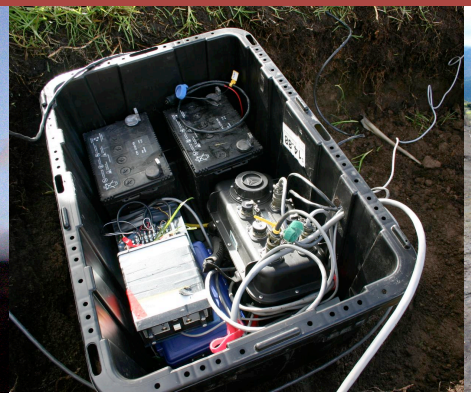
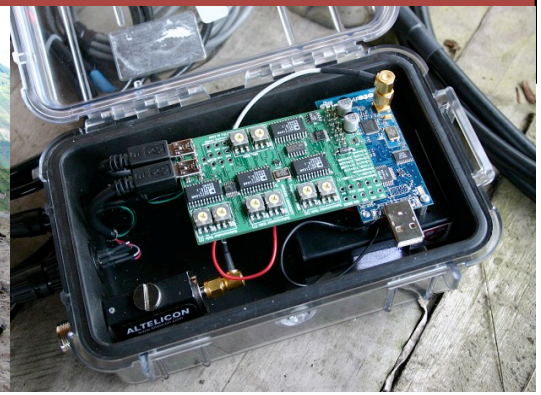
Sensor Networks for the Sciences: Lessons from the Field



Matt Welsh

Harvard University

Talk Outline

Some background.

- Volcano monitoring as a driving application.

Some lessons.

- What works, what doesn't.

Research directions.

- Making scientists happy, while publishing CS papers.

Some Background

Our group has been working with seismologists for the last four years

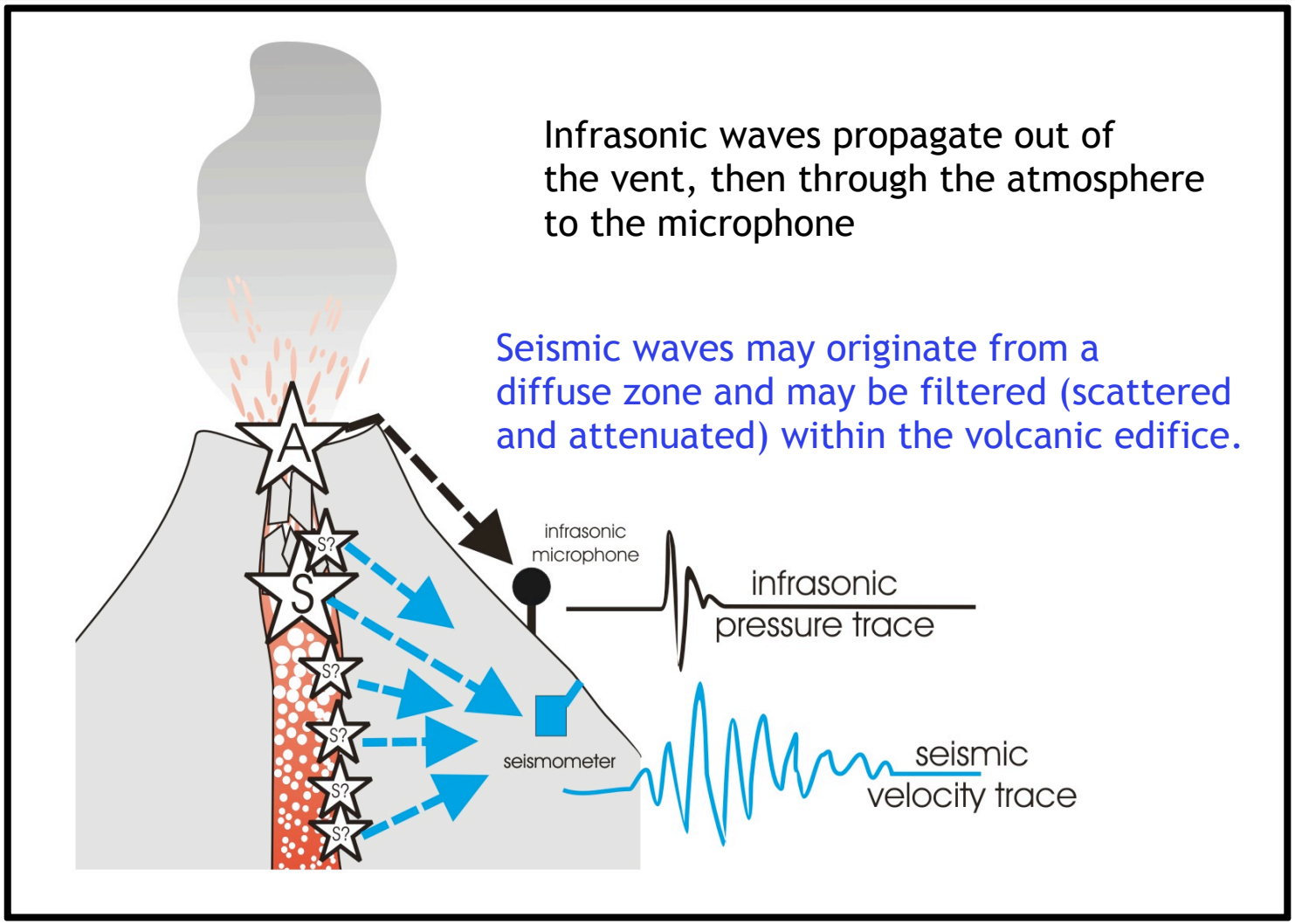
Deployed three sensor networks on active volcanoes in Ecuador

This application pushes the boundaries of sensor net design:

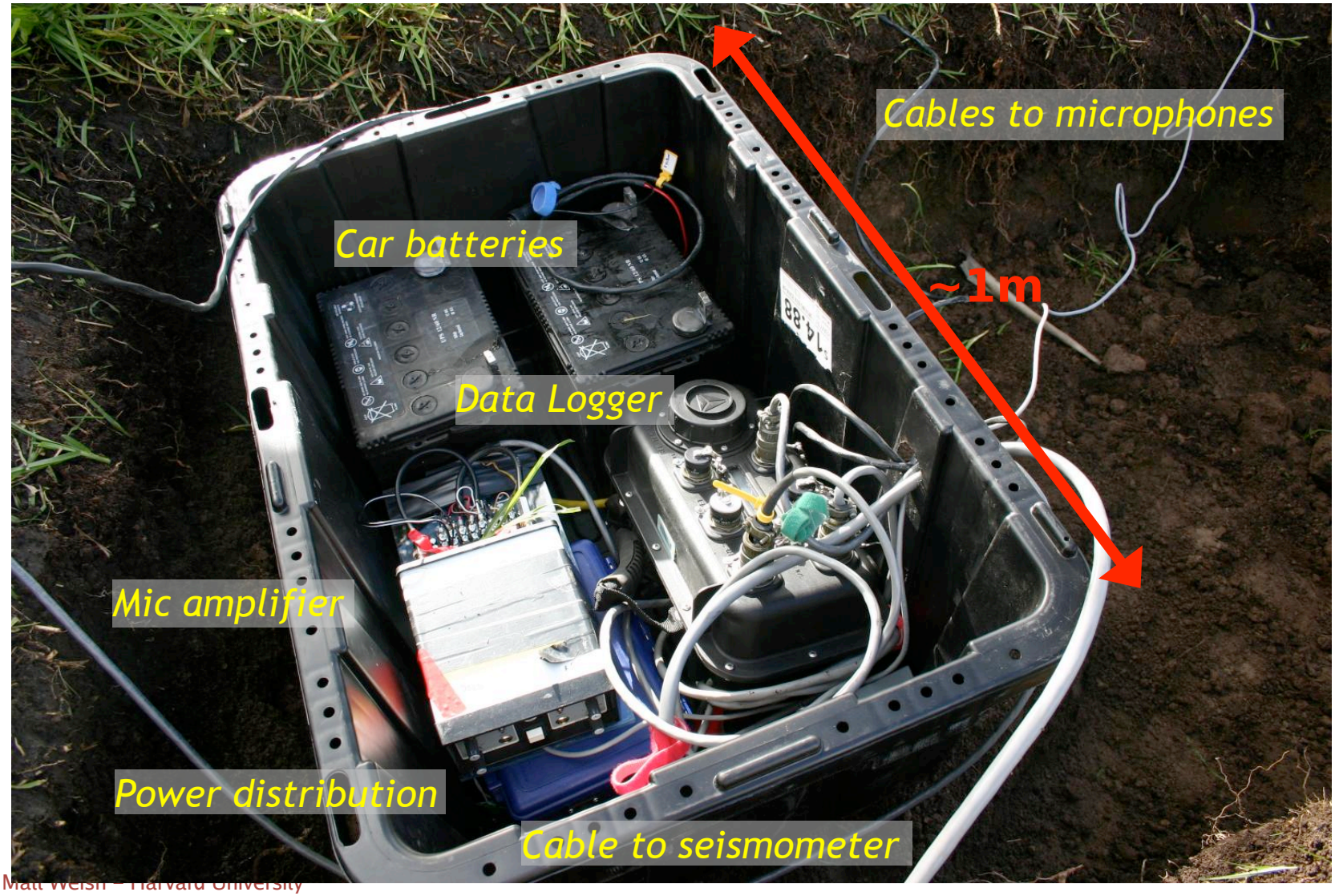
- High data rates
- Fine-grained time synchronization
- Reliable data collection
- Discerning interesting signals from noise

We've learned a lot about what domain scientists need from sensor nets.

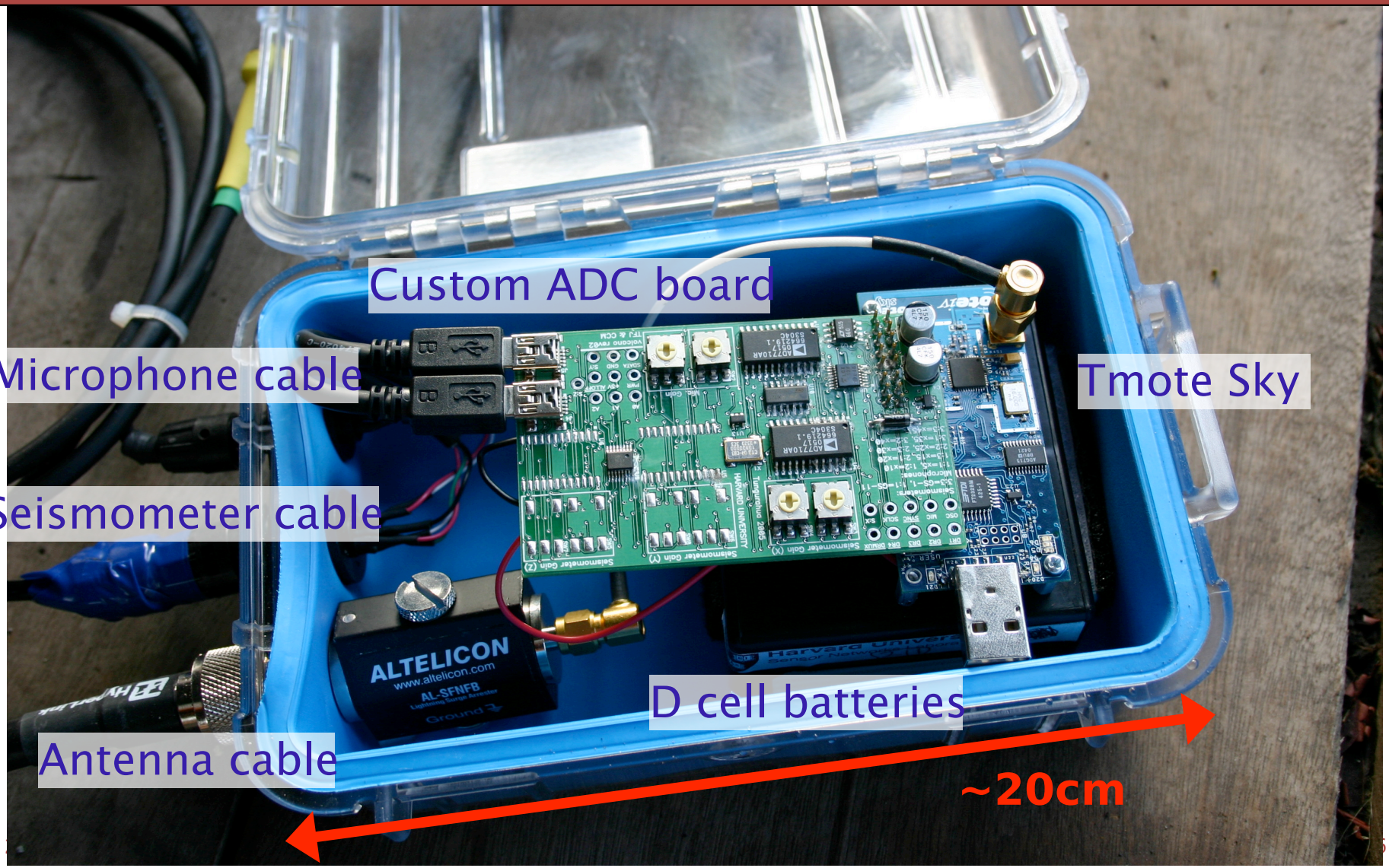
Sensor Networks for Volcanic Monitoring



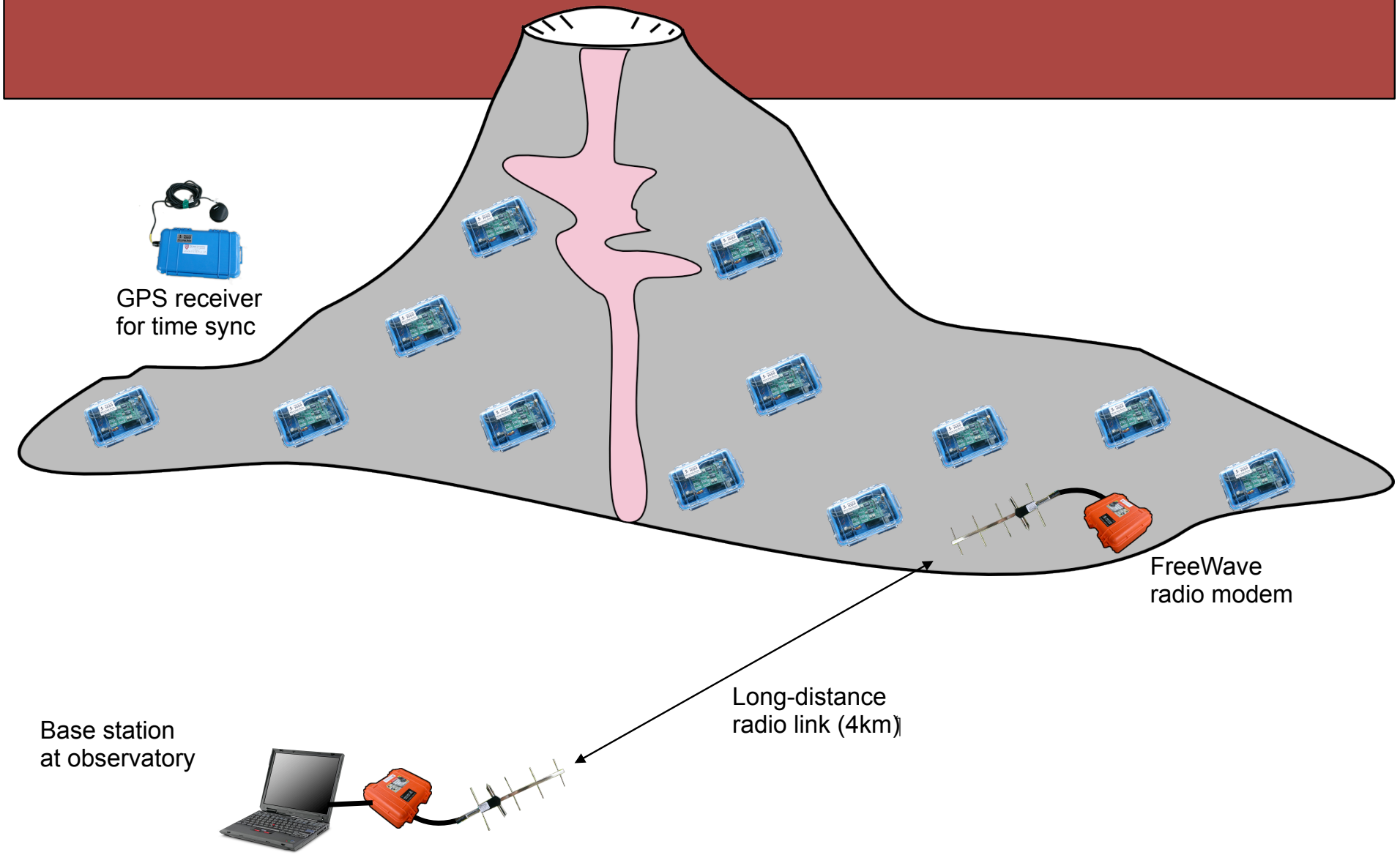
Existing Volcanic Sensor Station



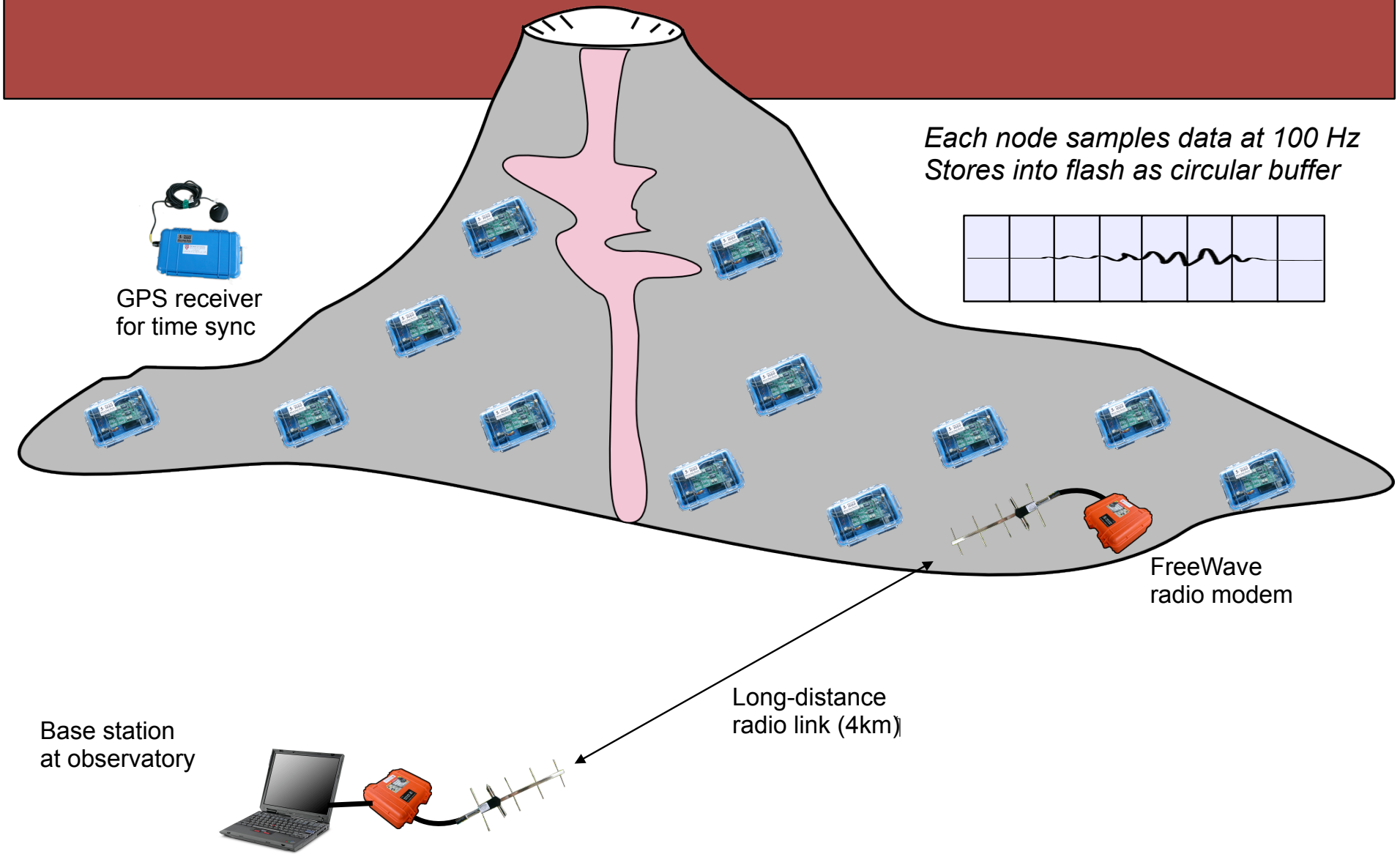
Our Mote-Based Sensor Node



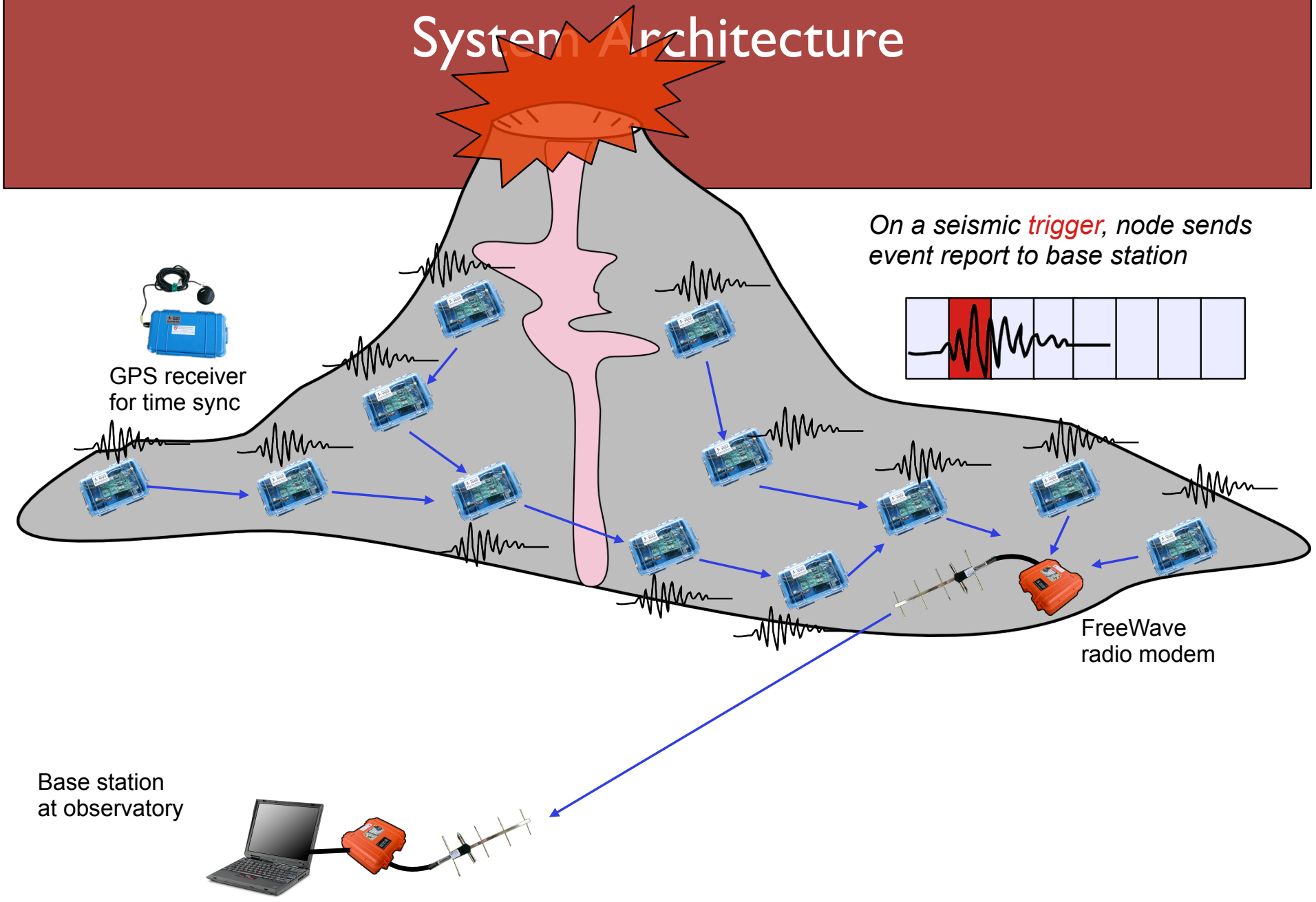
System Architecture



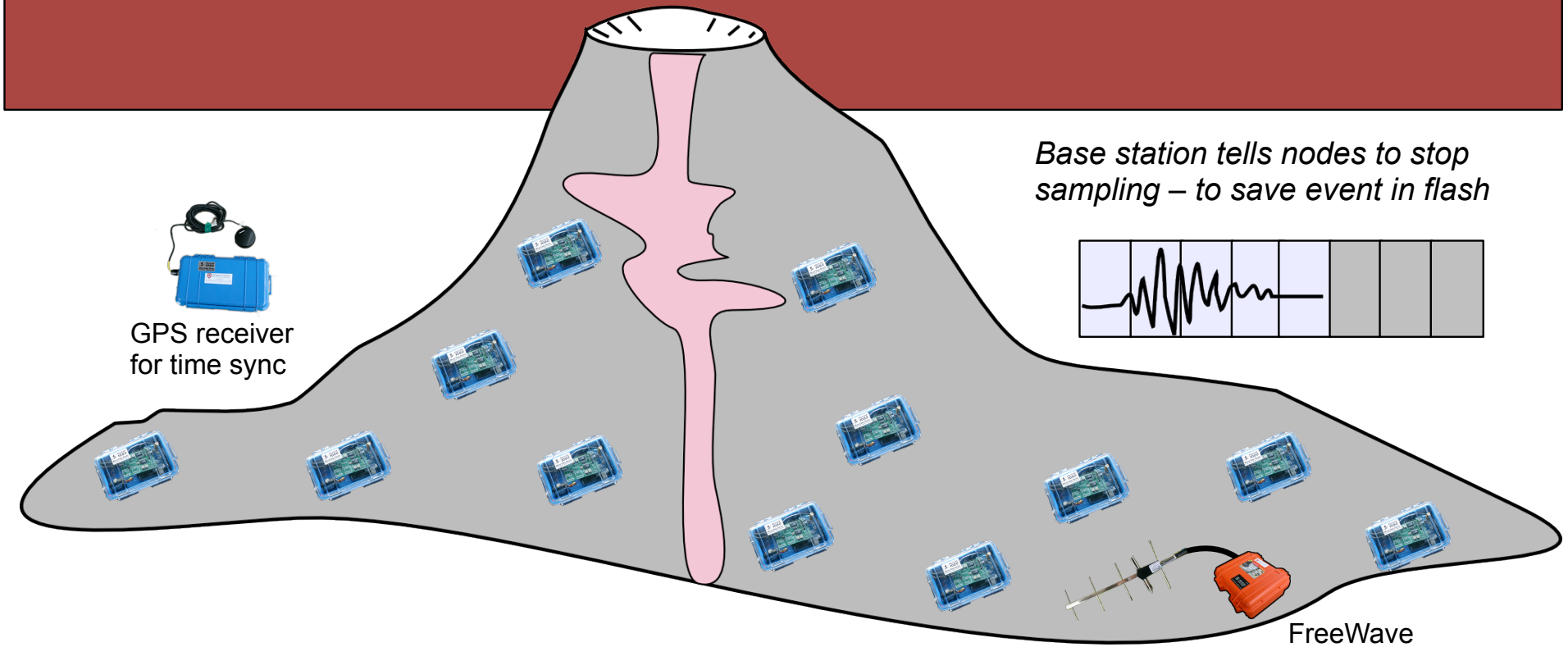
System Architecture



System Architecture



System Architecture



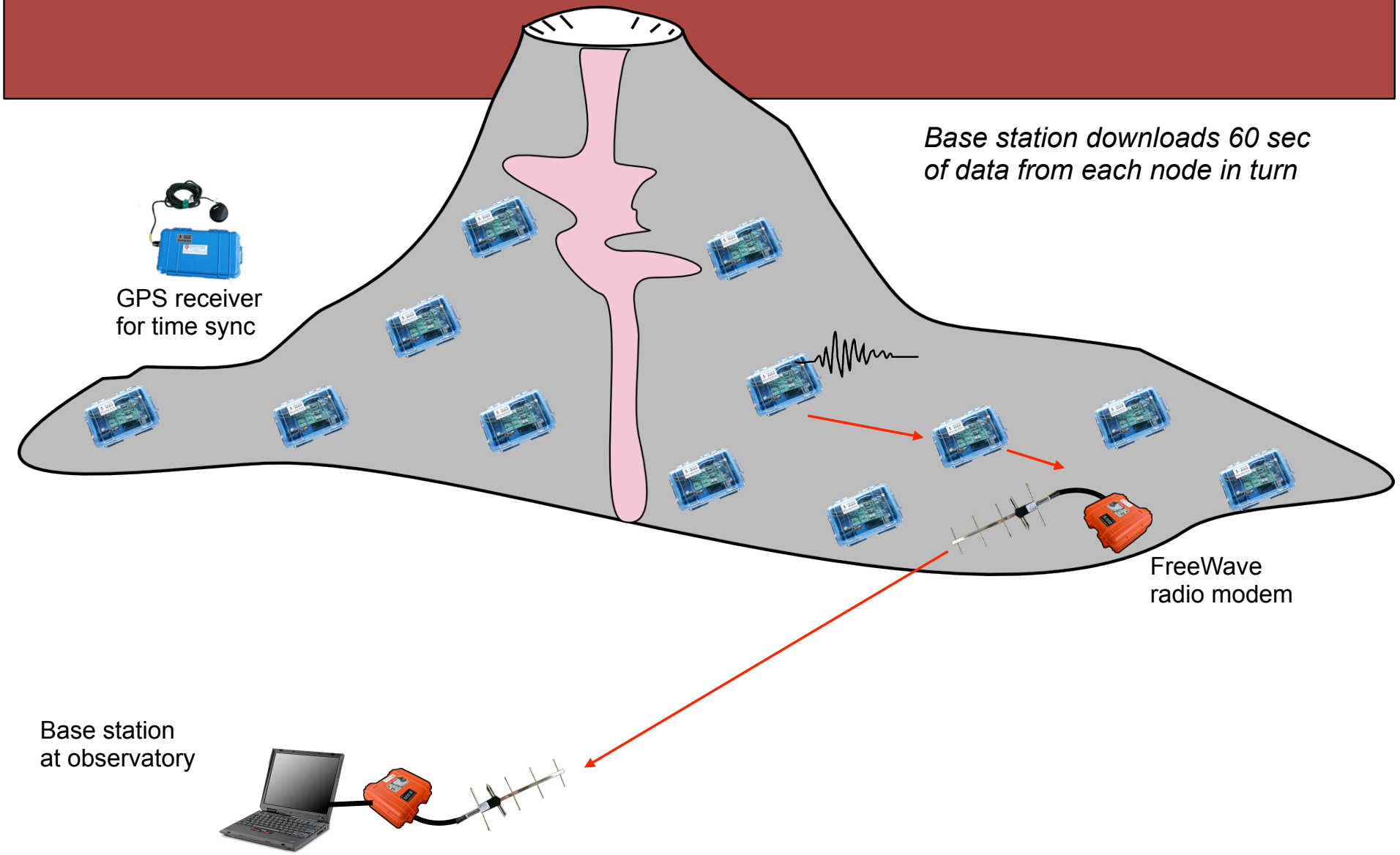
Base station tells nodes to stop sampling – to save event in flash

GPS receiver
for time sync

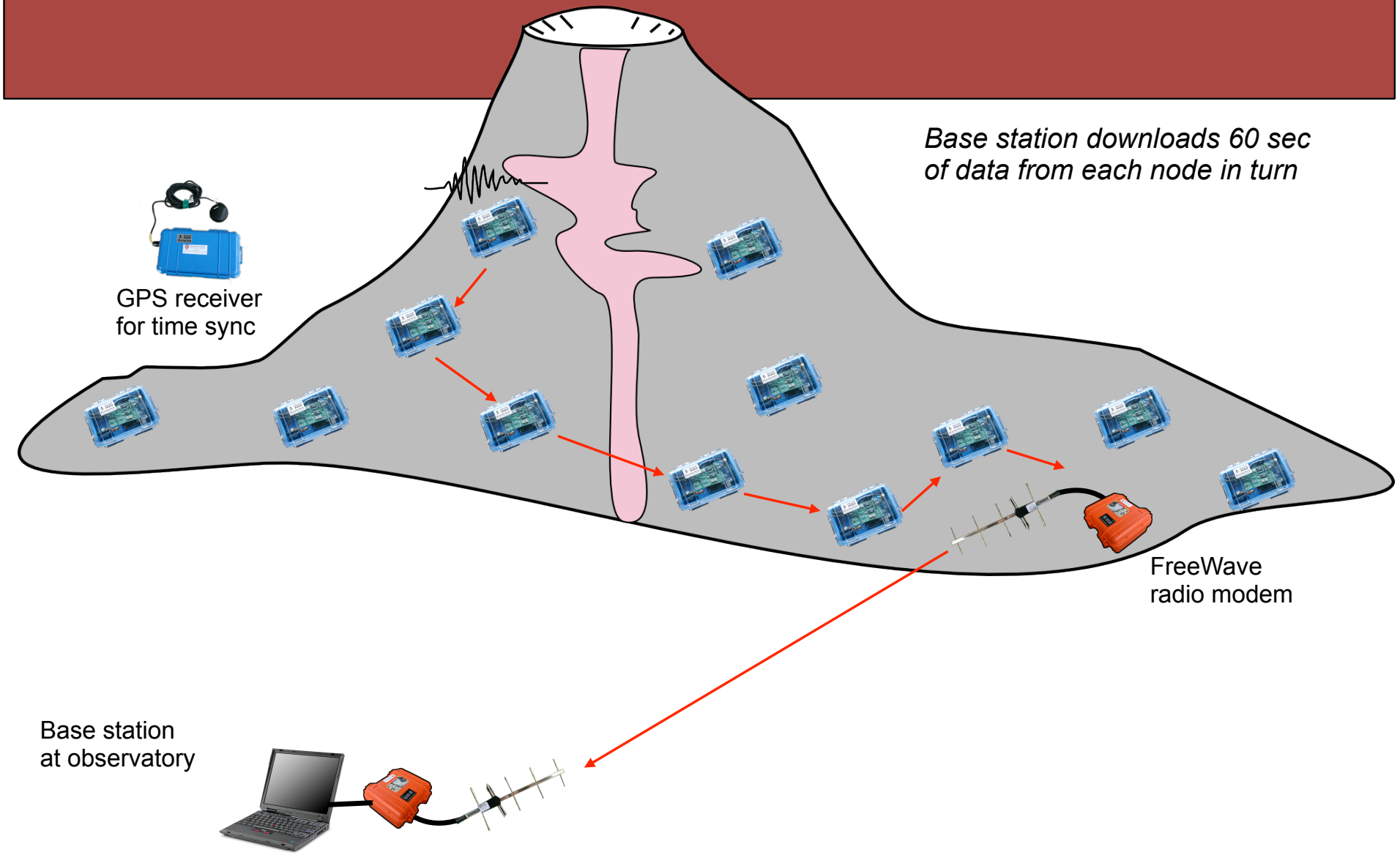
FreeWave
radio modem

Base station
at observatory

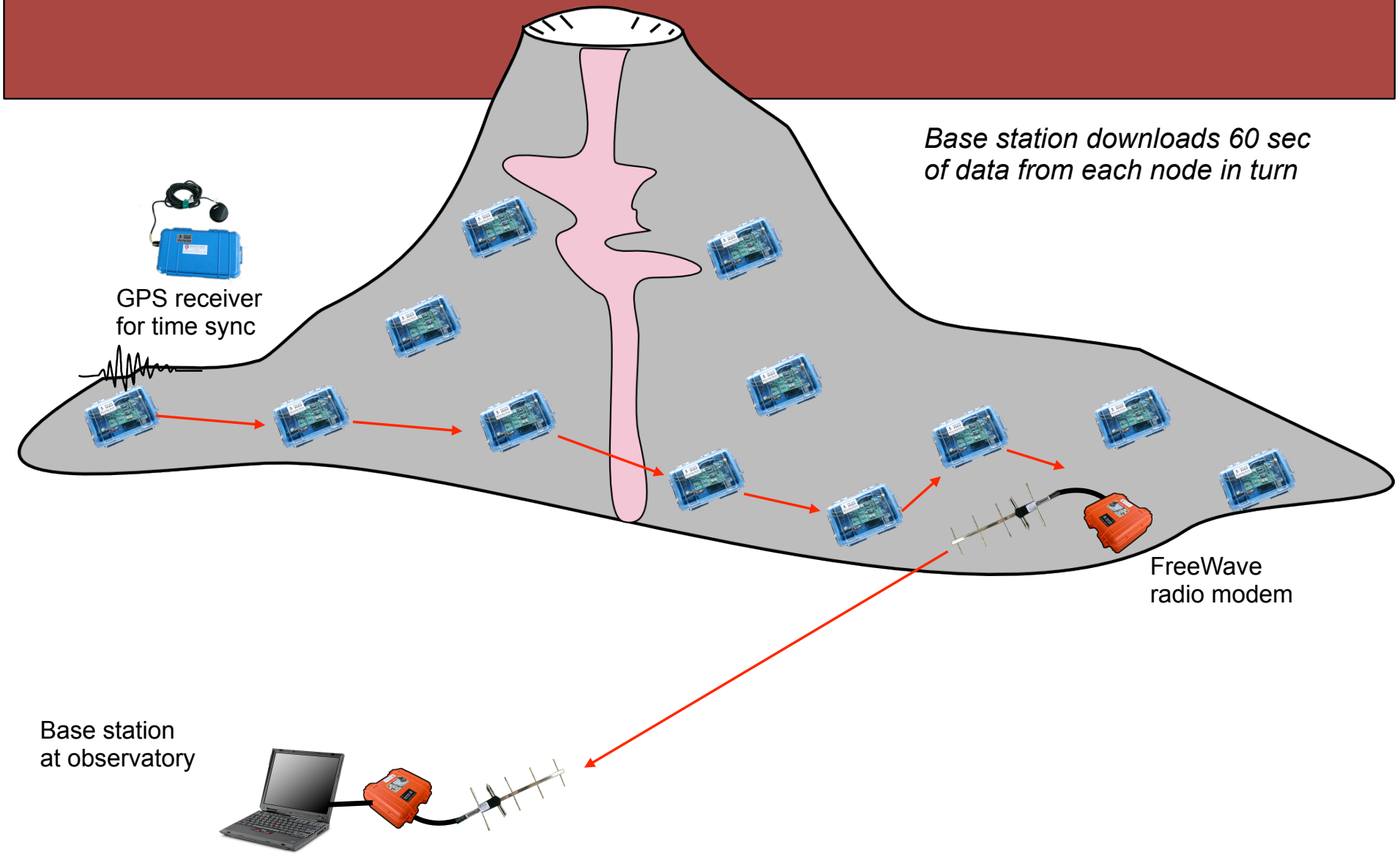
System Architecture



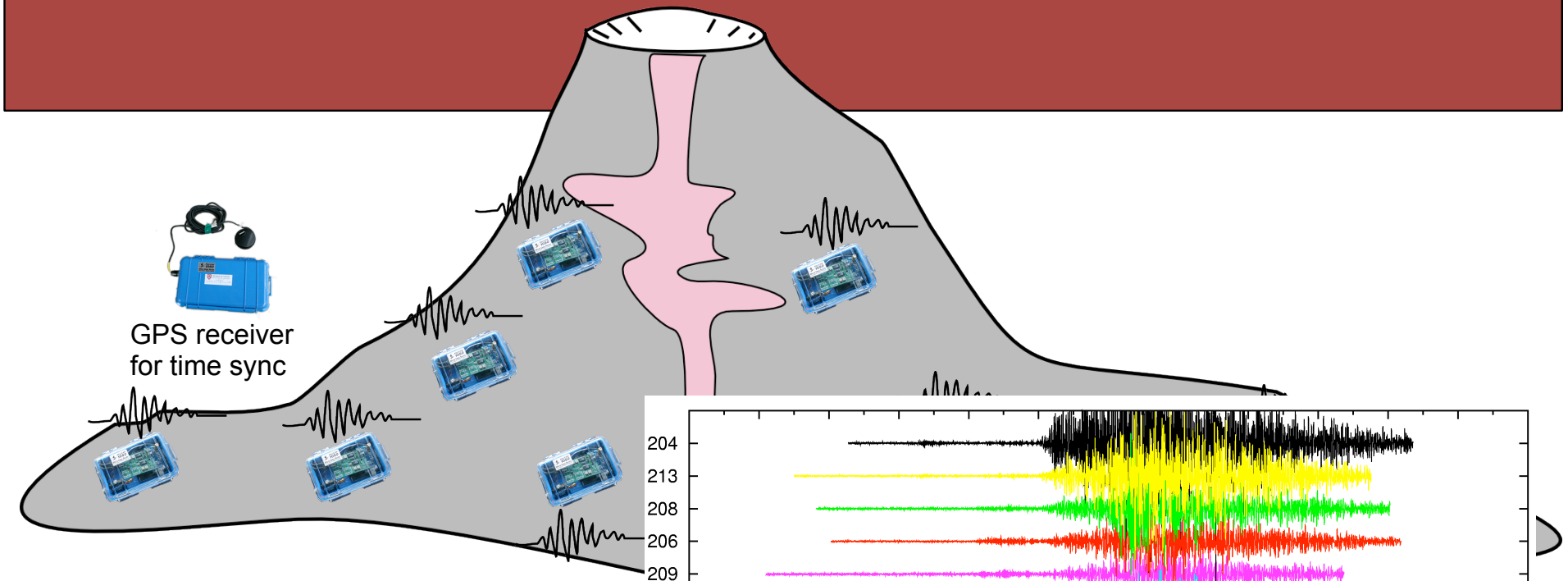
System Architecture



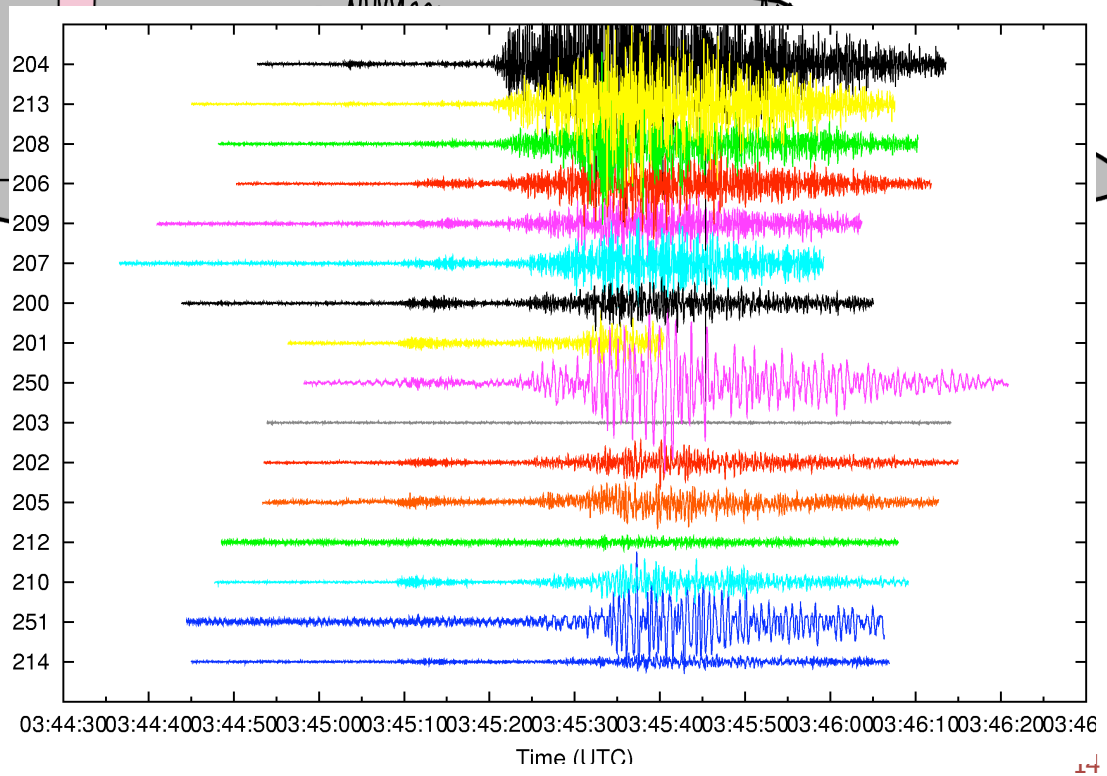
System Architecture



System Architecture

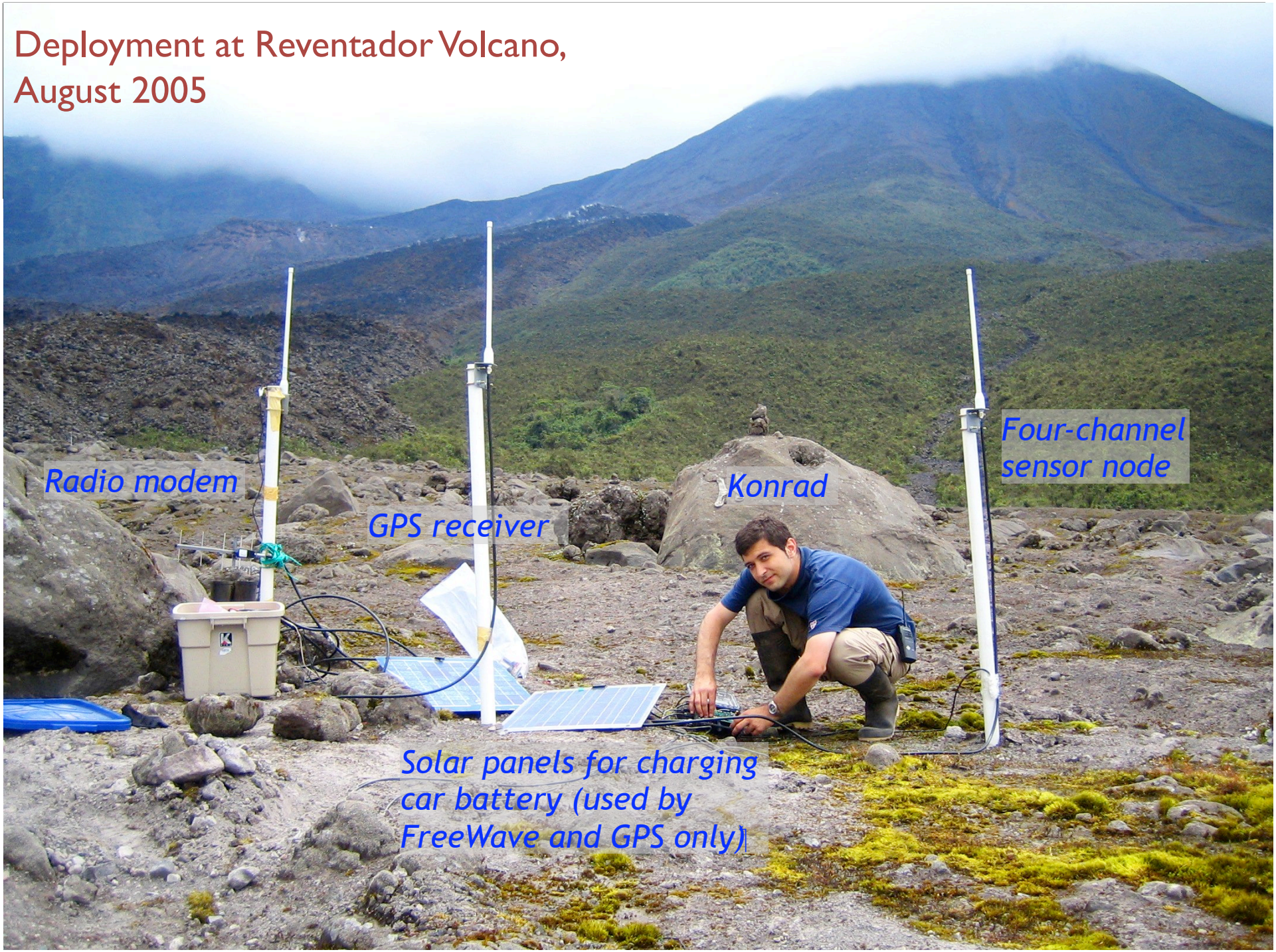


GPS receiver
for time sync



Base station
at observatory

Deployment at Reventador Volcano, August 2005



Some sensor network myths…

Nodes deployed randomly (e.g., dropping from airplanes)

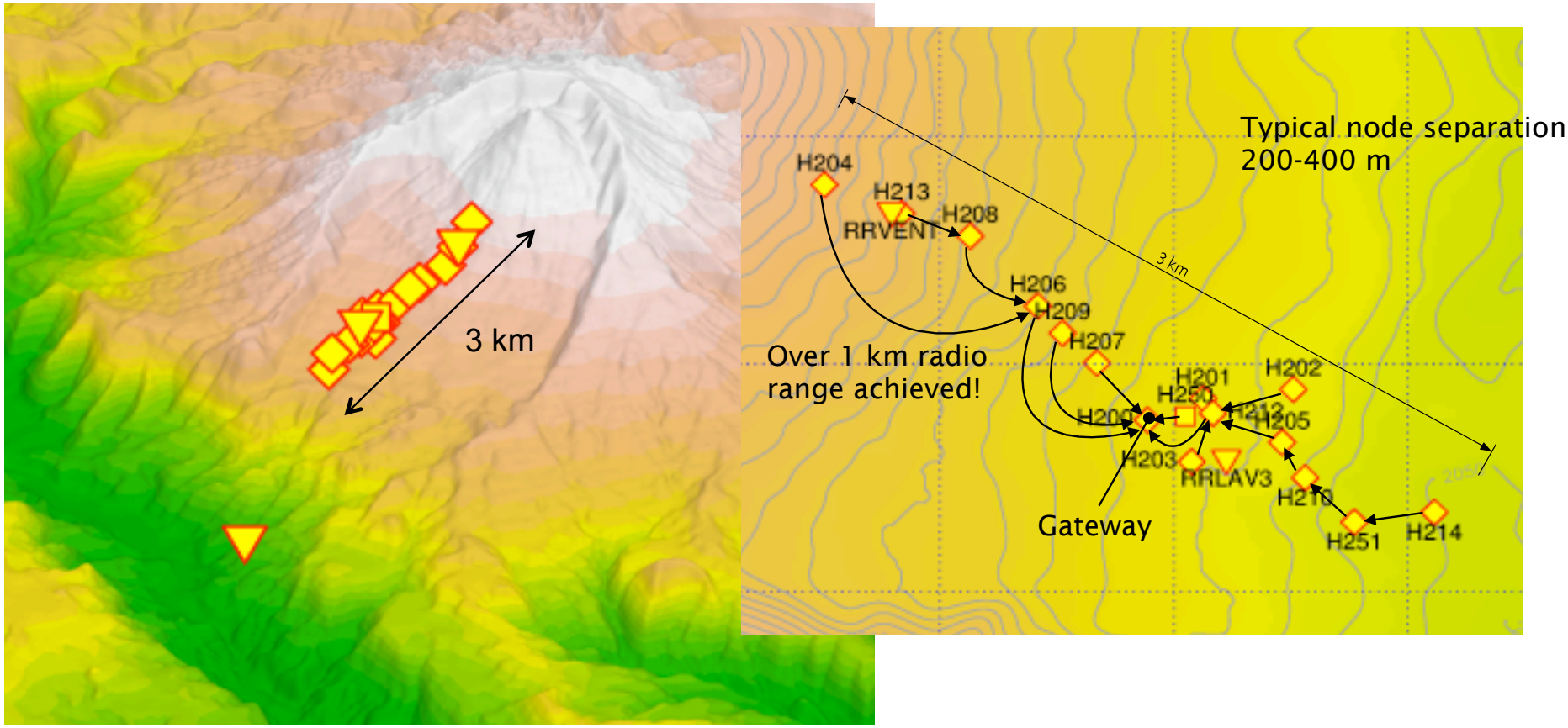
Some sensor network myths…

Nodes are cheap, and tiny

Mote: $75
Custom ADC board: $300
Case: $20
Sensors: $100 - $1500
Total: **up to $2000/node**

Some sensor network myths…

Network deployment is dense and spatially homogenous



What we learned…

Field work is harder than it really should be.

- Donkeys are not generally reliable as an equipment transport mechanism.

There is a tremendous gap between "it works in the lab" and "it works in the real world."

- "In the lab, if you don't like the result, run it again" – D. Culler

Real Scientists have much, much more rigorous standards than Computer Scientists.

Lesson #1

Scientists just want all the data!

- Not *some* of the data
- Not an aggregate of the data from across multiple nodes
- Not some computer scientist's interpretation of an *ad hoc* model applied to the data

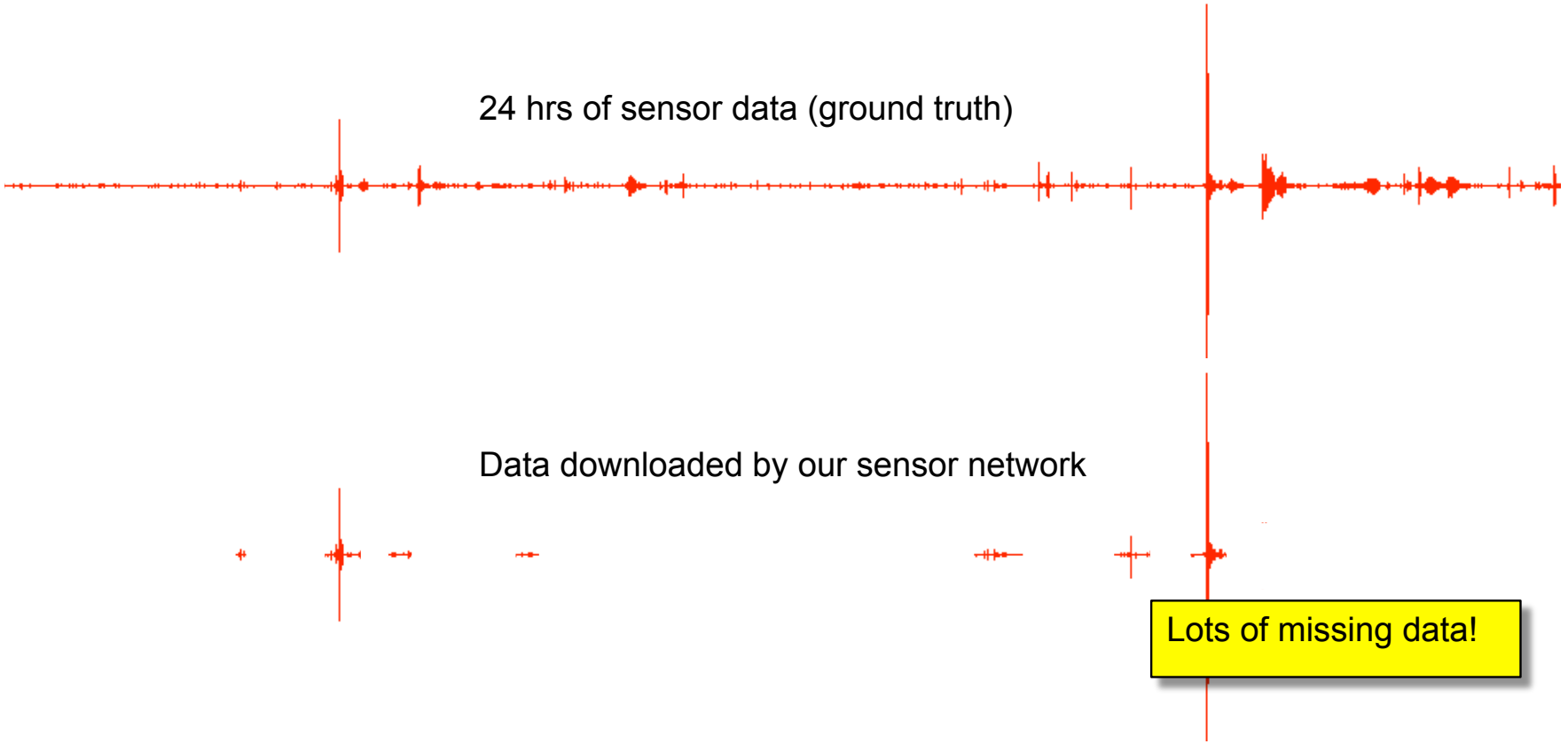
Fundamental mode: **Collect first, ask questions later**

- Data collection is expensive and time-consuming
- Want to squeeze as many journal papers from the data as possible
- Assumes long latency and lots of manual effort to clean up raw data

So, how did we do?

System designed to trigger on event detected across 5 or more nodes

- Only downloaded 60 sec. of each event per node, due to storage and bandwidth limitations.



The problem: Resource limitations

Sensor nodes have only 1 MByte of flash

- Enough to store ~ 20 minutes of data

Reliable transfer protocols are slow!

- Best speed we've seen is around 500 bytes per second
- Often do much worse: below 100 bytes/sec in some cases
  - Can take several minutes to download 60 sec. of data from a single node
- Flush [Dutta et al.] claims up to 1024 bytes/sec in multihop cases
- Also, can only reasonably perform one download at a time

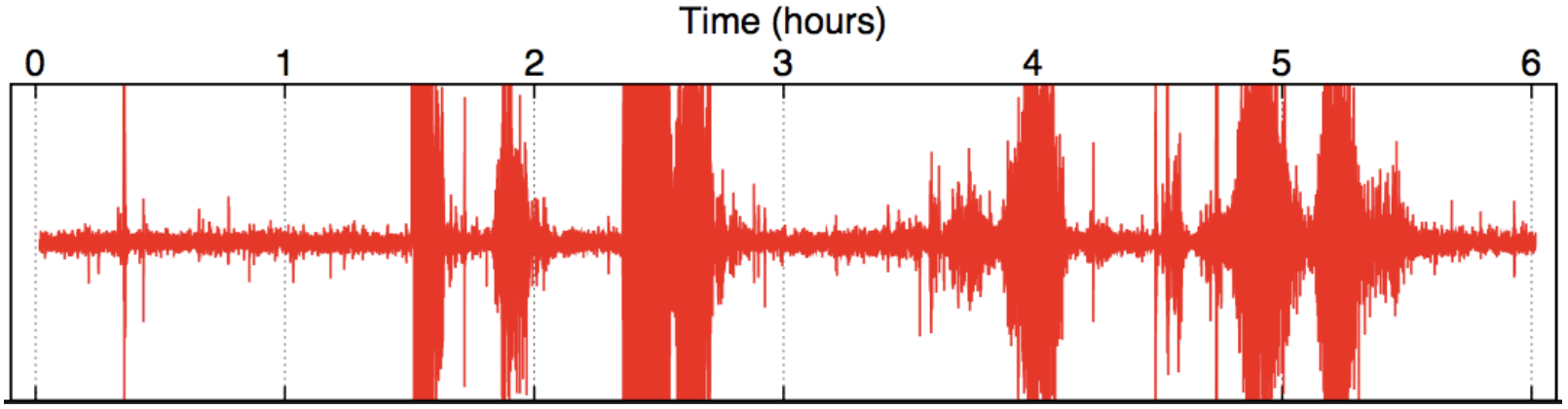
So, need to be careful about how we allocate storage and bandwidth.

Not all data is created equal

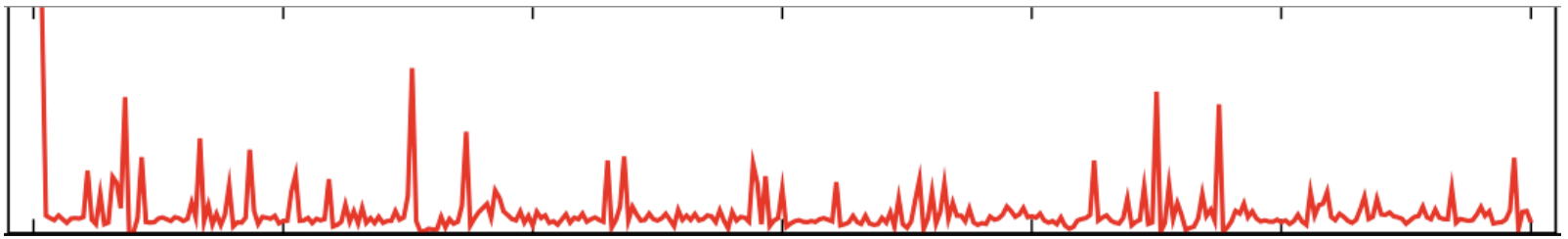
Core assumption: some data is “better” than others.

- Application's goal is to extract the “best” data given resource limitations

Raw signal



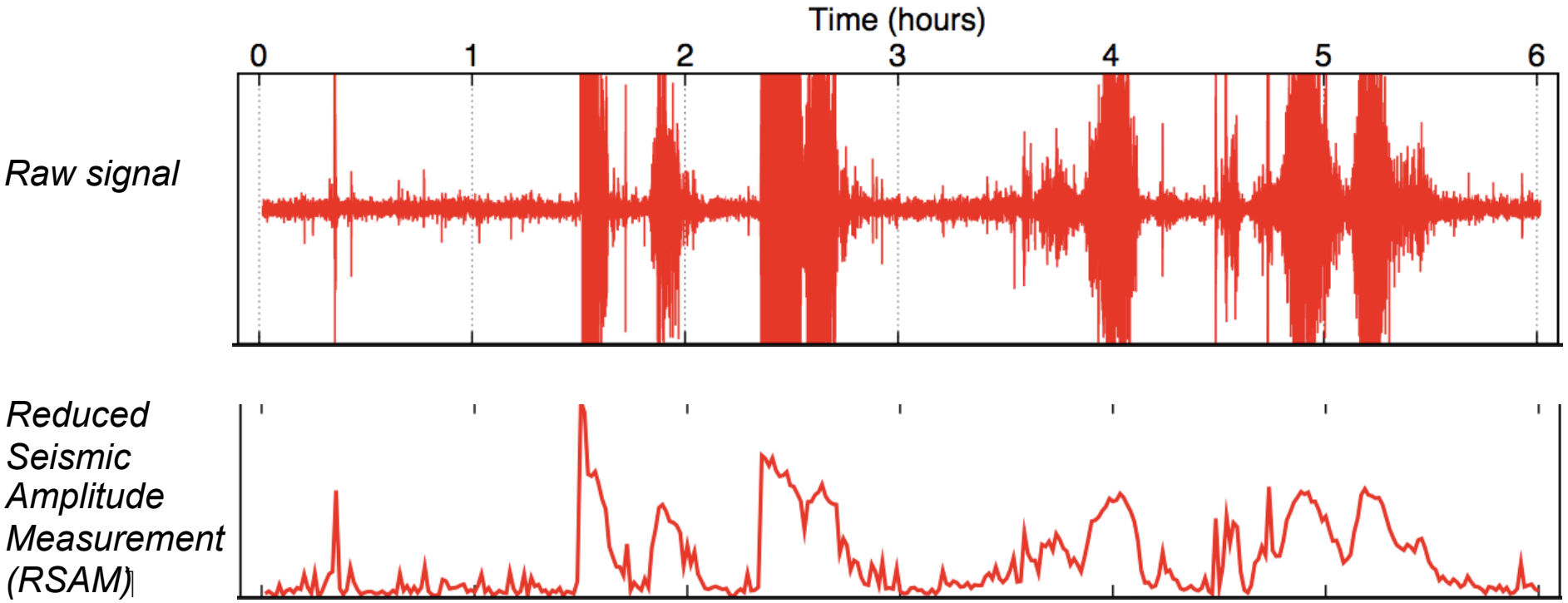
Short-term average divided by long-term average



Not all data is created equal

Core assumption: some data is “better” than others.

- Application's goal is to extract the “best” data given resource limitations



Our Solution: Lance

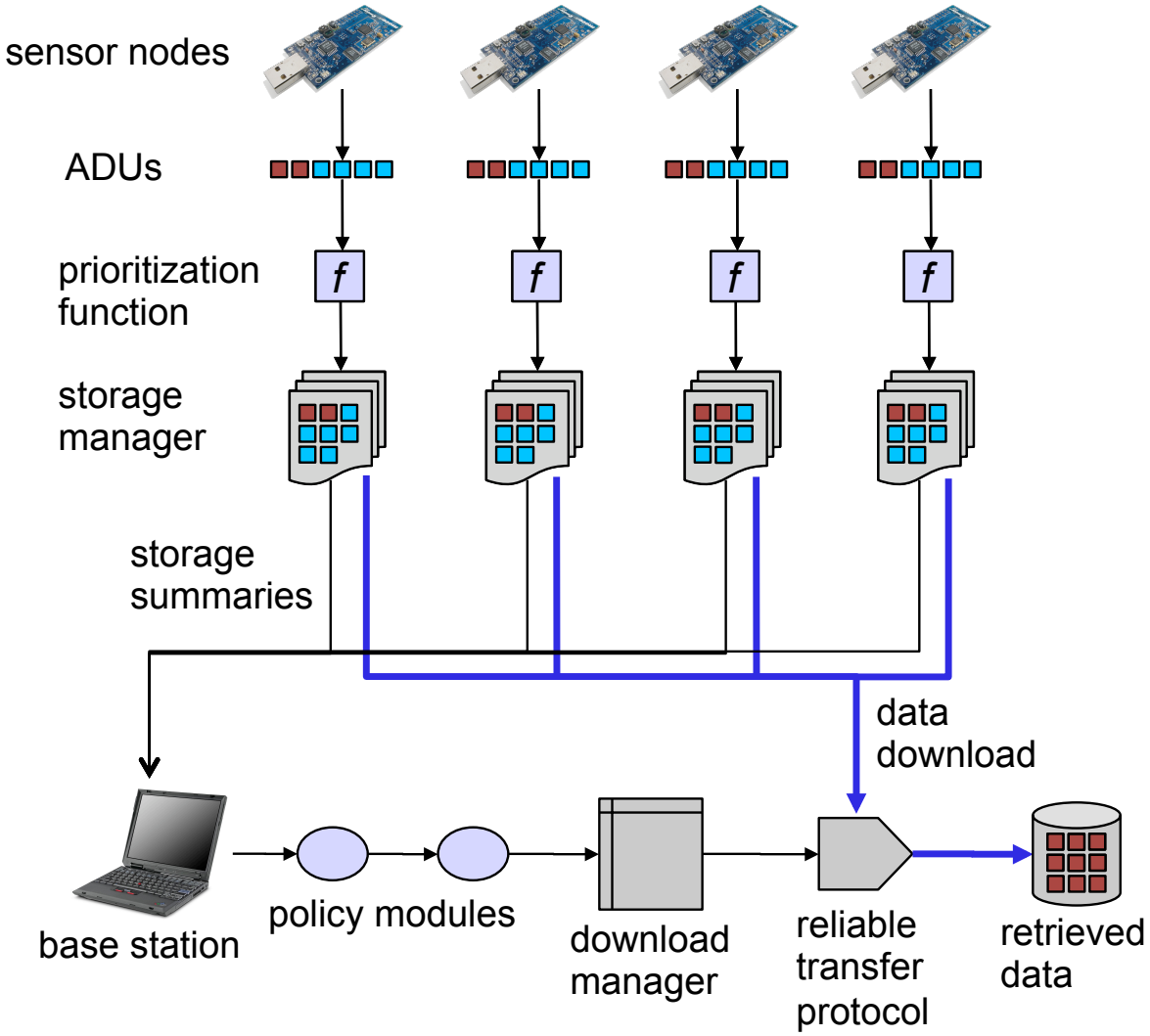
System for priority driven allocation of bandwidth and storage resources within a sensor network

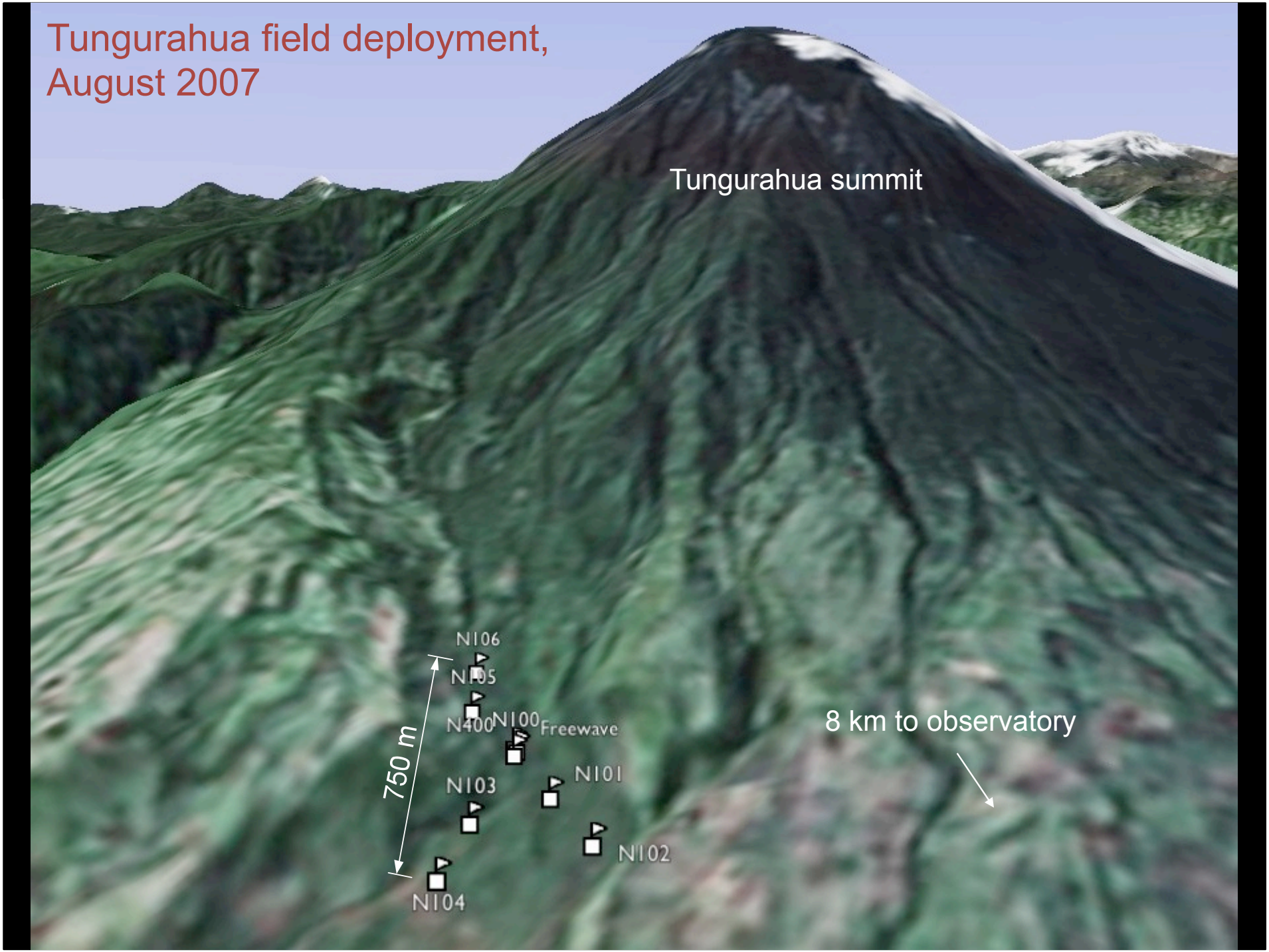
- Sensor nodes assign **priority** to data based on application specific metric
- Priority drives allocation of nodes' flash storage
- Data **summaries** sent to base station, used for allocating radio bandwidth for data extraction

Lance decouples mechanisms for resource allocation from app-specific policies

- App-supplied **policy modules** permit a wide range of allocation policies to be implemented
- Can target many optimization metrics: priority maximization, fairness, spatial/temporal data distribution, and more

Lance Architecture




Tungurahua field deployment,
August 2007
Tungurahua summit
N106
N105
N400
N100
Freewave
N101
N103
N102
N104
750 m
8 km to observatory

Tungurahua Deployment Results

Lance was deployed at Tungurahua volcano in August 2007

- 8 sensor nodes, 71 hours total deployment
- Downloaded a total of 77 MB of signals, nearly continuously, during the deployment

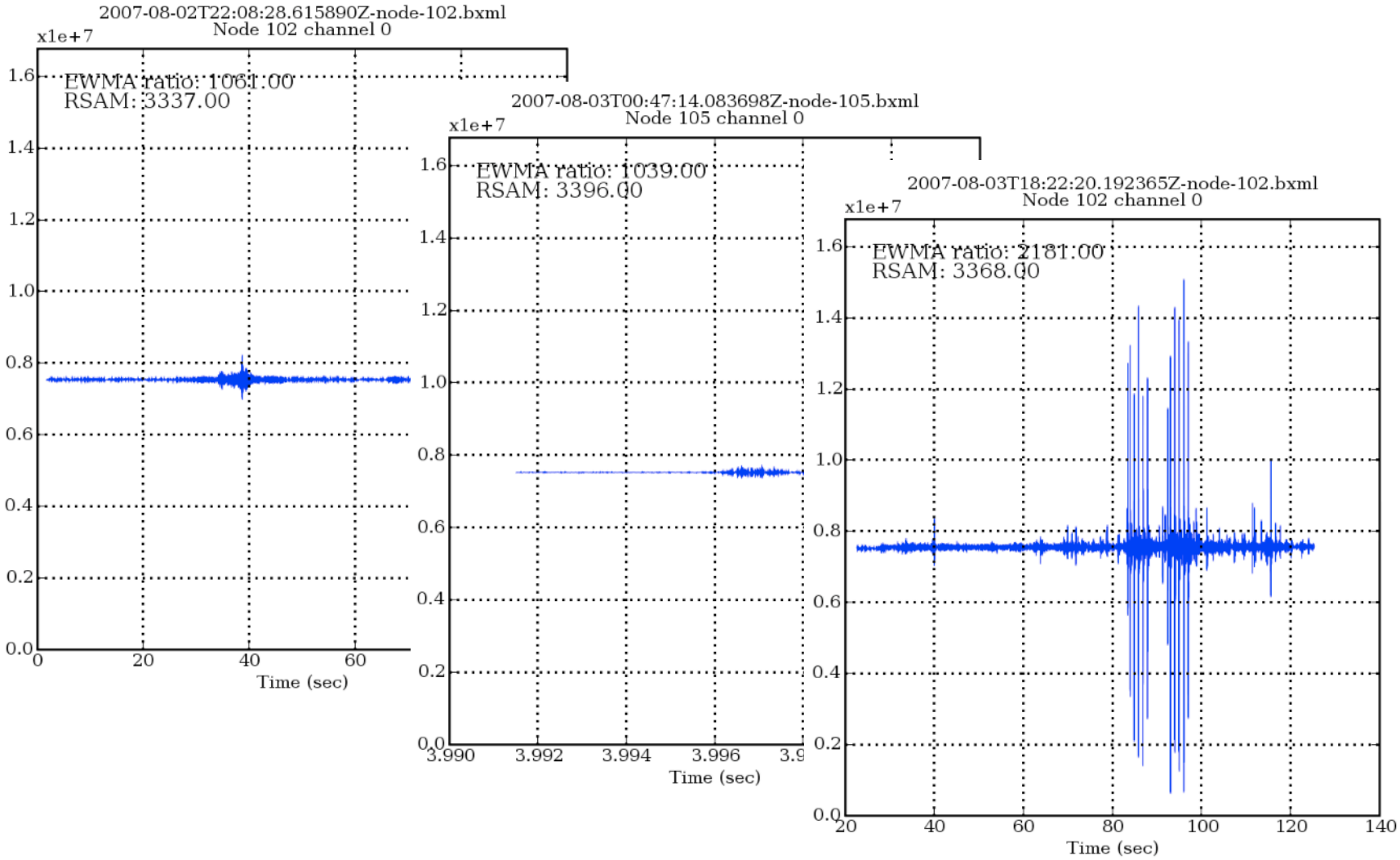
Optimality metric

- Compare data downloaded by network to an "oracle" system that has complete knowledge of all signals (past and present)
- In simulation, Lance achieves **at least 95% optimality** under a wide range of storage and bandwidth constraints, data distributions, and network topologies.

In the field: Lance achieved **73%-80%** of an optimal system

- Lower than simulations, largely due to lack of exciting volcanic activity during the deployment.

We fixed the volcano...



Lesson #2

Domain scientists and computer scientists need common ground

- Both sides need to get something out of the collaboration.

Computer scientists generally want to do “cool stuff”

- New protocols, new algorithms, new programming models, new hardware platforms

Lesson #2

Domain scientists and computer scientists need common ground

- Both sides need to get something out of the collaboration.

Computer scientists generally want to do “cool stuff”

- **New** protocols, **new** algorithms, **new** programming models, **new** hardware platforms
- Our field thrives on novelty…

Lesson #2

Domain scientists and computer scientists need common ground

- Both sides need to get something out of the collaboration.

Computer scientists generally want to do “cool stuff”

- **New** protocols, **new** algorithms, **new** programming models, **new** hardware platforms
- Our field thrives on novelty…

Domain scientists mostly want high-quality data

- Need to publish journal papers!
- Conservative (but in a good way)

Anything new we throw at them makes it harder to publish…

Getting on the same page (in three easy steps!)

Step 1: Start small.

- Don't presume you know anything about the domain.
- Our first deployment was 3 nodes running for 2 days.

Getting on the same page (in three easy steps!)

Step 1: Start small.

- Don't presume you know anything about the domain.
- Our first deployment was 3 nodes running for 2 days.

Step 2: Build confidence.

- Collect data with an eye towards making the scientists happy ☺
- Understand their needs and metrics for success.
- Be prepared to throw away many aspects of your design.

Getting on the same page (in three easy steps!)

Step 1: Start small.

- Don't presume you know anything about the domain.
- Our first deployment was 3 nodes running for 2 days.

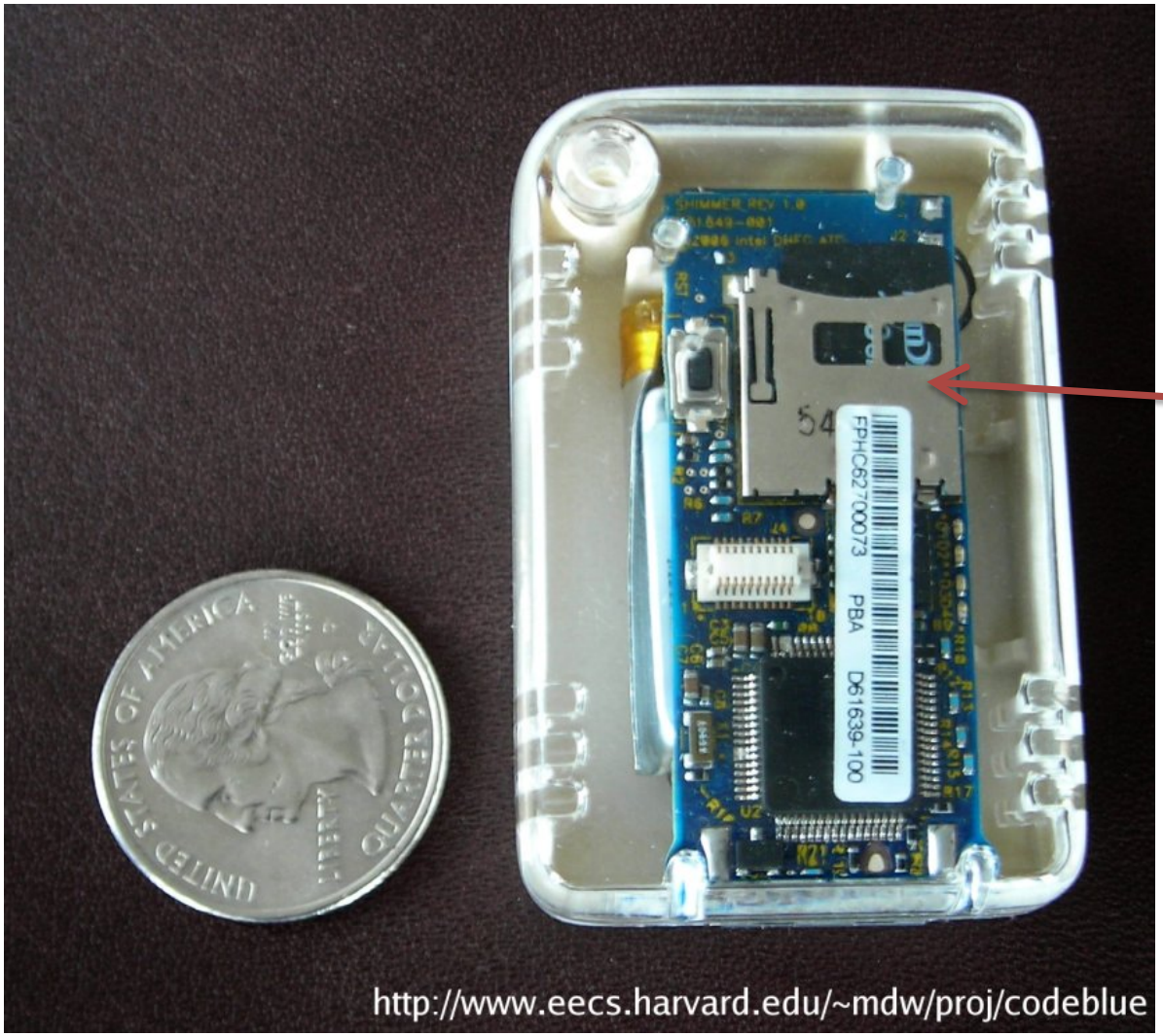
Step 2: Build confidence.

- Collect data with an eye towards making the scientists happy ☺
- Understand their needs and metrics for success.
- Be prepared to throw away many aspects of your design.

Step 3: Write a joint grant proposal.

- Incorporate research directions from CS and domain science
- Tell a story about your successes so far
- Build in mechanisms to ensure domain science success (e.g., log everything!)

One approach to universal happiness

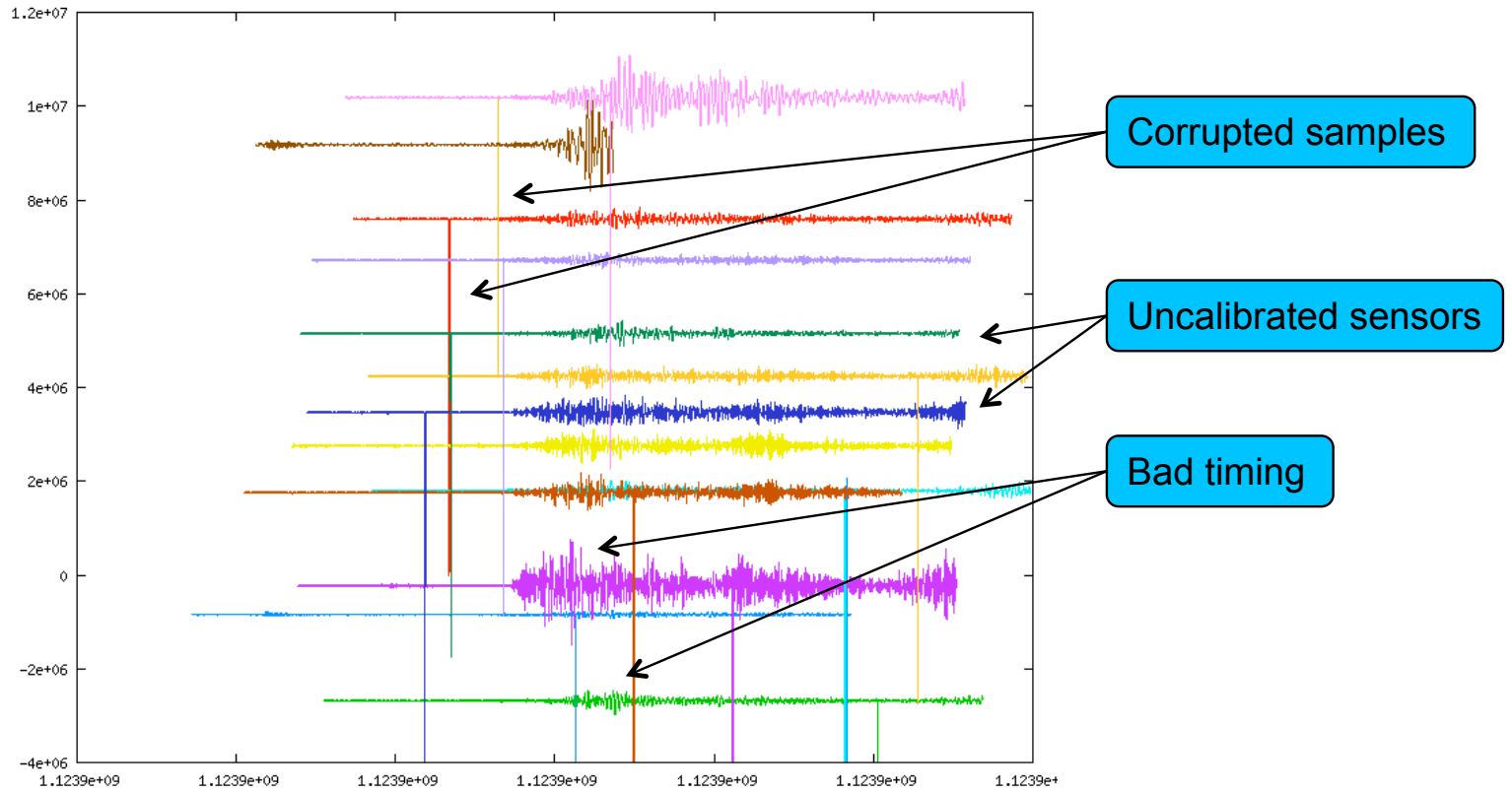


2 GB MicroSD flash (7 mA current consumption, 58 ms block write latency)

Lesson #3

Validate, validate, validate!

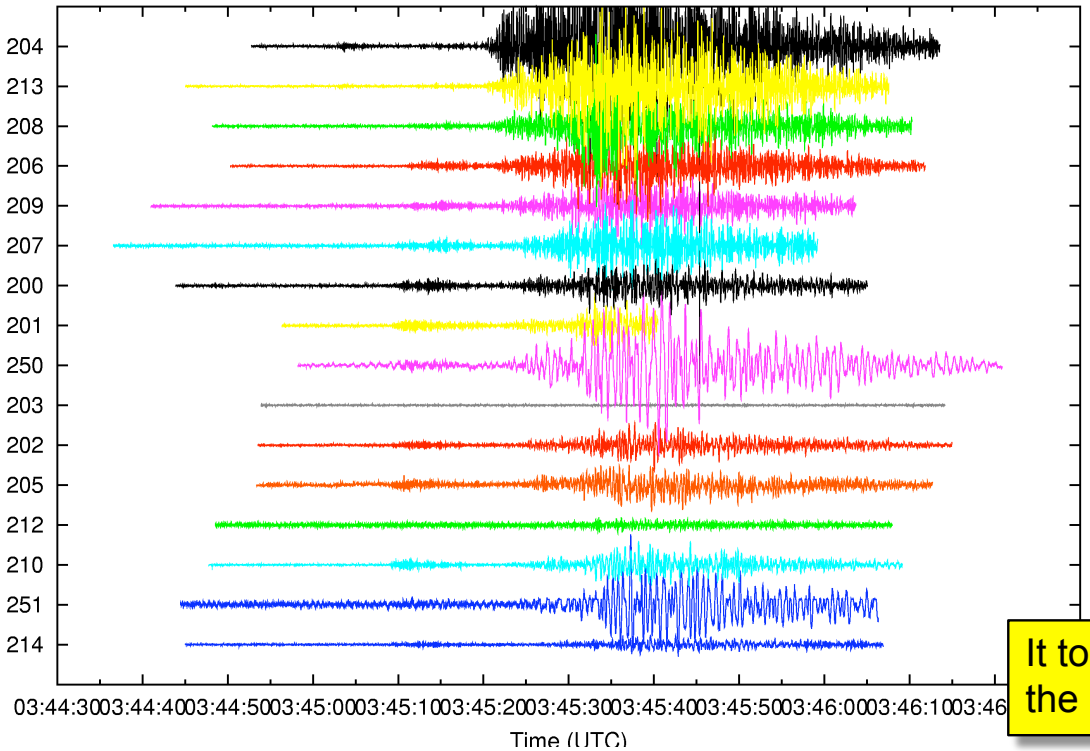
- Take **ground truth** data in the field



Lesson #3

Validate, validate, validate!

- Take **ground truth** data in the field



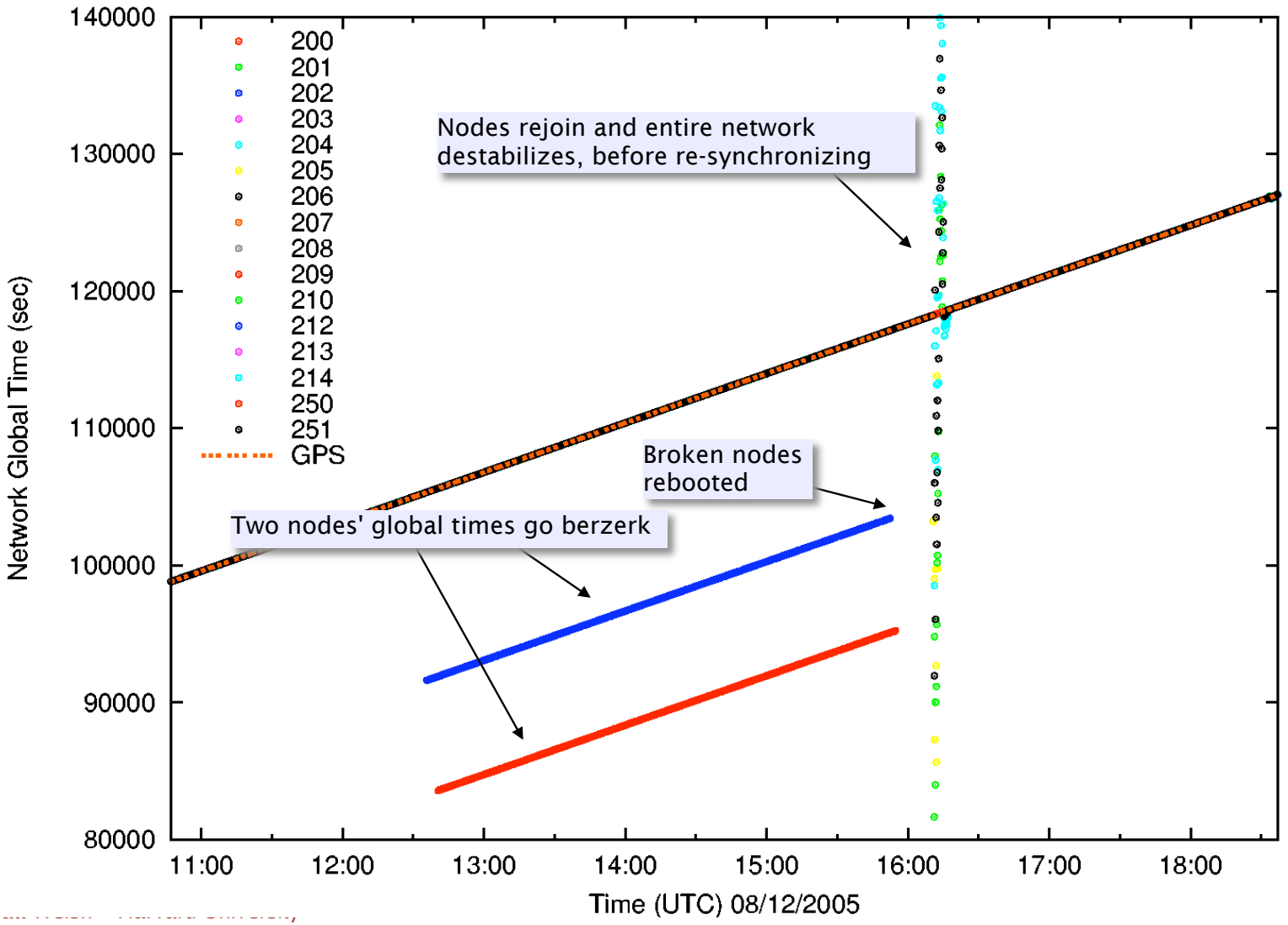
Bad samples removed

Data normalized

Timing corrected

It took 6 months to get the data cleaned up!

FTSP stability issues



FTSP stability issues

In FTSP, nodes periodically exchange information on global timebase

- Each node sends heartbeat messages containing MAC-delay corrected global timestamp
- Nodes use this information to calculate local clock skew w.r.t. global clock

Problem: Nodes can receive *corrupted* FTSP messages

- Bug in MSP430 clock driver caused nodes to sometimes read wrong time.
- FTSP trusts this information and includes it in local skew/offset calculations
  - Also, bad information can propagate throughout the network.

FTSP should perform internal consistency checks

- e.g., Avoid radically updating local phase/skew if well-synchronized already

Post hoc time rectification

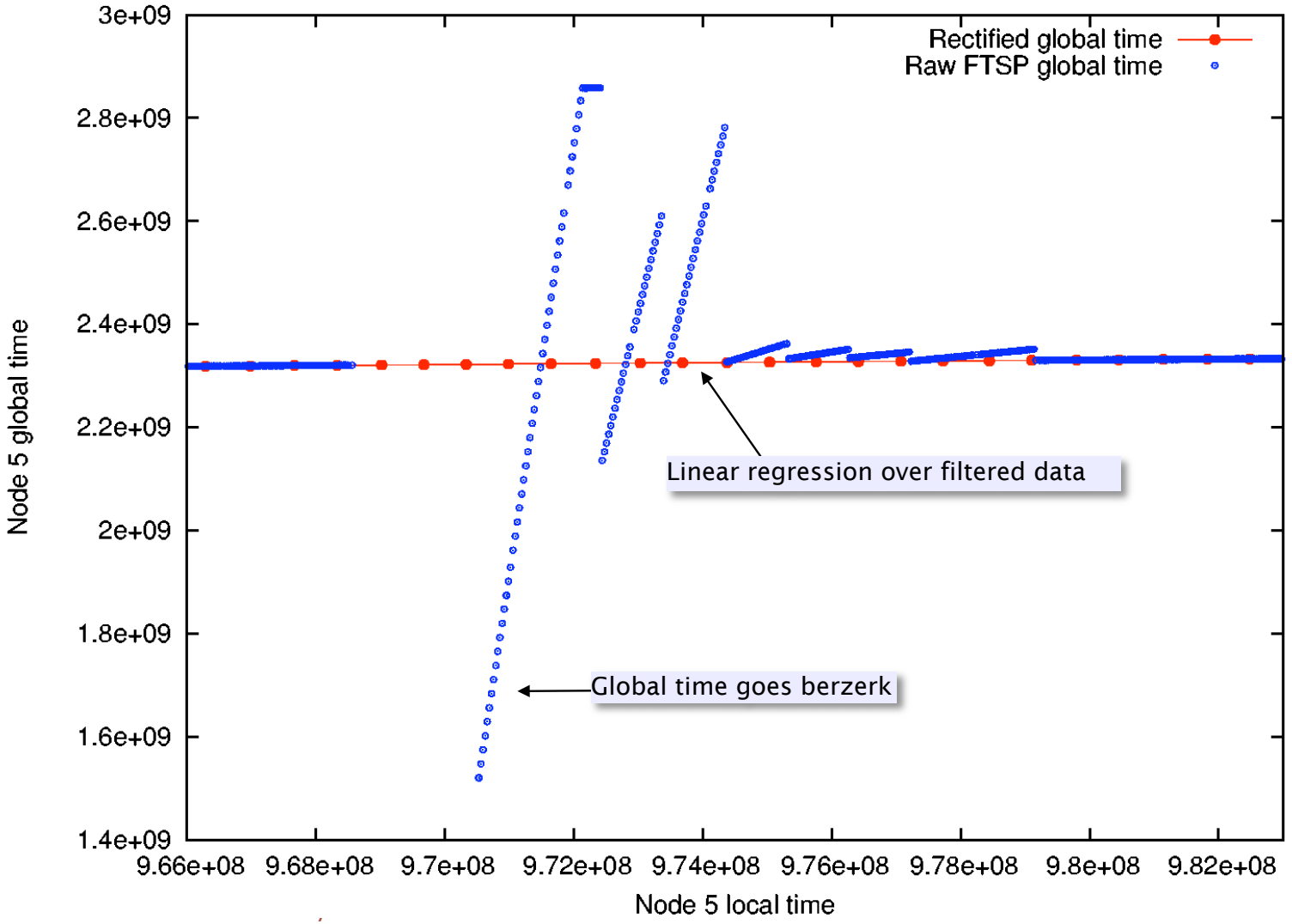
Must correct global timestamps in recorded data set

1) Filter out obviously wrong global timestamps
   - e.g., Those that differ significantly from global time recorded by FTSP root
   - Threshold used: 1 sec.

2) Perform piecewise linear regression on remaining data set
   - Produce a mapping from node's local time to “correct” global time
   - Linear regression extends for no more than 30 minutes

3) Apply this mapping to the recorded data for all events

Time rectification example



How do we know if we got the timing right?

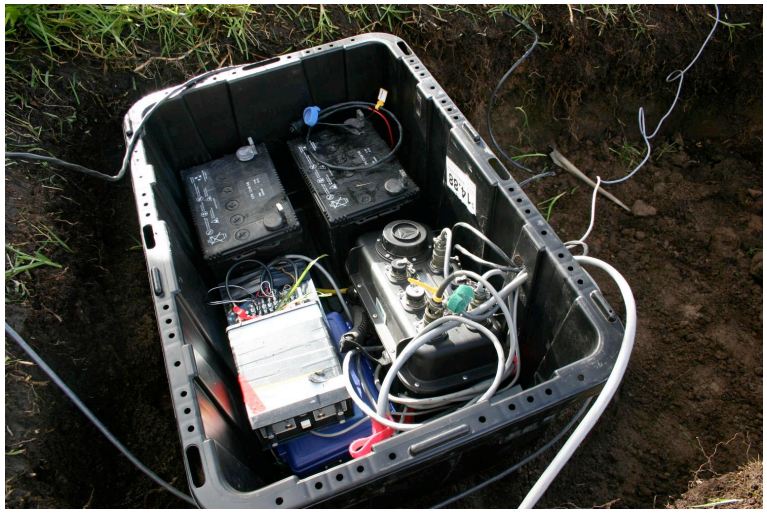
- Compare data captured by our mote to a datalogger nearby:

Data timestamped by **FTSP**

Node 213
(3 hops from FTSP root)

56 meters

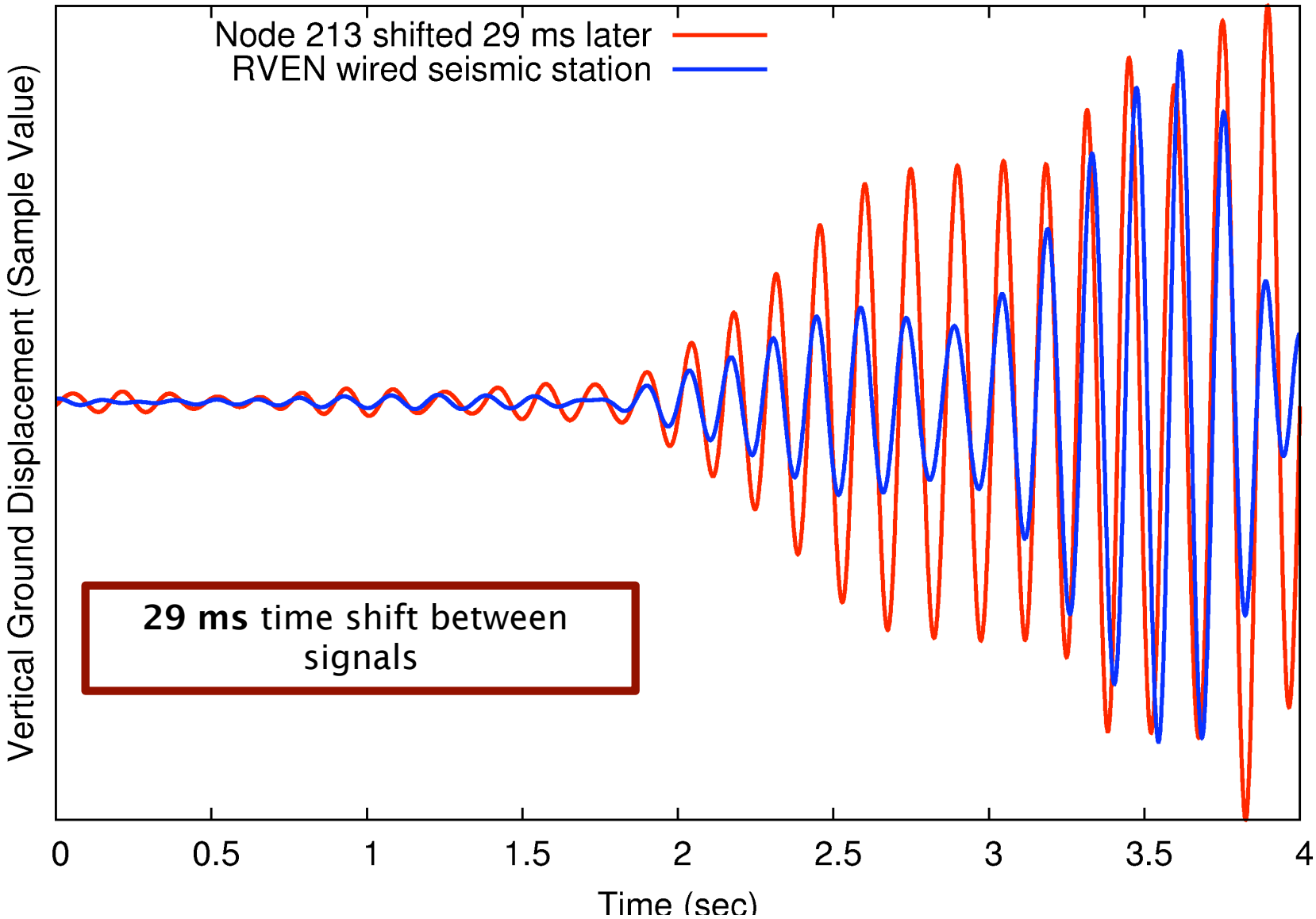
Data timestamped by **Internal GPS**



RVEN
Reventador **VEN**t
standalone seismic station

Largest expected offset: 47ms
(based on distance between stations)

Comparison with Reftek station



Sensor Networks as Scientific Instruments

If we want to treat sensor networks as scientific instruments, we need to subject them to rigorous validation.

- Domain scientists have little-to-no experience with WSNs.

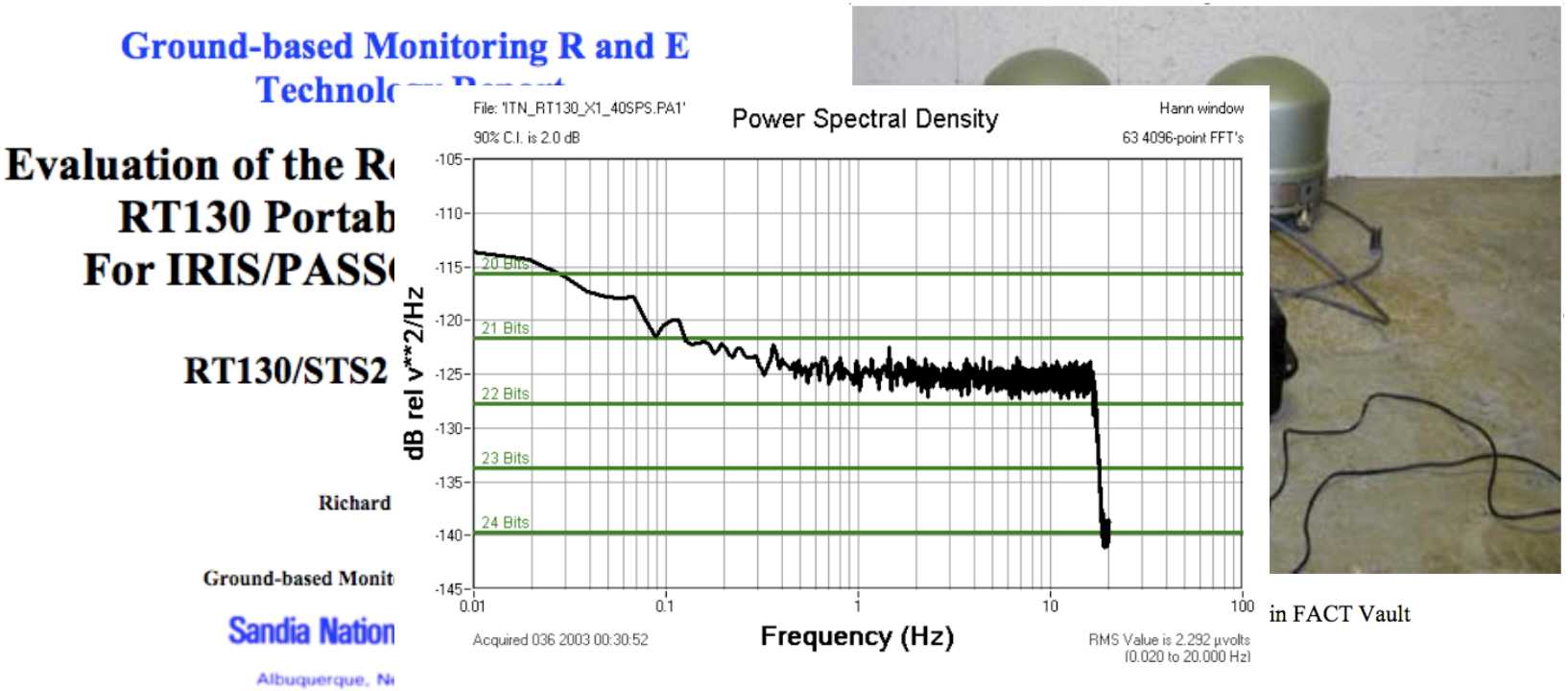


Figure 6.4.1 RT130 40SPS Channel 1 Input Terminated Noise

Sensor Networks as Scientific Instruments

If we want to treat sensor networks as scientific instruments, we need to subject them to rigorous validation.

- Domain scientists have little-to-no experience with WSNs.

This goes for the software, too!

- Any change to the code might invalidate the scientific results.
- Apart from testing, need mechanisms to track data provenance

Our approach: Log *everything*

- Flexible XML file format for recording everything:
  - *Metadata, raw sensor data, radio packets, GUI events, timestamps, …*
- Log all changes to software and use version control religiously
  - *Mote code has built-in version tag, sent in each status message*
- Log the raw bytes received at the radio level – not a tools' interpretation of the structure

Lesson #4

Don’t forget about the base station!

Lesson #4

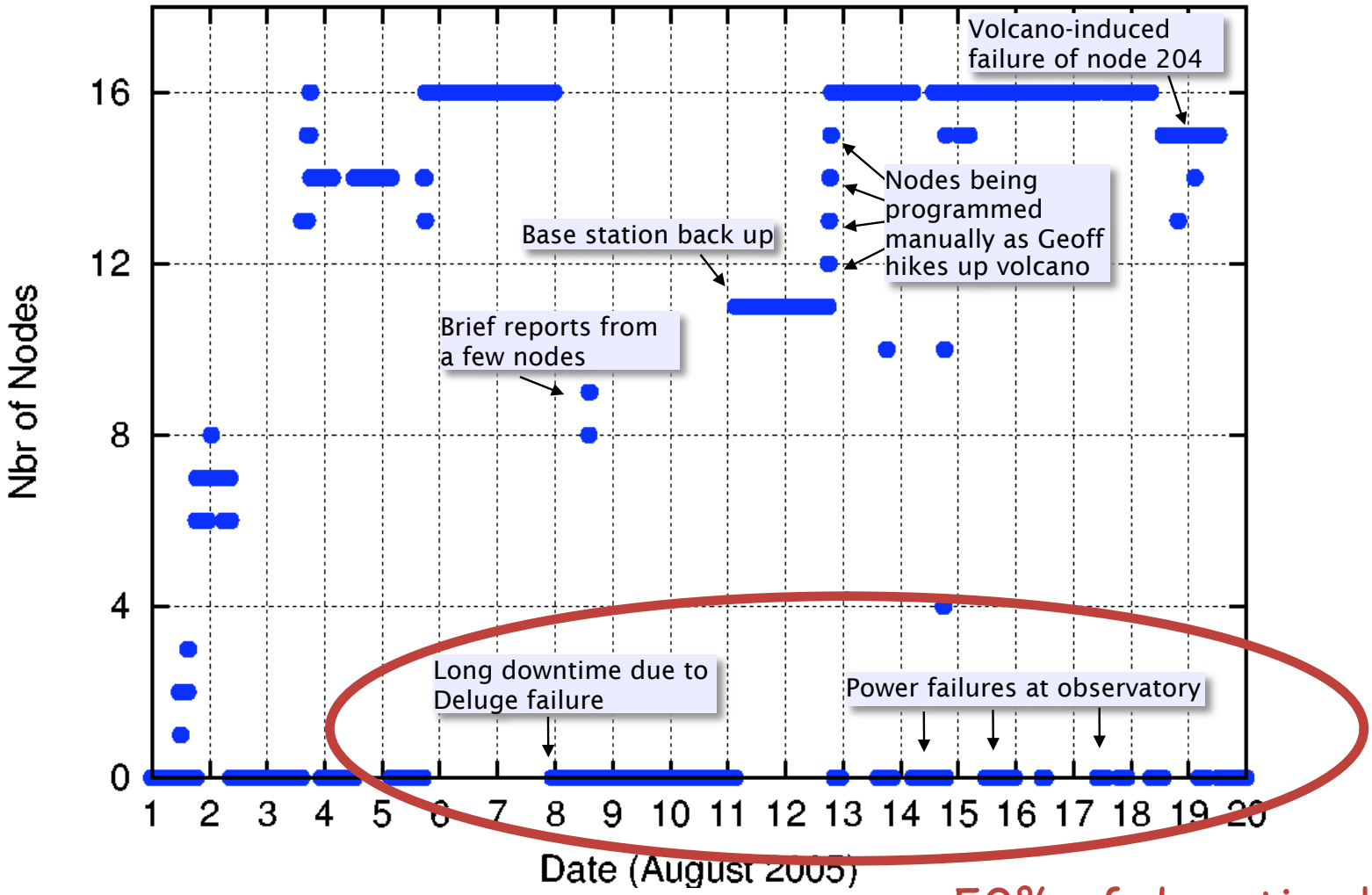
Don’t forget about the base station!

Electricity is not always so reliable (especially in developing countries)

Neither is the guy you hire to fill up the diesel generator every night…

Base station outages accounted for ~50% of our network downtime

Network reliability over time



50% of downtime!

The base station is non-trivial

We spent 90% of our engineering effort on the mote code.

Base station code was mostly an afterthought…

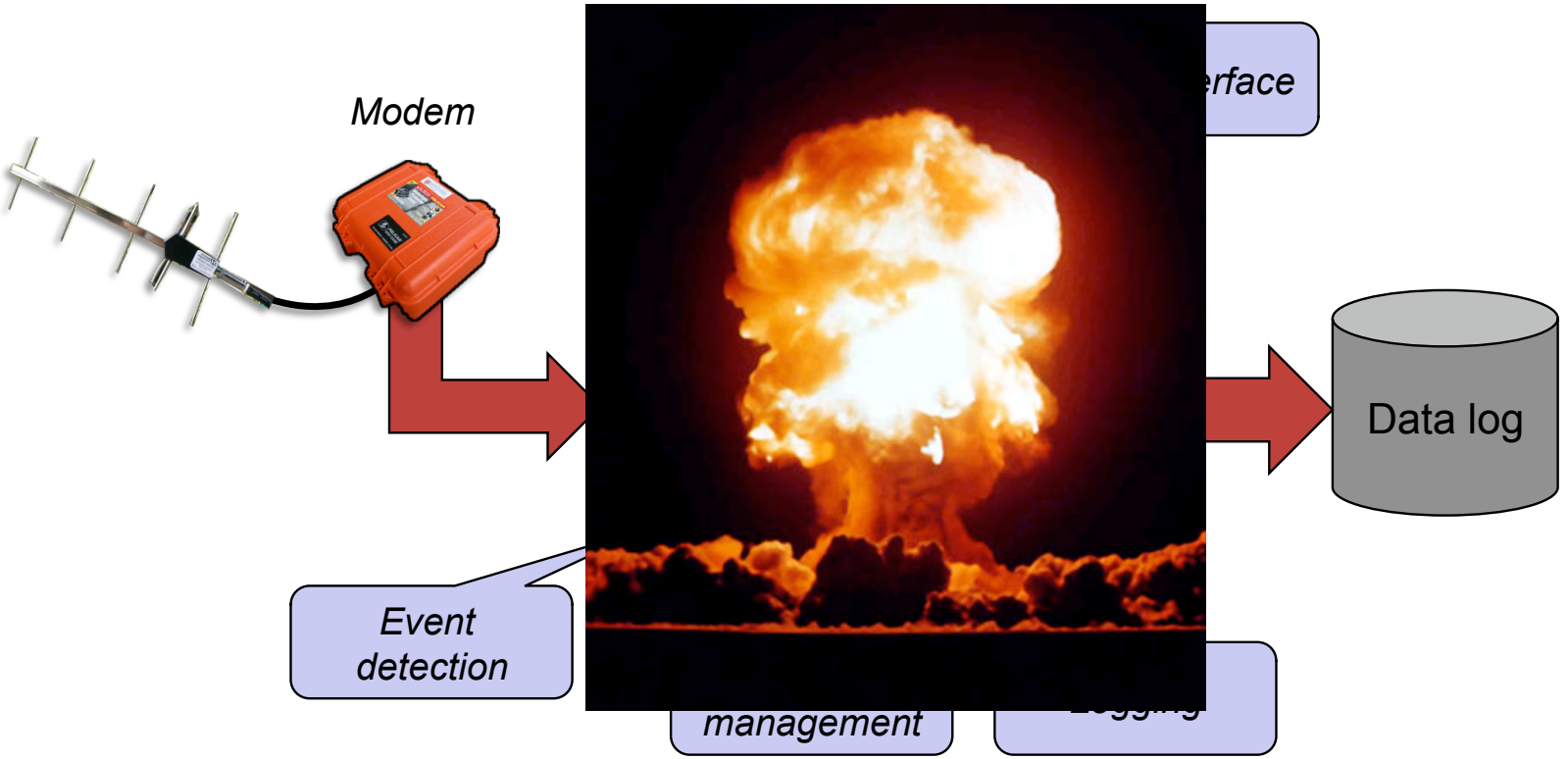
- Slapped together Perl scripts and Java GUI
- Lots of last-minute hacking to add features not anticipated before arrival
- All resting on top of the (somewhat fragile) SerialForwarder

One race condition in the Java GUI ➔ 8 hours downtime

- While we were sleeping, of course.

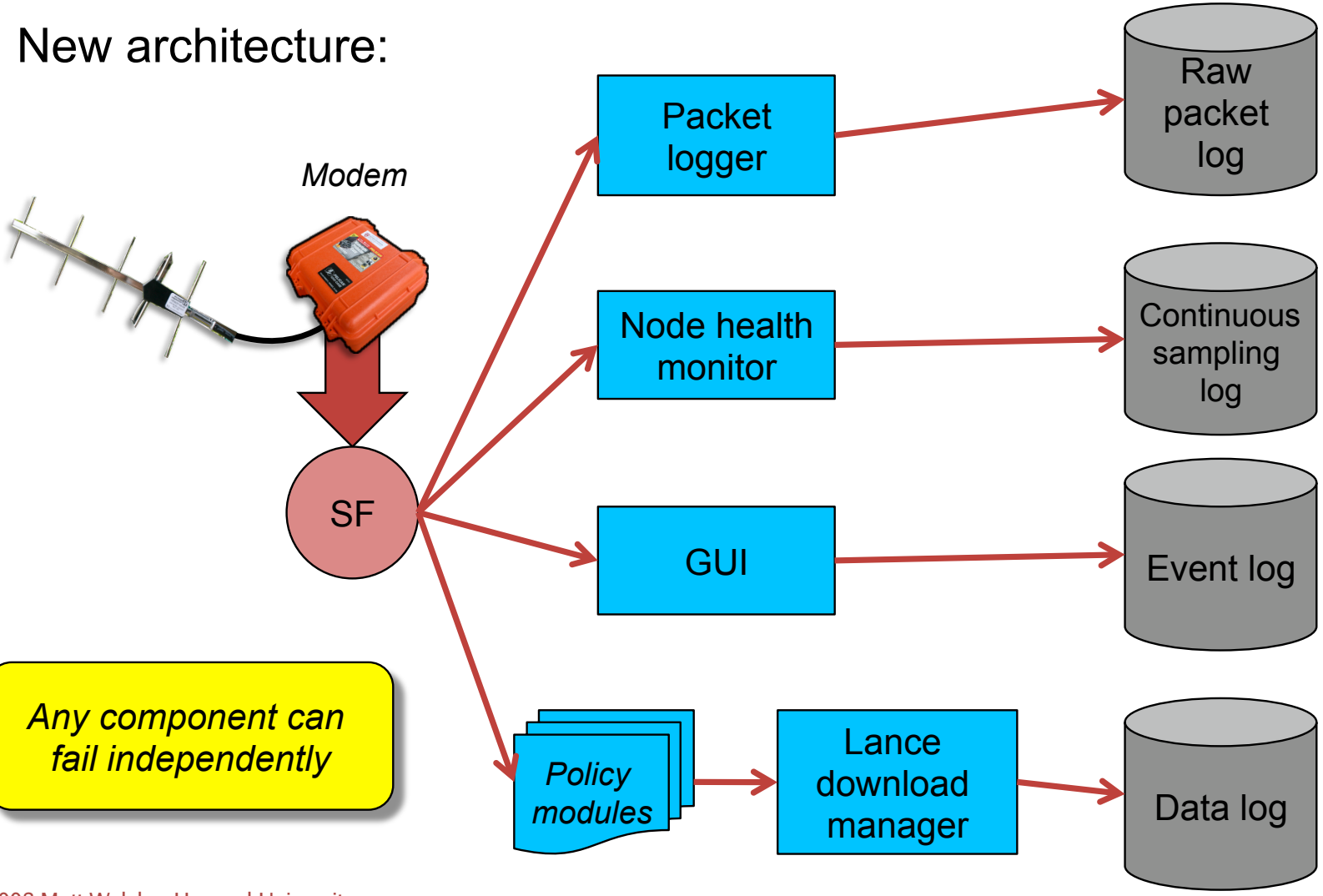
A better approach: Modularize!

Previous architecture:



A better approach: Modularize!

New architecture:



Closing the loop

90% of sensor network research has focused on the lowest layers:

- Hardware platforms
- MAC and routing protocols
- Time sync, localization, etc…

We're reaching a point where this stuff is "good enough"

- Time to start fielding existing solutions, learn what works in the real world

Closing the loop

90% of sensor network research has focused on the lowest layers:

- Hardware platforms
- MAC and routing protocols
- Time sync, localization, etc…

We're reaching a point where this stuff is "good enough"

- Time to start fielding existing solutions, learn what works in the real world

Example: Deluge [Berkeley]

- Nice protocol design
- Worked well when tested in the lab
- Fell apart in the field – probably due to sparse deployment
- No comments in the source code!

The role of applications

I believe that **all** sensor network research should be application-driven

After all, we have enough problems already

- No need to invent new ones.

This leads to some interesting questions…

Is it necessary to customize the software stack for each app deployment?

Is there any hope for general-purpose, reusable hardware and software platforms?

Some open questions

How can we enable scientists to program sensor nets directly?

- That is, rather than using a CS Ph.D. student as a proxy…
- Our approach: Macroprogramming

How do we hide complexity while achieving good resource efficiency?

- Knobs are evil, but perhaps a necessary one.
- 80% rule

How important is efficiency, anyway?

- Lifetime is an important metric, but not the only metric.
- Many scientists would be happy with a short-lived network that provided excellent data.

Where does this leave us?

What do scientists really want?

- Are we inventing funny-shaped hammers?

Many would be happy with cheap, portable, GPS-timestamped dataloggers…

- Real-time wireless telemetry is a distant second
- Consider the (slow) scientific process.

Emphasis on *sensor*, rather than *network*.

Onwards and upwards

The next decade of scientific discovery *must* be driven by advances in information technology.

The sciences are fundamentally computational.

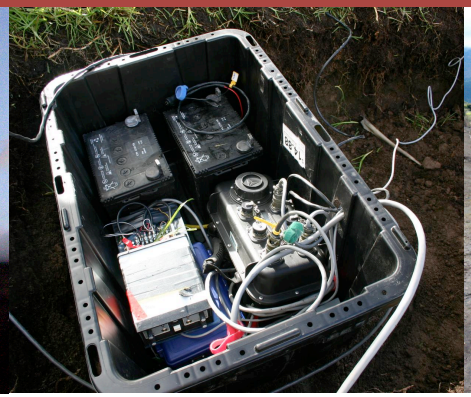
We have two choices about how to do this.

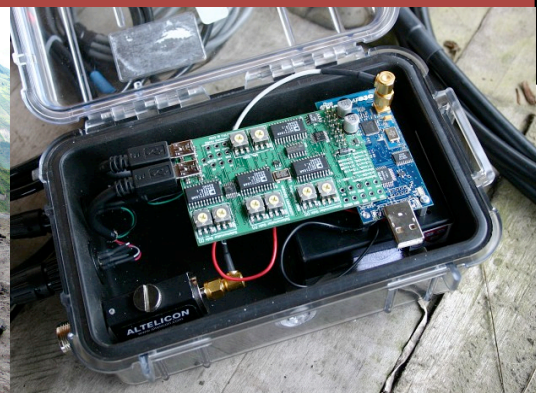
This requires more than technology!

- Politics, funding structures, tenure case reviews, academic incentives.

Sensor networks are but one tool in our toolbox.

Thanks!

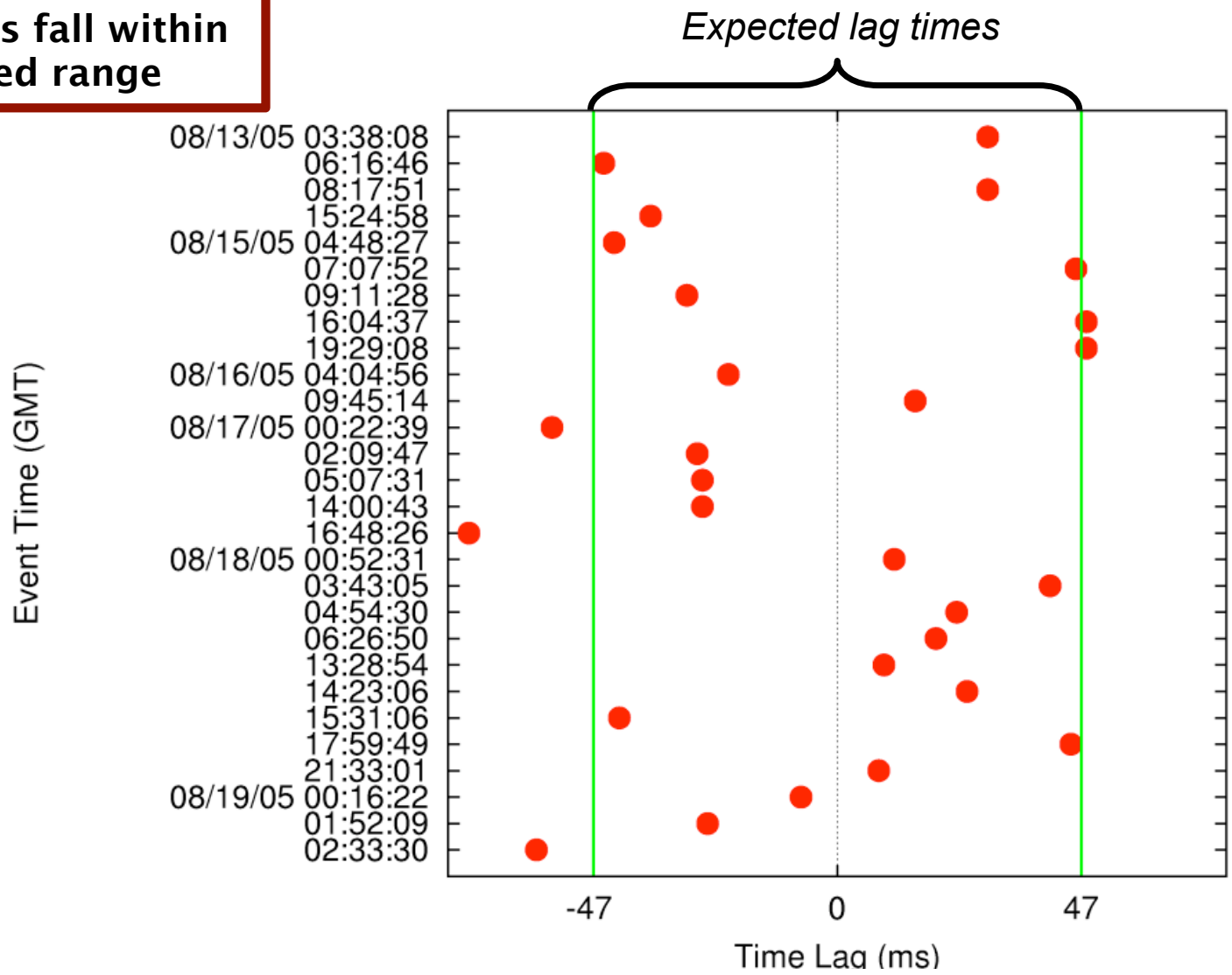



Special thanks to Geoff Werner-Allen, Konrad Lorincz, Stephen Dawson-Haggerty, Jeffrey Johnson (NMT), Jonathan Lees (UNC), Mario Ruiz (IGEPN), Omar Marcillo (UNH), Instituto Geofísico Ecuador, Microsoft, and the National Science Foundation

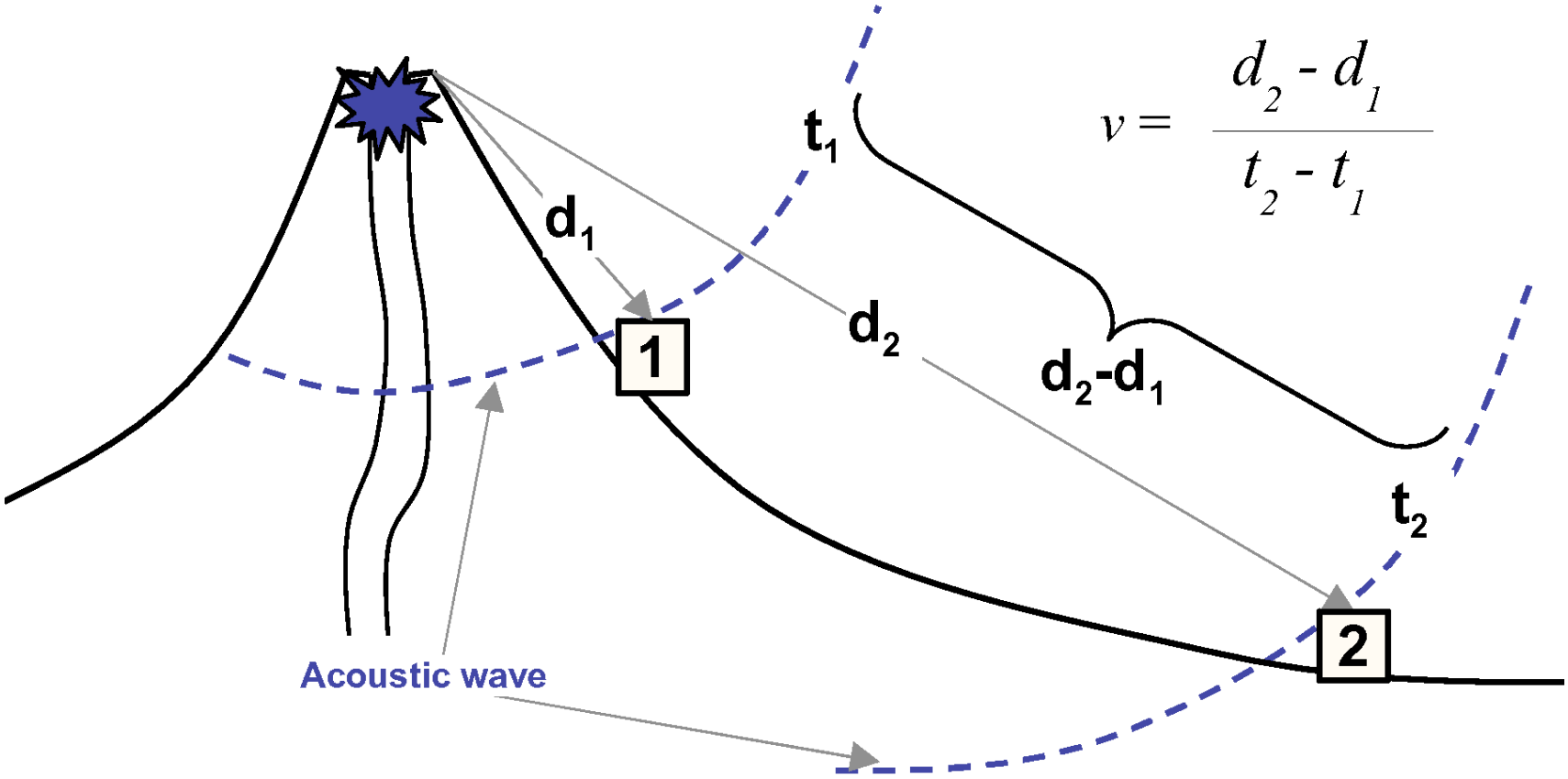
Time lags across multiple events

Most events fall within expected range



Data validation: Acoustic wave traversal

- Check the speed of acoustic waves propagating across the array!



Data validation: Acoustic wave traversal

Measured acoustic wave velocity is in the expected range (331-349 m/s @ 0-30˚C)