# D-SAR: A Distributed Scheduling Algorithm for Real-time, Closed-Loop Control in Industrial Wireless Sensor and Actuator Networks

Pouria Zand[1] Supriyo Chatterjea[1] Jeroen Ketema[2] Paul Havinga[1]
Pervasive Systems Group[1], Formal Methods and Tools Group[2],
Faculty of EEMCS, University of Twente, P.O. Box 217, 7500AE, Enschede, The Netherlands
{p.zand, s.chatterjea, j.ketema, p.j.m.havinga}@utwente.nl

## ABSTRACT

*Current wireless standards and protocols for industrial applications such as WirelessHART and ISA100.11a typically use centralized network management techniques for communication scheduling and route establishment. However, large-scale centralized systems can have several drawbacks. They have difficulty in coping with disturbances or changes within the network in real-time. Large-scale centralized systems can also have highly variable latencies thus making them unsuitable for closed-loop control applications. To address these problems, this paper describes D-SAR, a distributed resource reservation algorithm which would allow source nodes to meet the Quality-of-Service (QoS) requirements of the application in real-time, when carrying out peer-to-peer communication. The presented solution uses concepts derived from relevant networking-related domains such as circuit switching and Asynchronous Transfer Mode (ATM) networks and applies them to wireless sensor and actuator networks.*

## 1. Introduction

Industrial wireless technologies such as WirelessHART [1] and ISA100.11a [2] use centralized network management techniques for communication scheduling and establishing routes. While such an approach may be easier in terms of implementation, they have numerous disadvantages. Centralized systems often perform poorly in terms of reaction time as all updates need to be first sent to the centralized system manager (i.e. gateway) for further processing. The gateway then performs recalculations and disseminates updated instructions to the relevant nodes in the network. As the round-trip time for such decision-making actions can be very high (especially when network contention is high), centralized approaches are unable to cope with highly dynamic situations (e.g. bursty data traffic/varying link quality, and node mobility). This problem is further exacerbated as the network is scaled up. Moreover, the longer the route (in terms of hops) between the source and the gateway, the higher the variability of the latency of the data traffic between these two nodes. This uncertainty makes centralized approaches especially unsuitable for applications which require closed-loop control as they require messages to be delivered within a fixed time frame.

To mitigate these problems, this paper presents D-SAR which is a distributed scheduling algorithm for enabling real-time, closed-loop control that is suitable for harsh industrial environments. The distributed nature of our approach allows the system to adapt quickly to disturbances or changes within the network in real-time. Our approach, which focuses on allocating bandwidth resources, is based on concepts derived from Asynchronous Transfer Mode (ATM) networks. This is because ATM signaling protocols also address certain performance issues in terms of reliability and timeliness of packet delivery that are of importance in industrial applications that require closed-loop, real-time control. This paper presents initial ideas of our approach which we believe will act as the foundation of our future work in this area.

Section 2 describes the state-of-the-art in current technologies for wireless sensing, actuation and control for industrial automation. Section 3 provides some background about ATM and the MAC layer which is used in this algorithm. In addition, this article provides some details about the D-SAR algorithm in section 4. Section 5 describes the steps we take to verify the design of the protocol. Section 6 describes our future research directions in this area. Finally, Section 7 concludes the paper.

## 2. State of the art

Resource reservation from the perspective of bandwidth allocation is an essential part of a control system. As mentioned previously, both WirelessHART and ISA100.11a take centralized resource reservation approaches. For example, ISA100.11a uses a combination of resource reservation and traffic classification techniques for providing different QoS requests. Resource reservation, involves a device trying to establish communication with the central system manager or another device, by sending a contract request to the system manager. This contract request includes input parameters such as communication service type (scheduled or unscheduled), destination address, traffic classification (best effort queued, real time sequential, real time buffer and network control), etc. The system manager then uses its centralized optimization algorithm to determine the required allocation of the network resources (such as graphs and links) and sends a contract response to the source after all necessary network resources have been configured and reserved along the

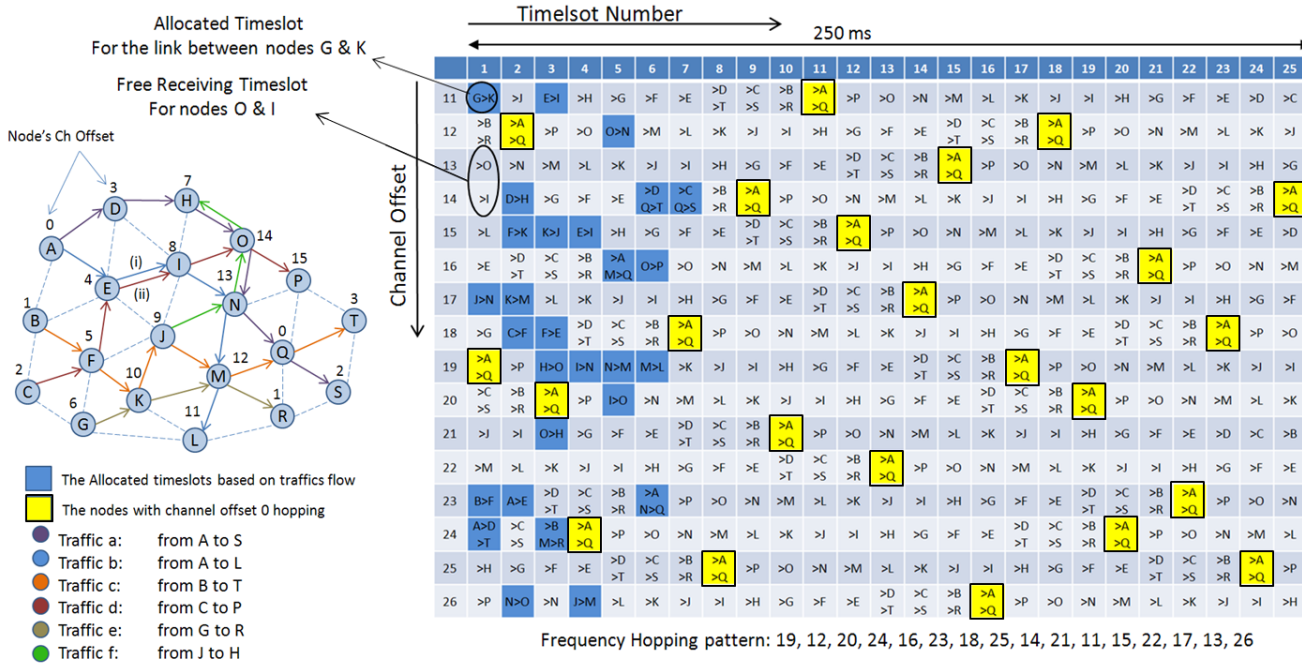


**Figure 1 Distributed timeslot and frequency allocation for different traffic flows**

path. However, ISA100.11a does not specify the specific optimization algorithms that can be used by the system manager to allocate resources. In [3-5] the authors propose a centralized scheduling algorithm in WirelessHART for convergecast by considering linear and tree network models.

In addition to resource reservation, reliability is also an essential part of a control system. The link quality between a source and destination node can heavily influence whether closed-loop control can be carried out successfully. One of the mechanisms used to improve link quality by trying to eliminate or minimize interference is *channel hopping*. Channel hopping can help prevent external interference and multipath fading. Channel hopping techniques are used in several industrial 802.15.4-based standards such as WirelessHART, ISA100.11a, and IEEE 802.15.4e (Time Slotted Channel Hopping (TSCH) mode). Among these, WirelessHART, ISA100.11a and IEEE 802.15.4e are designed using concepts derived from the Time Synchronized Mesh Protocol (TSMP) [6]. TSMP is a media access and networking protocol that is designed for low power and low bandwidth reliable communication. TSCH is a Medium Access Control (MAC) scheme which is a subset of TSMP and enables robust communication through channel hopping and high data rates through synchronization. It is based on a time-slotted architecture, where a schedule dictates on which slot and which channel a node should transmit/receive data to/from a particular neighbor. Unlike TSMP, TSCH does not address routing issues and leaves this to the upper layers.

While TSCH describes the channel hopping mechanism, it does not describe how the schedule is built, i.e. it does not define when a node should communicate with a particular neighbor. However, the next upper level (6LoWPAN) that resides on top of TSCH, assumes that nodes are capable of communicating with all their neighbors. This clearly indicates that there is a "gap" that exists between these two adjacent layers.

This paper presents a distributed scheduling algorithm that would allow the TSCH MAC protocol to be glued to the next higher layer. The presented approach defines how and when nodes communicate with their neighbors. Also, as our approach is based on techniques used in ATM networks, our final aim is to develop techniques to support both constant rate and bursty traffic. This paper, however, only considers the case of constant rate traffic. Thus we assume that the data traffic between sensors and actuators has a constant rate.

## 3. Background

### 3.1 ATM networks and Circuit Switching

Large-scale, distributed, real-time control applications require data to be transmitted over long distances through a multi-hop network in a timely manner. A distributed resource reservation algorithm is needed which would allow source nodes, based on the requirements of the application and traffic characteristic, to reserve network resources for its peer communications along their paths for addressing different QoS needs. The distributed nature allows the system to adapt quickly to disturbances or changes within the network in real-time. While such mechanisms do not exist for present day sensor nodes, relevant techniques from other networking-related domains could potentially be adapted to develop solutions that are

suitable for wireless sensor and actuator networks. QoS in multi-hop networks can be supported by different mechanisms and one of these techniques is ATM.

QoS in multi-hop networks can be provided using certain mechanisms found in circuit and packet switching protocols and the ATM protocol. Some of these mechanisms allow a source node to request a special end-to-end QoS for specific data flows or classes of data by reserving the resources and setting up a path between the source and destination(s).

Circuit switching is a technology primarily designed for telecommunication networks. It establishes a dedicated link between the source and destination for the duration of the communication thus guaranteeing a certain level of QoS. This reservation mechanism can play an important role in transferring real-time traffic. However, reserving routes and resources only for certain specific flows, means that the routes cannot be used by other flows. In other words, the route remains reserved even if it is not being actively used. This makes it unsuitable for bursty traffic conditions. Packet switching, however, is specifically designed for delivering bursty traffic over a shared network by using statistical multiplexing but it does not provide any QoS guarantees.

The ATM protocol uses a switching technique that combines the concepts of circuit switching and packet switching. For example, similar to circuit switching, before initiating data transfer, a virtual circuit is first established between the source and destination. This is performed by ensuring that time slots are available in each of the nodes along the reserved route. The connection fails if the required portion of the bandwidth cannot be allocated on each of the links. The protocol also includes admission control mechanisms that help determine whether the required QoS guarantees can be provided. ATM uses statistical multiplexing techniques, similar to those used in packet switching in order to cope with variable bit rates (i.e. bursty traffic).

## 3.2 Time Slotted Channel Hopping (TSCH)

TSCH is a MAC protocol that allows reliable communication by using a channel hopping mechanism. It divides the wireless channel into time and frequency. Time is divided into discrete time slots. TSCH models the RF space as a matrix of slot-channel cells. Figure 1 shows a similar approach.

TSCH uses the concept of a superframe which is a collection of cells which repeat at regular intervals. For example, Figure 1 illustrates that a slot of length 10ms repeats once every 250ms when the superframe consists of 25 slots. By scheduling each transaction (i.e. Tx-Rx operation) in one cell, the hidden terminal problem is prevented, as adjacent links never transmit simultaneously on the same frequency. A link is a transaction that occurs within a cell. It consists of a superframe ID, source and destination IDs, a slot number referenced to the beginning of the superframe, and a channel offset. The simplest version of a link contains one transmitter and one receiver. The two nodes at either end of the link communicate periodically once every superframe. If only one transmitter is scheduled, the link is contention-free, but a slotted CSMA approach can be used if multiple transmitters are scheduled to use the same cell simultaneously. TSCH links hop pseudo-randomly over a set of predefined channels, one packet at a time. Each time a link is activated, both sides of the link calculate the radio channel of the communication by taking (Absolute Slot Number + Channel offset) % Number of channels. For example, in Figure 1, Node E which has an offset of 4 will use channel 16 in slot 1 based on the frequency hopping pattern that is provided in the figure.

# 4. The D-SAR Algorithm

As we focus specifically on applications that require constant data traffic rates, our solution allocates a virtual circuit for each traffic flow. This implies that the resources reserved for each endpoint-to-endpoint connection will depend on the traffic characteristics.

There are two separate approaches for carrying out resource reservation. One approach based on the circuit switching concept would be to dedicate specific links in the network *only* for one particular traffic flow. The second approach based on ATM networks would allow links in the network to be *shared* between multiple traffic flows. For example, let us consider *Traffic b* (involving nodes A, E, I, N, M, L) and *Traffic d* (involving nodes C, F, E, I, O, P) in Figure 1. Using the circuit switching concept, between nodes E and I, *Traffic b* will only be allowed to flow through *Link (i)* and *Traffic d* will only be allowed to flow through *Link (ii)*. However, based on the second approach, both *Traffic b* and *Traffic d* will be allowed to use both *Link (i)* and *Link (ii)*. The advantage of this approach is that it allows for better utilization of every individual link. In addition, a node could also choose to send its data to multiple adjacent neighbors (i.e. links) thus reducing latency. We follow the second approach in this paper due to the above-mentioned benefits. We now provide the details of our distributed algorithm for resource reservation.

Similar to TSCH, we use a set of predefined frequency hopping sequences. The number of possible channel offsets is equal to the total number of channels used. Nodes broadcast advertisements to enable network formation and exchange timing information. An advertisement also includes channel offset information about a node and its immediate neighbors. This effectively allows a receiving node to gather channel offset information about its two-hop neighborhood. Each new node can choose a free channel offset based on this information. In order to communicate, a transmitter node switches its frequency to that of the receiver using a combination of the neighboring node's channel offset and the predefined frequency hopping sequence.

Existing technologies such as WirelessHART and ISA100.11a carry out their scheduling in a centralized manner by using the system manager to define the channel offsets and hopping sequences of every link in the network.

Our distributed resource reservation algorithm has three phases. The first phase, which involves network formation, uses a mechanism based on TSCH. A new node joining the network has to be assigned the appropriate resources so that it can carry out tasks such as broadcasting advertisements, receiving join requests, sending join responses, and communicating with others. The following step is to define the individual links between a node and all its adjacent neighbors so that they can be used by the routing layer. A handshaking mechanism is needed between the new device and each of its neighbors in order to choose the free timeslot (which was announced in their advertisement) thus allowing neighboring nodes to agree to communicate in a particular cell. As the network formation and route set up are low priority operations as opposed to control data traffic, we use shared instead of dedicated cells based on a CSMA approach.

In the second phase, the routing layer will be responsible for finding routes between the endpoints. We make the assumption that the routing algorithm already exists.

The focus of the third phase is to establish the end-to-end connection for transporting the application's control data. More importantly, this phase is responsible for allocating bandwidth resources based on the traffic characteristics requested by the source node. This is distinctly different from the approach taken in [7] where the authors only focus on defining the links but do not consider the traffic characteristics. The message exchange operation used to set up this connection is similar to the procedure followed by the ATM signaling protocol [8]. The source node initiates this phase by sending a SETUP message. The format of this message is similar to the *Contract Request* in ISA100.11a. However, unlike in ISA100.11a, which sends the *Contract Request* to centralized system manager, the source node in this protocol sends the SETUP message to the following node along the route defined in phase two. The message includes input parameters such as the selected timeslot number for the communication with the next hop[1] when communication is established, destination address, traffic classification (extended QoS parameters), end-to-end transit delay, traffic ID, and requested period. The SETUP message will be forwarded in the network along the path to the destination through the links previously defined in the first phase. The sender node sets Timer $T_1$ after sending the SETUP message and waits for the response in the form of a CALL PROCEEDING message, from the following node along the route defined in the second phase as shown in Figure 2.

[1] The sender selects this free timeslot by listening to the receiver's advertisement.

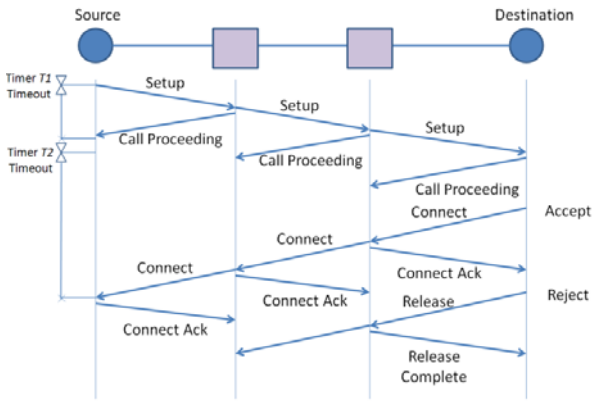


**Figure 2 Overview of connection establishment protocol**

The receiver of the SETUP message then performs a check of its available resources by performing an admission control operation. This operation checks whether free outgoing timeslots to the following hop are available. If the required timeslots are available, a CALL PROCEEDING message is sent to the sender. Upon receiving this message, the sender stops the Timer $T_1$ and starts Timer $T_2$. Next, the receiver of the SETUP message forwards the SETUP message to the next hop in the route. This process is continued until the SETUP message reaches the destination node. At this stage, all reserved timeslots along the route are only temporarily occupied.

If, however, the receiver of the SETUP message is unable to accommodate the new connection, it refuses it by responding with a RELEASE COMPLETE message.

Once the destination node receives the SETUP message it can either accept or decline the new connection request from the source node. Accepting the connection results in the destination node responding with a CONNECT message. This CONNECT message traverses along the multihop network back to the source node. Every intermediate node that receives a CONNECT message first stops Timer $T_2$ and then responds with a CONNECT ACKNOWLEDGE message directed towards the previous node in the direction of the destination node. When an intermediate node confirms the connection using a CONNECT ACKNOWLEDGEMENT message, it switches all the temporary timeslot reservations to permanent reservations. This two-step reservation is performed to ensure that timeslot reservations are not carried out if the connection request is unsuccessful. The destination node sends a RELEASE COMPLETE message to the source node if it decides to decline the connection request.

The details of the algorithm for the source, intermediate, and destination node are provided in algorithms 1, 2, and 3 respectively.

Algorithm 1: Connection establishment at the source node

```
1.  Receive the Setup-request primitive
2.  Select the free timeslot by listening to the next hop advertisement
3.  Send (SETUP) & Start Timer T1
4.  if Timer T1 expires before CALL PROCEEDING received
5.    if  retry counter exceeded
6.        Clear the connection
7.    else
8.        Send (SETUP) again
9.    end if
10. else
11.   Stop T1
12.   Send the proceeding-indication primitive
13.   Start T2
14.   Temporarily reserve the requested outgoing timeslot
15.   if Timer T2 expires before CONNECT received
16.       Clear the connection
17.   else
18.       Receive (CONNECT)
19.       Stop T2
20.       Send setup-confirm primitive
21.       Send(CONNECT ACK)
22.       Permanently reserve the requested outgoing timeslot
23.   end if
24. end if
```

Algorithm 2: Connection establishment at the intermediate node

```
1.  The intermediate node receives SETUP message
2.  Send setup-indication primitive
3.  if enough outgoing timeslots are available and the requested
    timeslot is accepted for the new connection
4.    Send (CALL PROCEEDING) to the previous node
5.    Temporarily reserve the requested incoming timeslot
6.    Select the free outgoing timeslot by listening to the next
    hop advertisement
7.    Forward (SETUP) to the next node
8.    Start T1
9.    if Timer T1 expires before CALL PROCEEDING or CONNECT
    received from next node
10.       if retry counter exceeded
11.           Clear the connection
12.       else
13.           Forward (SETUP) again to the next node
14.       end if
15.   else if the CALL PROCEEDING received from next node
16.       Stop T1
17.       Send proceeding-indication primitive
18.       Start T2
19.       Temporarily reserve the requested outgoing timeslot
20.       if Timer T2 expires before CONNECT received
21.           Clear the connection
22.       else
23.           goto line 26
24.       end if
25.   else if the CONNECT message received from next node
26.       Stop T2 or T1
27.       Send setup-confirm primitive
28.       Send (CONNECT ACK) to next node
29.       Permanently reserve the requested outgoing timeslot
30.       Forward (CONNECT) to previous node
31.       Start the Timer T3
32.       if Timer T3 expires before receiving CONNECT ACK
33.           Clear the connection
34.       else
35.           Receive (CONNECT ACK) from previous node
36.           Permanently reserve the requested incoming timeslot
37.       end if
38.   end if
39. else Send (RELEASE COMPLETE) to previous node
40. end if
```

Algorithm 3: Connection establishment at the destination node

```
1.  The node received the SETUP message
2.  Send setup-indication primitive
3.  If the node accepted the connection
4.    if received proceeding-request primitive
5.        Send (CALL PROCEEDING)
6.        Temporarily reserve the requested incoming timeslot
7.    end if
8.    if received setup-response primitive
9.        Send (CONNECT)
10.       Start T3
11.       if timer T3 expires before receiving CONNECT ACK
12.           Clear the connection
13.       else
14.           Receive (CONNECT ACK)
15.           Permanently reserve the requested incoming timeslot
16.           Activate the connection
17.       end if
18.   end if
19. end if
```

We allow the network to cope with varying data traffic rates by preventing established permanent connections to remain even if the connection is not required by the source and destination nodes or intermediate nodes which wish to terminate the connection due to resource constraints. To cope with this scenario, a node which wishes to end the connection transmits a RELEASE message. This message ensures that all nodes along the route release all the resources previously allocated for the connection.

## 5. Verification of the D-SAR Algorithm

In order to increase our confidence in the design of the protocol, we constructed a formal specification of the connection establishment protocol in mCRL2 [9]. Using this formal specification, we were able to verify almost fully automatically that a connection is always eventually established and that the protocol is deadlock free (both in case of normal operation, i.e. without message loss).

Concretely, the properties were verified by means of model checking: We considered a linear array of nodes from a source to a destination and generated all possible states the linear array can assume. The model checker of mCRL2 was employed. This model checker was chosen from among number of tools with almost identical functionality, such as Spin [10] and Uppaal [11], as it is most familiar to the

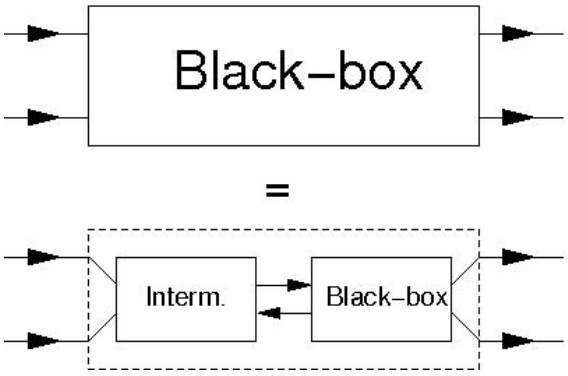


**Figure 3 Visual representation of the proof technique used in verification**

authors. Like Spin, but unlike Uppaal, mCRL2 does not support any notion of time. In our case, this means the specification does not include exact durations until timeouts occur; instead timeouts may happen at arbitrary moments, which implies the system exhibits more behavior. The additional behavior is such that the properties we were interested in could still be verified (essentially because only a finite number of timeouts are possible during connection establishment).

The model checking is slightly complicated, as there is a priori no bound on the number of intermediate nodes in a linear array and, hence, no fixed linear array of a certain size can be used. To handle this, we first used the model checker to calculate a specification of all observable (or black-box) behavior of an intermediate node (hiding internal details of the node). Next, the generated specification was composed with another intermediate node and again a specification of the observable behavior was generated (hiding also the messages passed between the two nodes) as shown in Figure 3. Finally, it was established that the observable behavior of intermediate node is in fact identical to that of the composition. By mathematical induction, the previous technique ensures that the number of intermediate nodes is irrelevant and that it suffices to consider a system composed of a source node, the observable behavior of the intermediate nodes, and a destination node. In this setting we were able to establish that a connection is always eventually established and that no deadlocks occur.

Note that we did not take into account either message loss or information about the topology and routes within the topology. Although the approach described above is easily extended to take into account message loss, it cannot deal with arbitrary topologies. To still gain some insight in the behavior of the protocol within arbitrary topologies, we plan to exhaustively generate all possible topologies up to a certain small number of nodes (say up to ten) and to apply model checking to all these topologies, similar to what is done in, for example, [12]. Although this will not give a full guarantee that the protocol works within every topology, it is reasonable to assume that flaws will already surface in these small topologies. Hence, this will still increase our confidence in the protocol.

## 6. Future Work

Significant advances are required before this protocol can be used for reliable, real-time, distributed control operations. We now highlight some of the key areas which need to be addressed to make this a reality.

### 6.1 Supporting bursty traffic

Our approach uses concepts from ATM networks to fulfill the real-time requirements. While our present protocol solely focuses on constant bit-rate traffic, we intend to extend it to support bursty traffic as well. Thus, the network can cope with the bursty nature of data traffic generated by the applications in the case of event occurrence when the large amount of traffic or report is needed to be forwarded to their destination. For delivering bursty traffic over a shared network, ATM provides the solution by considering a virtual circuit with statistical multiplexing. A similar mechanism can be applied to D-SAR to support bursty traffic.

### 6.2 Applying multipath mechanism

Additionally, to ensure robust communication multiple paths can be defined for each node to reach a particular destination in the network. In this approach if communication between a nodes and its next hop is disrupted due to interference, an alternative path can be used to transport the data. This approach is followed in several industrial wireless standards such as WirelessHART and ISA100.11a. We intend to consider this capability in the routing layer, and modify the D-SAR algorithm to support this approach.

### 6.3 Supporting point-to-multipoint

This paper focuses on establishing a point-to-point connection between one sensor and one actuator node but in certain industrial closed-loop control applications involving a sensor and multiple actuators, raw sensor readings are streamed from the sensor to the actuators. In traditional Fieldbus technologies such as Foundation Fieldbus, WorldFIP, and ControlNet, certain sensor nodes (the publishers) produce information which they publish to the network. Other groups of sensors or actuators (the subscribers) that are interested in that information listen to the publishers and update their local copy. This scenario can also occur in the wireless approach. In this case we have to consider establishing a point-to-multipoint connection. A point-to-multipoint connection allows one end point to send its traffic to two or more endpoints. The endpoint which generates the traffic is referred to as the root of the connection, whereas an endpoint that receives this traffic is referred to as a leaf. This feature exists in ATM networks and we intend to use the same concepts to add this capability to D-SAR.

### 6.4 Distributed collaborative power control method

In our present protocol, we allow a receiving node to gather channel offset information about its two-hop neighborhood, and choose a free channel offset based on this information. This scenario does not guarantee that the hidden terminal problem is solved because even offset information from the 2-hop neighborhood does not guarantee that two nodes that are in interference range do not transmit at the same time and hence cause collisions. We intend to use distributed and collaborative power control techniques to enable the node detecting interference to instruct the interfering node to re-adjust its Transmission power to reduce interference.

### 6.5 Considering adaptive channel hopping (ACH) mechanism

Channel hopping is often used to mitigate external interference and multipath fading. In this paper we considered the blind channel hopping technique. The other solution is using the adaptive channel hopping (ACH) technique in which the channel is changed on link-by-link basis only when necessary. There is a tradeoff between using blind channel hopping and ACH. In the former, if the node switches to another congested channel or switches from a good channel to a congested one, this hopping does not help to mitigate the interference and just wastes energy [13], however in ACH, nodes only change their frequencies when interference is detected on the current operating channel. Using ACH instead of considering the blind channel hoping can be helpful. However, nodes need to collaborate to decide which channel to switch to and this can introduce a significant overhead since nodes need to continuously scan all channels for interference levels and also because nodes need to ensure that while communicating nodes choose the same frequency, neighboring node pairs use different channels. We intend to add ACH to D-SAR in the future.

### 6.6 Simulation

In addition to verifying the correctness of our approach, we also intend to implement this protocol in a network simulator (NS-2) so as to make performance comparisons between our approach and existing technologies such as WirelessHART. The IEEE 802.15.4 standard 2003 package is available in NS and we intend to add IEEE 802.15.4e (Time Slotted Channel Hopping (TSCH) mode) to this package to support network-wide time synchronization, channel hopping, dedicated slotted unicast communication bandwidth, link layer ACKs, concurrent link activation, and omitting the poor method for synchronization such as sending Beacon. Several new MAC layer management entity (MLME) primitive should be added to the existing IEEE 802.15.4 standard 2003 package in NS in addition to the changes that are considered in IEEE 802.15.4 standard 2006. Network formation mechanism such as advertising and joining, network-wide time synchronization, and channel hopping are the main changes which should be applied in the existing NS package.

After adding the TSCH, D-SAR will be added to the existing package.

The final goal will be to carry out performance evaluations on actual sensor nodes in a harsh industrial environment.

## 7. Conclusion

This paper has described a distributed resource reservation protocol that is designed specifically for sensor and actuator networks that can be used in industrial applications that require real-time, closed-loop control. Our approach uses concepts from ATM networks to fulfill the real-time requirements. Since this solution uses a distributed approach, it can cope with disturbance or changes within the network in real-time and large-scale networks can also be supported. As our approach uses temporary connections which can be terminated at any time by the source and destination or intermediate nodes, the network can cope with varying data traffic rates and resource constraints or disturbances in the network. In addition to describing the algorithm in detail, this paper also describes how we verify the correctness of our approach using model checkers. We also outline the following steps we intend to take in the future to enhance the capabilities of D-SAR.

## 8. REFERENCES


[1] HART Communication Foundation. WirelessHART Technical Data Sheet. http://www.hartcomm.org

[2] International Society of Automation. ISA-100.11a-2009, Wireless Systems for Industrial Automation: Process Control and Related Applications. http://www.isa.org

[3] H. Zhang, P. Soldati, and M. Johansson, “Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks,” IEEE WiOpt, Seoul, Korea, Jun 2009.

[4] P. Soldati, H. Zhang, and M. Johansson, “Deadline-constrained transmission scheduling and data evacuation in WirelessHART networks,” The European Control Conference 2009 (ECC ’09)

[5] H. Zhang, P. Soldati, and M. Johansson, “Efficient Link Scheduling and Channel Hopping for Convergecast in WirelessHART Networks,” School of Electrical Engineering, Royal Institute of Technology (KTH), Tech. Rep., 2009

[6] Kristofer S. J. Pister and Lance Doherty, TSMP: Time Synchronized Mesh Protocol, Proceedings of the IASTED International Symposium on Distributed Sensor Networks (DSN08), Orlando, Florida, USA, November 2008.

[7] Andrew Tinka, Thomas Watteyne, Kris Pister. A Decentralized Scheduling Algorithm for Time Synchronized Channel Hopping. International Conference on Ad Hoc Networks (ADHOCNETS), Victoria, BC, Canada, 18-20 August 2010.

[8] Harry G. Perros, Ed., “Connection-oriented Networks SONET/SDH, ATM, MPLS and Optical Networks”. Wiley; 1 edition, May 6, 2005.

[9] Jan Friso Groote, Aad Mathijssen, Michel A. Reniers, Yaroslav S. Usenko, Muck van Weerdenburg, “The Formal Specification Language mCRL2,” Proc. of Methods for Modelling Software Systems, 2007, Dagstuhl Seminar Proceedings, 06351

[10] Gerard J. Holzmann, “The Spin Model Checker: Primer and Reference Manual”, Addison-Wesley, 2004

[11] Kim G. Larsen, Paul Pettersson, Wang Yi, “Uppaal in a Nutshell”, Int. Journal on Software Tools for Technology Transfer, Springer-Verlag, 134-152, 1997

[12] A. Fehnker, L. van Hoesel and A. Mader, Modelling and Verification of the LMAC Protocol for Wireless Sensor Networks, In Proceedings of the 6th International Conference on Integrated Formal Methods (IFM 2007). Springer, 2007.

[13] J. Ortiz and D. Culler, “Multichannel Reliability Assessment in Real World WSNs”, IPSN '10 Proceedings of the ninth ACM/IEEE International Conference on Information Processing in Sensor Networks, 162-173.