
CS199-6: Wide Area Application Design, Deployment, and Management

<http://cs199.planet-lab.org/>

David Culler
Brent Chun
Timothy Roscoe

Wednesday 22nd January 2003



PLANETLAB

Overview of today's talk

- Quick summary of yesterday
 - Planetary-scale services
 - PlanetLab
 - This class
- The current state of PlanetLab
- The near future of PlanetLab
- Assignment for next week



Planetary-scale and wide-area distributed systems

What is a distributed system?

- Distinct components running on distinct machines
 - WWW, NFS, CIFS, Email, Ultima Online, Quake3, Saber, SS7, etc., etc.
- Characterized by:
 - Concurrency
 - Partial failures
 - Latency
- Writing distributed systems is *hard*
 - C.f. Waldo et.al.

Distributed systems: concurrency

- Concurrency can often be dodged in centralized systems
 - Event-driven systems, one-offs
- Alternatively, locks are available
 - E.g. Java concurrency primitives
- Distributed systems are inherently concurrent
 - And shared-memory-based synchronization is not an option

Distributed systems: latency

- Typical procedure call: $\sim 1\text{ms}$ vs. $\sim 10\text{ns}$.
- High-level system design must take this into account
 - Pipelining
 - Parallelism
 - Etc.

Distributed systems: partial failure

- "A distributed system is in which I can't get my work done because I computer I've never heard of has failed"
 - Butler Lampson
- Dist. Systems are not fail-stop
 - Bits keep running
 - Failures may be undetected
 - Etc.

Distributed Systems

- Despite all this, distributed systems are, these days, relatively commonplace
 - Some are well-engineered
 - » e.g. SS7, Ultima, etc.
 - Some are sufficiently simple
 - » e.g. WWW
 - Some people just live with
 - » e.g. WWW, NFS, CIFS
 - Some people are told to just live with
 - » e.g. most corporate calendaring systems

Wide-Area (or Planetary-Scale) Systems

- Wide area applications are for people who find ordinary distributed applications too easy :-)
- Wide-area applications span a significant portion of the globe
 - *Google is not* a planetary-scale system
 - *Akamai is* a planetary-scale system

Why build planetary-scale systems?

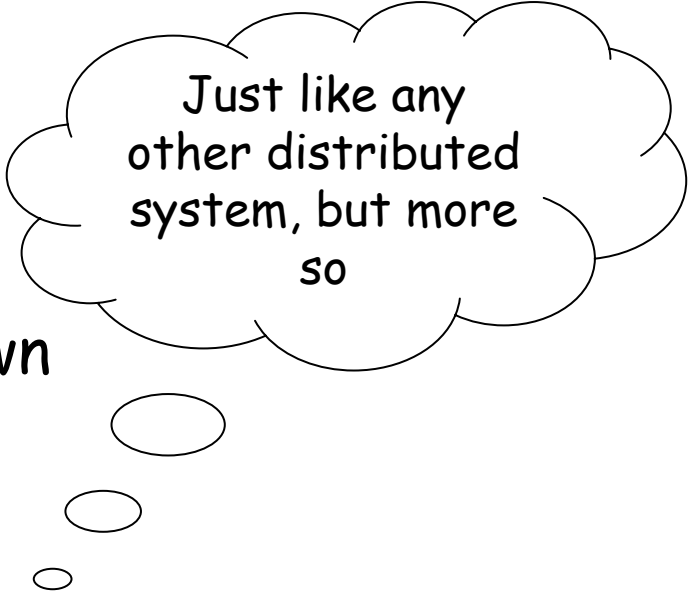
- Latency
 - Beating the speed of light
 - Move computation and data closer to users
- Multilateration
 - Stand in 1000s of viewpoints at the same time
 - Triangulation, correlation, measurement
- Politics
 - Spanning boundaries
 - Selecting (or avoiding) domains
 - judicial, financial, administrative, national, etc

Examples of wide-area systems

- Content-distribution networks
 - Akamai, Inktomi, etc.
- Overlay routing networks
 - RON (Resilient Overlay Networks), etc.
- Global storage systems
 - OceanStore, PAST, etc.
- True Peer-to-peer systems
 - FreeNet, KaZaA, etc.

What's hard about these systems?

- Scalability
 - > 100,000s users
- Reliability
 - System should never go down
- Performance
 - It shouldn't suck
- Management
 - How does something this big stay manageable?



Just like any other distributed system, but more so

What's hard (and new) about these?

- Heterogeneity

- Lots of different machines, and different components which have to talk to each other

- Security

- We're now spanning organizational boundaries
- Perimeter-based security doesn't really work

- Evolution

- Parts of the system must change incrementally over time
- We can't just restart everything.



Wide-area application research

- Lots of recent research work:
 - RON, ESM...
 - Storage: Oceanstore, IBP, CFS, Past...
 - DHTs: Tapestry, Chord, CAN, Pastry...
 - Event systems: Scribe, Herald, Bayeux...
 - CDNs
- Results tend to be based on:
 - Simulation
 - Emulation (clusters, etc.)
 - Small-scale deployment (call your friends)



PlanetLab: What and Why?

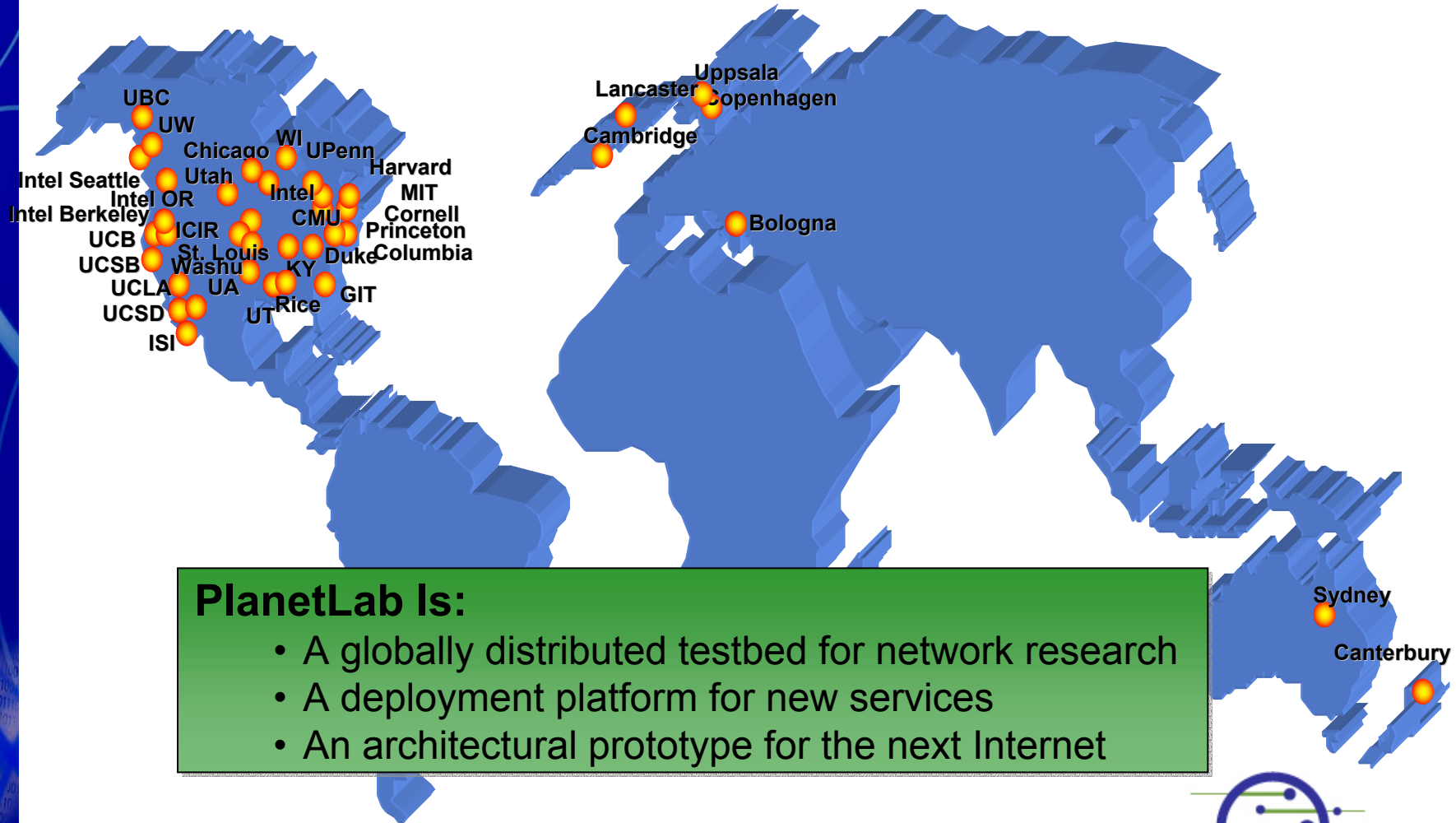
Doing all this for real...

- ... is hard for a researcher
- Where do you get access to 1000 geographically dispersed machines?
- How do you do it legitimately?
 - No worms
 - No cracking
 - No Venture Capital

So what is PlanetLab?

- An open, shared testbed for developing, deploying, and accessing planetary scale services
- <http://www.planet-lab.org>
- Boils down to:
 - A set of machines to run your code on all over the world
 - An operating system to make this safe
 - Management software to keep it working
 - Useful services to save you time and effort

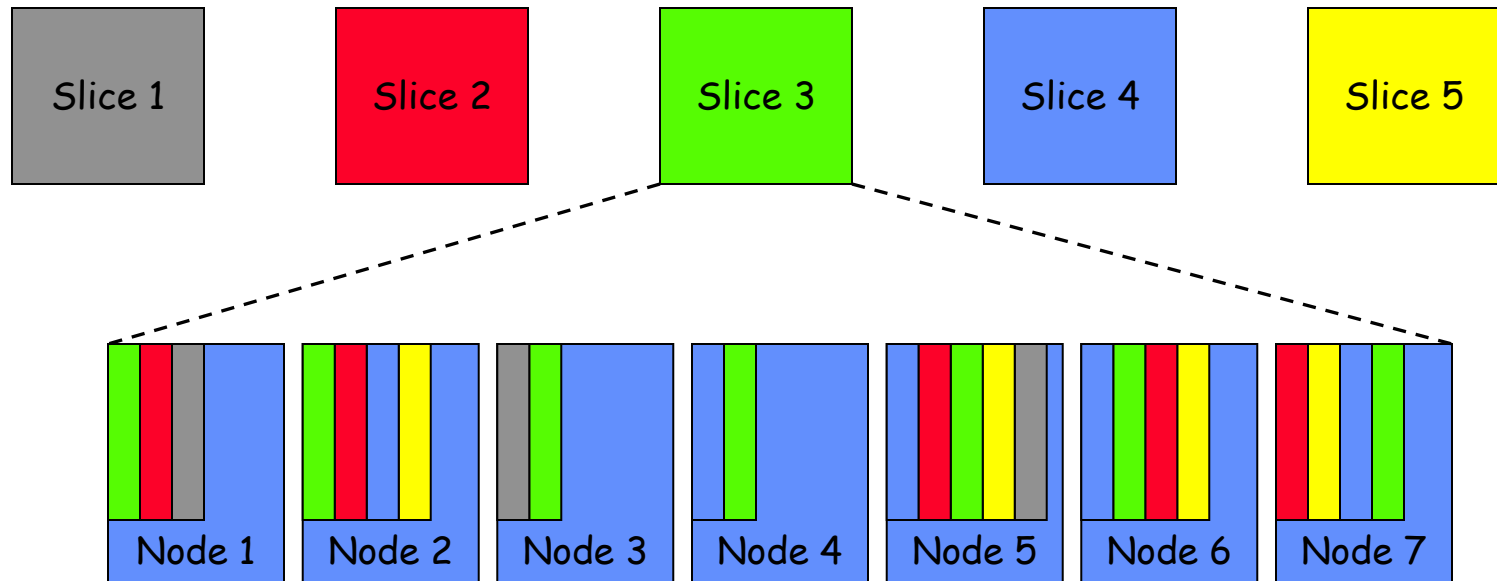
PlanetLab



PlanetLab Is:

- A globally distributed testbed for network research
- A deployment platform for new services
- An architectural prototype for the next Internet

PlanetLab Slices/Slivers



A network service is broken into components that can be distributed throughout the Internet

- Slice: total resources for the service
- Sliver: resources required on a specific node

What does PlanetLab give you?

- PlanetLab gives you (yes you!)
 - Ability to deploy a service around the world
 - Chance to contribute to the research
- PlanetLab is under development
 - Management services
 - Naming and location
 - Etc.
- PlanetLab is shared
 - Responsible use is called for!

How do I get onto PlanetLab?

- Brent will send out details :-)
 1. Make sure you have an ssh key pair
 - Find out what this is if you don't know
 2. Register at <http://www.planet-lab.org/>
 3. See the course page for the list of 8 nodes for use by the class
 4. Upload your public key to the web site
 5. Use ssh/scp for software distribution and control
 6. See sf.net/planetlab for possibly useful tools
 7. Note: it looks like Linux.



The Administrative Stuff

What's this course about?

- Writing distributed services to run in the wide area Internet
 - spread over a large no. of machines and large geographical area
 - which can be deployed over PlanetLab
 - which might become part of PlanetLab
- Experience in the design of large systems
 - Network programming
 - Handling failures
 - etc.
- Introduction to real systems research
- Emphasis on *building* rather than lectures

Approach

- Programming experience is assumed
- Few introductory lectures
- Reading material
- Get into design and implementation as soon as possible
- Work in teams of 2-3 people

Tentative Schedule

Week 1:

- Introductory lectures
- Initial team assignment

Week 2:

- Project discussion
- Form teams for main project

Week 3:

- Review proposals

Thereafter:

- Implementation and Team meetings
- Project Milestones



Milestones

- 28th January: Initial assignment due
- 4th February: Team project proposals due
- 24th February: Initial prototype due
- 17th March: Enhancements to prototype
- 21st April: Deliver final service
- 27th April, 6th May: Presentations/demos
- 16th May: Project writeups due.

Reading material

- See:

<http://cs199.planet-lab.org/reading.html>

- Required reading for next week:

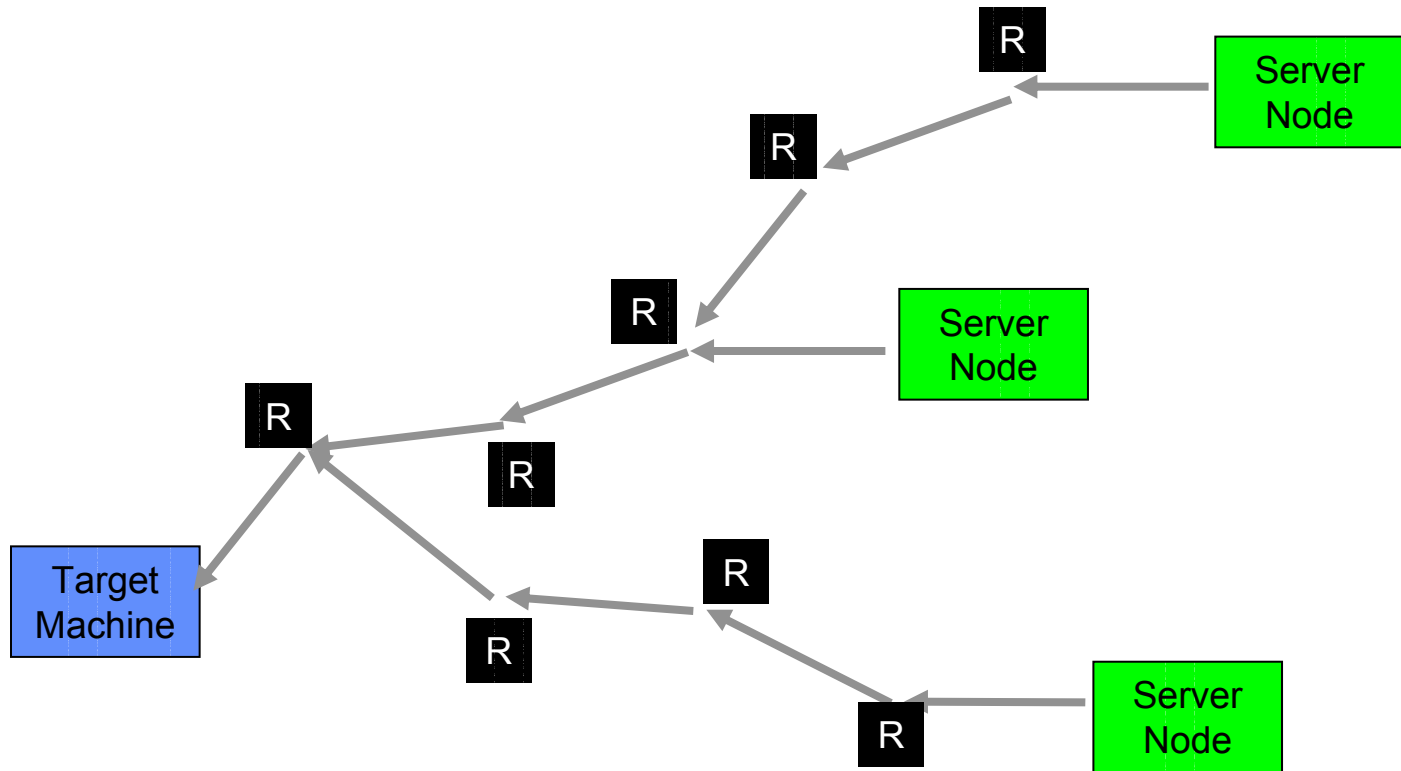
- "A Blueprint for Introducing Disruptive Technology into the Internet"
- "A Note on Distributed Computing"
- "Hints for Computer System Design"

**Initial assignment (for next week):
Parallel traceroute service**

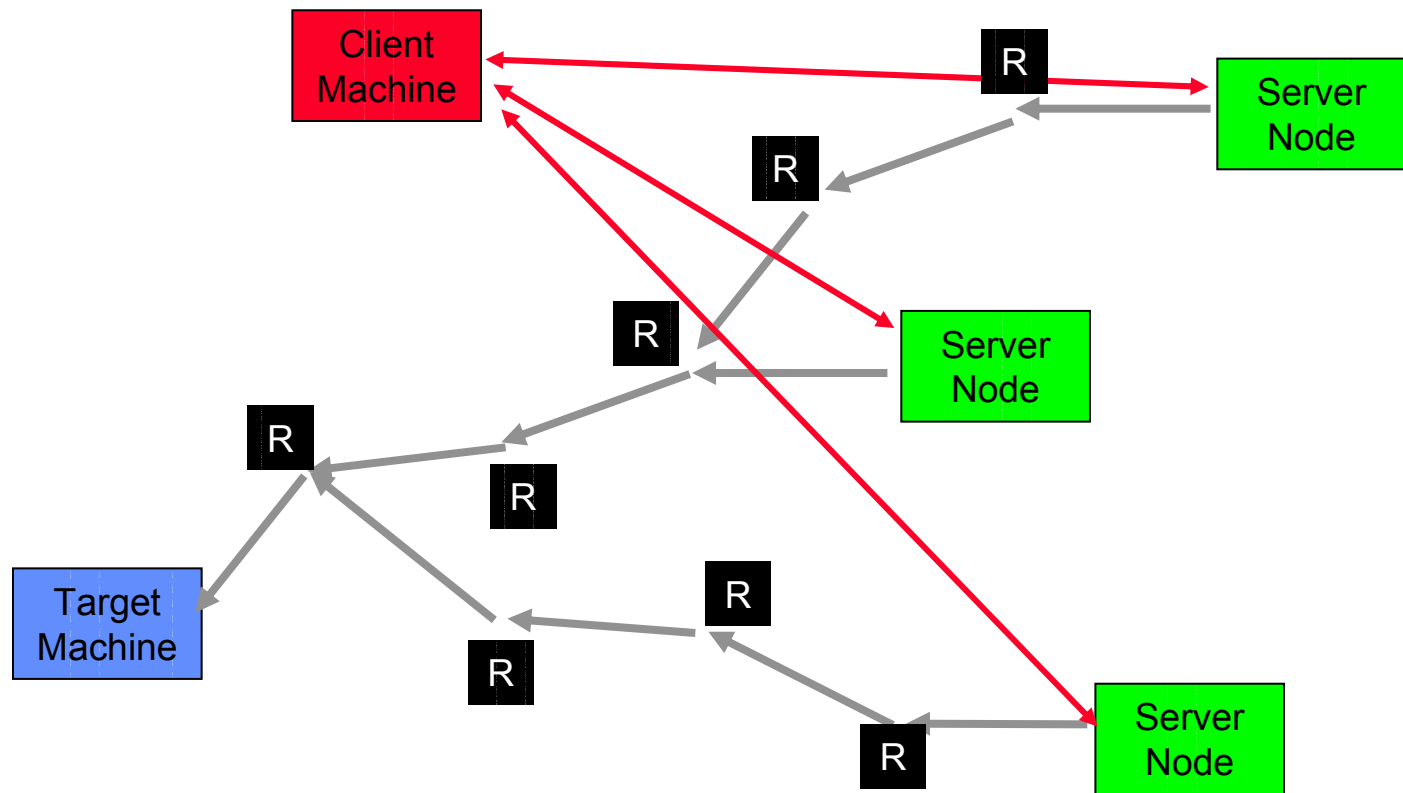
Initial assignment: goals

- Write a simple distributed service
 - And work out what's hard, easy, fast, slow...
- Get some experience with PlanetLab
 - See Brent for accounts and details
- Sneak a peek at the structure of the Internet
 - Service can be used for simple network maps

Basic Concept: mapping the network



Basic Concept: mapping the network



Using traceroute

```
deleuze> traceroute -n www.cam.ac.uk
traceroute to www.cam.ac.uk (131.111.8.46), 30 hops max, 38 byte
  packets
 1  10.212.2.1  2.365 ms  1.677 ms  0.483 ms
 2  12.155.161.129  9.423 ms  0.520 ms  9.433 ms
 3  12.124.44.29  9.984 ms  9.933 ms  0.512 ms
 4  12.123.12.206  11.506 ms  7.959 ms  9.938 ms
 5  12.122.11.93  2.379 ms  17.595 ms  9.957 ms
 6  12.123.13.194  10.009 ms  9.974 ms  9.942 ms
 7  209.0.227.29  10.019 ms  0.561 ms  9.664 ms
 8  209.244.3.137  10.405 ms  9.174 ms  9.976 ms
 9  64.159.1.73  9.914 ms  9.914 ms  9.914 ms
10  64.159.1.86  79.918 ms  85.407 ms  79.761 ms
11  212.187.128.137  153.521 ms  150.774 ms  149.966 ms
12  212.187.128.50  151.452 ms  158.274 ms  149.879 ms
13  212.113.3.2  149.990 ms  159.802 ms  149.915 ms
14  195.50.116.206  159.740 ms  149.882 ms  149.875 ms
15  146.97.37.85  159.900 ms  150.626 ms  149.156 ms
16  146.97.33.9  159.945 ms  161.207 ms  148.500 ms
17  146.97.35.10  159.868 ms  159.821 ms  159.935 ms
18  146.97.40.50  159.901 ms  149.872 ms  160.033 ms
19  192.153.213.194  159.738 ms  159.845 ms  159.961 ms
20  131.111.8.74  159.871 ms  159.853 ms  159.927 ms
deleuze>
```



Deliverables

- Server:

```
$ traceroutesvr -p <port>
```

- Client:

```
$ tracerouteclnt -p <port> <target>
```

```
<server1> <ip1> <ip2> <ip3> ... <host>
```

```
<server2> <ip1> <ip2> <ip3> ... <host>
```

```
...
```

```
<server8> <ip1> <ip2> <ip3> ... <host>
```

- See web site today or tomorrow for more detail on the output



Extra Kudos: tree display

- Client:

```
$ tracerouteclnt -p <port> <target>
<host>
  <ip1a>
    <ip2a> ...
  <ip1b>
    <ip2a> ...
    <ip2b> ...
  ...
```

Extra Kudos #2: performance

- Why is it slow (if it is!)?
 - How many requests does the client have on the go?
 - How many requests does the server have on the go?
- What happens if a server fails?
- What happens if traceroute fails?
 - E.g. '*'s in the output.

Other details:

- Anyone should be able to run your `traceroute.c` from their workstation.
- Pick your favourite programming language (Java, C, C++, Python, etc.)
- Put up web page with a writeup, tarball (including source), and sample output.
- See Brent for details on how to get PlanetLab accounts

Questions?

- Any thoughts on how to approach this?
- If you have problems:
 - Send us email
 - Arrange to meet
 - Tell us how it's going