How to develop and validate a scalable mesh routing solution for IEEE 802.15.4 sensor networks

Altran Benelux

Leuven, 29 October 2015

Daniele Lacamera <daniele.lacamera@altran.com>

picoTCP

The reference TCP/IP Stack for the Internet of Things

Modularity	Portability
Performance	Quality

A fully featured

highly portable

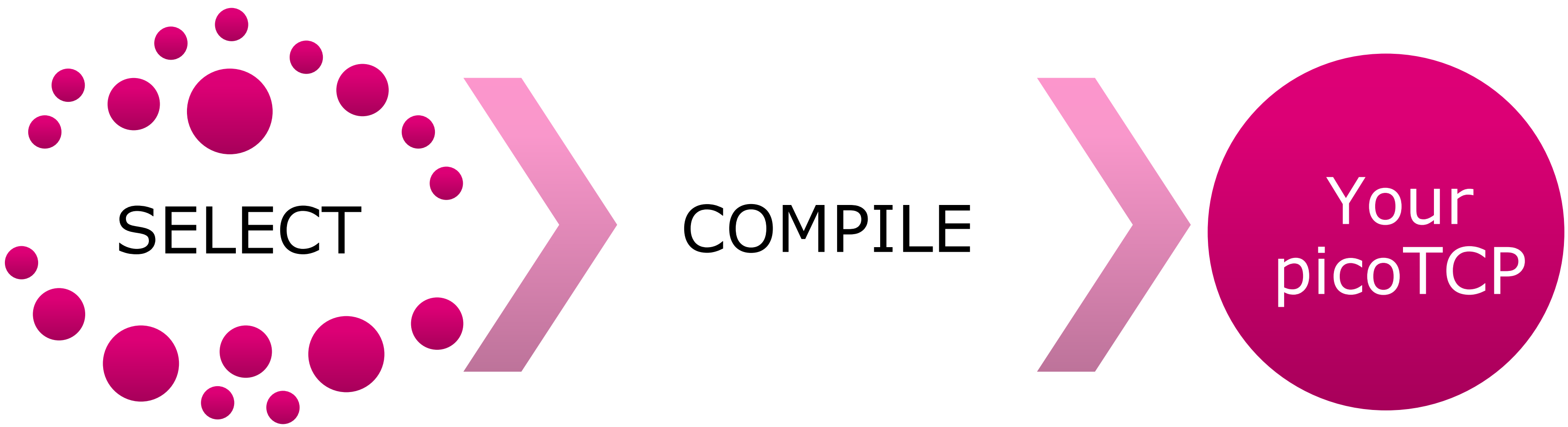
TCP/IP

Stack

designed for

embedded devices

Modularity




```
make ARCH=stm32 CROSS_COMPILE=arm-none-eabi- IPV4=1 TCP=1 UDP=1
```

Portability

- CPU Architecture independent
- 8, 16, 32 & 64 bit. Big or Little endian
- Bare Metal / Embedded OS / OS / RTOS

- 10+ different platforms already supported
- New platforms easily added

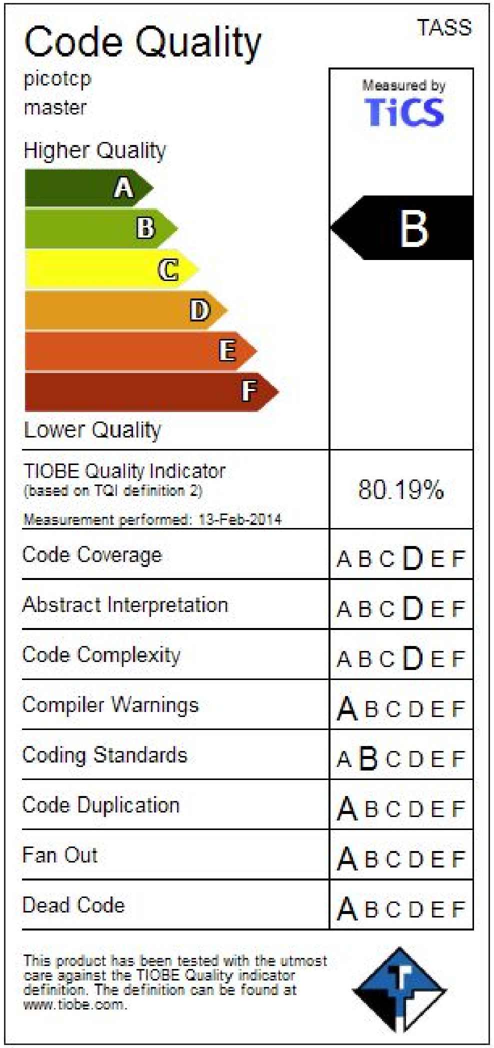
- (RT)OS easily added: e.g. FreeRTOS → 10 days

Type	Device	Porting effort
MCU	PIC24Fxxx	4 days
MCU	CC430, MSP430	1 day
Comm.	Broadcom BCM43362	3 days
Comm.	Microchip MRF24WG	2 days

Quality

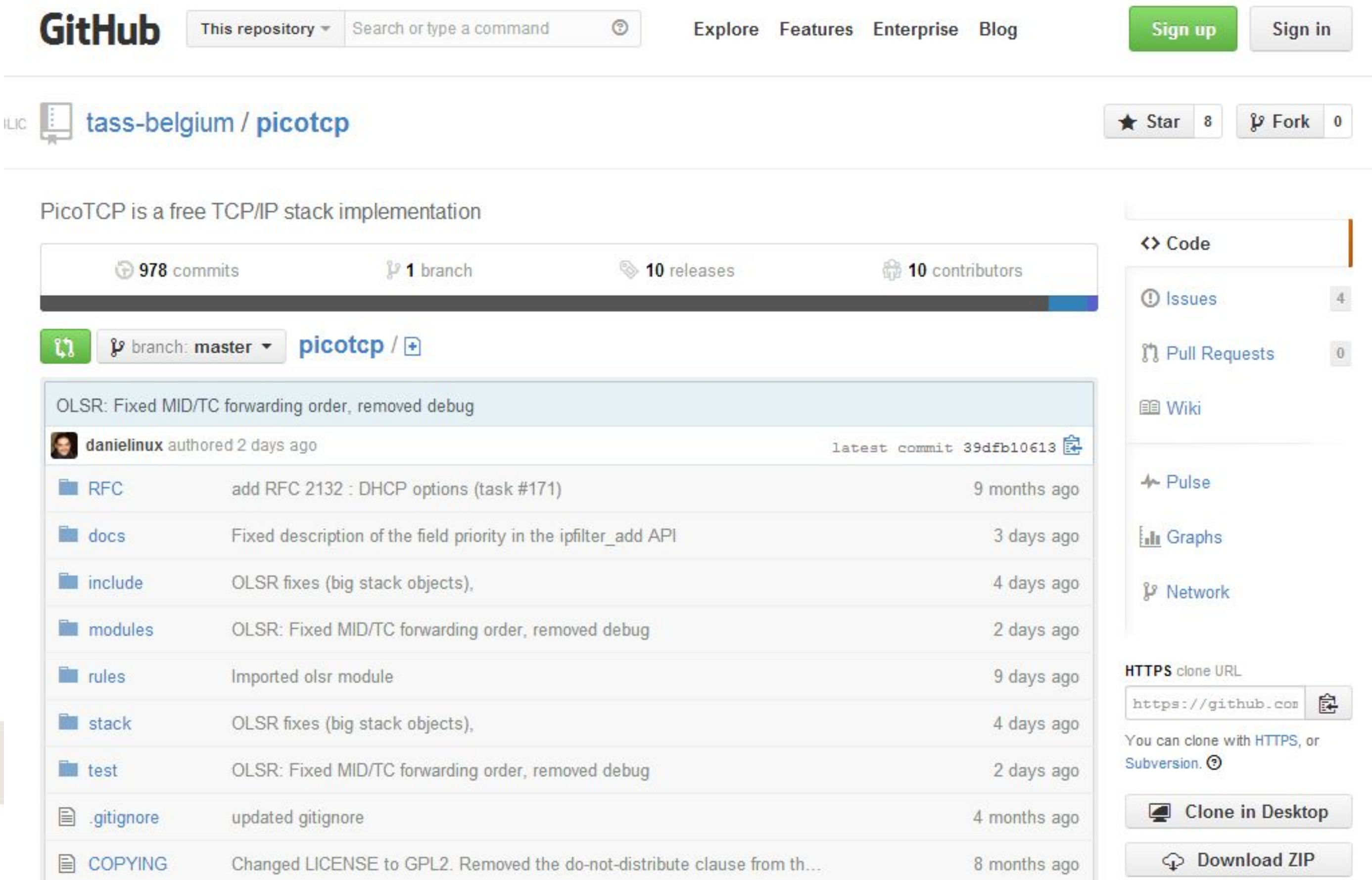
- **Quality oriented development environment**
  - Test Driven Development
  - Continuous Integration & Automated testing (Virtual testing is possible)
  - Code Quality check: Tiobe
    - Tics Continuous improvement
- **Full compliance with IETF TCP/IP standards**

RFC 768 User Datagram Protocol (UDP)
RFC 791 Internet Protocol (IP)
RFC 792 Internet Control Message Protocol (ICMP)
RFC 793 Transmission Control Protocol (TCP)
RFC 816 Fault Isolation and Recovery
RFC 826 Address Resolution Protocol (ARP)
RFC 879 The TCP Maximum Segment Size and Related Topics
RFC 894 IP over Ethernet
RFC 896 Congestion Control in IP/TCP Internetworks
RFC 919 Broadcasting Internet Datagrams
RFC 922 Broadcasting Internet Datagrams in the Presence of Subnets
RFC 950 Internet Standard Sub-netting Procedure
RFC 1009 Requirements for Internet Gateways
RFC 1034 Domain Names Concepts and Facilities
RFC 1035 Domain Names Implementation and Specification
RFC 1071 Computing the Internet Checksum
RFC 1112 Internet Group Management Protocol (IGMP)
RFC 1122 Requirements for Internet Hosts Communication Layers
RFC 1191 Path MTU Discovery
RFC 1323 TCP Extensions for High Performance
RFC 1337 TIME-WAIT Assassination Hazards in TCP
RFC 1534 Interoperation Between DHCP and BOOTP
RFC 1542 Clarifications and Extensions for the Bootstrap Protocol



picoTCP

- **Community driven**
  - Full source code freely available
  - Public issue-tracking system



picoTCP

- **Free/Open Source licensing policy in place**
  - Freely distributed under the GNU General Public License for the benefit of the community
  - Proprietary license available at user's option, when the platform code can't fully comply with the terms of GPL
  - Full copyright is owned by TASS: different licensing agreements are possible for special cases
- **github!**
  - http://www.github.org/tass-belgium/picotcp

Mesh networks

Nodes

- low-power
- limited resources (RAM, Flash)
- small payload
- short RF range
- limited availability (nodes turning off, moving off-range, ...)

Reaching a distant node in the mesh

- requires other node to forward the message
- every node in the path knows the route to destination

Mesh networks

- **Cognitive networking**
  - Each node adapts its routing table to reflect the current topology
  - Nodes communicates to each other to retrieve updated topology information

- **Protocols standardized by IETF**
  - OLSR - proactive dynamic routing
  - AODV - reactive dynamic routing
  - 6loWPAN - L2+IPv6 based routing

picoMesh

Technical Challenge

- port picoTCP to tinyOS
- establish TCP/IP routes on small nodes using IP Fragmentation
- test the dynamic routing mechanism on a real MESH network

Goals

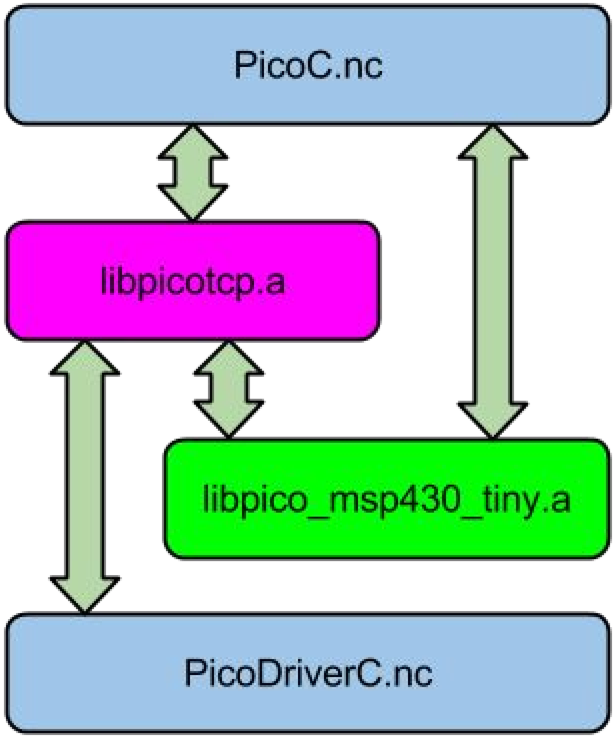
- provide picoTCP driver for IEEE802.15.4 radio device
- improve OLSR support
- integrate real test scenarios with emulation and virtualization tools
- release validated software as Open Source

picoMesh

- **Step 1: port picoTCP to tinyOS**
  - Modify tinyOS application Makefile to link with the picoTCP library
  - add a nesC wrapper for the picoTCP API
  - add a nesC wrapper for the radio Driver

picoMesh

- **Step 2: putting it all together**

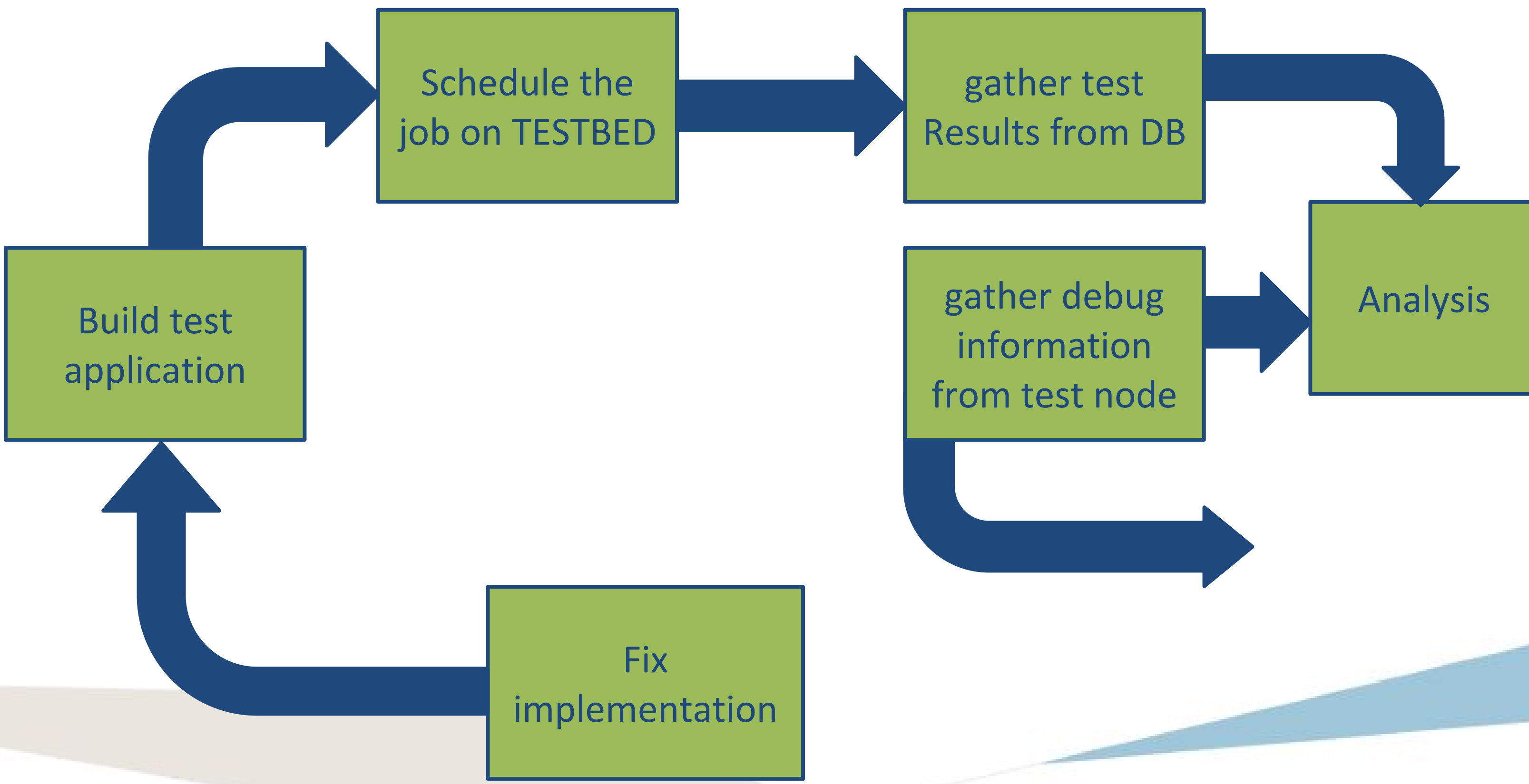


picoMesh

- **Step 3: Program the nodes in the testbed**

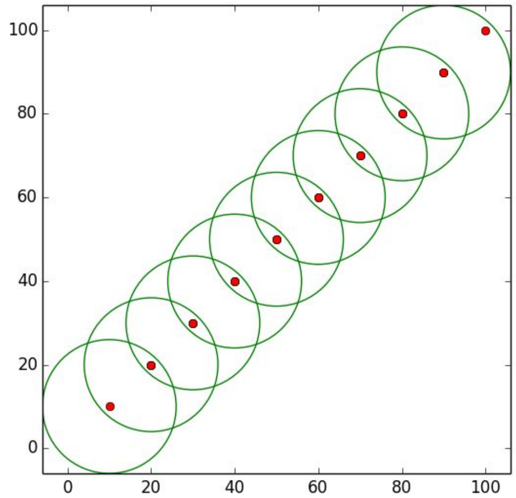
picoMesh

- **Once the development loop is in place: compile, test, debug, repeat**



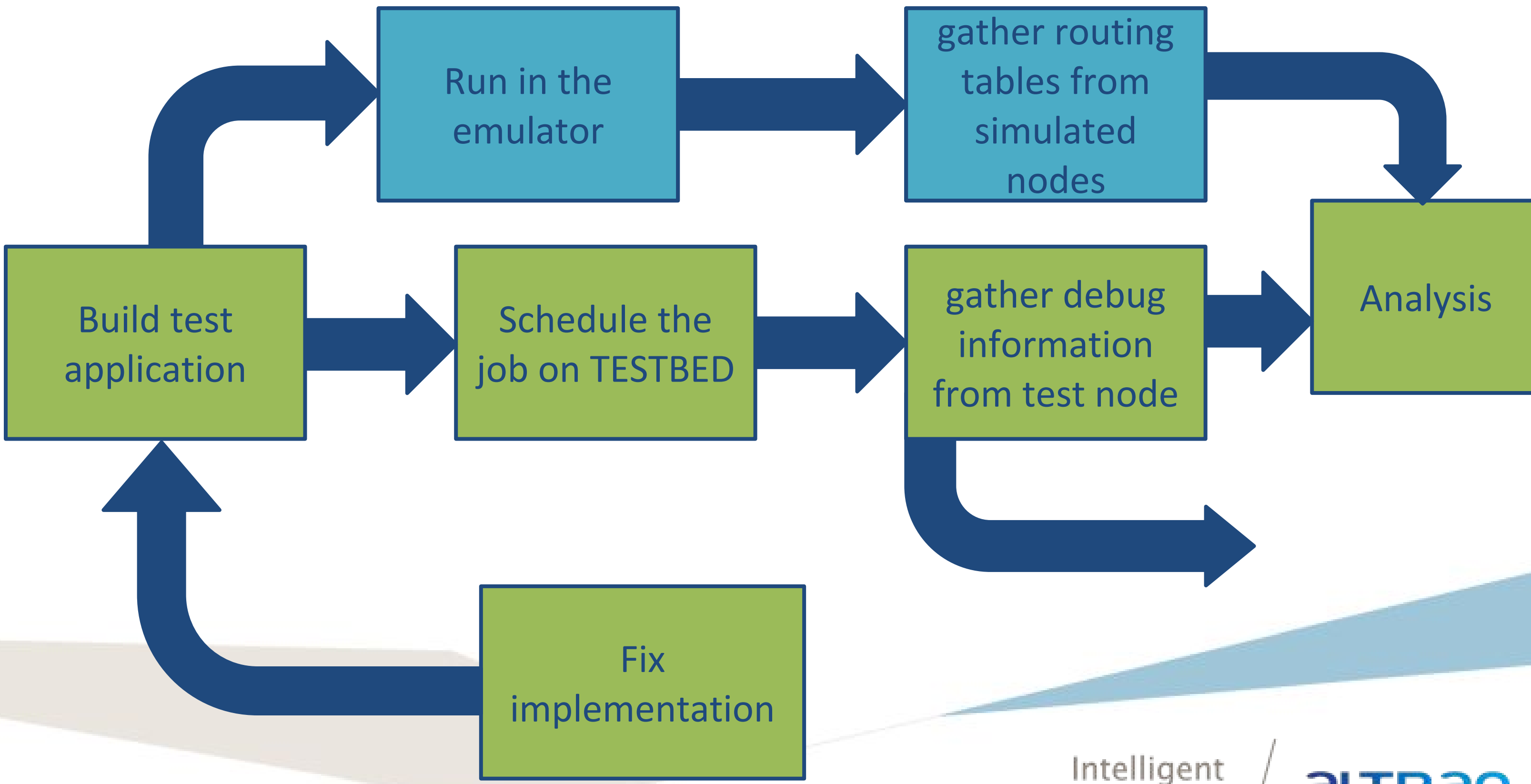
picoMesh

- **In order to speed up development, an emulator was written inspired by the wilab-t testbed**
  - released under GPL
  - available on github:
  - https://github.com/danielinux/geomess



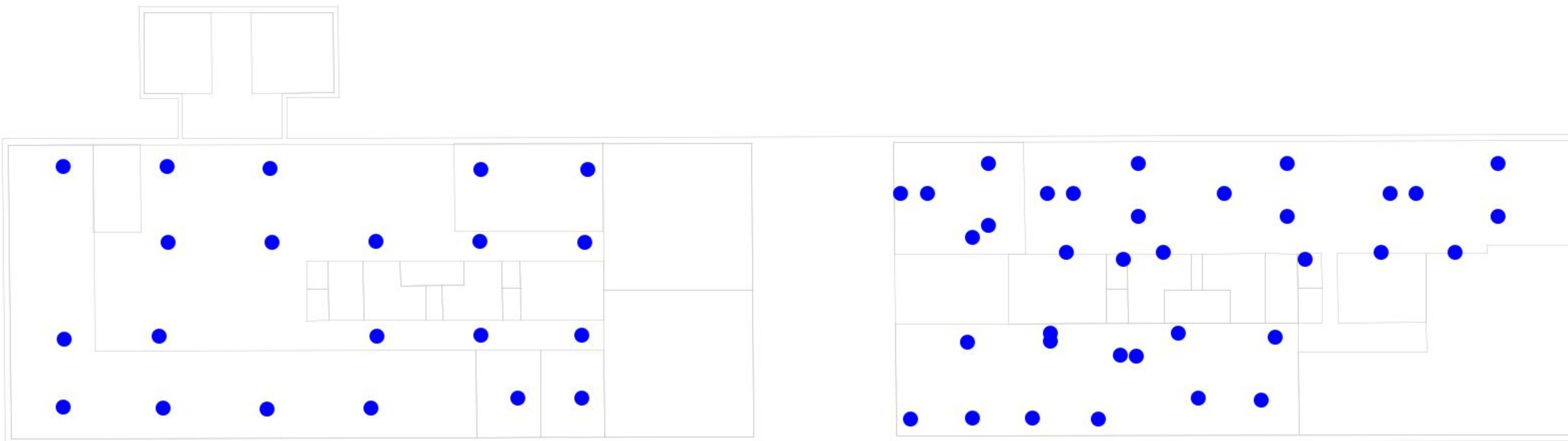
picoMesh

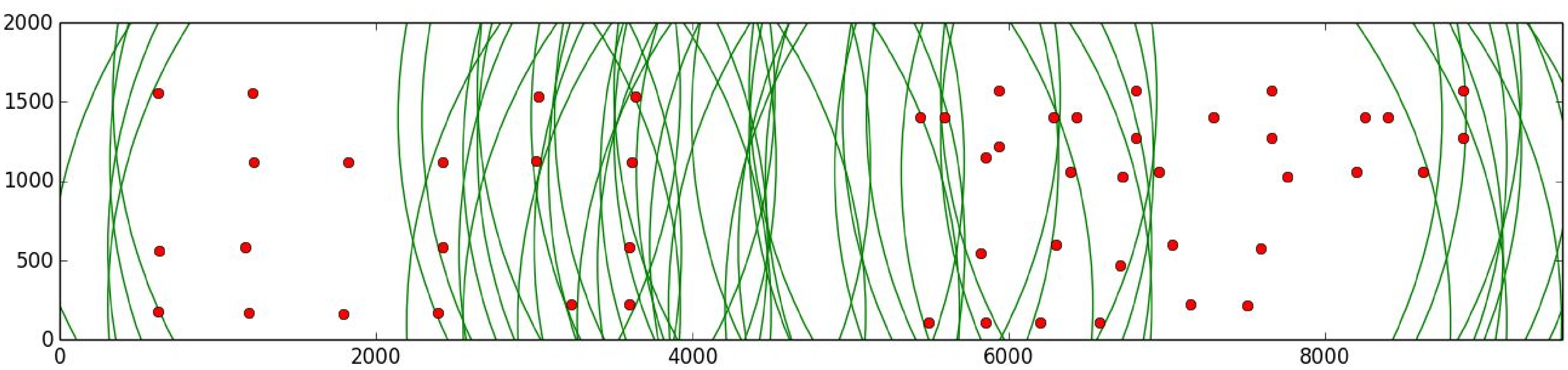
- **The emulator introduced a new workflow, to debug protocol-specific issues not related to the porting**



picoMesh

- **The emulator can reproduce the exact location of the physical nodes**





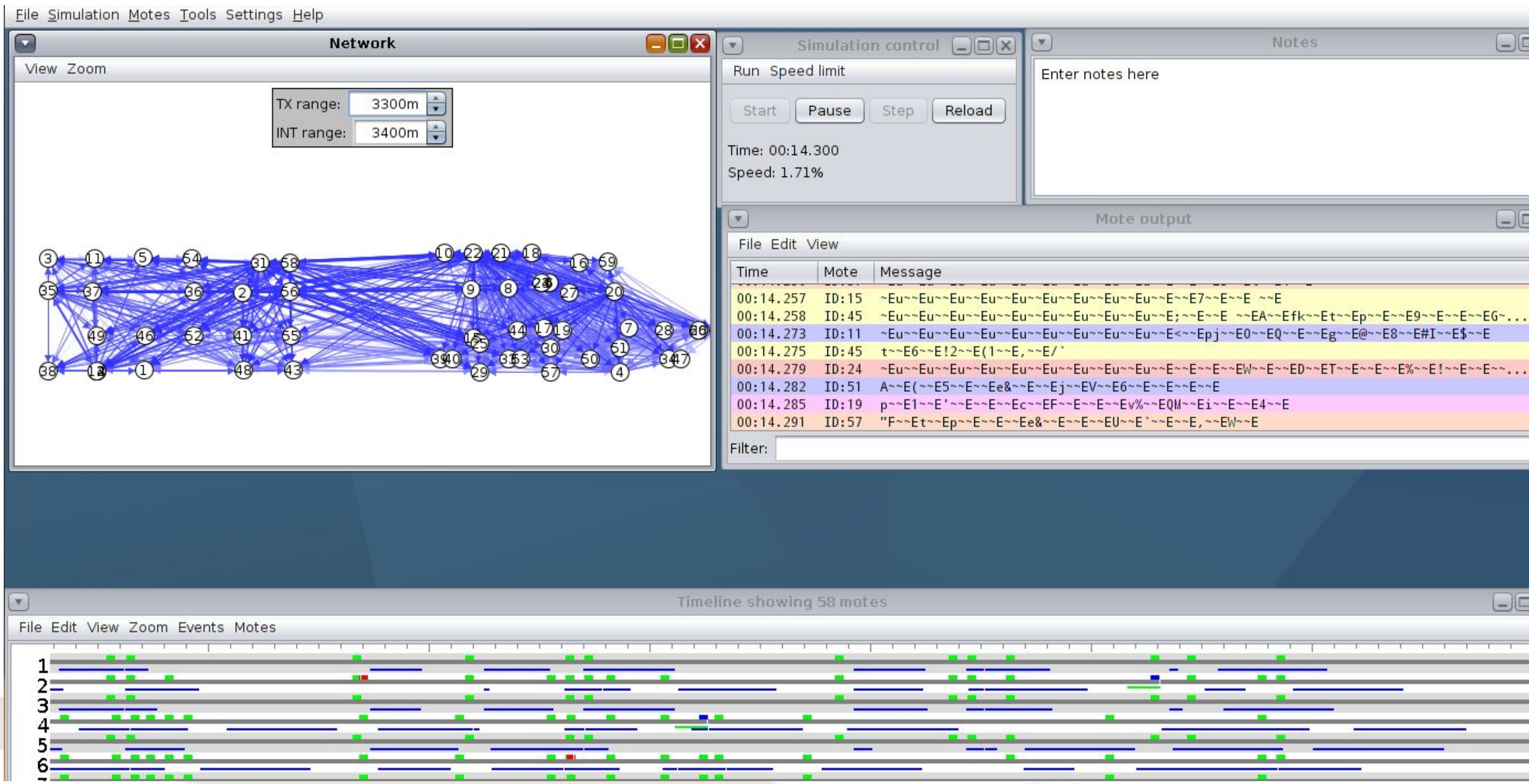
picoMesh

- **the routing table from a test node on the emulator is the same as the same node in the real testbed**
  - the emulator is validated

```
==== ROUTING TABLE =====
Route to 000000e0/000000f0, gw 00000000, dev: picomesh00aa, metric: 1
Route to 00002a0a/0000ffff, gw 00000000, dev: picomesh00aa, metric: 1
Route to 75002a0a/ffffffff, gw ba002a0a, dev: picomesh00aa, metric: 3
Route to 77002a0a/ffffffff, gw ba002a0a, dev: picomesh00aa, metric: 3
Route to 7b002a0a/ffffffff, gw ba002a0a, dev: picomesh00aa, metric: 2
Route to 7d002a0a/ffffffff, gw ba002a0a, dev: picomesh00aa, metric: 2
Route to 81002a0a/ffffffff, gw b9002a0a, dev: picomesh00aa, metric: 2
Route to 82002a0a/ffffffff, gw ba002a0a, dev: picomesh00aa, metric: 2
Route to 83002a0a/ffffffff, gw ba002a0a, dev: picomesh00aa, metric: 2
Route to 84002a0a/ffffffff, gw ba002a0a, dev: picomesh00aa, metric: 2
Route to 87002a0a/ffffffff, gw ba002a0a, dev: picomesh00aa, metric: 2
Route to 89002a0a/ffffffff, gw ba002a0a, dev: picomesh00aa, metric: 2
Route to 8c002a0a/ffffffff, gw ba002a0a, dev: picomesh00aa, metric: 3
```

picoMesh

- **Similar results can be obtained using Cooja, a wireless sensor simulator which is able to run the tinyOS firmware image linked with picoTCP**

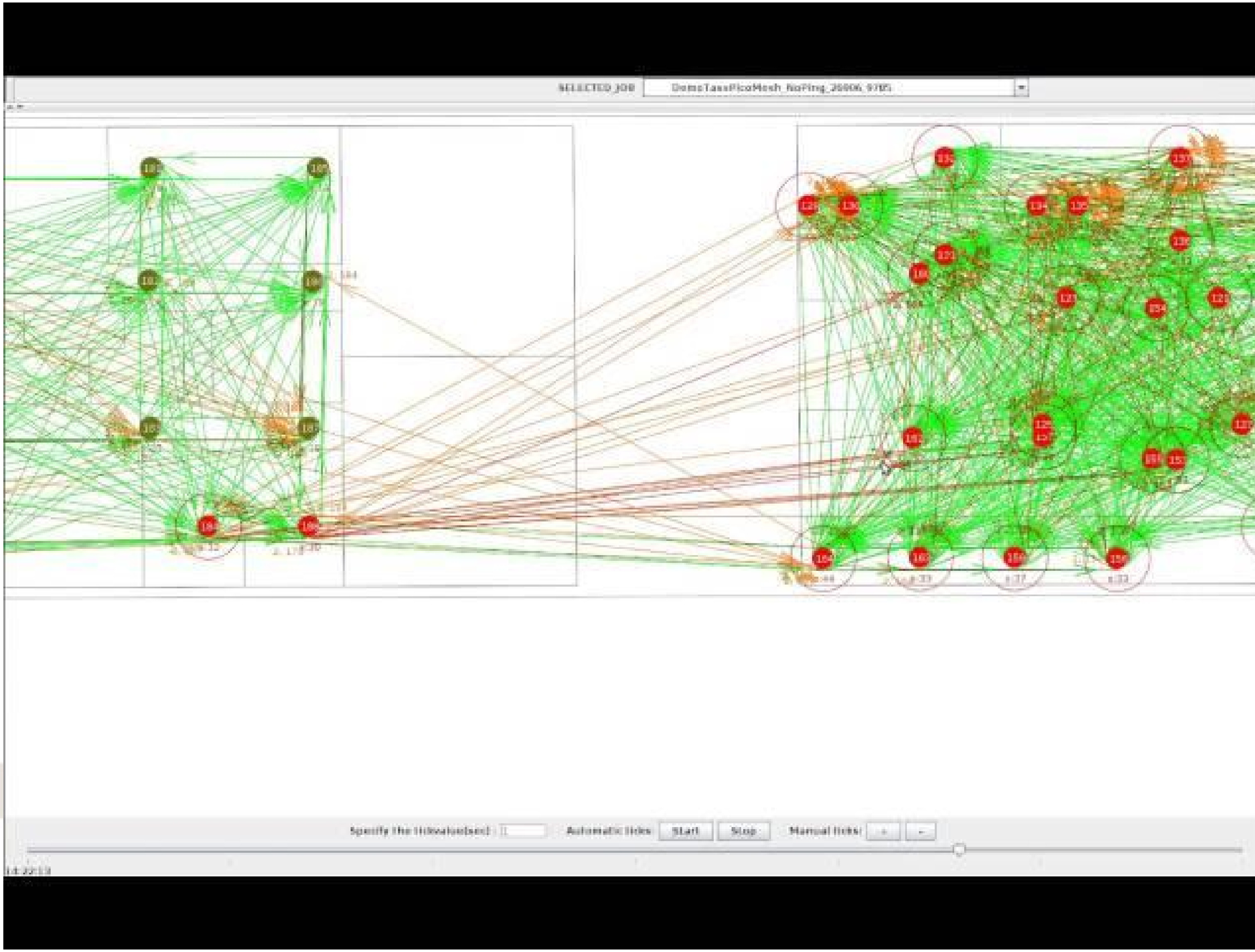


picoMesh: Crew testbed results

- **OLSR application running on wilab-t**
  - Nodes change colors depending on the number of route entries
  - Links have different colors, based on the number of hops:
    - green indicates 1-hop links
    - orange indicates 2-hops links
    - red indicates >2-hops links

picoMesh: Crew testbed results

- **OLSR application running on wilab-t**



picoMesh: project conclusions

- successfully ported picoTCP to tinyOS
- developed an Open Source MESH network emulator, based on geographical positions of the nodes and their ranges
- improved OLSR (RFC3626) implementation for picoTCP
- proven feasibility of real IPv4 MESH networking over a IEEE802.15.4 topology
- solved the hidden-node problem using dynamic routing

Feedback

- w-iLab.t eases to scale the number of nodes
- using JIRA as an issue tracker is a very efficient way of problem solving
- CREW team has been very supportive during the whole duration of the experiments
- All of this has helped a lot to reach the expected results

Statistics

- Nr of reservation slots: 223
- Avg duration of reservation slot: 14 minutes
- Avg number of wireless nodes: 28.9
- Avg number of mobile nodes: 0
- Total duration of reservations: 2.2 days
- First slot: March 17th 2014
- Last slot: November 14th 2014

Recent improvements

- **January 2015: the company becomes “Intelligent Systems/Altran”. The focus remains on research and development of advanced networking solutions**
- **Looking for alternatives to OLSR in the field of cognitive networking**
- **Q1.2015: Implemented Ad-hoc On-demand Distance Vector Routing (AODV) following specifications from IETF RFC 3561**
- **Q3.2015: Experimental implementation for 6LoWPAN, including 802.15.4 Datalink layer mesh topologies, following specifications from IETF RFC 4944**

AODV

- **Different approach from OLSR**
- **On-demand route discovery (reactive rather than proactive topology composition)**
- **More ultra-low power friendly**
  - routes are discovered only when needed)
  - No overhead traffic generated by the discovery algorithm when no payload traffic is present
- **Solution based on network layer routes, applicable to different physical layers (802.15.4, 802.11 ad-hoc, etc.)**
- **Part of design and implementation is in common with 802.11s**
- **Link-failure awareness**

6LoWPAN

- Based on datalink layer addresses
- Native solution for compressing and fragmenting IPv6 packets
- De-facto standard for 802.15.4 networks, proposed by IEEE
- Support for Broadcast, Multicast and Unicast traffic
- Built-in mesh solution for multi-hop routing
- Cross-layer extensions for IPv6 Neighbor Discovery algorithm (RFC 6775)
- Production ready in picoTCP by Q1.2016

Future experiments

- **The validation of the three available solution and the performance comparisons can benefit from the use of a real Testbed like the one provided by CREW**
- **Implementing the newer cognitive networking solutions would only require a small porting effort**
- **Other parameters can be measured and compared**
    - power consumed
    - worst-case route determination time
    - impact of fragmentation strategies on throughput and packet loss
    - topology robustness against node failures

Thanks!

Questions?

Daniele Lacamera <daniele.lacamera@altran.com>