Design Space Exploration for Wireless-Integrated Factory Automation Systems

Honglei Li*, Jing Geng*, Yongkang Liu†,
Mohamed Kashef†, Richard Candell†, Shuvra S. Bhattacharyya*
* University of Maryland, College Park, USA
† Intelligent Systems Division, National Institute of Standards and Technology, USA
Email: {honglei, jgeng}@umd.edu, {yongkang.liu, mohamed.hany, richard.candell}@nist.gov, ssb@umd.edu

***Abstract*—Recent years have brought significantly increased interest in integrating wireless communication capability within factory automation systems. Such integration motivates the study of interactions among the physical layout of factory workcells, wireless communication among workcells, and improving the overall factory system performance. In this paper, we develop methods for modeling and simulating these interactions, and implement these methods into the experimental study of complex design spaces for factory automation systems. The proposed methodology for modeling, simulation, and design space exploration can be used to gain insight into approaches for improving the configuration (e.g., physical layout or wireless protocol settings) of existing factory systems, and for understanding trade-offs in the design of new systems.**

I. Introduction

Modern factory automation systems are equipped with advanced wireless communications capability. Integration of such capability provides important potential advantages, such as lower cost to deploy and maintain networking capabilities within factories, and the ability to install sensors and monitoring functionality in parts of factories that are not possible to be efficiently instrumented using wired communications (e.g., see [1]).

Along with these potential advantages, integration of wireless communications introduces new challenges and novel constraints in the analysis and design of factory automation systems. A major source of these new challenges and constraints is the complex interaction among the factory layout and configuration. This interaction includes the placement of factory subsystems and their partitioning into nodes of the wireless network (network nodes); the performance of the wireless network that connects network nodes; and overall factory system performance. These factors lead to complex design spaces, which are composed of factory layouts, wireless communication networks, and interactions between them in system configuration and operations. We refer to these design spaces as *wireless-integrated factory system* (*WIFS*) design spaces.

In this paper, we develop new models and evaluation tools for understanding and experimenting with WIFS design spaces. Since evaluating these design spaces by physically constructing the different layout/networking combinations is in general infeasible, we present a new simulation-based design space exploration tool called *WISE* (*Wireless-Integrated factory System Evaluator*). WISE is designed for model-based simulation of factory automation subsystems that are equipped with wireless communications capability, and rapid simulation-based evaluation of alternative networked factory system designs.

Here, by *model-based*, we mean that the modeling techniques that underlie the tool are based on formal models of computation rather than on ad-hoc, tool-specific techniques that are difficult to precisely understand or to adapt to other modeling and simulation environments. Model-based design is a useful concept for many areas of cyber-physical systems and signal and information processing (e.g., see [2], [3]). The specific forms of model-based design emphasized in WISE are *dataflow modeling* for factory process-flows, and systematic interfacing of dataflow models with arbitrary network simulators that are based on discrete-event modeling.

The emphasis on dataflow is useful due to the utility of dataflow modeling across the areas of signal processing, control, and machine learning [3], which are all relevant to design and implementation of factory automation systems. This allows not only the high-level process-flow behavior of process networks to be modeled naturally and formally with WISE, but also lower level subsystems of the process-flows. Such a unified, model-based approach across levels of design hierarchy is useful for enhancing design modularity, analysis, and optimization.

Important features of WISE include capabilities for automatically generating (autogenerating) complex lower-level simulation models from compact representations at higher levels of abstraction. WISE also applies a new concept of *cyber-physical flow graphs* (*CPFGs*) as a graph-theoretic model for factory process-flows and other flow-oriented types of cyber-physical systems. We demonstrate WISE through extensive experiments that highlight its utility for exploring complex WIFS design spaces.

II. Background and Related Work

A significant body of the existing literature is relevant to modeling and simulation of factory automation systems that are equipped with wireless communication capabilities. Some of these works are based on novel applications of

978-1-7281-1268-8/19/$31.00 ©2019 IEEE

existing simulation frameworks. For example, Liu et al. apply the OMNET++ simulation library to develop an integrated framework for factory process control simulation and wireless network simulation [4]. Marghescu et al. study the simulation of Zigbee-based wireless sensor networks using OPNET to evaluate and optimize the various network parameters [5]. Harding et al. develop a simulator that incorporates mathematical modeling and feedback control by developing an interface between MATLAB and OPNET [6].

Other works emphasize new models or simulation methods. For example, Vogel-Heuser et al. present approaches for modeling real-time requirements and properties of networked automation systems [7]. Schlick discusses advances, such as component-based automation and self-organizing production systems, in cyber-physical systems for factory automation [8]. Kurte et al. introduce a simulator for wireless sensor and actuator networks that allows simulation of heterogeneous systems through a novel interface abstraction for the operation of physical radio hardware [9]. Chaves et al. present a design environment for simulation and testing that is based on a service-oriented software architecture [10].

The novelty of the contribution in this paper centers on the development and application of WISE to explore complex WIFS design spaces. Compared to related work such as the works summarized above, distinguishing characteristics of WISE include its model-based architecture, which systematically integrates dataflow-based modeling of factory process-flows with discrete event modeling of wireless communication networks. WISE also provides autogeneration of low-level simulation code from high-level models, and cyber-physical flowgraph modeling, which further enhance the utility of the tool for WIFS design space exploration.

III. Design Flow

Fig. 1 illustrates the design flow associated with applying WISE for WIFS design space exploration.

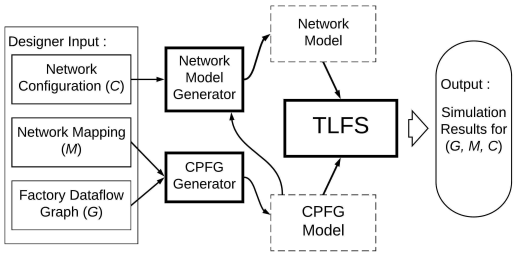

Fig. 1. An illustration of the new design flow involved in applying WISE for WIFS design space exploration.

As illustrated in Fig. 1, WISE builds upon a recently-introduced co-simulation tool called Tau Lide Factory Sim (TLFS) [11]. TLFS provides dataflow-based modeling of factory process-flows and systematic integration of the resulting process-flow models with arbitrary discrete event tools for network simulation. As illustrated in Fig. 1, WISE introduces and integrates with TLFS two new software tools, called the Network Model Generator and the CPFG Generator, and one new intermediate representation (graphical modeling data structure), called the CPFG Model. Additionally, the implementation of TLFS is extended in this work to support details of the CPFG Model. In Fig. 1, designer input, intermediate representations, and software tools are represented with thin-solid, dashed, and thick-solid borders, respectively.

A. Model-Based Architecture

The model-based architectures of WISE and TLFS emphasize dataflow-based modeling of factory process-flows and systematic interfacing between the process-flow models and arbitrary discrete-event simulators for communication architectures. Due to the abstract, model-based architectures of WISE and TLFS, the co-simulation and design space exploration techniques can be adapted readily to different dataflow-based design tools (for the process-flow modeling), and different communication network simulators.

A specific configuration of WISE involves two "plug-in" components for dataflow and communication network simulation. We refer to the two plug-ins as the *dataflow simulation plug-in* and *network simulation plug-in*, respectively. WISE systematically integrates the given pair of plug-ins into a model-based environment for exploring WIFS design spaces. In our experiments, which we report on in Section V, we utilize two specific dataflow and network simulation tools as plug-ins. These tools are, respectively, (1) the lightweight dataflow environment (LIDE) [12], and (2) the NS3 network simulation tool [13]. However, as described above, the model-based design of the WISE architecture enables retargeting the design space exploration techniques to other tools for dataflow and network simulation. This retargetability is useful because both of these areas for tool development — dataflow and network simulation tool development — are active areas for research and innovation.

B. Designer Input

The blocks in Fig. 1 labeled *Factory Dataflow Graph*, *Network Mapping*, and *Network Configuration* refer to simulation model input that is provided by the designer to represent the WIFS that is currently being studied.

The Factory Dataflow Graph models the factory process-flow between factory subsystems as a dataflow graph. The Factory Dataflow Graph is specified in a manner that is independent of the wireless network that is used for communication across distributed subsystems of the factory. Instead, the partitioning of Factory Dataflow Graph components into nodes of a wireless communication network, and the configuration of the network are specified separately. These specifications, represented by the blocks in Fig. 1 labeled Network Mapping and Network Configuration, are elaborated on in Section III-D. More details about Factory Dataflow Graph models are discussed in Section III-C.

The separation of concerns among the Factory Dataflow Graph, Network Mapping, and Network Configuration representations improves the efficiency and automation with which

the system designer can explore different ways of integrating wireless communication functionality into a given factory process-flow. In particular, the designer does not have to modify the Factory Dataflow Graph when the communication architecture changes; instead, only the relevant parts of the Network Mapping and Network Configuration specifications need to be changed. Then the detailed factory/communication co-simulation model is generated automatically. This separation of concerns and associated autogeneration capability is a major advance of WISE beyond TLFS.

C. Factory Dataflow Graphs

Formally, a Factory Dataflow Graph is a directed graph $G = (V, E)$, where the vertices (elements of V) represent factory subsystems such as machines, rails, parts generators, and machine/rail controllers. Directed edges (elements of E) in G represent the flow of information or physical entities (such as manufacturing parts) between factory subsystems. In the general terminology of dataflow graphs, the graph vertices are referred to as *actors*. Thus, actors in the Factory Dataflow Graph correspond to factory subsystems.

A dataflow graph executes by repeatedly executing actors that are ready (*enabled*) for execution, where the dataflow model provides a precise formulation for this form of readiness. As actors execute, they exchange packets of information (*tokens*) across the edges in the graph. These packets can have arbitrary data types associated with them, ranging from primitive types such as integers or floating point values to composite data types that correspond to user-defined objects (in an object-oriented programming sense). In Factory Dataflow Graphs, tokens may, for example, encapsulate information associated with the flow of physical parts, control messages, or instrumentation data.

Execution of a dataflow actor is decomposed into well-defined quanta of execution, called *firings*. Each firing is associated with characterizations of the amount of input data (number of tokens on the input edges) that is consumed by the firing, and the number of output tokens that is produced by the firing. These amounts of input and output data are referred to, respectively, as the consumption and production rates associated with the firing. An actor is said to be enabled for execution when there is a sufficient quantity of tokens buffered on its input edges, and a sufficient amount of empty buffer space available on its output edges to support the firing, as determined by the buffer sizes associated with the edges and the consumption and production rates of the firing.

For more background on the use of dataflow methods to model factory process-flows, we refer the reader to the detailed presentation of TLFS [11]. A notable difference, however, between the Factory Dataflow Graph of WISE and the dataflow graphs employed in TLFS is that Factory Dataflow Graphs do not incorporate any information about the communication architecture. These graphs are therefore simpler for the designer to work with. Furthermore, in conjunction with the separation of concerns described in Section III-B and the new automated model generation capabilities in WISE, Factory Dataflow Graphs are part of a more efficient approach for WIFS design space exploration. We elaborate on the automation capabilities further in Section IV, along with their utility in supporting design space exploration.

D. Network Mapping and Configuration

As shown in Fig. 1, the Network Mapping and Network Configuration are the two designer-provided inputs to specify the communication architecture that is to be integrated with the Factory Dataflow Graph for a given WIFS co-simulation (factory/network co-simulation). The Network Configuration input includes aspects related to factory layout.

Intuitively, the Network Mapping specifies how the given factory process-flow (as represented by the Factory Dataflow Graph) is distributed across different network nodes that communicate through wireless communication. The Network Mapping M for a Factory Dataflow Graph $G = (V, E)$ can therefore be represented as as a partitioning $M = N_1, N_2, \ldots, N_m$ ($m \geq 1$) of V — that is, the N_is are mutually disjoint subsets ($N_i \cap N_j = \emptyset$ for all $i \neq j$), and $N_1 \cup N_2 \cup \ldots \cup N_m = V$. To represent a fully centralized process-flow (with no wireless communication involved), one can simply set $m = 1$ so that the Network Mapping consists of just a single set $N_1 = V$. This type of mapping can be useful, for example, as a baseline to assess basic trade-offs associated with introducing wireless communication into the factory system.

The Network Configuration is another component of designer-provided input to WISE, as illustrated in Fig. 1. This input includes wireless communication parameter settings, such as the type of protocol and the propagation loss model. The desired Network Configuration settings are provided by the designer in a simple text file called `net_parameters.txt`. These parameter values are then converted to corresponding settings associated with the network simulation plug-in. To run a family of simulations with varying network parameters, the designer can easily edit the `net_parameters.txt` file or auto-generate a collection of files that can be iterated through for a set of simulation runs.

The parameters that can be specified in the `net_parameters.txt` file include the wireless communication protocol, propagation loss model, antenna transmitter gain, antenna receiver gain, noise figure for the noise signal, and others.

A Network Configuration specification for WISE also includes factory layout settings, which pertain to the spatial layout of factory subsystems, and can have significant impact on communication system performance. Factory layout settings in WISE network configurations are discussed in more detail in Section V.

E. Lower Level Models and Auto-generation

The input provided by the designer (user of WISE) is at a high-level of abstraction. This facilitates design space exploration because the models are easier to manipulate and reason about. However, to perform complete system simulation, the high level models must be translated into a lower-level form, which includes the simulation input to the network

simulation plug-in, and details of interfacing between the dataflow simulation plug-in and the network simulation plug-in. Such details are autogenerated in WISE by the blocks in Fig. 1 that are labeled Network Model Generator and CPFG Generator, respectively.

The output models that are generated by these two autogeneration subsystems are called the Network Model and CPFG Model, respectively. These two autogenerated models can be simulated together using WISE to achieve WIFS cosimulation between the given factory process-flow model and wireless networking capability that is integrated with the process-flow based on the given Network Mapping.

The structure and format of the generated network model are determined by the network simulation plug-in. As discussed previously, we presently employ NS3 as the network simulation plug-in. Thus, the Network Model Generator frees the designer from having to write NS3 code. The NS3 model is generated automatically from the designer's dataflow-based specification of the factory process-flow together with the Network Mapping and Network Configuration information.

The CPFG model includes special components, called *communication interface actors*, that model sending and communication of data between subsystems in a process-flow model. Communication interface actors model the exchange of data across a wireless communication network, and provide an abstract, modular interface between the dataflow simulation plug-in and the network simulation plug-in [11].

In Section IV, we discuss CPFG modeling concepts further, and provide an example of the CPFG model and parameterized network model that are generated from a given Factory Dataflow Graph and Network Mapping.

IV. Modeling and Autogeneration

In this section, we introduce details of the CPFG model, and its use as an intermediate representation in WISE. Second, we discuss communication link modeling for wireless channels in WISE. We also present a WIFS modeling example to illustrate the autogeneration of CPFG models and NS3 network models from the higher-level models provided as input to WISE.

A. Cyber Physical Flow Graph

The CPFG model is a specialized form of dataflow model that is useful for modeling and simulating WIFSs. In addition to its suitability for WIFSs, as we demonstrate in this paper, the CPFG model is applicable to a broad variety of modeling scenarios in cyber-physical systems. The CPFG model formulated here generalizes and formalizes an integrated, dataflow-based modeling approach for networked factory process-flows that was presented in preliminary form in [11].

Additionally, in this paper we introduce capabilities in WISE for autogenerating CPFG models from higher level representations. This is an important feature in streamlining the design process so that complex WIFS design spaces can be explored more efficiently, and more accurately.

A CPFG $G_{\rm cp} = (V_{\rm cp}, E_{\rm cp})$ is a dataflow graph whose actors can be partitioned into three subsets V_p, V_c, V_i, which are called the physical, computational, and communication interface actors of $G_{\rm cp}$, respectively. The computational actors correspond to actors in the usual sense of actors in signal processing oriented dataflow graphs (*dataflow process networks*) [14]. Such actors represent computational modules that represent discrete units of computation, called *firings*, as described in Section III-C.

Whereas an actor in a conventional signal processing oriented dataflow graph represents a computational module, a physical actor in a CPFG represents a physical subsystem or device, such as a factory machine or rail. A physical actor may encapsulate computational processing within it (e.g., processing that determines when to input a new part into a machine).

What distinguishes physical actors in the CPFG modeling approach is that any given physical actor must consume or produce *physical tokens* on at least one actor input or output, respectively. A physical token in turn models a discrete physical form of output (such as a generated or partially-processed part in a factory) rather than a packet of data, which is what a conventional dataflow token models. If a CPFG edge carries physical tokens, it is is referred to as a *physical edge*, otherwise, we call it a *cyber edge*.

As described in Section III-E, a communication interface actor (i.e., an element of V_i) models the sending or receiving of data across a communication network. In the CPFGs that we are concerned with in this work, the communication interface actors model wireless communication across distributed subsystems within a WIFS.

In WISE, communication interface actors provide a modular, model-based interface between the dataflow simulation plug-in and network simulation plug-in. For example, to retarget a CPFG to a different network simulator, one only has to change the implementations of the communication interface actor types. In WISE, we use only two types of communication actors, called *send interface actors* (*SIAs*) and *receive interface actors* (*RIAs*). Thus, only these two software components need to be retargeted to adapt a CPFG in WISE to work with a different network simulator.

As their names suggest, SIAs and RIAs model the sending and receiving of data, respectively, between dataflow actors across a communication network. For more background on SIAs and RIAs, we refer the reader to [11].

An example of CPFG modeling and associated use of SIAs and RIAs is presented in Section IV-C.

In summary, the CPFG model is distinguished by the partitioning of actors into physical, computational, and communication interface actors, and a dichotomy of edges as physical or cyber edges. A CPFG can apply general dataflow process networks [14] as the underlying dataflow model of computation or any specialized form of signal processing oriented dataflow that is compatible with the modeling requirements of communication interface actors. In this paper, we employ *core functional dataflow* (*CFDF*) [15] as the underlying dataflow model of computation. Background on CFDF and its utility in modeling factory process-flows is discussed in [11].

B. Communication Link Modeling

Fig. 2 illustrates different components of communication link modeling in WISE. Parameters associated with these components are configured by the designer as part of the Network Configuration block in Fig. 1, as described in Section III-D. Different antenna models are available for reception and transmission; the antenna is modeled as isotropic by default.

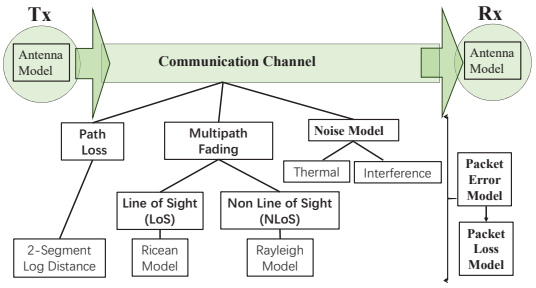

Fig. 2. Communication link modeling in WISE.

For the experiments reported on in this paper (Section V), signal noise is characterized as additive white Gaussian noise (AWGN). For the propagation loss model, a two-segment log distance model is applied. For multipath fading, Ricean and Raleigh models are used. For calculation of packet loss, the error rate is modeled based on a model presented by Miller [16], and subsequently validated by Pei and Henderson [17].

C. Autogeneration Example

In this section, we illustrate the models and autogeneration capabilities in WISE with a simple WIFS example.

Fig. 3 illustrates a Factory Dataflow Graph that is used to model a small-scale, pipeline-structured factory process-flow. The actor P represents a *parts generator*, which generates parts that are processed by the factory pipeline. The actors M_1 and M_2 model two machines that process parts, one by one, to add specific features to the parts. Parts are sent to and from each machine through rails, which are represented by the actors R_1 and R_2. The last stage in the pipeline is represented by the actor K. This actor, called the *parts sink*, represents a subsystem that collects and stores the parts after they are fully processed by the pipeline.

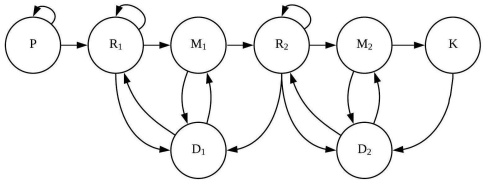

Fig. 3. An example of a Factory Dataflow Graph.

The actors D_1 and D_2 in Fig. 3 represent dual-rail, single machine (DRSM) controllers. A DRSM controller is a factory subsystem controller that is designed to interface with a single machine, a rail connected to the input of this machine, and a rail or parts sink that is connected to the machine output. Each DRSM controller sends commands to coordinate the flow of parts through the set of subsystems that it controls. For more details on the operation and modeling of DRSM controllers, we refer the reader to [11].

Fig. 4 illustrates the CPFG that is autogenerated by WISE for the Factory Dataflow Graph of Fig. 3 together with an example Network Mapping M. The mapping M involves seven distinct network nodes $N_1, N_2, \ldots, N_7$, and assigns the actors P, R_1, M_1, D_1, R_2, M_2, D_2, K, respectively to network nodes $N_1, N_1, N_2, N_6, N_3, N_4, N_7, N_5$. The solid edges in Fig. 4 carry physical tokens, while the dashed edges carry conventional dataflow tokens. In WISE, the determination of whether or not a given CPFG edge is a physical edge can be made automatically from the type of data that is associated with the Factory Dataflow Graph.

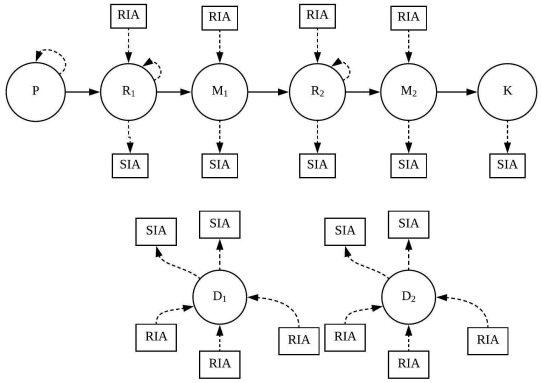

Fig. 4. Autogenerated CPFG.

The actors labeled SIA and RIA in Fig. 4 are communication interface actors that are automatically inserted by WISE in the process of autogenerating the CPFG. For each cyber edge whose source and sink actors are mapped to different network nodes, the communication associated with the edge is modeled with a separate (SIA, RIA) pair. For example, R_2 sends data to D_1, as shown by the edge (R_2, D_1) in Fig. 3, and these actors are mapped by M to distinct network nodes, N_3 and N_6, respectively. Accordingly an SIA S is connected to R_2 to model the sending of data to D_1 through a wireless channel, and a corresponding RIA is connected to D_1 to model the reception of data that is sent by S.

In WISE, all wireless communication is modeled in the autogenerated CPFGs through SIA-RIA pairs. Thus, all cyber edges in the CPFGs are associated with wired communication. In the current version of WISE, the latency of wired communication is assumed to be negligible compared to the latency of wireless communication and the execution time of machines. However, WIFS can readily be extended to incorporate latency models for wired communication — for example, by adding additional interfaces to the network simulation plug-in or by adding actors in the CPFG that model wired communication delays.

Fig. 5 illustrates the network model that is autogenerated

by WISE for the CPFG in Fig. 4. This graph shows the structure of the NS3 simulation model that is generated for co-simulation by TLFS with the generated CPFG. Each vertex N_i in Fig. 5 corresponds to a network node and each edge corresponds to a communication channel. The vertex Ap represents a single access point that is associated with the network nodes.

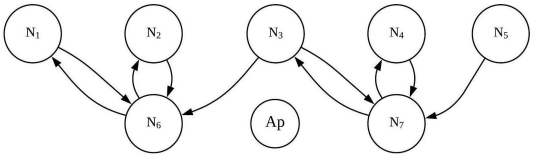

Fig. 5. The network model that is autogenerated by WISE for the example associated with Fig. 3 and Fig. 4.

Even for this simple, small-scale example, we see that the complexity of the CPFG together with the network model is significantly higher than that of the Factory Dataflow Graph, which is the designer's primary interface for working with WISE. This increase in complexity includes larger model sizes (more vertices and edges in the graph), as well as detailed software code that must be provided to correctly specify the lower-level models and ensure their consistency. The new models and autogeneration capabilities in WISE free the designer from the burden of managing this lower level design complexity.

V. Experiments

In this section, we demonstrate the utility of WISE through extensive experiments related to exploration of WIFS design spaces. We apply WISE in experiments with representative factory scenarios. Our experiments are performed using a desktop computer equipped with a 3.10 GHz Intel i5-2400 CPU, 4GB RAM, and the Ubuntu 16.04 LTS operating system.

A. Factory Layout Parameters

Presently, WISE assumes that a factory layout is in the form of one or more pipelines. Machines that belong to the same pipeline are arranged "horizontally", while different pipelines are arranged "vertically". Factory layout is therefore specified in terms of two distance-related parameters d_x and d_y, which respectively specify uniform (horizontal) spacing between successive subsystems (e.g., machines and rails) of a given pipeline, and uniform (vertical) spacing between successive pipelines in the vertical arrangement. Two additional layout-related parameters, N_p and N_m, specify the number of pipelines, and the number of factory machines within a given pipeline, respectively.

In most experiments in this section, we assume that each pipeline is assumed to have its own access point (AP), with a dedicated wireless channel assigned to each AP. It is assumed that if $N_p > 1$, then all of the dataflow occurs within the individual pipelines; that is, there is no communication across the pipelines. In Section V-G, we experiment with a set of scenarios in which all pipelines share a common access point.

The parameterized model of factory layouts supported in WISE represents a large class of factory systems with which capabilities of WISE can be demonstrated and experimented with. Also, the parameterized structure of the supported class of layouts is useful for demonstrating scalability-related factory performance trends. The extensible architecture of WISE makes it readily generalizable to support larger classes of factory layouts, such as layouts in which different pipelines have different numbers of machines, horizontal or vertical spacing between adjacent subsystems is non-uniform, or the overall layout structure does not necessarily involve horizontally-arranged pipelines. Such generalization is a useful direction for future work in WISE.

Fig. 6 shows an example of a factory layout of the form currently supported in WISE. In this example, $N_p = 2$, $N_m = 3$, and each pipeline has its own access point. Here, each $M_{i,j}$, $R_{i,j}$, and $D_{i,j}$ represents the jth machine, jth rail, and jth DRSM controller, respectively, for the ith pipeline. Each P_i, K_i, and A_i represents, respectively, the parts generator, parts sink, and access point for the ith pipeline.

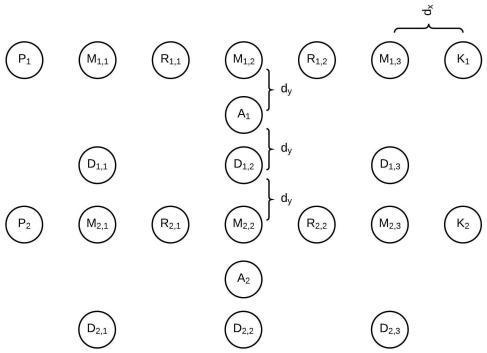

Fig. 6. Factory layout example.

B. Experiment Parameters

For each type of factory configuration simulated, we ran 50 WISE simulations independently and averaged the results. In each experiment, the simulation involved the production of 100 parts by each parts generator in each of the N_p pipelines, and the complete processing by the machines in each pipeline of the parts generated by the corresponding parts generator. The working time of each machine (the time required to process a given part) was determined randomly by the simulator using a designer-specified *mean working time* parameter μ. More specifically, the time for a given machine to process a given part was determined from a uniform distribution on $[0.9\mu, 1.1\mu]$. Each simulation terminated after the N_p parts sinks had each received 100 fully-processed parts.

The wireless communication protocol employed in all of the experiments reported on in this section is IEEE 802.11b. Since the protocol can be conveniently configured as part of the Network Configuration input to WISE, the experiments discussed here can be easily adapted to other protocols of interest.

WISE measures the communication delay associated with a packet P as $t_r(P) - t_s(P)$, where $t_s(P)$ is the time when P is sent by the corresponding SIA (see Section IV-A), and t_r is the time when P is received by the corresponding RIA. The average communication delay for a given simulation experiment is computed by averaging the difference $t_r(P) - t_s(P)$ over all communication packets.

C. Variation of Communication Delay with N_p

Fig. 7(a) shows how the average communication delay varies with the number of pipelines N_p. In this experiment, the Wi-Fi manager is configured to be the CARA (Collision-Aware Rate Adaptation) algorithm; $d_x = 10$ meters (m); $d_y = 10$ m; and $N_m = 3$.

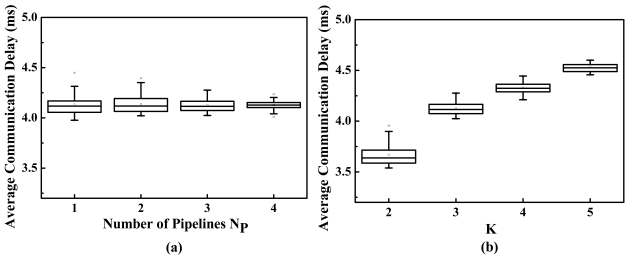

Fig. 7. (a) Variation in average communication delay with N_p. (b) Variation in average communication delay with $N_p = N_m = K$.

As shown in Fig. 7(a), the results for each N_p value are summarized in the form of a box plot. The endpoints of the vertical line segment for each plot extend from the minimum observed value to the maximum observed value. The three horizontal lines in each large box represent, from top to bottom, the 75th percentile, median, and 25th percentile of the corresponding set of 50 measurements. The small box inside each large box represents the mean value.

As shown in Fig. 7(a), the number of pipelines N_p has little influence on average communication delay for the class of factory systems considered in this experiment. This is because we allocate an independent access point for each pipeline and there is no communication between different pipelines.

D. Variation of Communication Delay with Both N_m and N_p

Fig. 7(b) shows results from an experiment where we have varied both the number of machines N_m and number of pipelines N_p. The variation is performed such that $N_m = N_p$. This allows us to visualize the effects of layout-complexity scaling in terms of a single parameter K, which is defined as the common value of N_p and N_m. The Wi-Fi manager algorithm is configured to be CARA as in Section V-C, and all other experiment parameters are as specified in Section V-B. The distance parameters are again configured as $d_x = 10$ m and $d_y = 10$ m.

As shown in Fig. 7(b), the average communication delay increases with larger K. This trend is largely due to two factors. First, the length of each pipeline increases with K, and correspondingly, the average distance from communication transceivers to the access point in each pipeline increases with K. Second, longer pipelines with more subsystems introduce more contention in the access points. The simulation results in this experiment provide specific insights on how communication delays vary and corresponding real-time performance issues are affected as a function of K, while other factory layout parameters are fixed.

E. Varying the Distance Parameters d_x and d_y

Fig. 8 presents a histogram of average communication delay, as determined by WISE simulation, with varying values of the distance parameters d_x and d_y. Each bar of the histogram is determined by averaging across 50 simulation runs. In this experiment, $N_p = 1$ and $N_m = 3$. The Wi-Fi manager algorithm is again configured to be CARA, and all other experiment parameters are as summarized in Section V-B.

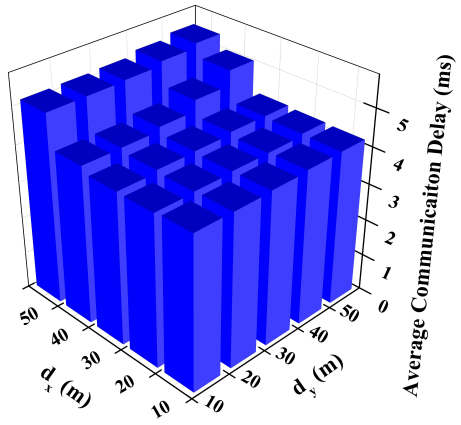

Fig. 8. Histogram of average communication delay with varying d_x, d_y.

This experiment shows a gradual trend toward increasing communication delay for $d_x, d_y \in \{10 \text{ m}, 20 \text{ m}, 30 \text{ m}\}$, while for values of $d_x, d_y \in \{40 \text{ m}, 50 \text{ m}\}$, we see steeper rates of increase. We expect that this accelerated increase arises due to nonlinear effects such as the way in which the Wi-Fi manager downgrades the data rate when a significant frequency of communication failures is encountered.

F. Varying the Wi-Fi Manager Algorithm

Fig. 9 shows changes in the average communication delay with changes in the Wi-Fi manager algorithm and number of machines N_m. For these experiments, $N_p = 1$, and $d_x = d_y = 10$ m. All other parameters are set as summarized in Section V-B. The Wi-Fi manager algorithms investigated in this experiment are: Collision-Aware Rate Adaptation (CARA), Adaptive Auto Rate Fallback (AARF), collision detection for adaptive auto rate fallback (AARFCD), and Adaptive Multi Rate Retry (AMRR) [18].

G. Shared Access Point across Pipelines

In this section, we revisit the experimental setup of Section V-C with one change: we use a single, shared access point across all pipelines instead of a separate access point for each pipeline. Thus, the total number of access points in a given factory layout is reduced from N_p to 1.

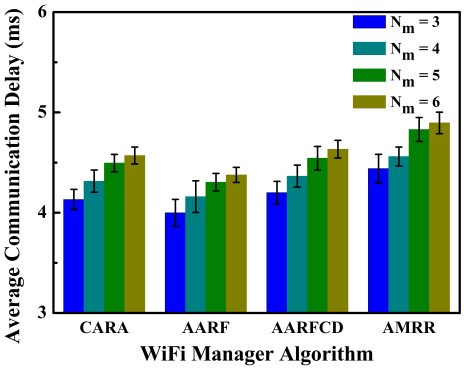

Fig. 9. Variation in average communication delay with the Wi-Fi manager algorithm and number of machines N_m.

As in Section V-C, the Wi-Fi manager algorithm is configured to be CARA; $d_x = d_y = 10$ m; and $N_m = 3$. All other experiment settings are as described in Section V-B.

Fig. 10 shows how the average communication delay varies with variation in the number of pipelines N_p under a single, shared access point configuration. We see in Fig. 10 a clear trend toward increasing average communication delay with increasing N_p. We anticipate that this is because with a single access point across all pipelines, increasing N_p results in more contention in the access point. Moreover, since d_x and d_y are fixed in this experiment, the average distance between communication transceivers and the access point increases with increasing N_p (see Fig. 6).

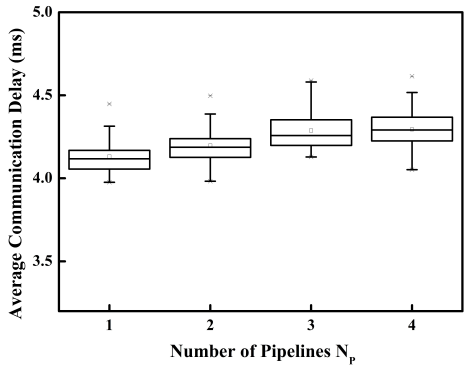

Fig. 10. Variation in average communication delay with N_p when a single, shared access point is used across all pipelines.

VI. Conclusions

In this paper, we have developed new models and computer-aided design tools that help in understanding and experimenting with complex, wireless-integrated factory system (WIFS) design spaces. Through extensive experiments, we have demonstrated the utility of our proposed new tools in exposing insights and performance trends involving multidimensional interactions among factory layout and communication system parameters. Useful directions for future work include developing optimization strategies, such as those based on randomized search (e.g., evolutionary algorithms or particle swarm optimization), for strategically iterating through families of simulations using our new WIFS-oriented models and tools.

Disclaimer

Certain commercial equipment, instruments, materials, software or systems are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

References

[1] A. A. Kumar S., K. Ovsthus, and L. M. Kristensen, "An industrial perspective on wireless sensor networks — a survey of requirements, protocols, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1391–1412, 2014.

[2] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems, A Cyber-Physical Systems Approach*, 2011, `http://LeeSeshia.org`, ISBN 978-0-557-70857-4.

[3] S. S. Bhattacharyya, E. Deprettere, R. Leupers, and J. Takala, Eds., *Handbook of Signal Processing Systems*, 3rd ed. Springer, 2019.

[4] Y. Liu, R. Candell, K. Lee, and N. Moayeri, "A simulation framework for industrial wireless networks and process control systems," in *Proceedings of the IEEE World Conference on Factory Communication Systems*, 2016, pp. 1–11.

[5] C. Marghescu, M. Pantazica, A. Brodeala, and P. Svasta, "Simulation of a wireless sensor network using OPNET," in *Proceedings of the IEEE International Symposium for Design and Technology in Electronic Packaging*, 2011, pp. 249–252.

[6] C. Harding, A. Griffiths, and H. Yu, "An interface between MATLAB and OPNET to allow simulation of WNCS with MANETs," in *Proceedings of the International Conference on Networking, Sensing and Control*, 2007, pp. 711–716.

[7] B. Vogel-Heuser *et al.*, "Modeling of networked automation systems for simulation and model checking of time behavior," in *Proceedings of the International Multi-Conference on Systems, Signals & Devices*, 2012, pp. 1–5.

[8] J. Schlick, "Cyber-physical systems in factory automation — towards the 4th industrial revolution," in *Proceedings of the IEEE World Conference on Factory Communication Systems*, 2012.

[9] R. Kurte, Z. Salcic, and K. Wang, "A system level simulator for heterogeneous wireless sensor and actuator networks," in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2018, pp. 776–783.

[10] A. Chaves *et al.*, "KhronoSim: A platform for complex systems simulation and testing," in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2018, pp. 131–138.

[11] J. Geng *et al.*, "Model-based cosimulation for industrial wireless networks," in *Proceedings of the IEEE International Workshop on Factory Communication Systems*, 2018, pp. 1–10.

[12] S. Lin *et al.*, "The DSPCAD framework for modeling and synthesis of signal processing systems," in *Handbook of Hardware/Software Codesign*, S. Ha and J. Teich, Eds. Springer, 2017, pp. 1–35.

[13] *ns–3 Tutorial, Release ns–3.25*, ns–3 Project, 2016.

[14] E. A. Lee and T. M. Parks, "Dataflow process networks," *Proceedings of the IEEE*, pp. 773–799, May 1995.

[15] W. Plishker, N. Sane, M. Kiemb, K. Anand, and S. S. Bhattacharyya, "Functional DIF for rapid prototyping," in *Proceedings of the International Symposium on Rapid System Prototyping*, 2008, pp. 17–23.

[16] L. E. Miller, "Validation of 802.11a/UWB coexistence simulation," Tech. Rep., 2003.

[17] G. Pei and T. Henderson, "Validation of ns-3 802.11b PHY model," The Boeing Company, Tech. Rep., May 2009.

[18] *The ns–3 Wi-Fi Module Documentation*, ns–3 Project, 2016.