Alessandrini *et al. EURASIP Journal on Embedded Systems* (2017) 2017:7
DOI 10.1186/s13639-016-0042-x

EURASIP Journal on Embedded Systems

RESEARCH Open Access

Optimizing linear routing in the ToLHnet protocol to improve performance over long RS-485 buses

Michele Alessandrini, Giorgio Biagetti*, Paolo Crippa, Laura Falaschetti, Simone Orcioni and Claudio Turchetti

Abstract

As the adoption of sensing and control networks rises to encompass the most diverse fields, the need for simple, efficient interconnection between many different devices will become ever more pressing. Though wireless communication is certainly appealing, current technological limits still prevent its usage where high reliability is needed or where the electromagnetical environment is not really apt to let radio waves through. In these cases, a wired link, based on a robust and well-consolidated standard such as an RS-485 bus, might prove to be a good choice. In this paper, we present an extension to the routing strategy originally implemented in the recently proposed "tree or linear hopping network" (ToLHnet) protocol, aimed at better handling the special but important case of linear routing over a (possibly very long) wired link, such as an RS-485 bus. The ToLHnet protocol was especially developed to suit the need of low complexity for deployments on large control networks. Indeed, using it over RS-485 already makes it possible to overcome many of the traditional limitations regarding cable length, without requiring segmenting the bus to install repeaters. With the extension here proposed, it will also be possible to simultaneously reduce latency (i.e., increase throughput, should it matter) for short-distance communications over the same cable, largely increasing the overall network efficiency, with a negligible increase in the complexity of the nodes' firmware.

Keywords: Routing protocols, Sensor networks, Control networks, Linear routing

1 Introduction

Sensing and control networks are becoming ubiquitous in many application fields, including industrial and building automation, intelligent lighting control, body area networks, and environmental monitoring. They are promising for their ability of potentially lowering the maintenance, installation, and refurbishment costs, increasing situation awareness, and allowing useful data to be gathered in difficult situations.

Nevertheless, widespread interconnection of the most simple devices is often hindered by the cost associated with adding the communication and processing resources needed to allow them to talk to the network. To mitigate these cost problems, the development of a simple network protocol that is able to let thousands of devices talk over a variety of transmission media, while requiring very limited computational resources and small memory footprints on the nodes, would surely help.

Recently, a simple and efficient network protocol suitable to be implemented in nodes with limited computational resources has been proposed [1], with details of the network-level packet structure, an outline of the addressing mechanism employed for unicast routing, and a freely available reference implementation [2].

It was designed from the grounds up to allow a strongly asymmetrical implementation, i.e., very simple code on the network (slaves) nodes that rely on a much more complex implementation on the master node. Moreover, a clever bit-packing of the packet header optional addresses and fields allowed a very low-overhead implementation across a variety of transmission media, resulting in a very-high maximum channel occupancy for many of them.

To achieve this performance, the routing mechanism devised was limited to be tree-based, i.e., packets were

*Correspondence: g.biagetti@univpm.it
Department of Information Engineering, Università Politecnica delle Marche, Via Brecce Bianche 12, 60131 Ancona, Italy

only allowed to follow the branches of a predetermined tree that spans the whole network. Though many algorithms exist to build such a suitable tree, in practice, gathering the necessary information to run them, and tuning their parameters to find the optimal routing tree, turns out to be a non-trivial task.

Indeed, the original paper only focused on the slave-side (end-device) implementation and processing, leaving the details of the much harder task of the master-side processing open to further research.

The scope of this paper is thus to extend such work, focusing on the special but important case of routing over a linear medium [3], with the aim of further improving the performance attainable over possibly very long RS-485 buses.

This paper is organized as follows. Section 2 presents a brief discussion of previous works related to this subject. Section 3 gives a brief summary of the "tree or linear hopping network" (ToLHnet) protocol, while Sect. 4 describes the details of the proposed routing strategy. Simulation and experimental results are given in Section 5, while conclusions are drawn in Section 6.

2 Related works

A lot of research is devoted to trying and make the so-called Internet-of-Things possible by using mostly wireless communications media [4–7]. Nevertheless, many applications require the intrinsic safety and reliability of a wired medium [8], or need higher or predictable bandwidth, or operate in an unfavorable electromagnetic environment [9]. Some devices might still need to be continuously powered to operate, and these might all be reasons to make a cable the preferred medium over a wireless link.

The RS-485 bus [10] is an extremely well-consolidated and widespread standard for medium-range communication between industrial devices (up to about 1 km) over a simple twisted-pair cable [11]. Unfortunately, traditional systems communicating over RS-485 face a trade-off between cable length, which constitutes one of the most important factors that determine bus loading, and the employed baud rate, to ensure end-to-end signal integrity.

This is not a limitation of the medium itself. It originates from the fact that protocols operating over RS-485 typically assume that each device is able to communicate directly with any other device on the bus, without the need of dealing with the complexities of routing. When the need to span greater distances than allowed arises, two separate buses can be joined together by means of specialized RS-485 repeaters that bridge the cables, regenerating the signal.

Alternatively, application-specific solutions need to be developed. For instance, Modbus nodes' [12] firmware can be modified to double the maximum distance, by allowing devices to repeat queries. Still, the lack of explicit routing capabilities in the underlying protocol makes the solution cumbersome, as it must rely on timeouts to avoid collisions, and so performance is limited.

Repeaters, on the other hand, do not limit performance, but they do not come without problems anyway. Using them requires to physically segment the network, and their fixed placement reduces installation flexibility, besides adding to the total system cost and management complexity. They also reduce the overall network reliability, introducing points of failure that may isolate sizeable portions of the network.

It should be apparent that adding intrinsic routability in the protocol used to talk over the bus might be the best solution. Preliminary experiments [3] showed that using ToLHnet [1] over RS-485 practically lifts all limits about total cable length, allowing each node to repeat the signal if need be, and can also be nearly optimal in terms of attainable data rate. This paper builds upon these results to detail the routing strategy that can be established to purse the maximum throughput (or, which is the same in this case but can often matter most, the lowest latency), and a simple modification to the original routing algorithm is proposed to further increase the performance.

3 ToLHnet protocol overview

The ToLHnet protocol sits at the network layer of the ISO/OSI model. It is able to span different transmission media, being largely unaware of the details of the underlying data link, media access control (MAC), and physical layers. The physical network topology is arbitrary, and it does not need any MAC-level addressing or filtering capabilities. Nodes are permanently identified by 48-bit hardware addresses and are automatically assigned temporary 16-bit network addresses upon network configuration.

ToLHnet is by design strongly asymmetrical. There exists a master node, at address 0, which knows the whole network topology and is equipped with quite complex software able to compute routes and configure the network. All the other nodes are slaves, requiring very little code and memory space to run the protocol and forward packets.

It is also extremely lightweight: typically only 4 bytes of network header suffice to route packets between the master and any other node, in any direction, and 6 bytes suffice for communication between two generic nodes.

3.1 Network topology and tree routing

To simplify routing, a logical network topology, consisting in a tree rooted at the master node, is built on top of the physical topology. Details on how this tree is built will be given in the next section.

If packets are only allowed to follow the branches of a tree, routing tables will be greatly simplified, containing

only one entry for each physical interface with children attached (assuming they are assigned a contiguous span of addresses) and a default route to direct all other destinations to the parent.

To enforce this, each node must know its own depth in the tree (i.e., the number of hops separating itself from the root), and every packet traveling on the network must also contain a depth level that will be compared by each node to their own. Only packets with a matching depth will be acted upon.

This, together with a "direction" flag also contained in the packet header and denoting whether the packet is traveling towards the root or towards a leaf, is enough to ensure that packet retransmission/processing only follows tree branches and no collisions are ever generated during packet forwarding [1].

This is of course guaranteed to work even if the physical topology of the network is such that many other nodes besides the intended recipients are actually able to sense the packet on the media. They will simply ignore it.

These provisions help keep routing tables small and packet header overhead at bay, since each traveling packet only need to contain the (possibly compressed) endpoint addresses, there is no need to carry and store next-hop addresses, whose job is essentially replaced by the depth field (which for typical trees can usually be stored in far fewer bits than the next-hop address would have otherwise required).

3.2 Packet processing

In order to understand how to optimize the routing algorithm, a deeper understanding of how nodes process incoming packets is needed.

First of all, the network header of a packet specifies the value of at least the following fields (identified by uppercase names):

- `SRC`: the 16-bit network address of the originating node
- `DST`: the 16-bit network address of the destination node
- `HOPS`: current routing depth of the packet, automatically updated by routers as the packet flows through the network (maximum of 16 bits)
- `DIR`: packet direction flag (1 bit)

Each node has the following information (dynamically assigned by the master, identified by lowercase names):

- `address`: the 16-bit local network address
- `depth`: the 16-bit node depth in the logical tree

Finally, each entry in the routing table contains the following fields:

- `span`: a pair of network addresses used for matching the rule, every address numerically within the specified range matches
- `iface`: the physical network interface associated to this route
- `dir`: the direction, i.e., the value to add to HOPS when following this route.

As a packet arrives, its `HOPS` field is compared to `depth` and discarded if they do not match. Then, `SRC` and `DST` are looked up in the routing table, and the packet discarded if `SRC`'s `dir` and `DST`'s `dir` are both negative (i.e., both addresses matched the default route) or if `SRC`'s `dir` sign equals `DIR`'s (can happen when a packet unrelated to the current tree branch was nevertheless sensed on an interface). Otherwise, the packet is either accepted locally or forwarded according to whether or not `DST` equals `address`. An example of how these rules affect packet forwarding will be given in the next section.

4 Routing algorithms

In order to optimally route packets within the network, a basic knowledge of the properties of the used media is necessary. This knowledge pertains to the master and consists essentially of the network topography and an indicative transmission range r of each medium, where by "indicative," we mean the distance above which it would be helpful to find an intermediate repeater.

4.1 Mesh and weights generation

For each pair of nodes sharing a common physical transmission medium, a link is put in the mesh and associated with a weight (or cost) $c(d_n)$, given by

$$c(d_n) = a\left[1 + 2\left(\frac{d_n}{r}\right)^2\right], \tag{1}$$

where d_n is the distance between the two nodes and a is a medium-specific weight that can be used to specify different costs for transmission along different media. The parameter a can be useful if the node has multiple interfaces and different means of reaching some of its peers, so that the cheapest path can be chosen. If there is only one type of transmission media, its value becomes obviously irrelevant.

As can be seen, a quadratic increase in cost has been assumed to mimic the quadratic decrease in signal power typical of radio communications. Other transmission media might have different distance behaviors, but for the sake of simplicity, we used the same formula for every medium. The extension to use different algorithms for different media is nevertheless trivial to implement. It is worth noting that (1) is formulated so that

$$c(r) = 2\,c(r/2) \tag{2}$$

thus obeying the empirical definition of range given at the beginning.

4.2 Routing tree computation

The well-known Dijkstra algorithm can be applied to the mesh obtained as described in the previous subsection, to extract an optimal tree having its root at the master node. The tree is optimal in the sense of minimizing the total cost of transmitting a packet from or to the root.

An example is shown in Fig. 1, where a chain of ten nodes attached to a master controller is depicted. Arcs represent all the possible edges in the mesh; those in red were selected by the algorithm to belong to the tree. By visiting the tree in a depth-first pre-order fashion, nodes can be numbered (assigned network addresses) so that all the children of a particular node share a common span of addresses, ensuing in the smallest possible routing tables.

This is shown in Fig. 2, where the same tree found in Fig. 1 is redrawn to better highlight its topology and the addressing scheme of the nodes. The resulting routing tables for each node are also shown. At most one entry is used in each table, apart from the two implicit rules, one for the node own address and one for the fallback (default) router.

4.3 Linear routing

Though the Dijkstra algorithm employed as outlined before is able to generate optimal routes from the master to any other node, direct communication between the two nodes might require uselessly long tree traversals to reach a common ancestor, and in extreme cases, this might mean reaching the tree root even to send a packet to a neighboring node. To avoid this shortcoming, a couple of optimizations can be performed.

First, a specialized algorithm is used instead of Dijkstra for one-dimensional media. This ensures that the tree is well balanced, so that neighboring nodes are at most three hops away. To this end, a simple greedy strategy suffices. From the first node, the farthest reachable node is selected to act as a router, and the process repeated from this new router until the cable ends. All the remaining nodes are then assigned as children of the nearest router. An example of this procedure is depicted in Fig. 3.

Second, the routing tables can be extended with specialized entries only used for locally generated packets. These entries list all the directly reachable nodes and thus allow communication between them to occur without the hopping overhead. As usual, to keep the routing table short, special numbering of the nodes is required. By numbering the children of each router in the same order as they are connected on the cable, at most four entries in the routing table are needed, as shown in Fig. 4.

This ensures that nodes within the transmission range do not need repeaters to communicate with each other, thus providing extremely low-latency short-distance communication. Further nodes still require hopping, but the total distance traveled by the packet is the sum of the distance between the two routers involved plus the distances between each endpoint and its local router. As the distance increases, this asymptotically approaches just the distance between the nodes, resulting in asymptotically optimal behavior.

To better illustrate this, Fig. 5 depicts two examples of what happens to a packet as it travels along the network. The first case involves a transmission from NodeB (address 2) to NodeE (address 5). On the tree, these two nodes are three hops apart, but on the cable, they are close enough to be able to directly talk to each other. Indeed, the extended routing table of NodeB states that destination 5 can be reached by transmitting the packet at a depth level of one more NodeB own's depth. This packet will of course be sensed by other nodes at depth 3, e.g., surely by NodeC and possibly by NodeG. These nodes will only consult the basic, tree-derived, routing table and discover that they are not interested in the packet as both `SRC` and `DST` match the default route. As a second example, consider the transmission always from NodeB but to NodeF (address 7). This time NodeF is beyond the imposed direct communication limit, and NodeB has no special entry for it in its routing table. It thus sends the packet up on the tree (default route), i.e., at depth 1, for others to deal with it. Only NodeA is at depth 1, and `DST` matches its route #2, so the packet is sent again to depth 2. NodeB

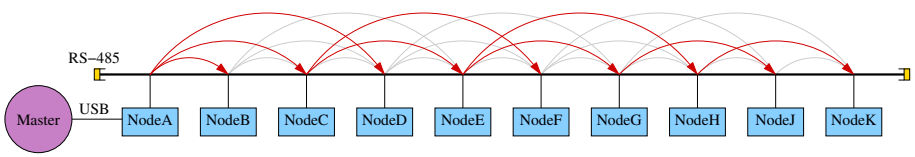
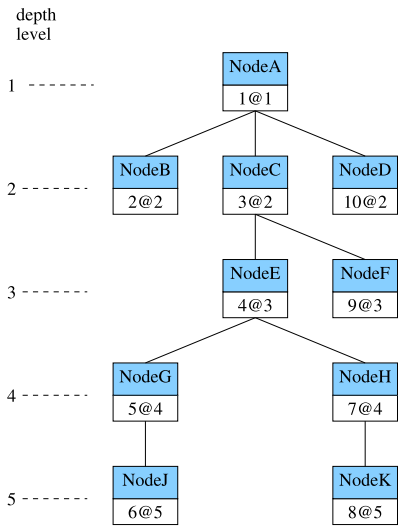


Fig. 1 Linear routing with Dijkstra algorithm. An example of an RS-485 bus connecting ten nodes uniformly spaced with an inter-node distance ℓ is presented. The transmission speed is assumed to have been chosen so as to allow reliable reception up to a distance of at least 3ℓ. Among all the possible graph edges, represented by arcs, the Dijkstra routing algorithms selected the tree highlighted in *red*. If the inter-node distance is exactly uniform, it results in many different paths having exactly the same cost, so the algorithm picks one essentially at random (the algorithm itself is deterministic, but the results will depend on the details of the implementation and on hardly predictable rounding errors)



node name	node address	route #1 span	route #1 dir	default route span	default route dir
NodeA	1	**2–10**	**+1**	**ALL**	**−1**
NodeB	2			**ALL**	**−1**
NodeC	3	**4–9**	**+1**	**ALL**	**−1**
NodeD	10			**ALL**	**−1**
NodeE	4	**5–8**	**+1**	**ALL**	**−1**
NodeF	9			**ALL**	**−1**
NodeG	5	**6–6**	**+1**	**ALL**	**−1**
NodeH	7	**8–8**	**+1**	**ALL**	**−1**
NodeJ	6			**ALL**	**−1**
NodeK	8			**ALL**	**−1**

Fig. 2 Dijkstra-generated tree and routing tables. The logical topology of the example depicted in the previous figure is shown below. Addresses are assigned to the children according to a depth-first pre-order visit of the tree (and are shown in the tag attached to the nodes in the format `address@depth`). This leads to the minimal routing tables shown on the *right*, as each node needs at most one entry, besides the implicit routes (own address and default route). There, `span` is the address range to match, `dir` is the amount to add to the current depth. Lookup stops after the first match

senses the packet again but does not act on it because both `SRC`'s `dir` and `DIR` are non-negative. NodeD, on the other hand, senses the packet and finds `DST` within the span of its route #3, so it retransmits the packet at depth 3. Again, all nodes at depth 3 but NodeG will ignore this packet, NodeG will finally retransmit it at depth 4, where NodeF receives it. This is of course the worst-case scenario, needing four transmissions to reach a node just further away than the direct communication limit, but as the distance increase, the overhead becomes negligible.

This simple routing algorithm is thus able to provide superior performance than Dijkstra's for linear media. Unfortunately, it cannot be easily extended to two-dimensional media, as there no simple and suitable ordering of the physical position that can be defined. It is nevertheless quite easy to incorporate in a larger network, sufficing to run it first for the linear portions of the network, and then feed a reduced graph containing only the selected edges to the full-blown Dijkstra router.

5 Performance evaluation

In order to assess the effectivity of the proposed improved linear routing handling, a study was performed considering an EIA RS-485 bus [10] as the communication medium.

It consists of a twisted-pair bifilar line with a 100–120 Ω characteristic impedance and employs differential signaling, typically carrying baseband non-return-to-zero asynchronous serial data and allowing half-duplex communication. Cheap transceivers allow simple interconnection with standard serial ports found in most microcontrollers, and hundreds of transceivers can be connected to the same bus.

The achievable speed depends on cable length and transceiver performance. Considering a typical transceiver with a maximum data rate of 10 Mbit/s, the data-rate cable-length product must be below 122 Mbit · m/s, and the length cannot exceed 1.22 km. A summary of other specifications is reported in Table 1.

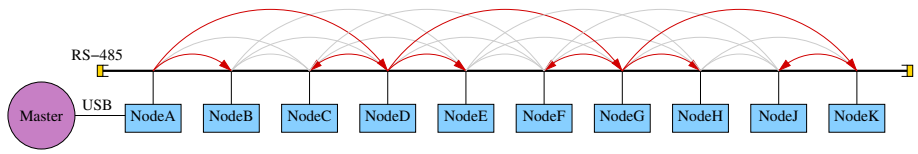
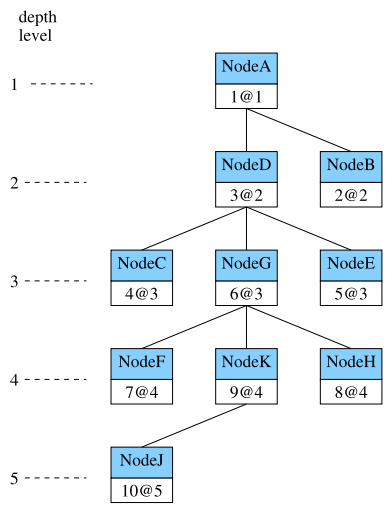


Fig. 3 Optimized linear routing example. An example of an RS-485 bus connecting ten nodes uniformly spaced with an inter-node distance ℓ is presented. The transmission speed is assumed to have been chosen so as to allow reliable reception up to a distance of at least 3ℓ. Among all the possible graph edges, represented by *arcs*, the specialized linear routing algorithms selects the tree highlighted in *red*



node name	node addr	route #1 span	route #1 dir	route #2 span	route #2 dir	route #3 span	route #3 dir	route #4 span	route #4 dir	default route span	default route dir
NodeA	1	4–4	+2	**2–10**	**+1**					**ALL**	**−1**
NodeB	2	3–3	+0	4–5	+1					**ALL**	**−1**
NodeC	4	1–1	−2	2–2	−1	5–5	+0	7–7	+1	**ALL**	**−1**
NodeD	3	2–2	+0	7–7	+2	**4–10**	**+1**			**ALL**	**−1**
NodeE	5	2–2	−1	4–6	+0	7–8	+1			**ALL**	**−1**
NodeF	7	3–3	−2	4–5	−1	8–8	+0	10–10	+1	**ALL**	**−1**
NodeG	6	5–5	+0	10–10	+2	**7–10**	**+1**			**ALL**	**−1**
NodeH	8	5–5	−1	7–9	+0	10–11	+1			**ALL**	**−1**
NodeJ	10	6–6	−2	7–8	−1	11–11	+0			**ALL**	**−1**
NodeK	9	8–8	+0	**10–10**	**+1**					**ALL**	**−1**

Fig. 4 Optimized logical tree and routing tables. The logical topology of the example depicted in the previous figure is shown. Addresses are assigned to the children ordered by physical position on the cable. This leads to the routing tables shown on the *right*, where each node needs at most four explicit entries, besides the implicit routes (own address and default route). Entries in *bold* are those that describe the tree and are the only ones considered to forward packets. The other entries were added to allow direct communication between nodes within direct visibility

When using the ToLHnet protocol on top of RS-485, the data-rate cable-length trade-off can somehow be lifted as there is no need for every node to be able to communicate with every other node. Of course, in a properly setup and terminated transmission line, signal degradation depends only on what is present between the transmitting and the receiving nodes. The total length of cable beyond the nodes involved in the communication is essentially irrelevant, but for the reflected waves from possibly inexact termination. These should anyway be kept well attenuated, as they would otherwise hamper communication even without ToLHnet.

To demonstrate this, a first experiment was set up [3], connecting four nodes to an RS-485 bus as shown in Fig. 6.

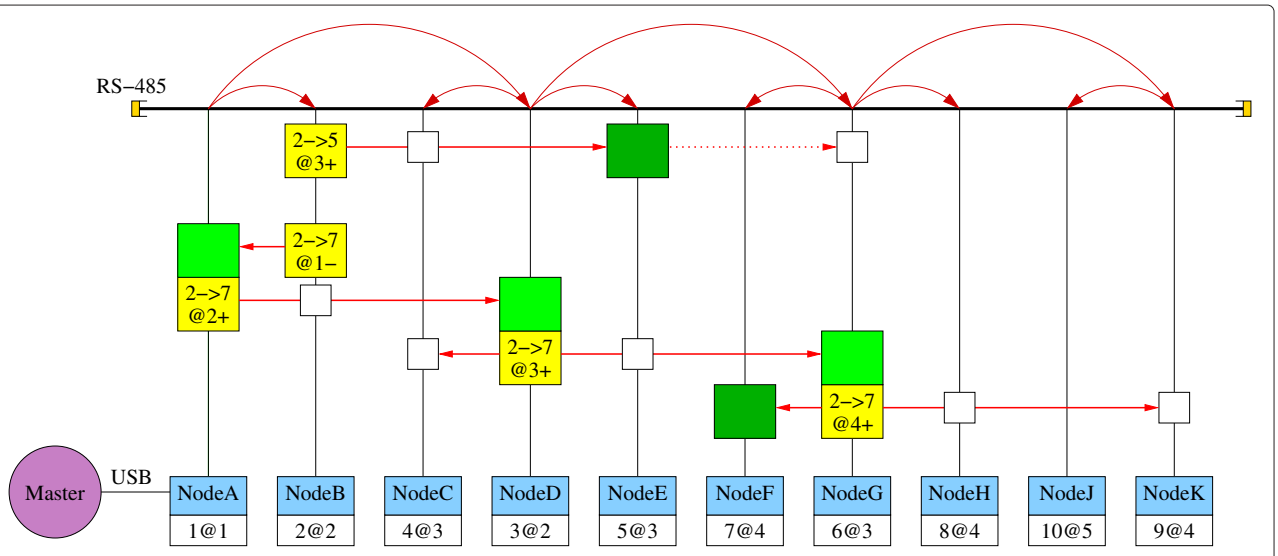


Fig. 5 Packet flow along the network. This figure depicts the transmissions that occur on the network to send a packet from NodeB to NodeE (*top example*) and from NodeB to NodeF (*bottom example*). *Yellow boxes* denote packet transmissions. The packet header fields are represented inside each such *square* in the format `SRC -> DST @ HOPS DIR`. *Green boxes* denote packet acceptance, either to be forwarded (*light green*) or to be accepted locally (*dark green*). Of course, many other nodes on the network may actually receive the packet. A few of them, marked by *smaller white boxes*, are also at the correct depth level to possibly accept the packet, but the routing rules are designed so that all but the designated forwarder discard the packet

Table 1 EIA RS-485 specifications

Parameter	Conditions	Min	Max
Driver output voltage	Open circuit	±1.5 V	±6 V
Driver output voltage	$R_L = 54\,\Omega$	±1.5 V	±6 V
Driver output short-circuit current	Output at 12 V or −7 V		±250 mA
Driver output rise time	$R_L = 54\,\Omega$, $C_L = 50$ pF		30 % of bit width
Driver common-mode voltage compliance	$R_L = 54\,\Omega$	−1	±3 V
Receiver sensitivity	$-7\text{ V} < V_{cm} < 12\text{ V}$		±200 mV
Receiver common-mode voltage range		−7 V	−12 V
Receiver input resistance		12 kΩ	

The routing tables, also shown in the same figure, were set up to make each node act as a repeater, to be able to study signal distortion at various distances from the transmitters. The baud rate was set to 1 Mbit/s, just a little above the standard limit for the span of cable between the first two nodes, but since the bus loading is minimal (no other nodes are attached between), communication is still reliable.

The signal on the bus was recorded at the terminals of Board4 while the master PC sent a ping command to it. Figure 7 shows the recording, with bursts corresponding to packets as they are transmitted along the chain of repeaters. Each of the first four bursts was analyzed for signal quality. A detail of the signal with four bit edges in each is shown in Fig. 8, with their amplitude, rise, and fall times reported in Table 2.

As can be seen, signal attenuation is not really an issue in this case, while rise and fall times degradation increases almost linearly with the distance separating the nodes, as

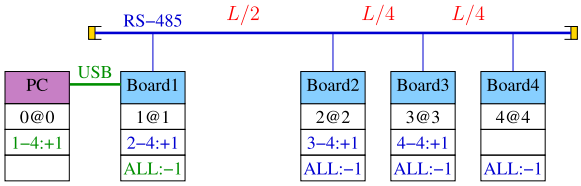


Fig. 6 Experimental setup. Four nodes were placed on an RS-485 bus with a total cable length $L \simeq 320$ m. Board1 and Board4 are at the cable ends, Board2 is at mid-cable, Board3 midway between Board2 and Board4. Board1 communicates with the master controller (PC) through an USB cable. The routing tables specifically set up for the test are attached below each node. The first tag is `address@depth`, the others are `span:dir`, while `iface` is color-coded

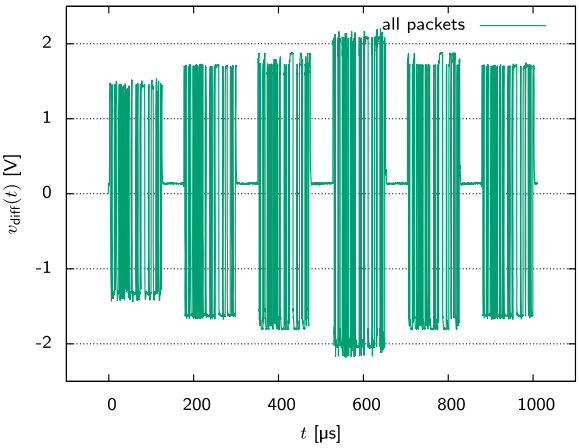


Fig. 7 Ping sequence on a chain of repeaters. The differential RS-485 signal as received by Board4 during the arrival of a ping request and the following reply. In order of arrival, the bursts are the packet sent from Board1, then repeated by Board2, and finally repeated by Board3. The strongest signal is the reply transmitted by Board4 itself, followed by the reversed chain of repeats by Board3 and Board2. Board1 finally forwards the packet to the master PC over the USB link. The total round trip time for the process, as seen by the master, is 1.4 ms

can be expected for relatively sharp edges at not very long distances. Under these extremely light-load conditions, of course, routing wouldn't really be needed, as the most distorted signal is still clearly easily recognizable despite the degradation in rise and fall times. All the nodes could simply talk directly to each other, though with a slightly out-of-spec rate.

To verify that the routing performed by ToLHnet actually helps, we pushed the baud rate close to the hardware limit, at 5 Mbit/s. At this speed, we measured the packet loss rate resulting from sending 100,000 PING requests from the master to each of the nodes, with (P_H) and without (P_1) the routing performed by ToLHnet. The results are shown in Table 3.

Of course at these speeds, more than six times above what the length of cable would have guaranteed, a few transmission errors occur even between close nodes. The packet loss rate clearly increases for longer payloads, as is normal because the packet checksum check discards the whole packet even if it detects a single bit error.

At longer distances, the benefits of routing become apparent. It would be almost impossible to send 240 bytes across the cable at 5 Mbit/s without intermediate repeaters, losing more than 96 % of the packets. With ToLHnet, instead, less than 0.1 % of the packets get lost, making communication possible and quite reliable (for real, non-PING packets, the master will of course automatically retry transmissions in case of packet loss).

Unfortunately, repeating packets by routing them, instead than by employing electrical means, increases

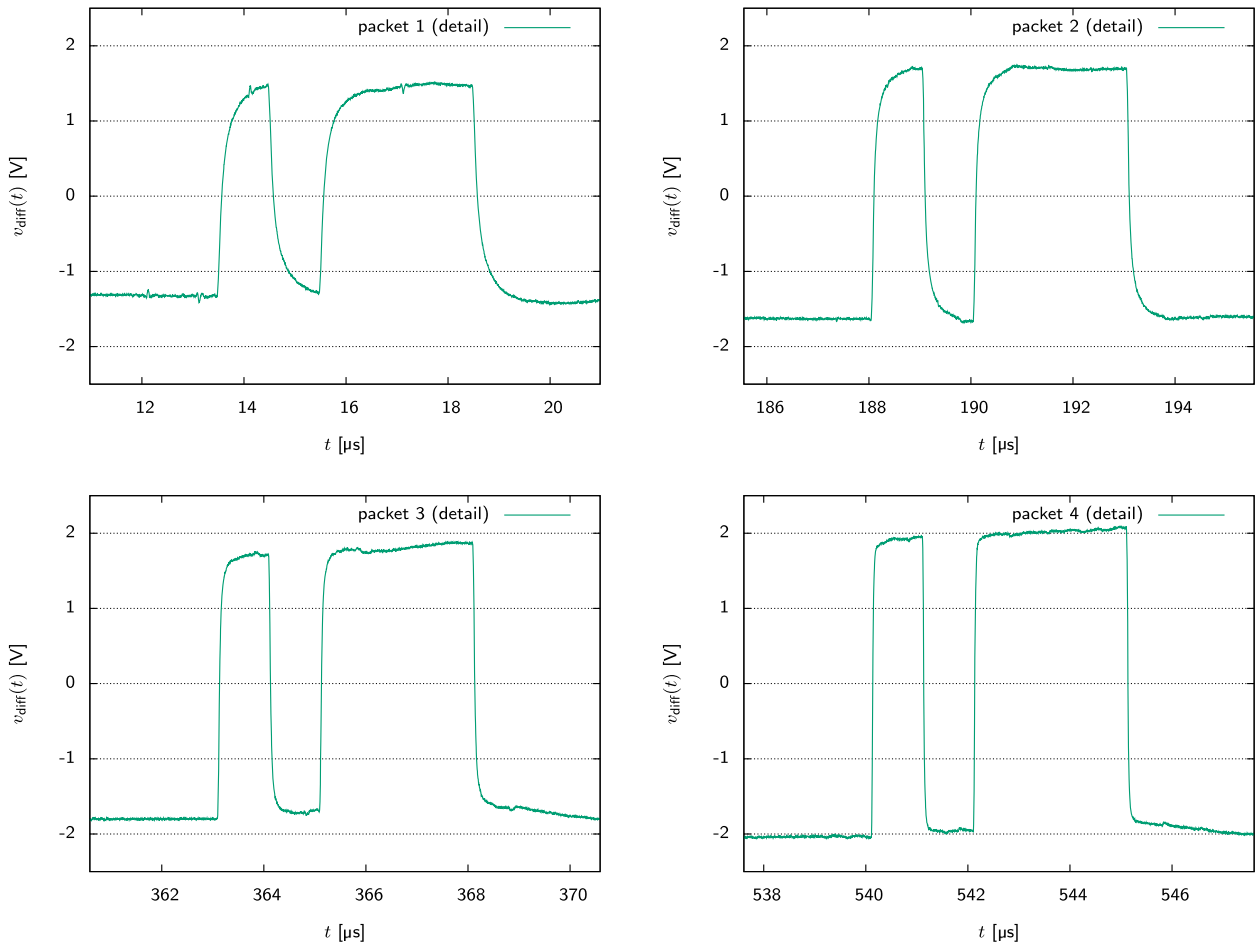


Fig. 8 Measured signal distortion. The differential RS-485 signal as received by Board4 during the three retransmissions from Board1, Board2, Board3, and the reply generated by Board4 itself, are shown

latency and decreases effective bandwidth. This is unavoidable, and to better quantify its effect, a thorough analysis of the performance attainable with the proposed routing scheme was also carried out.

To this end, we simulated networks composed of up to thousands of nodes, attached to cables of varying length and operating at different speeds. As we focus on the performance of the routing scheme, only the number of hops imposed by the routing strategy was taken into account to estimate the attainable throughput, disregarding any implementation overhead, that strongly depends on the particular hardware used. We developed a custom simulator that randomly positions nodes along the simulated cable and then applies the proposed algorithm to compute routing tables, assuming that the standard transmission range limit of 122 Mbit · m/s applies to the distance between any two nodes. Of course, all the nodes must operate at the same data rate, so the above range limits the maximum distance between closest neighbors.

Table 2 Peak-to-peak voltages and rise and fall times of packets involved in a ping command at a bit rate of 1 Mbit/s

Packet #	Voltage (V)	Rise time (ns)	Fall time (ns)
1	2.76	412.0	374.0
2	3.34	249.6	244.0
3	3.67	177.2	156.0
4	4.12	62.80	58.00

Table 3 Packet losses without the forwarding capability of ToLHnet (P_1), and with the routing tables shown in Fig. 6 (P_H), for a bit rate of 5 Mbit/s, and different payload lengths N

		$N = 128$ bytes		$N = 240$ bytes	
Destination	Distance (m)	P_1 (%)	P_H (%)	P_1 (%)	P_H (%)
Board2	160	0.00	0.00	0.01	0.01
Board3	240	0.01	0.00	18.57	0.03
Board4	320	42.37	0.02	96.73	0.03

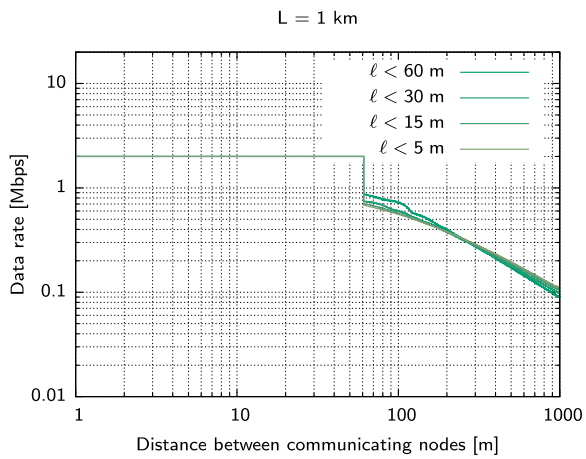


Fig. 9 Performance of the ToLHnet protocol for different node densities. This figure reports the simulated average attainable data rate of the ToLHnet protocol operating at 2 Mbit/s over a 1-km-long cable, as a function of the distance between the source and destination nodes. At this baud rate, direct communication is possible only up to about 60 m, then intermediate hops are needed, causing a drop in throughput. *Different curves* represent results obtained with different node densities (specified as maximal inter-node distance). It can be seen that the average performance is very little sensitive to node density, except possibly for communication distances just above the direct visibility limit

These routing tables are then exploited to count the number of hops required to reach any node from any other node on the cable. Since, at typical RS-485 data rates, the propagation delay is almost negligible compared to the time required to transmit the packet, we assumed the attainable throughput to be just inversely proportional to the number of hops. The throughput was then averaged among all the possible pairs of nodes in the network, considering all sources and destinations equally likely.

As a first result, we investigated the effect of node density (inverse of the average distance between nearest neighbors ℓ) on the average data rate. The results are shown in Fig. 9, drawn for a 1-km cable operated at 2 Mbit/s. As can be seen, for a distance between communicating nodes below about 60 m, no hops are necessary and the full bandwidth is available. Above 60 m, direct communication is no longer possible, and the density of nodes determines the probability of finding a suitable candidate to repeat the packet where it is needed; otherwise, a slightly sub-optimal path must be taken. Nevertheless, as the figure shows, it does not affect much the performance, which can be considered essentially independent of node density.

The other parameter that might influence performance is the total cable length. Figure 10 shows the simulated average data rate for three different cable lengths, 100 m, 1 km, and 10 km. The latter length is well above the limit for direct communication with a standard RS-485 setup. For each length, different baud rates were simulated, allowing different trade-offs between direct communication range and latency. Both the newly proposed routing scheme, with the augmented routing tables that allow direct communication between siblings and close relatives, and the original, tree-only routing scheme, were tested.

Of course, there is no difference between the two routing schemes at distances longer than the direct visibility limit, both asymptotically approach the theoretical limit.

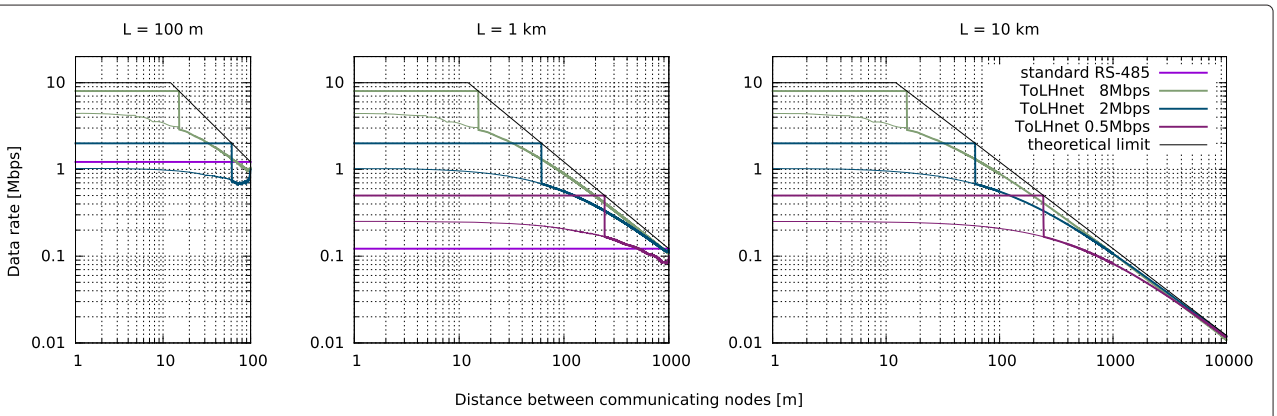


Fig. 10 Performance of the ToLHnet protocol for different cable lengths. The previous figure showed that ToLHnet performance is nearly independent of node density. Here, its sensitivity to total cable length is verified. The three plots below illustrate the average raw performance of the ToLHnet protocol operating on three different lengths of RS-485 cable, as a function of the distance between the nodes involved in the communication and for different choices of the baud rate. Inter-node distance is assumed to be uniformly distributed up to a maximum of 5 m. As a reference, both the theoretical limit and the performance of traditional RS-485 bus communication (which depends only on total cable length and not on communicating node distance and is impossible at 10 km) are shown. *Colored thin dashed lines* represent the performance of the ToLHnet protocol limited to follow tree branches, while *thick solid lines* represent the performance attainable with the proposed extension to the routing strategy that also allows direct communication between neighboring nodes

But for shorter distances, a distinctive improvement of a factor between 2 and 3 can be achieved with the new scheme. Again, the performance of the protocol can be seen to be almost unaffected by total cable length, allowing for a great deal of flexibility in the installation of the network.

6 Conclusions

In this paper, we proposed an extension to the routing algorithm employed in the ToLHnet protocol to increase its performance in case of routing over a linear medium. Experimental results conducted with RS-485 transceivers demonstrated the possibility of communicating at baud rates well above those imposed by the total cable length, when enough intermediate nodes are present to repeat the signal. This allows a very large network to be deployed almost freely, without worrying about the placement of dedicated repeater circuits, and the "soft" nature of the forwarding mechanism, employed by every node, allows the master to optimize their placement so as to reach near-optimal performance. This is demonstrated by extensive simulations on networks of varying breadth and density. The additional complexity imposed to the nodes to deal with this optimized routing is deemed negligible, as only a small increase in routing table size is required.

Competing interests
The authors declare that they have no competing interests.

Received: 1 March 2016 Accepted: 7 July 2016
Published online: 25 July 2016

References

1. G Biagetti, P Crippa, A Curzi, S Orcioni, C Turchetti, in *Proceedings of the 6th European Embedded Design in Education & Research Conference (EDERC 2014)*. ToLHnet: a low-complexity protocol for mixed wired and wireless low-rate control networks (IEEE, Milan, Italy, 2014), pp. 177–181. doi:10.1109/EDERC.2014.6924383
2. ToLHnet—tree or linear hopping network. http://www.tolhnet.org/. Accessed Jan 2016
3. G Biagetti, P Crippa, L Falaschetti, S Orcioni, N Ortolani, C Turchetti, in *Proceedings of the 2015 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES 2015)*. Improvement of RS-485 performance over long distances using the ToLHnet protocol (IEEE, Ancona, Italy, 2015), pp. 85–89
4. Y Chen, Z Liu, in *International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC '09)*. Distributed intelligent city street lamp monitoring and control system based on wireless communication chip nRF401, vol. 2 (IEEE, 2009), pp. 278–281. doi:10.1109/NSWCTC.2009.69
5. MA George, S Choudhary, D Sahay, T Yerra, CP Kurian, in *2013 Texas Instruments India Educators' Conference (TIIEC)*. Digitally addressable wireless interface for lighting control system (IEEE, 2013), pp. 222–229. doi:10.1109/TIIEC.2013.46
6. JM Corchado, J Bajo, DI Tapia, A Abraham, Using heterogeneous wireless sensor networks in a telemonitoring system for healthcare. IEEE Trans. Inf. Technol. Biomed. **14**(2), 234–240 (2010). doi:10.1109/TITB.2009.2034369
7. J Song, S Han, AK Mok, D Chen, M Lucas, M Nixon, in *IEEE Real-Time and Embedded Technology and Applications Symposium, (RTAS '08)*. WirelessHART: Applying wireless technology in real-time industrial process control (IEEE, 2008), pp. 377–386. doi:10.1109/RTAS.2008.15
8. AR Wilson, PS Vincent, Networked low-power sensing: network interface and main operating system. IEEE Sensors J. **10**(9), 1495–1507 (2010). doi:10.1109/JSEN.2010.2044879
9. MA McHenry, D Roberson, RJ Matheson, Phone to fridge: shut up! Spectrum IEEE. **52**(9), 50–56 (2015). doi:10.1109/MSPEC.2015.7226614
10. TIA-485, electrical characteristics of generators and receivers for use in balanced digital multipoint systems. Technical Report rev. A Telecommunications Industry Assn. (TIA) (1998)
11. GBM Guarese, FG Sieben, T Webber, MR Dillenburg, C Marcon, in *2012 Brazilian Symposium on Computing System Engineering (SBESC)*. Exploiting Modbus protocol in wired and wireless multilevel communication architecture (IEEE, 2012), pp. 13–18. doi:10.1109/SBESC.2012.12
12. V Tipsuwanporn, A Numsomran, S Samaimak, S Hamnarong, in *13th International Conference on Control, Automation and Systems (ICCAS 2013)*. Software developments for Modbus SCADA to extend distance to 2x for controlled devices (IEEE, 2013), pp. 605–609. doi:10.1109/ICCAS.2013.6703939