# Cerebro: Presence Protocol for Large Mesh Networks

Polychronis Ypodimatopoulos
Viral Communications group
MIT Media Laboratory

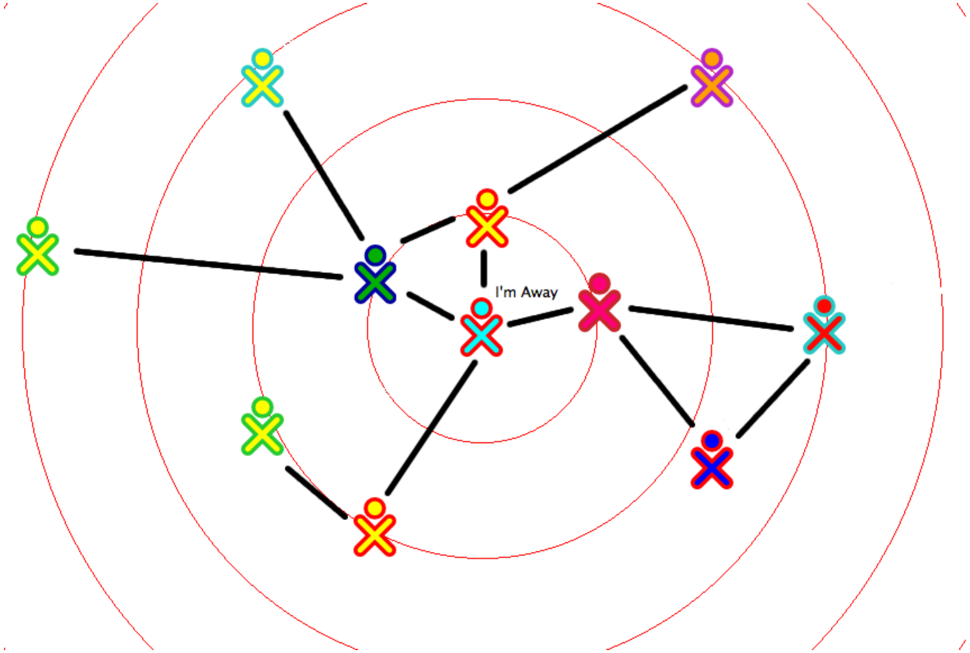
April 2008

# Project goal

- Organize the presence, profile and social interaction of humans and objects in the same physical area and make this information accessible and useful

# Presence Protocol Overview

- A node receives presence beacons from its neighbors, eliminates duplicate information, creates a new beacon that combines information the node already has with information it received and broadcasts a new beacon to its own neighbors.



# Beacon Frame contents

| Node ID | Witness | Distance | Serial No. |
| --- | --- | --- | --- |

**Node ID:** ID of the node this entry is about (MAC address, 6 bytes)

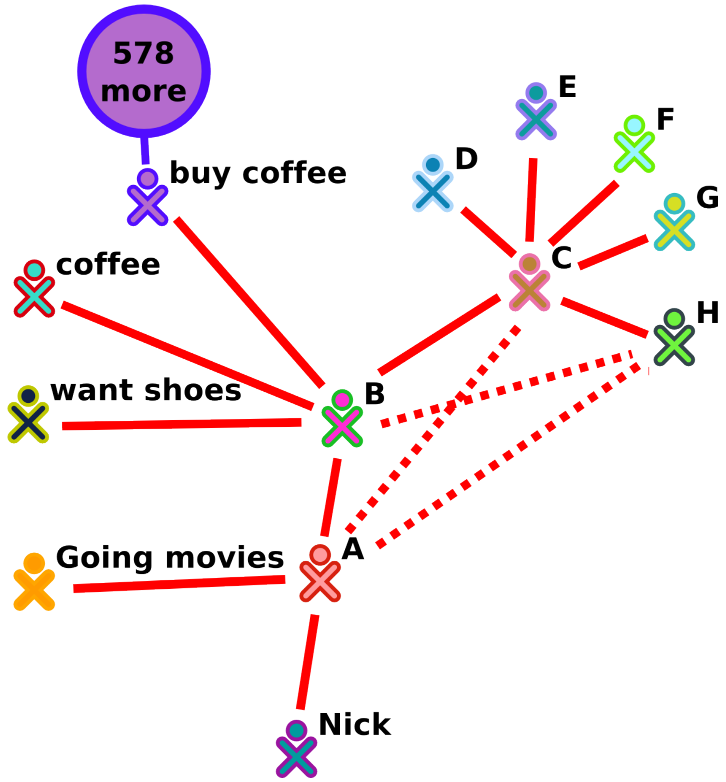
**Witness:** ID of the node from which this entry was received (MAC address, 6 bytes)

**Distance:** Distance from the current node to `<Node ID>` (similar to ETX metric) (1 byte)

**Serial No.:** Unique ID for each entry published by node `<Node ID>` (1 byte)

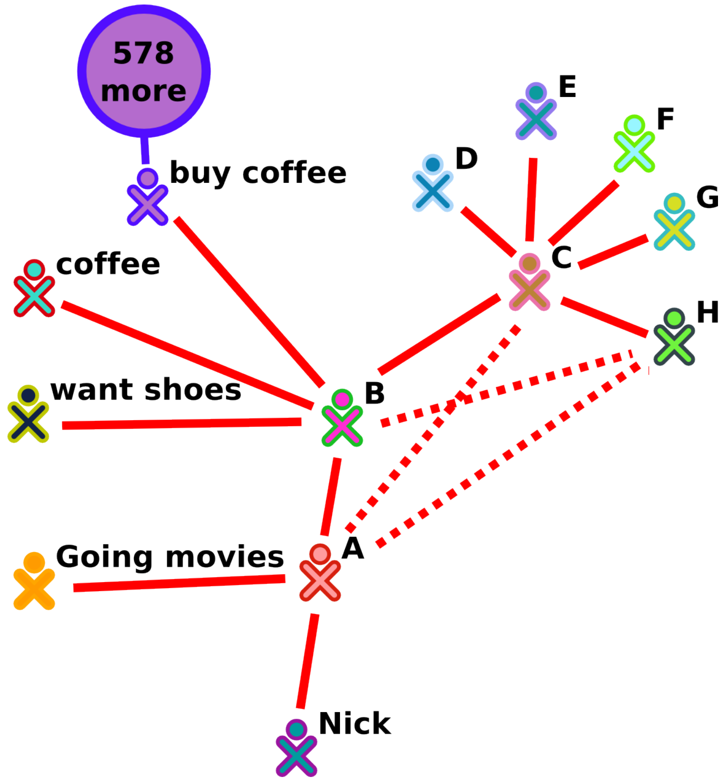
# Beacon Frame contents

Presence table at **A:**




| Node ID | Witness | Distance |
|---|---|---|
| B | A | 1 |
| C | B | 1 |
| C | A | 2 |
| D | B | 3 |
| E | B | 3 |
| F | B | 3 |
| G | B | 3 |
| H | B | 3 |
| H | A | 5 |

# Beacon Frame contents




Presence table at **B:**

| Node ID | Witness | Distance |
|---|---|---|
| A | B | 1 |
| C | B | 1 |
| D | C | 2 |
| E | C | 2 |
| F | C | 2 |
| G | C | 2 |
| H | C | 2 |
| H | B | 2.5 |
| C | A | 3 |

# Presence Protocol

1) Wait for a period T for beacons from neighboring nodes
2) For every beacon received:
   - ✓ For every entry in beacon:
     - if it is about the current node, or the current node is the witness, or the the serial number is not newer than the existing one, discard it!
     - if node/witness pair exists in presence table update number of arrivals and next arrival estimate both for node and witness
     - else add new entry in the table
3) Eliminate stale entries in presence table (ie. entries where the next arrival estimate has lapsed)
4) Create a new beacon using the minimum distances to each node in presence table
5) Broadcast beacon to neighbors

# Arrival estimates

1) Count arrivals over time period T.
2) Formulate a Poisson arrival rate (assuming arrivals are independent events)
3) Estimate time of next arrival with 90% accuracy

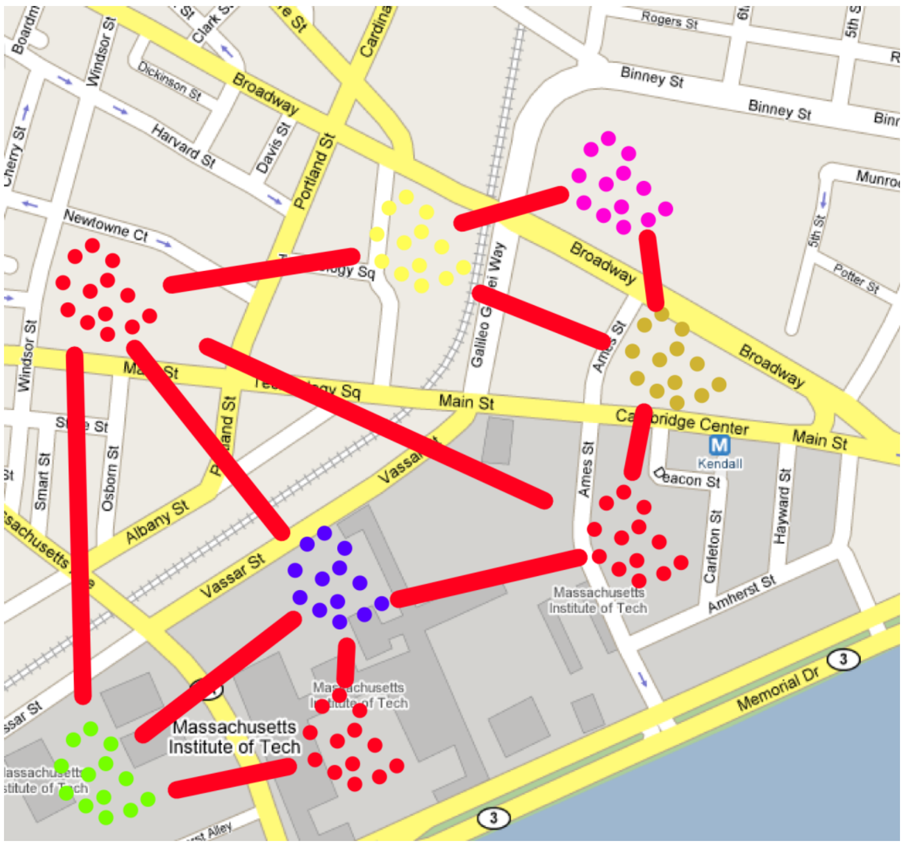
Example:

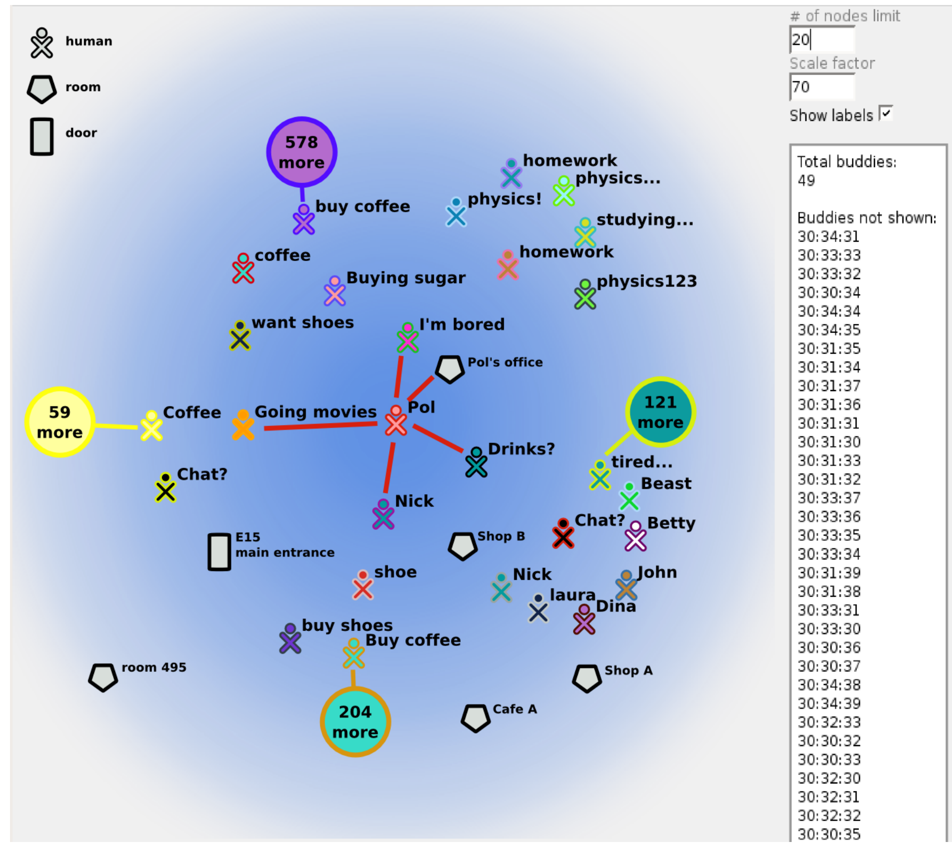
For T=1sec, accuracy=90%, next arrival in 2.3sec

For T=1sec, accuracy=100%, next arrival in infinity (!)

# Proposed solution: Cerebro

- Multiple mesh networks tunneled together form a “Parallel Internet”



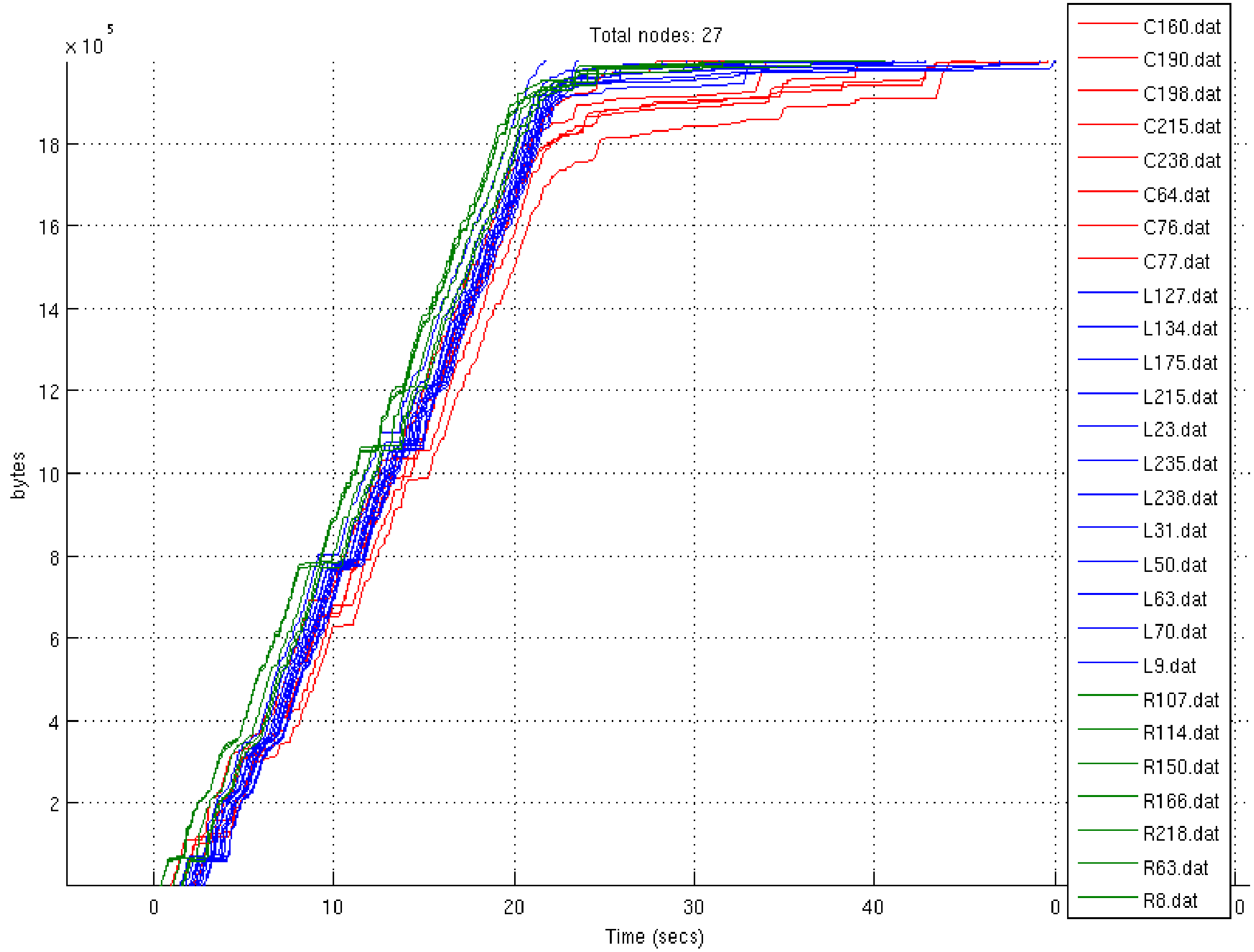
# Features

- Achieving scalability “on a diet”: Connected 100 nodes in mesh network using a single frame per node, per 10 seconds (15kb/sec in the worst case)
- Adjustable beacon rate based on node mobility

```
Overhead = N*B/T = 15*N^2/T = 15*N/t, T = N/t, t = 10Hz
```

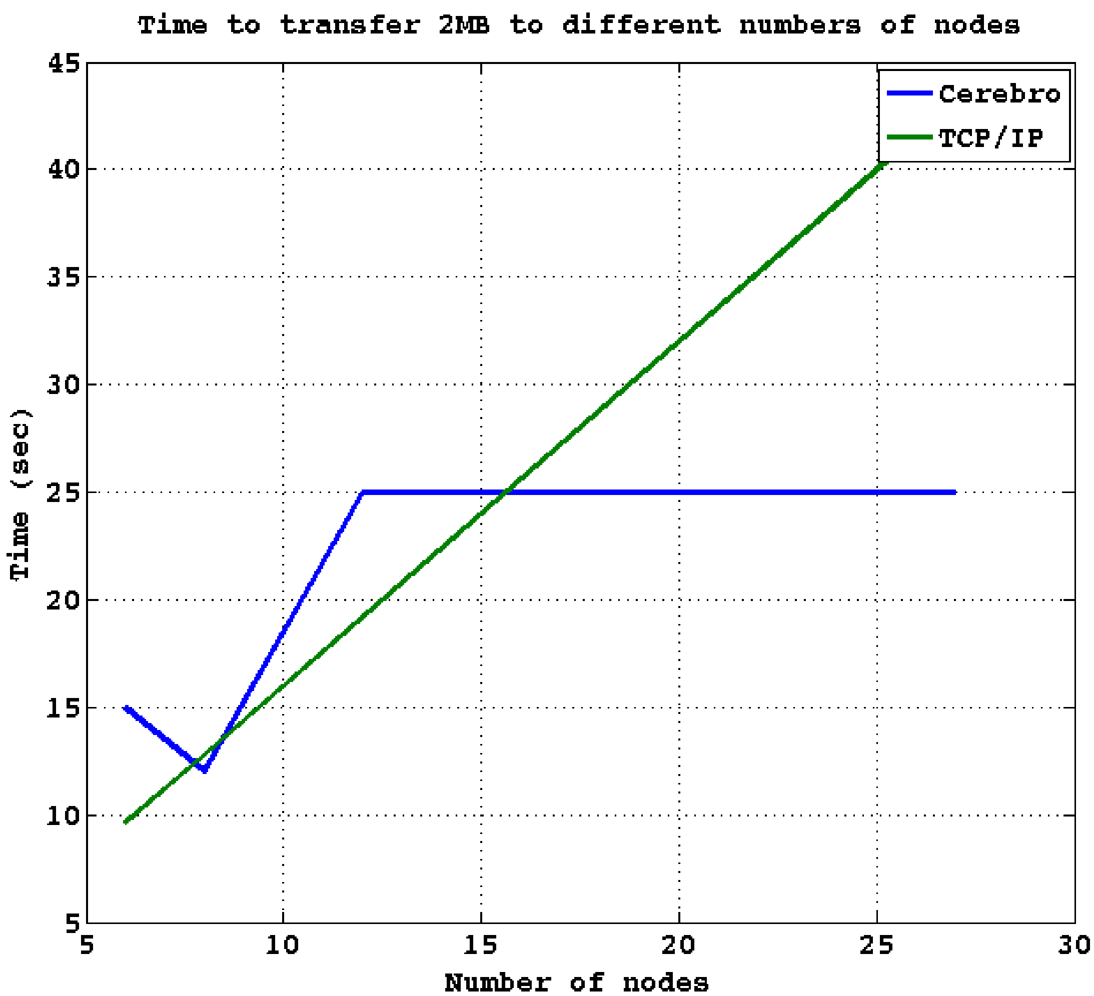
- Different beacon rates in different areas of the mesh network, according to node concentration
- Portability: Cerebro runs on x86, OLPC XO, Nokia N800 and ARM-based embedded computers (python)
- Routing protocol for communication with any other node in the network

# Numbers 1/2



Total nodes: 27
x 10^5
bytes
18
16
14
12
10
8
6
4
2
0
10
20
30
40
Time (secs)
C160.dat
C190.dat
C198.dat
C215.dat
C238.dat
C64.dat
C76.dat
C77.dat
L127.dat
L134.dat
L175.dat
L215.dat
L23.dat
L235.dat
L238.dat
L31.dat
L50.dat
L63.dat
L70.dat
L9.dat
R107.dat
R114.dat
R150.dat
R166.dat
R218.dat
R63.dat
R8.dat

# Numbers 2/2




Cerebro: 1Mbps (broadcast) simultaneous transfer.

TCP/IP: sequential, 10Mbps (actual) transfer is assumed to target nodes.

# Interfacing with Cerebro

- Based on DBus
- Methods currently with DBus interface:
  - `register: register activities/apps`
  - `push_data(destinations, data, port)`
  - `onNodeArrival(node_array)`
- Methods without DBus interface (yet):
  - `set_status_info, get_status_info`
  - `request_data(dest, port, request="")`
  - `request_multicast_data(destinations, port, request="")`
  - `push_multicast_data(destinations, port, data)`
  - `handle_new_req(src, port, req_payload)`
  - `handle_new_data(data, fhash, src, port)`
  - `onNodeLeave`

# Demo (or Die!)

- Chat
- `/tree`: shows network tree rooted at your XO
- `/sendfile`
- `/getstats`
- `/savestats`