UNIVERSITÀ
POLITECNICA
DELLE MARCHE

ToLHnet: A Low-Complexity Protocol for Mixed Wired and Wireless Low-Rate Control Networks

Giorgio Biagetti, Paolo Crippa, Alessandro Curzi, Simone Orcioni, Claudio Turchetti

{g.biagetti, p.crippa, a.curzi, s.orcioni, c.turchetti}@univpm.it
DII — Dipartimento di Ingegneria dell'Informazione
Via Brecce Bianche 12 — 60131 Ancona — Italy

6th European Embedded Design in Education and Research Conference

EDERC 2014 - September 11–12, 2014 - Milan, Italy

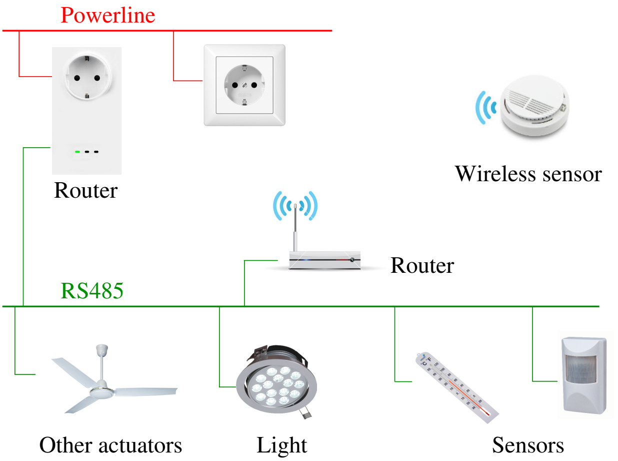
Outline

1. **Introduction**
2. **Protocol definition**
   - Network architecture
   - Routing strategy
   - Packet structure
3. **Implementation on a TM4C123GH6PMI**
   - External interfaces
   - Firmware architecture
4. **Experimental evaluation**
5. **Conclusions**

Introduction

- **ToLHnet**: Tree or Linear Hopping network
  - based on tree routing → simple implementation on nodes
  - special care to support the degenerate case of linear routing
- **key features**:
  - seamlessly support mixing wired and wireless media
  - relatively large address space: about 60000 nodes
  - low network overhead
    - typically 4 bytes header size
    - no need for MAC-layer addressing
    - payloads up to 240 bytes
  - low complexity of node firmware (<12 kB on 32 bit ARM μC)
  - higher complexity code running only on the master controller node

Application context: example



Application context: case study

- protocol implemented on a Texas Instruments TM4C123GH6PMI using Tiva-C TM4C123GXL evaluation boards
- power-line-communication (PLC) based on the ST7580 modem by STMicroelectronics externally connected through a UART port
- RF communication based on the HopeRF RFM23B module externally connected through an SPI port
- serial-line communication using internal UART interfaces (could be made RS-485 compatible through an SN65HVD11 transceiver, expansion board currently under development)

Network architecture: definitions

- **Transmission medium**:

the network can span multiple physical media.
Each is described by 3 properties: **bandwidth**, **latency**, **range**.
Example of media names: **SL**, **RF**, **PL**, . . .

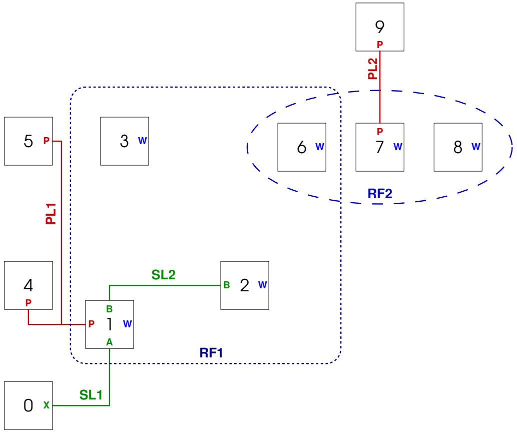
- **Interface**:

the interconnection between the node and the medium.
Example of interface names: **X**, **A**, **B**, **W**, **P**, . . .

- **Physical broadcast domain (PBD)**:

a set of nodes that share the same transmission medium
e.g. **SL1**, **SL2**, **RF1**, **RF2**, **PL1**, **PL2**, . . .

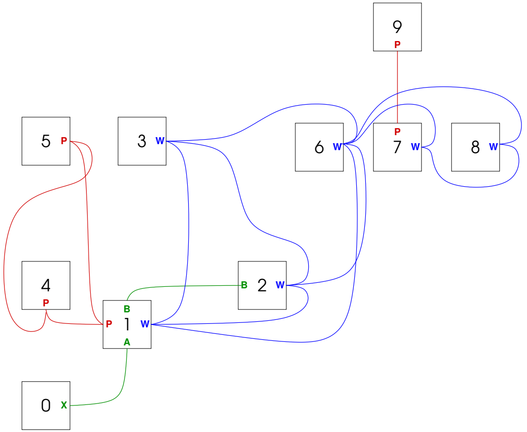
Network architecture: physical topology



```
0 X=SL1
1 A=SL1 P=PL1(0.0) \
  B=SL2 W=RF1(1.2,1.2)
2 B=SL2 W=RF1(4.0,1.0)
3 W=RF1(1.5,4.0)
4 P=PL1(1.3)
5 P=PL1(4.1)
6 W=RF1(5.2,3.9) \
  W=RF2(-1.5,0)
7 W=RF2( 0.0,0) P=PL2
8 W=RF2(+1.5,0)
9 P=PL2
```

A simple example of node positions and connections to PBDs.
(numbers in parenthesis are geometrical, physical coordinates)

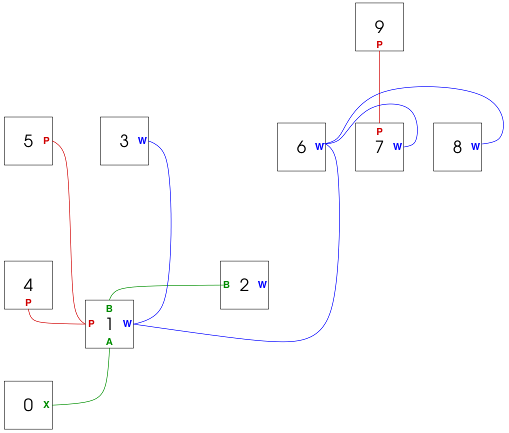
Network architecture: mesh of possible links



a **weight** will be given to each graph **edge** based on a cost factor depending on medium (bandwidth & latency) and distance spanned w.r.t. medium's range.

Mesh derived by fully connecting nodes within PBDs.

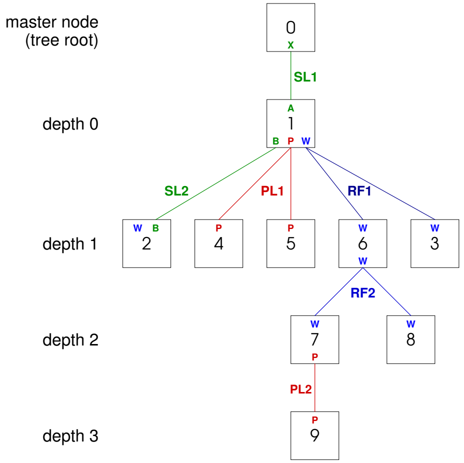
Network architecture: extracted tree



Node connection tree.

a **tree** is extracted from the graph, e.g, by means of *Dijkstra*'s algorithm. It will define the **minimum cost** path from node 0 (master) to any other node.

Network architecture: logical topology



Routing tables:

- node 0 (master):
  `1-9 X`
- node 1:
  ```
   2  B
  4-5 P
  3-9 W
  ```
- node 6:
  `7-9 W`
- node 7:
  ` 9  P`

A simple example of node addressing showing the logical connectivity.

Addressing

There are two kinds of addresses:

- hardware address, 48 bits long,
  (sometimes referred to as MAC address)
  that uniquely and permanently identifies each node.
- network address, 16 bits long,
  assigned by the master during network configuration.
  The master always has network address 0.

MAC addresses are typically used only during network configuration, when peripheral nodes do not yet possess usable network addresses. Then, in normal usage, only network addresses are used.

Additionally, each node possesses a configurable

- depth level, 16 bits long
  assigned by the master during network configuration.

Routing strategy I

The routing strategy must ensure that packets follow the branches of the tree — “side-edge” receptions must be discarded!

Each packet contains the following information used for routing:

- MAC address and target DEPTH (optional fields)
- HOPS count (current tree depth of the packet)
- Direction (towards tree leaves (+1) or root (–1))
- SRC address
- DST address

plus a SEQuence number used to detect repeated transmissions.

Routing strategy II

- **Routing algorithm**:

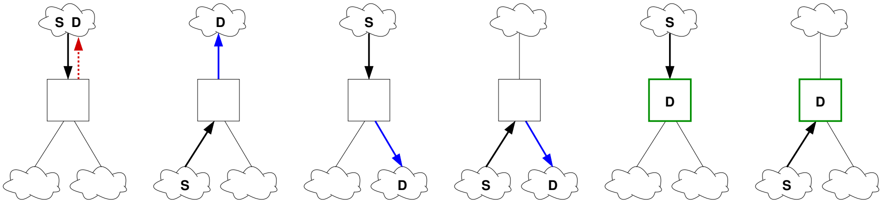
1. if packet contains a matching MAC, and DEPTH = HOPS, accept.
2. if DEPTH ≠ HOPS, discard.
3. lookup SRC and DST addresses on the routing table and determine their "side":
   - −1 if attached to the parent-side
   - 0 if it's the node address itself
   - +1 if attached to the children-side

   SRC side = 0 shouldn't happen, so there are 12 possibilities to consider: 2 SRC sides × 3 DST sides × 2 Directions.
4. if SRC side = Direction, discard.

   6 possibilities excluded: packet from above going further up, or from below going further down, irregarding of destination. Some of these combinations only occur in malformed packets.

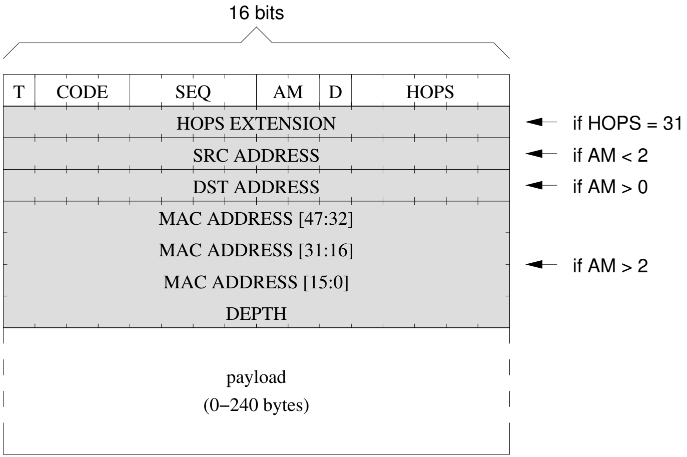
Routing strategy III

Only six possibilities remain:



5. if SRC side < 0 and DST side < 0, discard.
6. if DST side ≠ 0, forward.
7. accept.

Packet format



Network-level packet structure, header is typically just 32 bits long.
(Presence of gray fields depends on the value of other flags as indicated.)

Addressing modes

AM	fields present
00	SRC
01	SRC + DST
10	DST
11	DST + MAC

These two bits specify which addresses are present in the header. A missing network address is assumed to be 0 (the master node).

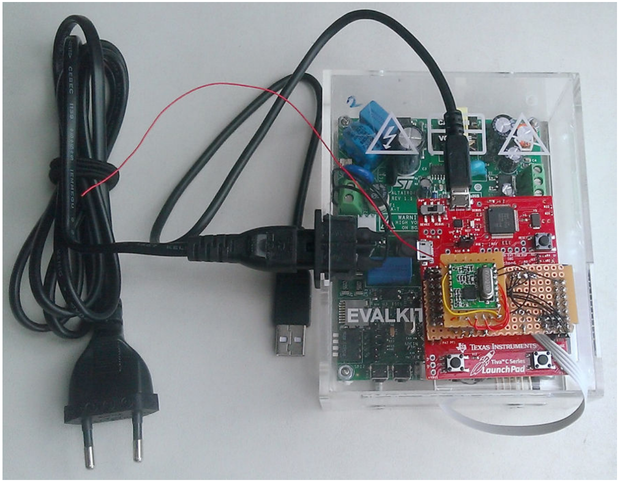
The MAC address can only be specified together with the destination network address (needed to route the packet). Only the master can set up the network, so there is no need to also specify the source.

Command codes

CODE	T=0 (nwk layer)	T=1 (app layer)
000	ACK	ACK
001	NACK e	NACK e
010	GET r	GET r
011	TRACE	GET t r
100	MSG r	MSG r
101	PING	GET t n r
110	SET r	SET r
111	CONFIG	SET t r

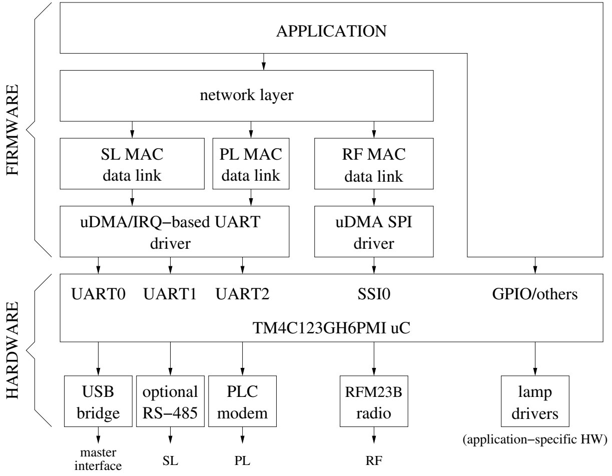
e, r, t, n are fields placed at the beginning of the payload.
e: error code in case of a NACK, r: register number to be set or get or signaled, t: timeout for the operation, n: limit on acceptable reply size.

System implementation



Photograph of one of the nodes assembled for the experimentation.

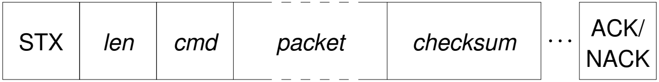
System architecture



Firmware and hardware block diagram.

UART drivers

Shared code between PLC modem and direct serial line connections.



- **TX**: can be done entirely within a single uDMA transaction.
  - Data still need to be checksummed by the CPU beforehand.
- **RX**: need to detect inter-character timeout for framing!
  - uDMA cannot be used if data length is unknown.
  - FIFO will still relieve the CPU of many interrupts:
    at every FIFO event (FIFO full or timeout)
    a timer is reset to detect the inter-character timeout.

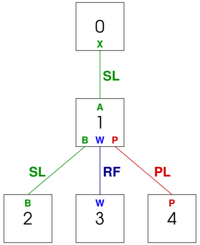
SPI driver

Freescale SPI mode 3: framing on SS line done by hardware.
RFM23B needs to slice a packet on more than one SPI frame.

- **TX**: 8 bit register number followed by data
  - scatter-gather uDMA drives TX side of SSI:
    first transaction sends register number,
    second transaction sends desired slice of packet data.
  - simple uDMA drives RX side of SSI:
    data is simply thrown away to keep RX buffer empty.
- **RX**: need 8 bit register number, half-duplex operation
  - simple uDMA drives TX side of SSI:
    sends desired register number, then dummy data to clock the bus.
  - scatter-gather uDMA drives RX side of SSI:
    first transaction discards dummy read from register number,
    second transaction reads data into desired slice of packet buffer.

Experimental setup

PINGs directed to the first four nodes, connected with different media:



		overhead		
IF	raw speed	PHY	MAC	DL
SL	1000 kb/s	25%	—	6 B
RF	100 kb/s	6 B	2 B	3 B
PL	9.6 kb/s	9 B	—	5 B

Round-trip-times, as seen from the master node 0
pinging nodes 1, 2, 3, 4, are recorded.

Experimental results

		target node			
payload	CPU	**1** (SL)	**2** (SL+SL)	**3** (SL+RF)	**4** (SL+PL)
240 B	7.1%	5.4 ms	10.6 ms	47.0 ms	656.5 ms
64 B	7.0%	1.8 ms	3.4 ms	15.2 ms	208.9 ms
16 B	6.8%	0.8 ms	1.4 ms	6.5 ms	86.7 ms
4 B	6.5%	0.5 ms	0.9 ms	4.3 ms	56.1 ms
0 B	6.6%	0.4 ms	0.8 ms	3.6 ms	46.0 ms

PING round trip time for various payload sizes,
and different combinations of transmission media,
CPU utilization is for the serial line case (node 1).

Analysis of protocol and implementation efficiency

- c: raw channel capacity
- x_ℓ: layer ℓ overhead, $\ell \in \{\mathrm{PHY}, \mathrm{MAC}, \mathrm{DL}, \mathrm{NET}\}$
- $f_\mathrm{a} = 1/\prod_\ell (1 + x_\ell)$: available fraction of channel capacity
- $f_\mathrm{e} = \frac{n}{c} \cdot \frac{2}{t_{\mathrm{SL}+m} - t_\mathrm{SL}}$: measured capacity, $m \in \{\mathrm{SL}, \mathrm{RF}, \mathrm{PL}\}$

Results shown for payload sizes of $n = 240$ bytes:

	SL	RF	PL
PHY+MAC+DL overhead	28.1%	4.5%	5.7%
network layer overhead	1.7%	1.7%	1.7%
available channel capacity f_a	76.8%	94.1%	93.0%
measured channel capacity f_e	73.8%	92.3%	61.4%
channel occupation $f_\mathrm{e}/f_\mathrm{a}$	96.1%	98.1%	66.0%

Conclusions

- **ToLHnet**:
  - low-overhead open-source network layer
  - asymmetric implementation allows for large networks
    - very small footprint on device nodes
    - complexity moved to the master controller
  - effectively supports a variety of transmission media
  - transparently handles mixed wired and wireless links
  - features a custom tree-based routing scheme
  - scales up to the degenerate case of linear routing
  - efficiently implemented on a TM4C123GH6PMI microcontroller
    - high-efficiency drivers developed for serial links
    - drivers for external packet radios and PLC modems also available

Thankings

- **Thank you for your attention!**

The ToLHnet source code is freely available at:

www.tolhnet.org