Design and development of a software architecture for seamless vertical handover in mobile communications

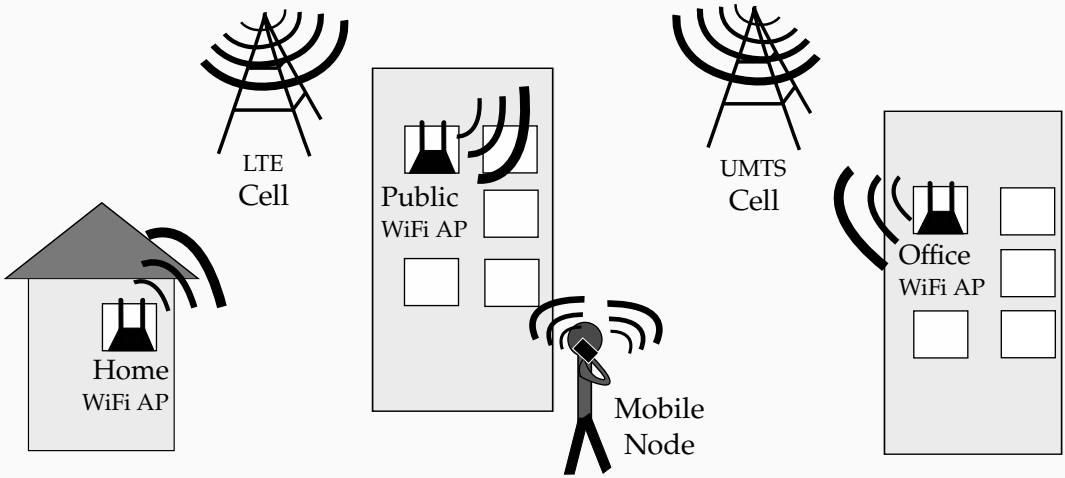
Matteo Martelli

December 2016

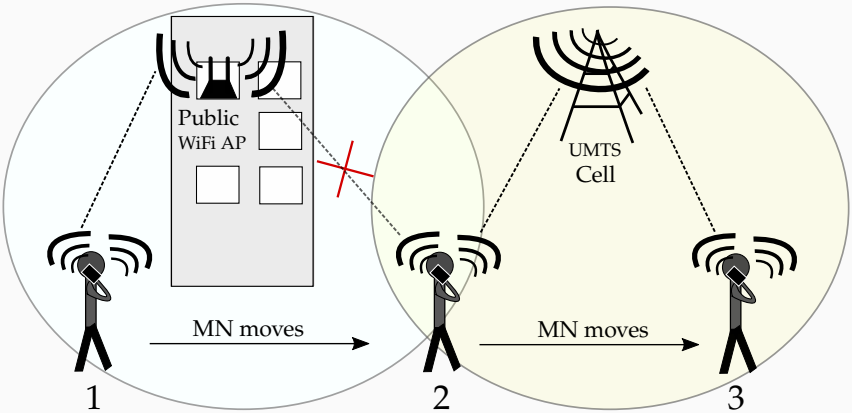
Supervisor: prof. Vittorio Ghini

Mobility

Scenario: Multihomed MNs and different wireless technologies



Vertical Handover



Worse if VHO is entirely controlled by the MN: WiFi and cellular network.

Issues

- **Unavailability periods**
- **Service interruptions**
- **Unreachability** due to *NATs* and *firewalls*
- **Session reinitialization** may be required

Solution

Always Best Packet Switching - Handover

The MN chooses which NIC should be used **per each packet**.

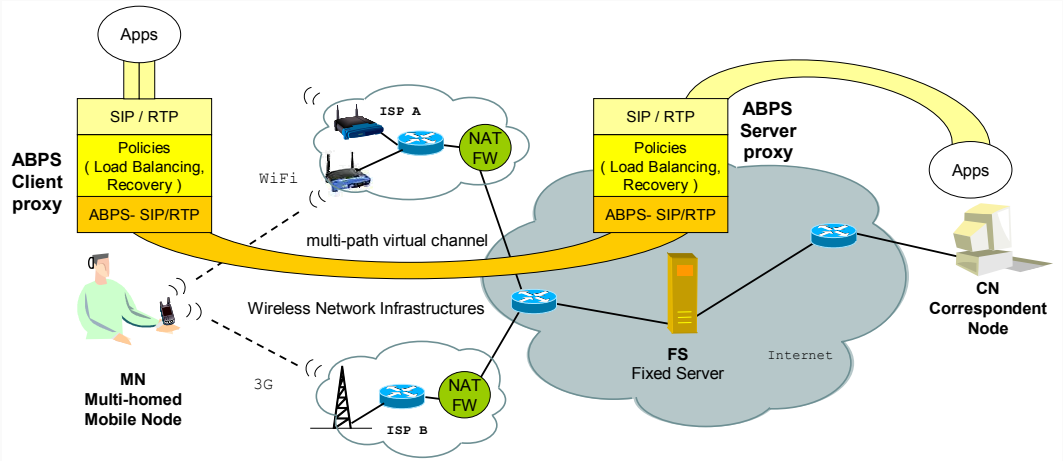
Simultaneous transmission on both NICs during vertical handover.

Relies on *Transmission Error Detector (TED)* infos for handover decision:

- **frame delivery status** (to the AP)
- **frame retransmissions**.

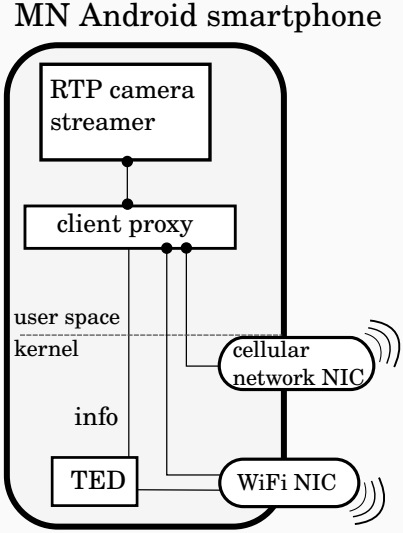
Always Best Packet Switching - Host Mobility

- One proxy for NICs managing (Client Proxy)
- One proxy relay for NAT and firewall traversal (Fixed Server)



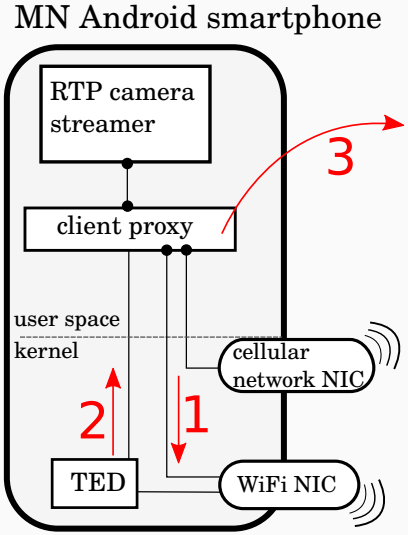
Implementation

MN Implementation



- Android, most popular OS on smartphones
- Some similarities on common GNU/Linux Distros
- Linux (modified) kernel.
- I adapted TED for the “Android kernel”
- I developed the client proxy

MN Implementation




```
int recv_ted_info(ted_info_s *ted_info) {
        /* ... */
        if (ted_info->status != ACK ||
           (ted_info->retry_count > RETRY_TH))
                pkt_risk++;
        else
                pkt_ok++;

        risk_ratio = pkt_risk / CWIN;

        if (risk_ratio > RISK_RATIO_TH) {
                enable_cell_nic();
                pkt_risk = pkt_ok = 0;
        }
        if (pkt_risk + pkt_ok > CWIN) {
                if (risk_ratio < TOLERANCE) {
                        disable_cell_dev()
                }
                pkt_risk = pkt_ok = 0;
        }
        /* ... */
}
```

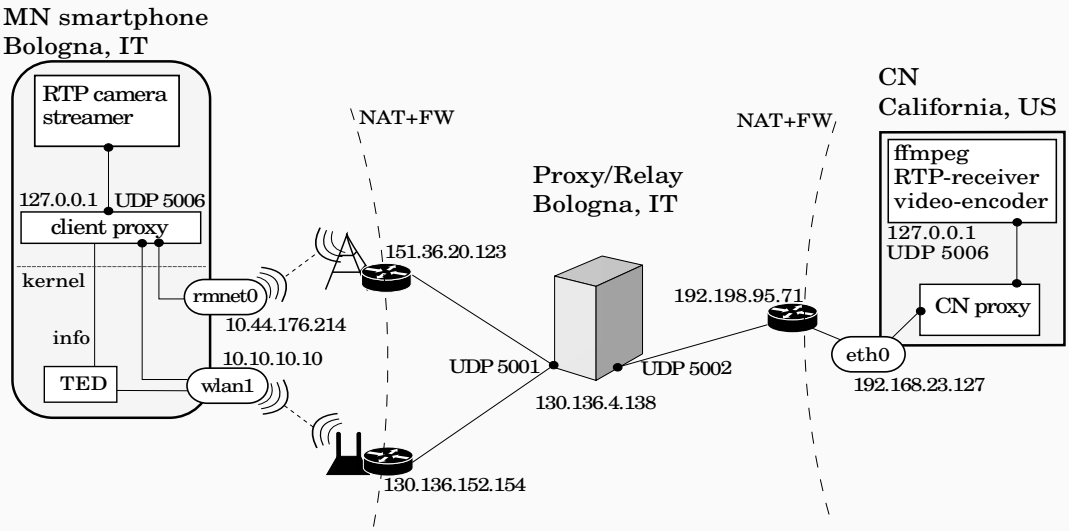
Relay and CN tools

I also developed:

- simple relay application to let end-nodes overcome NATs and firewalls
- CN software tools to enable a remote receiving counterpart

Experiments and Evaluations

Experimental Environment

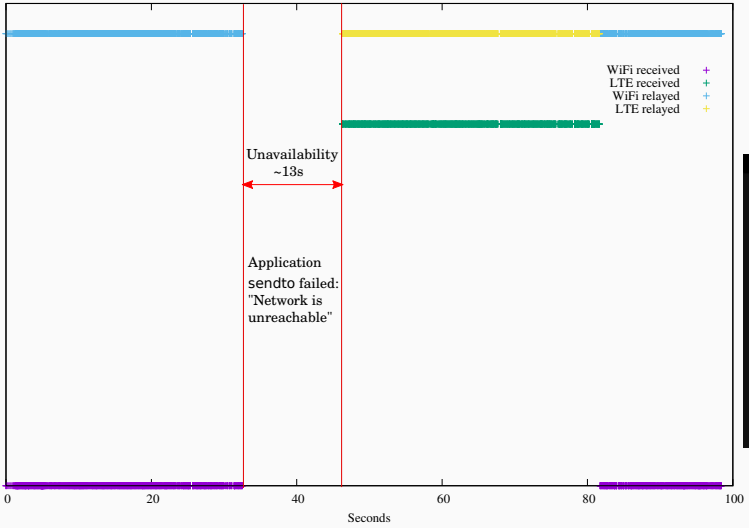


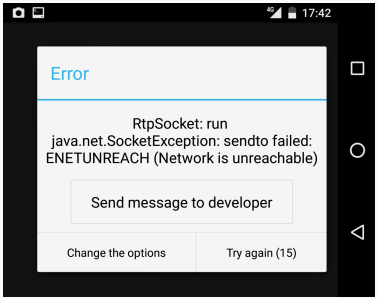
Results

Experiments conducted while walking from indoor with good WiFi link to outdoor (out of WiFi coverage but good LTE link)

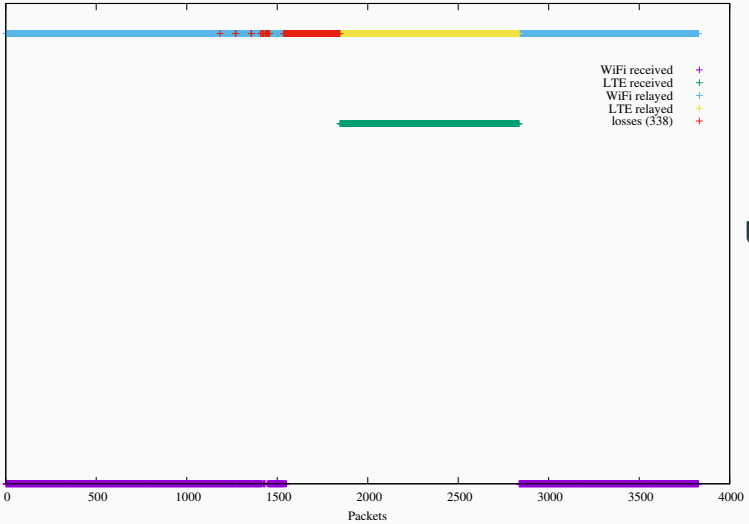
- 10 experiments **without** TED and client proxy
  - In 5 of them the streamer app failed with session reinitialization
  - In 5 no reinitialization but long interruptions
- 10 experiments **with** TED and client proxy
  - shorter unavailability periods

Results - no TED, no client proxy, app failure





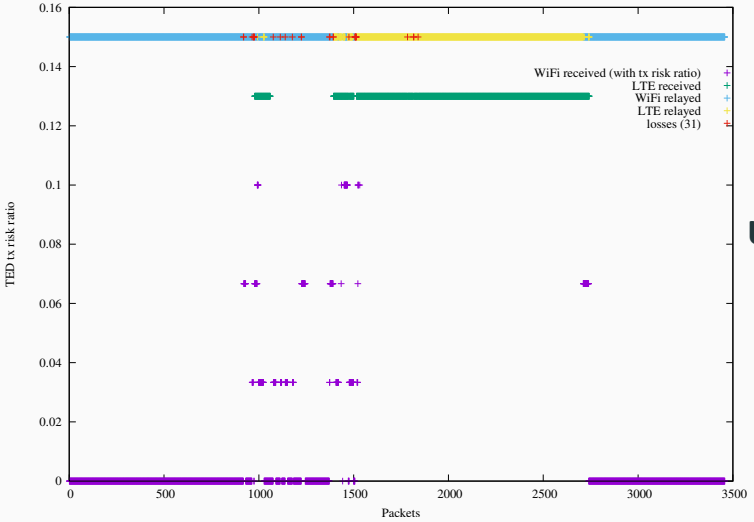
Results - no TED, no client proxy, no app failure



Loss: 338

Unavailability: 6455ms

Results - TED, client proxy



Loss: 31

Unavailability: 286ms

Conclusion

Conclusion

Similar results from other experiments.

Results are still inaccurate but promising.

Many possible future works.

Software and documentation currently hold on

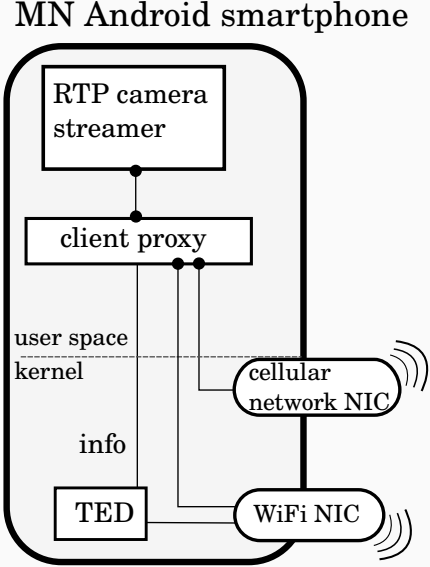
`github.com/matteomartelli/ABPS`

Results in average

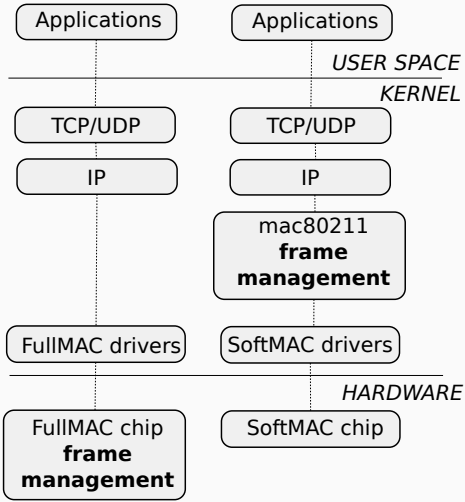
	`tedproxy` & TED	no `tedproxy` no TED	no `tedproxy` no TED with error app
average packet loss	36	364	450^{estimate}
average unavailability time	458 ms	7219 ms	10393 ms

Table 1: Results of conducted experiments in average.

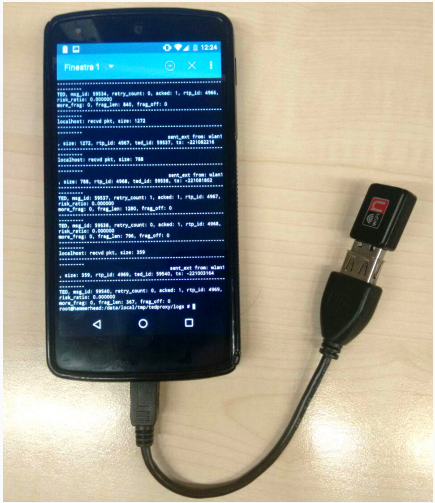
MN Implementation - Limitation



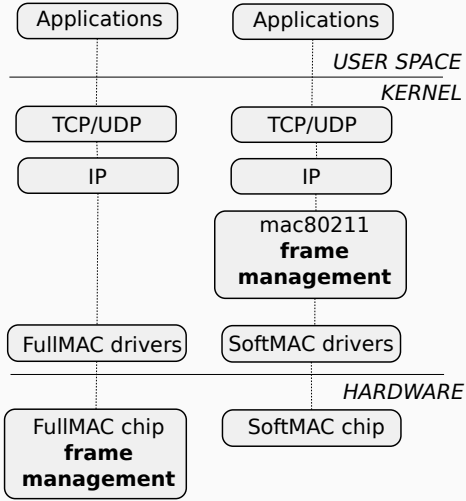
- No support for FullMAC chips



MN Implementation - Limitation



- No support for FullMAC chips



Future Works

- Integrate TED with `SO_WIFI_STATUS` socket option
- More handover criteria in client proxy
- Datagram retransmission on support NIC
- Integration of the *Oracle* to enable power saving
- Enable support for signalling, videoconference and VoIP application
- Find a solution for FullMAC chips. See NexMon project for reverse engineering and code injection into firmwares.