A Flexible Retransmission Policy For Industrial Wireless Sensor Actuator Networks

Ryan Brummet[1], Dolvara Gunatilaka[2], Dhruv Vyas[1], Octav Chipara[1], Chenyang Lu[2]
[1]Department of Computer Science, University of Iowa
[2]Department of Computer Science and Engineering, Washington University in St. Louis

***Abstract*—Real-time and reliable communication is essential for industrial wireless sensor-actuator networks. To this end, researchers have proposed a wide range of transmission scheduling techniques. However, these methods usually employ a link-centric policy which allocates a fixed number of retransmissions for each link of a flow. The lack of flexibility of this approach is problematic because failures do not occur uniformly across links and link quality changes over time. In this paper, we propose a flow-centric policy to flexibly and dynamically reallocate retransmissions among the links of a multi-hop flow at runtime. This contribution is complemented by a method for determining the number of retransmissions necessary to achieve a user-specified reliability level under two failures models that capture the common wireless properties of industrial environments. We demonstrate the effectiveness of flow centric policies using empirical evaluations and trace-driven simulations. Testbed experiments indicate a flow-centric policy can provide higher reliability than a link-centric policy because of its flexibility. Trace-driven experiments compare link-centric and flow-centric policies under the two reliability models. Results indicate that when the two approaches are configured to achieve the same reliability level, a flow-centric approach increases the median real-time capacity by as much as 1.42 times and reduces the end-to-end response times by as much as 2.63 times.**

I. Introduction

Wireless sensor actuator networks (WSANs) supported by standards such as WirelessHART[1] and ISA100.11a [2] are becoming an attractive solution for process control industries such as oil refineries, chemical plants, and factories. WSANs provide a low-cost and versatile alternative to wired networks for connecting sensors, actuators, and controllers as part of feedback-control loops. Since communication delays and packet losses may lead to severe degradation of control performance or even control instability, it is critical for wireless solutions to provide reliable and real-time communication while remaining agile in the face of network dynamics. A fundamental challenge faced by WSANs, and the focus of this paper, is to maintain predictable latency and reliability as link quality fluctuates due to the movement of people or heavy equipment, multi-path fading, and electromagnetic interference.

WirelessHART is the state-of-the-art standard for industrial wireless communication and has successfully provided high reliability in a broad range of industrial settings. At the heart of WirelessHART is Time Slotted Channel Hopping (TSCH) — a MAC-layer that uses fixed schedules to specify the slot and channel for each transmission. While researchers have studied algorithms to build fixed schedules extensively (e.g., [3], [4], [5], [6], [7]), one aspect that is often overlooked is how retransmissions are scheduled. The state-of-the-art is a *Link-Centric Policy* (LCP) which allocates a fixed number of retransmissions for each link. At runtime, Automatic Retry reQuest (ARQ) triggers retransmissions in response to failures. The fundamental limitation of this approach is that the network uses a fixed transmission schedule that cannot handle variations in link quality effectively. To illustrate the limitations of this approach, consider a multi-hop flow. The number of retransmissions per link is usually configured based on the worst-case behavior of the link. However, since the links of a multi-hop flow are unlikely to experience their respective worst-case simultaneously when channel hopping is used [8], many of the retransmissions allocated for an individual packet remain unused when it is delivered to the next hop. Since schedules are fixed, it is difficult to reuse these slots to improve the capacity of the network. Similarly, when the quality of a link drops suddenly (e.g., due to the movement of heavy equipment [9], [10]) packets are dropped after the allocated number of retransmissions is exhausted. The likelihood of these failures may be reduced if we could reallocate the unused retransmissions of other links to handle sudden drops in link quality. These examples illustrate how the inflexibility of fixed transmission schedules may reduce the real-time capacity, the reliability of flows, or both.

To address this problem, we propose a transition from using fixed transmission sequences to forward packets to flexible policies that dynamically determine the transmissions that will be performed based on the sequence of transmission successes and failures observed at runtime. Specifically, we make the following contributions:

- We propose a *Flow-Centric Policy* (FCP) that allocates several retransmissions for a real-time flow. In sharp contrast to LCP, FCP dynamically redistributes the allocated retransmissions to different links depending on the success or failure of transmissions observed at runtime. The benefit of FCP is that it can *share and redistribute* retransmissions across links of the same multi-hop flow to handle network dynamics more efficiently.
- We have developed a unified scheduling framework where flows may be executed using either LCP or FCP. Centralized scheduling allows us to consider flows in isolation and to avoid the difficulty of managing the complex interactions between flows while policy-driven adaption

allows us to handle network dynamics efficiently.
- We developed a method for configuring the number of retransmissions required to achieve a user-specified end-to-end reliability when using LCP and FCP under two failure models: the *uniform link failure model* (UFM) and *localized failure model* (LFM). UFM assumes that the quality of all links is a similar and stable. In contrast, LFM is motivated by scenarios that are common in industrial settings where one link may be disproportionately affected by events such as the movement of heavy equipment [9], [10].

We have evaluated FCP and LCP extensively using testbed experiments and realistic simulations. The testbed experiments show that FCP can dynamically reallocate retransmissions to improve reliability while simultaneously reducing the number of slots used to schedule flows. Based on empirical traces, we demonstrate the effectiveness of our approach to configuring the number of retransmissions. Realistic simulations are used to identify when FCP provides superior performance over LCP. Specifically, our simulations compare FCP and LCP and show that FCP provides better performance than LCP in terms of real-time capacity and response time under a wide range of settings. Specifically, under UFM and LFM, FCP improves median real-time capacity by as much as 1.35 and 1.42 times and reduces the worst-case response time by as much as 2.5 and 2.63 times, respectively. Our results indicate that FCP is particularly effective when links have dynamic failure characteristics common to industrial settings.

II. Related Work

In the following, we will review the state-of-the-art approaches to provide real-time communication and resilience to transient link failures.

Predictability vs. Flexibility: CSMA/CA protocols are extensively used for Media Access Control (MAC) due to their simple implementation and flexibility in handling network dynamics. However, CSMA/CA protocols do not provide predictable performance: it is challenging to ensure latency and reliability bounds using traditional WSAN protocols due to the randomized back-off and localized adaptation mechanisms they employ. In contrast, TDMA protocols provide predictable worst-case performance and support a higher throughput than CSMA/CA under heavy load. As a result, TDMA has become the de facto standard of industrial WSANs with numerous proposed scheduling algorithms in the literature (e.g., [3], [4], [6], [7]). A limitation of these protocols is that they either do not consider retransmissions or provision retransmissions for the worst-case behavior of each link.

Transient Link Failures: Transient link failures are common in wireless networks [11], [12] and even more prevalent in harsh industrial environments [13], [14]. Since such failures occur over short time scales, a light-weight adaptation mechanism is necessary to react quickly. The state-of-the-art is to schedule a fixed number of retransmissions for each link, potentially using different channels. At run-time, ARQ is used to trigger retransmissions in response to failures. Unfortunately, little consideration is usually given to provisioning the right number of retransmissions based on link quality although there has been some work to tune the number of retransmissions based on the burstiness of the links [15]. While this is a step in the right direction, the fundamental problem is that links are treated in isolation and provisioned to handle worst-case behavior. A notable exception is recent work by Yang et al. [16] on scheduling batches of packets that leverages the observation that worst-case link behavior does not usually occur at the same time for all links. Unfortunately, for the more common case when flows generate a single packet every period (i.e., batch size is one), Yang et al.'s approach degenerates to LCP. In this paper, we propose FCP which does not require batches and can dynamically reallocate retransmissions among the links of a multi-hop flow. This mechanism can yield significant improvements in network capacity and reliability by taking advantage of the bursty and dynamic nature of wireless links.

III. System Models

Our network model is based on WirelessHART. Networks consist of a gateway and a set of field devices that include sensors and actuators. Controllers are modeled as being centrally located at the gateway. All nodes are equipped with half-duplex 802.15.4 compatible radios. Consistent with the WirelessHART standard, all nodes are time-synchronized with a slot duration of 10 ms, support 16 communication channels, and respond with an acknowledgment (ack) upon a successful packet reception.

We adopt *real-time flows* as a communication primitive. A real-time flow i is characterized by the following parameters: phase ϕ_i, period P_i, deadline D_i, end-to-end (i.e. source to destination) reliability requirement, path Π_i, and fixed priority. The kth instance of flow i, $J_{i,k}$, is released at time $r_{i,k} = \phi_i + k*P_i$ and has an absolute deadline $d_{i,k} = r_{i,k}+D_i$. We assume that $D_i \leq P_i$. A flow i has a source V_{0_i} and a destination V_{N_i} with path $\Pi_i = \{(V_{0_i}V_{1_i}), ..., (V_{N_{i-1}}V_{N_i})\}$. The notation (V_aV_b) denotes a transmission from V_a to V_b. We will also use the notation $(V_aV_b)_R$ to indicate that R transmissions are scheduled over (V_aV_b). The workload L of a network is a list of flows that are ordered based on their priority (lower index values indicate higher priority). Priorities are assigned deadline monotonically, i.e., flows with shorter deadlines have higher priority. Path lengths are used to break ties such that flows with longer paths have higher priority.

The centralized scheduler determines the slot and the channel for each transmission. The schedule is represented by a *scheduling matrix*. A schedule is feasible if it meets the following constraints: (1) Each node transmits or receives only once in a time slot and on a single channel. (2) Hop-by-hop forwarding constraints are maintained such that senders receive packets before forwarding them. (3) No more than one flow transmits or receives on each channel in a time slot. (4) Each flow instance meets its respective deadline and reliability constraints.

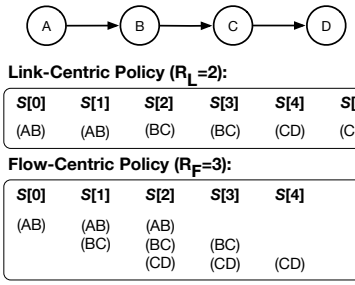


Fig. 1: Example plans for LCP (R_L = 2) and FCP (R_F = 3). Note that R_L = 2 corresponds to WirelessHART's retransmission policy under source routing.

IV. Design

We will create a transmission scheduling framework that supports both LCP and FCP. First, we consider the problem of *scheduling the transmissions of a single flow instance in isolation*. An offline planner constructs *plans* that are used to execute each flow instance. A plan is a sequence of steps that specifies the transmission sequences that may be used to forward a flow's packets from its source to its destination. The LCP planner assigns a single transmission in each step of a plan. In contrast, the FCP planner may assign multiple transmissions in the same step allowing FCP to deliver a packet using multiple sequences of transmissions. FCP uses a lightweight local adaptation mechanism to determine which transmission to execute based on the successes and failures experienced by a packet at runtime. We will develop an analytical technique to configure the retransmissions required by LCP and FCP to achieve a user-specified reliability for two reliability models that capture the loss characteristics of industrial WSANs.

Next, we consider the problem of *scheduling multiple flow instances concurrently*. The scheduler creates a static schedule that specifies the slot and channel used to execute the transmissions of each step in the plan of a flow instance. The scheduler ensures there will be no transmission conflicts between instances as the adaptation mechanism dynamically allocates retransmissions without coordination. The scheduler works identically for both LCP and FCP since it operates at the level of plans that abstract the differences between LCP and FCP. LCP uses ARQ as its runtime adaptation mechanism while FCP includes the proposed adaptation mechanism that dynamically reallocates retransmissions.

A. Handling a Single Real-time Flow

1) Planning: A *plan* is a sequence of steps consisting of transmissions (and retransmissions) that execute an instance of a flow. By executing a plan, the packet associated with a flow instance is delivered from its source to its destination. In the following, we focus on the construction of plans for LCP and FCP, and defer the discussion of how to configure the number of retransmissions to achieve a user-specified reliability to Section IV-B.

LCP Planner: The LCP planner constructs plans that have the following properties: (1) Each link is assigned to transmit R_L times. (2) Each step of a plan contains a single transmission. (3) The order of transmissions dictated by the plan's steps respect hop-by-hop forwarding constraints: if node V_b is closer to the destination than node V_a, then all transmissions assigned to V_a must be assigned earlier than the transmissions assigned to V_b.

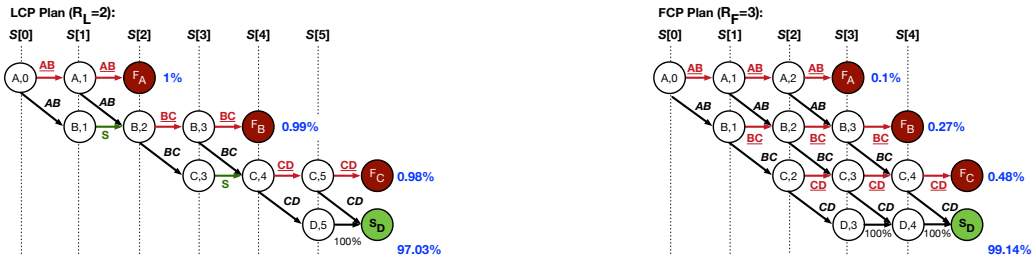
An example of an LCP plan for a 3-hop topology with $R_L = 2$ is shown in the top of Figure 1. In each step, a single transmission is assigned. The precedence constraints introduced by hop-by-hop forwarding are respected: node A transmits in an earlier step than node B. At run-time, the LCP plan is executed using ARQ. Accordingly, node B waits until slot $\mathcal{S}[2]$ before executing transmission (BC). If the transmission is unsuccessful, B will retransmit the packet in the next step $\mathcal{S}[3]$; otherwise, it will sleep in $\mathcal{S}[3]$.

FCP Planner: An effective FCP plan must allow for many sequences of transmissions to forward a flow's packets. Accordingly, an FCP plan must meet the following relaxed constraints: (1) Each link is assigned to transmit R_F times. (2) One or more, possibly conflicting, transmissions are assigned in each step. (3) Hop-by-hop forwarding constraints are relaxed to allow packets to follow different transmission sequences: if node V_b is closer to the destination than V_a, then the s^{th} transmission assigned to V_a must be earlier than the s^{th} transmission assigned to V_b.

The FCP planner works as follows: The earliest time a node V_a may transmit a packet released by flow i is step a. The planner assigns V_a to transmit in steps $\mathcal{S}[a]$ through $\mathcal{S}[a + R_F - 1]$. Plans are constructed by applying this procedure to each node of each flow except for the destination.

An example of a plan for a 3-hop flow with $R_F = 3$ is shown in the bottom of Figure 1. A step may include multiple transmissions. For example, node C, may receive a packet at the end of step $\mathcal{S}[1]$ after two successful transmissions. Node C is then scheduled to transmit in steps $\mathcal{S}[2]$, $\mathcal{S}[3]$, and $\mathcal{S}[4]$ of the plan. A key property of an FCP plan is that it allows different transmission sequences including $(AB)_1, (BC)_1, (CD)_1$ and $(AB)_2, (BC)_1, (CD)_2$ to forward a packet from A to D. The runtime adaptation mechanism described next dynamically selects which transmissions to use in response to packet losses observed at runtime.

2) Runtime Adaptation: We propose a local and lightweight transmission adaptation mechanism to determine dynamically at runtime which transmission to execute in a step depending on the successes and failures of previous packet transmissions. The starting point of our approach is to use the transmission of a packet and its associated ack to trigger subsequent transmissions. Consider a packet of a flow instance that is at node A and has to be forwarded to node D using the route given in Figure 1. Initially, the only node scheduled to transmit is A since it has the packet. After an initial attempted transmission A knows its transmission was successful if it receives an ack and may safely ignore the remaining times the given flow instance has been scheduled. However, if either the transmitted packet or ack is lost A will attempt to retransmit the packet if it is included in the next step of the plan;



(a) Markov Chain for LCP

(b) Markov Chain for FCP

Fig. 2: MCs used to analyze LCP and FCP. Successful and unsuccessful transmissions are shown in black italic and red underlined fonts, respectively. Sleep is indicated by a green S. The error states are F_A, F_B, and F_C and S_D is the successful state. The probability of each final absorbing state is shown in the graph when the likelihood of a successful transmission is 90%.

otherwise, the packet will be dropped. While in most cases this approach works correctly, it is possible for collisions to occur when nodes disagree about the success of a transmission. To illustrate this scenario let us return to the example of A forwarding a packet to D. If B receives the packet after A's initial transmission, but its ack is lost, A will incorrectly believe that B does not have the packet. In the next slot, A and B may both transmit corrupting both packets. To handle this issue we prioritize transmissions closer to the destination. This is possible by having the node that just received a packet (B in our example) transmit earlier in the next slot than a node that performs a retransmission (A in our example). A will monitor the channel, and if it detects B's transmission, it will cancel its retransmission.

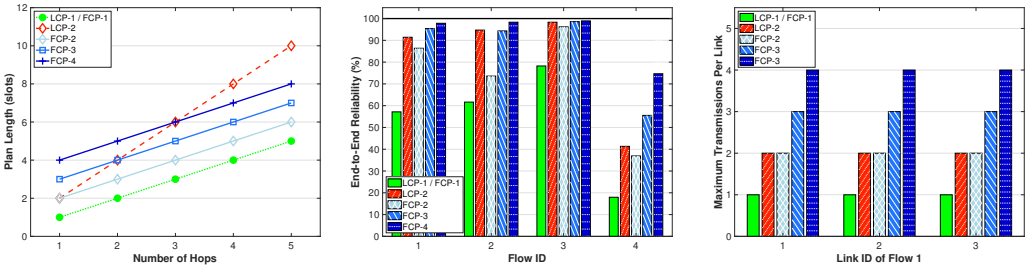
The combination of off-line planning and runtime adaptation enables FCP to dynamically reallocate retransmissions without generating new schedules and provides tolerance to a wider range of failure scenarios than LCP. Continuing with our previous example consider a packet released at A utilizing the FCP plan given at the bottom Figure 1 to reach D. Because of network dynamics, two retransmissions may be necessary to transmit a packet successfully across (BC) resulting in the transmission sequence $(AB)_1, (BC)_3, (CD)_1$. For the next instance, the quality of link (AB) may degrade significantly while the quality of link (BC) may improve. To handle this sudden change, the transmission sequence $(AB)_3, (BC)_1, (CD)_1$ may be used. By dynamically redistributing retransmissions, FCP can handle a wide range of variations in link quality without having to reconstruct the global schedule. In these examples, LCP would drop the packet since retransmissions cannot be redistributed (since $R_L = 2$).

B. Configuring Retransmissions

We use a linear search procedure to configure the number of retransmissions (i.e., the parameters R_L and R_F) necessary to meet a specified end-to-end reliability. Starting with one retransmission, we generate a plan for a given flow and evaluate the reliability provided by the constructed plan. We then increment the number of retransmissions until a plan that meets the flow's end-to-end reliability constraint is found. The open question, therefore, is how to evaluate the likelihood of delivering a packet to its destination successfully after executing a plan for different reliability models.

The execution of a plan can be modeled as a Markov Chain (MC) whose states encode the likelihood that a node has received a packet after executing t steps of a plan. To ground our discussion, consider the forwarding of a packet over a 3-hop flow using either LCP or FCP (see Figure 2). Both protocols can be modeled using MCs with similar structure. An MC state is a pair (V_a, t) which indicates that node V_a has the packet after at step t of its associated plan. Accordingly, a row of states in the Figure 2 includes states with the same sender. Column t contains the nodes that may be reached after executing the first t steps of a plan. For each step, a node running the adaptation mechanism (ARQ for LCP and our own adaptation mechanism for FCP) will either *transmit successfully*, *transmit unsuccessfully*, or *sleep*. The plan specifies the actions performed by a node which drive the transitions in the MC until an error state or the success state is reached. An error state is reached when the allocated retransmissions are exhausted without delivering the packet to the next hop while the success state is reached when the packet is delivered successfully to its destination. Both the error and success states are absorbing. At runtime, a packet associated with a given MC will be in one of the states of the MC. In contrast, during scheduling, we will not know the state of the MC, but we can determine the likelihood of being in a state.

The precise transitions in the MC model depend on the logic of the protocol. The LCP scheduler considers each link sequentially and allocates R_L possible transmissions per link. LCP schedules a single transmission per step and thus, in any column in the MC, there is only one node executing a transmission. The runtime behavior of LCP is encoded as sequences of transitions in the MC. Initially, the MC is in the state $(A, 0)$. After executing (AB) in step 0, the MC transitions to state $(B, 1)$ if the transmission was successful and to state $(A, 1)$ otherwise. In the state $(B, 1)$, node B must delay since the LCP scheduler does not allocate any transmissions for B until A completes all its possible transmissions at the end of step 2. Subsequent transmission failures and successes will drive the MC, eventually reaching either the error states F_A, F_B, or F_C or the success state S_D. The FCP scheduler assigns multiple



(a) Plan length (b) Reliability for 3-hop flows (c) Flow 1 max used transmissions

Fig. 3: Empirical results comparing the plan lengths, reliability for four 3-hop flows, and the maximum transmissions for flow 1.

transmissions in each step. After executing (AB) in step 0, the MC may be either in the state $(A, 1)$ or $(B, 1)$ depending on whether the transmission was successful. FCP schedules both (AB) and (BC) in the next step leveraging its runtime adaptation mechanism to resolve the conflict. By overlapping multiple transmissions, FCP constructs a shorter schedule and improves the likelihood of delivering packets over LCP.

The probability of entering a state (V_a, t) may be computed under an *uniform link failure model* (UFM) with two properties: First, consecutive packet transmissions are modeled as being independent. Empirical studies have shown that this property holds when channel hopping is used [17], [18]. Second, if the quality of a link is lower bounded by a Minimum Packet Reception Rate, $MPRR$, then the end-to-end reliability is no worse than the success probability computed using $MPRR$ in place of the actual Packet Reception Rate of the link. The end-to-end reliability is the likelihood of reaching the success state and can be computed based on the probability of entering the intermediary states as follows:

$$P(V_a, t) = P(V_{a-1}, t-1) \times MPRR + P(V_a, t-1) \times f_a$$

where $P(V_a, t)$, $P(V_{a-1}, t-1)$ and $P(V_a, t-1)$ are the likelihoods that the MC is in state (V_a, t), $(V_{a-1}, t-1)$ and $(V_a, t-1)$, respectively, and $f_a = 1 - MPRR$ when a is not the destination and $f_a = 1$ when it is. We note that both of our initial assumptions may be relaxed by allowing the transitions in the MC to take different values rather than constraining them to equal the $MPRR$. This additional flexibility allows the MC chain to essentially memorize the short-term burst statistics necessary for modeling links accurately and is the approach we use in our simulations.

An open question is whether our approach can be extended to handle other failure models. A typical failure scenario in real-world deployments is one in which failures are localized to one or a few links rather than widely distributed. For example, when moving a large piece of equipment, the quality of a link may drop significantly while having no impact on the quality of other links. To this end, we propose a *localized failure model* (LFM) that assumes failures to be localized to a single bottleneck link on a flow's path. The quality of bottleneck link is lower bounded by the Single-link Packet Reception Rate ($SPRR$) while the quality of the other links is no worse than $MPRR$. The LFM model allows any of the links of a flow to be the bottleneck link; it is not known which link will be the affected link until runtime.

The LFM model cannot be represented directly as a (time-homogeneous) MC since an MC requires transition probabilities to be constant; the probability of a successful transmission can be either $MPRR$ or $SPRR$. We can address this issue by characterizing the behavior of the network using a set of MCs rather than a single MC. For LFM, the set of MCs can be constructed by iteratively treating each link as the bottleneck link. In the MC, the transitions associated with the bottleneck link will be either $SPRR$ or $1-SPRR$ depending on whether they indicate a success or a failure, respectively. In contrast, all other transitions will be either $MPRR$ or $1 - MPRR$. This procedure will result in a set of MC chains whose cardinality is equal to the number of hops in the flow. In general, the worst-case reliability of the system is lower bounded by the minimum of the end-to-end reliabilities of the chains.

C. Empirical Results for Single Flows

In this subsection, we focus on (1) demonstrating FCP's ability to adapt its retransmissions which results in significant performance improvements over LCP and (2) evaluating the effectiveness of the MC-based approach for configuring retransmissions. We have implemented LCP and FCP on TelosB motes and evaluated both on a 16 node tested deployed at the University of Iowa. The motes run a TSCH schedule based on plans constructed by the LCP and FCP planners configured with different numbers of retransmissions. The motes switch channels in each slot, cycling through channels 11, 12, 13, and 14. Note that we use LCP-X to represent a LCP plan generated with $R_L = X$ transmissions and FCP-X to represent a FCP plan generated with $R_F = X$ transmissions.

Figure 3a plots the length of the plans constructed according to FCP and LCP. The figure shows plans constructed using FCP increase in length more slowly than those constructed using LCP as the route length is increased. More specifically, it shows that LCP plans roughly increase *multiplicatively* with path length while FCP plans increase *additively*. This is because for a flow i, the length of an LCP plan is $R_L \times |\Pi_i|$ while that of an FCP plan length is $R_F + |\Pi_i| - 1$. Note that when $R_L = R_F = 1$ LCP and FCP have the same plans for all path lengths.

Next, we turn to the question of whether FCP can provide similar or higher reliability than LCP using shorter plans. To this end, we have compared the performance of FCP and LCP for four, 3-hop flows when different numbers of retransmissions are used. The reported statistics are obtained by transmitting 100,000 packets for each configuration. Figure 3b

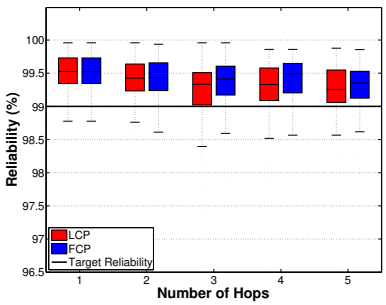


Fig. 4: Testing set plan reliability when configured for 99%.

shows the percentage of packets that reached their destinations. The reliability of FCP-2 averages 8% below that of LCP-2; however, FCP-2 uses 33.3% shorter plans. FCP-3 provides higher reliability than LCP-2 for all flows except for flow 2 by .38%. However, it uses a plan that is 16.7% shorter. Finally, FCP-4 provides the highest reliability and in the case of flow 4 provides nearly double the reliability of LCP-2 at the same plan length. These results indicate that FCP can provide shorter plans than LCP for the same or higher reliability. Additionally, FCP provides finer grained control over LCP allowing application developers to select the most appropriate trade-off for their application.

The key reason behind FCP's improved performance is its ability to reallocate retransmissions depending on the successes and failures observed at run-time. To showcase this mechanism, we have also collected statistics regarding the number of transmissions used. Figure 3c shows the maximum number of transmissions used on each link of flow 1. The figure indicates the LCP-2 uses at most two transmissions for each link. In contrast, FCP-4 allocates up to four transmissions on each link. This result demonstrates that FCP-4 can reallocate its 3 assigned retransmissions dynamically between links as needed explaining why FCP-4 provides higher reliability than LCP-2.

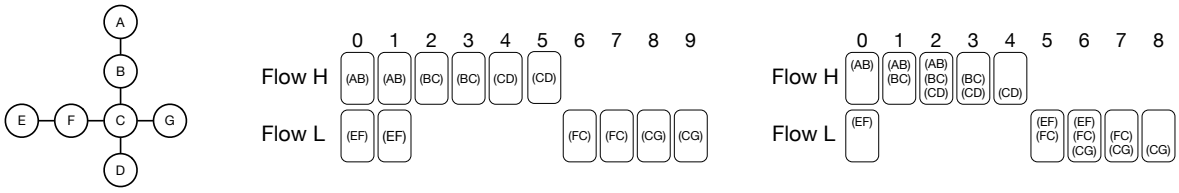
To validate our MC can be used to configure the number of retransmissions needed to meet a specified end-to-end reliability we characterized the properties of 53 links on the 41 node testbed deployed at Washington University in St. Louis by sequentially transmitting 9,800 packets. After each packet transmission the sender and receiver cycle among ten channels synchronously. From these single hop traces, we generated multi-hop traces by randomly assigning each link a trace from our dataset. Our goal was to use the MC model to configure plans that achieve 99% reliability for LCP and FCP. To this end, we divided each multi-hop trace into a training and testing set. The number of retransmissions for LCP and FCP was configured by iteratively increasing the number of retransmissions until a plan was found meeting the target end-to-end reliability. To estimate the end-to-end reliability, we used the training set to determine the transition probabilities of the MC associated with the current plan being considered. The MC is then used to determine the probability of reaching the destination. Figure 4 plots the reliability of plans created via this approach for 100 flows of different lengths when evaluated on the testing set. The box plots indicate that approximately 98% of plans either meet the end-to-end target reliability or are within 0.5%. Overall, LCP and FCP achieve comparable configuration errors. This result indicates that our MC model can be used to tune the retransmissions of FCP and LCP with high accuracy.

D. Handling Multiple Real-time Flows

Thus far we have considered only a single instance of a flow in isolation and shown how plans can be constructed and locally adapted to achieve a specified reliability. Next, we turn our attention to how multiple real-time flows may be scheduled. After plans have been constructed for each flow the scheduler determines the slot and channel to execute the steps of each plan. Specifically, the scheduling process ensures: (1) no transmission conflicts occur even as adaptation mechanisms dynamically reallocate retransmissions and (2) transmissions in consecutive slots use different channels to enforce independence of transmissions pertaining to the same flow instance (as required by the MC models). The primary benefit of separating planning and scheduling is reliability concerns are isolated to the planner allowing us to build general schedules using FCP or LCP under UFM or LFM.

The pseudocode of the scheduler is included as Algorithm 1. During scheduling, the scheduler maintains a $released$ list that contains the flow instances that have been released but have not finished being scheduled. The $released$ list is sorted based on the priority of flows; the release time and a flow identifier are used to break ties. The scheduler maintains a $step$ and a seq variable associated with each flow instance. The $J_{i,k}.step$ captures the next step of instance $J_{i,k}$ that will be scheduled. The $J_{i,k}.seq$ points to a channel hopping sequence as described later in this section.

The scheduler constructs a global schedule by iteratively determining the steps that will execute in each slot. The length of the schedule is given by the hyper-period (i.e., the least common multiple of the flow periods). The scheduler first determines the flow instances that are released in the current slot (see **release_instances** procedure) and then adds them to the $released$ set. Next, the scheduler constructs the list $exec$ that includes the instances which will be executed in the current slot. This requires the scheduler to consider each instance $J_{i,k}$ in the $released$ list iteratively. The instance $J_{i,k}$ is added to $exec$ if it does not conflict with any instance already in $exec$. The **check_conflicts** procedure ensures nodes are not scheduled to transmit or receive multiple times in a slot. This is accomplished by constructing a set B that includes the nodes scheduled in the steps of the instances in $exec$. If the nodes in $\mathcal{S}_i[J_{i,k}.step]$ do not overlap with the nodes in B there are no conflicts. The maximum number of instances in the $exec$ list is $num\_channels$ since at most one instance may be executed on a channel. For each instance $J_{i,k}$ in $exec$ we will assign its step $\mathcal{S}_i[J_{i,k}.step]$ to execute in the current slot. The execution process then increments the step counter of $J_{i,k}$. If all the steps of $J_{i,k}$'s plan have been scheduled $J_{i,k}$ is removed from the $released$ list.



(a) Topology (b) LCP-2 Schedule (c) FCP-3 Schedule

Fig. 5: An example schedule constructed for the topology shown in (a) considering flows H and L. The flows H and L forward packets from A to D and from E to G, respectively. The schedule is constructed for when instances of H and L are released in slot 0. Flow H has higher priority than flow L. The transmissions shown in a box belong to the same step. Each step is assigned a different channel. For clarity, we do not show the actual channel assignments.

Input : $\mathcal{F}$ - list of flows in the network sorted by priority
$\mathcal{S} = \{\mathcal{S}_i\}$ - set of plans for each flow
C - channel matrix
Output: M - scheduling matrix
Data : $released$ - released instances
$exec$ - instances that will be executed
$avail\_seq$ - unassigned rows in the channel matrix

for slot = 0 to $lcm(P_i)$ **do**
  $released = released \bigcup$ **release_instances**(slot, $\mathcal{F}$)
  /* Determine instances to execute in current slot */
  $exec = \emptyset$
  foreach $J_{i,k}$ in $released$ **do**
    $noconflicts$ = **check_conflicts**($J_{i,k}$, $exec$)
    if $noconflicts$ **and** $|exec| < num\_channels$ **then**
      $exec = exec \cup \{J_{i,k}\}$
  $suspended = released \setminus exec$
  /* Give channel hopping sequence to instances that do not have one */
  foreach $J_{i,k}$ in $exec$ **do**
    if $J_{i,k}.seq == -1$ **then**
      assign_channel($J_{i,k}.seq$, $suspended$)
  /* Execute the flow instances in $exec$ */
  foreach $J_{i,k}$ in $exec$ **do**
    $channel = C[J_{i,k}.seq][slot \ \% \ num\_channels]$
    $M[channel][slot] = \mathcal{S}_i[J_{i,k}.step]$
    $J_{i,k}.step = J_{i,k}.step + 1$
    if $J_{i,k}.step == |\mathcal{S}_i|$ **then**
      $released = released \setminus \{J_{i,k}\}$
      $avail\_seq = avail\_seq \cup \{J_{i,k}.seq\}$

Procedure check_conflicts($J_{i,k}$, $exec$)
  B - set of nodes busy transmitting or receiving
  foreach $J_{i',k'}$ in $exec$ **do**
    $B = B \cup links2nodes(\mathcal{S}_{i'}[J_{i',k'}.step])$
  return $links2nodes(\mathcal{S}_i[J_{i,k}.step]) \cap B == \emptyset$

Procedure assign_channel($J_{i,k}$, $suspended$)
  if $|avail\_seq| < num\_channels$ **then**
    $J_{i,k}.seq$ = pick a hopping sequence at random from $avail\_seq$
    $avail\_seq = avail\_seq \setminus \{J_{i,k}.seq\}$
  else
    $J_{i',k'}$ = the lowest priority instance in $suspend$ with a sequence
    $J_{i,k}.seq = J_{i',k'}.seq$
    $J_{i',k'}.seq = -1$

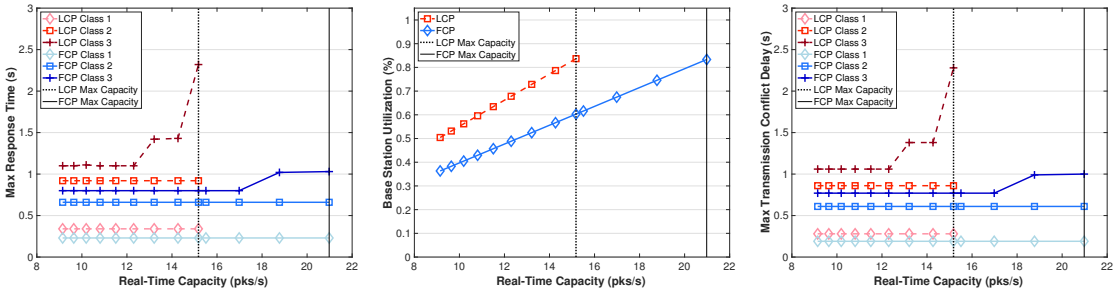
Procedure release_instances(slot, $\mathcal{F}$)
  $R = \emptyset$
  foreach i in $\mathcal{F}$ **do**
    if $slot \ \% \ P_i == \phi_i$ **then**
      $k = (slot - \phi_i)/P_i$
      $J_{i,k}.step = 0$
      $J_{i,k}.seq = -1$
      $R = R \cup \{J_{i,k}\}$
  return R

Algorithm 1: Multi-channel flow scheduler

The FCP and LCP plans are constructed under the assumption that transmissions between consecutive steps are independent. The scheduler must ensure that if two consecutive steps in the plan of an instance are executed in consecutive slots, they will be assigned different channels. We accomplish this goal by creating a channel matrix C that contains channel hopping sequences as rows. The entries of the matrix are generated to ensure two properties: (1) In each column, a channel is assigned only once to ensure executed instances are assigned distinct channels. (2) In each row, adjacent entries have different channels to enforce the requirement that consecutive steps of an instance use different channels.

An instance that is executing uses a channel hopping sequence indicated by $J_{i,k}.seq$ (which points to a row in C). The variable $avail\_seq$ keeps track of which rows in C are currently unassigned. If an executing instance has not been assigned a sequence and there is one available (i.e., $avail\_seq < num\_channels$) the **assign_channel** procedure will assign a sequence from $avail\_seq$ at random. Otherwise, a sequence must be freed up. We select to free up the sequence associated with the lowest priority instance in $suspended$. We select the lowest priority instance because it is likely to be suspended for the longest time. Thus, even if upon its resumption it reuses a channel, its transmissions are unlikely to be correlated. Note that it is easy to see if two consecutive steps are executed in consecutive slots, they will have different channels since flow instances maintain the same sequence until either they enter the $suspended$ set or all steps are scheduled.

Figure 5 shows an example schedule constructed for two flows H and L using LCP-2 and FCP-3. This is the same configuration used in Figures 1 and 2. Let us consider the construction of the FCP-3 schedule shown in Figure 5c. Instances from flows L and H are released in slot 0. Since $\mathcal{S}_H[0] = \{(AB)\}$ and $\mathcal{S}_L[0] = \{(EF)\}$ have no conflicts, the two steps are scheduled concurrently in slot 0. In slot 1, the scheduler considers executing steps $\mathcal{S}_H[1]$ and $\mathcal{S}_L[1]$. However, it can only execute $\mathcal{S}_H[1]$ since $\mathcal{S}_L[1] \cap \mathcal{S}_L[1]$ includes node C. Thus, the scheduler will suspend the instance of the lower priority flow L and execute the high priority instance of H. The scheduler continues to produce the schedule shown in Figure 5c. The schedule produced by the scheduler using LCP-2 is shown in Figure 5b. Besides being longer than the schedule produced by FCP-3 with fewer maximum available retransmissions per link, there is an important difference worth highlighting. The schedule produced with LCP-2 has two slots where the two flows are scheduled concurrently (on different channels) whereas FCP-3 has only one slot where the two flows are scheduled currently. This difference is due to the way the LCP and FCP planners produce plans: the LCP planner limits the number of transmissions per step to



(a) Response time per class (b) Base station utilization (c) Max transmission conflicts

Fig. 6: Impact of the trade-off between plan length and transmission conflict on FCP and LCP.

one whereas the FCP planner includes transmissions for all possible locations of a flow instance. As a result, when the number of transmissions in a step is larger (i.e., a flow has more hops), FCP plans are more likely to have transmission conflicts that reduce concurrency than LCP plans. In our example, FCP-3 makes up for the loss in concurrency by having shorter plans than LCP-2. One of the goals of the experimental section is to understand which one of these two factors dominate: (1) the shorter plans produced FCP or (2) the higher concurrency plans produced by LCP.

V. Experimental Results

In this section, we evaluate FCP and LCP on multiple concurrent flows in order to: (1) understand the trade-off between plan length and concurrency for FCP and LCP, (2) evaluate the real-time capacity of FCP and LCP when configured to achieve the same end-to-end reliability, and (3) empirically evaluate the reliability of FCP and LCP workloads. To accomplish (1) and (2) we have developed a trace-driven simulator that uses a topology and traces collected from the 41-node testbed deployed at Washington University in St. Louis. To accomplish (3), we evaluated LCP and FCP on the 16-node testbed deployed at the University of Iowa.

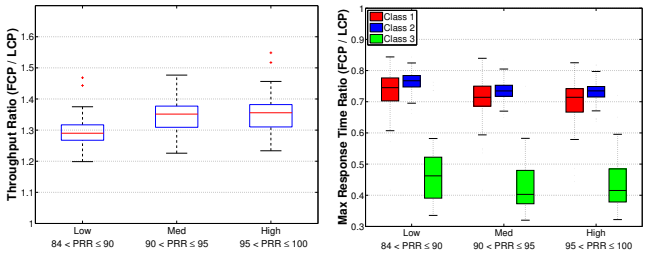
The simulator provides us the flexibility necessary to evaluate the scheduling framework under both UFM and LFM. Accordingly, we classify the reliability of the traces collected from the testbed as *High* ($PRR \in (95, 100]$), *Med* ($PRR \in (90, 95]$), or *Low* ($PRR \in [84, 90]$). We generate UFM workloads by randomly assigning each link in the topology a trace from the *High*, *Med*, or *Low* classes. Under LFM, traces are assigned to links such that all links have *High* reliability except base station links. For these links, we vary the assigned traces from *High* to *Med* and to *Low* reliability to simulate localized failures. Our simulator emulates feedback control loops consisting of sensors, actuators, and a shared controller. The node in the center of the topology is selected as the base station and connects the sensors and actuators to the simulated controller. We randomly select half of the remaining nodes as sensors and the other half to be actuators. We pair sensors and actuators to generate workloads and assign each pair one of three flow classes, whose periods have a 2:3:5 ratio, at random. For example, if the first flow class (*Class 1*) has a period of 200 ms the second flow class (*Class 2*) has a period of 300 ms and the last flow class (*Class 3*) has a period of 500 ms. Deadlines are configured to be equal to periods for all flows. Flow priorities are determined according to deadline monotonic policy such that flow classes with shorter deadlines have higher priority. Within a class, flows with longer routes are given higher priority. Remaining ties are broken arbitrarily.

We quantify the performance of LCP and FCP using *throughput*, *real-time capacity*, *max response time*. Throughput is the rate at which packets arrive at their destinations. Real-time capacity is the maximum throughput that can be achieved without missing deadlines or dropping packets. The latency of a packet pertaining to a flow instance is the time from when the instance is released until it delivered to the destination. Max response time is the maximum latency of all simulated instances of a flow.

A. Transmission Conflicts vs. Shorter Plans

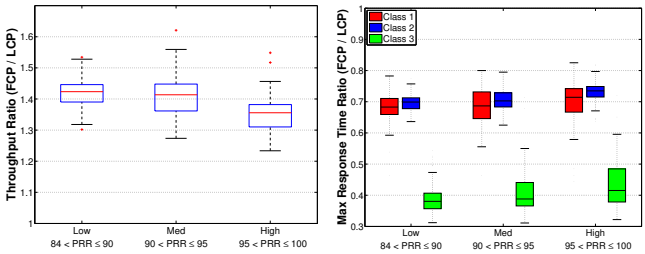
We begin our evaluation by characterizing the performance trade-off between the shorter plans of FCP and the increased channel reuse of LCP. FCP creates shorter plans than LCP by scheduling multiple transmissions in parallel and leveraging runtime adaptation to dynamically select which transmission to execute at runtime. This increased transmission parallelism within individual flows, however, comes at the expense of increased transmission conflict across flows: FCP may not be able to execute the same number of flows concurrently as LCP. We consider a scenario consistent with the UFM model and use *High* reliability traces to simulate a single workload. The number of retransmissions is configured using the MC method to achieve a 99% end-to-end reliability.

Figure 6a plots the maximum response time of the flows pertaining to the same class. The figure shows that FCP achieves significantly lower latency than LCP. For example, at the highest real-time capacity achieved by LCP before deadlines begin to be missed, LCP and FCP provide a worst-case response time of 2.32 s and 0.8 s, respectively, for the lowest priority class. This corresponds approximately to a 65% reduction in worst-case response time. Reductions in worst-case response time are observed for the other flow classes as well. As expected, lower priority classes have longer response times since the scheduler enforces prioritization between flows. As the load is increased by decreasing the period of the flows, the maximum response time of the lowest priority



(a) Throughput ratio (b) Response time ratio

Fig. 7: Throughput and response time improvements under UFM.



(a) Throughput ratio (b) Response time ratio

Fig. 8: Throughput and response time improvements under LFM.

class increases. The vertical lines in the graphs indicate the maximum real-time capacity before any deadline misses. The maximum real-time capacity of LCP and FCP is 15.17 $pkts/s$ and 20.98 $pkts/s$, respectively. Accordingly, FCP has a 38% higher real-time capacity for the given workload.

To understand these results, we plot the network utilization and the maximum delay caused by transmission conflicts. The utilization at the base station is computed as the fraction of entries in the scheduling matrix where the base station was scheduled to transmit or receive and is given in Figure 6b. Since the base station is the only node shared by all flows, its utilization is a good proxy for the real-time capacity of the network. As the workload is increased, FCP utilizes the base station more efficiently resulting in less utilization than LCP. The lower utilization of the base station is the primary reason for FCP outperforming LCP. When the workload utilization reaches approximatively 83%, the response time of LCP and FCP for the lowest priority class exceeds its deadline. We note that both protocols can achieve 100% utilization if the deadlines would be further increased. Figure 6c shows the transmission conflict delay for LCP and FCP as the throughput is increased. As expected, under both LCP and FCP, flows pertaining to a higher priority class have lower transmission conflict delay due to prioritization. We note that the maximum conflict delay for the highest priority class is non-zero because multiple flows are included in that class. More interestingly, FCP introduces *less* transmission conflict delay than LCP. As a result, a lower priority instance is preempted for fewer slots under FCP than under LCP. This result can be explained by accounting for the fact that FCP has shorter plans, so they are less likely to be interrupted. Therefore, the performance improvements of FCP indicate that the reduction in the length of its plans offset its reduced concurrency.

B. Real-time Capacity Under UFM and LFM

In this subsection, we evaluate whether FCP outperforms LCP using a wide range of workloads under the UFM and LFM reliability models. Towards this end we created 100 workloads and evaluated each under UFM and LFM with varying link quality. For each workload retransmissions are provisioned to achieve a minimum end-to-end reliability of 99% for all flows.

Figures 7a and 7b plot the ratio distribution for the 100 workloads under UFM between LCP's and FCP's throughput at maximum real-time capacity and worst-case response time per flow class. The range of the median throughput ratios was relatively consistent across *High* and *Med* link reliabilities at 1.35 and slightly lower at 1.29 for *Low* link reliabilities. The max response time ratio for the two highest priority classes had a median range of 0.71 – 0.76 and the lowest priority class had a median range of 0.4 – 0.46. This represents a reduction in response time between 1.31 – 2.5 times. Under all workloads, FCP had higher throughput and lower max response time than LCP for all flow classes. These results demonstrate that FCP can significantly improve throughput and max response time over LCP under UFM for a wide range of workloads and links with different quality.

Next, we consider the performance of LCP and FCP under LFM by varying the quality of links that use the base station. The results are plotted in Figures 8a and 8b. Note that the UFM and LFM results at *High* link reliabilities are the same. As with UFM, under all workloads, FCP outperformed LCP in terms of throughput and max response time for all flow classes. However, unlike UFM, the improvements of FCP increase with the size of the reduction in the quality of links. Specifically, FCP provides a median throughput ratio of 1.35 and 1.42 for *High* and *Low* reliability localized failure links as opposed to the 1.29 ratio of UFM's *Low* reliability links. Max response time improvement for the lowest priority class was similar with a median ratio of 0.41 and 0.38 for *High* and *Low* reliability links, respectively. This represents reduction in response time between 2.44 – 2.63 times. FCP's increased performance over LCP under LFM is the result of the inflexibility of LCP in the presence of non-uniform link quality drops. In contrast, FCP copes with localized link failures by dynamically reallocating retransmissions allowing it to handle a wide range of failure scenarios with significantly fewer retransmissions than LCP. These results indicate that FCP cannot only outperform LCP when links fail the same way simultaneously but also when link quality tends to be non-uniform. This is a scenario observed in previous studies of industrial plants [9], [10].

C. Testbed Evaluation of Reliability

We evaluated the reliability of LCP and FCP on the University of Iowa tested in a scenario involving 6 flows: one 2-hop flow, two 3-hop flows, two 4-hop flows, and one 5-hop flow. We constructed schedules for LCP-1/FCP-1, LCP-2, LCP-3, FCP-3, and FCP-4 which resulted in schedules of length 12, 24, 36, 24, and 30, respectively. Therefore, the schedule length of FCP-3 is the same as LCP-2 and 33.3% lower than LCP-3,

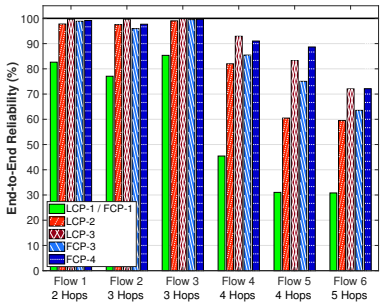


Fig. 9: Empirical workload reliability of LCP and FCP.

and for FCP-4 is 20% higher than LCP-2 and 16.7% lower than LCP-3.

We evaluated the reliability of each schedule by running it for 50,000 packets per flow. The channel hopping sequence is determined by the scheduling algorithm described in Section IV-D. We used the same channels as those from Section IV-C. Figure 9 plots the end-to-end reliability of each flow. As expected, the graph shows that higher reliability is achieved by increasing the number of retransmissions. For example, the reliability of Flow 4 using LCP-1/FCP-1 is 45.44%. LCP-2 uses an additional 4 transmissions (one per each hop) to achieve a reliability of 81.99%. In contrast, FCP-3 achieves a reliability of 85.44% while using only 2 additional transmissions. The increased reliability is the result of FCP's flexibility to dynamically reallocate transmissions at runtime. The saved retransmissions are used to increase the real-time capacity of the network. This trend is observed in all but one of the flows. As discussed in Section IV-B, the number of retransmissions should be tuned using our MC approach to achieved a desirable end-to-end reliability.

VI. Conclusions

A critical challenge of scheduling transmissions for WSANs is to handle dynamics such as those caused by transmission failures. The state of the art technique is LCP which allocates a fixed number of retransmissions to each link. To address the inflexibility of LCP, we have presented FCP which dynamically reallocates retransmissions at runtime. Our solution combines offline planning and scheduling with an online transmission adaptation mechanism. The planning process specifies numerous transmission sequences that may be used to forward a flow's packets and ensures that each packet is reliably delivered. The scheduling process provides prioritization between flows and increases throughput by scheduling multiple flows concurrently. The transmission adaptation mechanism dynamically selects which transmissions to perform based on the successes and failures a packet encounters at runtime. This mechanism is lightweight and requires only local coordination. We developed analytical techniques to configure the number of retransmissions required by LCP and FCP to achieve a user-specified end-to-end reliability.

We have evaluated the performance of LCP and FCP using both empirical measurements and trace-driven simulations. Our empirical results indicate that FCP can provide higher reliability than LCP due its ability to dynamically redistribute retransmissions among the links of a flow at runtime. Additionally, trace driven simulations demonstrate the effectiveness of our MC approach to configuring the number of retransmissions for LCP and FCP. Our simulations evaluate the performance of LCP and FCP under different reliability models when the retransmissions are configured to achieve the same reliability under the two approaches. Experiments indicate that under UFM, FCP improves real-time capacity by 1.29 – 1.35 times and reduces the response time by 1.31 – 2.5 times. FCP provides even higher improvements under LFM by increasing real-time capacity by 1.35 – 1.42 times and reducing the maximum response time by 2.44 – 2.63 times. These results demonstrate the superiority of FCP in a wide range of scenarios common in industrial settings.

Acknowledgement

This work is funded in part by NSF under CNS-1750155.

References

[1] Wirelesshart. [Online]. Available: https://fieldcommgroup.org/
[2] Isa100.11a. [Online]. Available: https://www.isa.org/isa100/
[3] P. Soldati, H. Zhang, and M. Johansson, "Deadline-constrained transmission scheduling and data evacuation in WirelessHART networks," in *ECC*, 2009.
[4] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-time scheduling for WirelessHART networks," in *RTSS*, 2010.
[5] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and real-time communication in industrial wireless mesh networks," in *RTAS*, 2011.
[6] W.-B. Pöttner, H. Seidel, J. Brown, U. Roedig, and L. Wolf, "Constructing schedules for time-critical data delivery in wireless sensor networks," *TOSN*, 2014.
[7] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *SenSys*, 2015.
[8] D. Gunatilaka, M. Sha, and C. Lu, "Impacts of channel selection on industrial wireless sensor-actuator networks," in *INFOCOM*, 2017.
[9] M. Eriksson and T. Olofsson, "On long-term statistical dependences in channel gains for fixed wireless links in factories," *IEEE Transactions on Communications*, 2016.
[10] L. Tang, M. Liu, K.-C. Wang, Y. Huang, F. Yang, and D. Zhang, "Study of path loss and data transmission error of ieee 802.15.4 compliant wireless sensors in small-scale manufacturing environments," *The International Journal of Advanced Manufacturing Technology*, 2012.
[11] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, "The β-factor: measuring wireless link burstiness," in *SenSys*, 2008.
[12] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin, "Temporal properties of low power wireless links: modeling and implications on multi-hop routing," in *MobiHoc*, 2005.
[13] R. Candell, C. A. Remley, J. T. Quimby, D. R. Novotny, A. E. Curtin, P. B. Papazian, G. H. Koepke, J. E. Diener, and M. T. Hany, "Industrial wireless systems: Radio propagation measurements," *Technical Note (NIST TN)-1951*, 2017.
[14] K. Ferens, L. Woo, and W. Kinsner, "Performance of ZigBee networks in the presence of broadband electromagnetic noise," in *CCECE*, 2009.
[15] S. Munir, S. Lin, E. Hoque, S. Nirjon, J. A. Stankovic, and K. Whitehouse, "Addressing burstiness for reliable communication and latency bound generation in wireless sensor networks," in *IPSN*, 2010.
[16] H.-T. Yang, K. S. Liu, J. Gao, S. Lin, S. Munir, K. Whitehouse, and J. Stankovic, "Reliable stream scheduling with minimum latency for wireless sensor networks," in *SECON*, 2017.
[17] O. D. Incel, "A survey on multi-channel communication in wireless sensor networks," *Computer Networks*, 2011.
[18] A. Gonga, O. Landsiedel, P. Soldati, and M. Johansson, "Revisiting multi-channel communication to mitigate interference and link dynamics in wireless sensor networks," in *ICDCS*, 2012.