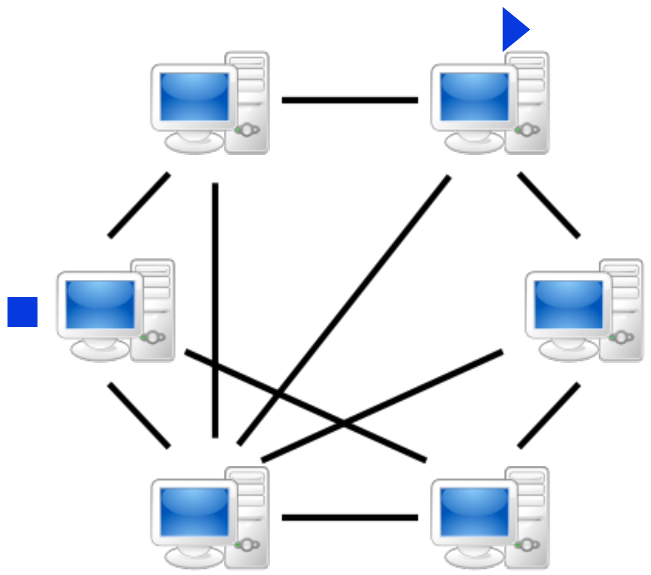


http://en.wikipedia.org/wiki/File:P2P-network.svg

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

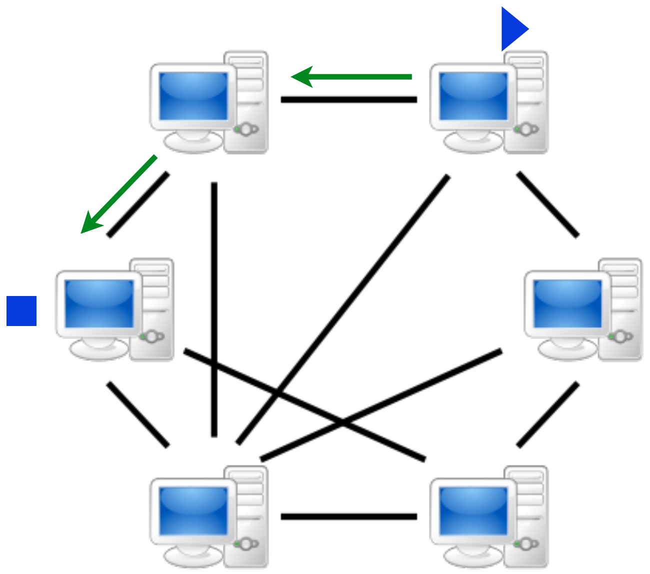


http://en.wikipedia.org/wiki/File:P2P-network.svg

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

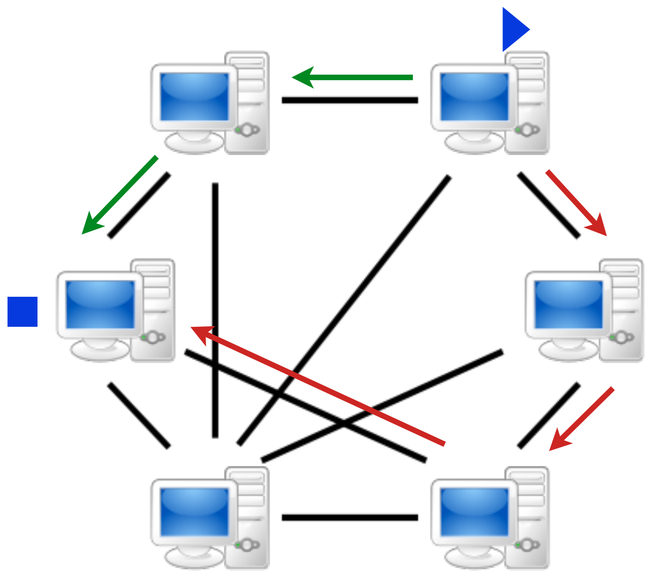


http://en.wikipedia.org/wiki/File:P2P-network.svg

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

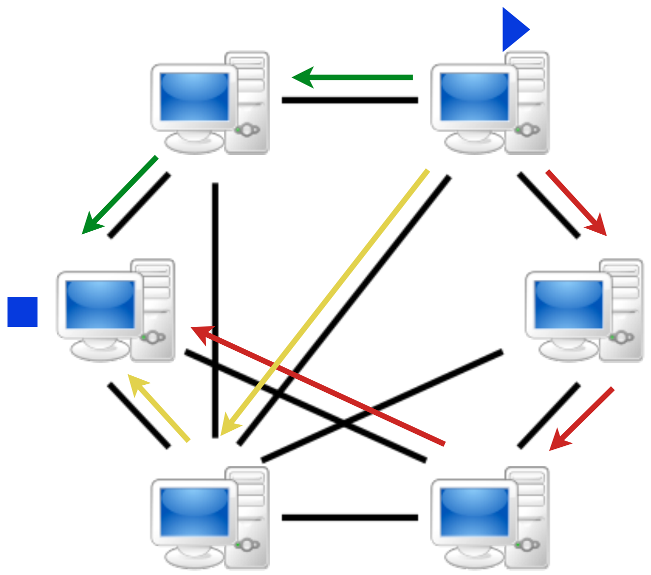


http://en.wikipedia.org/wiki/File:P2P-network.svg

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



http://en.wikipedia.org/wiki/File:P2P-network.svg

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

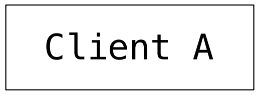
Example: Skype

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

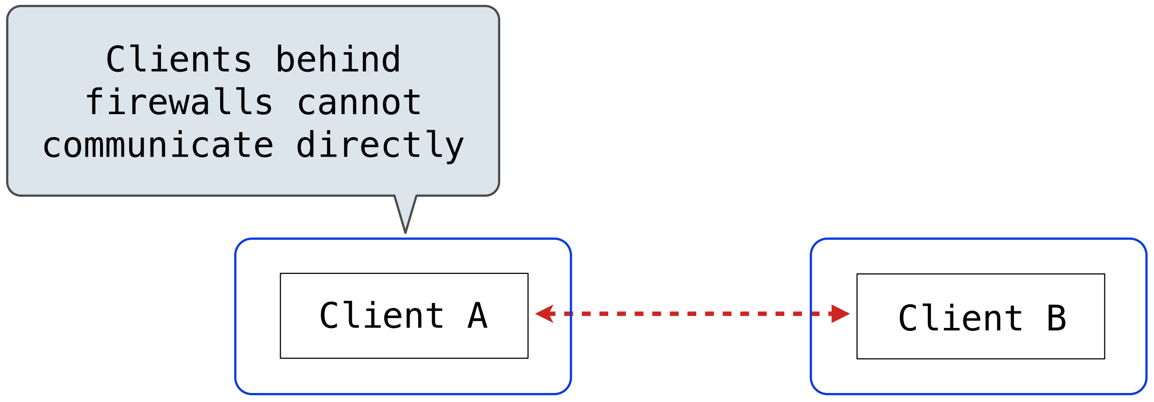
Example: Skype

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



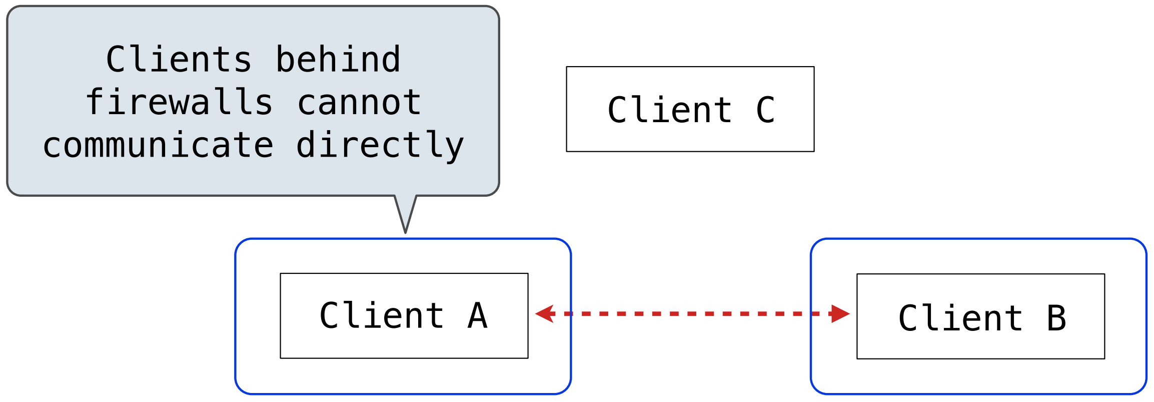
Example: Skype

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



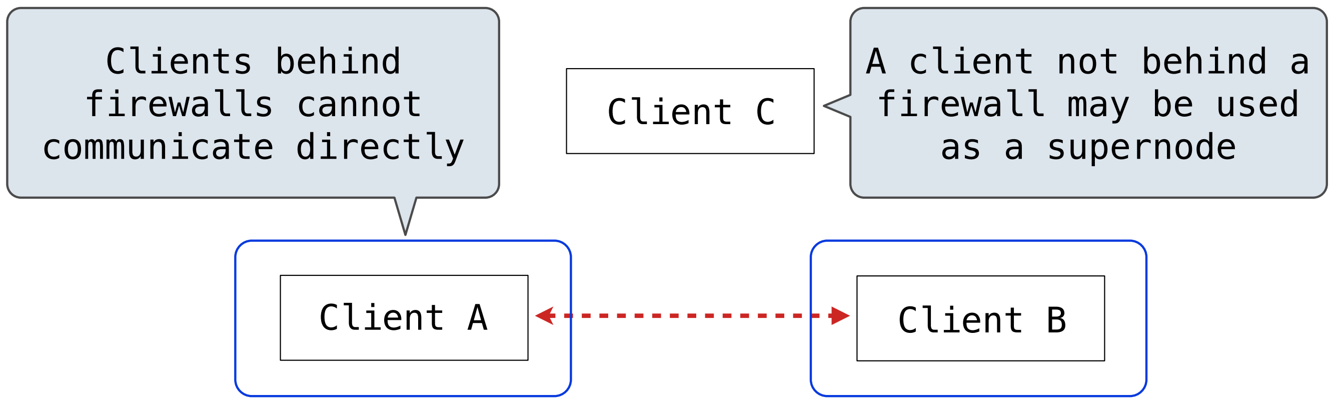
Example: Skype

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
Client C
A client not behind a firewall may be used as a supernode
Client A
Client B

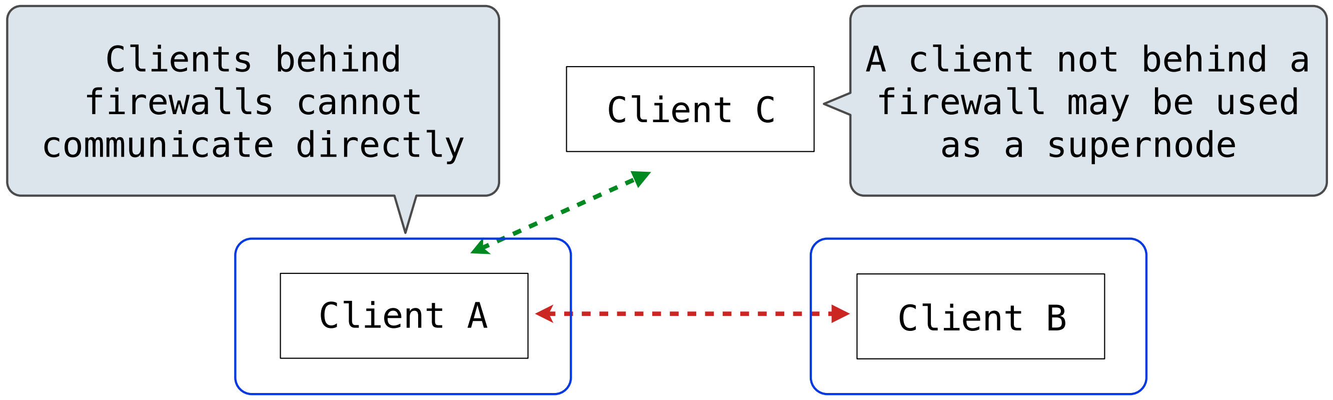
Example: Skype

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
Client C
A client not behind a firewall may be used as a supernode
Client A
Client B

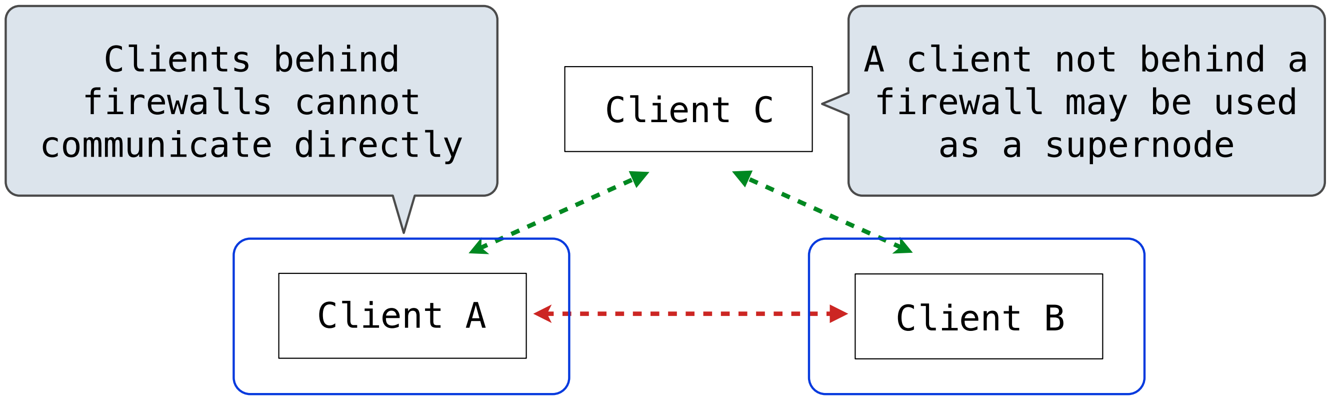
Example: Skype

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
Client C
A client not behind a firewall may be used as a supernode
Client A
Client B