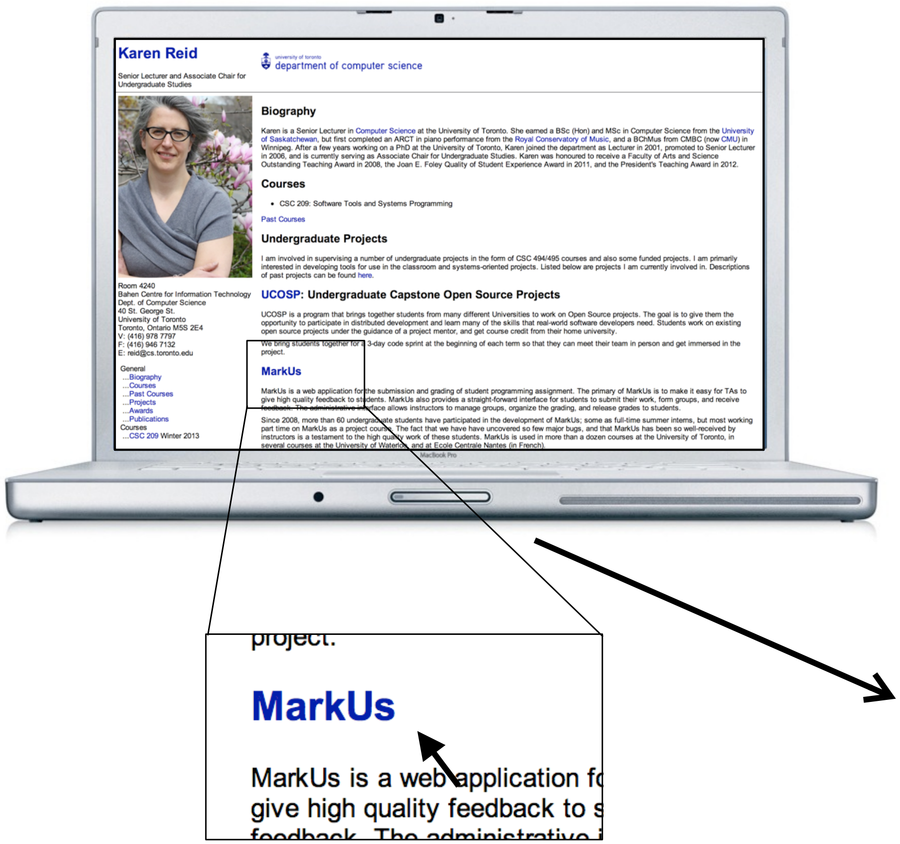
# What makes the world wide web work?

Karen Reid

image:Peter Werkman (www.peterwerkman.nl)

# Simple Web Request



## Karen Reid

Senior Lecturer and Associate Chair for Undergraduate Studies

university of toronto
department of computer science

### Biography

Karen is a Senior Lecturer in Computer Science at the University of Toronto. She earned a BSc (Hon) and MSc in Computer Science from the University of Saskatchewan, but first completed an ARCT in piano performance from the Royal Conservatory of Music, and a BChMus from CMBC (now CMU) in Winnipeg. After a few years working on a PhD at the University of Toronto, Karen joined the department as Lecturer in 2001, promoted to Senior Lecturer in 2006, and is currently serving as Associate Chair for Undergraduate Studies. Karen was honoured to receive a Faculty of Arts and Science Outstanding Teaching Award in 2008, the Joan E. Foley Quality of Student Experience Award in 2011, and the President's Teaching Award in 2012.

### Courses

- CSC 209: Software Tools and Systems Programming

Past Courses

### Undergraduate Projects

I am involved in supervising a number of undergraduate projects in the form of CSC 494/495 courses and also some funded projects. I am primarily interested in developing tools for use in the classroom and systems-oriented projects. Listed below are projects I am currently involved in. Descriptions of past projects can be found here.

### UCOSP: Undergraduate Capstone Open Source Projects

UCOSP is a program that brings together students from many different Universities to work on Open Source projects. The goal is to give them the opportunity to participate in distributed development and learn many of the skills that real-world software developers need. Students work on existing open source projects under the guidance of a project mentor, and get course credit from their home university.

We bring students together for a 3-day code sprint at the beginning of each term so that they can meet their team in person and get immersed in the project.

#### MarkUs

MarkUs is a web application for the submission and grading of student programming assignment. The primary of MarkUs is to make it easy for TAs to give high quality feedback to students. MarkUs also provides a straight-forward interface for students to submit their work, form groups, and receive feedback. The administrative interface allows instructors to manage groups, organize the grading, and release grades to students.

Since 2008, more than 60 undergraduate students have participated in the development of MarkUs; some as full-time summer interns, but most working part time on MarkUs as a project course. The fact that we have have uncovered so few major bugs, and that MarkUs has been so well-received by instructors is a testament to the high quality work of these students. MarkUs is used in more than a dozen courses at the University of Toronto, in several courses at the University of Waterloo, and at Ecole Centrale Nantes (in French).

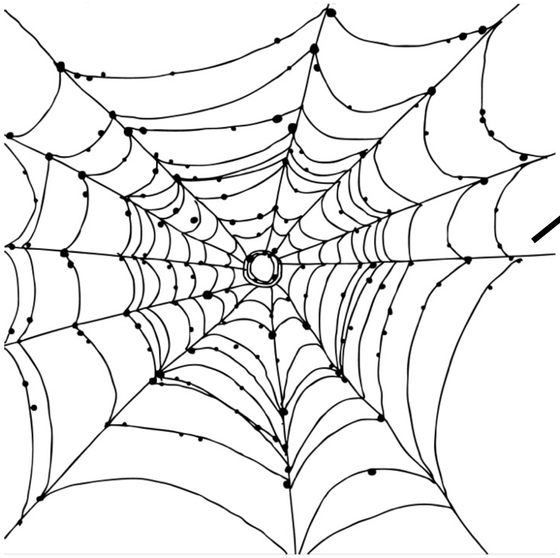
Room 4240
Bahen Centre for Information Technology
Dept. of Computer Science
40 St. George St.
University of Toronto
Toronto, Ontario M5S 2E4
V: (416) 978 7797
F: (416) 946 7132
E: reid@cs.toronto.edu

General
...Biography
...Courses
...Past Courses
...Projects
...Awards
...Publications
Courses
...CSC 209 Winter 2013

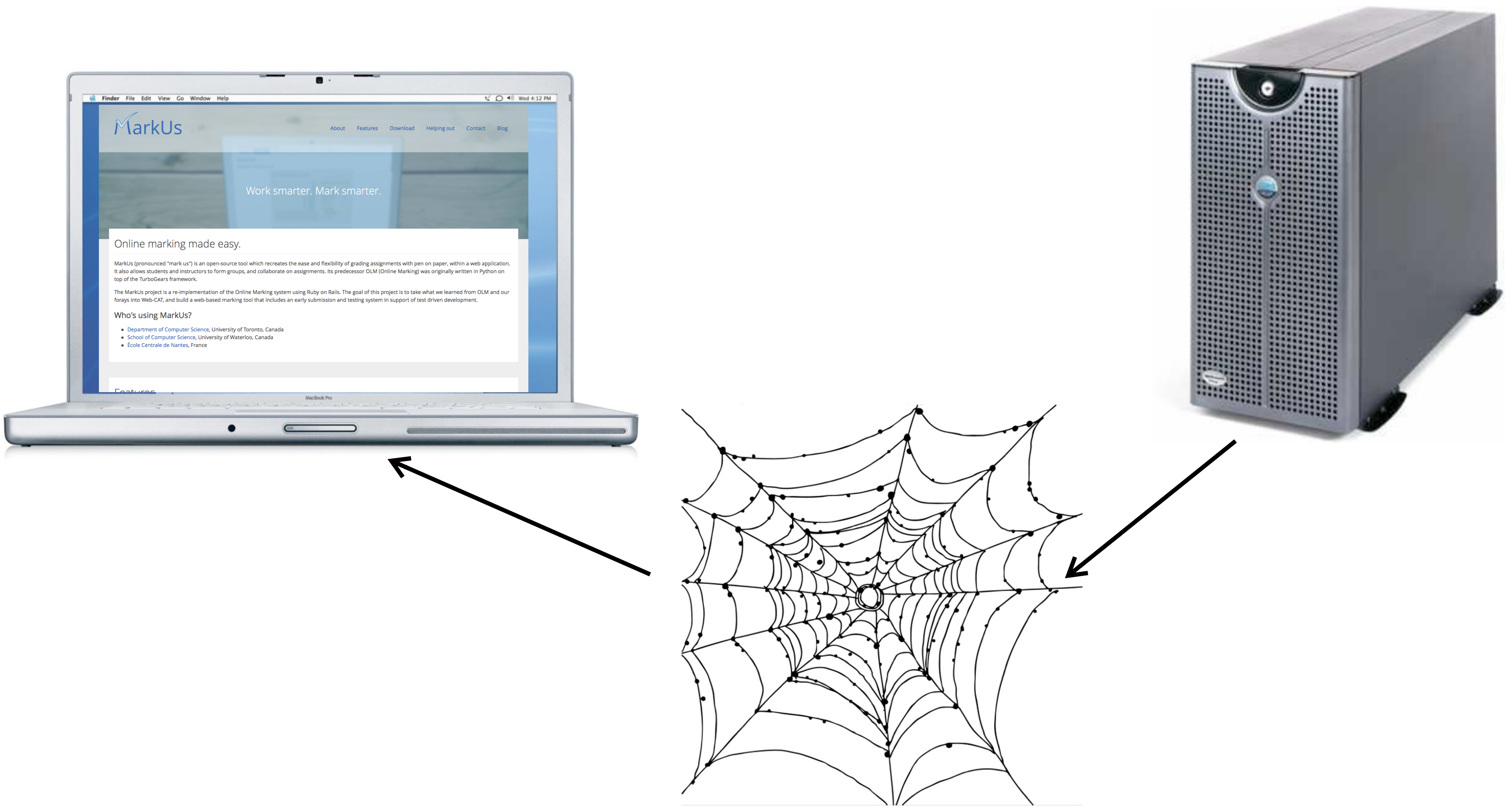
project.

MarkUs

MarkUs is a web application f
give high quality feedback to s
feedback. The administrative i



# Response



Work smarter. Mark smarter.

## Online marking made easy.

MarkUs (pronounced "mark us") is an open-source tool which recreates the ease and flexibility of grading assignments with pen on paper, within a web application. It also allows students and instructors to form groups, and collaborate on assignments. Its predecessor OLM (Online Marking) was originally written in Python on top of the TurboGears framework.

The MarkUs project is a re-implementation of the Online Marking system using Ruby on Rails. The goal of this project is to take what we learned from OLM and our forays into Web-CAT, and build a web-based marking tool that includes an early submission and testing system in support of test driven development.

### Who's using MarkUs?

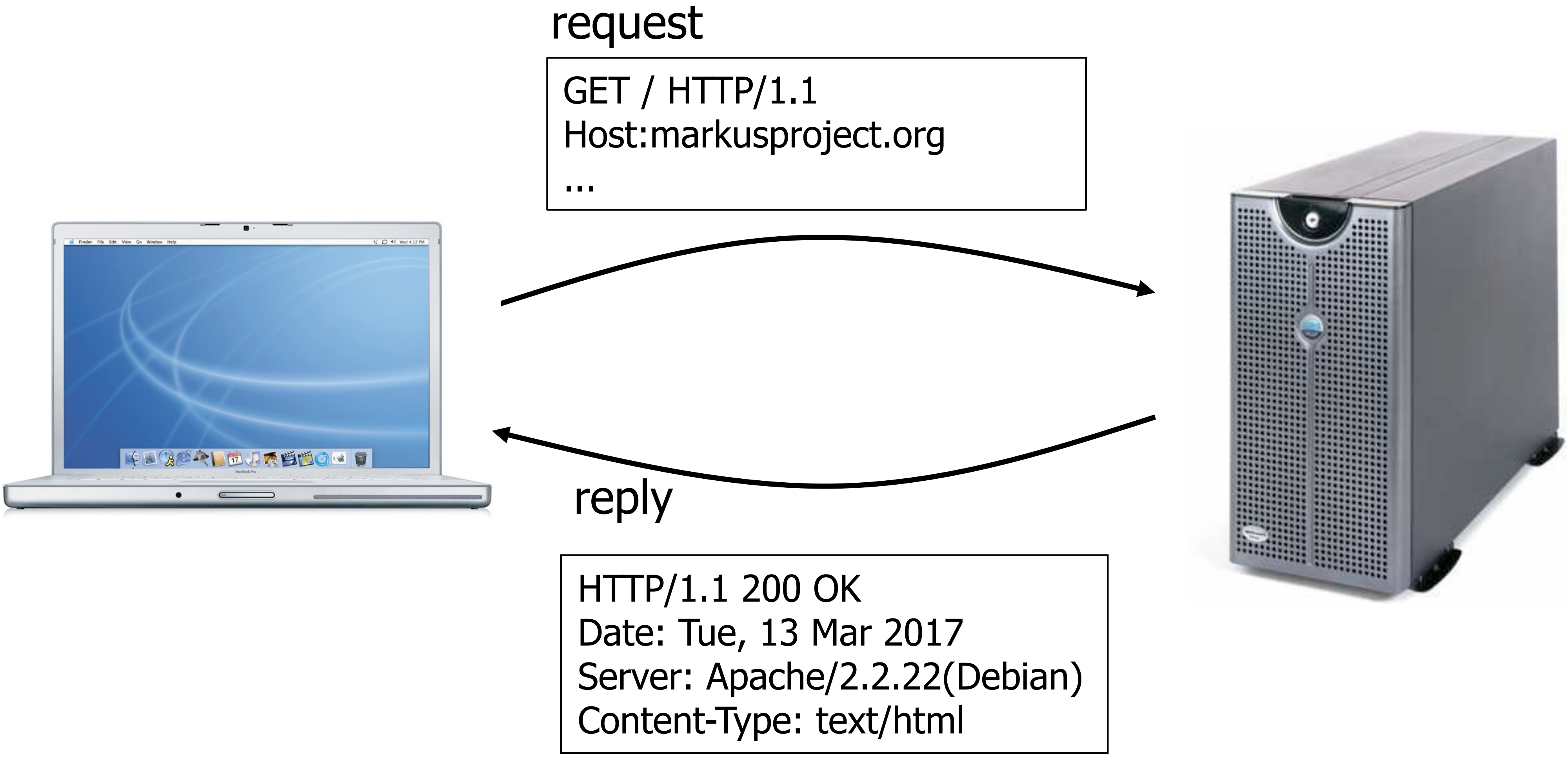
- Department of Computer Science, University of Toronto, Canada
- School of Computer Science, University of Waterloo, Canada
- École Centrale de Nantes, France

Features

# The Request

- How do we tell the web server what we want?
- How do we even find the web server?
- How do the web server and browser talk to each other?

# HTTP Request



request

```
GET / HTTP/1.1
Host:markusproject.org
...
```

reply

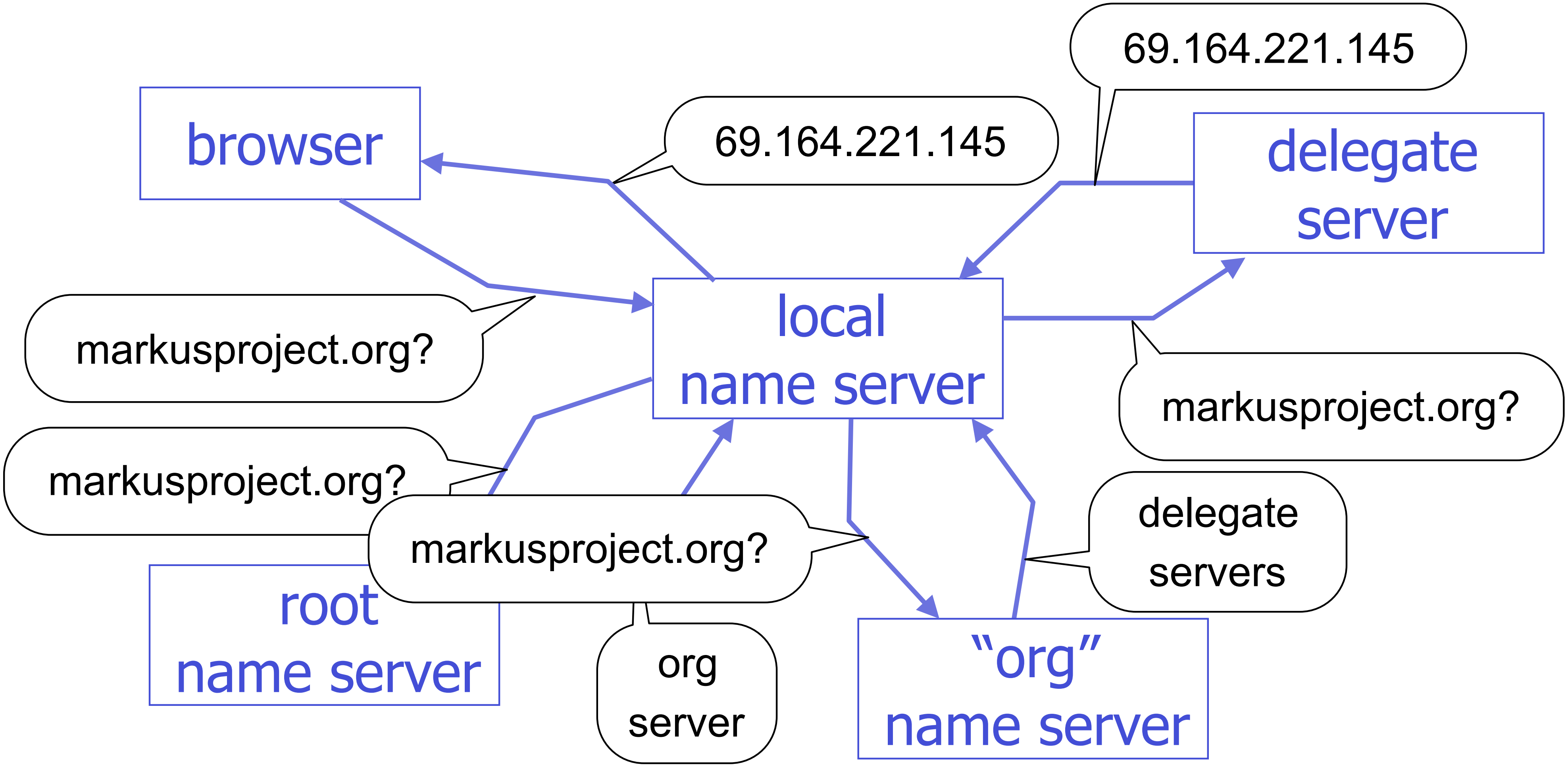
```
HTTP/1.1 200 OK
Date: Tue, 13 Mar 2017
Server: Apache/2.2.22(Debian)
Content-Type: text/html
```

# How do we find the server?

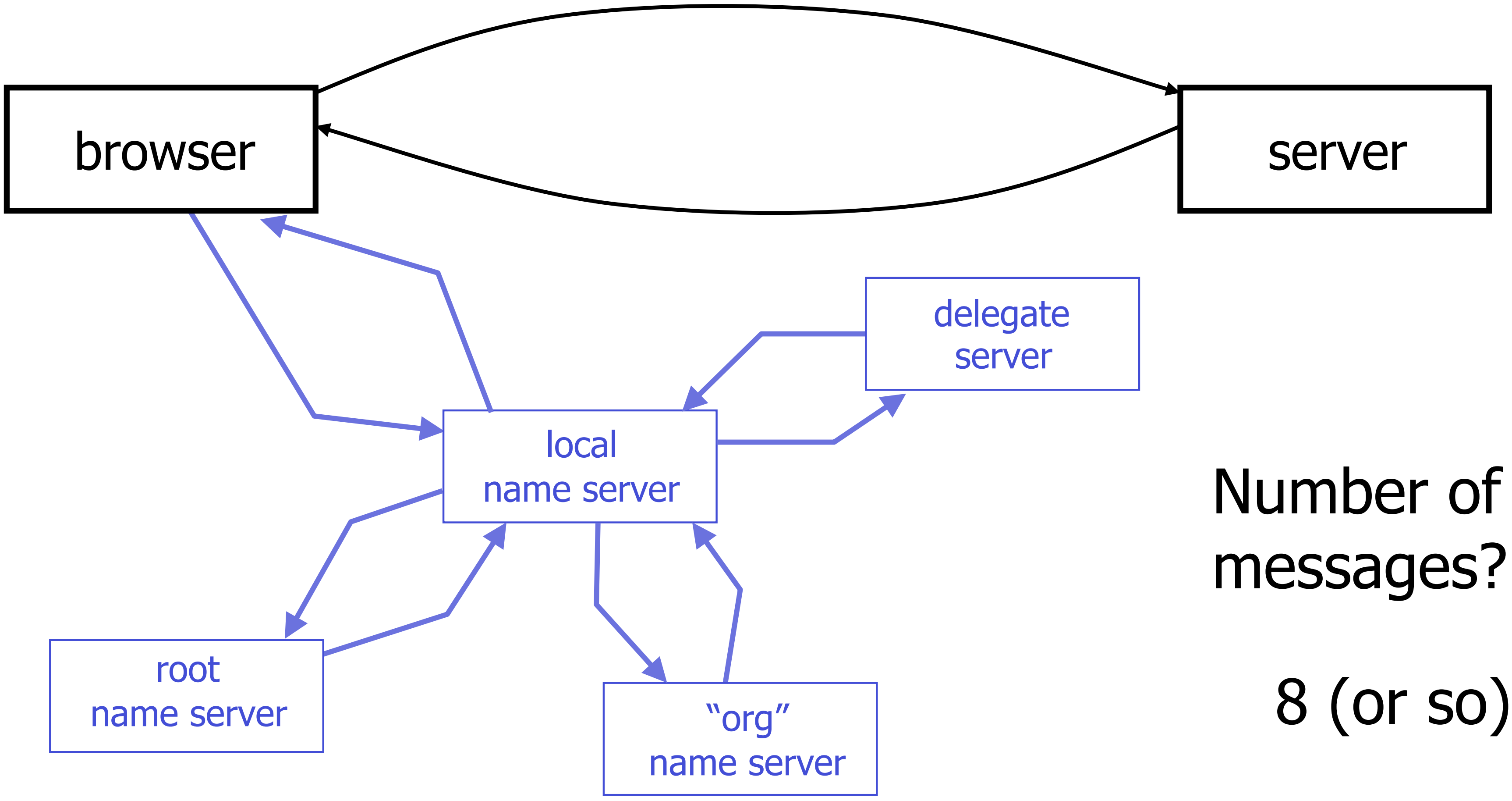
- Every computer on the Internet has an Internet address.
- Called an IP address (Internet Protocol)
- An IP address is 4 numbers separated by dots.

markusproject.org = 69.164.221.145

# Domain Name Servers



# This is getting complicated!



Number of messages?

8 (or so)

# Now what?

- Okay, we have the address.
- What do we do with it?
- Let’s look at how two computers communicate.
- HTTP is a high-level protocol
- HTTP is specific to the web.
- Computers communicate for many reasons.

# Protocols

- Computers use several layers of general protocols to communicate.
- To understand why these layers are important, think about how a company sends you an invoice for a purchase.

# Protocols

Invoice:
Customer: Karen Reid
Order No: 5379

| Qty: | | Unit Price | Total |
|---|---|---|---|
| 1 | Athalon | 219.00 | 219.00 |
| 2 | 128 MB | 149.95 | 299.90 |
| | | Subtotal | 518.90 |
| | | Tax | 77.84 |
| | | TOTAL | 596.74 |

Karen Reid Feb 18, 2001

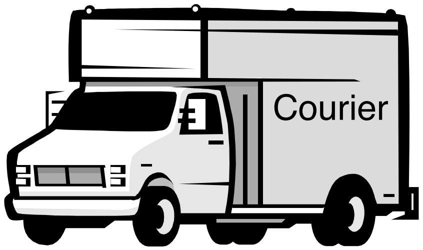
Payable to: CPUS are us $596.74
Five hundred ninety six 74/100

CPUS are us

Karen Reid
Dept. of Computer Science
University of Toronto

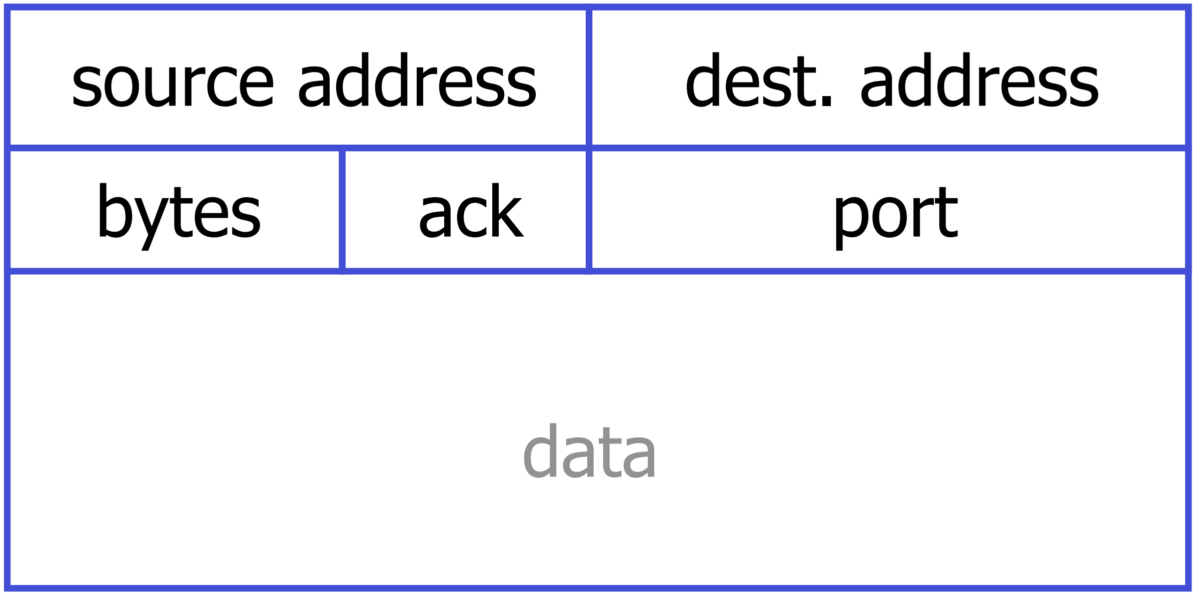
Karen Reid

CPUS are us
0 College Street
Toronto Ontario M5S 3G4



# TCP/IP

- Transmission Control Protocol.
- Tells us how to package up the data.



source address

dest. address

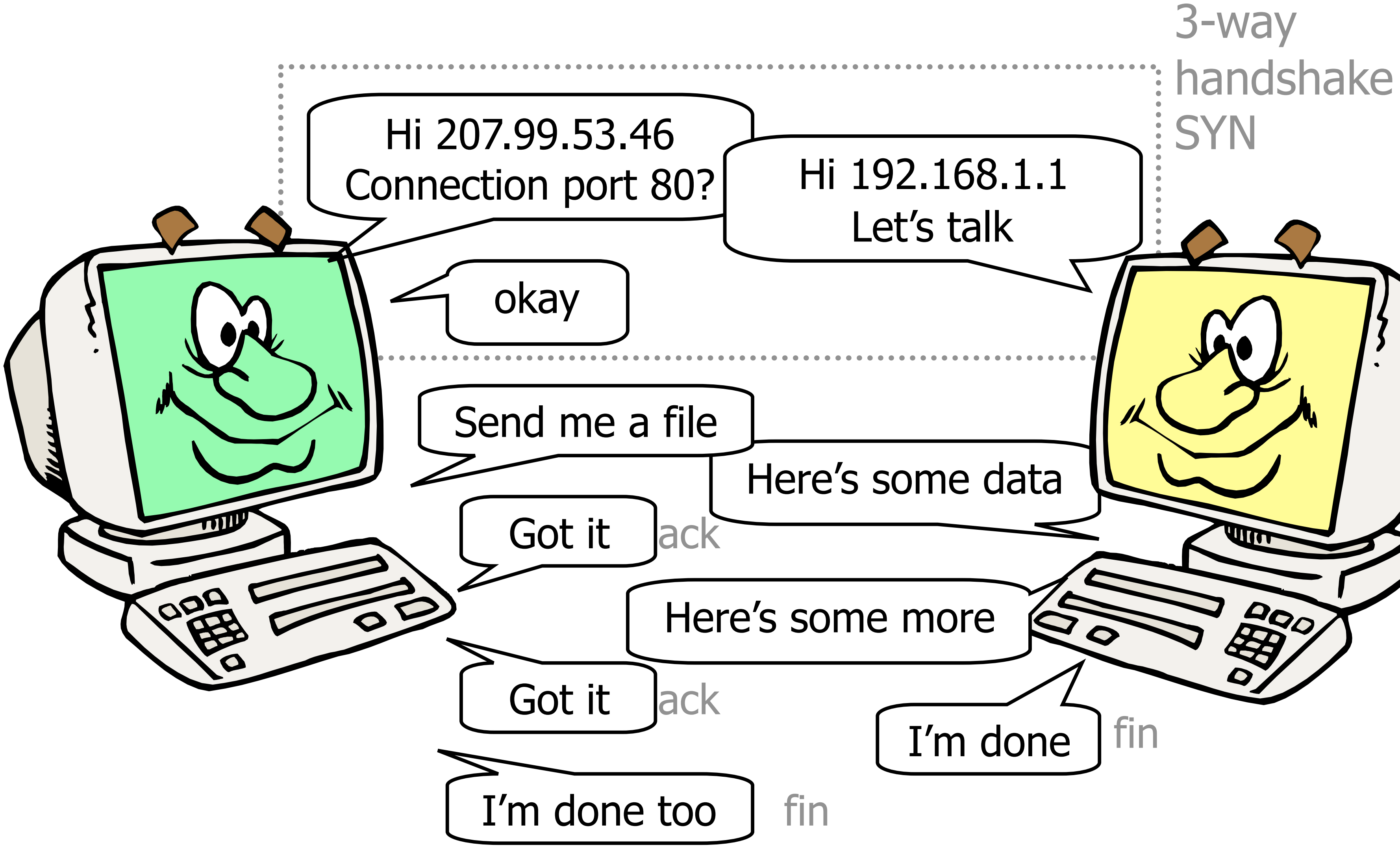
bytes

ack

port

data

# TCP Connection

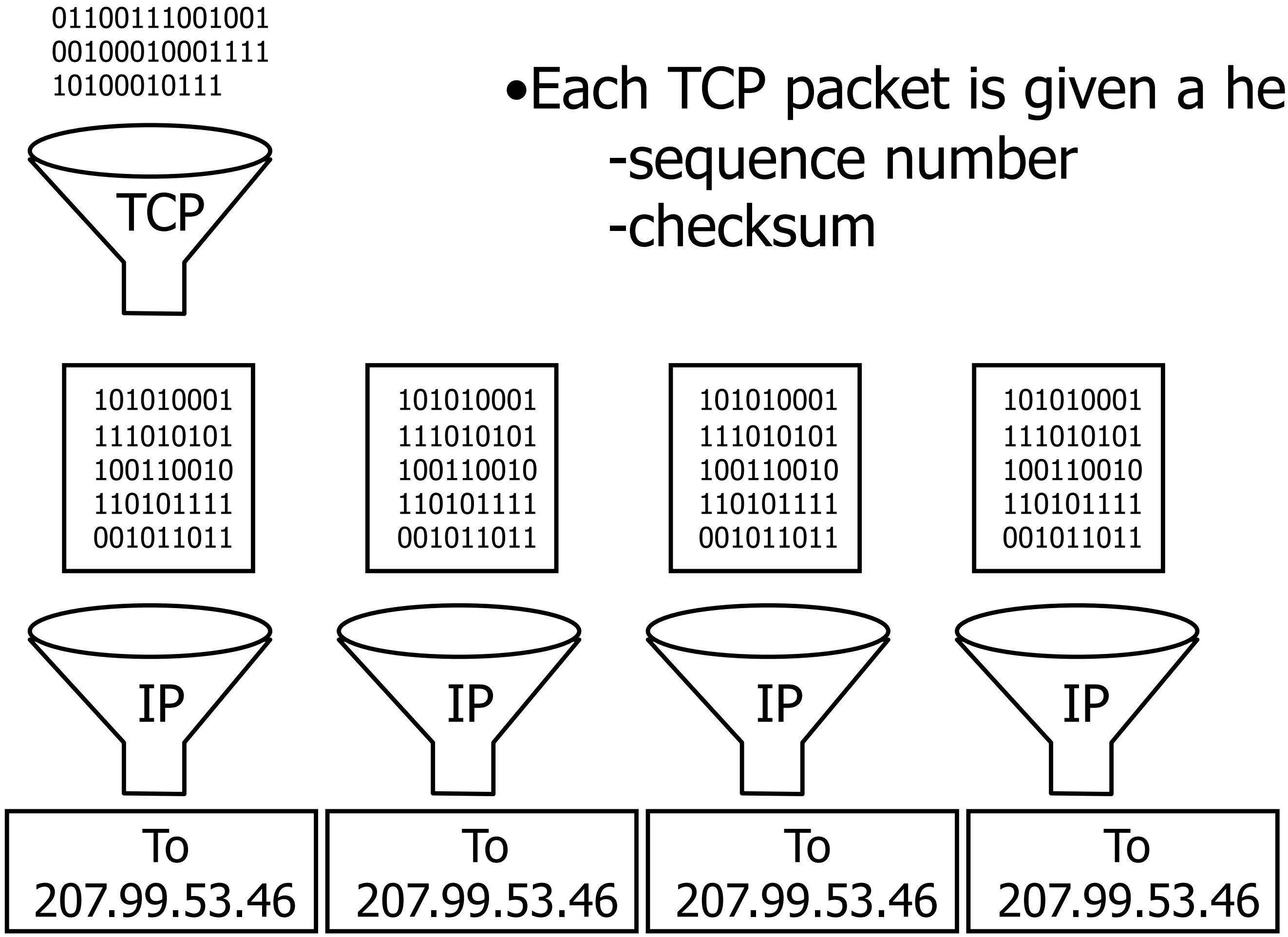


# Packaging up the data

- make packets

- Each TCP packet is given a header
  - -sequence number
  - -checksum

- put in an IP envelope with another header

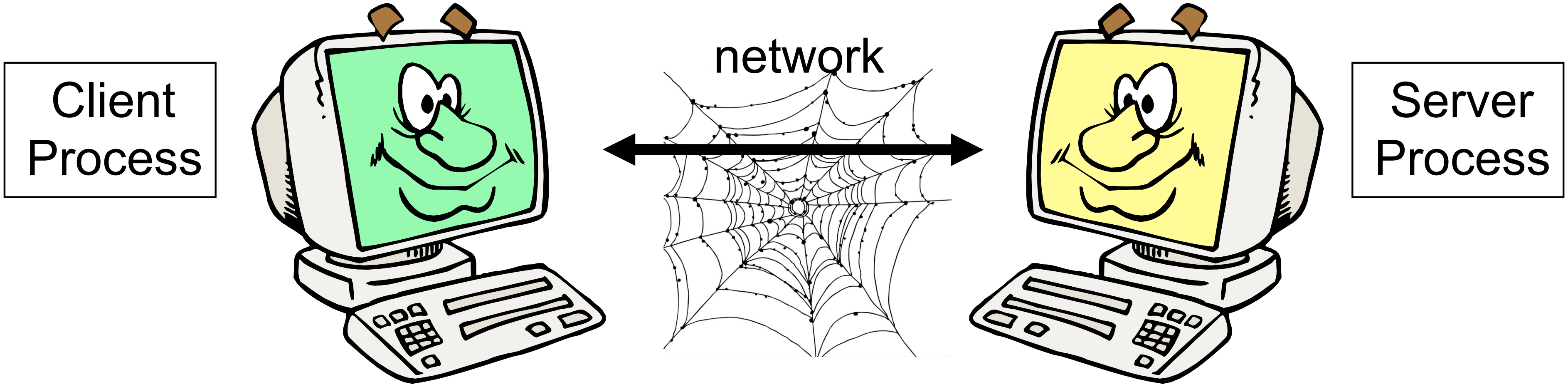


01100111001001
00100010001111
10100010111

TCP

| | | | |
|---|---|---|---|
| 101010001<br>111010101<br>100110010<br>110101111<br>001011011 | 101010001<br>111010101<br>100110010<br>110101111<br>001011011 | 101010001<br>111010101<br>100110010<br>110101111<br>001011011 | 101010001<br>111010101<br>100110010<br>110101111<br>001011011 |
| IP | IP | IP | IP |
| To<br>207.99.53.46 | To<br>207.99.53.46 | To<br>207.99.53.46 | To<br>207.99.53.46 |

# The Big Picture



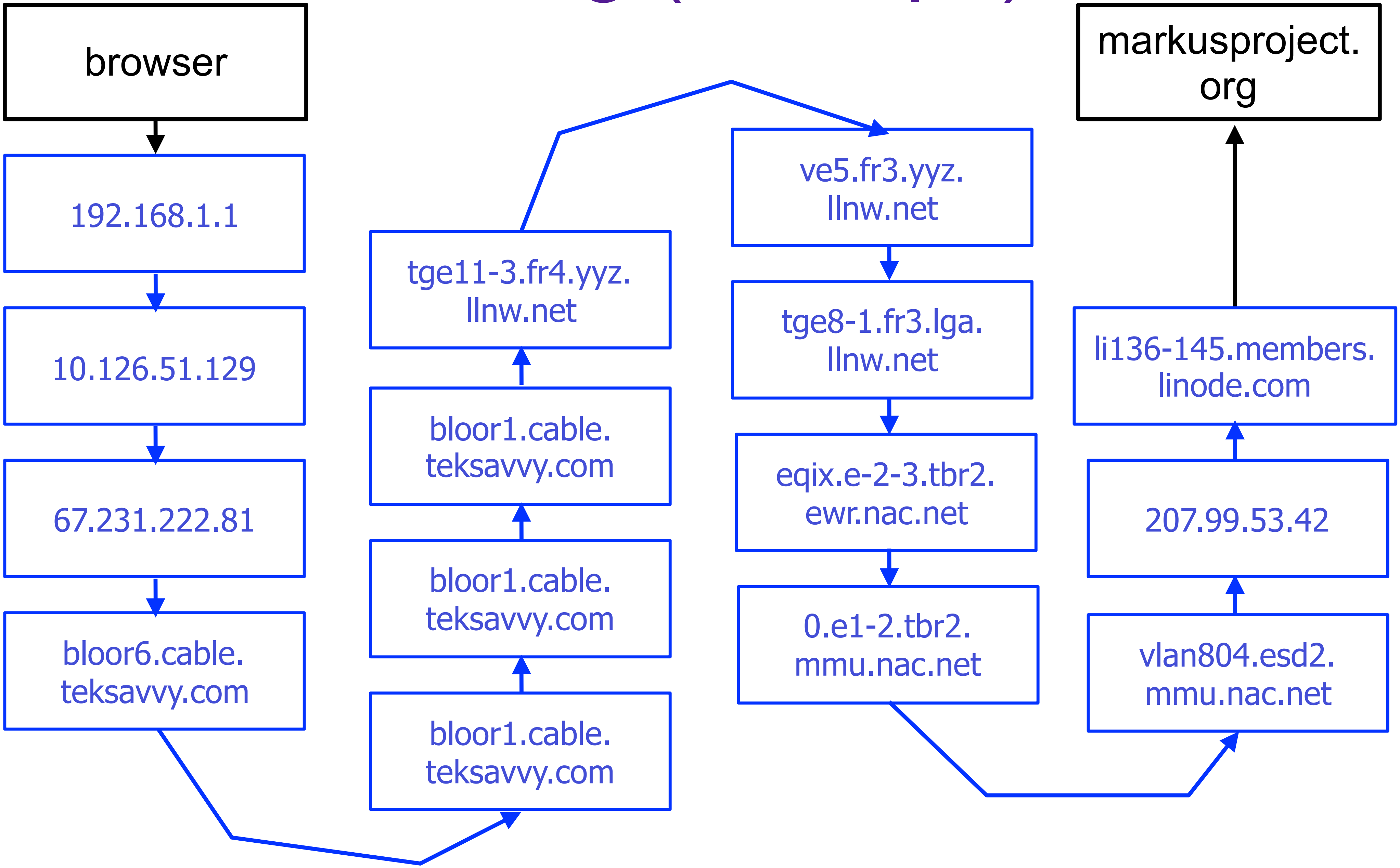
Client Process

network

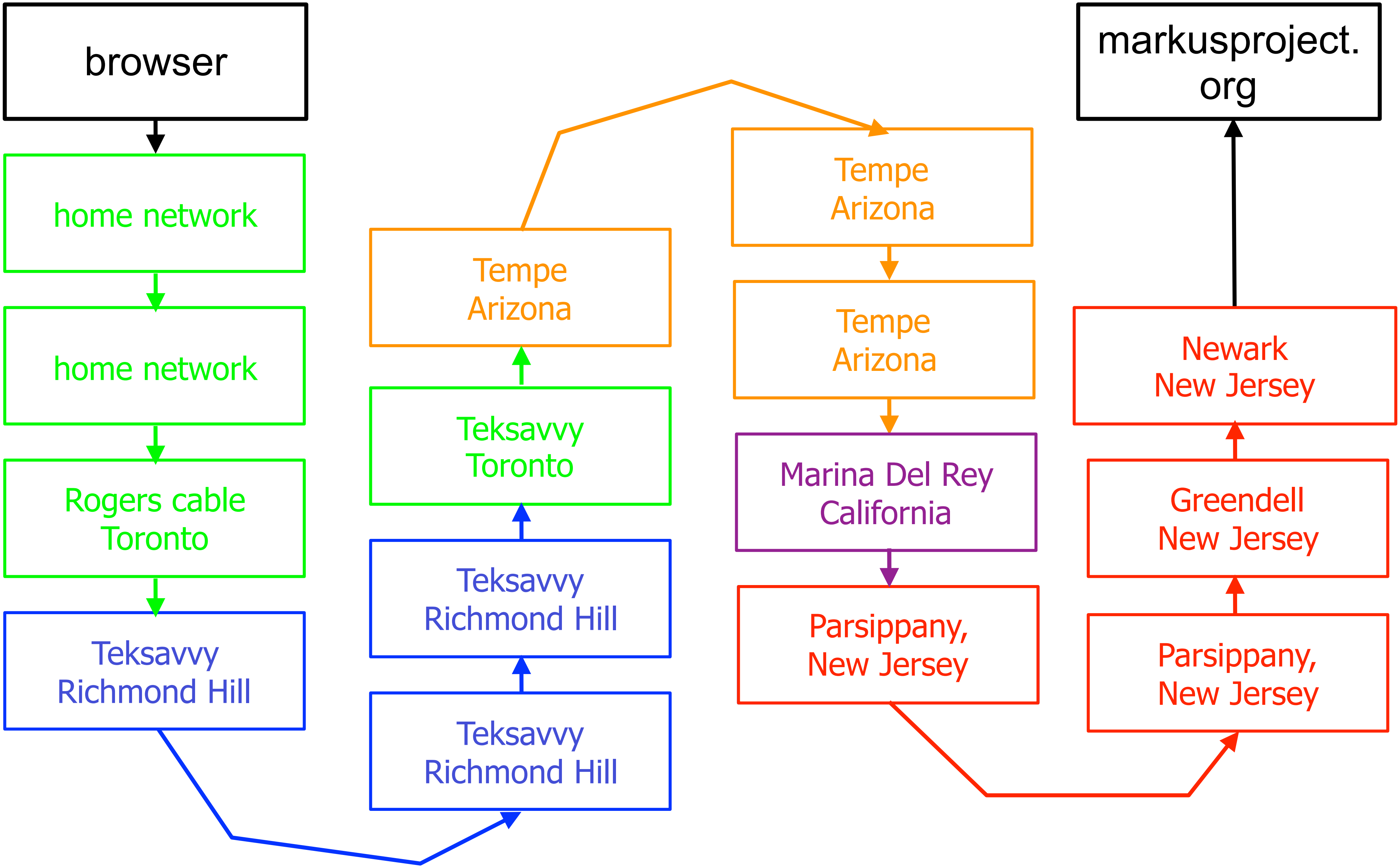
Server Process

- **Client-Server model**: a client process wants to talk to a server process
- Client must find server - **DNS lookup**
- Client must find process on server - **ports**
- Finally **establish a connection** so two processes can talk

# Routing (15 hops)



# 7 cities, 5 states/prov, 2 countries



browser

home network

home network

Rogers cable
Toronto

Teksavvy
Richmond Hill

Teksavvy
Richmond Hill

Teksavvy
Richmond Hill

Teksavvy
Toronto

Tempe
Arizona

Tempe
Arizona

Tempe
Arizona

Marina Del Rey
California

Parsippany,
New Jersey

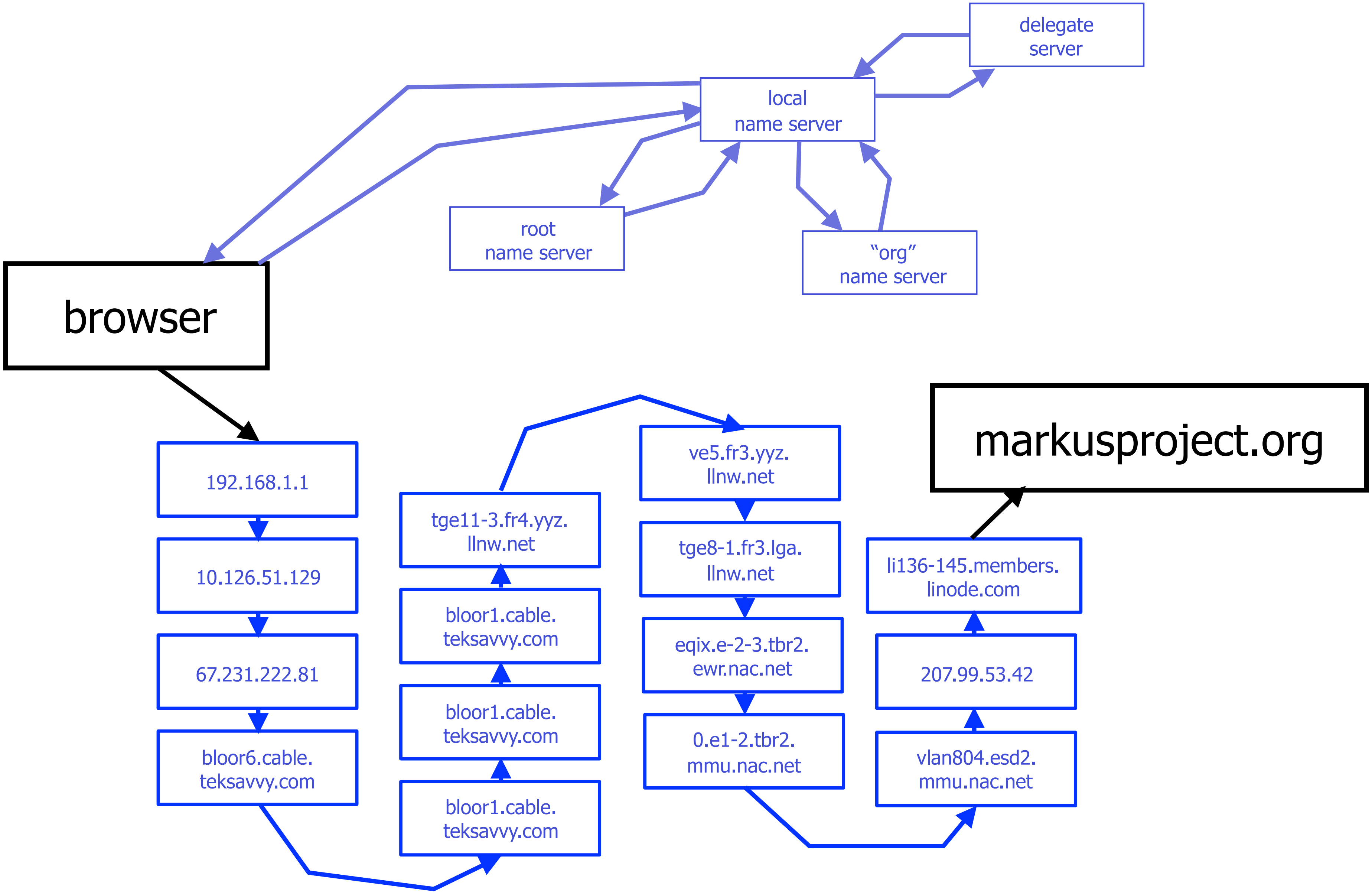
Parsippany,
New Jersey

Greendell
New Jersey

Newark
New Jersey

markusproject.
org

# Putting it together


delegate server
local name server
root name server
"org" name server
browser
markusproject.org
192.168.1.1
10.126.51.129
67.231.222.81
bloor6.cable. teksavvy.com
bloor1.cable. teksavvy.com
bloor1.cable. teksavvy.com
bloor1.cable. teksavvy.com
tge11-3.fr4.yyz. llnw.net
ve5.fr3.yyz. llnw.net
tge8-1.fr3.lga. llnw.net
eqix.e-2-3.tbr2. ewr.nac.net
0.e1-2.tbr2. mmu.nac.net
vlan804.esd2. mmu.nac.net
207.99.53.42
li136-145.members. linode.com

# How many messages?

- It depends on the size of the web page
- The web page that appears for markusproject.org is less than 30 Kbytes
- If the web page is 30 Kbytes (small!) it will likely be broken up into ~20 IP packets.

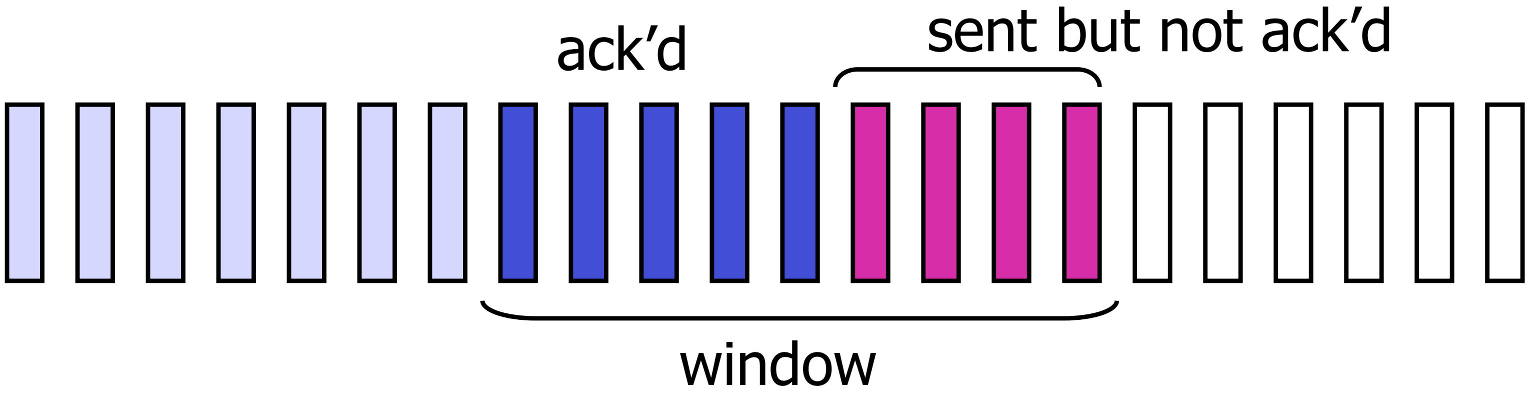
8 (DNS) + 20 * 15 hops
= 308 messages

# When something goes wrong

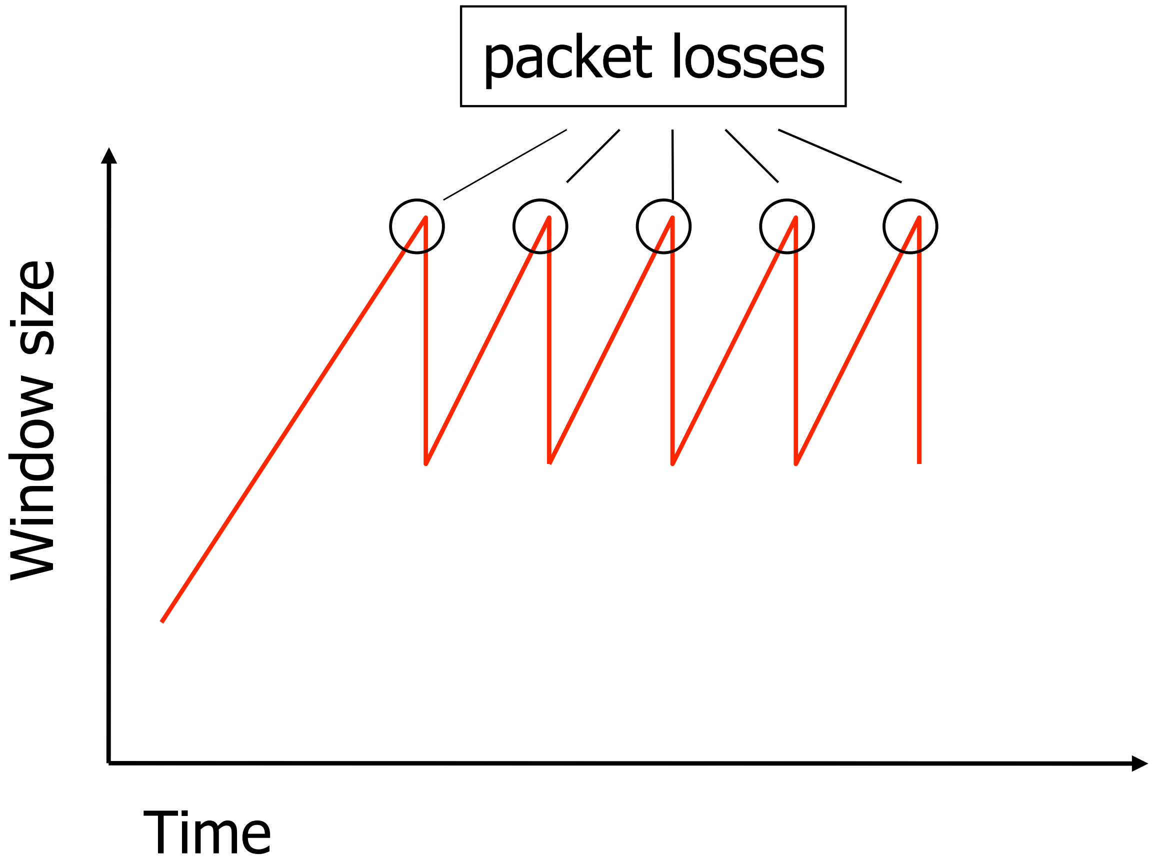
- A packet might not arrive
  - traffic overload
  - bit corruption
- Receiver asks for missing packets to be resent.
- Want to send data as fast as possible.
- But sending too fast wastes resources.

# TCP Congestion Control

- Window-based:
  - some number of packets allowed to be sent and not ack'd
  - as successful ack's arrive, grow window
  - if packet loss is detected, cut window size



# TCP Congestion Control



All we did was click on a link...

# Take aways

- The web today is made up of complex layers of software
- No one person, organization, or company could have created it in isolation
- We can understand it because we can study one layer at a time
- We can create new things by building on top of existing layers