The COMO Project

Pere Barlet-Ros, UPC Barcelona
Gianluca Iannaccone, Intel Research Berkeley

Disclaimer

- This presentation focuses on COMO v2.0 that is currently under development (unstable-release available on sourceforge.net).
- COMO v2.0 stable release is expected by the end of June 2009 for pilot deployment on Onelab2.
- COMO is currently supported by Intel Research, UPC Barcelona and Quantavis.

Motivation

- Developing and deploying new network monitoring apps is **unnecessarily time-consuming**
- Familiar challenges
  - Need deep understanding of data sets (including details of the multitude of network devices)
  - Need tools to extract information of interest and evaluate accuracy and resolution of data (e.g., timestamps, completeness of data, etc.)
  - Need to efficiently perform computation across multiple data streams coming from different locations
- …and all this prior to building the actual applications!

Motivation (cont'd)

- Familiar consequences
  - Developers tend to find shortcuts
  - Develop ad-hoc solutions for each application
  - Slow development and even slower deployment
  - Hard to maintain and operate
  - Hard to validate that new monitoring service does not interfere with existing infrastructure

What is COMO?

- Platform for Fast Prototyping Network Monitoring Applications
  - Rich API for quick deployment
  - Support for many different monitoring devices
  - Efficient handling of available system resources
  - Query engine to retrieve data, connect multiple monitors and perform distributed measurements

Features

1. Abstract away device internals
   - Applications can run *without modification* over a variety of network devices and data formats
   - Support for netflow/sflow/radio/DAG/etc.

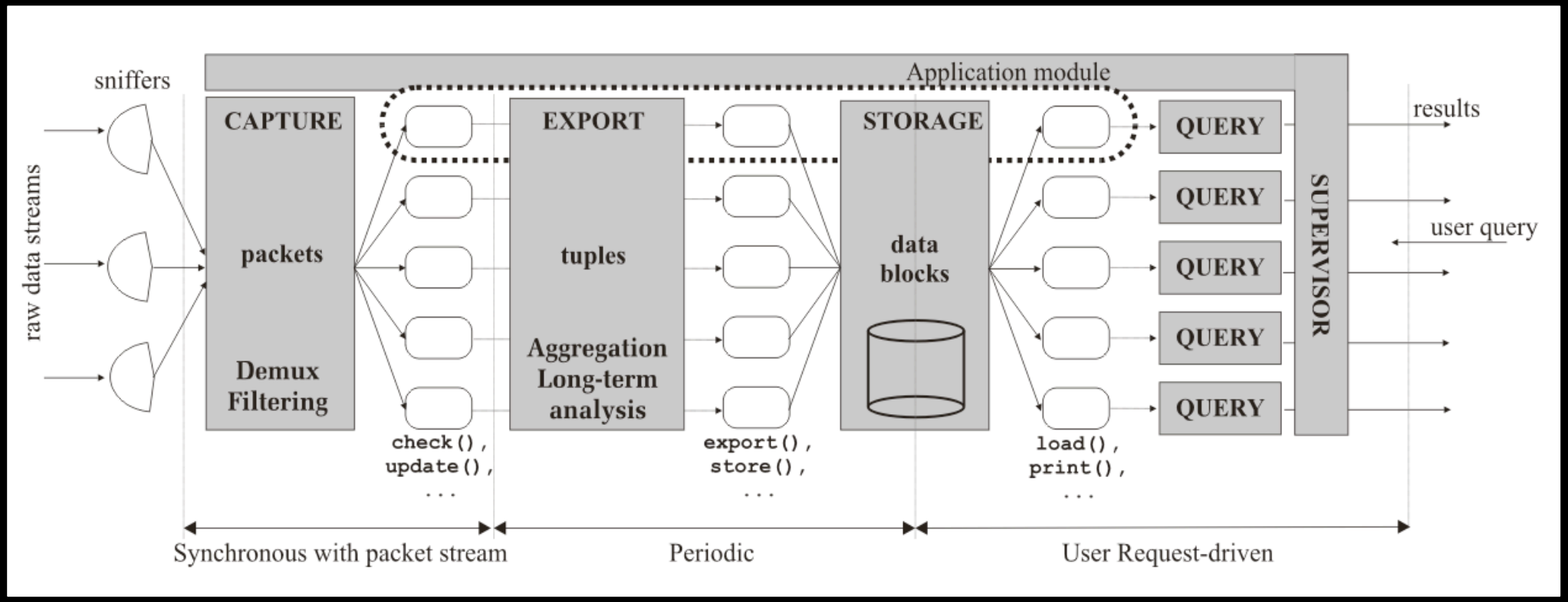
2. Automated Resource Management
   - System handles overload through packet/flow sampling
   - Applications notify system of maximum sampling rate they can sustain (if any)
   - Applications will be informed of sampling applied to packet stream and may correct results accordingly
   - Applications are turned off if sampling is not an option

Features (cont'd)

3. Easy Distribution and deployment
   - Applications come in stand-alone plugins
   - Can be deployed on any monitoring device
   - The system will run them if the data the application needs is available (e.g. wireless-specific monitoring app on a netflow stream…)

4. Support for many programming languages
   - Plugins written in C if performance is important
   - Otherwise, they can be written in any language (with some restrictions)
   - In June we will release python-specific API

COMO basics



EXPORT/STORAGE can be replicated for load balancing

CAPTURE is the main choke point. It periodically discards all state to reduce overhead and maintain a relative stable operating point

How it works (for developers)

- Write a COMO plug-in module per application
  - Five components:
    init, capture, transform, store, query
  - **capture** operates on real-time "packet stream"
    - It receives packets as input and generates tuples out
    - "packet stream" is a common representation of incoming network data (it may be IP packets, 802.11 frames, netflow records, ascii log entries or the output of other modules)
    - **capture** must be efficient. It's always written in C.
    - Being on the critical path of the system it may be turned off if too expensive
    - Library of pre-defined, optimized **capture** modules will be available (e.g., flow classification, pattern search)

How it works (cont'd)

- Other components can be in other languages
  - **transform** receives periodic set of tuples from capture and generates tuples out to **store**
  - **store** generates/selects the set of tuples to be saved on disk
  - **query** reads tuples from disk (or directly from store) and returns a representation as requested by the user (e.g., ASCII, graph, script to plot a graph, etc.)

Example: how to compute top-k destination addresses

- **init** reads config (to set k and the measurement interval)
- **capture** aggregates traffic by destination and periodically sends a stream of {destination, bytes} tuples
- **transform** aggregates the tuples over the measurement interval and outputs a ranked list of {destination, bytes} tuples
- **store** select the first k tuples and saves on disk one tuple with the timestamp and the original {destination,bytes} pairs
- **query** fetches the tuples by timestamp and returns a JSON representation of the top-k destinations

... but this is just one way to do it

Onelab Deployment

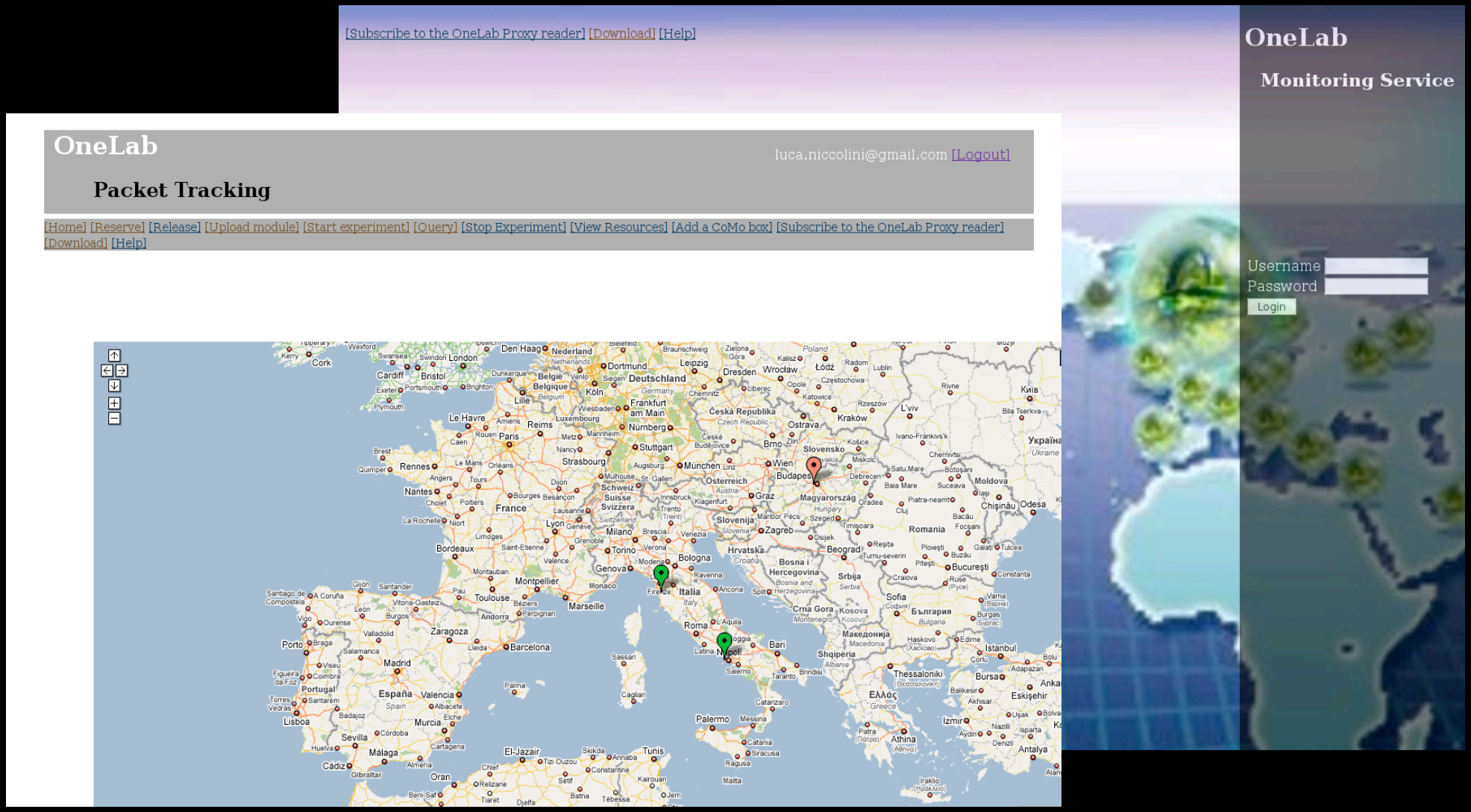
- Monitoring facility for Planetlab Europe
  - Deploying over 20+ institutions across Europe
  - Onelab users can test monitoring applications on traffic generated by Onelab/Planetlab nodes.

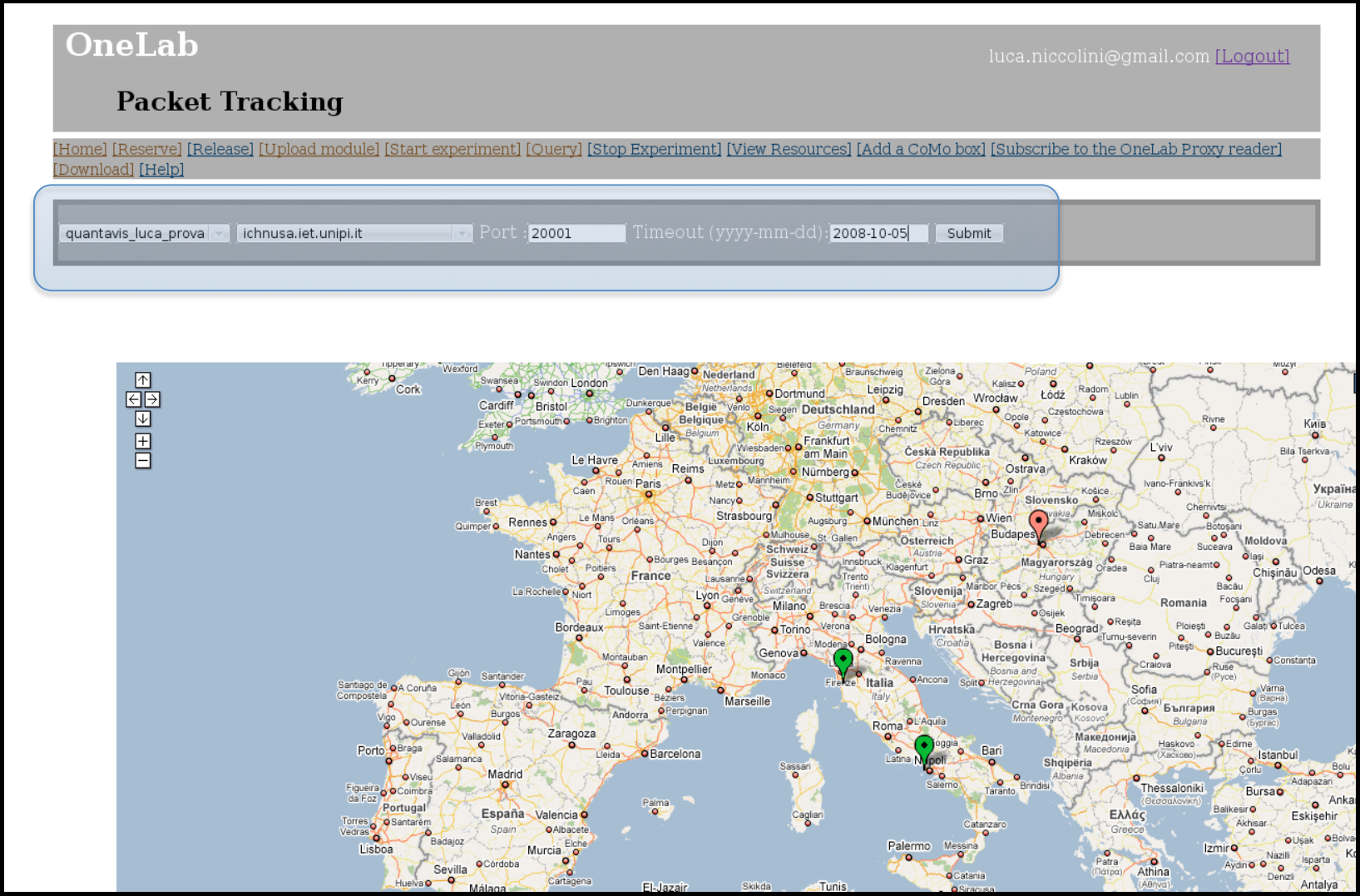
- *Infrastructure for demonstrating, testing and sharing monitoring applications before deployment in production environments*

Onelab Usage Model

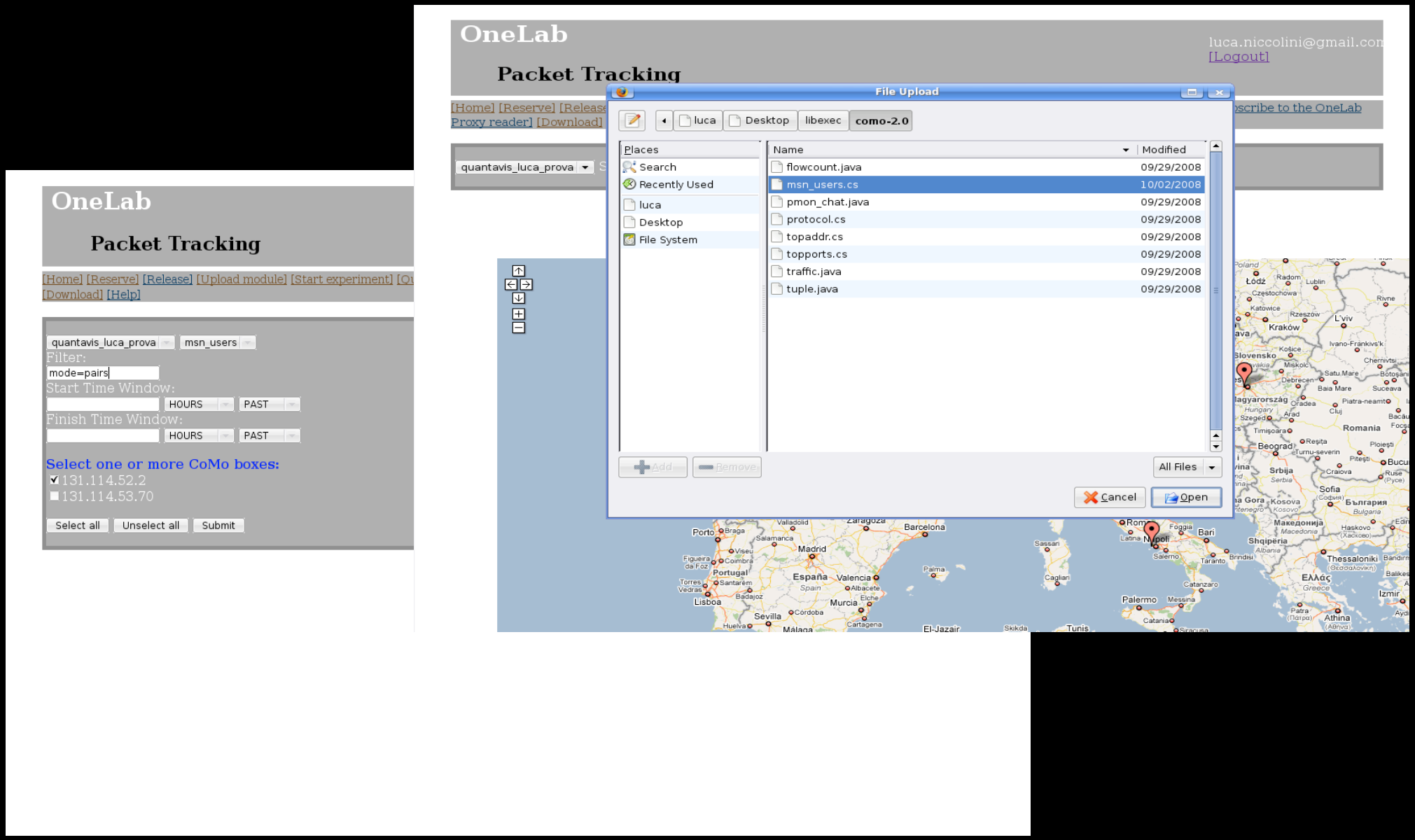
- Users can deploy monitoring apps everywhere
- Users can monitor only the traffic *their own applications* generate or receive
  - Not possible to monitor other users' traffic
  - Requires registration of experiments with Onelab monitoring service.
- COMO modules undergo review from Quantavis before deployment if written in C.
  - Other languages don't need review
  - Review process is confidential

Onelab web interface to register users





Users register experiment and receive list of allowed port numbers (i.e. UDP/TCP packets on those port numbers will be monitored)



Users can upload modules to (and receive results from) all COMO nodes from the web interface

Documentation

- Badly lacking… sorry.
- First draft will come with June 2009 release
- Visit http://como-project.org for updates (launch date: June 2009)

References

- *Robust Network Monitoring in the presence of Non-Cooperative Traffic Queries*
  Pere Barlet-Ros, Gianluca Iannaccone, Josep Sanjuas-Cuxart, Josep Sole-Pareta
  Computer Networks, February 2009

- *Load Shedding in Network Monitoring Applications*
  Pere Barlet-Ros, Gianluca Iannaccone, Josep Sanjuas-Cuxart, Diego Amores-Lopez, Josep Sole-Pareta, USENIX Annual Technical Conference, June 2007

- *Reformulating the monitor placement problem: Optimal Network-wide Sampling*
  Gion Reto Cantieni, Gianluca Iannaccone, Chadi Barakat, Christophe Diot, Patrick Thiran, ACM CoNEXT, December 2006

- *MIND: A Distributed Multi-Dimensional Indexing System for Network Diagnosis*
  Xin Li, Fang Bian, Hui Zhang, Christophe Diot, Ramesh Govindan, Wei Hong, Gianluca Iannaccone, IEEE Infocom 2006

- *Fast Prototyping of Network Data Mining Applications*
  Gianluca Iannaccone, Passive and Active Measurement Workshop, March 2006

http://como-project.org

June 2009