Network Synthesis for Industry 4.0

Enrico Fraccaroli, Alan Michael Padovani*, Davide Quaglia, Franco Fummi
Department of Computer Science, University of Verona, Verona, Italy,
Email: name.surname@univr.it, *Email: alanmichael.padovani@studenti.univr.it

***Abstract*—Today's factory machines are ever more connected with SCADA, MES, ERP applications as well as external systems for data analysis. Different types of network architectures must be used for this purpose. For instance, control applications at the lowest level are susceptible to delays and errors while data analysis with machine learning procedures requires to move a large amount of data without real-time constraints. Standard data formats, like Automation Markup Language (AML), have been established to document factory environment, machine placement and network deployment, however, no automatic technique is currently available in the context of Industry 4.0 to choose the best mix of network architectures according to spacial constraints, cost, and performance. We propose to fill this gap by formulating an optimization problem. First of all, spatial and communication requirements are extracted from the AML description. Then, the optimal interconnection of wired or wireless channels is obtained according to application objectives. Finally, this result is back-annotated to AML to be used in the life cycle of the production system. The proposed methodology is described through a small, but complete, smart production plant.**

***Index Terms*—Networked embedded systems, design-space exploration, synthesis, optimization, MILP.**

I. Introduction and Problem Statement

Recent trends in factory automation aim at increasing efficiency by making machines more connected together and with external applications. This implies that network infrastructure becomes another design dimension of the production system. Figure 1 shows an example of a factory layout. It is small to support the understanding of the proposed approach but it is also complete and realistic to show its general validity. It consists of four areas, i.e., two production areas, a warehouse, and a data center. In each production area, there are two conveyor belts and a machine; the belt on the left takes raw materials from a basket to be transformed by the machine; then finished goods are put in packages by the belt on the right. Raw materials are taken from the warehouse, which also stores finished goods. An Automatic Guided Vehicle (AGV) is used to move objects between the warehouse and the two production areas.

The shop floor is managed by applications hosted by the office network, which is usually called Information Technology (IT) network. Supervisory Control and Data Acquisition (SCADA) consists of applications and graphical user interfaces for high-level process supervisory management; it communicates with devices in the shop floor such as Programmable Logic Controllers (PLCs) to implement machine-level control strategies. Manufacturing Execution Systems (MES) is used to track and document the transformation of raw materials to finished goods. MES provides information that helps decision-makers to understand how current conditions on the plant floor can be optimized to improve production output. Enterprise Resource Planning (ERP) is a suite of integrated applications that an organization can use to collect, store, manage, and analyze data from its business activities. Other non business-related applications can be present, *e.g.*, to monitor machines and environmental parameters for predictive maintenance [1].

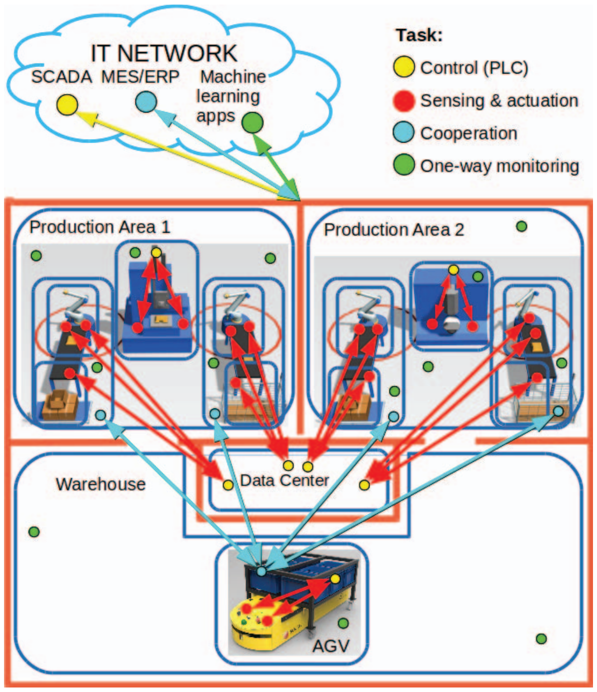

Fig. 1. Description of an Industry 4.0 case study with communication flows.

Communications are based on software tasks hosted by networked embedded systems placed inside machines and in the environment; these tasks exchange data with each other and with applications in the IT network. In Figure 1 the factory layout is annotated with *tasks* (denoted by circles) and their *data flows* (denoted by arrows). Red circles denote sensing and actuation tasks which generate measurements and apply commands, respectively. Yellow circles denote control tasks (traditionally implemented in PLCs) that interact with sensing and actuation tasks (red arrows) to implement control strategies. Tasks' position is a design requirement, *i.e.*, in the discussed example, the designer stated that machines and the vehicle have local control tasks while belts are controlled by the data center. Cyan circles denote cooperation tasks that interact all together to coordinate the drop of raw materials and the pick up of finished products between belts and the vehicle. Green circles denote monitoring tasks that collect data from the environment, machines, belts, and the vehicle for further analysis (e.g., for predictive maintenance). Furthermore, all control tasks interact with SCADA (yellow arrow), all cooperation tasks interact with MES/ERP (cyan arrow), and monitoring tasks interact with machine learning applications (green arrow). In Figure 1, specific arrows between these tasks and the IT network have been omitted to simplify the drawing. Data exchanges on different arrow types have different requirements. Red arrows require very low delays (around 1-10 ms) and low error rates (1% max), while the data rate is usually below 400 kb/s. Cyan arrows require moderate delays (around 10-100 ms), moderate error rates (3% max) with a data rate around 20 kb/s. Green arrows have no constraints on delay and error rate but exhibit a very high data rate of about 100 Mb/s. All these Quality of Service (QoS) requirements should be considered in the design of the network infrastructure of the shop floor usually called Operational Technology (OT) network.

The design of the OT network should also take into account the position of tasks in the factory layout. This requirement affects the placement of hardware devices (called *nodes* in the following text) which host tasks. As depicted in Figure 1, the milling machine, and its corresponding tasks are in Production Area 1 while the 3D printer is in the Production Area 2. In the specific case of belts, a sensor should be placed at each end of the equipment, while the actuator is placed in the upper end. Therefore, the two sensing tasks cannot be hosted by the same node while in the upper end of the belt, the sensing task and the actuating task can be implemented in the same node if this choice reduces costs. Furthermore, the size of the rooms and the presence of walls affect the network topology both for wired and wireless communications, while we can assume that network properties are homogeneous inside a room. In Figure 1, rounded blue boxes are depicted to denote *zones* with comparable properties and to separate tasks that cannot be placed in the same hardware device. As it can be seen in Figure 1, the specification of spatial requirements may lead to defining a large number of zones, but their reciprocal inclusion can be exploited to simplify the specification of their properties as described in Section II-C. It is worth noting that the AGV moves in different areas, and therefore, we cannot say that it belongs to one of the production areas or the warehouse. Furthermore, data exchanges for vehicle control (red arrows) are inside the AGV and thus static inside the vehicle, while data exchanges for cooperation and monitoring cannot be handled by wired communication channels.

The spatial features (walls and distances) may block communications. For instance, communications between belt and AGV in Figure 1, may be blocked by distance when the vehicle is far from the production areas and especially when it is behind the Data Center room. The design framework must also detect these cases and find *routing strategies* to allow communications while still fulfilling their QoS requirements. Section II-B addresses this issue.

The contributions of the paper are:

- Definition of a design flow for the industrial network infrastructure;
- Hierarchical specification of the spatial properties of the factory;
- Efficient deployment and routing based on spatial and QoS constraints;
- Experimental validation on a real example.

The paper is organized as follows. Section II describes the proposed design flow with focus on the routing algorithm and on the hierarchical approach for the definition of spatial requirements. Section III reports experimental results. Section IV introduces previous literature on this topic and finally Section V draws some conclusions.

II. AML-BASED DESIGN FLOW

The Automation Markup Language (AML) is a vendor-independent XML-based data exchange format, developed to unify all the multi-disciplinary designing tools used by production line engineers [2]. AML also allows to model network infrastructures [3] and thus it can provide a perfect environment for this work. Figure 2 depicts the proposed AML-based flow for the automatic synthesis of the industrial communication network. Information about plant environment and structure, application tasks and QoS of data exchanges are obtained through the requirement elicitation phase and written in the AML document. The designer should also provide a catalog of available physical devices (*e.g.*, models for PLCs, sensors and actuators) and network architectures (*e.g.*, Profinet, Ethernet, WiFi) together with their features. AML should be extended to host this information as explained in the next Section II-A. Network synthesis process takes all this information and provides the optimal (*e.g.*, *w.r.t.* to cost) network infrastructure with the placement of the physical devices with task assignment and of the communication channels. In particular a routing process is needed to concatenate basic channels to improve reachability as described in Section II-B.

A. AML-based Specification for Network Synthesis

AML stores engineering information following the object-oriented paradigm [4]. Figure 3 shows an example of AML-based description of industrial communication systems according to the approach specified in [3]. The plant structure depicted in the upper left corner can be expressed as an *instance hierarchy (IH)* containing *internal elements (IE)* reported in the lower left corner. They are instances of system objects that can be specified by using *system unit classes* as shown in the upper right corner. The semantics of these objects can be specified by using *role classes* reported in the lower right corner. Interfaces to interlink objects can be specified by using *interfaces classes*. Data processing functionality (*e.g.*, control and sensing) is captured by the `LogicalDevice` role while the hosting hardware device (*e.g.*, PLC or IO Device) is represented by the `PhysicalDevice` role. Similarly, the

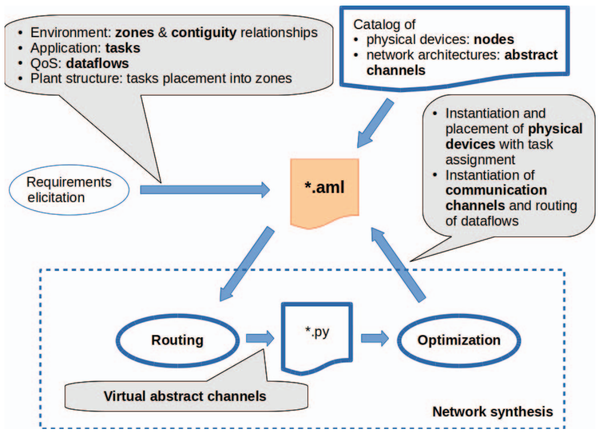
Environment: zones & contiguity relationships
Application: tasks
QoS: dataflows
Plant structure: tasks placement into zones
Catalog of
physical devices: nodes
network architectures: abstract channels
Requirements elicitation
*.aml
Instantiation and placement of physical devices with task assignment
Instantiation of communication channels and routing of dataflows
Routing
*.py
Optimization
Virtual abstract channels
Network synthesis

Fig. 2. AML-based synthesis of the industrial network.

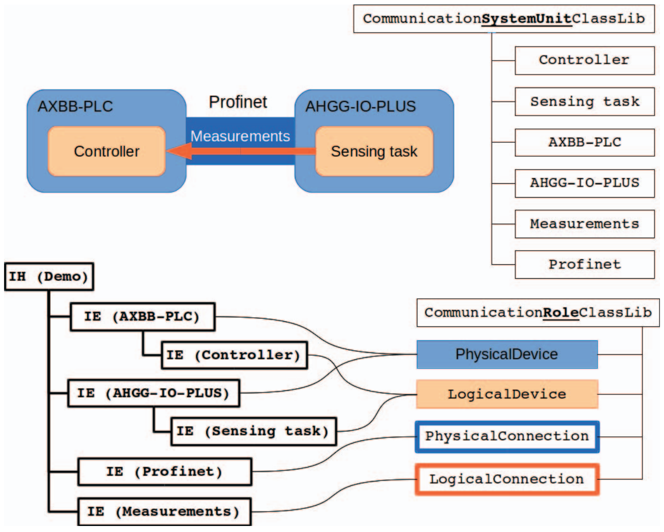

Fig. 3. AML-based description of a communication system.

data flow between logical devices (*e.g.*, measurements) and the corresponding hosting channel (*e.g.*, Profinet) are represented by `LogicalConnection` and `PhysicalConnection` roles, respectively.

Figure 4 shows the extension of the AML description to provide information for the network synthesis process. A new role class named `Zone` is introduced to partition the 3D space of the shop floor so that devices can be precisely placed and communication-relevant information, such as walls and distances, can be specified. Zones cannot be partially overlapped but one zone can be completely included into another zone. The class `Task` specializes `LogicalDevice` to introduce new attributes such as the computational requirements of the application function, if it is mobile or not and the zone to be placed. The class `Node` extends `PhysicalDevice` to specify its computational power, cost, power consumption and mobility. A node can host one or more tasks according to the match between their corresponding attributes. The class `DataFlow` extends `LogicalConnection` to specify the

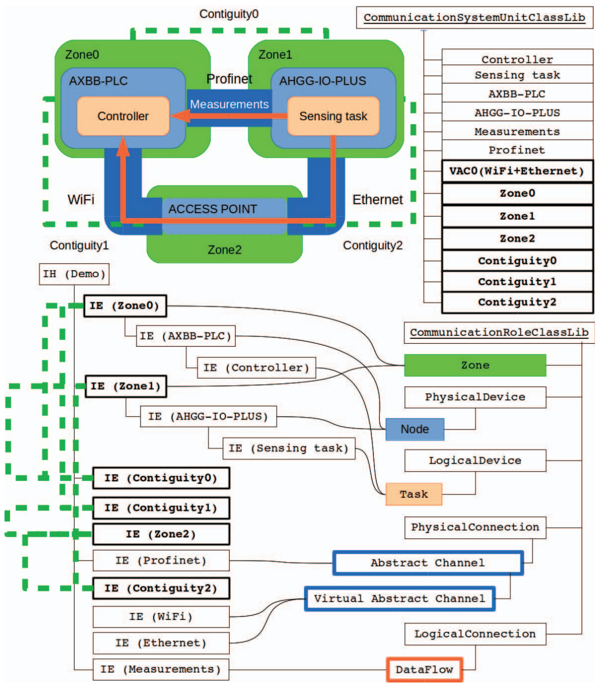

Fig. 4. Extension of AML description for the network synthesis.

QoS requirements for the data exchange (*i.e.*, bitrate, max delay, max error rate). The class `AbstractChannel` extends `PhysicalConnection` to specify the QoS performance provided by a complete protocol stack, its deployment cost and the wired/wireless nature of its physical layer. An abstract channel can host one or more dataflows according to the match between their corresponding attributes. A new interface class named `Contiguity` is defined to put two zones and an abstract channel in relation. It has an attribute that modifies the QoS of the given abstract channel according to the effect of crossing the two given zones. It also has an attribute which is relevant only for wired abstract channel to represent the cabling cost. Contiguity relations are used during dataflow-to-abstract channel allocation and ideally should be specified for each zones pair by performing a site survey of the shop floor. In Section II-C a methodology is introduced to save effort.

Starting from this formalization, network synthesis consists in finding the solution of an optimization problem which describes the communication infrastructure in terms of mapping of Tasks onto Nodes, their spatial displacement onto Zones, and the Abstract Channels among them. A possible Mixed Integer Linear Programming (MILP) implementation of this process can be found at [5] even if this code is not able to concatenate Abstract Channels to route dataflows that cannot be allocated to a single Abstract Channel. With reference to Figure 4, let us suppose that $Zone0$ and $Zone1$ are not directly reachable and that the only alternative consists in routing $Measurements$ dataflow through $Zone2$ by concatenating a WiFi channel to an Ethernet channel through an access point. As far as we know, no previous work proposed to allocate the dataflow into this route automatically. The next Section II-B describes a routing phase which we introduced before the optimization phase to look for sequences of abstract channels that allow to make two tasks reachable. Each sequence is defined *Virtual Abstract Channel* (VAC) and represented by the corresponding role class which extends `AbstractChannel`. VACs are added to the list of abstract channels and used in the optimization phase to allocate dataflows.

B. Routing

In the proposed routing phase, a graph is built by using the entities stored in the AML document. Graph vertices represent Zones while edges represent instances of Abstract Channels whose QoS is determined by the Contiguity relation between the two zones. The routing process is activated for each dataflow that cannot be hosted by an instance of a single abstract channel. The process starts from the source zone of the dataflow, and performs the following steps:

1) For each contiguous zone (*i.e.*, adjacent vertex) it gathers all the *compatible* channels which can be used to reach that zone. Characteristics of the channel and existing contiguity relations determine if the channel is *compatible*. More precisely, a channel is *compatible* if:
   a) it has the QoS performance (*i.e.*, bandwidth, delay and error rate) which matches with the dataflow requirements. Its error rate must be lower than the maximum allowed one. Its delay, summed with the delays introduced by previously selected channels, must be lower than the maximum allowed one;
   b) the *current* zone is the initial one, and the source task of the dataflow has the *mobile* attribute, then the channel should be *wireless*;
   c) the *next* zone is the final one, and the target task of the dataflow has the *mobile* attribute, then the channel should be *wireless*.
2) Once the algorithm has gathered all the suitable edges, it recursively moves to the zones specified by the edges.
3) It updates the accumulated *economic cost*, *delay*, and *energy consumption*, based on the selected channel.
4) Reached the new zone, following the selected edge, the algorithm repeats Point 1). The algorithm stops if, from the current zone, there are no viable routes to contiguous ones, or if it reaches the destination. In the latter case, the algorithm stores the solution as a new VAC.

The routing algorithm aims to find all the viable VACs for each dataflow and, for each VAC, to compute its overall QoS and cost by considering the list of component channels and the introduction of intermediate systems. This information is used in the subsequent optimization phase to find the best path based on the optimization metric.

C. Hierarchical Spatial Specification

From the AML description a set of trees is obtained by defining that a zone is a parent of all contained zones. Figure 5 shows this set for the case study. The main areas of the shop

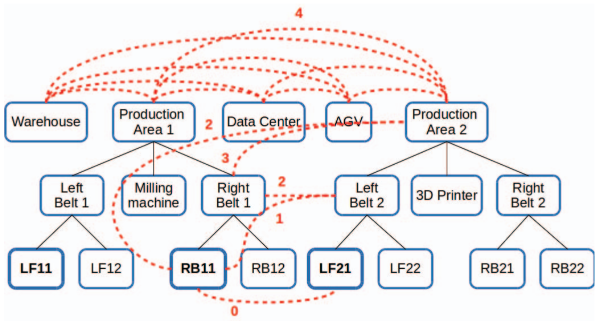

Fig. 5. The hierarchy of zones of the case study for a given abstract channel.

floor are described by five top-level zones. Each production area is divided to specify the position of machines and belts. Belts' space is further divided to indicate that belt tasks should be kept separated. For a given type of abstract channel, contiguity relations are denoted by dashed red edges.

Contiguity relations between zones are needed to check the capability of a given abstract channel to host a dataflow between tasks placed in those zones. Ideally, the designer should specify contiguity relations between all zone pairs with an effort that is quadratic with respect to the number of zones. However the hierarchical arrangements of zones and the procedure described by Algorithm 1 allow reducing the number of contiguity relations to be specified manually. This algorithm is based on the following assumptions:

- contiguity information is always specified between top-level zones;
- contiguity information is propagated from each parent to its children in the hierarchy;
- a more refined contiguity information can be specified between any two zones if available and, in this case, parent's information is overwritten by this refined information.

If the AML description specifies the contiguity information for the given zones pair and channel, the algorithm returns it; otherwise, it follows parent links until it reaches the two root zones or a common parent (*i.e.*, provided by the *CommonParent* function). In the latter case contiguity is not considered. The *ExistsContiguity* function checks whether a contiguity relation has been specified between the given pair of zones by using the given channel. The *GetDepth* function returns the number of edges from the root zone to the given zone. In the beginning, the *min_distance* is set to the worst case, which is the sum of the depths of the two nodes plus one.

III. Experimental Validation

For the optimization we used Gurobi 8.1.0 with a Python 3.6.6 front-end extended from the network synthesis code available at [5]. Execution has been performed on a 64-bit Windows 10 machine featuring an Intel(R) Core(TM) i7-7500 U CPU @ 2.70GHz and 8 GB memory.

Input: *z1*: First zone, *z2*: Second zone, *ch*: Channel
Result: Contiguity between *z1* and *z1* with *ch*

```
1  if ExistsContiguity(z1, z2, ch) then
2      return GetContiguity(z1, z2, ch);
3  min_distance = GetDepth(z1) + GetDepth(z2) + 1;
4  common = CommonParent(z1, z2);
5  it1 = z1, level1 = 0;
6  do
7      it2 = z2, level2 = 0;
8      do
9          if ExistsContiguity(it1, it2, ch) then
10             if level1 + level2 < min_distance then
11                 contiguity = GetContiguity(it1, it2, ch);
12                 min_distance = level1 + level2;
13         if it2 == common then break ;
14         it2 = GetParent(it2), level2 = level2 + 1;
15     while not IsRoot(it2);
16     if it1 == common then break ;
17     it1 = GetParent(it1), level1 = level1 + 1;
18 while not IsRoot(it1);
19 return contiguity;
```

Algorithm 1: Procedure to find contiguity information between two zones for a given abstract channel.

A. *Validation of the Overall Flow*

We applied the design flow to the case study presented in Figure 1 assuming the availability of the set of network types and physical devices reported in Table I. The case study has a total of 20 *zones*, 30 explicitly specified *contiguity relations* (instead of 190 as explained in Section II-C), 47 *tasks*, and 43 *dataflows*. In the resulting infrastructure, all the control dataflows (*i.e.*, red arrows in Figure 1) have been placed into Profinet channels, which are the only ones that satisfy their delay requirements. The four dataflows between the AGV and the IT network cannot be directly placed inside a single channel because the corresponding zones are not directly reachable each other. Therefore the routing algorithm generated four virtual abstract channels by combining a WiFi segment and an Ethernet segment with an access point in the middle. The routing algorithm spent on average 7.73 s and 19.90 MB memory to compute the route for each of the four dataflows, with a total execution time of 29.60 s. The optimization step spent 148.67 s with a maximum peak of occupied memory of 81.38 MB. All the other non-critical flows have been placed into Ethernet channels to satisfy the best trade-off between cost and performance. Monitoring, sensing, actuation, and cooperation tasks have been placed inside SoC components, while the control tasks have been placed inside server-type nodes usually employed to build PLCs.

TABLE I
NETWORK TYPES AND PHYSICAL DEVICES FOR THE CASE STUDY

Name	*cost ($)*	*size (kb/s)*	*power (W)*	*delay (ms)*	*error (%)*	*wireless*
WiFi	75	10	20	100	3	*true*
Ethernet	25	100	10	10	1	*false*
Profinet	100	100	10	1	1	*false*

Name	*cost ($)*	*size (MB)*	*power (W)*	*mobile*
SoC	5	32	5	*false*
MPSoC	22	64	8	*true*
Mini-PC	98	128	12	*true*
Server	1000	1000	100	*false*

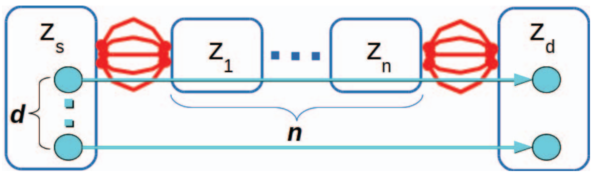

Fig. 6. The scenario used to test routing performance.

B. *Validation of the Routing Algorithm*

The validation of the routing algorithm is performed on a specific scenario (shown in Figure 6) to easily generate a large number of test configurations. There are d dataflows (denoted by cyan arrows) between source zone Z_s and destination zone Z_d. Since Z_s and Z_d are not directly reachable, routing is required through the sequence of n zones whose contiguity relations are arranged such that there are five abstract channels (denoted by red links) available between Z_s and Z_1, between Z_i and Z_{i+1}, and between Z_n and Z_d. The plot in Figure 7 reports the execution time on a logarithmic scale as a function of n and d. The number of zones is denoted by the colors while the number of dataflows is reported in the x axis. The plot shows that the time increases linearly *w.r.t.* the number of dataflows and exponentially *w.r.t.* the number of intermediate zones. This behavior is compatible with the theory of combinatorial optimization showing that the implementation is efficient.

IV. State of the Art

The IETF NETCONF Data Modelling Language Workgroup extended YANG language to model network topologies [6].

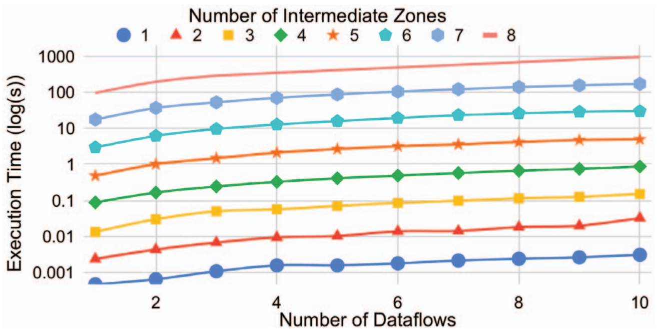

Fig. 7. Execution time of the routing algorithm as a function of the number of zones and dataflows.

It focuses only on the network, while we also address task-device allocation and device placement. Model-Driven Networking (MDN) [7] can be used to generate software-defined networks given a proper model created by using the Domain-Specific Modelling Language (DSML). A AML extension for network security has been also proposed [8]. It allows detailed modeling of each ISO/OSI layer of the communication infrastructure for validation purpose but it does not annotate AML with application requirements related to task placement and communication QoS for design purpose.

Many research works addressed network design. In [9], network topology and high-level functionality are used to configure the virtual architecture of Wireless Sensor Networkss (WSNs). However, this work is mainly focused on the application part of the system rather than on communication aspects. In [10] task graphs, HW/SW partitioning and task scheduling are used to design networked embedded systems. However it does not consider that scheduling could be optimized if communication aspects were considered earlier. In [11], the application is designed at a high level and then mapped onto a set of possible actual candidates for the nodes. However, no guideline is provided for selecting an appropriate network architecture and communication protocol. Scope-based techniques have been proposed in macro-programming to specify complex interactions between heterogeneous nodes of a WSN [12]. However, nodes number and network topology are an input rather than a result, as in the proposed approach. In [13] a tool for the synthesis of building automation networks is proposed but protocol choice is not addressed. Vice versa the work in [14] also considers this aspect. The affinity of that work with the problem we want to solve and the availability of the optimization code [5] make this approach very interesting for us even if it does not take into account routing.

A synthesis process for the routing of physical wires inside an automotive system is proposed in [15]. Xu et al. [16] propose a MILP formulation that comprises four specific groups of constraints: devices placement, link activation for routing, connections scheduling, and communication QoS. Their formulation is limited to ZigBee architecture while our approach is architecture-independent. In [17] a routing-aware tool for the optimal design of WSNs for building automation has been proposed. Routing is also addressed in the so-called Virtual Network Embedding Problem [18].

V. Conclusions

We presented a complete flow for the design of industrial communication networks which is based on AML standard and linear programming. We *1)* extended AML to document application requirements related to task placement and communication QoS, *2)* developed a routing algorithm to automatically aggregate sub-networks to obtain complete and QoS-aware reachability between tasks, and *3)* created an algorithm to handle spatial requirements in a hierarchical way. Experiments show that network technologies are allocated in a cost- and requirement-based way, *e.g.*, Profinet is used for real-time control flows, WiFi for vehicle communications and Ethernet for all other data flows. Furthermore, the computational complexity of the approach is acceptable.

VI. Acknowledgements

The work has been partially supported by the Italian Ministry of Education, University and Research (MIUR) "Dipartimenti di Eccellenza" 2018-2022 grant.

References

[1] R. K. Mobley, Ed., *An Introduction to Predictive Maintenance*, 2nd ed. Elsevier, 2002. [Online]. Available: https://doi.org/10.1016/b978-0-7506-7531-4.x5000-3

[2] R. Drath, A. Luder, J. Peschke, and L. Hundt, "Automationml - the glue for seamless automation engineering," in *IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Sep. 2008, pp. 616–623.

[3] F. Bendik and N. Schmidt, "Exchange of engineering data for communication systems based on AutomationML using an EtherNet/IP example," in *ODVA Industry Conference and 17th Annual Meeting*, 2015, pp. 1–21.

[4] International Electrotechnical Commission, "IEC 62714 - Engineering data exchange format for use in industrial automation systems engineering - AutomationML," 2014. [Online]. Available: http://www.iec.ch

[5] F. Enrico, "Network Synthesizer," May 2017. [Online]. Available: http://doi.org/10.5281/zenodo.3407162

[6] A. Clemm, J. Medved, R. Varga, N. Bahadur, H. Ananthakrishnan, and X. Liu, "A YANG Data Model for Network Topologies," RFC 8345, Mar. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8345.txt

[7] F. A. Lopes, M. Santos, R. Fidalgo, and S. Fernandes, "Model-driven networking: A novel approach for sdn applications development," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 770–773.

[8] F. Patzer, A. Sarkar, P. Birnstill, M. Schleipen, and J. Beyerer, "Towards the modelling of complex communication networks in AutomationML," in *IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Sep. 2017, pp. 1–8.

[9] A. Bakshi and V. Prasanna, "Algorithm design and synthesis for wireless sensor networks," in *Proc. of Int. Conf. on Parallel Processing*, 2004, pp. 423–430 vol.1.

[10] G. Gogniat, M. Auguin, L. Bianco, and A. Pegatoquet, "Communication synthesis and HW/SW integration for embedded system design," in *Proc. of the Int. Workshop on Hardware/Software Codesign*, 1998, pp. 49–53.

[11] A. Bonivento, L. P. Carloni, and A. Sangiovanni-Vincentelli, "Platform-based design of wireless sensor networks for industrial applications," in *Proc. of IEEE DATE*, 2006, pp. 1103–1107.

[12] L. Mottola, A. Pathak, A. Bakshi, V. K. Prasanna, and G. P. Picco, "Enabling scope-based interactions in sensor network macroprogramming," in *Proc. of IEEE Conf. on Mobile Adhoc and Sensor Systems*, 2007.

[13] A. Guinard, A. McGibney, and D. Pesch, "A wireless sensor network design tool to support building energy management," in *Proc. of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, 2009, pp. 25–30.

[14] E. Fraccaroli, F. Stefanni, R. Rizzi, D. Quaglia, and F. Fummi, "Network synthesis for distributed embedded systems," *IEEE Transactions on Computers*, vol. 67, no. 9, pp. 1315–1330, Sep. 2018.

[15] C. W. Lin, L. Rao, P. Giusto, J. D'Ambrosio, and A. L. Sangiovanni-Vincentelli, "Efficient wire routing and wire sizing for weight minimization of automotive systems," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 11, pp. 1730–1741, Nov. 2015.

[16] S. Xu, R. Kumar, and A. Pinto, "Correct-by-construction and optimal synthesis of beacon-enabled ZigBee network," *IEEE Trans. on Automation Science and Engineering*, vol. 10, no. 1, pp. 137–144, Jan. 2013.

[17] A. Puggelli, M. M. R. Mozumdar, L. Lavagno, and A. L. Sangiovanni-Vincentelli, "Routing-aware design of indoor wireless sensor networks using an interactive tool," *IEEE Systems Journal*, vol. 9, no. 3, pp. 717–727, Sep. 2015.

[18] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual Network Embedding with Coordinated Node and Link Mapping," in *Proc. of IEEE INFOCOM*, April 2009, pp. 783–791.