Soft Real-Time Traffic Communication in Loaded Wireless Mesh Networks

Jesus Aisa∗, Hossein Fotouhi†, Jose L. Villarroel∗, and Luis Almeida‡
∗Universidad de Zaragoza, Zaragoza, Spain
†Mälardalen University, Västerås, Sweden
‡IT / Fac. Eng., University of Porto, Porto, Portugal

Abstract—Industrial applications have been shifting towards wireless multi-hop networks in recent years due to their lower cost of deployment and reconfiguration compared with their wired counterparts. These wireless networks usually must support real-time communication to meet the application requirements. For this reason, Wireless Mesh Networks (WMNs) are potential candidates for industrial applications as they support a fixed infrastructure of static nodes for relaying packets. To meet the application demands, we modify the wireless chain network protocol (WICKPro) to support real-time traffic in WMNs with chain topologies over IEEE 802.11. We employ tele-operation of mobile robots as our case study, and perform extensive simulation and laboratory experiments. We show that the data delivery ratio is increased up to 42% in a scenario with 7 nodes, when the maximum end-to-end delay tolerated by the application is doubled. This is particularly suited to soft real-time applications that can trade longer delays by higher reliability. Moreover, when compared with a distributed priority-based token-passing protocol (RT-WMP), the lower overhead of WICKPro allows, in an error-free scenario, obtaining a throughput improvement of 33.42% on average.

Index Terms—Wireless mesh networks, chain topology, soft real-time, cyclic scheduling, token passing, IEEE 802.11

I. INTRODUCTION

The future vision of factory automation and process control involves component state monitoring in a continuous manner. The Factories of the Future 2020 roadmap [1] forecasts “the need for advanced machine interaction with humans through ubiquity of mobile devices to receive relevant production information”. For these tasks, wireless networks have emerged as key technologies due to their lower cost of deployment and reconfiguration compared with their wired counterparts [2]. In these scenarios, real-time monitoring may benefit from mobile robotics, for instance using mobile robot tele-operation [3], [4]. Various wireless networks and technologies have been employed in industrial applications. Wireless Mesh Networks (WMNs) are promising enablers for supporting real-time communication as they provide a wireless fixed infrastructure (mesh routers), which can be utilized by mobile terminals (mesh clients). These networks should provide the real-time traffic support that action-perception loops typical of industrial systems require.

In this paper, we focus on Soft Real-Time (SRT) traffic support in WMNs with chain topologies. This kind of networks are very useful in confined and underground areas such as tunnels and mines due to their shape. Specifically, we modify the Wireless Chain network Protocol (WICKPro) [5] to support SRT traffic. WICKPro is a Medium Access Control (MAC) and routing protocol designed for WMNs with chain topologies which supports Firm Real-Time (FRT) traffic. It employs a synchronous token-passing scheme and a cyclic packet scheduler, and is implemented over the IEEE 802.11 standard. As a case study, we carry out the tele-operation of a mobile robot, including simulation and laboratory experiments.

The rest of the paper is organized as follows. Section II describes related work. Section III presents WICKPro with SRT traffic support, and Section IV evaluates this approach. Finally, Section V concludes the paper.

II. RELATED WORK

In this section, we review wireless protocols that support dynamic real-time communication. On the one hand, the IEEE 802.11 protocols, including the amendment for mesh networking (IEEE 802.11s), introduce timeliness limitations by using a random-based MAC mechanism, i.e., Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). On the other hand, dynamic Time Division Multiple Access (TDMA) protocols focus mainly on efficient slot allocation, rerouting and quality of service adaptation but with queuing policies that are usually local and uncoordinated. Thus, we focus on dynamic protocols that use coordinated real-time packet schedulers (cyclic or priority-based), and whose implementation is carried out with TDMA or token-passing MAC schemes, since these mechanisms are contention-free. In Table I, we show four protocols that are analyzed based on the aforementioned features.

RT-WMP [7] is a protocol designed for mobile ad-hoc networks implemented over 802.11. It uses a token-passing approach and a global priority-based packet scheduler working in three phases. In the first phase, the token is passed through all nodes in a way that nodes fill the token with the priority of the message they want to send, so that the Most Priority Message (MPM) is figured out. In the second phase, the token is sent directly to the node with the MPM whereas in the third phase the multi-hop message is actually transmitted. It should be noted that the first and the second phases are necessary as the information about message priorities is distributed.

<table>
<thead>
<tr>
<th>Packet scheduler</th>
<th>MAC scheme</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority-based</td>
<td>Token-passing</td>
<td>RT-WMP</td>
</tr>
<tr>
<td>Cyclic</td>
<td>Token-passing</td>
<td>WICKPro</td>
</tr>
<tr>
<td>Priority-based</td>
<td>TDMA</td>
<td>[6]</td>
</tr>
<tr>
<td>Cyclic</td>
<td>TDMA</td>
<td>CYclicMAC</td>
</tr>
</tbody>
</table>
WICKPro [5] also uses a token-passing scheme, but it is specifically designed for WMNs. It employs less token transfers than RT-WMP because it implements cyclic packet scheduling. For this reason, it carries out an admission phase to accept new multi-hop flows and a tear-down phase to remove multi-hop flows. Thus, unlike RT-WMP that develops a data-gram network, WICKPro implements a virtual-circuit network. Moreover, support for hand-offs is still under development and it uses a token that is released synchronously by a token master. Thus, all nodes have their communication schedules synchronized to the token cycles.

Regarding TDMA-based protocols, we present the following two proposals. In [6], the implicit Earliest Deadline First (implicit-EDF) algorithm is used to execute a scheduler replicated in all nodes in lock step. It requires clock synchronization to implement the implicit EDF without collisions within the TDMA slots. Another interesting protocol is Cyclic-MAC [8], which carries out a cyclic link scheduling taking into account the deadlines of the supported periodic flows. It also requires clock synchronization due to the use of a TDMA scheme. Other TDMA-based protocols do not use clock synchronization, such as [9], but they are relatively slow in tracking dynamic topologies or dynamic communication requirements.

A typical advantage of TDMA protocols is the possibility to reuse the channel in space, increasing the total capacity. In token-passing protocols, to the best of our knowledge, this is still an open issue. Conversely, packet losses are easily handled by token-passing protocols that can retransmit packets when necessary. This is not possible in TDMA protocols that typically use predefined tight schedules. Some proposals such as RT-WiFi [10] employ in-slot retransmission, i.e., a slot has a duration enough to accommodate a packet transmission and a retransmission, as well as two ACK packets. If this time is scheduled in every slot, more packet deadlines will be satisfied at the expense of throughput decrease. Actually, this problem may happen anyway when TDMA slots are larger than the time required to transmit the respective packets in error-free conditions, reducing efficiency. Concerning complexity, some TDMA schemes require clock synchronization, whereas token-passing protocols use explicit control. However, the latter allow implementing global priorities, despite requiring many token transfers, e.g., RT-WMP. TDMA protocols support local priorities, only. A comparison between WICKPro and TDMA protocols for WMNs with chain topologies was carried out in [11] in an error-free scenario, thus a comparison in an error-prone network is left for future work.

Therefore, we conclude that WICKPro presents a good trade-off between complexity and efficiency in small-scale networks in which spatial reuse is impossible or limited. This trade-off is further improved with the relaxed synchronization we propose in this paper for cases of SRT traffic, only.

III. THE WIRELESS CHAIN NETWORK PROTOCOL

The WICKPro protocol is a MAC and routing protocol designed for WMNs with chain topologies. It was presented as a Hard Real-Time (HRT) protocol by considering an error-free network [11], [12], and as an FRT protocol (FRT WICKPro) in the presence of packet losses [5], [11]. In this paper, we present the SRT version of WICKPro (SRT WICKPro) which supports SRT traffic, only. This is enough for many multimedia-based robotic applications such as surveillance and tele-control, possibly complemented with a low-level local safety subsystem.

A. Network Description

We consider a WMN comprised of routers and clients. Routers form a chain with high quality links with their one-hop neighbor routers, and clients are connected exclusively to one router. Thus, there is only one possible route between any two nodes. For the sake of simplicity of the analysis that follows, we made the following assumptions:

1) the topology is fixed and known to all nodes;
2) the first router in the chain is the token master. This node synchronizes the whole traffic scheduling and its updates when nodes join or leave or when communication requirements change. Here we consider fixed schedules known by all nodes. In case of failure, this node should be dynamically replaced;
3) there is one radio channel, which is shared by all nodes;
4) spatial reuse is not applied.

B. Traffic model

We consider time-triggered communication because it is well suited for control applications that require periodic transmissions [13], such as motion control. As these applications require SRT traffic support, multi-hop data flows can be characterized by source node of flow (src), destination node of flow (dst), transmission time (C), period between packets (T), deadline (D), and a cost function \( v \) that weights the value of a delivered packet depending on its delay. Moreover, we assume \( D=T \) and we include in \( C \) any transmission-related overheads.

Regarding cost functions for SRT applications, Fig. 1 shows a generic cost function [14] and the cost function used by WICKPro. These functions consider that packets whose delays are higher than their deadlines still provide some value to the application. For this reason, we define \( EED_{max} \) as the maximum End-to-End Delay which makes a packet be useful for the application. The objective should be maximizing the cumulative value of every multi-hop flow, given by the sum of the value of all the delivered packets.

The cost function in Fig. 1a could be used by a voice application which transmits packets across a communication network. In this kind of applications, it is hard to follow a conversation when the time between speaking in one side and listening on the other side differs more than 300 ms [15]. Thus, packets with delays higher than 300 ms can be dropped (\( EED_{max} = 300\text{ms} \)). Moreover, the lower the end-to-end delay of voice packets, the more likely the application users would perceive an improvement, so that these packets should be more valuable. The same behaviour is found in control applications, e.g., robot tele-operation, where there is usually an operator who sends control commands to a mobile robot that transmits feedback information to the operator, such as
laser scan or video information. If the delay in the feedback loop is higher than a specific value, the system is unstable and, consequently, these packets are useless. In our case study, we consider that improvements in the system performance for any delay lower than $EED_{\text{max}}^\text{max}$ may be neglected, as shown in Fig. 1b. With this assumption, we can maximize the cumulative value of every multi-hop flow by maximizing its percentage of delivered packets from source to destination within the $EED_{\text{max}}^\text{max}$ deadline, i.e., its Data Delivery Ratio (DDR). This strategy was also used during a robot tele-operation in [3].

C. Packet Scheduling and Admission Control

As the flows in the network are periodic, there will be a global period (so-called the hyper-period or major cycle) in the overall traffic pattern, thus WICKPro tries to find a scheduling for this major cycle. As common in cyclic scheduling, the major cycle is divided into various minor cycles of the same duration in a way that the major cycle is an integer multiple of the number of minor cycles. The cyclic schedule is calculated taking into account that, in each minor cycle, a token rotation is carried out which starts and ends in the token master.

WICKPro implements a token-passing protocol over 802.11 to accommodate a cyclic packet scheduling. As commented in Section II, before a new multi-hop flow is admitted, an admission phase is carried out in which all nodes must accept the new flow. If all nodes find a new feasible schedule that includes the new flow, then it is accepted, otherwise it is rejected. After all nodes accept the new flow, its transmission can start. A similar process is accomplished when a flow is removed, in the so-called tear-down phase. In case of simultaneous requests, only one of them can be managed at a time, because the same information must be held in all nodes.

D. FRT vs SRT WICKPro

Let us illustrate the difference between FRT and SRT WICKPro with the example given in Fig. 2 (Scenario 1), which shows a WMN with chain topology composed of five routers (R1-R5) and two clients (C6-C7). This network is supporting two multi-hop flows with the same C and T: flow number 1 goes from C6 to C7 and flow 2 travels from C7 to C6. As the two multi-hop flows have the same period, the major cycle has only one minor cycle, whose value is the period (and the deadline) of the multi-hop flows. The transmissions carried out are shown in Fig. 3 for FRT WICKPro and in Fig. 4 for SRT WICKPro. We should note that in these figures: (i) the direction of packet transmission is indicated by an arrow which starts in the packet itself; (ii) data and token packets are piggybacked when possible; (iii) implicit acknowledgement (ACK) is used; and (iv) R1 is the token master.

We assume that in the first minor cycle of Fig. 3 (from 0 to 32.85 ms), there is no packet loss, so that all the scheduled transmissions are carried out, after which the token master keeps the token until the next minor cycle begins. In the second minor cycle (from 32.85 to 2x32.85 ms), there are two packet losses which causes that all the scheduled transmissions cannot be performed before the second minor cycle finishes. Specifically, when the packet of flow 1 transmitted by R4 is lost, this is retransmitted after a time-out. However, R3 transmits a packet of flow 2 which is not received properly.

\[\text{EED} = \text{max} \]

1When using implicit ACK, a transmitter confirms receptions by forwarding transmissions, unlike with explicit ACK where specific ACK packets are sent. but this is not retransmitted because, when the second minor cycle finishes, all nodes invoke an on-line packet scheduler for dropping all packets which have not been delivered. In this way, the second minor cycle is not overran and the token master starts the third minor cycle at the appropriate time. This feature implies a packet synchronization policy.

In SRT WICKPro, the same cyclic scheduling is calculated than in FRT WICKPro, but the token master keeps the token circulating continuously and in common token-passing protocols (asynchronous token-passing). For this reason, unlike in FRT WICKPro, the actual minor cycle duration differs from the theoretical minor cycle value. Likewise, due to the lack of synchronization, the generation of the different data flows do not occur at the same time, as shown in Fig. 4. In this example, we also assume that there is no packet loss from 0 to 32.85 ms, thus all the scheduled transmissions in the first minor cycle are carried out. After all these transmissions, the first minor cycle finishes and the token starts the second minor cycle, however, only the token is transmitted since any data packet is ready to be sent. Therefore, the first and the second minor cycle are shorter than the theoretic value. In the third minor cycle, the are two packet losses so that SRT WICKPro carries out two retransmissions and this minor cycle becomes larger than the theoretic minor cycle. In general, SRT WICKPro transmits data packets until their delay is higher than $EED_{\text{max}}^\text{max}$. At that point, the data packet is dropped and only the token packet is transmitted. If the number of retransmissions is high, all nodes execute the topology discovery algorithm to restart the network. This decision is driven by a maximum number of retransmissions in the node which holds the token, and by a maximum time without receiving a token packet in the other nodes.

In summary, in FRT WICKPro, the token master releases the token synchronously with the periodic minor cycles (asynchronous token-passing) and thus every minor cycle is independent of each other. Conversely, SRT WICKPro carries out asynchronous token-passing, thereby decoupling the theoretic and the actual minor cycles.

E. Details on SRT WICKPro

As synchronization is removed from WICKPro, data flows will suffer release jitter, because the traffic is periodic and the protocol is asynchronous. This may lead to poor performance in terms of DDR, so that SRT WICKPro provides response times that can grow beyond D, specifically until $EED_{\text{max}}^\text{max}$. Because of this limitation, this new WICKPro version is suited for SRT traffic, only. The main motivation for this change is that mobility management becomes simpler, which is our short-term objective. Moreover, aperiodic traffic could be efficiently supported due to the asynchronous feature of the protocol.

Regarding network congestion, in error-prone networks, the higher the $EED_{\text{max}}^\text{max}$, the higher the required throughput for the same supported traffic. The reason is that data packets will be retransmitted as many times as necessary until their delay is higher than $EED_{\text{max}}^\text{max}$. This affects all network nodes. Moreover, it is possible that source nodes have more than one

Fig. 2. Scenario 1: A WMN with chain topology with 5 routers and 2 clients.
time among all packets of flow is satisfied:

the DDR of the flow

reason, we define the following test. Given a set of data flows,

the asynchronous nature of the protocol, it is possible that

is that the network utilization is no larger than 100%.

To the cyclic scheduling can be found, where a necessary condition

scenario, a set of data flows will be accepted if a feasible

is higher than

Fig. 4. An example of SRT WICKPro in Scenario 1 with two packet losses

packet to transmit of the same data flow, given that $EED_{max}$
is higher than $T$. This is not the case for relay nodes due to
the token-passing scheme used by WICKPro.

Next we provide a formal analysis of SRT WICKPro
latencies, initially in the absence of errors and then accounting
for these phenomena reserving sufficient spare time in the
schedule.

1) Schedulability analysis in error-free scenario: In this
scenario, a set of data flows will be accepted if a feasible
cyclical scheduling can be found, where a necessary condition
is that the network utilization is no larger than 100%.

2) DDR test in error-free scenario:

Although SRT WICKPro considers $D=T$ for the schedulability analysis, due to
the asynchronous nature of the protocol, it is possible that
DDR cannot reach 100% even in error-free networks. For this
reason, we define the following test. Given a set of data flows,
the DDR of the flow $i$ will be 100% if the following condition
is satisfied:

$$R_i = J_i + B_i + \sum_{j=1}^{N_{HOPS}} C_{ij} \leq EED_{max}^{i}$$ (1)

$R_i$ is the response time of flow $i$, i.e., the maximum response
time among all packets of flow $i$. This, in turn, is the difference
between the packet arrival time at destination node and the
packet generation time at source node. $R_i$ can be expressed as
the sum of the release jitter of flow $i$ ($J_i$), the blocking time
($B_i$), and the end-to-end transmission time ($\sum_{j=1}^{N_{HOPS}} C_{ij}$).

$J_i$ is the deviation from exact periodic release caused by the
cycles duration variations. $B_i$ is the delay in relay nodes, i.e.,
the time that the wireless medium is used by other flows
different than flow $i$ during the end-to-end transmission, which
may include flows scheduled in the same minor cycle. Finally,
$N_{HOPS}$ is the hop count from $src$ to $dst$ of flow $i$, and $C_{ij}$
is the transmission time of flow $i$ in the hop $j$, which is equal
to $C_i$ because all nodes are equal.

3) DDR$^{max}$: It defines the maximum DDR which may be
achieved in an error-prone network modeled by a memoryless
packet loss model. This upper bound takes into account that
no packet is dropped, i.e., $EED^{max} = infinite$. DDR$^{max}$
can be expressed as follows:

$$DDR^{max} = \sum_{i=1}^{N_{FLOWS}} ETT_i / M, DDR^{max} \in [0, 100] \%$$ (2)

$M$ is the major cycle duration and $ETT_i$ [16] is the
Expected Transmission Time of flow $i$. In turn, we define
$ETT_i$ as shown in (3):

$$ETT_i = \sum_{j=1}^{N_{HOPS}} (C_i + (ETX_{i,j} - 1)(timeout + C_i))$$ (3)

$ETX_{i,j}$ [16] is the Expected Transmission Count of flow $i$
in the link formed by the relay number $j$ (node $x$) and its
destination node (node $y$). $ETT_i$ is computed taking into account
that the first transmission lasts $C_i$ and that the subsequent
transmissions consume a time equal to $timeout$ plus $C_i$. As
WICKPro uses implicit ACK, $ETX_{i,j} = 1/PDR_{x,y}$, where
$PDR_{x,y}$ is the Packet Delivery Ratio of the link composed
by a transmitter node $x$ and a receiver node $y$.

IV. EVALUATION. SRT WICKPro

We assessed SRT WICKPro in simulation and laboratory
experiments. In the laboratory tests, we set the transmission
frequency to 5.3 GHz (802.11a) at 6 Mbps using CSMA/CA
without RTS/CTS. All transmissions were broadcast, so that
the hardware-level ACKs of 802.11 were not employed. In
the simulation evaluations, we used this configuration to
calculate the transmission time. All simulation experiments
were executed in 100,000 theoretic major cycles while each
laboratory experiment was conducted in 1000 seconds.

In Scenarios 1 and 2 (Sections IV-A and IV-B), the network
load and $EED^{max}$ were changed to analyze the effect in the
DDR. Moreover, a simple congestion avoidance algorithm was
implemented at source nodes to examine its importance. In
Scenario 3 (Section IV-C), WICKPro was compared with RT-
WMP in an error-free scenario.

A. Scenario 1 - 7-node WMN with Chain Topology

We performed extensive simulations in MATLAB to evalu-
ate Scenario 1. For this purpose, we implemented WICKPro
as a discrete-event system which calculates the cyclic packet
scheduling and manages the token passing as WICKPro does.
Packet losses can occur but token duplication is not taken
into account. In real experiments, as implicit ACK is used, a
token duplication may happen as follows. Node $k+1$ correctly
receives a token packet sent by node $k$ and sends another token
to node $k+2$, but node $k$ does not monitor it correctly. In this

$^2$PDR is a link quality metric measured in each link, while DDR is a quality
of service metric of the application which is computed for every data flow.
is higher in Table IV than in Table III when the network utilization is very high, as measured in [5].

Scenario 1 is described in Section III-D (Fig. 2, 3 and 4). The flow characteristics are shown in Table II, where C corresponds to a piggy-back packet transmission. Moreover, the token transmission time was 0.75 ms and the time-out was 4.18 ms. More details about the token transmission time calculation can be found in [5]. As both flows had the same period, the major cycle had only one minor cycle whose value was the theoretical minor cycle duration of the multi-hop flows. Note that, at this point, we are describing the calculated scheduling based on the flow features, so that this minor cycle duration is the theoretical value. In this experiment, we carried out four tests where the period T and the minor cycle duration were modified. In the first test, the minor cycle duration was the shortest possible, i.e., 26.58 ms, which was the time to transmit six packets of flow 1, six packets of flow 2 and two token packets. In the other three tests, the minor cycle duration was the minimum minor cycle duration plus the time to retransmit one, two and three piggy-back packets, respectively, i.e., 26.58 + 4.18 + 2.09, 39.12 and 45.39 ms. The case of 1 retransmission is shown in Fig. 3 and 4. Thus, the network utilization of the four tests was 100, 80.91, 67.94 and 58.56%, respectively. Moreover, the simulations were conducted using a memoryless packet loss model with a PDR equal to 100% (error-free scenario), 99%, 97% and 95%. Without loss of generality, this PDR is assumed to be the same for all links and packet sizes.

Tables III and IV show the DDR achieved when PDR equals 97% and 1.3%. We simulated six types of experiments for SRT WICK-Pro where $EED_{max}$ was set to be the period T multiplied by 1, 1.1, 1.3, 1.5, 2 and infinite. Moreover, we considered two cases regarding how the node network congestion. In the first case, source nodes use a Congestion Control (CC) algorithm in a way that these nodes can only have one packet to be sent in their transmission buffer (Table IV). This means that source nodes drop packets of the flows they generate if their delay is higher than T. In the second case, source nodes neglect using any CC algorithm, and thus can have all the packets whose delay is not higher than $EED_{max}$ in its transmission buffer (Table III). Next we discuss results from Tables III and IV.

1) Congestion control: We can observe that the DDR is higher in Table IV than in Table III when the network utilization is 100%. If there is one or more retransmissions allocated, both cases are similar.

2) $EED_{max} = T$ vs $EED_{max} = 2T$: In Tables III and IV, it is shown that the DDR is increased when $EED_{max}$ is augmented. When no retransmission is allocated, the overall DDR is increased by 42% when $EED_{max}$ is changed from T to 2T (Table IV). Moreover, unlike the case with $EED_{max} = 2T$ that satisfies network fairness, the case with $EED_{max} = T$ fails in fairness. The main reason is that the scheduling is not symmetric (as shown in the first minor cycle in Fig. 3) and the case $EED_{max} = 2T$ adds some degree of freedom which improves fairness, given that both flows have the same features (C, T, D, and v). Actually, the case $EED_{max} = I.T$ already achieves fairness. Regarding the case of three retransmissions, the cases $EED_{max} = T$ and $EED_{max} = 2T$ obtain similar results (99.20% vs 100%). We also have to take into account that the higher the $EED_{max}$, the higher the average delay of the delivered packets.

3) $EED_{max} = 2T$ vs $EED_{max} = infinite$: The DDR achieved in these cases is very similar. It is sizeable in Table III only when no retransmission is allocated. This comparison highlights the DDR marginal increase provided by $EED_{max}$ higher than 2T.

4) DDR test in error-free scenario: TheDDR test in Eq. (1) shows that DDR will always be 100% in an error-free scenario. When no retransmission is allocated, as there is no packet loss, the theorectic cyclic scheduling will be strictly carried out every minor cycle. Therefore, DDR will always be 100%.

5) DDRmax: We verify this parameter when PDR = 97%. When no retransmission is allocated, the DDRmax calculated using Eq. (2) is 90.99%. In this case, the major cycle M is 26.58 ms and $\sum_{n=1}^{N_{flows}} ETT_n$ is 29.21 ms. This value is very similar to the DDR given by Table III when $EED_{max}$ is infinite, i.e., 91.06%. When one retransmission is allocated,

<table>
<thead>
<tr>
<th>Flow number</th>
<th>C (ms)</th>
<th>T (ms)</th>
<th>$EED_{max}$</th>
<th>src dst</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.09</td>
<td>Configurable</td>
<td>Configurable</td>
<td>C6 C7</td>
</tr>
<tr>
<td>2</td>
<td>2.09</td>
<td>Configurable</td>
<td>Configurable</td>
<td>C7 C6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flow number</th>
<th>C (ms)</th>
<th>T (ms)</th>
<th>$EED_{max}$</th>
<th>src dst</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.09</td>
<td>Configurable</td>
<td>Configurable</td>
<td>C6 C7</td>
</tr>
<tr>
<td>2</td>
<td>2.09</td>
<td>Configurable</td>
<td>Configurable</td>
<td>C7 C6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DDR values in Scenario 1 using SRT WICKPro without CC</th>
<th>DDR values in Scenario 1 using SRT WICKPro with CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>EED_{max} Flow</td>
<td>0 Retx</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>95.30</td>
</tr>
<tr>
<td>2</td>
<td>33.19</td>
</tr>
<tr>
<td>1.1T</td>
<td>97.03</td>
</tr>
<tr>
<td>1.3T</td>
<td>97.42</td>
</tr>
<tr>
<td>1.5T</td>
<td>97.67</td>
</tr>
<tr>
<td>2T</td>
<td>66.74</td>
</tr>
<tr>
<td>Infinite</td>
<td>91.06</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DDR values in Scenario 1 using SRT WICKPro with CC</th>
<th>DDR values in Scenario 1 using SRT WICKPro with CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>EED_{max} Flow</td>
<td>0 Retx</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>95.32</td>
</tr>
<tr>
<td>2</td>
<td>32.89</td>
</tr>
<tr>
<td>1.1T</td>
<td>97.30</td>
</tr>
<tr>
<td>1.3T</td>
<td>97.69</td>
</tr>
<tr>
<td>1.5T</td>
<td>88.24</td>
</tr>
<tr>
<td>2T</td>
<td>91.04</td>
</tr>
<tr>
<td>Infinite</td>
<td>90.96</td>
</tr>
</tbody>
</table>

It is increased from 64.10% ((95.32 + 32.89) / 2) to 91.04% ((91.04 + 91.03) / 2).
Eq. (2) yields a $\text{DDR}^\text{max}$ of 100%, as $M$ is 32.85 ms and $\sum_{i=1}^{N_{\text{FLOWS}}} EET_i$ is 29.21 ms. When two and three retransmissions are allocated, $\text{DDR}^\text{max}$ is also 100%. The same results are shown in Table III when $\text{EED}^\text{max}$ is infinite.

**B. Scenario 2 - Robot Tele-operation**

In this scenario, we carried out a laboratory experiment where a Pioneer P3AT robot was tele-operated. We developed WICKPro in C and tested it using Linux and the robot operating system (ROS) framework [17]. For the ROS integration, we took advantage of the ROS version of RT-WMP [18]. The network topology was composed of three routers (R1-R3) and one client (C4) as illustrated in Fig. 6. C4 was the mobile robot and R1 was the base station where the tele-operator was located. We configured the base station as a router because it was not mobile. Moreover, there were two multi-hop flows: (i) the tele-operation commands from R1 to C4 (flow 1), and (ii) the laser scan data from C4 to R1 (flow 2). The tele-operation commands were generated by a PlayStation3 joystick [19] and the laser scan data by a SICK LMS2xx laser [20].

We measured the generation period of both flows as can be seen in Fig. 7. On the one hand, the laser data period was never lower than 101 ms and its mean was about 106 ms so that we set the period to 100 ms. On the other hand, the tele-operation command period was more complex to characterize because its generation period decreases if there is not any joystick movement. It is actually a sporadic flow with a mean of 17.15 ms and a median of 11.24 ms. We chose a minimum inter-arrival time of 10 ms because it has less computational cost to calculate the cyclic scheduling if all flows have harmonic periods. It should be noted that only 0.01% of the packets arrived with a period lower than 10 ms. The complete flow features are given by Table V. Moreover, the token transmission time was 0.35 ms and the time-out was 10 ms. Comparing these values with Scenario 1, we observe that the token has a lower transmission time, as less log information is sent with the token packet, and the time-out is higher, while in Scenario 2 we used a non real-time operating system (Linux), but considered the MaRTE real-time operating system [21] in Scenario 1 (similarly to [5]).

The theoretic cyclic scheduling calculated for this scenario is illustrated in Fig. 8. The major and the minor cycles lasted 100 ms and 10 ms, respectively, and consequently there were 10 minor cycles. The network utilization of the minor cycle 1 (from 0 to 10 ms) was 65.40%, whereas the network utilization of the minor cycles ranging from 2 to 10 was 31.50%.

1) **MATLAB Simulation and Real Implementation using Linux**: Before testing the robot tele-operation using ROS, we carried out MATLAB simulations and laboratory experiments, where we used Linux and fake data flows generated by external threads with the features given by Table V. The DDR obtained is shown in Table VI for both types of experiments in the case of a memoryless packet loss model with PDR equal to 97% and 95% (when PDR was 100%, the DDR was always 100%). In the case of real experiments, deliberate errors were introduced by software to achieve these link qualities. In MATLAB, we simulated both SRT WICKPro with and without congestion control (MATCC and MAT in Table VI, respectively). It can be seen that, in Scenario 2, SRT WICKPro with congestion control attains lower DDR because the network utilization is about 35% in average and, as shown in Scenario 1, this strategy is better when network utilization...
is high, as network congestion is higher. Thus, a congestion control algorithm may increase the network performance but it must be smart to adapt to every particular situation.

When comparing SRT WICKPro without congestion control in MATLAB and Linux, results are very similar, however there is a slight difference due to the following reasons: (i) token duplication is ignored in the simulation, (ii) the transmission time varies in the real experiments due to the back-off mechanism in IEEE 802.11, (iii) using non-real-time operating system in the real experiments, which causes some timing imprecision, and (iv) the lack of global clock information for computing data packet delay in the real experiments. This latter issue concerns how to calculate the accumulated delay of a data packet which has passed through several nodes if there is no synchronization in the network. For this purpose, WICKPro has a field in the data packets which indicates the accumulated delay. This field is filled at every node by adding the process time as well as the theoretical transmission time, but this is not constant, and the delay computation is therefore an approximation. This situation is different from MATLAB simulation where all times are deterministic.

We also simulated this scenario for the case $EED^{max} = 2T$ using a bursty packet loss model. To this end, we used the Gilbert-Elliott Model, specifically the simple Gilbert Model which takes into account a two-state Markov chain: in the good state all packets are properly received whereas in the bad state all packets are lost. The transition probability from good to bad state is $p$ and the transition probability from bad to good state is $q$. In this way, the mean Packet Loss Rate (PLR) is $p/(p + q)$ and the Average Burst Length (ABL) is $1/q$. Table VII illustrates the DDR when simulating four packet loss models with PDR equals 97% ($PDR = 100\% - PLR$): one memoryless and three bursty models with ABL equal to 1.1T, 2 and 4. Surprisingly, the four cases exhibit a similar behavior. The explanation lies in the fact that the channel estimation considers a saturated channel (continuous transmission in every link), but this is not the case in practice. For this reason, the memory effect disappears, as is thoroughly analyzed in [22]. For a deeper understanding, we can observe that the maximum ABL is 4 but the time-out (10 ms) is higher than four times the maximum packet transmission time (1.48 ms), and the burst effect is therefore totally lost in our simulations.

In this scenario, when no retransmission is allocated, the DDR test in Eq. (1) does not always return 100% for flow 1 in an error-free scenario. Specifically, $R_1$ is 7.59 ms and $EED^{max}_1$ is 6.54, 7.19 and 8.50 ms for the cases $EED^{max}_1$ equal to $T_1$, 1.1$T_1$ and 1.3$T_1$, respectively. Thus, DDR for flow 1 will not be 100% when $EED^{max}_1 = T_1$ or $EED^{max}_1 = 1.1T_1$. This matches the simulation results, where we obtained a DDR equal to 98.99% and 99.33%, respectively. For $EED^{max}_1$ higher than 1.1$T_1$, as well as for flow 2, simulation and Eq. (1) yield a DDR of 100%.

2) Real implementation using ROS: Finally, we conducted the real tele-operation using real hardware and ROS. The achieved DDR can be seen in Table VIII when considering a memoryless packet loss model (deliberate errors were introduced by software). The results are similar to those of simulation and real implementation using fake data, although there is some difference. However, note that when PDR equals 100%, unlike simulation and real implementation using fake data, the DDR of flow 1 is not always 100%. These results are explained by the extra release jitter produced by two issues: (i) the real periods of flow 1 and 2 last about 11 and 106 ms, respectively, while the scheduler considers 10 and 100 ms, thereby defining a minor cycle of 10 ms; and (ii) flow 1 is actually sporadic traffic.

These results become useful for the design of a robot tele-operation application. For instance, in our network, choosing $EED^{max} = 2T$ could match the application requirements and the protocol would provide a good DDR, but this depends on the link quality. The application could also decimate the tele-operation packets at source node, e.g., taking one packet every five [4]. However, we did not use this option to obtain a more loaded network.

### C. Scenario 3. Comparison with RT-WMP

In this section, we compare WICKPro with a version of RT-WMP enhanced for transmitting several packets in a single token holding [4]. RT-WMP was simulated in an error-free scenario considering a chain network which supported a bidirectional communication between the ends of the chain. Simulation was carried out for various values of network nodes and packet sizes, in a way that the network utilization was always 100%. The Maximum Transmission Unit (MTU) was 1388 bytes, so that packets with greater sizes than MTU required actually more than one packet transmission. Regarding the physical level, this simulation considered IEEE 802.11a at 6 Mbps.
We calculated the throughput achieved by WICKPro in the same conditions. As the scenario is error-free and the network utilization is 100%, FRT and SRT WICKPro provide ideally the same performance. In this case, we can use theoretical formulas to obtain the throughput in each situation, since the off-line cyclic scheduling calculated by the scheduler is repeated every minor cycle. With these results, WICKPro achieves higher throughput than RT-WMP, as shown in Table IX, where the number of nodes ranges from 2 to 32, and the packet size from 256 bytes to 256 kilobytes. Depending on the case, the improvement can be small or grow up to 271%, while the average improvement is 33.42%. The throughput improvement is achieved because WICKPro needs less token transfers to support these data flows. RT-WMP, being distributed and without global synchronization, must carry out a token rotation before every multi-hop packet transmission to know the highest priority packet, while WICKPro schedules its transmissions in advance thanks to its cyclic scheduler. It should also be noted that RT-WMP takes advantage of this token rotation to share the RSSI between all network links, which is employed to manage mobility. However, WICKPro could handle mobility without exchanging so much information, given that it considers a network with fixed routers, unlike RT-WMP which takes into account that all nodes are mobile. Thus, this comparison is still realistic.

V. CONCLUSIONS

This paper presented a periodic traffic management approach based on token-passing protocol, specifically the WICKPro protocol. In particular, we adapt this protocol to support applications that find some value in packets with delays higher than their deadline, considered equal to the packets minimum inter-arrival time ($T$), and therefore the response time of the protocol can grow until $EED^{D_{max}}$, i.e., a packets delivery window, and the throughput may be significantly increased. We tested two different scenarios in simulation and laboratory experiments. The main conclusion is that the higher the network utilization and the packet losses, the higher the $EED^{D_{max}}$ should be achieved to the same throughput. However, if $EED^{D_{max}}$ is increased, the average packet delay will also be incremented. In a given scenario, the DDR increased by 42% when $EED^{D_{max}}$ was changed from $T$ to $2T$, considering SRT WICKPro with congestion control in a saturated network. Moreover, we showed that congestion control may increase network performance in our protocol, too.

As a case study, we presented a robot tele-operation in a WMN with chain topology. The complete process from simulation to real experiment was detailed. We also carried out a comparison with a priority-based token-passing protocol (RT-WMP) in an error-free network, where the throughput gain was 33.42% on average. We are currently designing a hand-off algorithm to allow robots move across routers.

ACKNOWLEDGMENTS

This work has been partially supported by the project DGA-FSE (T04) and by the Swedish Knowledge Foundation (KK9) through the Research Environment for Advancing Low Latency Internet (READY).

REFERENCES


