

CS260r: Topics in Computer Systems

Internet Scale Sensor Networking

Matt Welsh

Harvard University

Lecture 1: Course Introduction

September 19, 2006



Foundations: Sensor Networks



WeC (1999)



Rene (2000)



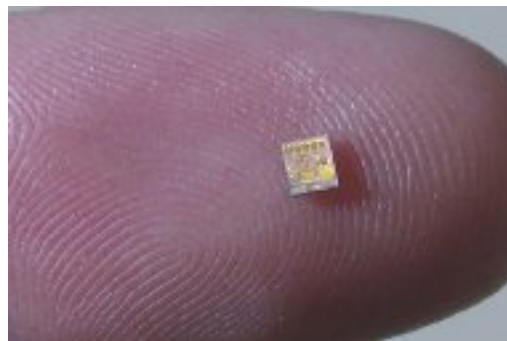
Dot (2001)

Integration of sensing, computation, and communication

- Low-power, wireless “motes” with tiny amount of CPU/memory
- Large federated networks for high-resolution sensing of environment



MICA (2002)



Speck (2003)



Telos (2004)

The Telos “Mote”



- TI MSP430 processor
- 128 KB code, 2 KB data SRAM
- 512 KB flash
- CC2420 radio (2.4 Ghz, 802.15.4)
- 250 kbps, 100 m range

Several thousand produced, used by 100s of research groups

Great platform for experimentation (though not particularly small)

- Easy to integrate new sensors & actuators
- 15-20 mA active (5-6 days on 2 AAs)
- 5 μ A sleeping (40+ years, but limited by shelf life of battery!)

Caveat

This is **not** a course about sensor networks!

We will touch on them, but mostly will ignore issues such as:

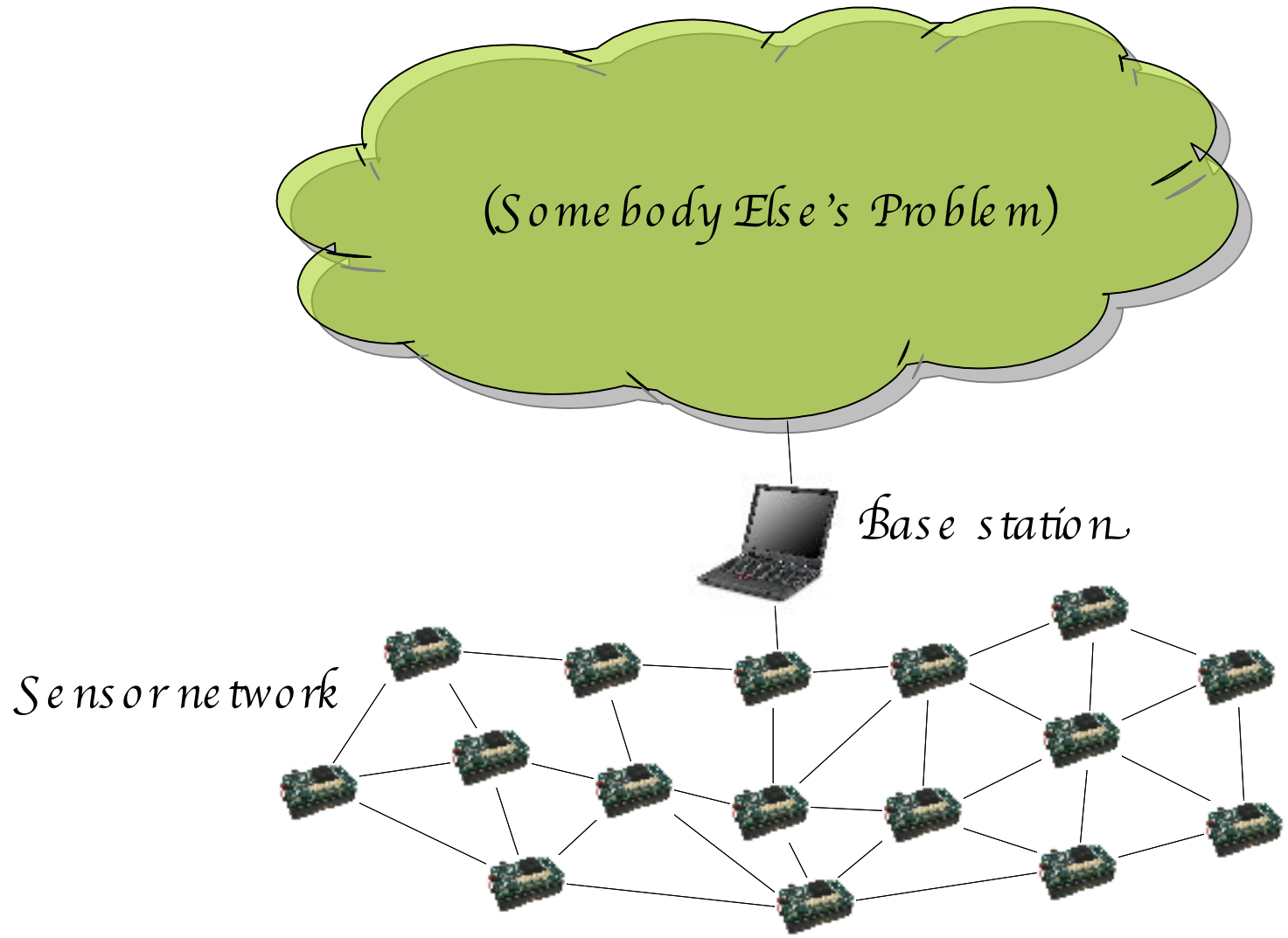
- Wireless communications
- Embedded programming
- Communication protocols, time synchronization, localization, etc.

There has been vast amount of research in sensor networks lately:

- How to make small, low power hardware platforms
- Embedded OS design and programming models
- MAC and routing protocols
- Time synchronization, localization, data aggregation, etc.

The Problem

Traditional view of sensor networks:

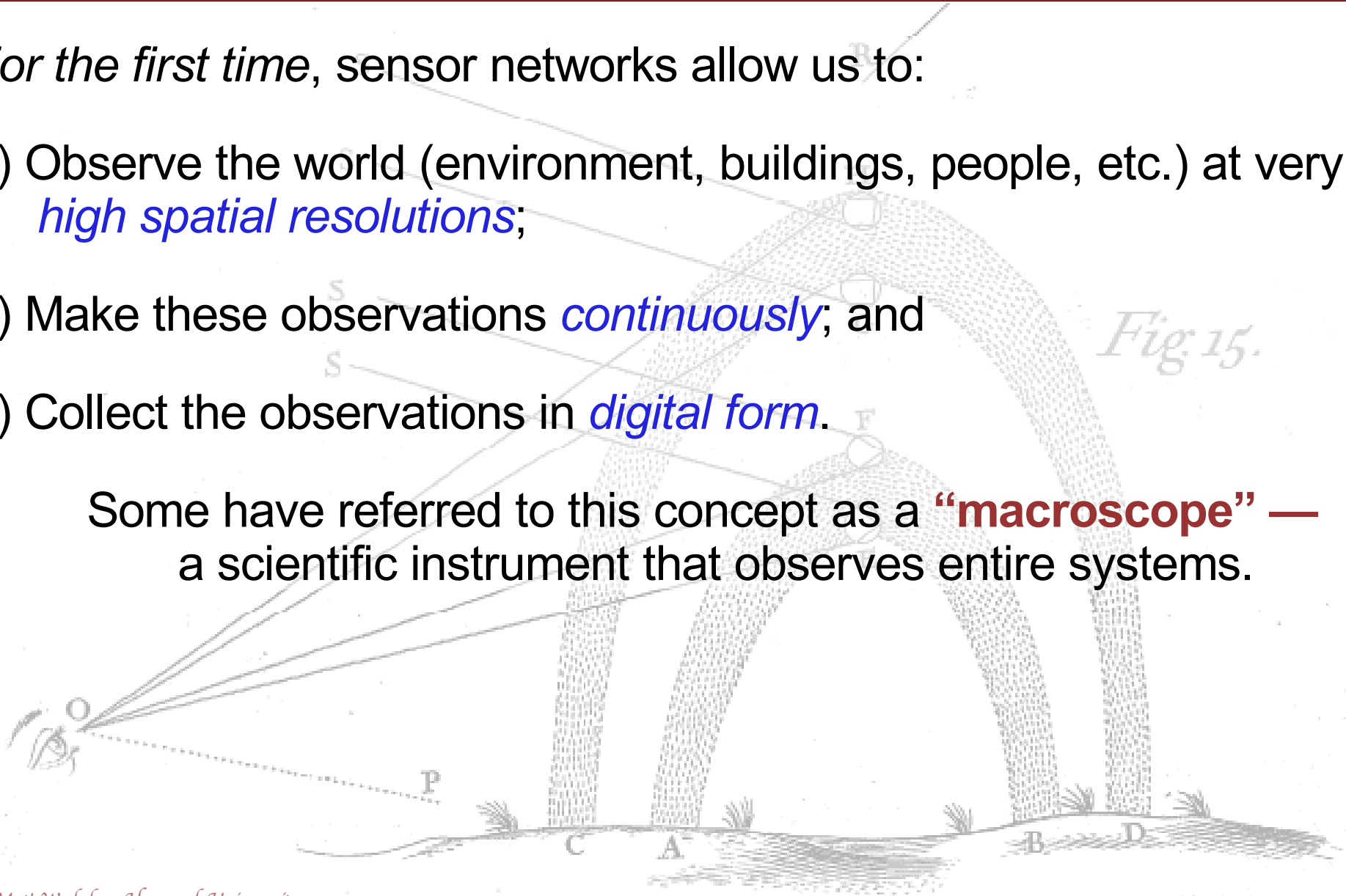


The Macroscope

For the first time, sensor networks allow us to:

- 1) Observe the world (environment, buildings, people, etc.) at very *high spatial resolutions*;
- 2) Make these observations *continuously*; and
- 3) Collect the observations in *digital form*.

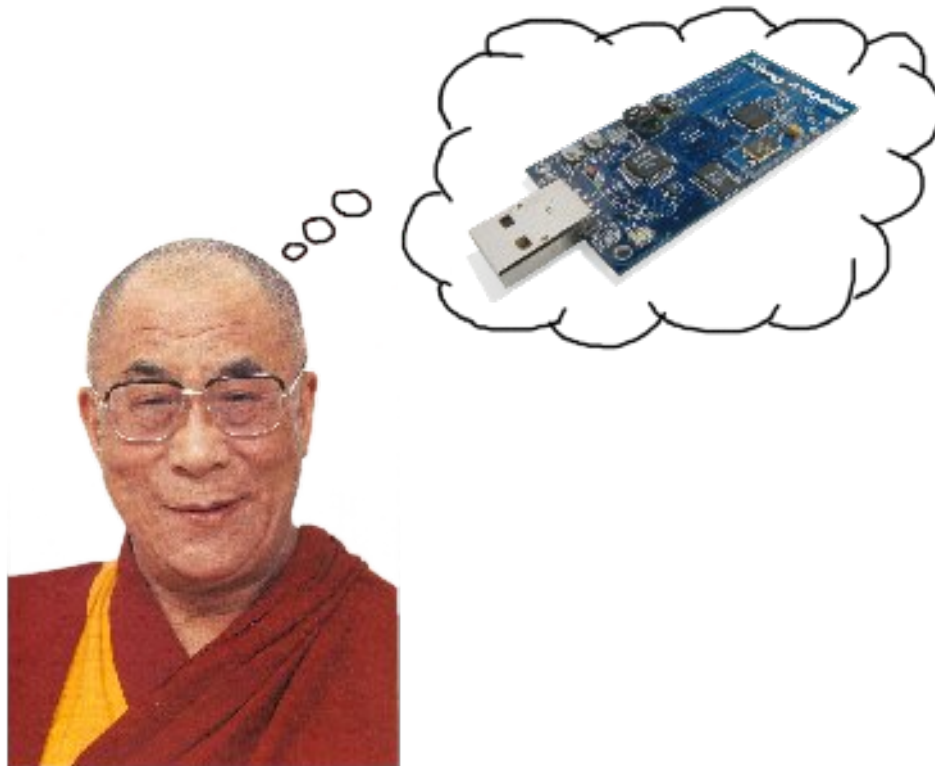
Some have referred to this concept as a “**macroscope**” — a scientific instrument that observes entire systems.



What this course is about

I want to ask the question:

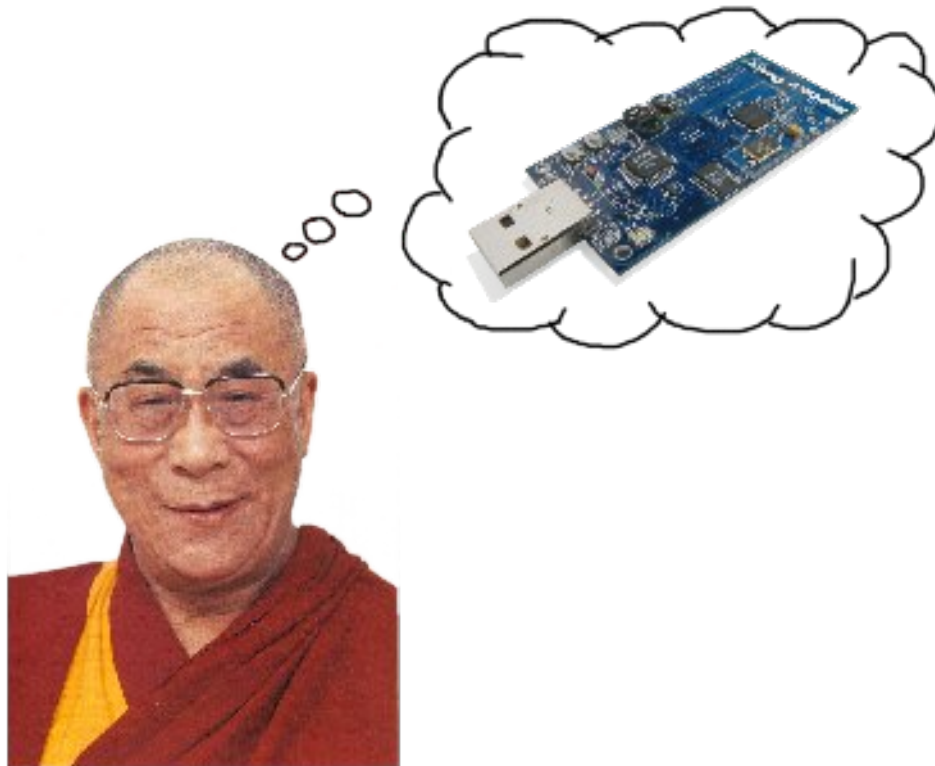
How can sensor networks *really* change the world?



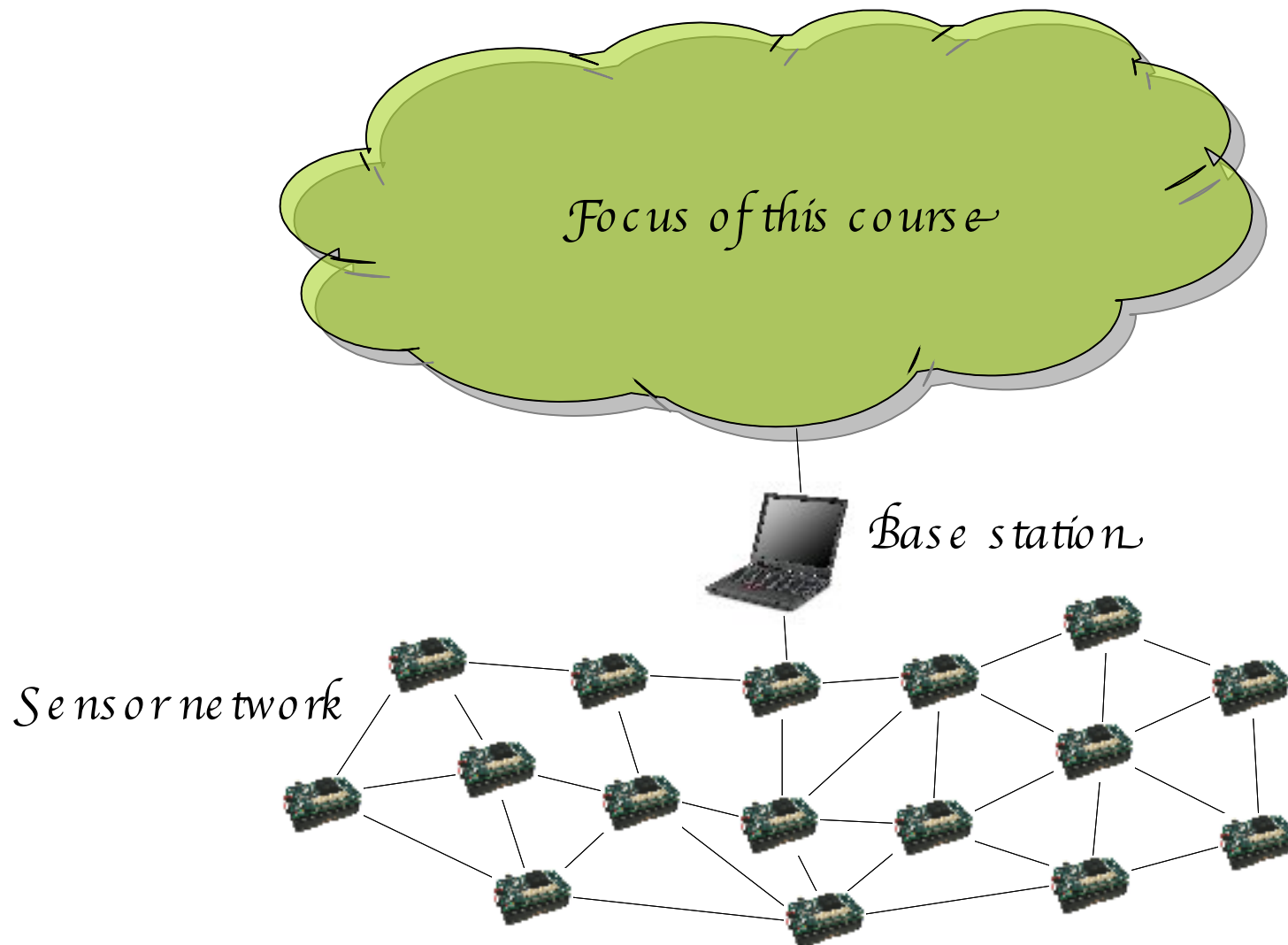
What this course is about

I want to ask the question:

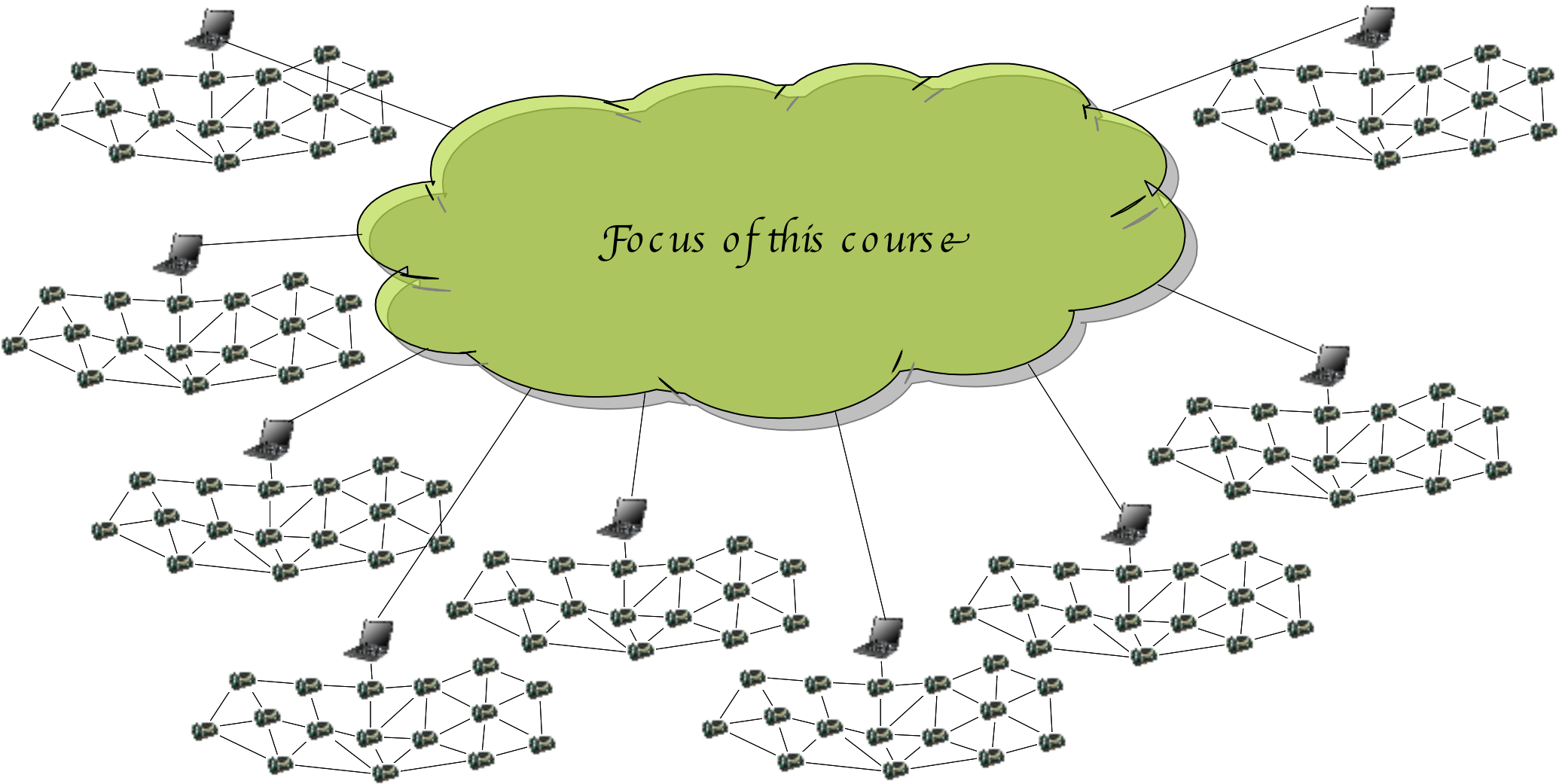
And what do we need to do to get there from here?



This course



This course



Intelligent Instrumentation

Sensor networks are not just passive instruments!

We can push processing and “intelligence” into the network.

Processing can happen at many levels:

- On individual sensor nodes.
- At aggregation points within the network.
- At the base station or gateway.

Sensor networks fundamentally change the notion of “scientific observation” from a *passive* process to an *active* one.

- This has a deep impact on many aspects of science.

Lots of exciting research topics here...

How do we expose sensor data to the Internet?

- HTTP? SOAP? RDF? Raw bytestreams?

How do we open up sensor networks for queries or reprogramming?

- Need mechanisms to manage access, protect privacy, support multiple applications

How do we discover and harness data from multiple sensor networks?

- “sensor.google.com” -- Now searching 8,162,375,201,799 motes!

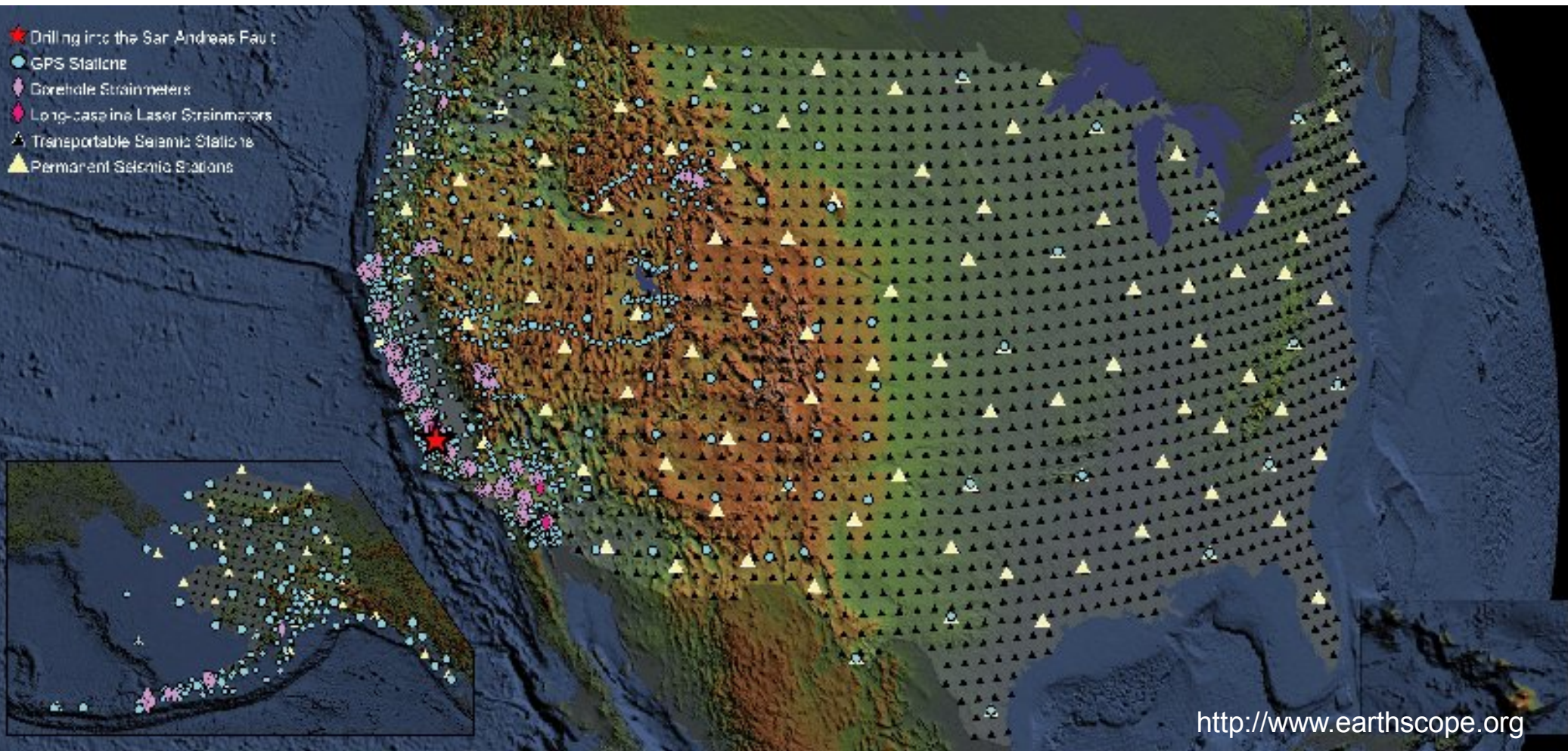
How do we distribute query processing across the Internet?

- Must handle vast numbers of data sources and simultaneous queries.

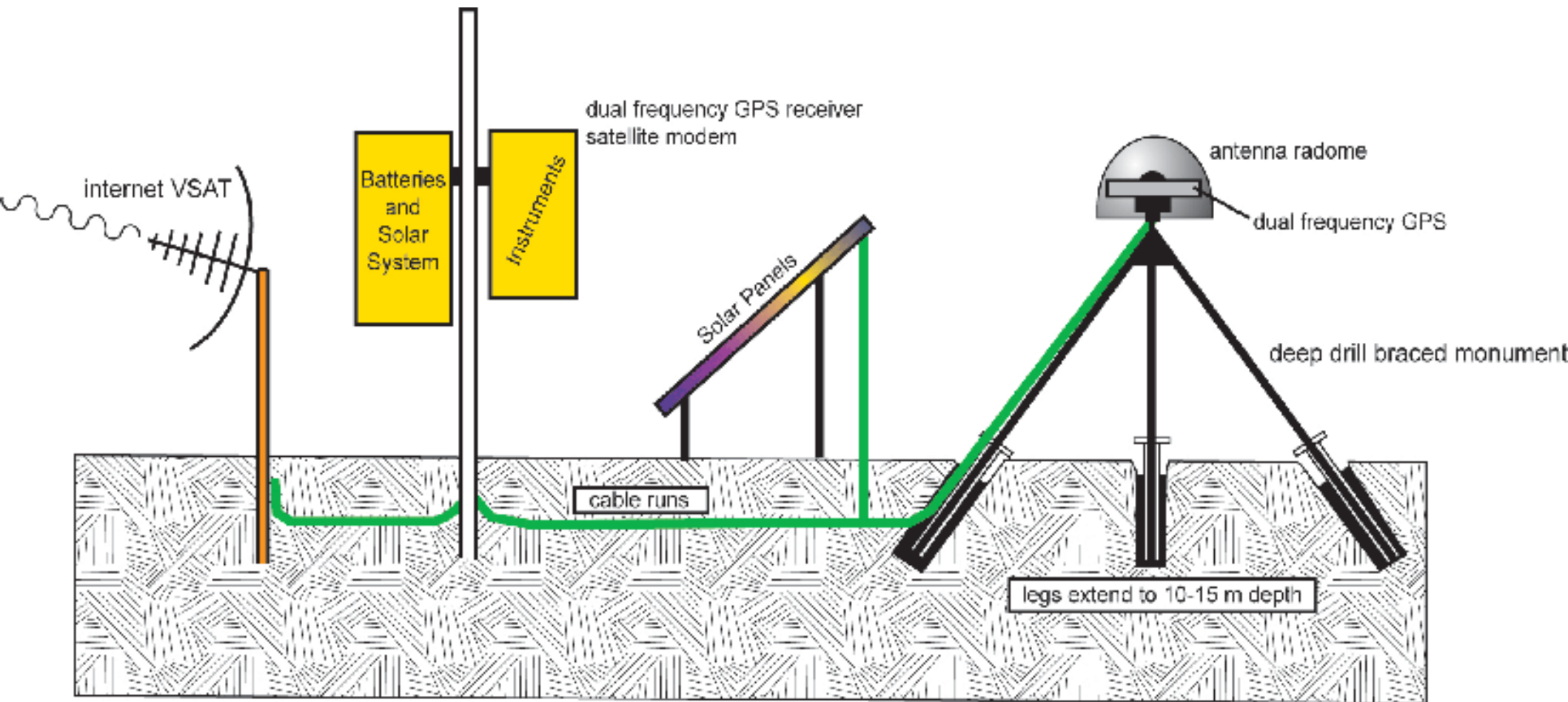
EarthScope (NSF)

15-year effort to understand earthquakes, volcanism, and plate movements in N. America

- 400 seismometers, 1000 GPS stations, 180 strainmeters



Earthscope GPS Station

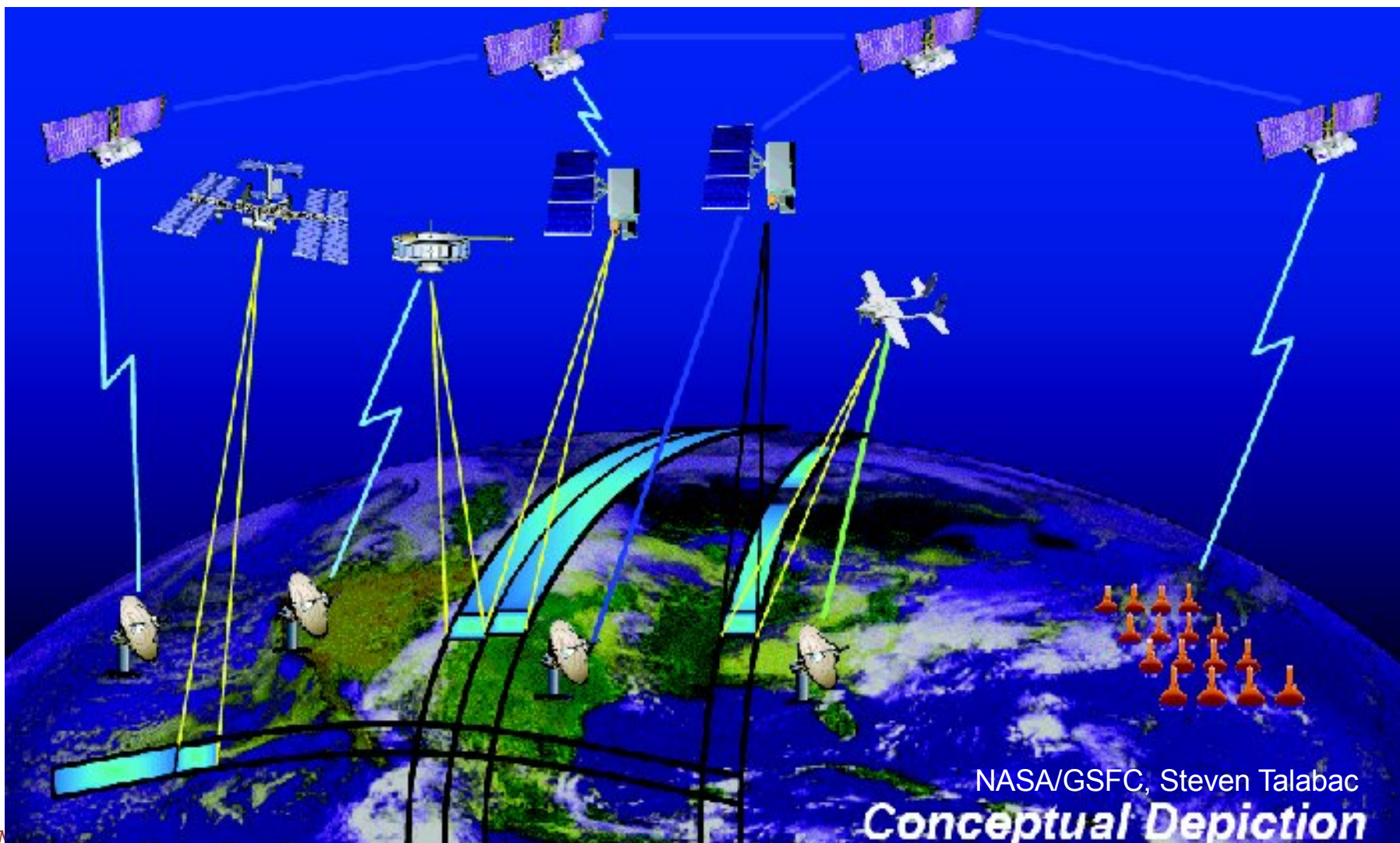


<http://www.earthscope.org>

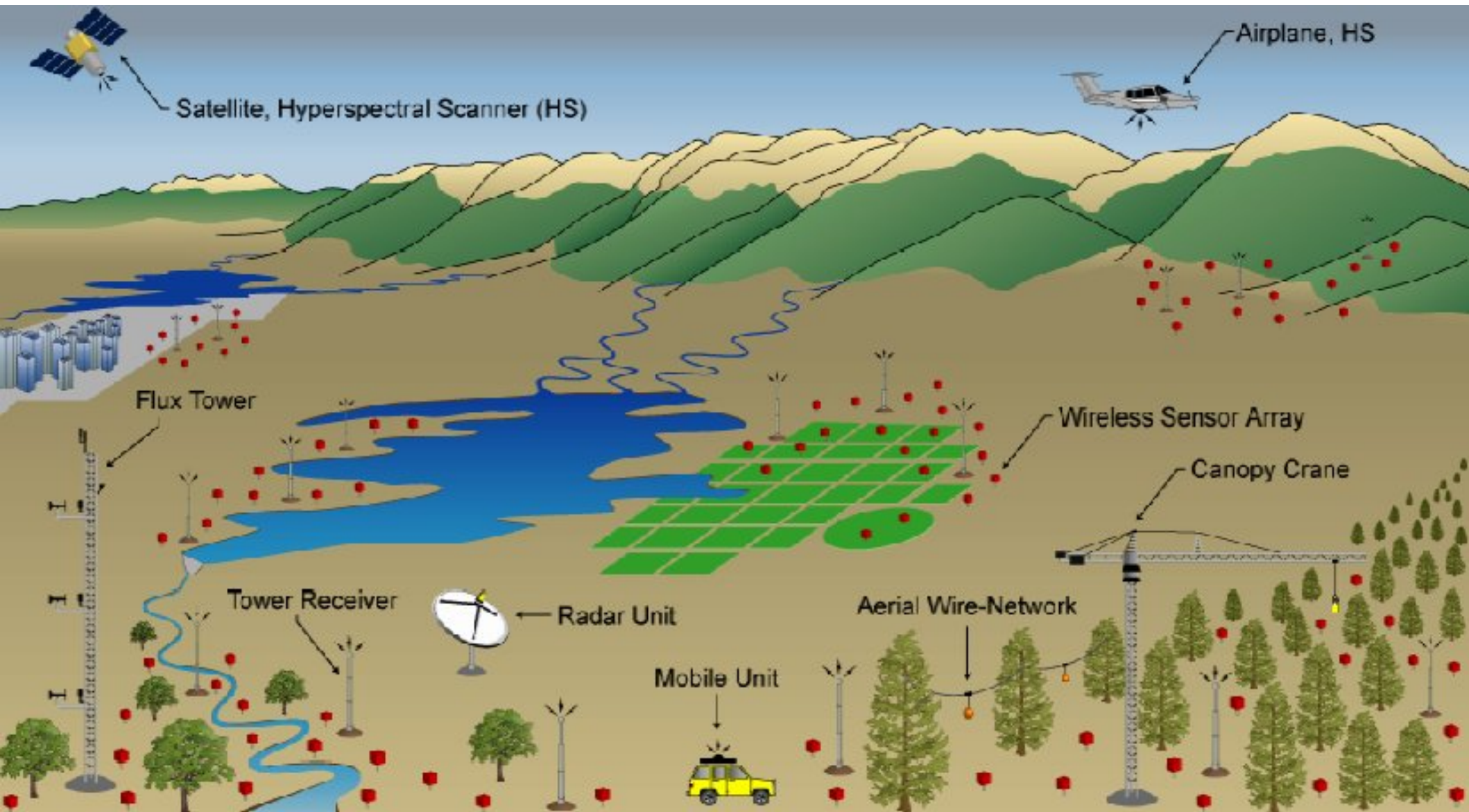
SensorWebs

NASA Goddard Space Flight Center

Adaptive, model-based sensing with ground and space-based sensors



NEON (National Ecological Observatory Network)



TinyDB

(Sam Madden, MIT)

Declarative SQL-like query interface

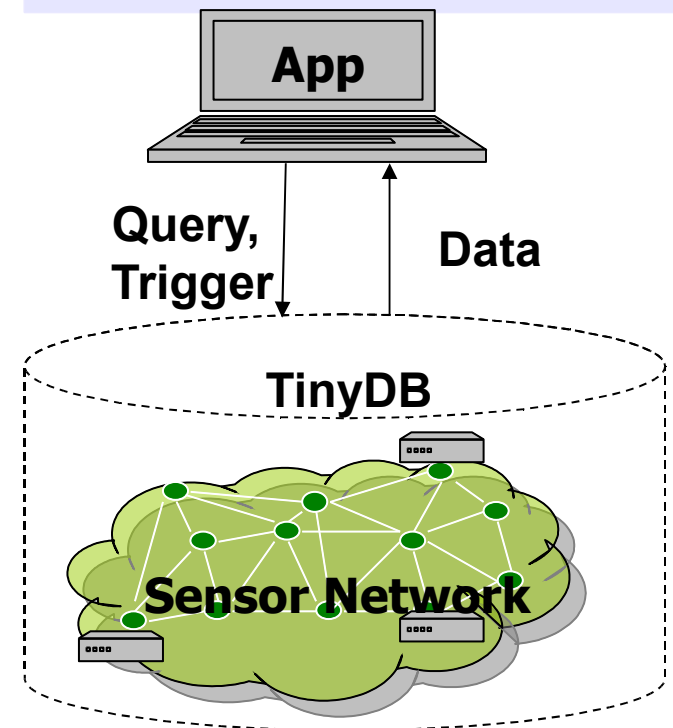
- Treat entire sensor network as a “table”
- Columns correspond to sensor attributes: sensor values, ID, location, etc.
- Each tuple represents a reading from one sensor

In-network aggregation

- Aggregates such as **average**, **max**, **min**, etc. computed within the network
- Nodes push data up a spanning tree to the base station, aggregation performed at intermediate hops

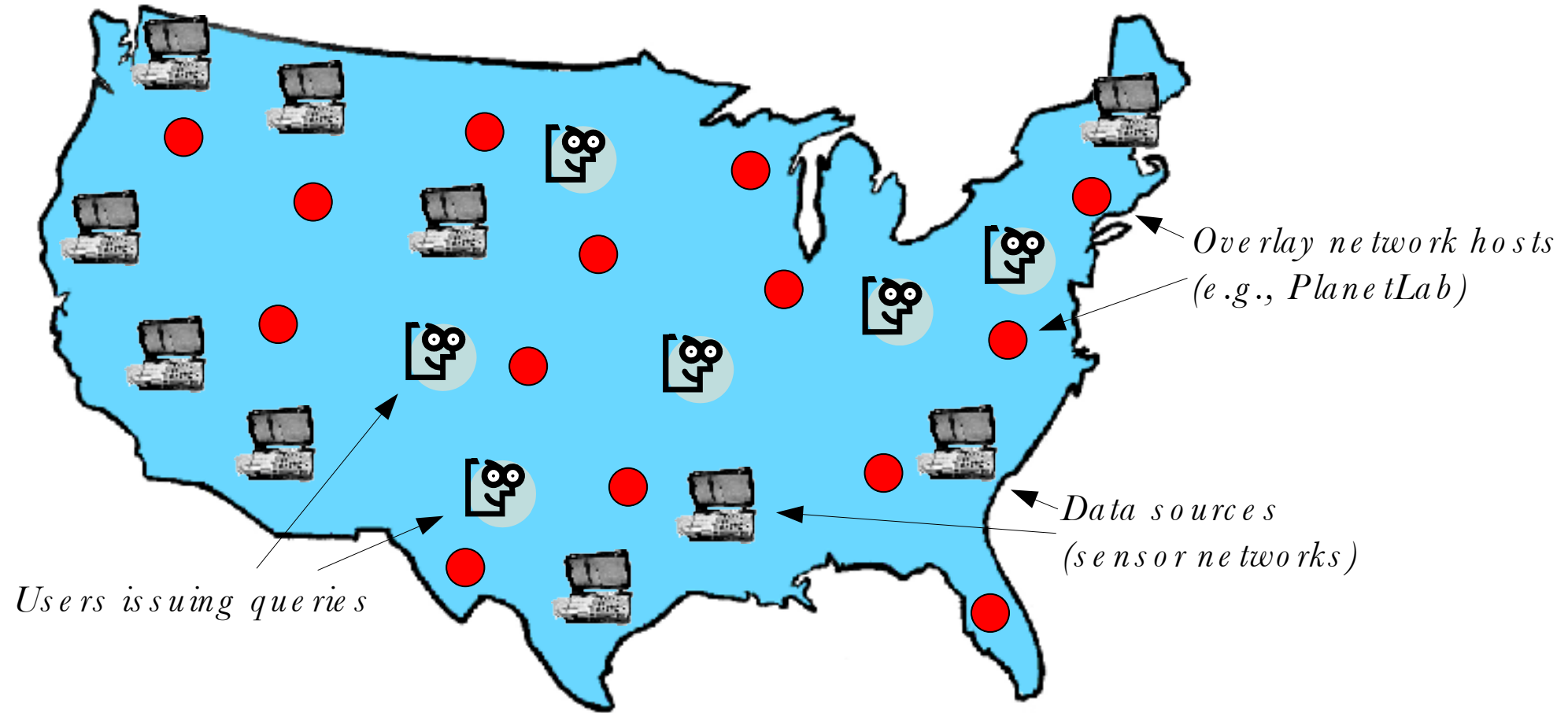
Extremely powerful, yet simple, abstraction

```
SELECT MAX(mag)  
FROM sensors  
WHERE mag > thresh  
SAMPLE PERIOD 64ms
```



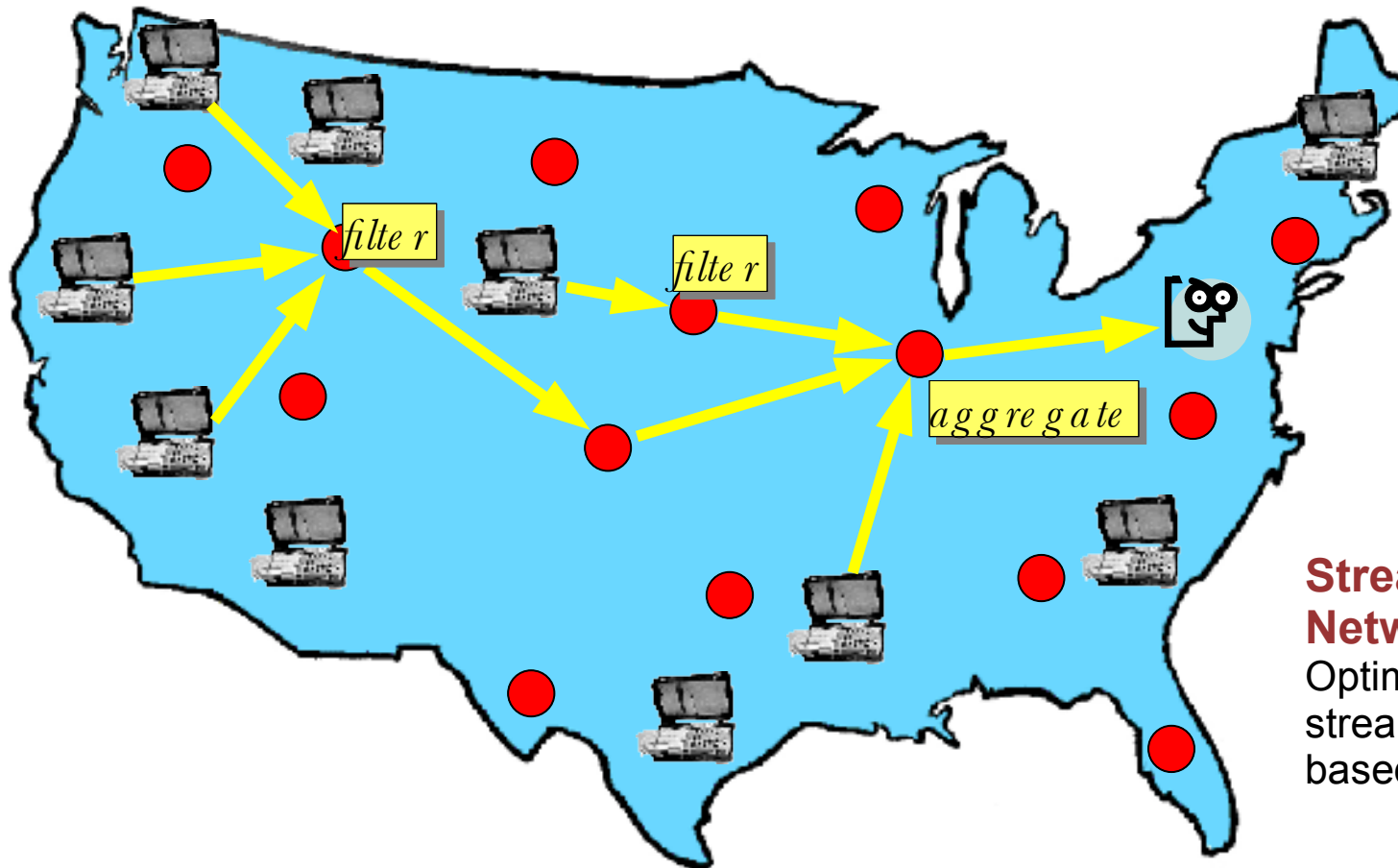
Hourglass and SBONs

Hourglass: Set of services to harness real-time sensor data across many geographically-distributed sources



Hourglass and SBONs

Hourglass: Set of services to harness real-time sensor data across many geographically-distributed sources



Stream-Based Overlay Network (SBON):
Optimize placement of stream-processing services based on network conditions

Internet measurement

Another rich source of sensor data: the Internet itself!

- Packet traces from routers
- BGP feeds

Distributed surveillance

- Honeypots and honeyfarms
- Automated worm fingerprinting
- Attack detection

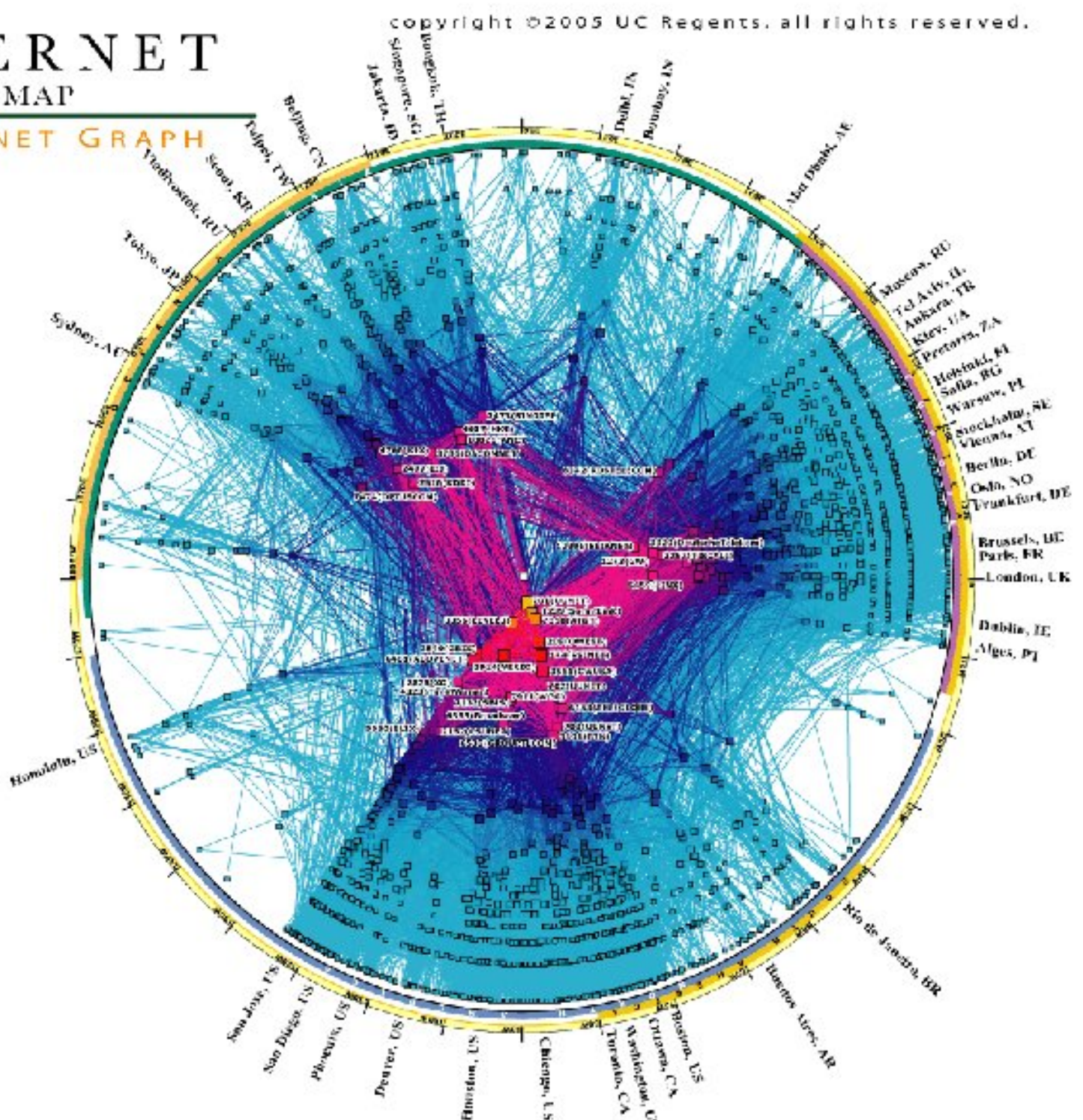
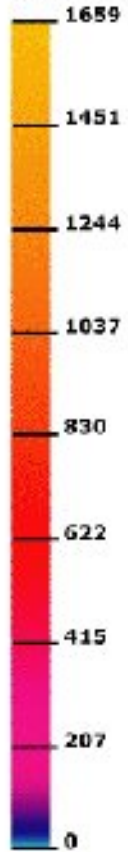
Blogs and RSS Feeds

- Lots of real-time data being pushed onto the Internet
- 1.6 million blog postings a day [Technorati]
- Can we search all of this data in real time?

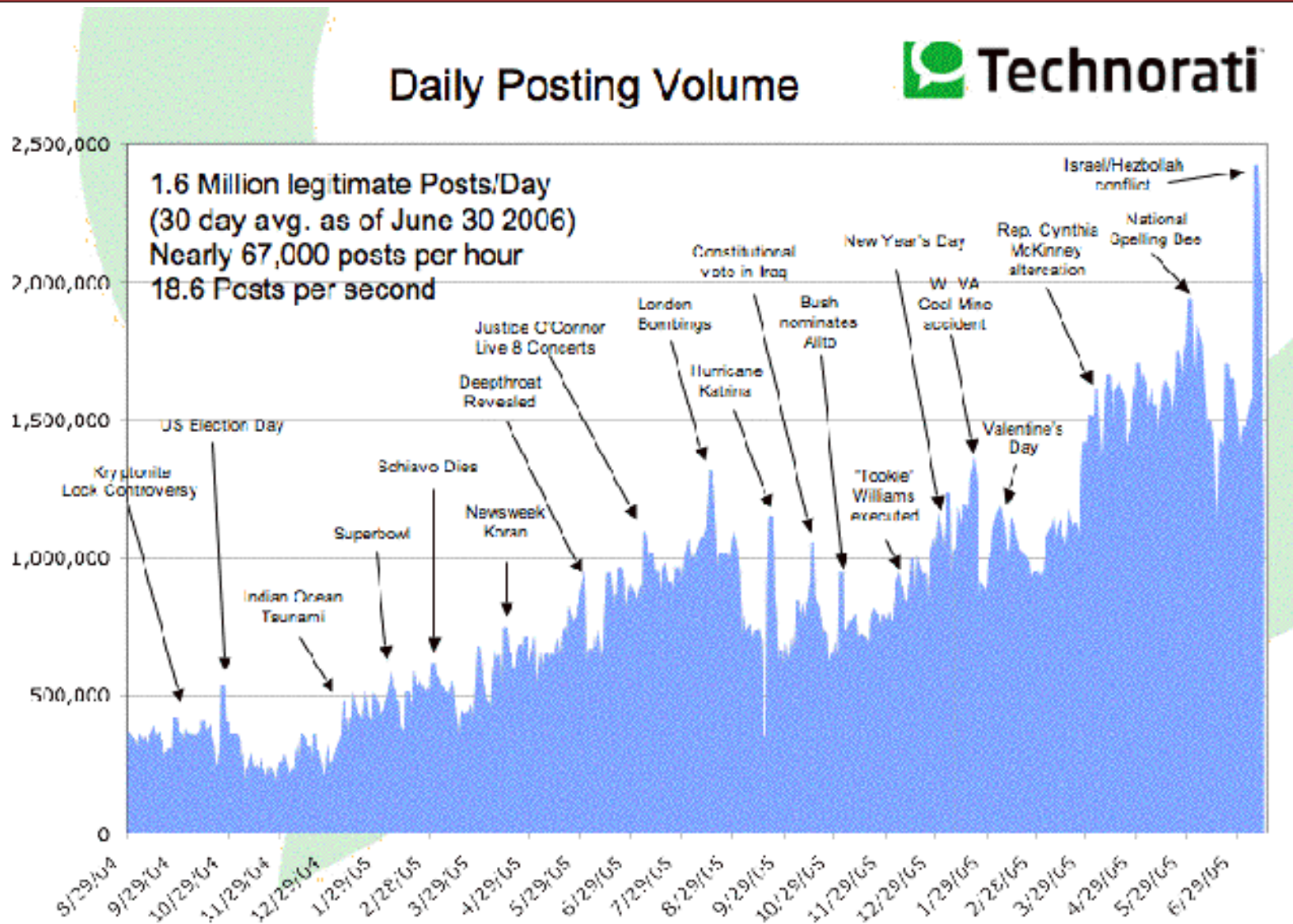
IPV4 INTERNET TOPOLOGY MAP

AS-level INTERNET GRAPH

Peering:
OutDegree



Blog posting volume, 8/04-7/06



About this course

This is a new graduate course on “Topics in Computer Systems”

Topic and professor will rotate year-to-year

- This will be the only time the course is offered on this topic

Goal: Intense discussions and push deep into a new research area

- This area is relatively unexplored

Syllabus overview

We will start with papers outlining the vision and application domains

- IrisNet: Distributed networks of webcams; NASA SensorWebs effort
- Berkeley HiFi project
- Historical motivation: Vannevar Bush

Streaming query systems: Aurora, Borealis, PIER

- Lots of work from the database community

Optimizations and load balancing in large-scale query systems

Querying wireless sensor networks

- TinyDB and extensions

Syllabus overview (continued)

Internet measurement

- ScriptRoute, scalable distributed attack detection, reconstructing the Witty worm
- Honeypots and honeyfarms: Potemkin and Netbait

Distributed systems monitoring

- Astrolabe, SDIMS, Sophia, and P2
- Using Blogs and RSS feeds as real-time data sources: Corona and Cobra

Guest lectures:

- Jim Waldo, Sun Microsystems – JINI and Neuromancer
- Jim Kurose, Umass Amherst – Collaborative Adaptive Sensing of the Atmosphere
- Jim Davis, Harvard – EarthScope and GPS-based geodesy
- Ugur Cetintemel, Brown – The Borealis Stream Processing System

Course Requirements

This is a graduate level Computer Science course.

- All Computer Science and EE grad students are eligible
- Undergrads welcome ...
- As are grad students from other fields.
- But, you must have taken undergraduate *operating systems* or *networking*

What will you do in this class?

- You will read and comment on advanced technical papers
- One significant programming project (C or Java programming required)
- Design and carry out an independent research project

Visitor and audit policy

- Talk to me first!

Course staff and administrivia

Instructor: Matt Welsh (mdw@eecs)

- Office: Maxwell Dworkin 233
- Office hours: Thursdays, 10am – 12pm

TF: Ian Rose (ianrose@eecs)

- Office: Maxwell Dworkin 238
- Office hours: TBD
- General course consulting and help with programming assignment

All papers, due dates, etc. on course web page:

- <http://www.eecs.harvard.edu/~mdw/course/cs260r/>

Readings and Reviews

You are responsible for completing assigned readings *before* lecture

- 1-2 papers for each lecture

Email a short review of the reading to cs260r-reviews@eecs

- Review is due before beginning of lecture
- A couple of paragraphs or “bullets” about the reading
- Highlight the main “take away” point of the reading
- For research papers, include a short critique of the work as well
 - *Be concise, critical, and thoughtful*

Reviews constitute 25% of your course grade

- You are allowed to miss two paper reviews over the term
- Each lecture counts for 1 point out of 20 possible

Programming Assignment

There is one programming assignment for the course

- Main goal: Get experience interacting with an Internet-wide sensing system
- You will use this experience for your course project

Project will involve collecting, processing, and visualizing data from real-time data sources

- Seismological data, weather data, traffic conditions, Webcams, etc. on the Net

You should be comfortable writing code in Java or C

- Some flexibility in your own implementation

Research Project

Main goal of this course: Do some research

- Work individually or in pairs (pairs preferred)
- Select a juicy research problem that fits the theme of this course

Use the project to further your own research goals

- Ideal project is one that fits in with your own thesis topic in some way
- Focus of project need not be on “systems” and “networks”
 - *e.g., theory, AI, hardware design, etc. are all valid*
 - *As long as it ties into the course topic in some way*

Project Requirements

Project Proposal (due October 26)

- Short (2 pages max) on what you propose to do, why the project is interesting, and how you plan to get started
- Should include rough schedule of project milestones
- Short project update due 11/21 – short email on where you are and how you plan to finish up your project

Research presentations (last two days of class)

- Give a short, fun talk telling us what you did
- Learn from each other's experiences

Research papers

- Conference-style research paper (10 pages max) detailing your project
- Goal is to get used to writing these things – it's important
- I can work with you afterwards to turn it into a conference/journal submission

Project Ideas

Develop a new query language or programming model for large-scale real-time data processing

Explore tradeoff between overhead, availability, and consistency in stream-processing query systems

Develop a security mechanism giving different classes of users access to data at varying levels of freshness, fidelity, or precision

Interface the Harvard MoteLab sensor network to an existing query system, such as Borealis

Implement a scalable, distributed worm detection system

Grading overview

Paper reviews – 25%

- Must get them in before lecture; allowed to miss two

Class participation – 15%

- Attendance in class is required; participate in discussions
- Come to class with a few questions you want to raise about each paper.

Programming assignment – 15%

Project proposal – 5%

Project presentation – 15%

- Will be graded on both technical content and quality of talk

Final paper – 25%

- Also graded on both technical content and writing quality

Next lecture

Two (short) papers to read for next lecture:

IrisNet: An Architecture for a Worldwide Sensor Web

- Phil Gibbons et al., *IEEE Pervasive Computing*, 2003

An Autonomous Earth-Observing Sensorweb

- Steve Chien et al., *IEEE Intelligent Systems*, 2005

Send reviews to cs260r-reviews@eecs before class!