Real Social Networks

The SocioPatterns Sensing Platform

Department of Computer, Control, and Management Engineering "Antonio Ruberti"

Sapienza University of Rome

Francesco Ficarola

(ficarola <at> dis.uniroma1 <dot> it)

Rome, Italy – March 19, 2013

Real Social Networks

The SocioPatterns Sensing Platform

Toward a more social world... or not?

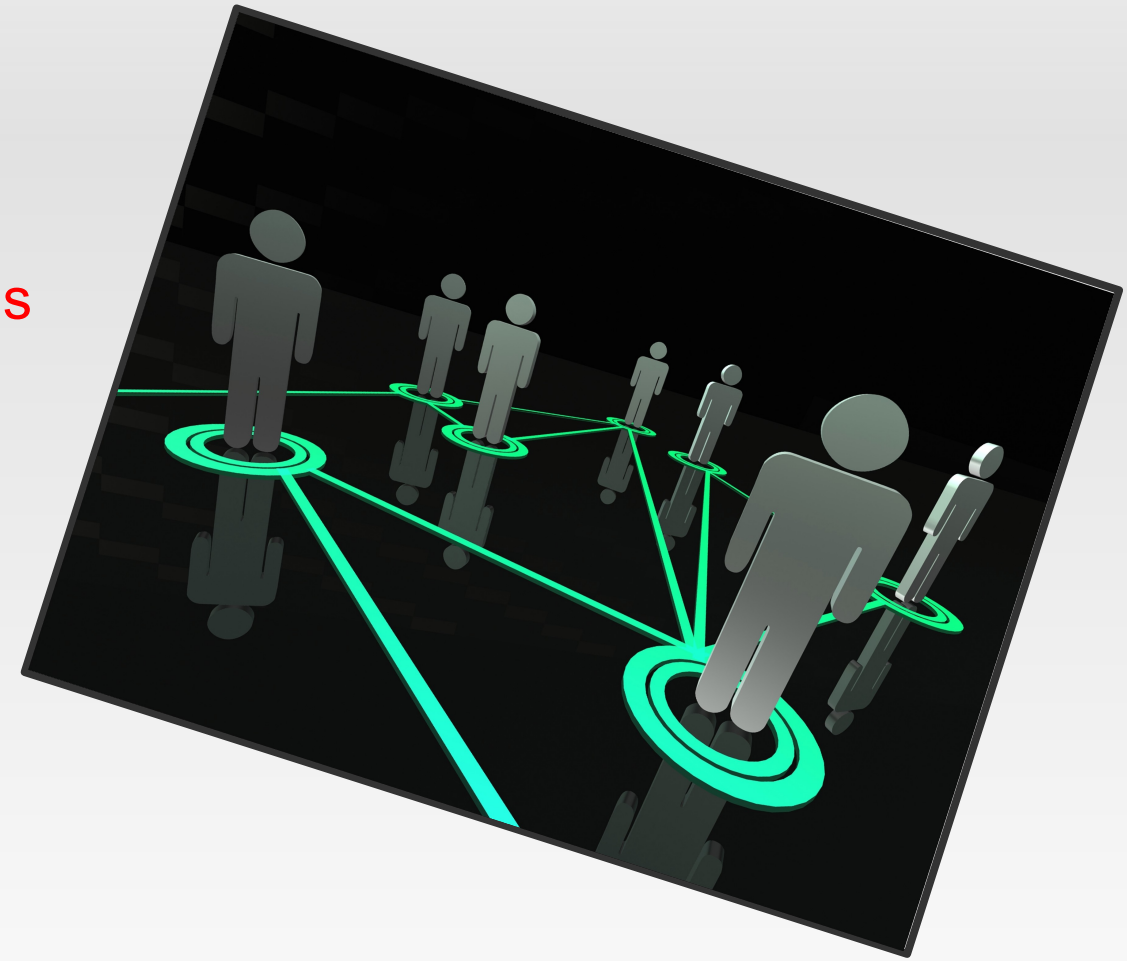
What is a Social Network?

A lot of definitions...

1. a network of social interactions and personal relationships.

2. a dedicated website or other application which enables users to communicate with each other by posting information, messages, images, etc..

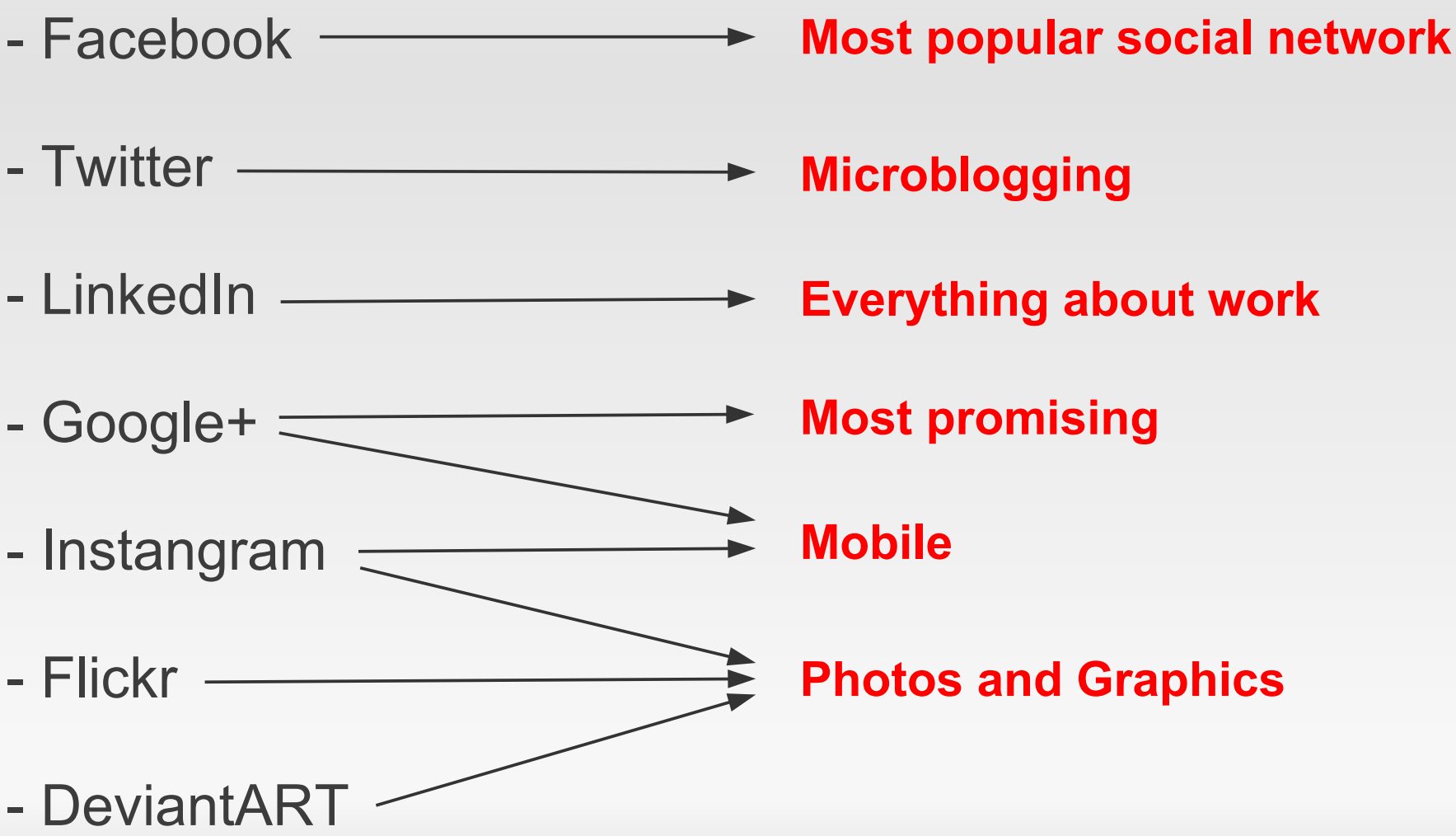
(oxforddictionaries.com/)



Real Social Networks

The SocioPatterns Sensing Platform

Virtual Social Networks



Real Social Networks

The SocioPatterns Sensing Platform

Virtual Social Networks

- Facebook
- Twitter
- LinkedIn
- Google+
- Instangram
- Flickr
- DeviantART

Analysis and Statistics thanks to their API

Real Social Networks

???

Real Social Networks

???

- Gathering places: café, theaters, ...
- University
- Public spaces
- Companies

Real Social Networks

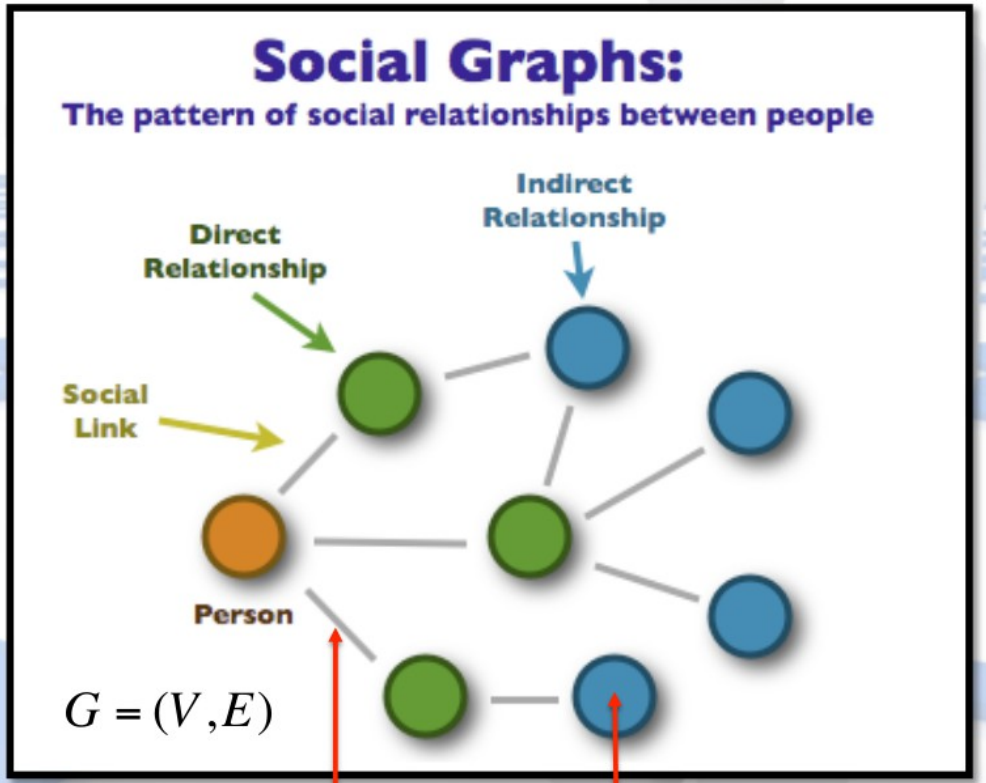
???

- Gathering places: café, theaters, ...
- University
- Public spaces
- Companies

But... What about analysis?

Social Networks as networks of contacts

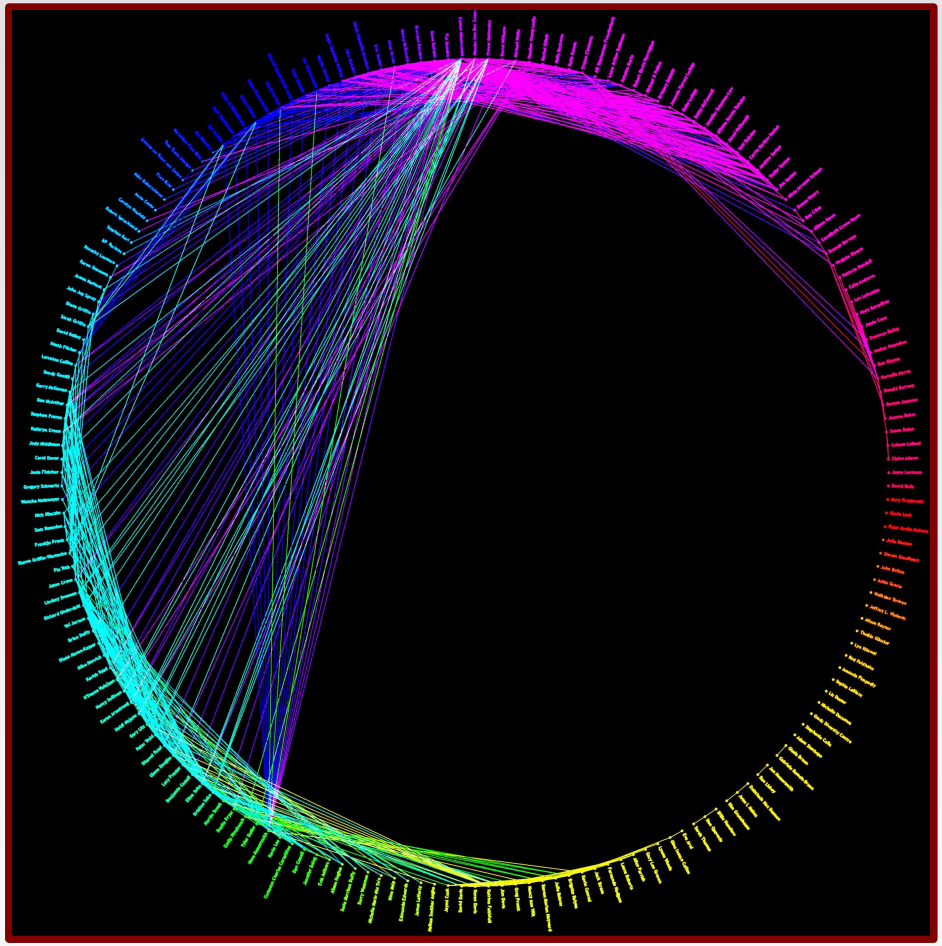
Representation



A definition of Social Graph

A social graph is the network of connections and relationships between people.

Real and Virtual Social Graphs



Real Social Graph

- **Person-to-Person interactions**:
  when an individual A physically meets another individual B.

A particular technology must be used to record and measure them.

Virtual Social Graph

- **Virtual interactions**:
  - A tag in Facebook
  - A re-tweet in Twitter
  - ….

Easy tracking by API

Real vs Virtual: state of the art

Virtual interactions

- Projects to collect and analyse data (by APIs)
- Facebook applications (e.g., games)
- Analysis of activities and friendships
- "Like button" to suggest targeted advertising

Real interactions

- Bluetooth interactions
- SocioPatterns projects

???

Live Social Semantics

LSS is an application that tracks and supports social networking between conference attendees.

Tracking of...

- virtual interactions (social networks)
- references of publications
- real-world F2F proximity

But limited ...

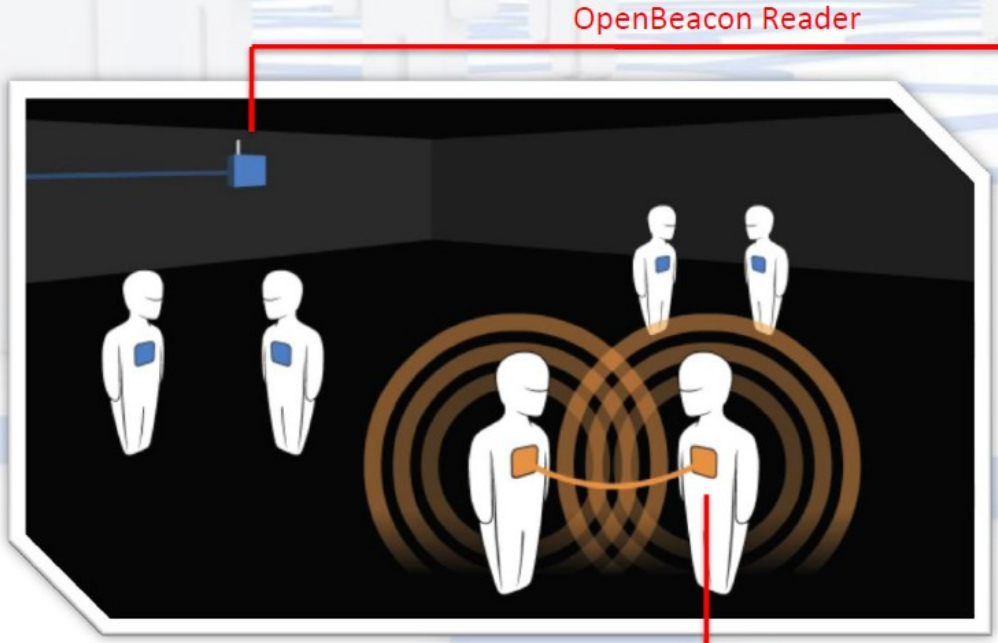
- time was too short
- narrow field of interest
- real vs virtual ???

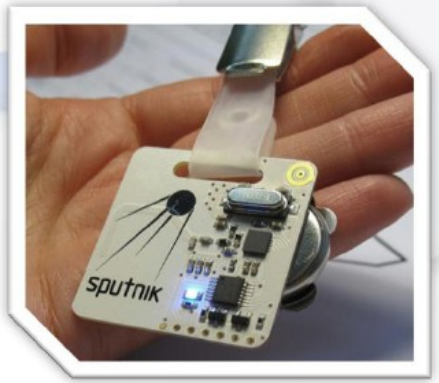
Results

- Similar behaviours among conferences
- More interaction for returning attendes
- Interaction time was very similar
- Mixing with similar seniority level

SocioPatterns platform: the OpenBeacon project

The SocioPatterns sensing platform employs wearable electronic badges to sense sustained face-to-face proximity among people.

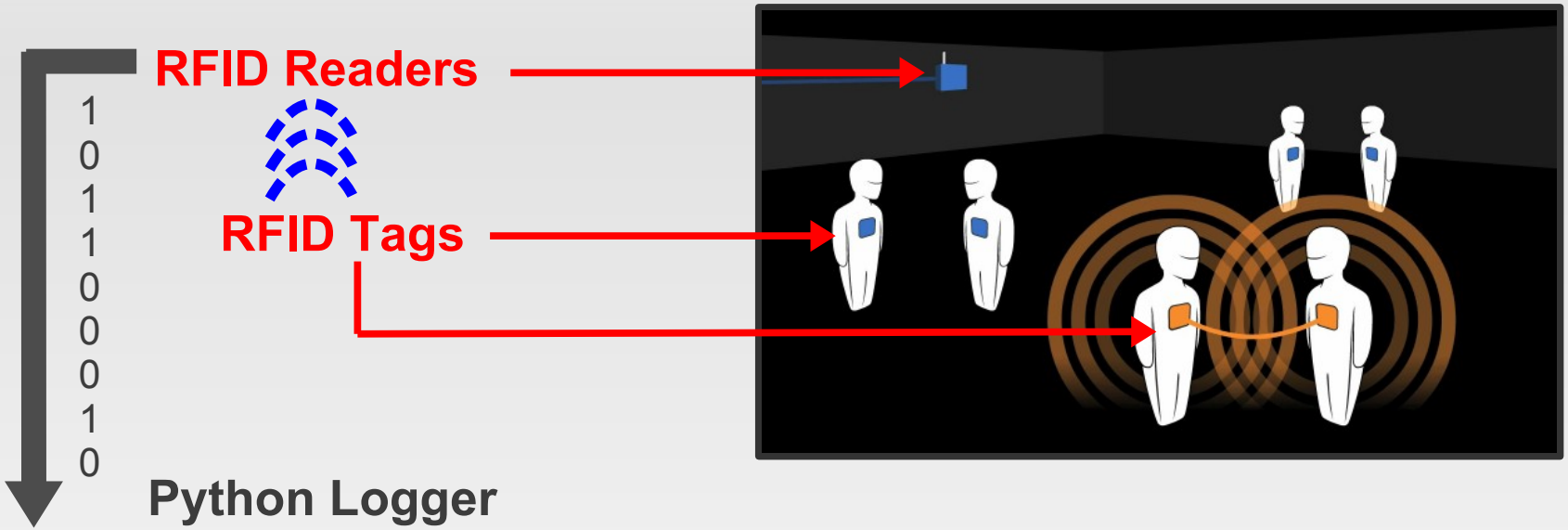




Real Social Networks

The SocioPatterns Sensing Platform

The full infrastructure

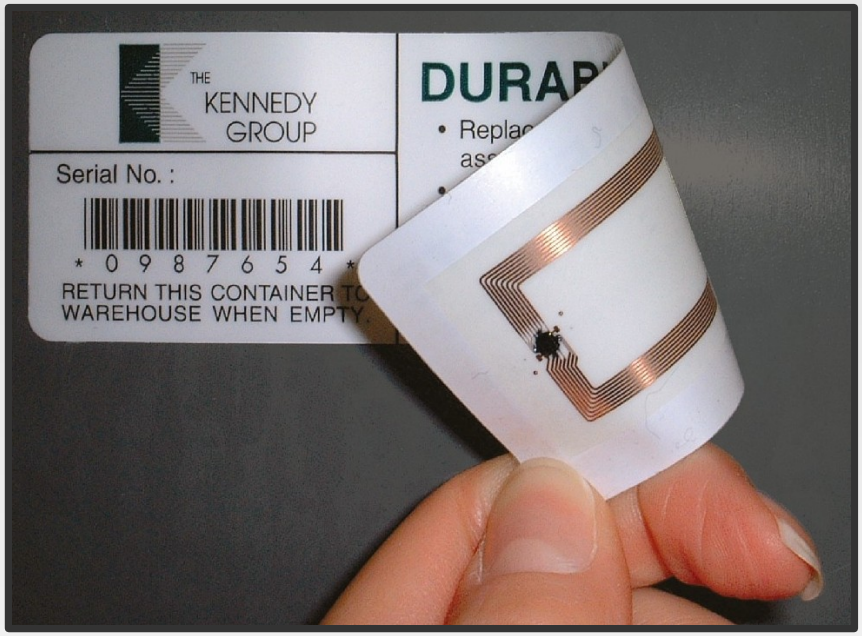



```
.....................
.....................
C t=1360238945 ip=0x0000000c id=1195 boot_count=0 seq=0x00001bfb flags=0 [1197(2) #9]
C t=1360238945 ip=0x00000015 id=1195 boot_count=0 seq=0x00001bfb flags=0 [1197(2) #9]
S t=1360238945 ip=0x00000017 id=1031 boot_count=95 seq=0x00004313 strgth=3 flgs=0 last_seen=1251
S t=1360238945 ip=0x00000017 id=1130 boot_count=95 seq=0x00004c8b strgth=3 flgs=2 last_seen=1118
S t=1360238945 ip=0x00000010 id=1050 boot_count=95 seq=0x00003a1f strgth=3 flgs=0 last_seen=1235
C t=1360238945 ip=0x0000000c id=1241 boot_count=0 seq=0x00004dbb flags=0 [1013(1) #5] [1212(1) #5]
S t=1360238945 ip=0x0000000c id=1166 boot_count=95 seq=0x00004323 strgth=3 flgs=0 last_seen=1013
S t=1360238945 ip=0x00000017 id=1193 boot_count=95 seq=0x00001f03 strgth=3 flgs=0 last_seen=1108
S t=1360238945 ip=0x00000010 id=1249 boot_count=95 seq=0x00004223 strgth=3 flgs=0 last_seen=1108
S t=1360238945 ip=0x0000000c id=1108 boot_count=95 seq=0x000040bf strgth=3 flgs=0 last_seen=1193
C t=1360238945 ip=0x0000000c id=1197 boot_count=0 seq=0x0000380f flags=0 [1195(1) #5]
C t=1360238945 ip=0x00000015 id=1197 boot_count=0 seq=0x0000380f flags=0 [1195(1) #5]
.....................
.....................
```

Log Parser

- Adjacency Lists
- Adjacency Matrix
- GEXF format
- JSON format
- DNF format

Pieces of information on RFID...



Radio-Frequency IDentification

Electromagnetic fields to transfer data from a tag to a reader and vice versa.

Main purpose: identification and tracking

Types: passive, semi-active, active

RFID frequency bands

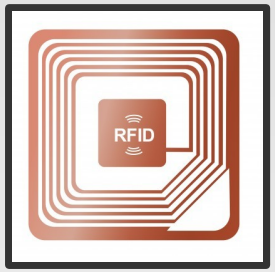
- 120 – 150 kHz (LF): passive tags [range = 10 cm]
- 13.56 MHz (HF): passive tags [range = 1 m]
- …
- 2450 – 5800 MHz: active tags [range = 1-5 m]

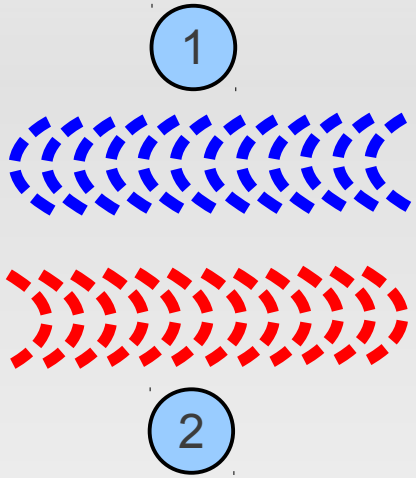
Real Social Networks

The SocioPatterns Sensing Platform

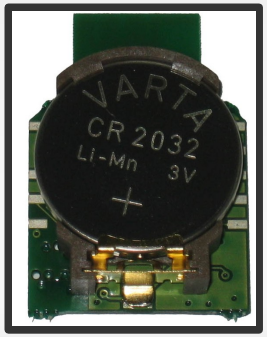
RFID: how it works

Passive

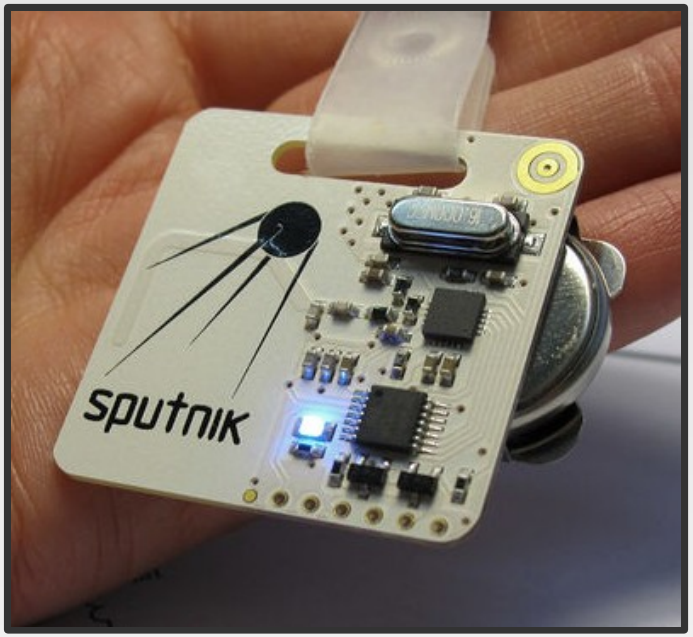




Active



Two types of OpenBeacon tags...



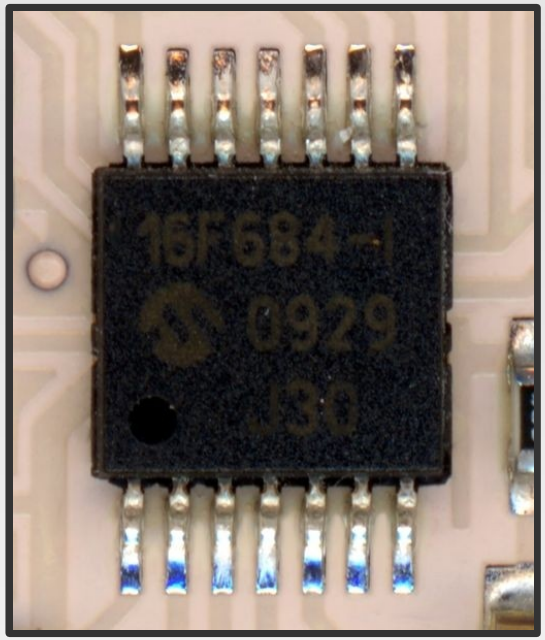
OpenBeacon Tag

- PIC16F688 microcontroller
- nRF24L01 2.4GHz transceiver

OpenBeacon USB 2

- LPC1343 ARM Cortex-M3 CPU
- nRF24L01 2.4GHz transceiver
- Bluetooth SPP
- 4MByte external flash memory

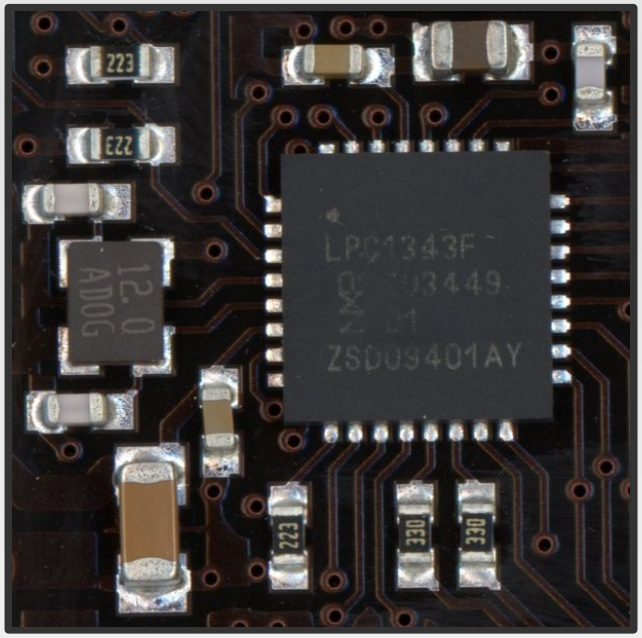
Two types of OpenBeacon tags...



OpenBeacon Tag

- PIC16F688 microcontroller:

- FREQ: 8 MHz to 31 KHz
- FLASH: 7 KB
- SRAM: 256 bytes



OpenBeacon USB 2

- LPC1343 ARM Cortex-M3 CPU:

- FREQ: up to 72 MHz
- FLASH: 32 KB
- SRAM: 8 KB

OpenBeacon Tag programming

What we need...

- PIC16F684 datasheet: http://bit.ly/Zm3dsM
- nRF24L01 datasheet: http://bit.ly/Z8SqyJ
- A GNU/Linux distribution (not required, but recommended) and a text editor
- MPLAB X Integrated Development Environment (IDE): http://bit.ly/xpKQZY [optional]
- HI-TECH C compiler for PIC10/12/16 MCUs: http://bit.ly/9cE47E
- PICkit 2 Command Line Interface (source): http://bit.ly/2ZGKdP
- PICkit 2 Development Programmer/Debugger

OpenBeacon Tag programming

Let's begin to configure everything... A mini-tutorial for Ubuntu users!

(I recommended to have a GNU/Linux distribution, do you remember?)

- Unzip the **HI-TECH C programmer file** and run the installer bin file:

```
$ unzip picc-9.83-linux.zip
$ chmod 755 picc-9.83-linux.run
$ sudo ./picc-9.83-linux.run
```

- Scroll down the license and enter ”y” to accept it.
- Click on the ”Enter” button to confirm the default installation path. **[do not change]**
- When the installer asks for the license type, append ”demo” to the word "HCPICP-".
- Open the ~/.bashrc file and append the following ”export” line:

```
export PATH=$PATH:/usr/hitech/picc/9.83/bin
```

OpenBeacon Tag programming

- Extract the **pk2cmdv1.20LinuxMacSource.tar.gz** file and compile the source:

```
$ tar xzvf pk2cmdv1.20LinuxMacSource.tar.gz
$ make
$ sudo make install
$ sudo install *.dat /usr/local/bin
```

- Connect the programmer and check if it's being detected by running:

```
$ lsusb -d 04d8
```

- Create a text file as shown below:

```
$ sudo gedit /etc/udev/rules.d/pickit.rules
```

- and past the following udev rule so as to automatically detect the programmer:

```
SYSFS{idVendor}=="04d8", SYSFS{idProduct}=="0033", MODE="0666"
```

- Disconnect and reconnect the programmer

OpenBeacon Tag programming

A simple typical application

```
while (1) {

     // packet building
     pkt.hdr.proto = RFBPROTO_BEACONTRACKER;
     pkt.tracker.strength = strength;
     pkt.tracker.powerup_count = htons (code_block);
     pkt.tracker.reserved = 0;
     pkt.tracker.seq = htonl (seq);
     pkt.tracker.oid_last_seen = htons(oid_last_seen);

     // add CRC to packet
     crc = crc16 (pkt.byte, sizeof (pkt.tracker) - sizeof (pkt.tracker.crc));
     pkt.tracker.crc = htons (crc);

     // adjust transmit strength
     nRFCMD_RegReadWrite (NRF_REG_RF_SETUP | WRITE_REG, NRF_RFOPTIONS | (strength << 1));

// CONT...
```

OpenBeacon Tag programming

A simple typical application

```
// ...CONT

     // encrypt the data
     protocol_encode ();

     // transmit data to nRF24L01 chip
     nRFCMD_RegWrite (WR_TX_PLOAD | WRITE_REG, (unsigned char *) &pkt, sizeof (pkt));

     // send data away
     nRFCMD_Execute ();

     // update code_block so on next power up
     // the seq will be higher or equal
     crc = (unsigned short) (seq >> 16);
     if (crc >= code_block) {
          store_incremented_codeblock ();
     }

     // finally increase sequence number
     seq++;
}
```

OpenBeacon Tag programming

Build files

- Relevant parts of the **Makefile**

```
COMPILER_PATH=/usr/hitech/picc/9.83/bin/
PK2=/usr/local/bin/pk2cmd
CC=$(COMPILER_PATH)picc

MODE=std
CHIP=16F688
PROGNAME=openbeacontag

MAIN=main.c
SRCS=$(MAIN) timer.c nRF_CMD.c

...

$(PROGNAME).hex: obj/$(PROGNAME).hex
    php ./create_counted_firmware.php
```

Change if you have a different version of the HI-TECH C compiler

pk2cmd path

source files

Script to generate multiple firmwares

OpenBeacon Tag programming

Build files

- Relevant parts of the **counted firmwares script**:

```
$startid = 1000;
$stopid = 1500;

for($i = $startid; $i <= $stopid; $i++) {
    // Default TEA encryption key of the tag
    $tea_key = array( 0x00112233, 0x44556677, 0x8899AABB, 0xCCDDEEFF );

    ...

    // Read HEX file from file
    patch_hexread(INPUT_FILE);

    // apply patch first time
    patch_apply($patches,TRUE);

    //echo "\n\nOpenBeacon tag ID set to '$patch_list[_oid]'\n\n";
    patch_hexwrite(OUTPUT_FILE.$i);
}
```

First and last tag ID

OpenBeacon Tag programming

Compiling process and tag programming

- Go into the application root folder:

```
$ cd /<path_application>
```

- Create an ”images” folder:

```
$ mkdir images
```

In this folder the compiling process will copy a firmware for each tag.

- Compile the application:

```
$ make clean
$ make
```

- Go into the ”images” folder:

```
$ cd images
```

- Connect the Pickit 2 programmer and attach a tag identified by ID XXXX.

- Program the tag XXXX by running the following command:

```
$ pk2cmd -PPIC16F688 -M -Fopenbeacontag.hex_XXXX
```

OpenBeacon Tag programming

Troubleshooting

- If during the compilation process you get the following error:

```
(923) unknown suboption "std"
make: *** [obj/main.p1] Error 1
```

- then you can try to reactivate the HI-TECH C license:

```
$ sudo /usr/hitech/picc/9.83/bin/Reactivate-PICC-PRO-9.83
```

OpenBeacon USB 2 programming

What we need...

- LPC1343 ARM Cortex-M3 datasheet: http://bit.ly/WybVWj
- nRF24L01 datasheet: http://bit.ly/Z8SqyJ
- A GNU/Linux distribution (not required, but recommended) and a text editor
- LPCXpresso (IDE): http://bit.ly/7GoX8G [optional]
- arm-2011.03-42-arm-none-eabi-i686-pc-linux-gnu.tar.bz2: http://bit.ly/YmrdK2
- Git software
- A micro-usb cable

OpenBeacon USB 2 programming

Let's begin to configure everything... A mini-tutorial for Ubuntu users!

(I recommended to have a GNU/Linux distribution, do you remember?)

- Extract the **ARM library file** and copy the whole folder into your favourite path:

```
$ tar xjvf arm-2011.03-42-arm-none-eabi-i686-pc-linux-gnu.tar.bz2
$ cp -r arm-2011.03 $HOME/Programs/
```

- Open the ~/.bashrc file and append the following ”export” line:

```
export PATH=$HOME/Programs/arm-2011.03/bin:$PATH
```

- Install other required software:

```
$ sudo apt-get install git
$ sudo apt-get install autoconf libusb-dev libtool libreadline-dev
```

OpenBeacon USB 2 programming

- Clone the git repository:

```
$ git clone https://github.com/ccattuto/openbeacon.git
$ cd openbeacon
$ git remote show origin
$ git checkout -t origin/reengineering_timeline
$ git branch (check if "reengineering_timeline" is selected)
```

- Change `CROSS=arm-cortexm3-eabi-` to `CROSS=arm-none-eabi-` :

```
$ cd firmware/lpc13xx
$ gedit core/Makefile.rules
```

OpenBeacon USB 2 programming

- Build the **lpc-flash** program:

```
$ cd lpc-flash
$ libtoolize --force
$ aclocal
$ autoheader
$ automake --force-missing --add-missing
$ autoconf
$ git log . > ChangeLog
$ autoreconf --install
$ ./configure
$ make
$ sudo make install
```

OpenBeacon USB 2 programming

Build files

- Relevant parts of the **Makefile**

```
TARGET=openbeacon-usb2
ARCH=LPC13
CPU=$(ARCH)43
```

Everything about the architecture, application's name and CPU.

```
...

APP_SRC= \
  src/main.c \
  src/pmu.c \
  src/nRF_CMD.c \
  src/nRF_API.c \
  src/storage.c \
  src/customIO.c \
  src/bluetooth.c \
  src/3d_acceleration.c

...

include ../core/Makefile.rules
```

source files

OpenBeacon USB 2 programming

Compiling process and tag programming

- Go into the application root folder:

```
$ cd /<path_application>
```

- Compile the application:

```
$ make clean && make
```

- Connect the tag to the PC by using a micro-usb cable
- Press and hold the reset button; then press the right button
- Wait about 3 seconds
- Release the reset button and then the right button
- Copy the firmware:

```
$ lpc-flash openbeacon-usb2.bin /media/CRP\ DISABLD/firmware.bin
```

- Press the reset button and disconnect the tag

OpenBeacon Ethernet PoE II programming

Compiling process and reader programming

- Go into the application root folder:

```
$ cd /<path_application>
```

- Compile the application:

```
$ make clean && make
```

- Connect the reader to the PC by using a mini-usb cable

- This command should show the device port that was created:

```
$ ls /dev/ttyACM?
```

- To flash the reader, you have two options:

1) press the Flash button inside the reader for some seconds until the reader goes off

2) reboot the reader in the firmware mode by using the minicom tool (recommended)

- Run the following command to flash the reader, then disconnect it:

```
$ sudo make flash
```

OpenBeacon Ethernet PoE II configuration

Network configuration and other stuff

- Connect the reader to the PC

- Run minicom and configure it to receive data from the reader:

```
$ minicom -s
```

- Exit and save the configuration. Run again minicom:

```
$ minicom
'h' - show help
'a' - set reader address ('a10.254.0.100' for 10.254.0.100)
'c' - show configuration
'g' - set gateway ip
'm' - netmask config ('m255.255.0.0')
't' - set target server ip ('t1.2.3.4')
's' - store configuration
```

A MAC protocol to record F2F interactions

RX: receiving mode, TX: transmitting mode, S: sleeping mode

- Specifically:

RX: receiving mode, S: sleeping mode, TG: TX to tags, AP: TX to readers

```
t(TG) = around 0.2 ms
t(AP) = around 0.2 ms
t(RX) = 150 + rand(0,100) ms
t(S) = 50 + rand(0,41) ms
```

Python Logger

- It receives packets from readers and it stores each message in a log file.

Timestamp Reader ID Source Tag Seq number TX power

```
...
S t=1360238164 ip=0x0000000c id=1128 boot_count=7 seq=0x00001133 strgth=3 flgs=0 last_seen=1007
S t=1360238164 ip=0x0000000b id=1134 boot_count=2 seq=0x0004000b strgth=3 flgs=0 last_seen=0
S t=1360238164 ip=0x00000017 id=1234 boot_count=95 seq=0x0000144f strgth=3 flgs=0 last_seen=1166
C t=1360238164 ip=0x0000000c id=1007 boot_count=0 seq=0x00001003 flags=0 [1279(2) #9]
C t=1360238164 ip=0x0000000b id=1007 boot_count=0 seq=0x00001003 flags=0 [1279(2) #9]
C t=1360238164 ip=0x00000023 id=1237 boot_count=0 seq=0x0000184b flags=0 [1125(2) #9]
C t=1360238164 ip=0x0000000c id=1237 boot_count=0 seq=0x0000184b flags=0 [1125(2) #9]
S t=1360238164 ip=0x0000000b id=1032 boot_count=15 seq=0x00000c7b strgth=3 flgs=0 last_seen=1261
S t=1360238164 ip=0x0000000c id=1125 boot_count=31 seq=0x000210ef strgth=3 flgs=0 last_seen=1237
C t=1360238164 ip=0x0000000c id=1279 boot_count=0 seq=0x0000141f flags=0 [1033(2) #9]
S t=1360238164 ip=0x00000017 id=1020 boot_count=95 seq=0x00012353 strgth=3 flgs=0 last_seen=1123
S t=1360238164 ip=0x00000017 id=1190 boot_count=95 seq=0x00000a83 strgth=3 flgs=0 last_seen=1123
S t=1360238164 ip=0x00000015 id=1244 boot_count=2 seq=0x00011fa7 strgth=3 flgs=0 last_seen=0
...
```

contact

Target Tag

Python Logger

```
def datagramReceived(self, payload, (ip, port)):
  tstamp = int(time.time())

  (e_crc, e_proto, e_interface, e_reader_id, e_size, e_sequence,
   e_timestamp) = struct.unpack('!HBBHHLL', payload[:16])
    inner_payload = payload[16:]

    if DECODE:
      decrypted_data = xxtea.decode(inner_payload)
    else:
      decrypted_data = inner_payload

    proto = struct.unpack("!B", decrypted_data[0])[0]
    id_tag = struct.unpack("!H", decrypted_data[1:3])[0]

# CONT
```

Message struct (tag application)

```
typedef struct
{
  u_int8_t proto; //1byte
  u_int16_t oid; //2byte
  u_int8_t flags; //1byte
} TBeaconHeader;

typedef struct
{
  TBeaconHeader hdr; //4byte
  u_int8_t strength; //1byte
  u_int16_t oid_last_seen; //2byte
  u_int16_t powerup_count; //2byte
  u_int8_t reserved; //1byte
  u_int32_t seq; //4byte
  u_int16_t crc; //2byte
} TBeaconTracker;

typedef struct
{
  TBeaconHeader hdr; //4byte
  u_int16_t oid_prox[PROX_MAX]; //8byte
  u_int16_t short_seq; //2byte
  u_int16_t crc; //2byte
} TBeaconProx;
```

Python Logger

```
# CONT
        if proto == Sighting.protocol:
            (proto, id, flags, strength, id_last_seen, ts_tag, battery,
             seq, crc) = struct.unpack("!BHBBHHBLH", decrypted_data)
            if crc != xxtea.crc16(decrypted_data[:14]):
                log.msg('Rejecting packet from %s on CRC' % e_reader_id)
                return

            sighting = Sighting(tstamp, e_reader_id, id, seq, strength,
            flags, last_seen=id_last_seen, boot_count=ts_tag)
            self.process_sighting(sighting)

        elif proto == Contact.protocol:
            (proto, id, flags, seen1, seen2, seen3, seen4, seq,
             crc) = struct.unpack("!BHBHHHHHH", decrypted_data)
            if crc != xxtea.crc16(decrypted_data[:14]):
                log.msg('Rejecting packet from %s on CRC' % e_reader_id)
                return

            seen_id = []
            seen_pwr = []
            seen_cnt = []

            for id2 in [seen1, seen2, seen3, seen4]:
                if not id2:
                    break
```

sighting packet

contact packet

Python Logger

```
            seen_id.append(id2 & 0x0FFF)
            seen_pwr.append(id2 >> 14)
            seen_cnt.append((id2 >> 12) & 0x03)

        contact = Contact(tstamp, e_reader_id, id, seq, seen_id, seen_pwr, seen_cnt, flags=flags)
        self.process_contact(contact)

    else:
        Return

...
```

because...

C Code – Tag Application

```
PROX_TAG_ID_BITS = 12
PROX_TAG_COUNT_BITS = 2
```

```
pkt.prox.oid_prox[j] = (j < seen_count) ? (htons (
                        (oid_seen[j]) |
                        (oid_seen_count[j] << PROX_TAG_ID_BITS) |
                        (oid_seen_pwr[j] << (PROX_TAG_ID_BITS+PROX_TAG_COUNT_BITS))
                        )) : 0;
```

Real Social Networks

The SocioPatterns Sensing Platform

Testbeds

- DIAG

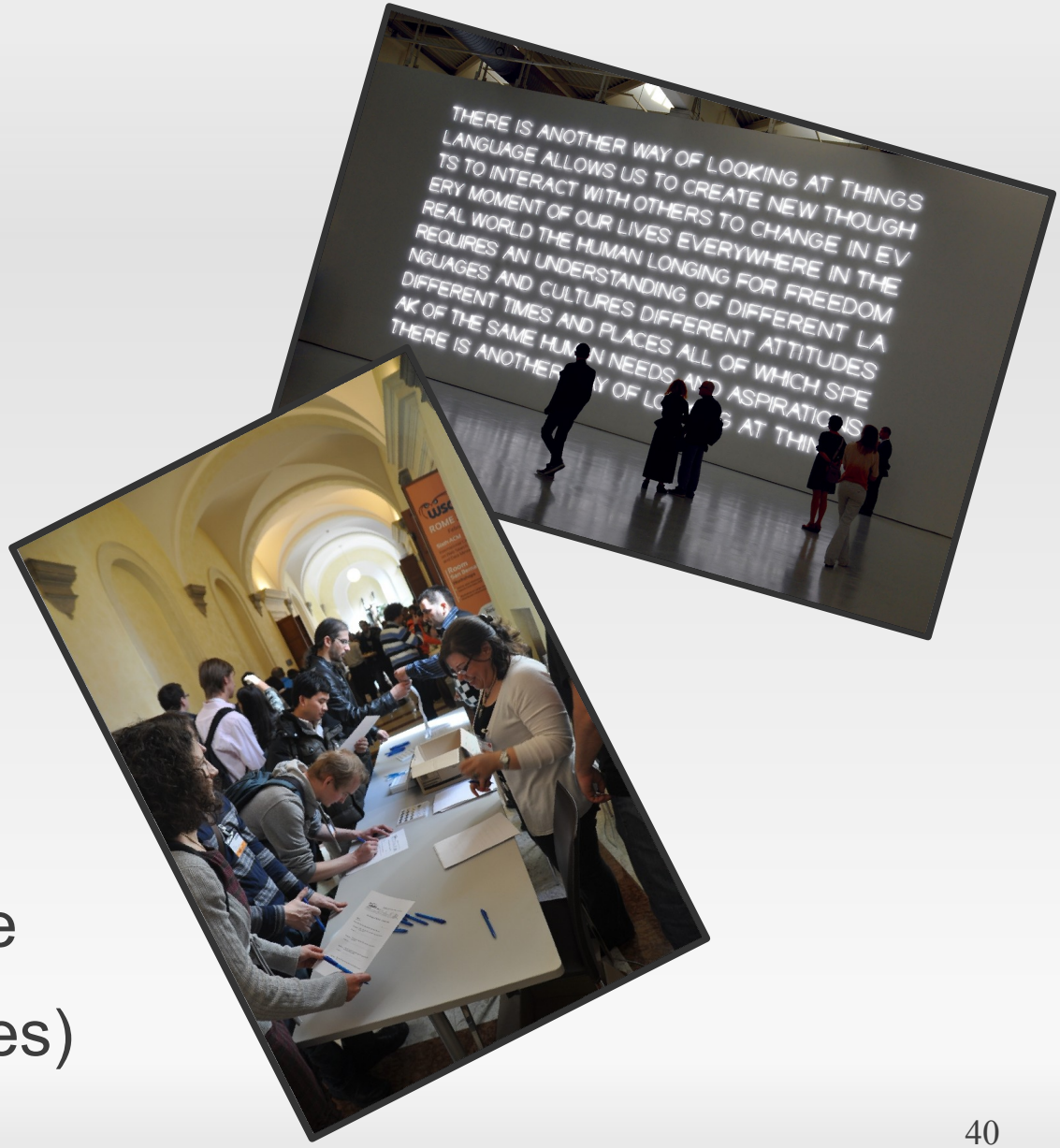
(5 days, ~ 120 students)

- MACRO Museum

(2.5 hours, ~ 120 visitors)

- WSDM 2013 Conference

(2.5 hours, ~ 100 attendees)



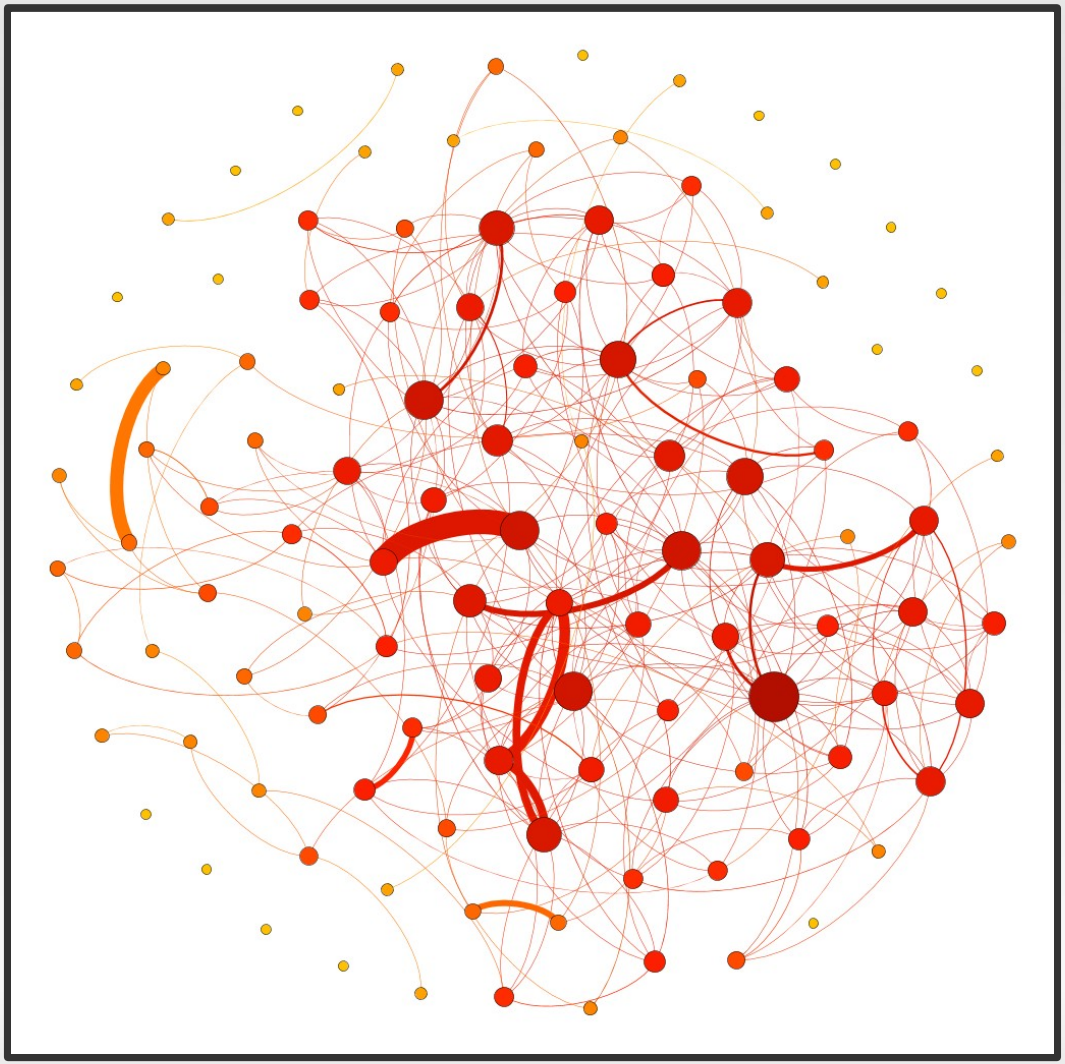
DIAG Experiment

- In 2011 we tracked in a completely anonymous way the movement of people at the ground floor of our department DIAG.
- The information we recorded for each person participating to the experiment were the following:
  - Age
  - Sex (male/female)
  - Student type
  - Course of study
  - Academic year
- At the end of the experiment we collected about 250MB of data logs; then we parsed the collected data to analyse the relationships among people:
  - Analyse the occupation of common spaces (rooms, library, corridors)
  - Analyse relationships between users

Real Social Networks

The SocioPatterns Sensing Platform

DIAG Experiment



Aggregated Social Graph

Average Degree: 5.103

Network Diameter: 7

Modularity: 0.837

Number of Communities: 31

Connected Components: 20

Avg Path Length: 3.152

Live Demo: http://bit.ly/WAUsv7

Information: http://bit.ly/GQKKJz

MACRO Experiment

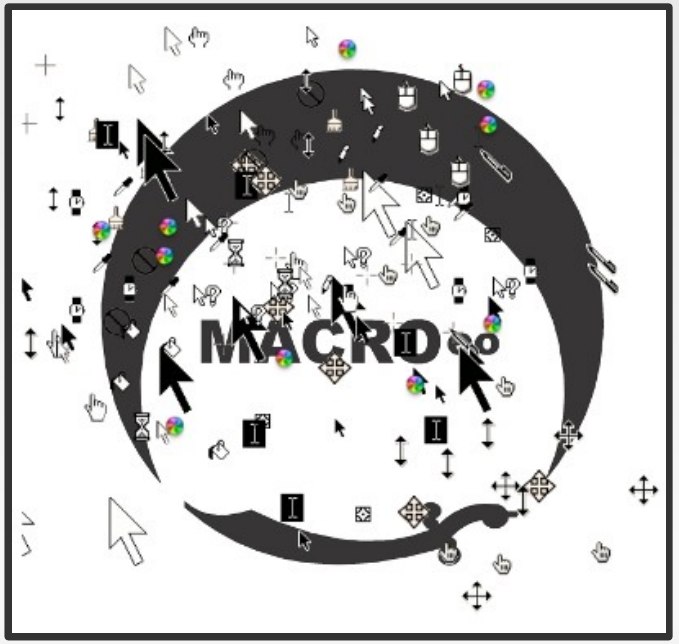
- After the deployment of our DIAG experiment at the Computer Engineering Department "Antonio Ruberti", we carried out a similar deployment at the MACRO museum during the NEON exhibition opening, the 20th of June 2012.

- The information we recorded for each visitor participating to the experiment were the following:
  - Age
  - Sex (male/female)
  - Education Level
  - Professional Area
  - Nationality (Italy or not)
  - Group (with other people or not)

MACRO Experiment

- At the end of the experiment we collected about 10MB of data logs; then we parsed the collected data to analyse the relationships among people and also among visitors and artworks:
  - Analyse how long visitors look at artworks according to them profiles.
  - Analyse relationships among visitors and artworks.
  - Analyse what kind of artworks most visitors prefer.
  - Analyse social interactions among visitors.
  - Total analysis of the success of the inauguration.

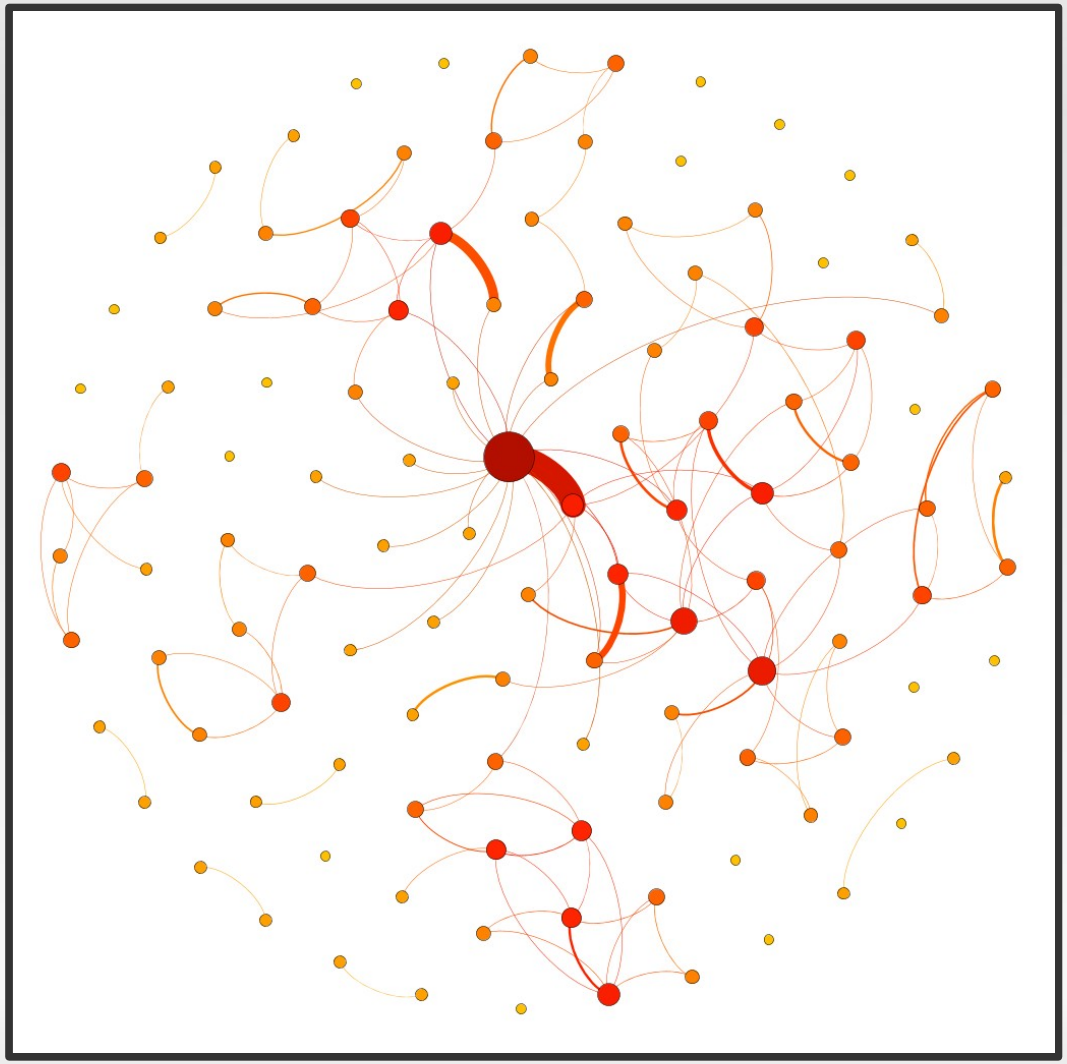
In addition, we strictly collaborated with the artist Miltos Manetas so as to realize its artwork: http://300visitors.com/.



Real Social Networks

The SocioPatterns Sensing Platform

MACRO Experiment



Aggregated Social Graph

Average Degree: 2.316

Network Diameter: 10

Modularity: 0.836

Number of Communities: 45

Connected Components: 27

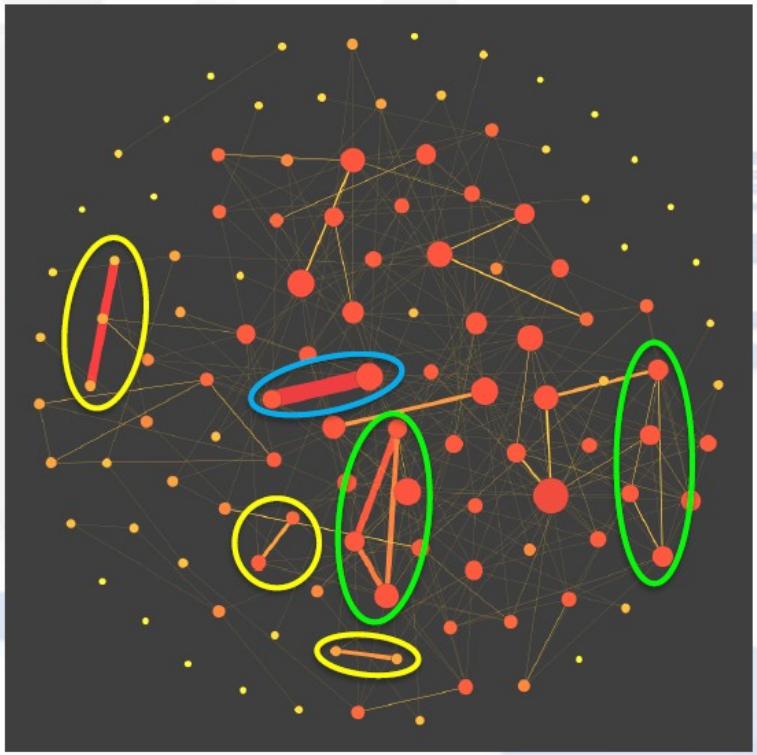
Avg Path Length: 4.221

Live Demo: http://bit.ly/YQo5s9

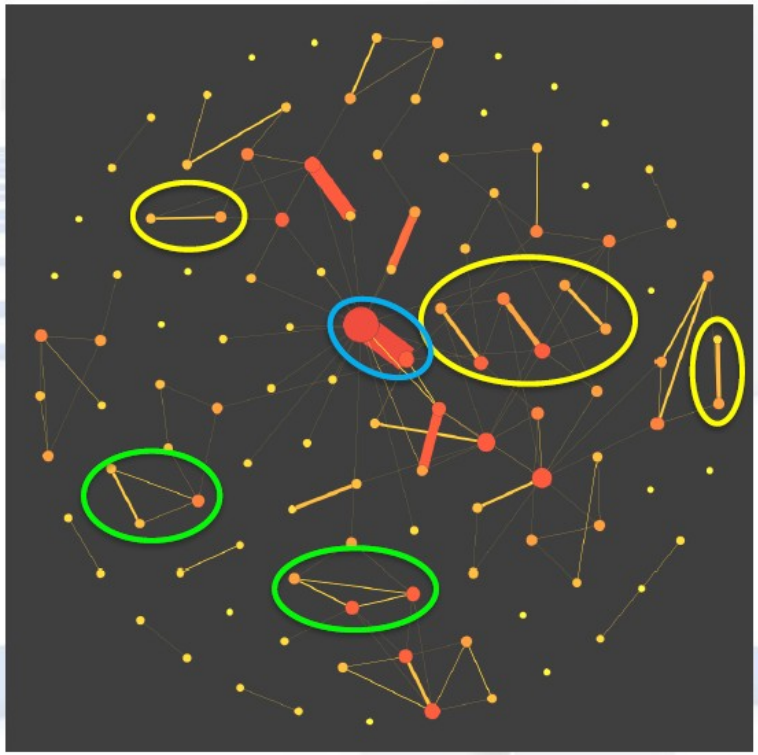
Information: http://bit.ly/MpWXfH

Brief analysis about our networks

- DIIAG experiment



- MACRO experiment



Similarity in terms of grouping: triangles and couples

WSDM Experiment

- The third experiment we realized, was deployed over the course of the WSDM 2013 conference in February 2013. Before starting the experiment, we gave about 100 attendees four questions to be answered. Participants had to answer all questions avoiding to talk to others. Afterwards, we started the social experiment and, in the end, attendees had to answer again the same questions.
- The questions were:

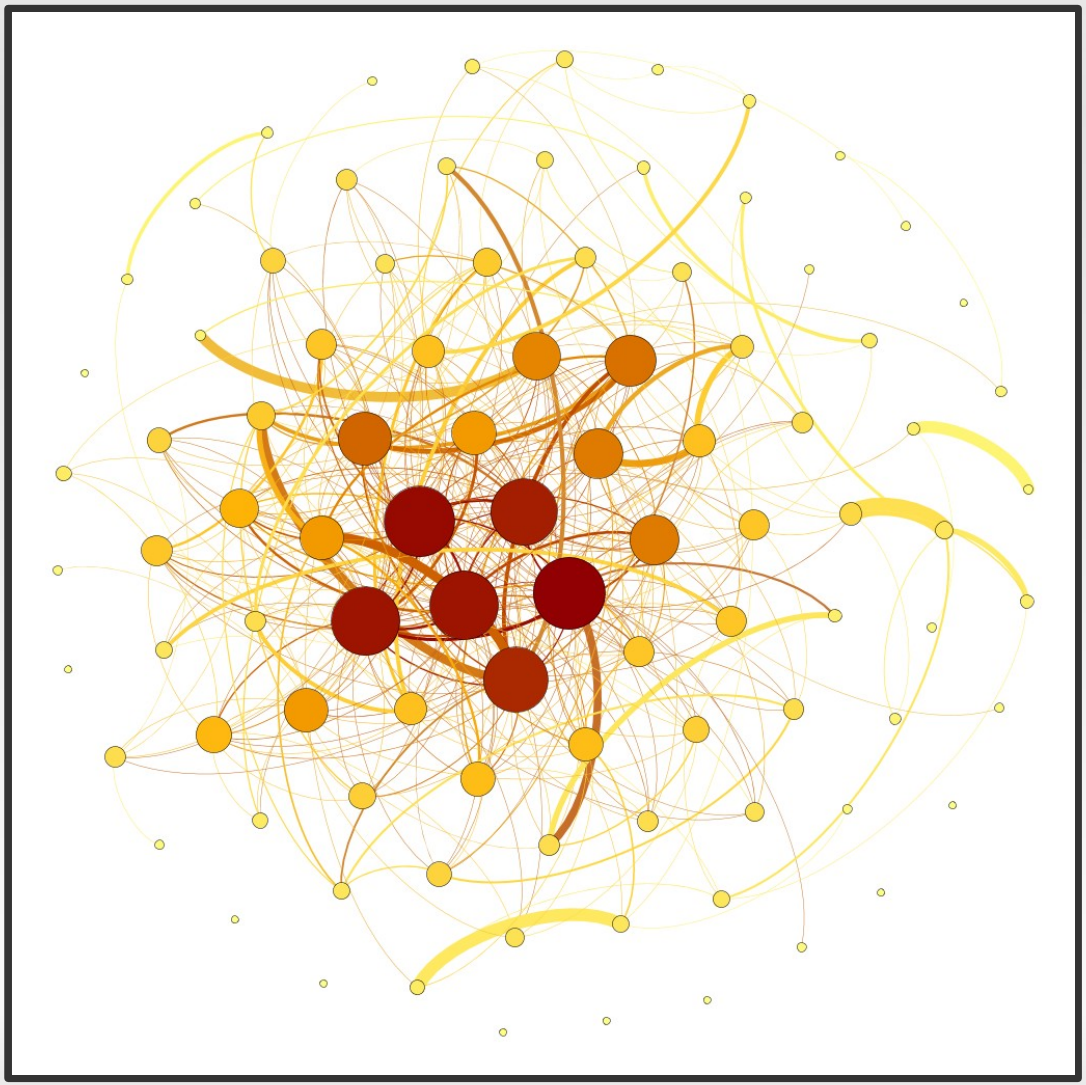
  - What was the total value in euro of all the coins thrown at the Trevi fountain in 2011?
  - What is the total length (in meters) of the corridor of the Auditorium Antonianum?
  - What is the average number of journal papers among the WSDM 2013 participants according to DBLP?
  - What was the number of Internet users in New Zealand by the end of 2011?

- GOAL: do attendees improve their answers after talking with others?

Real Social Networks

The SocioPatterns Sensing Platform

WSDM Experiment



Aggregated Social Graph

Average Degree: 8.362

Network Diameter: 6

Modularity: 0.636

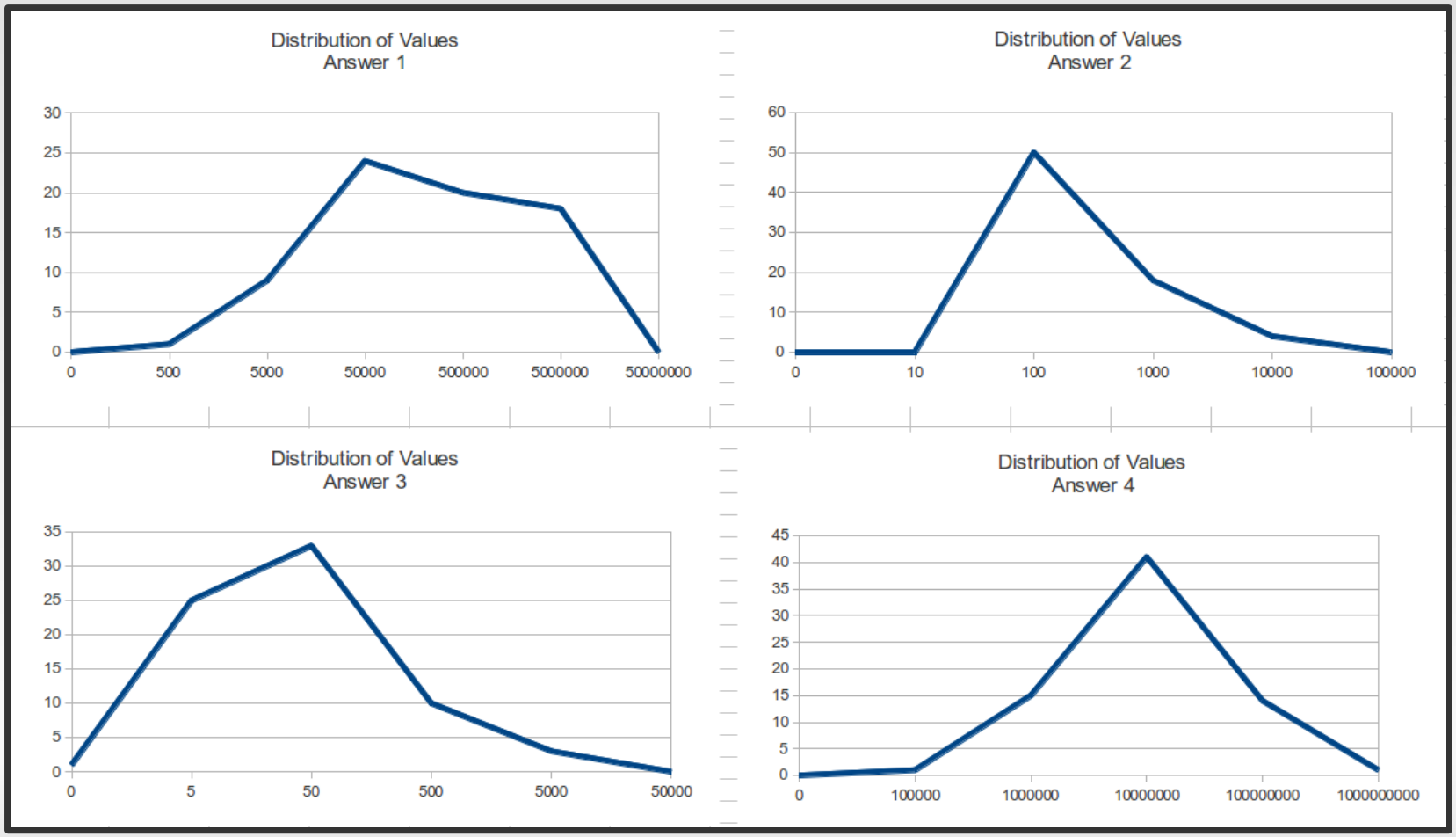
Number of Communities: 21

Connected Components: 11

Avg Path Length: 2.561

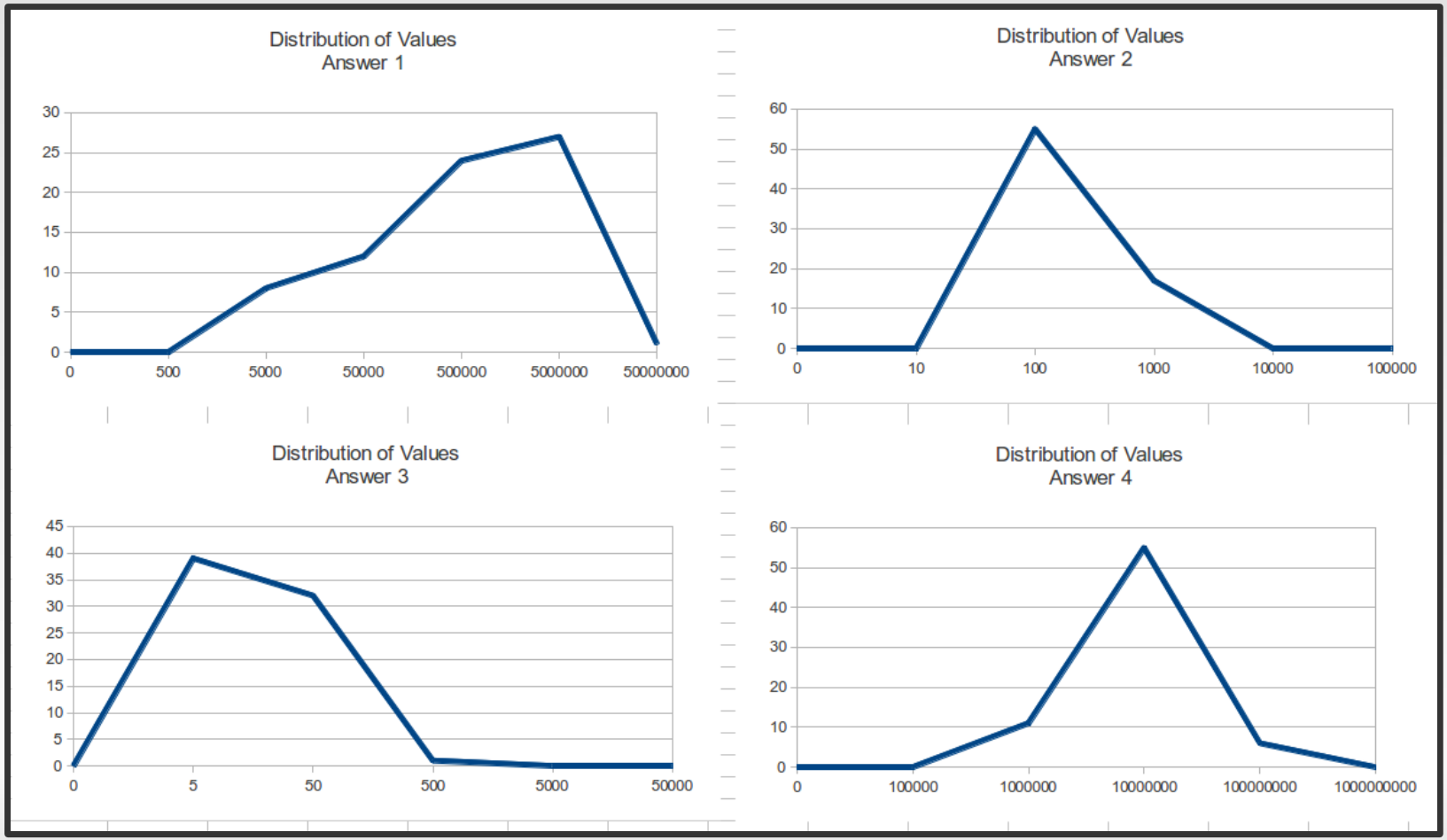
WSDM Experiment

Distribution of values before the social experiment



WSDM Experiment

Distribution of values after the social experiment



Project proposals

- Modify the reader and tag firmwares in order to work with 32-byte packets
- Implement the Blueetooth communication on OpenBeacon USB 2 devices
- Design an Android application to allow communication between smartphones and tags
- Modify the tag firmware to record contacts into the flash and to periodically send away
- Implement the sleeping mode on the OpenBeacon USB 2 devices
- Modify the reader firmware in order to transmit packets to tags