Packet Priority Assignment for Wireless Control Systems of Multiple Physical Systems[☆]

Wenchen Wang[a,*], Daniel Mosse[a], Alessandro Vittorio Papadopoulos[b]

[a] *Computer Science Department, University of Pittsburgh, Pittsburgh, PA, USA*
[b] *School of Innovation, Design and Engineering (IDT), Mälardalen University, Västerås, Sweden*

Abstract

Wireless control systems (WCSs) have gained much attention lately, due to their easy deployment and flexibility compared to wired control systems. However, this comes at the cost of possibly increased network delay and packet losses, that can significantly impact the control system performance, and possibly its stability. Such problems become even more relevant if the network is shared among different control systems, and thus becomes a scarce resource, like in Industrial Internet of Things applications.

In this paper, we aim at minimizing the performance degradation of the WCS with multiple physical systems sharing a network. We propose a dynamic packet scheduling solution to minimize the performance error of WCS, by dynamically determining the packet priorities of different control systems and characteristic of network paths. We consider two cases for network path selection: (1) network delay only by developing a worst-case end-to-end delay analysis; (2) network delay + reliability by proposing a new network quality model including both delay and packet losses, both of which are very important for the quality of output of the WCSs. Our solution is evaluated over two different use cases to show the generality of the approach: the WCS for a set of inverted pendula,

[☆]This work was partially supported by the Swedish Foundation for Strategic Research under the project "Future factories in the cloud (FiC)" with grant number GMT14-0032.

[*]Corresponding author

Email addresses: `wew50@pitt.edu` (Wenchen Wang), `mosse@pitt.edu` (Daniel Mosse), `alessandro.papadopoulos@mdh.se` (Alessandro Vittorio Papadopoulos)

and the WCS for small modular reactors in a nuclear power plant. The results show that the proposed approach allows for a more stable performance even in presence of highly nonlinear systems, sensitive to time-varying delays, as well as in presence of high network interference.

Keywords: Wireless Control System, Dynamic Priority Assignment, Real-Time Control Systems, Cyber-Physical System

1. Introduction

Networked and wireless control systems (WCSs) have been a major topic of research across different communities in the last decade [1, 2, 3, 4]. A WCS comprises controllers, sensors, relay nodes, and actuators connected via a wireless network. Such a deployment poses significant challenges in the control system design, in the network deployment and management, and in the implementation of suitable protocols that allow for a predictable behavior of the controlled system, in particular in multi-hop deployments [5, 6, 7, 8]. In fact, the network-induced imperfections, such as network delay and packet losses degrade the control system performance, especially during transients of the physical system.

Prior research [6, 9, 8, 10, 11] focused on the impact of the WCSs when controlling a single physical system. To the best of our knowledge, this is the first study on wireless real-time control system with multiple physical systems. The relevance of this problem is motivated by the increasing number of IoT (Internet of Things) and IIoT (Industrial IoT) systems that need to control different functionalities over a shared multi-hop wireless network [12, 13, 14, 15]. This calls for novel solutions able to cope with the design and implementation of WCSs for multiple physical systems.

This paper presents a cyber-physical system study on WCSs with one shared wireless network and multiple physical systems. When the control system requests changes in physical systems behaviors (e.g., changing setpoints over different time frames), the network imperfections will impact each control system differently, depending on the application demands.

To reduce the effect of network-induced imperfections on the overall control system RMSE (root mean square error), we propose an approach to dynamically assign priorities to packets of the different physical systems, choosing the appropriate network paths. We propose an approach to dynamically schedule measurement packets of different physical systems to the appropriate network paths (with redundancy or not) using a TDMA approach for both measurements and actuation packets. Our approach has two parts: (1) priority assignment of the measurement packets (highest priority for most urgent physical plant); (2) network path selection. For the second part, we consider two cases: (2a) **network has no packet losses.** We came up with an end-to-end delay analysis for network paths in a general case when it is possible the network deadline is greater than its period. We assign the highest priority packets to the fastest network path. To the best of our knowledge, this is the first paper that discusses the end-to-end delay analysis for network deadline greater than the control sampling period in the real-time WCS with traffic in both directions. (2b) **network with packet losses.** We propose a network path quality model to combine the impact of network delay and packet loss on the control systems together. Quality here is from the perspective of the control system: higher quality brings higher performance to the control system, which fills the gap between network imperfection and control system performance. The highest priority packet is assigned to the highest quality path.

To evaluate our approach, we first show how our analysis can determine the worst-case end-to-end delay on the TDMA network with multiple paths, deadlines longer than periods, and traffic in two directions. Then, given that different WCSs will have different behaviors, to show our approach is general, we evaluated the effect of our proposed network control over two very distinct use cases: (a) a set of inverted pendula, and (b) a set of small modular reactors in a nuclear power plant.

In particular, the contributions of this paper are the following:

- We propose the first attempt to address for the problem of controlling multiple physical systems over a shared wireless multi-hop network;

- We propose three heuristic methods for the packet priority assignment problem for minimizing the effect of delays and packet losses on the control system performance;

- We develop the end-to-end delay analysis for network paths (redundant and not) using TDMA, when the network deadline is greater than its period;

- We present a general network quality model for WCSs, that includes both end-to-end network delay and packet loss, to choose on which route to send packets of different priorities; and

- We evaluate the proposed approach measuring the RMSE of our approaches for two use cases of WCSs sharing a multi-hop wireless network: a set of inverted pendula, and a set of small modular reactors in a nuclear power plant.

2. Related Work

Although WCSs have several advantages, including an easy deployment and maintenance, one of the biggest challenges is dealing with network-induced imperfections [16]. The solutions of recent research works are typically divided into three categories: (i) control only, (ii) network only, and (iii) control+network co-design solutions.

Control solutions for dealing with network imperfections are promising. In [17], the closed-loop system is modeled and controlled as a switched system, considering both time delays and packet losses at the actuator nodes, and it is sabilized by using an optimal control approach. Other examples include [18, 19, 20] that use a model predictive control approach, which obtains a finite number of future control commands besides the current one for handling both time-varying delays and packet drops. However, these works only consider network as a black box and there is no packet scheduling mechanism taking into account different control system application demands, and they typically focus on a single physical plant to be controlled.

For the network solutions, online dynamic link layer scheduling algorithms have been proposed [21, 22] to meet the deadline of a rhythmic flow and minimize the number of dropped regular packets in a centralized and distributed way, respectively, based on a rhythmic task model proposed in [23]. However, these two works did not consider different control system application demands (i.e., a cruise controller may need to increase speed from 50 kilometers per hour to 60 kilometers per hour within one minute). Also, they assume network external disturbances occur sporadically, which is different from the problem addressed herein. Han et al. [5] propose three types of reliable routing graphs for different communication purposes and generate real-time data link layer communication schedules based on those graphs. Saifullah et al. [24, 25] analyze the worst-case end-to-end delay analysis for source and graph routing based on wirelessHart standard to guarantee the real-time communication in WCS. Alderisi et al. [26] propose a probabilistic scheduling method to provide guarantees on reliable packet delivery in WirelessHART based networks, and they do not account for the current demands of the control systems. In this paper we do not focus neither on real-time network delay analysis nor on proposing new network communication scheduling, since we assume there is a solid network design which contains different routing paths to transmit several periodic control system packets in parallel. We use the network quality model proposed in this paper to choose which routing path is the best for which packet considering control system application demands.

For the co-design solution which is the closest to ours, the integration of wireless network and control systems performance are studied in [27, 6, 9, 28, 8, 29, 30, 31]. The co-design of fault-tolerant wireless network and control in nuclear power plants are studied in [8, 29, 30, 31]. The work in [8] shows that the network delay and reliability both could affect the control system performance. Network reconfiguration schemes to minimize the network-induced error

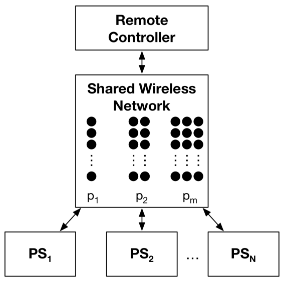


Figure 1: System overview: N physical systems, are connected to a remote controller, via a shared wireless network.

for WCS with a single control system are proposed in [31]. In [6], the authors show how the network reliability affects the control system failure ratio via a water tank case study. In [9], the authors discuss how the routing scheme affects the control system performance. A co-design of network topology conditions and control system stability is explored in [7]. In [27], the author derives a sufficient condition for the random access communication policy of shared wireless medium and design a control-aware random access communication policy. However, there are still three important shortcomings of these approaches. First, there is still a gap to describe the relationship between network performance and control system performance. There is no such a model to describe this gap, and thus we propose a general network quality model to describe this gap in terms of network delay and message loss. Second, these works all discuss wireless control system with single physical system. Third, none of them addressed the cyber-physical aspect of the interaction between dynamic packet scheduling and the control system performance, which is the focus of our work.

3. Problem Formulation

Consider N physical systems (PSs) that share one wireless network as shown in Figure 1. We define a series of time steps $t = \{t_0, t_1, \ldots, t_n\}$ and a set of N reference functions $\mathbf{r}(t) = \{r_1(t), r_2(t), \ldots, r_N(t)\}$. Each reference function corresponds to one PS. Reference functions define the desired behavior of the different PSs over time, such as temperature profiles, trajectories to follow, or other similar setpoint changes over different time frames. In the following, we consider ramp-like reference signals (linearly increasing or decreasing over time, up to a desired value), that can be characterized by: (i) Requested Change Amount (RCA), of the amount a reference quantity must increase/decrease in a linear way, (ii) Requested Change Duration (RCD), or the amount of time in which such change must take place, and (iii) Start Time (ST), or the initial time instant when such change should start. For example, a RCA of 1 unit, RCD of 10 time units, and an ST of 5 time units, means that the controlled variable of the control system must increase linearly by 1 unit, within 10 seconds, starting from time 5 seconds.

Each PS periodically sends out one packet of its measurements to the remote controller at a given frequency. As in [32], there are m choices of network paths/flows $\{p_1, p_2, ..., p_m\}$, each of which has a delay D_j, considered constant over time given that we consider TDMA networks with the fixed topology (i.e., the number of nodes in a path does not change), and a delivery ratio $dr_j(t)$, considered time-varying due to possible network noise and interference (the variation of network noise makes delivery ratio vary over time). Each network path delivers the measurements of one physical system to the remote controller at a time. We assume that if a measurement is lost, the controller will use the last received value for computing the control law. Approaches to optimize the control system behavior in presence of delayed or lost messages are possible, e.g., [33, 34, 35], but this issue is beyond the scope of this paper.

Our objective is to minimize the performance degradation induced by the wireless realization of the network, with respect to a wired realization, over a time horizon T_{trans}, when all the physical systems are experiencing a *transient* induced by a change in their respective reference function $\mathbf{r}(t)$. Therefore, for a PS $i \in \{1, \ldots, N\}$, we compute the Root Mean Squared Error (RMSE) as:

$$RMSE_i = \sqrt{\frac{1}{T_{trans}} \sum_{t=0}^{T_{trans}} \|y_i^W(t) - y_i^{WL}(t)\|^2} \tag{1}$$

where $y_i^W(t)$ is the output vector of system i with a wired network, $y_i^{WL}(t)$ is the output vector of system i with a wireless network, and $\|x\|$ indicates the 2-norm of the vector x.

The overall quality is assessed as the average performance degradation over the N physical systems, computed as

$$RMSE_{\text{tot}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} RMSE_i^2} \tag{2}$$

which must be minimized.

4. Solution Overview

Our approach to minimize the $RMSE_{\text{tot}}$ is to create a dynamic priority assignment for the packets sent through the shared wireless network with multiple paths. Each path has different quality levels, based on its reliability and its induced delay (the path quality is formally defined in Section 8). The proposed solution is composed of two steps. First, the systems are sorted by decreasing application demand at that particular time, and a priority is assigned accordingly. Then, a mapping between the priority of the control system and the path quality is created, that is, the packets associated with the physical system with the highest priority will be sent over the network path with the best quality.

Due to the combinatorial complexity of the problem, an optimal solution cannot be identified unless a brute-force approach is used. Therefore, we propose to solve the problem in two steps. We first propose three *heuristic methods to dynamically assign the priorities* to the N physical systems (Section 5). We then consider two cases: (1) $dr_j(t) = 1$, we develop an analysis of the worst-case end-to-end network delay for each network path and assign the most urgent measurement packet to the network path with the shortest delay (Sections 6 and 7); (2) $dr_j(t) < 1$, we propose a network path quality model to consider both end-to-end delay and reliability of network path. We assign the most urgent measurement packet to the network path that can deliver the measurement with as high reliability and as short delay as needed by the specific physical system to result in small RMSE to the control system (Section 8).

5. Priority Assignment of Measurement Packets

The basic idea of priority assignment of measurement packets is to give high priority to packets of the PS that would yield poor performance if its packets were delayed or lost, that is, to avoid unnecessarily increases in RMSE of each PS, and thus of $RMSE_{\text{tot}}$. To determine the packet priority, we propose three heuristic methods with different perspectives.

5.1. Dynamic Heuristic

To minimize $RMSE_{\text{tot}}$, this heuristic gives the higher priority to the PS with higher RMSE, because it is more necessary for that PS to transmit its message as soon and as reliably as possible (thus reducing the RMSE). Since we cannot get the RMSE of the wired control system at run time, we track the $rRMSE_i(t)$ for the i^{th} PS compared with its reference function $r_i(t)$ for PS$_i$ at run time at each time step, which is computed as:

$$rRMSE_i(t) = \sqrt{\frac{1}{t}\sum_{j=0}^{t} \|r_i(j) - y_i(j)\|^2} \tag{3}$$

where $y_i(j)$ is the measured output for PS$_i$ at time j. At each time step, we calculate $rRMSE_i(t)$, sort the rRMSEs of N PSs and assign the highest priority to the measurement packet of the PS with the highest current rRMSE.

From a computational complexity perspective, (3) can be implemented as a difference equation, to update the $rRMSE_i(t)$ on the previous value $rRMSE_i(t-1)$, and the current reference $r_i(t)$ and the measured output $y_i(t)$. Hence, the complexity is given by the sorting part, i.e., $\mathcal{O}(N \log N)$.

5.2. Static Heuristic

The static approach carries out a thorough offline analysis for all possible parameters, computing $rRMSE_i$ with no message loss for each PS, and assigns priorities before the system starts executing. Priorities are not altered during run-time.

From a computational complexity perspective, the offline phase has a complexity of $\mathcal{O}(N \log N)$, as in the dynamic heuristic, but the runtime behavior is prescribed, and the priorities do not change, thus having a computational complexity of $\mathcal{O}(1)$.

5.3. PID Dynamic Heuristic

The third heuristic is inspired by the widely used PID controller. Unlike in classical PID control [36], where the error is defined as

$$e_i(t) = r_i(t) - y_i(t), \tag{4}$$

we define the tracking error for each PS$_i$ as:

$$e_i(t) = |r_i(t) - y_i(t)| \tag{5}$$

The priority π_i for PS$_i$ follows the PID-inspired dynamic equation

$$\pi_i(t) = K_P \; e_i(t) + K_I \sum_{i=1}^{t} e_i(t) + K_D \; (e_i(t) - e_i(t-1))$$

The first term is the Proportional term (or P-term), and it tracks the latest/current error. The second term is the integral term (or I-term), and it tracks (cumulative) error over time. The last term is the derivative term (or D-term) and it approximates the trend of error in the future (e.g., if this term is negative, it means the tracking error tends to reduce).

The reason to introduce the modified expression of the tracking error as in Equation (5) comes from the fact that a discrepancy between the reference signal r and the system output y must cause the priority to always increase, independently of the sign of such discrepancy. With a classical tracking error as in Equation (4), a discrepancy such that $r < y$ would cause the priority to *decrease*, which is not the objective of the priority assignment mechanism. As a result of such choice, the integral term is a non-decreasing function of time, and cannot forget the past errors in presence of positive errors. Therefore, to tamper the magnitude of the integral term, we select $K_I = \lambda K_P/t$, which divides the value by the timeframe being considered, obtaining the following expression:

$$\pi_i(t) = K_P \left(e_i(t) + \frac{\lambda}{t} \sum_{i=1}^{t} e_i(t) \right) + K_D \; (e_i(t) - e_i(t-1)) \qquad (6)$$

with the interpretation of the new integral term as the average of the error over time. As common in control systems, the parameters K_P, λ and K_D needs to be tuned. Finally, we assign highest priority to the measurement of the physical system with highest $\pi_i(t)$ value at time t.

From a computational complexity perspective, PID dynamic heuristic requires the re-computation of the priorities based on (6), and the highest complexity is provided by the sorting part. Therefore, the overall computational complexity is $\mathcal{O}(N \log N)$.s

6. Network Path Model

Our network path model is shown in Figure 2 (only one network path): there is one primary line of relay nodes (marked as black) and zero or more lines of backup relay nodes (marked as grey). Each line of nodes in our network path model are complete. The relay nodes broadcast messages level by level towards the controller, then back to the actuator. Within each level, the primary node will broadcast first, then the first, second, and third backup nodes, in order. Therefore, the more relay nodes in the network path, the more messages are sent, and thus the higher network delay. Note that the node radio frequency in this paper is the same for all nodes (i.e., no multi-channel transmission). Every control sampling period, we assume there is one message containing all the measurement data sent out via the wireless network to the remote controller, which runs the control algorithm and then sends a message back via the same network again to the actuator. The worst-case end-to-end delay analysis is the worst-case time delay of any one message from the time it is sent out to the time it is received by the actuator. We also assume there is no measurement concatenation for measurements sensed from different time steps.

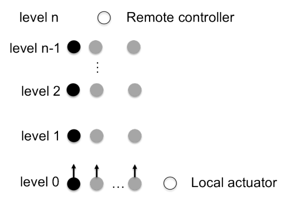


Figure 2: Network path model with one or more lines of relay nodes.

We assume that there are n hops from the sensors to the remote controller and l lines of relay nodes, that is, it takes l time slots at each level to transmit messages (one slot per node). To be reliable, the controller will send out l duplicate messages to the relay nodes (i.e. takes l time slots). We denote the current time slot as t ($t = 0, 1, 2, ...$), the current level as h ($h = 0, 1, ..., n$), and control sampling period as p. The number of time slots during one sampling period is $p_s = \frac{p}{\Delta t}$, where Δt is the duration of the time slot. Thus, with the time delay (i.e., stall time) caused by conflicting with other messages, d_0 (in terms of the number of time slots), message m_0 sent at time $t = 0$ up to the controller is at level $h(m_0) = \lfloor \frac{t-d_0}{l} \rfloor$ $(0 \leq \lfloor \frac{t-d_0}{l} \rfloor < n)$ and the same message on its way down to the actuator is at level $h(m_0) = 2n - \lfloor \frac{t-d_0}{l} \rfloor$ ($n \leq \lfloor \frac{t-d_0}{l} \rfloor \leq 2n-1$). More generally, with the time delay caused by conflicting with other messages, d_i, a message m_i sent out at time $t = ip_s$, ($i = 0, 1, ...$) traveling up is at level $h(m_i) = \left\lfloor \frac{t-d_i-ip_s}{l} \right\rfloor$ $(0 \leq \left\lfloor \frac{t-d_i-ip_s}{l} \right\rfloor < n)$ and traveling down is at level $h(m_i) = 2n - \left\lfloor \frac{t-d_i-ip_s}{l} \right\rfloor$ $(n \leq \left\lfloor \frac{t-d_i-ip_s}{l} \right\rfloor \leq 2n-1)$. We also use $t_c(m_i, m_j)$ to denote the time message m_i starts conflicting with m_j.

7. End-to-end Delay Analysis for Network Path

We want to determine the worst-case end-to-end delay for periodic messages in a general case, when the network delay is greater than the control sampling period. We focus on the delay analysis for fixed priority scheduling where message transmissions are scheduled based on the most recent message first and the oldest message first schemes. We only show our proof based on the most recent message first scheme, given that the derivation for the oldest message first scheme first is symmetric, which will be discussed in Section 10.1. We denote the priority of a message m_i as $pri(m_i)$. The delay without conflicts for transmitting one message up to the remote controller is nl and the same amount of delay for going down. Thus, the delay without conflict is $2nl$. When $2nl \leq p_s$, there will be no conflicts, given that the messages will go up and down before the next message is sent out. When $2nl > p_s$, the current message m_i will conflict with the message m_j with higher priority ($pri(m_i) < pri(m_j)$) and induce more network delay. In this section, we will do the conflict analysis of the message transmission.

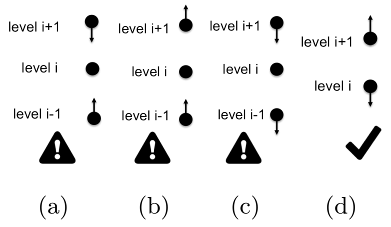


Figure 3: Conflict situation (a) (b) (c) and no conflict situation (d).

A node cannot both transmit and receive in the same time slot and two transmissions that have the same intended receiver interfere each other. If two transmissions are conflicting, they cannot be scheduled in the same slot, which induces more time delay to the lower-priority transmission. Given a set of n, l and p_s, where $2nl > p_s$, there are three canonical situations that two messages will conflict with each other. As usual in wireless networks, conflicts arise when simultaneous transmissions arrive at the same node. The three scenarios are shown as conflict situations 1, 2 and 3 in Figure 3a, 3b and 3c, respectively, for a single line of relay nodes (no backups). Conflict situation 1 shows the scenario when a message going up is at a lower level than the other message going down. Conflict situation 2 and 3 show the scenarios when two messages are going in the same direction but very close together. But for the situation shown in 3d, when the message going up is at a higher level than the message going down, there is no conflict (even though the level difference is 1), since

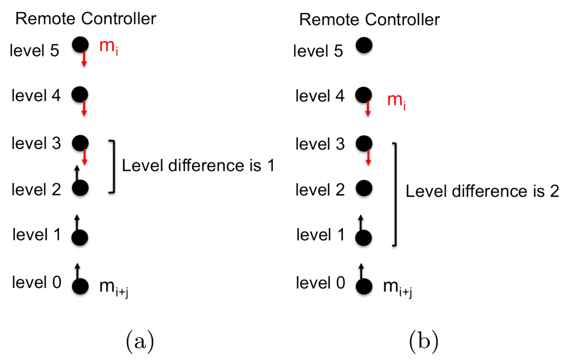


Figure 4: The conflicts of m_i when the level difference with m_{i+j} is 5 (a) and 4 (b)

their receivers are separated apart. Thus, the two messages start to conflict at the level difference, Δh is 1 or 2, only under the three conflict situations. When the $\Delta h \geq 3$, the two messages will not conflict with each other, since it is not possible that one receiver listening to the messages coming from more than one transmitters at the same time.

For conflict situation 1, when the $\Delta h = 1$ and the level of two messages is less or equal to $n-2$, it will take $2l$ time slots to resolve the conflict, given that the high-priority message will go up two levels while the low-priority message waits. At this time the conflict is resolved. Similarly, when the $\Delta h = 2$ and the level of two messages is less or equal to $n-2$, the conflict will be resolved in $3l$ time slots. In general, when message m_i starts going down, the level difference between m_i and m_{i+j}, $\Delta h(m_i, m_{i+j})$ can be odd or even. When $\Delta h(m_i, m_{i+j}) = |h(m_i) - h(m_{i+j})|$ is odd (as shown in Figure 4a), each of the two messages will make progress on one level at a time, until they are separated by exactly 1 level, at which time the conflict happens and will be resolved in $2l$ time slots. Similarly, when $\Delta h(m_i, m_{i+j})$ is even (as shown in Figure 4b), they will make progress until they are separated by exactly 2 levels, at which time the conflict happens and will be resolved in $3l$ time slots. For conflict situations 2 and 3, it will take $4l$ or $5l$ time slots to resolve the conflict, when the level difference is 1 or 2, respectively.

Let us consider consecutive messages, m_0, m_1, m_2, …, m_i that are sent at $t = 0$, $t = p_s$, $t = 2p_s$, ..., $t = ip_s$, respectively. Since we apply the most recent message first message priority scheduling scheme, where $pri(m_0) < pri(m_1) < \ldots < pri(m_i)$. We define level separation of two messages m_i and m_{i+j}, $ls(m_i, m_{i+j})$ as the number of levels m_{i+j} needs to go through to be at the same level and in the same direction of m_i if m_i stays still. The level separation of two consecutive messages before any conflicts is $ls(m_i, m_{i+1}) = \lfloor \frac{p_s}{l} \rfloor$, which describes how separated the two consecutive messages are (in other words, how long in terms of levels to wait to transmit the next sensor value). Intuitively, the more $\lfloor \frac{p_s}{l} \rfloor$, the fewer conflicts will happen. Note that level separation is different from level difference of two messages, when two messages go in opposite directions (e.g., if there are 5 hops in the network and m_i is going down at level 4 and m_j is going up at level 0, $\Delta h(m_i, m_j) = 4$ and $ls(m_i, m_j) = 6$). Recall that if $p_s \leq 2nl$, there will be no conflicts, given that the message will go up and down before the next message is sent out. Thus, three cases under the condition of $2nl > p_s$ are discussed in the following.

1. **Conflict Analysis for Case $3 \leq \lfloor p_s/l \rfloor \leq 2$.**

   Lemma 7.1. *When $\lfloor \frac{p_s}{l} \rfloor \leq 2$, no message can be delivered to the destination.*

   The proof of Lemma 7.1 is reported in Appendix Appendix A.

2. **Conflict Analysis for Case $3 \leq \lfloor p_s/l \rfloor \leq 4$.**

   Lemma 7.2. *When $3 \leq \lfloor \frac{p_s}{l} \rfloor \leq 4$, no message can be delivered to the destination.*

   The proof of Lemma 7.2 is reported in Appendix Appendix B.

3. We consider two cases for $\lfloor \frac{p_s}{l} \rfloor \geq 5$: the case of odd value of $\lfloor \frac{p_s}{l} \rfloor$ in Lemma 7.3 and the case of even value of $\lfloor \frac{p_s}{l} \rfloor$ in Lemma 7.4.

   (a) **$\lfloor \frac{p_s}{l} \rfloor$ is an odd number.**

   Lemma 7.3. *When $\lfloor \frac{p_s}{l} \rfloor \geq 5$, all messages will be delivered, if $\lfloor \frac{p_s}{l} \rfloor$ is odd.*

The proof of Lemma 7.3 is reported in Appendix Appendix C.

(b) $\left\lfloor \frac{p_s}{l} \right\rfloor$ **is an even number.**

Lemma 7.4. *When $\left\lfloor \frac{p_s}{l} \right\rfloor \geq 5$, all messages will be delivered, if $\left\lfloor \frac{p_s}{l} \right\rfloor$ is even.*

The proof of Lemma 7.4 is reported in Appendix Appendix D.

7.1. Worst-case End-to-end Delay Determination

Based on Lemmas 7.1, 7.2, 7.3 and 7.4, we have the message schedulability condition: $\left\lfloor \frac{p_s}{l} \right\rfloor \geq 5$, which is independent of the number of hops n. We assume that a message already conflicted with $(Q-1)$ higher priority messages. The upper-bound of the total stalls is $3l(Q-1)$ (given that each conflict can be resolved in at most $3l$ time slots for conflict situation 1). The following formula shows the difference in levels between m_i and m_{Q+i}, which is $\Delta h(m_i, m_{Q+i})$; if that value is 1 or 2, the Q^{th} conflict will happen: $1 \leq \Delta h(m_i, m_{Q+i}) = 2n - \left\lfloor \frac{t-ip_s-3(Q-1)l}{l} \right\rfloor - \left\lfloor \frac{t-ip_s-Qp_s}{l} \right\rfloor \leq 2$. Based on properties of the floor operation, we can get:

$$1 \leq 2n - \left\lfloor \frac{t-ip_s}{l} \right\rfloor + 3(Q-1) - \left\lfloor \frac{t-ip_s}{l} \right\rfloor + \left\lfloor \frac{Qp_s}{l} \right\rfloor \leq \Delta h(m_i, m_{Q+i}) + 1 \leq 2$$

Then, we get $2n - \left\lfloor \frac{t-ip_s}{l} \right\rfloor + 3(Q-1) - \left\lfloor \frac{t-ip_s}{l} \right\rfloor + \left\lfloor \frac{Qp_s}{l} \right\rfloor = 2$. Therefore, we can get Equation 7.

$$\left\lfloor \frac{t-ip_s}{l} \right\rfloor = n + \frac{3}{2}(Q-1) + \frac{1}{2}\left\lfloor \frac{Qp}{l} \right\rfloor - 1 \tag{7}$$

Since the conflicts happen only when a message is transmitted down, the following condition holds about the level of message m_i: $n \leq \left\lfloor \frac{t-3(Q-1)-ip_s}{l} \right\rfloor \leq 2n-1$, so $n + 3(Q-1) \leq \left\lfloor \frac{t-ip_s}{l} \right\rfloor \leq 2n - 1 + 3(Q-1)$. Put the Equation (7) into the above condition, we get $\frac{l}{3l-p_s} \leq Q \leq \frac{2nl-3l}{p_s-3l}$ and derive the maximum Q as $\left\lfloor \frac{2nl-3l}{p_s-3l} \right\rfloor$. After calculating the maximum number of conflicts, we can estimate the worst-case stalls in terms of the number of time slots caused by conflicts, $D_{conflict} = 3lQ = 3l\left\lfloor \frac{2nl-3l}{p_s-3l} \right\rfloor$. Recalling that the delay without conflict, $D_{pure} = 2nl$, the worst-case end-to-end delay in terms of the number of time slots is $D_{slots} = D_{pure} + D_{conflict} = 2nl + 3l\left\lfloor \frac{2nl-3l}{p_s-3l} \right\rfloor$.

To determine the worst-case end-to-end delay, we multiply D_{slots} by Δt, and obtain
$D^{worst}_{network} = (2nl + 3l\left\lfloor \frac{2nl-3l}{p_s-3l} \right\rfloor)\Delta t$. Note that our derivation is general and scalable to any network topology with n hops and l lines of nodes.

8. Network Path Quality Determination

Recall that in Section 5 we determined the priority of the measurement packets and in Section 7 we calculated the worst-case delay when considering a network without packet losses. Now we need to determine on which network path to transmit those measurements when considering packet losses (i.e., $dr(T) < 1$). Although previous research discussed how the network reliability and network delay affect the control system performance [8, 6], there is still a gap between network performance (i.e., network delay and message loss) and control system performance for problems with several physical systems. Because there is no such a model to describe the gap, we propose a general network quality model. We call it PQModel, include in it both network delay and losses for a path[1], as described by (8) and quantify how much the network imperfections affect the control system.

$$PQ = D_{net} + \alpha n_{loss}\Delta_{csp} \tag{8}$$

where D_{net} is the network end-to-end delay, Δ_{csp} is the control sampling period, α is a non-negative constant, and n_{loss} is the number of consecutive packet losses. Note that n_{loss} is computed based on the control system perspective, in this case the control sampling period, Δ_{csp}. For example, if the network sensing

[1]We omit the subscripts when no confusion arises.

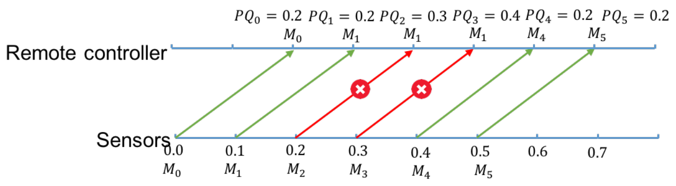


Figure 5: Network delay and delivery ratio tradeoff illustration, when network delay is greater than control sampling period.

sampling period is 0.5s and $\Delta_{csp} = 1.0$s, the network will send two messages during each Δ_{csp}; if the remote controller did not receive neither of the messages, $n_{loss} = 1$. Note that this PQModel quantifies the network imperfection impact to the control system, thus the less PQ value, the better quality the network is.

We use α to adjust the importance between network delay and network reliability. When $\alpha = 1$, network delay and network reliability have the same importance to the control system performance. For example, as shown in Figure 5, the control sampling period and network sampling period are both 0.1s, but when the network delay is 0.2s and measurement M_2 gets lost, PQ_2 is 0.3s because the controller will use measurement M_1 that arrived 0.3s before. If measurement M_3 also gets lost, the induced delay PQ_3 becomes 0.4s because the controller will (re-)use measurement M_1.

α must be tuned according to different control system we are dealing with. When the network delay is smaller than the control system sampling period (e.g., like the water tank system in [6]), α is set to a very large number since network reliability is the only factor to affect the control system performance. When the control sampling period is smaller than the network delay, the α is a number closer to 1. For instance, when the control system uses Kalman filters or any other technique to compensate the message loss reliably, we can reduce the network reliability importance and set α to be small. The value of α also needs to be adjusted under different network situations for the same control system. We will discuss the value of α under different network situations later in Section 10.

9. Case studies

The evaluation of the presented approach considers two different case studies that share the same architecture (as shown in Figure 1). In particular, we consider the case of $N = 3$ physical systems, and of $m = 3$ paths with 6 hops. The WCSs considered include 3 inverted pendula (IP) and 3 Small Modular Reactors (SMR) a Nuclear Power Plant (NPP).

For the wireless network, we use the bitvector protocol [37], which uses TDMA scheduling to guarantee real-time transmission with no contention among WCSs within each time slot. We consider the messages sent from sensors to the controller (measurements), and back from controller to the actuator (control signals). We consider a wireless network with three paths, each of 6 hops: path 1 (p_1) has no backups but the fastest delivery of packets due to no redundancy; path 2 (p_2) has double the number of sensors and thus higher reliability and higher delay, given the messages have to traverse twice as many nodes; path 3 (p_3) has 3 times as many nodes as p_1, with the highest reliability and highest delay. In other words, the reliability relationship of the three paths is $dr(p_1) < dr(p_2) < dr(p_3)$. We assume each network path can transmit messages independently from the others, that is, all 3 paths can transmit messages in parallel, without interfering with each other. We set different duration for the time slot of TDMA scheduling (Δt) for the IP use case and NPP use case, due to different sensitivity of time delay of the two systems. In the case study, $\Delta t = 1$ms for the IP use case and the delay of p_1, p_2 and p_3 is 0.01s, 0.02s and 0.054s respectively; and $\Delta t = 10$ms for the NPP, and the delay of p_1, p_2 and p_3 is 0.1s, 0.2s and 0.54s respectively.

As shown in Figure 6, we combined a state-of-the-art cyber-physical system simulator (WCPS 2.0 [6]) with a NPP simulator to mimic the WCS we consider. Our simulator allows m wireless network paths running together with m PHXs (dark blue block). We implemented the three heuristic methods proposed in Section 5 to assign priority to the measurement packets in the left-bottom block (yellow). We also implement the network quality model from Section 8

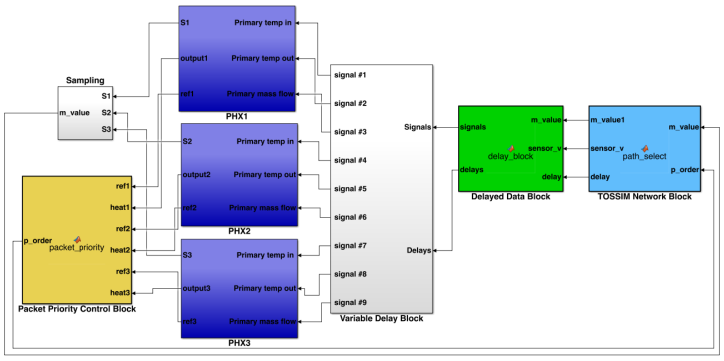


Figure 6: Simulation overview in Simulink, implemented in conjunction with WCPS for the NPP use case. The Simulink architecture is analogous for the IP use case.

to quantify the quality of network paths in the rightmost block. We use the TOSSIM network simulator (embedded in WCPS) with wireless traces from a 21-node subset of the WUSTL Testbed[2]. To evaluate the WCSs under a wide range of wireless conditions (e.g., different levels of noise/interference), similar to [8], we use controlled Received Signal Strength with uniform gaps to simulate various wireless signal strength (RSSI) values to change the quality of network links. Based on [6], we adjust the RSSI values for the average link success ratio (LSR) to be in the range $(0.71, 1.0)$ as shown in Figure 7.

9.1. Inverted Pendula Case Study

Our first use case is typical IIOT application, with three inverted pendula mounted to motorized carts [36, 38], that are controlled from remote wirelessly. This highly nonlinear dynamics are extremely sensitive to delays and losses induced by the network. A single pendulum scheme is represented in Figure 8. The controller receives the inputs of both the angle and the displacement, and applies a force F to the cart in order to keep the inverted pendulum balanced. The objective of the control system is to stabilize the pendulum in the vertical position ($\theta = 0$), and and to maintain the cart in (or move it to) a specific x

[2]`http://wsn.cse.wustl.edu/index.php/Testbed`

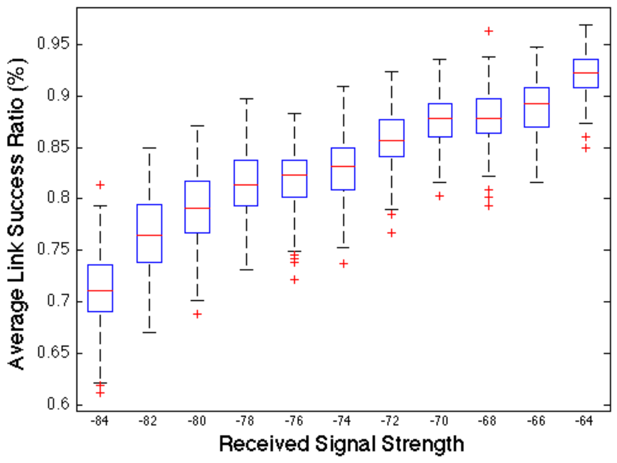


Figure 7: Average link success ratio.

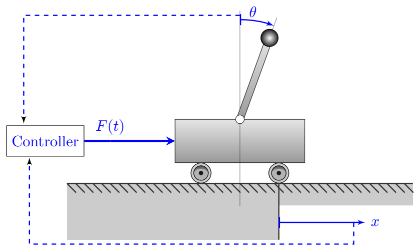


Figure 8: Inverted pendulum.

position, while keeping the pendulum in the vertical position. With a poor controller, the pendulum will fall down. Real-world examples that relates directly to this inverted pendulum system is the attitude control of a booster rocket at takeoff, segways, etc. In particular, several such systems can be deployed in the same environment, for example, segways that deliver packets in a building or in a factory (an IIOT application).

In this use case, we consider a two-dimensional problem where the pendulum is constrained to move in the vertical plane, and the cart has only a degree of freedom to move (back and forth). For this system, the control input is the force F that moves the cart horizontally and the outputs are the angular position of

Table 1: Parameters and values for the simulation of the IP and NPP use cases, with $l = 0, \dots, 4$ and $j = 1, \dots, 8$.

Parameters	IP	NPP
Sampling period T_s	0.01s	0.1s
Simulation time T_{sim}	100s	300s
RCA	$(6 + 4l)$m	$(2 + 2l)$MW
RCD	$5j$ sec	$15j$ sec
ST range	[0s, $T_{\text{sim}}-$RCD]	[0s, $T_{\text{sim}}-$RCD]

the pendulum θ and the horizontal position of the cart x.

In particular, we evaluate this use case over several different scenarios, where the reference signal requires the cart to move of a distance of RCA (while keeping the pendulum stable in the vertical position), within a time interval of RCD. The parameters used for the simulation are specified in Table 1.

9.2. Nuclear Power Plant Case Study

A modern NPP design considers several Small Modular Reactors (SMRs) [39], instead of a single large reactor, due to the flexibility and cost-benefit of starting and stopping SMRs. Typically there is one primary heat exchanger system (PHX) and two secondary heat exchangers (SHX) in each SMR. The PHX in the NPP has its main function as the exchange of heat from inside of the reactor to the outside. The PHX is typically modeled as a nonlinear system. For each PHX, we focus on three measurements that are sent periodically to the controller, namely outlet hot leg temperature, inlet hot leg temperature, and mass flow rate.

We consider a case study of a NPP with three SMRs (three PHXs and six secondary heat exchangers[3]), each of which transmits measurement data via a shared wireless network (we focus on the 9 measurements sent periodically).

[3]We only modeled one PHX, since the two secondary heat exchangers are backups for ssfety.

Given that there are several SMRs in an NPP, the power output of each SMR may differ and the controller may decide to change the power output (reference function) of each SMR dynamically, based on energy requirements and balance the power required to achieve a certain level of power output. The PHXs in SMRs are identical systems except for the reference functions. In reality, the reference function is set by the nuclear engineer/operator based on the NPP requirement. To be general in our case study, the RCA, RCD and ST value of each reference function is randomly chosen by uniform distribution from the range of values listed in Table 1. The parameters in *a set of reference functions* are 3 RCAs, 3 RCDs and 3 STs to set three reference functions. In order to include all the RCDs, we choose simulation time as 300s, taking into account the system settling time (even after the power change duration, the system still needs sometime to settle down to the setpoint). Each PHX will generate one measurement packet (include three measurements: outlet temperature, inlet temperature and mass flow rate) and send out the packet by wireless network periodically at the sampling period 0.1s. Recall that, if the measurement packet is lost during the wireless transmission, the system will use the latest received measurement value in the control algorithm.

10. Quantitative Results from Case Study

In this section, we first evaluate the worst-case end-to-end network delay analysis with the realistic simulation results. Second, we compare the reliability of the three network paths for different network conditions. We present how the different parameters of the presented approach are tuned, and we evaluate our network path quality model. Finally, we compare the three heuristic methods of measurement packets priority assignment with respect to the RMSE of the two use cases.

10.1. Worst-case End-to-end Delay Analysis Validation

To validate our worst-case end-to-end delay analysis, we implement a simulation to simulate the process of the dynamic message transmission. Recall

Table 2: Simulation parameters and values for worst-case end-to-end delay analysis validation

parameters	values
p	0.05s, 0.1s, 0.15s, 0.2s, 0.25s, 0.3s
p_s	5, 10, 15, 20, 25, 30
l	1, 2, 3, 4
n	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

that the schedulability condition ($\lfloor \frac{p_s}{l} \rfloor \geq 5$) is to determine whether a message can be delivered to the destination within a limited amount of time. We carried out a set of tests on our simulation with different values of p, l and n, as shown in Table 2, where each test corresponds to a value of p, l and n. Our test set can be divided into two test sets, *test set 1*, where all the tests meet the condition and *test set 2* where all the tests do not meet the condition. We test the schedulability condition on the test set and calculate the test accuracy by summing the percentage of the tests that can deliver the messages within a limited amount of time for test set 1 and the percentage of the tests that cannot deliver the messages within a limited amount of time for test set 2. We get 100% accuracy for the schedulability condition test, demonstrating the correctness of our schedulability condition. Under the test set 2, the worst-case delay analysis overestimates the delay by 4.2% compared with the realistic simulation results (always pessimistic, but a very tight pessimism).

Figure 9 shows the examples of message transmission process of the most recent message first (Figure 9a) and the oldest message first scheduling schemes (Figure 9b) with $p = 0.1s$, $p_s = 10$, $l = 2$ and $n = 10$. For the most recent message first scheme, as discussed above and shown in Figure 9a, the lower priority messages conflict with higher priority messages and are delayed when traveling "down". As analyzed in Section 7, when a message traveling down, it is delayed at every level starting with level $n - 2$ (level 8 in this example) but can still move down by 1 level until it reaches to the destination. Regardless, they still arrive at the controller within the deadlines, because the condition

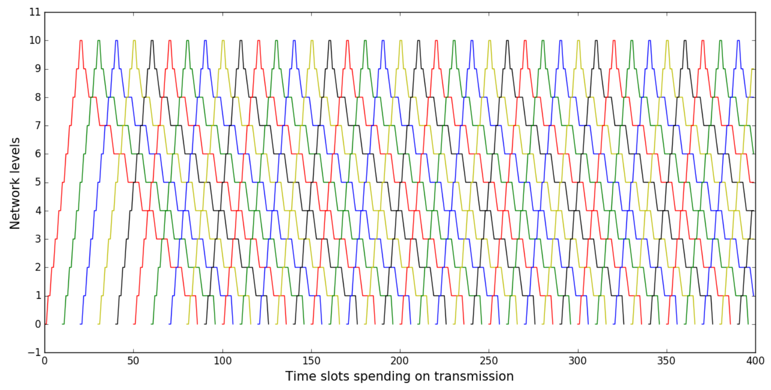
$\lfloor \frac{p_s}{l} \rfloor \geq 5$ is satisfied.

For the oldest message first, as shown in Figure 9b, the conflicts happen when lower priority messages (later messages) traveling up. The message transmissions are *unsteady* (i.e., the end-to-end delays of the messages are not the same) at first, given that there are not many higher priority messages ahead, so the delay is less for the earlier than for the later (lower priority) messages. The transmissions get *steady* (The steady state of message delays, that is, the end-to-end delays of the messages are the same.) after the 275 time slots and the transmission process is symmetric with the most recent message first scheme. The proof process for the oldest message first scheme is exactly the same as the most recent message first scheme, that is, starting with the first two lowest priority messages, which are the last two messages for the oldest message first scheme. Note that if the schedulability condition is not met, the oldest message first scheme will be always unsteady, but can still delivery messages to the destination; this is a significant difference from the most recent message first scheme. However, the end-to-end network delay is unbounded (becomes larger and larger), since more and more conflicts are accumulated and are unresolved.

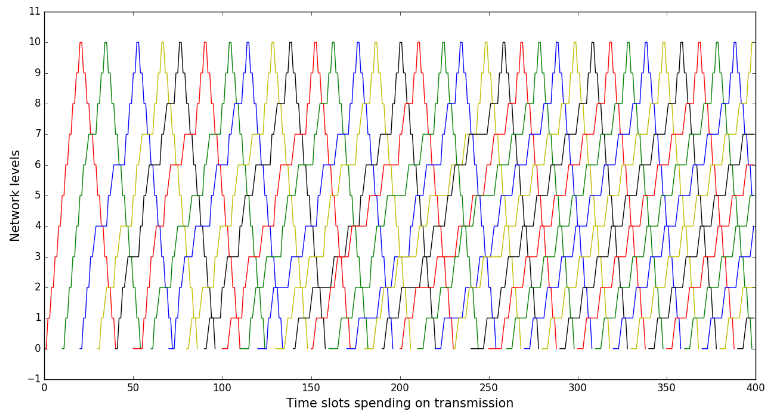
10.2. Network reliability results

Figure 10 shows the delivery ratio of three six-hop network paths under different RSSI values (-60dBm is a network with very little interference, and -84dBm is a very poor network). The delivery ratio increases as the number of backup paths increases. Since p_1 has no backup path, the delivery ratio is already about 0.4 when RSSI value is -64dBm. The delivery of p_3 still holds at 0.8 when RSSI $= -84$dBm, because p_3 has three backup paths.

The percentage of the number of consecutive packet losses for paths p_1, p_2 and p_3 are presented in Figure 11. As expected, for the same network interference condition, the higher the number of backup paths in the network, the fewer number of consecutive losses; the interesting aspect is the quantities shown in the figure. For each network path, as interference in the network increases (less RSSI value), the percentage of massive consecutive losses ($n_{loss} \geq 6$) in-



(a)



(b)

Figure 9: Examples of (a) the most recent message first scheme and (b) the oldest message first scheme transmission process with $p = 0.1s$, $p_s = 10$, $l = 2$ and $n = 10$. Note that the symmetry of the oldest message first scheduling scheme with the most recent message first scheduling scheme begins at the 275^{th} time slots.

creases (e.g., topmost region is much larger for -84dBm than for -60dBm in Figure 11a). Figure 11c has 0 for -60dBm since no message is lost.

10.3. Control system results

10.3.1. Parameters tuning

A tuning phase is required only for the PID heuristic, and for deciding what is the best value of α of the PQModel in Equation (8). In this subsection we explain how the tuning is done.

Tuning the PID heuristic. To tune the three parameters of the PID heuristics, i.e., K_P, λ, and K_D of (6), we consider the case where three physical systems are controlled over the network, and the RSSI is -60dBm, as a representative case because it is the best network condition. We select the values of the parameters that minimize the $RMSE_{\text{tot}}$. In particular, we considered the parameters in the following ranges: $K_P \in [-4, 15]$, $\lambda \in [-4, 15]$, and $K_D \in [-4, 15]$. The numerical results of the tuning process for the two use cases is presented in Figure 12. The selected values that minimize the $RMSE_{\text{tot}}$ are therefore $K_P = 1$, $\lambda = -1$, and $K_D = 9$ for the IP case, and $K_P = 1$, $\lambda = 2$, and $K_D = 0$ for the NPP case. Figure 12 provides a sensitivity analysis of the impact on the control system performance of different choices of control parameters, highlighting that the choice of K_D has little impact on the RMSE value, as well as choices of positive

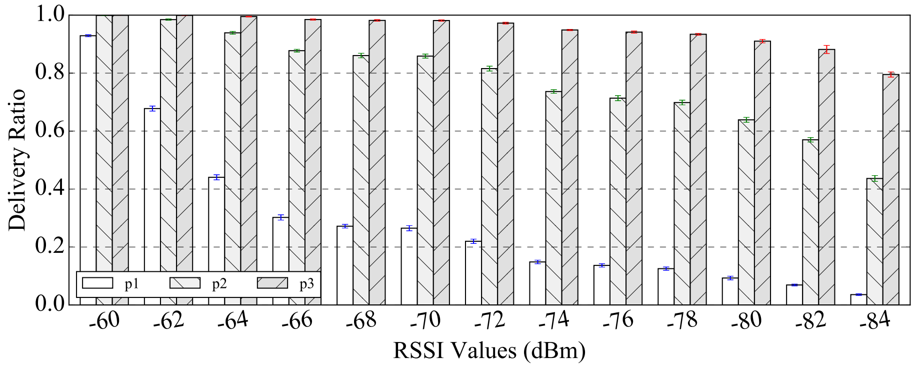
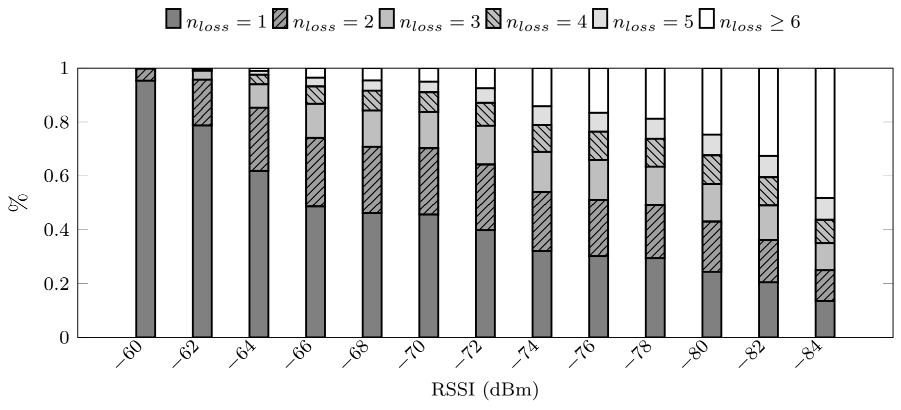
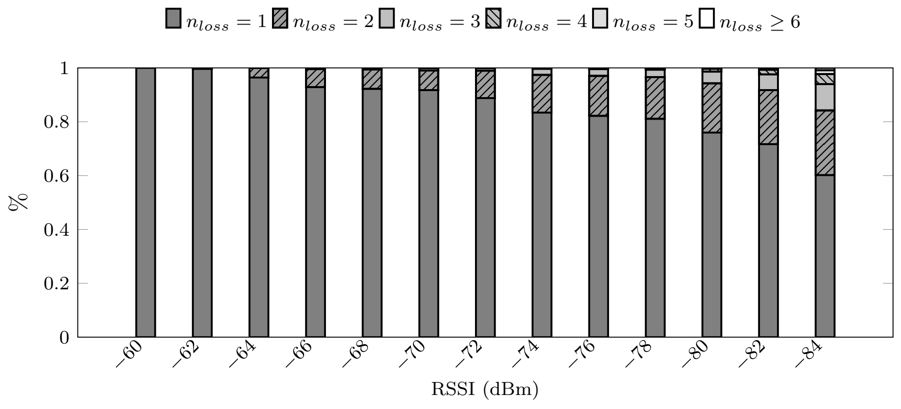


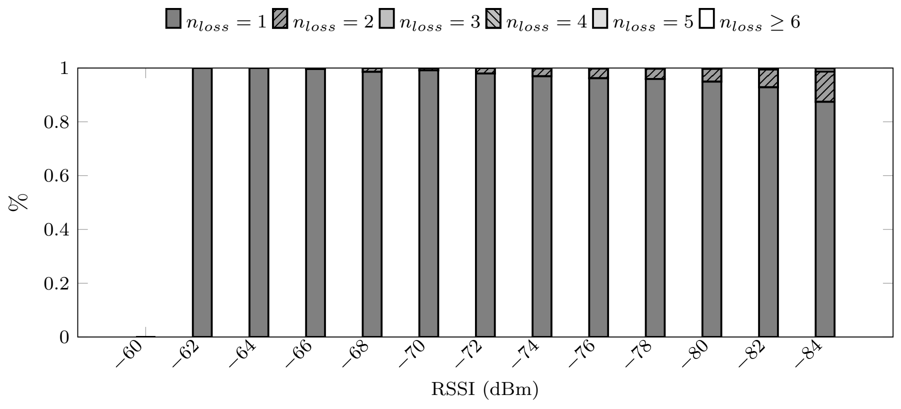
Figure 10: Delivery ratio of three network paths under different RSSI values.



(a) Path 1



(b) Path 2



(c) Path 3

Figure 11: Percentage of consecutive losses (n_{loss}) for three network path at different values of RSSI.

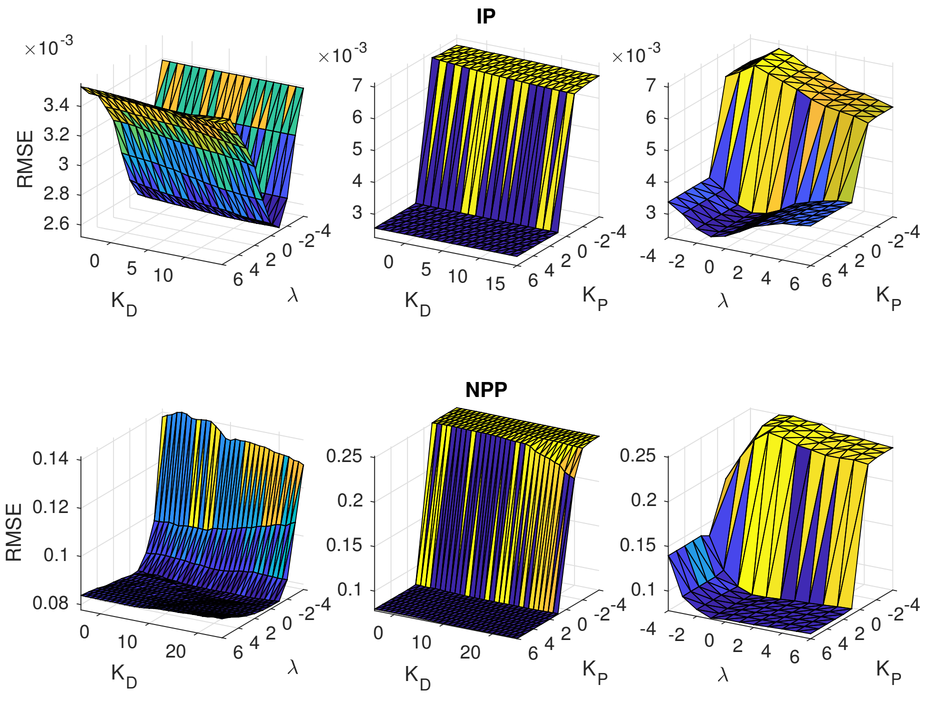


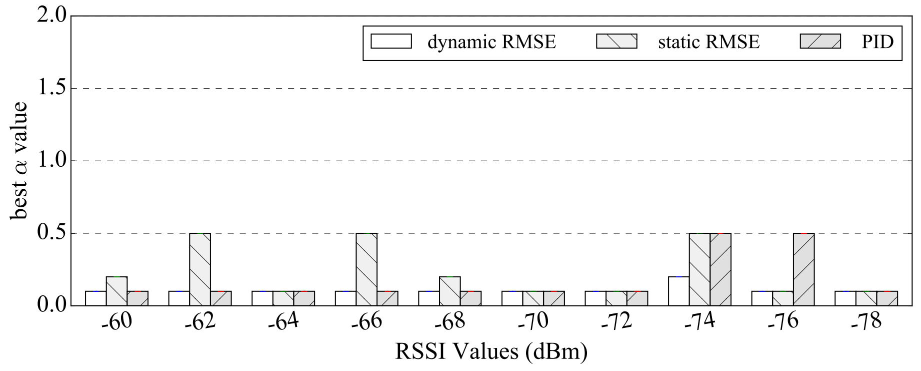
Figure 12: RMSE computed for the tuning of the PID heuristic.

values of K_P. Such analysis suggests that the choice of λ is more critical, since the RMSE value is more sensitive to deviation from its best value.

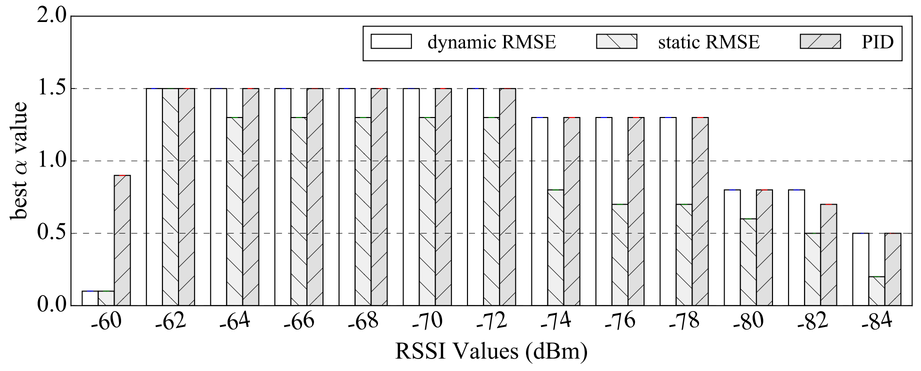
Tuning the value of α. To evaluate the network quality model proposed in Section 8, we run experiments with different α values from 0.0 to 2.0 for the three heuristic methods proposed in Section 5 over different RSSI values on 20 sets of reference functions. Each experiment runs 20 times on the network paths given the RSSI value. Figure 13 shows the value of α that minimizes $RMSE_{\text{tot}}$ for different values of RSSI. What is interesting to notice is that the values of α for the pendulum case are much lower than the ones for the NPP case. This highlights that the pendulum use case is much more sensitive to large delays.

For the pendulum case, we can conclude that almost independently of the value of the RSSI, a low value of α must be selected.

On the other hand, for the NPP case, in all three heuristic methods, the



(a) IP



(b) NPP

Figure 13: Best values of α as a function of RSSI.

value of α decreases as the interference in the network increases. To figure out the reason, we counted the average number of time steps that a path quality order is selected for the cases of the best α values compared with when $\alpha = 1.0$ for each RSSI value and each heuristic method. Table 3 shows the examples of RSSI -64dBm and -84dBm for heuristic method 1. Our method is as follows. The path order number column in the table shows the quality order of the three network paths. For example, 123 means the the quality order is $p_1 > p_2 > p_3$; in that case, the highest priority packet will be sent via p_1 path and lowest priority packet will be sent via p_3 path. By comparing the average $RMSE_{\text{tot}}$, we can see that the system with $\alpha = 1.9$, RSSI $= -64$dBm and $\alpha = 0.2$, RSSI $= -84$dBm

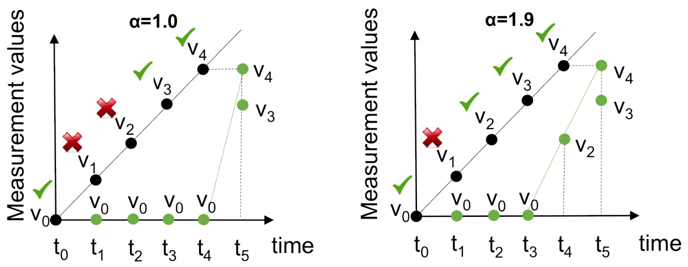
	-64dBm		-84dBm	
path order number	$\alpha = 1.0$	$\alpha = 1.9$	$\alpha = 1.0$	$\alpha = 0.2$
123	1943	1244	125	649
132	123	123	108	48
213	2	700	24	257
231	876	877	1408	1937
312	24	2	104	7
321	32	54	1231	102
$RMSE_{\text{tot}}$(MW)	0.039	0.038	0.121	0.112

Table 3: Comparison of the impact of α value on the control system performance for the NPP use case.

perform better than that with $\alpha = 1.0$ by 2.6% and 8%, respectively.

In addition to looking at reliability (given the move of changing the highest quality path from p_1 to p_2 as soon as there is a packet loss on path 1), one should wonder what happens to the delay when $\alpha = 1.9$ in contrast with $\alpha = 1.0$. Intuitively, the delay should be higher, given the new redundant paths chosen for $\alpha = 1.9$. Let's consider, with the visual help of Figure 14, the case when there are two consecutive losses on p_1 and the other paths always delivery the packets successfully.

In Figure 14, we show the values received in an ideal system (no packet losses and no delays) in black dots and the light green dots are the measurement actually used by physical system for control via wireless transmission. The error sign (red X) means the measurement was lost and the green check mark means the measurement was received. We assume that if the measurement packet is not received, the control system uses the latest received measurement value. In the table below the graph, we show highest priority measurement packet reception comparison over 5 time steps. The packet delivery status of three network paths is shown in the leftmost 3 columns of the table (blue), where 0



Time steps	Path1	Path2	Path3	First priority packet path selection (α=1.0)	First priority packet delay (α=1.0)	Packet received status (α=1.0)	First priority packet path selection (α=1.9)	First priority packet delay (α=1.9)	Packet received status (α=1.9)
t_0	1	1	1	P_1	D=0.1	1	P_1	D=0.1	1
t_1	0	1	1	P_1	D=0.1	0	P_1	D=0.1	0
t_2	0	1	1	P_1	D=0.1	0	P_2	D=0.2	1
t_3	1	1	1	P_2	D=0.2	1	P_2	D=0.2	1
t_4	1	1	1	P_1	D=0.1	1	P_1	D=0.1	1

Figure 14: An example of measurement reception comparison of different α values when RSSI is -64dBm.

means that the packet is not received and 1 means that it is received. The path selection of highest priority packet, packet delay and reception status are shown in the middle three columns (in yellow) and rightmost three columns (in green) for the situations of $\alpha = 1.0$ and $\alpha = 1.9$, respectively, corresponding to the graphs above the table. When $\alpha = 1.0$, at time t_0, p_1 is selected to transmit the highest priority measurement packet, v_0. Since the delay is 0.1s, v_0 is received at time t_1. At time t_1, p_1 is again selected to transmit the highest priority measurement packet, v_1 but the packet is lost. At t_2, the physical system uses the previous received v_0. At time t_2, v_2 is also sent via p_1 and gets lost. At t_3, v_0 is used in the control algorithm again. At time t_3, n_{loss} of p_1 is 2, and now p_2 has higher quality than p_1 ($DL(p_1) = 0.3 > DL(p_2) = 0.2$, from Equation 8); in this case, our scheme changes the highest priority packet path from p_1 to p_2. Again at time t_3, v_3 is sent via p_2 but it will only arrive at time t_5 because p_2 has longer delay. At t_4, v_0 is still used, since v_3 has not arrived yet. At time t_4, since p_1 delivers its packet successfully, and p_1 becomes the highest quality path again[4] ($DL(p_1) = 0.1 < DL(p_2) = 0.2$). v_4 is sent via p_1 and arrives at t_5. Since v_3 and v_4 arrive at the same time, the most recent measurement, v_4 is used. A similar process for $\alpha = 1.9$ is shown in the right plot in Figure 14.

We observe that the area between the black dots (ideal) and light dots (actual measurement reception or use of previous measurements) of $\alpha = 1.0$ is bigger than that of $\alpha = 1.9$. The area demonstrates how close the actual used measurement is to the ideal measurement. The other priority packets are not affected as much as the top priority packets by the path selection with different α values, since the percentage of $n_{loss} \geq 3$ of path 1 is small and when path order is changed to 213 or 231 (n_{loss} of p_1 is 1 or 2), the other priority paths using path 1 will not be affected much by losing packets.

When network RSSI is -84dBm, the three paths all drop a lot of messages and the reliability cannot be guaranteed (see Figure 10). We note that when we set $\alpha = 1$, we pay too much attention to the reliability of the top priority packets and ignore the other priority packets when compared with $\alpha = 0.2$. Let us assume $n_{loss} = 0$ on both p_1 and p_2 and consider the PQModel calculation. When $\alpha = 1.0$, p_1 will be selected as the lowest quality path for lowest priority packet as soon as n_{loss} of p_1 is 2; in contrast, when $\alpha = 0.2$, only when there are 6 consecutive message losses on path 1, the path order is changed from 123 to 213. So, with $\alpha = 0.2$, the reliability of the other priority packets is leveraged, that is, p_1 is less often selected to transmit lowest priority packets than $\alpha = 1.0$. As shown in Table 3, when $\alpha = 1.0$, the number of lowest priority packets (third priority) using path p_1 is about 2639 ($1408 + 1231$); when $\alpha = 0.2$, the lowest priority packets using path p_1 is 2038 ($1937 + 102$). Thus, the lowest priority packets of $\alpha = 1.0$ has more packet losses than that of $\alpha = 0.2$, which causes a higher $RMSE_{\text{tot}}$, since the delivery ratio of p_1 is very low (i.e., $dr_1 = 0.035$).

Figure 15 shows an illustrative example that compares one of the measurement (outlet hot leg temperature) of the physical system with lowest priority

[4]At the end of t_3, the remote controller knows p_1 delivered one packet successfully and decides to set p_1 as the highest priority path again.

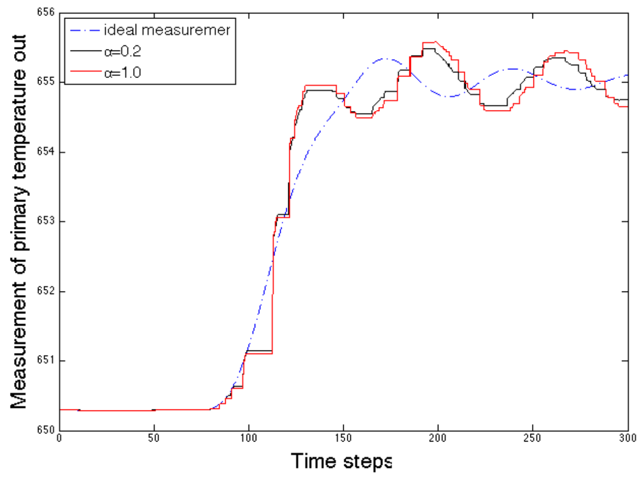


Figure 15: An example of measurement with the lowest priority of primary temperature out when RSSI is −84dBm.

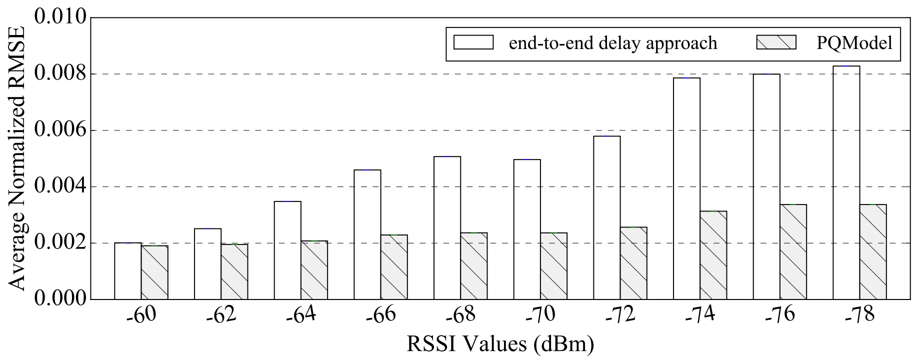
for most of the time steps, when RSSI= -84. The measurement of $\alpha = 1.0$ (red line) deviates more from the ideal measurement than that of $\alpha = 0.2$, thus causes higher $RMSE_{\text{tot}}$, which demonstrates our explanation.

In summary, when the network has less interference (high RSSI around -64dBm), only p_1 drops messages comparing with the other paths, the message loss of p_1 is emphasized by setting α big, so that the other paths can take care of the loss of p_1 by replacing p_1 to send the highest priority packet and the reliability of the highest priority packet is guaranteed. However, when the network has more interference (low RSSI around -74dBm), the three paths all drop messages and the reliability of different priority packets should be leveraged by setting α small. We cannot ignore the lowest priority packets by always assigning them to the lowest quality path, because it would induce higher $RMSE_{\text{tot}}$. Therefore, we have a counter-intuitive result: when the network has less interference, we pay more attention to message loss by setting α big and when the network has more interference, we pay less attention to message loss by setting α low. Moreover, the results demonstrate it is important to figure out the relationship between network delay and message loss under different network conditions to achieve high control system performance.

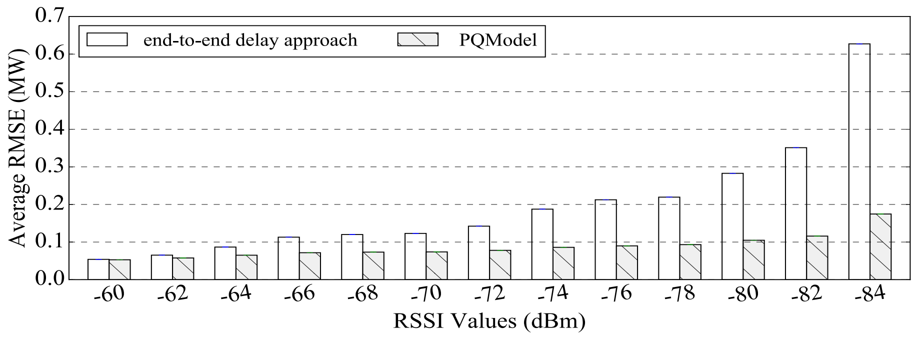
In the following, the best values o α are selected for a given RSSI, while the PID-heuristic parameters will be fixed.

10.3.2. End-to-end delay and PQModel comparison

We evaluate the $RMSE_{\text{tot}}$ for end-to-end delay approach and PQModel approach over 100 different sets of the reference functions of three physical systems (both for the IP and for the NPP use cases). For each set of reference functions, we run 20 times on the three wireless network paths for each RSSI value. The average $RMSE_{\text{tot}}$ is shown in Figure 16. The PQModel performs better



(a) IP



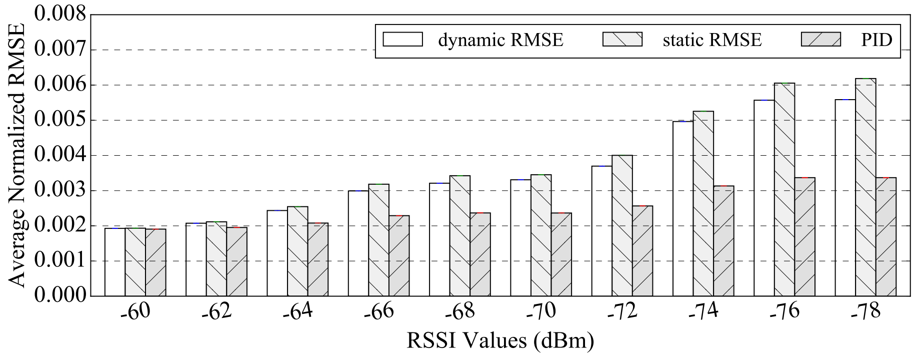
(b) NPP

Figure 16: Comparison of the two network models as a function of the RSSI.

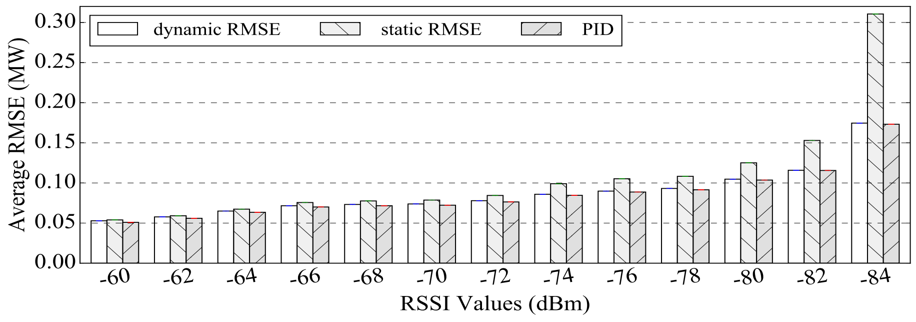
than only considering end-to-end delay in all network conditions in both the use cases, providing a much more stable performance over different RSSI conditions. Such a result demonstrates how accounting for both delay and packet losses in the network model can significantly improve the quality of the obtained results, while providing more robust solutions against interfering networks.

10.3.3. Packet priority assignment method comparison

Finally, we evaluated the presented approach in the two use cases, with the received signal strength that has different levels of network interference. Figure 17 shows the simulation results obtained in the two different use cases. We compare the packet priority assignment methods while changing the RSSI



(a) IP



(b) NPP

Figure 17: Comparison of the heuristic methods as a function of the RSSI, for the corresponding best values of α.

to have different levels of network interference. The two dynamic packet priority assignment methods always perform better than the static heuristic by 6% to as much as 79%. This is because the packet priority of the static heuristic is fixed during the execution, not providing flexibility required by the changing network conditions and the different demands of the different PSs. Therefore, dynamic packet priority are preferable, as expected.

Furthermore, Figure 17 also highlights that the PID heuristics provides a much more uniform performance with respect to an increasing network interference, performing always better than the dynamic RMSE heuristic. Such results suggest that the PID heuristic can provide a more robust priority assignment mechanism with respect to the RSSI than the other considered approaches.

11. Conclusion and future work

In this paper, we explored the interaction between dynamic packet scheduling and the control system performance in a WCS with one shared wireless network and multiple physical systems. Motivated by the observation that network delay and packet loss play the different effect on control system performance depending on the system application demand, we proposed a dynamic priority assignment mechanism with the goal of minimizing the overall control system error caused by network imperfections, in presence of multiple control systems. Specifically, our solution has two steps: measurement packet priority assignment and network path quality determination taking into account both the network delay and the message loss. From the worst-case end-to-end delay analysis, we get the schedulability condition, $\left\lfloor \frac{p_s}{l} \right\rfloor \geq 5$. Based on the condition, we derive the worst-case end-to-end delay as $D_{network}^{worst} = (2nl + 3l\left\lfloor \frac{2nl-3l}{p_s-3l} \right\rfloor)\Delta t$. To evaluate our solution, we carried out a case study on two use cases with one shared wireless network: three inverted pendula, and three SMRs in nuclear power plant.

We have two conclusions. First, the simulation results show 100% accuracy for the schedulability condition test. With the schedulablity condition satisfied, the simulation results show that our end-to-end delay analysis is accurate within

4.2% of the realistic simulation results (always pessimistic, but a very tight pessimism). Second, we came to a counter-intuitive conclusion that when the network has less interference, message loss is more important on quantifying the network quality; **but when the network has more interference, message loss is less important**, because the reliability of lower priority packets can be guaranteed anyway. In addition, our results also highlight the importance of exploring the relationship of network delay and message loss under different network conditions, which can help us reduce the control system performance degradation brought by the network imperfections.

This work allows also to highlight an interesting control problem, that has not been widely addressed in the control literature, namely seeking for characterization and fundamental bounds of time-varying delays in networked control systems [40, 41], that is typically limited to linear systems. As future work, a theoretical analysis of how to identify such characterization and bounds is envisioned.

References

[1] J. Araújo, M. Mazo, A. Anta, P. Tabuada, K. H. Johansson, System architectures, protocols and algorithms for aperiodic wireless control systems, IEEE Transactions on Industrial Informatics 10 (1) (2014) 175–184. `doi:10.1109/TII.2013.2262281`.

[2] K. Gatsis, A. Ribeiro, G. J. Pappas, Optimal power management in wireless control systems, IEEE Transactions on Automatic Control 59 (6) (2014) 1495–1510. `doi:10.1109/TAC.2014.2305951`.

[3] T. Blevins, D. Chen, M. Nixon, W. Wojsznis, Wireless Control Foundation: Continuous and Discrete Control for the Process Industry, ISA, 2015.

[4] A. Saifullah, C. Wu, P. B. Tiwari, Y. Xu, Y. Fu, C. Lu, Y. Chen, Near optimal rate selection for wireless control systems, ACM Trans. Embed.

Comput. Syst. 13 (4s) (2014) 128:1–128:25. doi:10.1145/2584652.
URL http://doi.acm.org/10.1145/2584652

[5] S. Han, X. Zhu, A. K. Mok, D. Chen, M. Nixon, Reliable and real-time communication in industrial wireless mesh networks, in: 2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium, IEEE, 2011, pp. 3–12.

[6] B. Li, L. Nie, C. Wu, H. Gonzalez, C. Lu, Incorporating emergency alarms in reliable wireless process control, in: Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems, ACM, 2015, pp. 218–227.

[7] M. Pajic, S. Sundaram, G. J. Pappas, R. Mangharam, Topological conditions for wireless control networks, in: 2011 50th IEEE Conference on Decision and Control and European Control Conference, IEEE, 2011, pp. 2353–2360.

[8] W. Wang, C. D'Angelo, D. Mosse, D. Cole, Integrating control and fault-tolerant wireless network design for small modular nuclear reactors, in: Information Reuse and Integration (IRI), 2016 IEEE 17th International Conference on, IEEE, 2016, pp. 332–342.

[9] B. Li, Y. Ma, T. Westenbroek, C. Wu, H. Gonzalez, C. Lu, Wireless routing and control: a cyber-physical case study, in: ACM/IEEE International Conference on Cyber-Physical Systems, 2016.

[10] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, Y. Chen, Real-time wireless sensor-actuator networks for industrial cyber-physical systems, Proceedings of the IEEE 104 (5) (2016) 1013–1024. doi:10.1109/JPROC.2015.2497161.

[11] Y. Liu, R. Candell, N. Moayeri, Effects of wireless packet loss in industrial process control systems, ISA transactions 68 (2017) 412–424.

[12] H. Pei Breivold, K. Sandström, Internet of things for industrial automation – challenges and technical solutions, in: 2015 IEEE International Conference on Data Science and Data Intensive Systems, 2015, pp. 532–539. doi:10.1109/DSDIS.2015.11.

[13] J. Wurm, K. Hoang, O. Arias, A. Sadeghi, Y. Jin, Security analysis on consumer and industrial iot devices, in: 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), 2016, pp. 519–524. doi:10.1109/ASPDAC.2016.7428064.

[14] S. Mubeen, S. A. Asadollah, A. Papadopoulos, M. Ashjaei, H. Pei-Breivold, M. Behnam, Management of service level agreements for cloud services in iot: A systematic mapping study, Journal of IEEE Access 6 (2018) 30184–30207.

[15] A. Grau, M. Indri, L. L. Bello, T. Sauter, Industrial robotics in factory automation: From the early stage to the internet of things, in: IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, 2017, pp. 6159–6164. doi:10.1109/IECON.2017.8217070.

[16] L. Zhang, H. Gao, O. Kaynak, Network-induced constraints in networked control systems – a survey, IEEE Transactions on Industrial Informatics 9 (1) (2013) 403–416.

[17] H. Li, M.-Y. Chow, Z. Sun, Optimal stabilizing gain selection for networked control systems with time delays and packet losses, IEEE Transactions on Control Systems Technology 17 (5) (2009) 1154–1162.

[18] A. Onat, T. Naskali, E. Parlakay, O. Mutluer, Control over imperfect networks: Model-based predictive networked control systems, IEEE Transactions on Industrial Electronics 58 (3) (2011) 905–913.

[19] G. Pin, T. Parisini, Networked predictive control of uncertain constrained nonlinear systems: recursive feasibility and input-to-state stability analysis, IEEE Transactions on Automatic Control 56 (1) (2011) 72–87.

[20] A. Ulusoy, O. Gurbuz, A. Onat, Wireless model-based predictive networked control system over cooperative wireless network, IEEE Transactions on Industrial Informatics 7 (1) (2011) 41–51.

[21] S. Hong, X. S. Hu, T. Gong, S. Han, On-line data link layer scheduling in wireless networked control systems, in: 2015 27th Euromicro Conference on Real-Time Systems, IEEE, 2015, pp. 57–66.

[22] T. Zhang, T. Gong, C. Gu, H. Ji, S. Han, Q. Deng, X. S. Hu, Distributed dynamic packet scheduling for handling disturbances in real-time wireless networks, in: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017 IEEE, IEEE, 2017, pp. 261–272.

[23] J. Kim, K. Lakshmanan, R. R. Rajkumar, Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems, in: Proceedings of the 2012 IEEE/ACM Third International Conference on Cyber-Physical Systems, IEEE Computer Society, 2012, pp. 55–64.

[24] A. Saifullah, Y. Xu, C. Lu, Y. Chen, End-to-end delay analysis for fixed priority scheduling in wirelesshart networks, in: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE, IEEE, 2011, pp. 13–22.

[25] A. Saifullah, D. Gunatilaka, P. Tiwari, M. Sha, C. Lu, B. Li, C. Wu, Y. Chen, Schedulability analysis under graph routing in wirelesshart networks, in: Real-Time Systems Symposium, 2015 IEEE, IEEE, 2015, pp. 165–174.

[26] G. Alderisi, S. Girs, L. L. Bello, E. Uhlemann, M. Björkman, Probabilistic scheduling and adaptive relaying for wirelesshart networks, in: 2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA), 2015, pp. 1–4. `doi:10.1109/ETFA.2015.7301600`.

[27] K. Gatsis, A. Ribeiro, G. J. Pappas, Control-aware random access com-

munication, in: 2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS), IEEE, 2016, pp. 1–9.

[28] M. Pajic, S. Sundaram, G. J. Pappas, R. Mangharam, The wireless control network: A new approach for control over networks, IEEE Transactions on Automatic Control 56 (10) (2011) 2305–2318.

[29] W. Wang, D. Mosse, J. G. Pickel, D. Cole, Work-in-progress: Cross-layer real-time scheduling for wireless control system, in: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017 IEEE, IEEE, 2017, pp. 149–152.

[30] W. Wang, D. Mosse, J. G. Pickel, D. Cole, Work-in-progress: Wireless network reconfiguration for control systems, in: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017 IEEE, IEEE, 2017, pp. 145–148.

[31] W. Wang, D. Mosse, D. Cole, J. G. Pickel, Dynamic wireless network reconfiguration for control system applied to a nuclear reactor case study, in: Proceedings of the 26th International Conference on Real-Time Networks and Systems, ACM, 2018, pp. 30–40.

[32] A. Saifullah, Y. Xu, C. Lu, Y. Chen, Real-time scheduling for wirelesshart networks, in: Real-Time Systems Symposium (RTSS), 2010 IEEE 31st, IEEE, 2010, pp. 150–159.

[33] P. Pazzaglia, L. Pannocchi, A. Biondi, M. Di Natale, Beyond the Weakly Hard Model: Measuring the Performance Cost of Deadline Misses, in: 30th Euromicro Conference on Real-Time Systems (ECRTS 2018), Vol. 106, 2018, pp. 10:1–10:22. `doi:10.4230/LIPIcs.ECRTS.2018.10`.

[34] V. Dolk, M. Heemels, Event-triggered control systems under packet losses, Automatica 80 (2017) 143 – 155. `doi:https://doi.org/10.1016/j.automatica.2017.02.029`.

URL http://www.sciencedirect.com/science/article/pii/S0005109817301036

[35] F.-L. Qu, Z.-H. Guan, T. Li, F.-S. Yuan, Stabilisation of wireless networked control systems with packet loss, IET Control Theory & Applications 6 (15) (2012) 2362–2366. doi:10.1049/iet-cta.2010.0562.

[36] K. Åström, R. Murray, Feedback Systems: An Introduction for Scientists and Engineers, Princeton University Press, 2018.

[37] W. Wang, D. Mosse, D. G. Cole, Bitvector: Fault tolerant aggregation scheme for monitoring in nuclear power plants, in: ICESS 2015.

[38] J. Eker, A. Cervin, A matlab toolbox for real-time and control systems co-design, in: Proceedings Sixth International Conference on Real-Time Computing Systems and Applications. RTCSA'99 (Cat. No.PR00306), 1999, pp. 320–327. doi:10.1109/RTCSA.1999.811266.

[39] S. R. Greene, J. C. Gehin, D. E. Holcomb, J. J. Carbajo, D. Ilas, A. T. Cisneros, V. K. Varma, W. R. Corwin, D. F. Wilson, G. L. Yoder Jr, et al., Preconceptual design of a fluoride-salt-cooled small modular advanced high-temperature reactor (smahtr), Tech. Rep. ORNL/TM-2010/199, Oak Ridge National Laboratory, Oak Ridge, TN, USA (2010). doi:10.2172/1008830.

[40] K. Gu, J. Chen, V. L. Kharitonov, Stability of time-delay systems, Springer Science & Business Media, 2003.

[41] J.-H. Kim, Note on stability of linear systems with time-varying delay, Automatica 47 (9) (2011) 2118 – 2121. doi:https://doi.org/10.1016/j.automatica.2011.05.023.

Appendix A. Proof of Lemma 7.1

Proof. For the base case of m_0 and m_1, when both m_0 and m_1 go up, their levels are $h(m_0) = \lfloor \frac{t}{l} \rfloor$ and $h(m_1) = \lfloor \frac{t-p_s}{l} \rfloor$ ($\lfloor \frac{t}{l} \rfloor < n$), respectively. The level difference of m_0 and m_1 is $\Delta h(m_0, m_1) = \lfloor \frac{p_s}{l} \rfloor \leq 2$. Conflict situation 2 happens, since m_0 and m_1 are separated by less than 3 levels. At time $t = p_s, h(m_0) = \lfloor \frac{p_s}{l} \rfloor$, m_1 is sent out, and m_0 needs to wait until m_1 is at level $h(m_1) = \lfloor \frac{p_s}{l} \rfloor + 3$ at time $t = p_s + 3l$. However, at time $t = 2p_s < p_s + 3l$ (i.e., before the conflict of m_0 and m_1 is resolved), m_2 will be transmitted and also block m_1. Since the conflict of m_0 and m_1 cannot be resolved, m_0 will never move past level $\lfloor \frac{p_s}{l} \rfloor$.

In general, the situation of any two consecutive messages m_i and m_{i+1} is similar to the situation of m_0 and m_1, where at time $t = (i+1)p_s$, m_{i+1} will start transmission and interrupt m_i at level $\lfloor \frac{p_s}{l} \rfloor$, creating a chain reaction. Therefore, all messages will be blocked by messages with higher priority and no message can be delivered to the destination. Since all messages are blocked at level $\lfloor \frac{p_s}{l} \rfloor$ when going up, we do not need to consider conflicts situation 1 and 3 because they will never occur if $\lfloor \frac{p_s}{l} \rfloor \leq 2$. □

Appendix B. Proof of Lemma 7.2

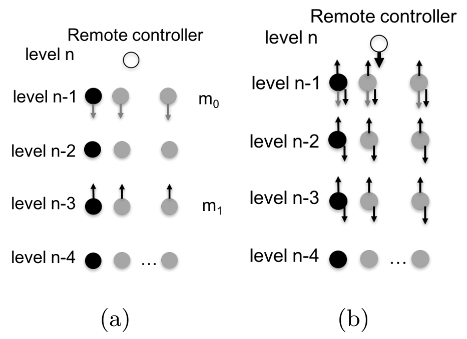


Figure B.18: Conflict situation when $\lfloor \frac{p_s}{l} \rfloor = 4$: (a) m_0 starts conflicting with m_1 and (b) the conflict is resolved in $7l$ time slots if the subsequent messages do not exist.

Proof. Let us first consider the best case (the largest separation between two consecutive messages): $\lfloor \frac{p_s}{l} \rfloor = 4$.

For the base case, when both m_0 and m_1 go up ($\lfloor \frac{t}{l} \rfloor < n$), $\Delta h(m_0, m_1) = \lfloor \frac{p_s}{l} \rfloor \geq 3$ with no conflict. When m_0 is already going down ($\lfloor \frac{t}{l} \rfloor \geq n$) and

m_1 is still going up ($\left\lfloor \frac{t-p_s}{l} \right\rfloor < n$), $\Delta h(m_0, m_1) = 2n - \left\lfloor \frac{t}{l} \right\rfloor - \left\lfloor \frac{t-p_s}{l} \right\rfloor \leq 2n - 2\left\lfloor \frac{t}{l} \right\rfloor + \left\lfloor \frac{p_s}{l} \right\rfloor \leq \left\lfloor \frac{p_s}{l} \right\rfloor = 4$. Let us consider the best case (the largest separation of m_0 and m_1) with $\Delta h(m_0, m_1) = 4$. As shown in Figure B.18a, the conflict happens when $h(m_0) = n - 1$ on the way down (grey arrow represents m_0) and $h(m_1) = n - 3$ on the way up (black arrow represents m_1). As shown in Figure B.18b, the conflict involves conflict situations 1 and 3: (1) during the conflict situation 1, m_0 is blocked by m_1, while m_1 goes up to the remote controller; (2) when m_1 reaches remote controller, the conflict becomes conflict situation 3 and is resolved until m_1 reaches level $n-3$. So the conflict is resolved in $7l$ time slots if m_2 and the following messages do not exist. However, after $4l$ slots of the conflict of m_0 and m_1, where m_1 is on the way down at level $n - 1$, m_1 will conflict with m_2 (like the situation in Figure B.18a) and the previous conflict of m_0 and m_1 will never be resolved. m_0 will be blocked at level $n - 1$ forever.

For general case of m_i and m_{i+1}, when m_i goes down, $h(m_i) = 2n - \left\lfloor \frac{t-ip_s}{l} \right\rfloor$ ($\left\lfloor \frac{t-ip_s}{l} \right\rfloor \geq n$); and when m_{i+1} goes up, $h(m_{i+1}) = \left\lfloor \frac{t-(i+1)p_s}{l} \right\rfloor$ ($\left\lfloor \frac{t-(i+1)p_s}{l} \right\rfloor < n$). Since $\Delta h(m_i, m_{i+1}) = 2n - \left\lfloor \frac{t-ip_s}{l} \right\rfloor - \left\lfloor \frac{t-(i+1)p_s}{l} \right\rfloor \leq 2n - 2\left\lfloor \frac{t-ip_s}{l} \right\rfloor + \left\lfloor \frac{p_s}{l} \right\rfloor \leq 4$, with the best case of the largest level difference of 4, m_i will conflict with m_{i+1} as the same situation of m_0 and m_1 above. After $4l$ of the conflict of m_i and m_{i+1} (the conflict takes $7l$ to resolve), m_{i+1} conflicts with m_{i+2}, and the conflict of m_i and m_{i+1} cannot be resolved. Therefore, all the messages will be blocked by higher priority messages at level $n - 1$ with $\left\lfloor \frac{p_s}{l} \right\rfloor = 4$.

Clearly, if the best case of $\left\lfloor \frac{p_s}{l} \right\rfloor = 4$ causes indefinite blocking, the case of $\left\lfloor \frac{p_s}{l} \right\rfloor = 3$ will come to the same conclusion.

□

Appendix C. Proof of Lemma 7.3

Proof. We prove this Lemma by showing that it is true for the worst case (smallest separation of two consecutive messages) when $\left\lfloor \frac{p_s}{l} \right\rfloor$ is odd, that is, $\frac{p_s}{l} = 5$. We show the Lemma is true for the base case of m_0 and m_1, and then generalize

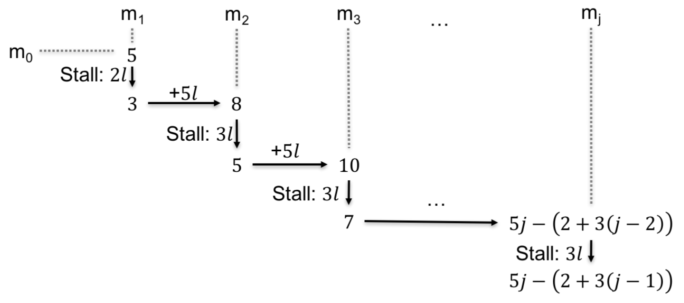


Figure C.19: The calculation process of level separations with higher priority messages of m_0 and m_1, when $\frac{p_s}{l} = 5$.

to any two consecutive messages, m_i and m_{i+1}. There are three cases:

(1) When both m_0 and m_1 go up ($\lfloor \frac{t}{l} \rfloor < n$), $\Delta h(m_0, m_1) = \lfloor \frac{t}{l} \rfloor - \lfloor \frac{t-p_s}{l} \rfloor = \frac{p_s}{l} = 5 \geq 3$, there is no conflict.

(2) When m_0 goes down ($\lfloor \frac{t}{l} \rfloor \geq n$) and m_1 goes up ($\lfloor \frac{t-p_s}{l} \rfloor < n$), $\Delta h(m_0, m_1) = 2n - \lfloor \frac{t}{l} \rfloor - \lfloor \frac{t-p_s}{l} \rfloor = 2n - 2\lfloor \frac{t}{l} \rfloor + \frac{p_s}{l}$. The conflict only involves the conflict situation 1. Since we are dealing with the case of $\frac{p_s}{l} = 5$, which means level separation is odd, so is $\Delta h(m_0, m_1)$, the conflict happens with $\Delta h(m_0, m_1) = 1$ and can be resolved in $2l$ time slots. By solving $\Delta h(m_0, m_1) = 2n - 2\lfloor \frac{t}{l} \rfloor + \frac{p_s}{l} = 1$, we get $\lfloor \frac{t}{l} \rfloor = n + 2$. After this conflict, m_0 stays at the same level as the conflict before (stalled), $h(m_0) = 2n - \lfloor \frac{t}{l} \rfloor = n - 2$; m_1 goes up 2 levels and $h(m_1) = \lfloor \frac{t-p_s}{l} \rfloor + 2 = n - 1$. Although the level difference is 1, m_0 and m_1 are in the situation shown in Figure 3d, there is no more conflict between m_0 and m_1. Figure C.19 shows the level separation of m_0 and m_1 is 5 to start with (before conflict), going down to 3, after the conflict (because m_1 advances 2 levels while m_0 stalls).

(3) When both m_0 and m_1 go down, m_0 and m_1 will conflict with higher priority messages, m_2, m_3, …, m_j. These conflicts involve the conflict situation 1, given that m_2, m_3, …, m_j are going up. For both m_0 and m_1, only the first conflict starts with an odd level separation (for m_0 see case (2) above) and the rest of conflicts are all even. Therefore, as shown in Figure C.19, conflicts after the first conflict are resolved in $3l$ time slots. A similar process can be followed

Table C.4: The total stalls of m_0 and m_1 (i.e., d_0 and d_1) when m_0 and m_1 conflict with higher priority messages ($\frac{p_s}{l} = 5$)

	m_1	m_2	m_3	$\ldots$	m_j
m_0	$2l$	$(2+3)l$	$(2+2*3)l$		$2l+3(j-1)l$
m_1	-	$2l$	$(2+3)l$		$2l+3(j-2)l$

for m_1. Table C.4 shows the total stalls in terms of the number of time slots when m_0 and m_1 conflict with m_2, m_3, …, m_j under the condition $\frac{p_s}{l} = 5$.

In addition to conflict situation 1, we also need to consider conflict situation 3, given that when both m_0 and m_1 go down and m_0 is ahead of m_1, m_0 will stall first given the conflict, causing m_1 to approach m_0, further causing situation 3 conflict. Below, we discuss three subcases to show how these conflicts are resolved: (3A) m_0 and m_1 conflicting with m_2, (3B) m_0 and m_1 conflicting with m_3 and (3C) m_0 and m_1 conflicting with m_j ($j \geq 2$).

Case 3A: m_0 and m_1 conflict with m_2. During the conflict of m_0 with m_2, m_0 will go down 1 level, and during the conflict of m_1 with m_2, m_1 will go down 2 levels, as follows. The level of m_0, m_1 and m_2 is $h(m_0) = 2n - \left\lfloor \frac{t-2l}{l} \right\rfloor$ (as shown in Table C.4, $d_0 = 2l$ due to the conflict with m_1), $h(m_1) = 2n - \left\lfloor \frac{t-p_s}{l} \right\rfloor$ and $h(m_2) = \left\lfloor \frac{t-2p_s}{l} \right\rfloor$, respectively. When m_0 starts conflicting with m_2, $\Delta h(m_0, m_2) = 2n - \left\lfloor \frac{t-2l}{l} \right\rfloor - \left\lfloor \frac{t-2p_s}{l} \right\rfloor = 2$, and we get $\left\lfloor \frac{t}{l} \right\rfloor = n + \frac{p_s}{l}$, so $t_c(m_0, m_2) = nl + p_s$ (as mentioned earlier, it is the time m_0 and m_2 starts conflicting) and $h(m_0) = 2n - \left\lfloor \frac{t-2l}{l} \right\rfloor = 2n - \left\lfloor \frac{t}{l} \right\rfloor + 2 = n - \frac{p_s}{l} + 2$. When m_1 starts conflicting with m_2, $\Delta h(m_1, m_2) = 2n - \left\lfloor \frac{t-p_s}{l} \right\rfloor - \left\lfloor \frac{t-2p_s}{l} \right\rfloor = 1$, and we get $\left\lfloor \frac{t-p_s}{l} \right\rfloor = n + \frac{1}{2}\frac{p_s}{l} - \frac{1}{2}$, so $t_c(m_1, m_2) = nl + \frac{3}{2}p_s - \frac{1}{2}l$ and $h(m_1) = 2n - \left\lfloor \frac{t-p_s}{l} \right\rfloor = n - \frac{1}{2}\frac{p_s}{l} + \frac{1}{2}$. Given that $\Delta h(m_1, m_0) = n - \frac{1}{2}\frac{p_s}{l} + \frac{1}{2} - (n - \frac{p_s}{l} + 2) = \frac{1}{2}\frac{p_s}{l} - \frac{3}{2} = 1 < 3$ (i.e., the level difference between m_0 and m_1 when m_0 and m_1 start conflicting with m_2), m_0 and m_1 will conflict again with each other (this time under conflict situation 3).

To explain how long m_0 gets stalled before m_1 starts its conflict with m_2, we turn to Figure C.20, which shows the stall time of m_0 from I_0 to I_2 and m_1 from

I_2 to I_3. The length of I_0, I_1, I_2 and I_3 is l, the time to transmit a message for one level. Since m_0 stalls for $3l$ and $t_c(m_1, m_2) - t_c(m_0, m_2) = \frac{1}{2}p_s - \frac{1}{2}l = 2l$, the overlap of m_0 and m_2 is l, that is, I_2. During I_0 to I_1, m_0 conflicts with m_2 (and stalls), while m_1 keeps going down 2 levels and m_2 goes up 2 levels. During I_2, both m_0 and m_1 conflict with m_2 and only m_2 (highest priority) goes up 1 level. During I_3, m_1 conflicts with m_2, allowing m_0 to go down 1 level and m_2 to go up 1 level.

m_0 and m_1 will not conflict with m_3, since during I_3 ($\lfloor \frac{t}{l} \rfloor = n + \frac{p_s}{l} + 3$), the level of m_0, m_1 and m_3 is $h(m_0) = n - \frac{p_s}{l} + 2$, $h(m_1) = n - \frac{1}{2}\frac{p_s}{l} + \frac{1}{2}$ and $h(m_3) = \lfloor \frac{t-3p_s}{l} \rfloor = \lfloor \frac{t}{l} \rfloor - \frac{3p_s}{l} = n - \frac{2p_s}{l} + 3$, respectively, with $\Delta h(m_0, m_3) = 4$ and $\Delta h(m_1, m_3) = 5$ both greater than 3. From I_0 to I_2, the level of m_0 and m_1 are both higher than their levels during I_3 and the level of m_3 is lower than its level of I_3. Since there is no conflict with m_3 during I_3, there is no conflict from I_0 to I_2. Thus, m_0 and m_1 will not conflict with m_3 and will not conflict with other messages (i.e., higher priority messages of m_3) either during I_0 to I_3.

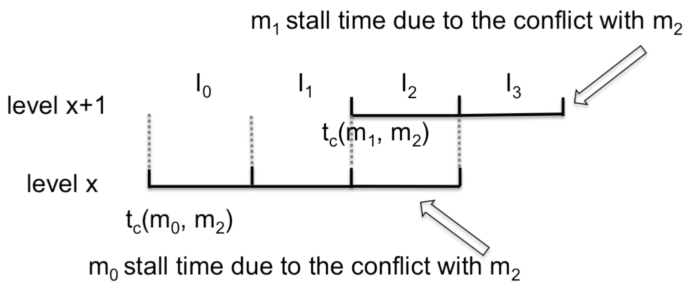
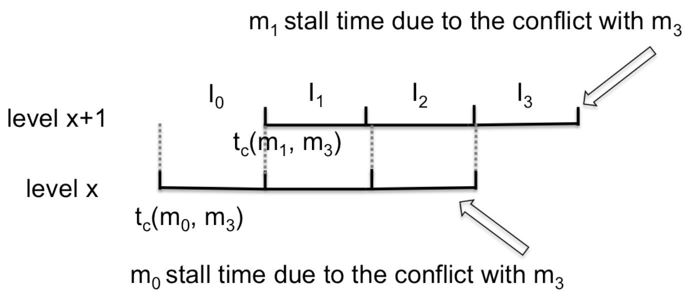

m1 stall time due to the conflict with m2
I0
I1
I2
I3
level x+1
tc(m1, m2)
level x
tc(m0, m2)
m0 stall time due to the conflict with m2

Figure C.20: The stall time for m_0 (lower segments) and m_1 (upper segments), when conflicting with m_2.

Case 3B: m_0 and m_1 conflict with m_3. m_0 and m_1 will not be completely blocked during the conflicts with m_3: m_0 and m_1 will both go down for 1 level. The level of m_0, m_1 and m_3 is $h(m_0) = 2n - \lfloor \frac{t-5l}{l} \rfloor$ (as shown in Table C.4, $d_0 = 5l$ due to the conflicts with m_1 and m_2), $h(m_1) = 2n - \left\lfloor \frac{t-p_s-2l}{l} \right\rfloor$ (as shown in Table C.4, $d_1 = 2l$ due to the conflicts with m_2) and $h(m_3) = \lfloor \frac{t-3p_s}{l} \rfloor$, respectively. When m_0 starts conflicting with m_3, $\Delta h(m_0, m_3) = 2n - \lfloor \frac{t-5l}{l} \rfloor -$



m$_1$ stall time due to the conflict with m$_3$
I$_0$
I$_1$
I$_2$
I$_3$
level x+1
t$_c$(m$_1$, m$_3$)
level x
t$_c$(m$_0$, m$_3$)
m$_0$ stall time due to the conflict with m$_3$

Figure C.21: The stall time for m_0 (lower segments) and m_1 (upper segments), when conflicting with m_3.

$\left\lfloor \frac{t-3p_s}{l} \right\rfloor = 2$, and we get $\left\lfloor \frac{t}{l} \right\rfloor = n + \frac{3}{2}\frac{p_s}{l} + \frac{3}{2}$, so $t_c(m_0, m_3) = nl + \frac{3}{2}p_s + \frac{3}{2}l$ and $h(m_0) = 2n - \left\lfloor \frac{t-5l}{l} \right\rfloor = 2n - \left\lfloor \frac{t}{l} \right\rfloor + 5 = n - \frac{3}{2}\frac{p_s}{l} + \frac{7}{2}$. When m_1 starts conflicting with m_3, $\Delta h(m_1, m_3) = 2n - \left\lfloor \frac{t-p_s-2l}{l} \right\rfloor - \left\lfloor \frac{t-3p_s}{l} \right\rfloor = 2$, and we get $\left\lfloor \frac{t-p_s}{l} \right\rfloor = n + \left\lfloor \frac{p_s}{l} \right\rfloor$, so $t_c(m_1, m_3) = nl + 2p_s$ and $h(m_1) = 2n - \left\lfloor \frac{t-p_s-2l}{l} \right\rfloor = 2n - \left\lfloor \frac{t-p_s}{l} \right\rfloor + 2 = n - \frac{p_s}{l} + 2$. Thus, $\Delta h(m_1, m_0) = \frac{1}{2}\frac{p_s}{l} - \frac{3}{2} = 1 > 3$, which means m_0 and m_1 have conflict (conflict situation 3). The start conflicting time difference is $t_c(m_1, m_3) - t_c(m_0, m_3) = \frac{1}{2}p_s - \frac{3}{2}l = l$. Figure C.21 illustrates the stall intervals for m_0 conflicting with m_1 and m_3. During I_0, m_0 conflicts with m_1 and m_3, allowing both m_1 and m_3 to go down and up for 1 level, respectively. During I_1 to I_2, m_0 conflicts with m_1 and m_3, and m_1 conflicts with m_0 and m_3, allowing only m_3 to go up 2 levels. During I_3, m_1 conflicts with m_0 and m_3, allowing m_0 to go down for 1 level and m_3 to go up 1 level. Even though m_0 and m_1 conflict, each gets a chance to move further by 1 level when the other one is stalled with m_3.

Similar to case 3A, m_4 cannot conflict with m_0 and m_1 during the conflict from I_0 to I_3. Since during I_3 ($\left\lfloor \frac{t}{l} \right\rfloor = n + \frac{3}{2}\frac{p_s}{l} + \frac{9}{2}$), the level of m_0, m_1 and m_4 is $h(m_0) = n - \frac{3}{2}\frac{p_s}{l} + \frac{7}{2}$, $h(m_1) = n - \frac{p_s}{l} + 2$ and $h(m_4) = n - \frac{5}{2}\frac{p_s}{l} + \frac{9}{2}$, respectively, with $\Delta h(m_0, m_4) = 4$ and $\Delta h(m_1, m_4) = 5$ both greater than 3, m_4 will not conflict with m_0 and m_1 during I_3. Therefore, m_4 will not conflict with any messages from I_0 to I_3 and thus no conflict of m_1, m_2 with other messages (i.e., the higher priority messages of m_4) also.

Case 3C: m_0 and m_1 conflict with m_j ($j \geq 2$). m_0 and m_1 will not be completely blocked during the conflict and can both go down 1 level. The level of m_0, m_1 and m_j is $h(m_0) = 2n - \left\lfloor \frac{t-(2+3(j-2))l}{l} \right\rfloor$ (as shown in Table C.4, $d_0 = (2+3(j-2))l$ due to the conflicts with m_0, m_1, ..., m_{j-1}), $h(m_1) = 2n - \left\lfloor \frac{t-p_s-(2+3(j-3))l}{l} \right\rfloor$ (as shown in Table C.4, $d_1 = (2+3(j-3))l$ due to the conflicts with m_1, ..., m_{j-1}) and $h(m_j) = \left\lfloor \frac{t-jp_s}{l} \right\rfloor$, respectively. In general, when m_0 starts conflicting with m_j, $\Delta h(m_0, m_j) = 2n - \left\lfloor \frac{t-(2+3(j-2))l}{l} \right\rfloor - \left\lfloor \frac{t-jp_s}{l} \right\rfloor = 2$, and we get $\left\lfloor \frac{t}{l} \right\rfloor = n + \frac{3}{2}(j-2) + \frac{j}{2}\frac{p_s}{l}$, so $t_c(m_0, m_j) = nl + \frac{3}{2}(j-2)l + \frac{j}{2}p_s$ and $h(m_0) = 2n - \left\lfloor \frac{t-(2+3(j-2))l}{l} \right\rfloor = 2n - \left\lfloor \frac{t}{l} \right\rfloor + (2+3(j-2)) = n - \frac{j}{2}\frac{p_s}{l} + \frac{3}{2}(j-2) + 2$. When m_1 starts conflicting with m_j, $\Delta h(m_1, m_j) = 2n - \left\lfloor \frac{t-p_s-(2+3(j-3))l}{l} \right\rfloor - \left\lfloor \frac{t-jp}{l} \right\rfloor = 2$, and we get $\left\lfloor \frac{t-p_s}{l} \right\rfloor = n + \frac{3}{2}(j-3) + \frac{1}{2}\left\lfloor \frac{(j-1)p_s}{l} \right\rfloor$, so $t_c(m_1, m_j) = nl + \frac{3}{2}(j-3)l + \frac{j}{2}p_s + \frac{1}{2}p_s$ and $h(m_1) = 2n - \left\lfloor \frac{t-p_s-(2+3(j-3))l}{l} \right\rfloor = 2n - \left\lfloor \frac{t-p_s}{l} \right\rfloor + (2+3(j-3)) = n - \frac{1}{2}(j-1)\frac{p_s}{l} + 2 + \frac{3}{2}(j-3)$. Thus, $\Delta h(m_1, m_0) = \frac{1}{2}\frac{p_s}{l} - \frac{3}{2} = 1$ and m_0 and m_1 are still conflicting with each other. The start conflict time difference is $t_c(m_1, m_j) - t_c(m_0, m_j) = \frac{1}{2}p_s - \frac{3}{2}l = l$. The stall time for both m_0 and m_1 is the same as Figure C.21: during the conflict, m_1 can go down 1 level during I_0; and m_0 can go down 1 level during I_3. Also, m_{j+1} will not conflict with conflict with m_0 and m_1 from I_0 to I_3. Since during I_3 ($\left\lfloor \frac{t}{l} \right\rfloor = n + \frac{3}{2}j + \frac{j}{2}\frac{p_s}{l}$), the level of m_0, m_1 and m_4 is $h(m_0) = n - \frac{j}{2}\frac{p_s}{l} + \frac{3}{2}(j-2) + 2$, $h(m_1) = n - \frac{1}{2}(j-1)\frac{p_s}{l} + 2 + \frac{3}{2}(j-3)$ and $h(m_{j+1}) = n + \frac{3}{2}j - (\frac{j}{2}+1)\frac{p_s}{l}$, respectively. With $\Delta h(m_0, m_{j+1}) = 4$ and $\Delta h(m_1, m_{j+1}) = 5$, m_{j+1} and other higher priority messages will not conflict with m_0 and m_1 from I_0 to I_3. This pattern will repeat itself indefinitely in the worst case.

No delay caused by the conflict of m_0 and m_1 for Case 3A, 3B and 3C According to the Case 3A, Case 3B and Case 3C, m_0 and m_1 always conflict with each other. However, the conflict does not induce more delay is because the duration between the start time of the conflict of m_0 with m_j ($j \geq 2$) and the start time of the conflict of m_0 with m_{j+1} is $t_c(m_0, m_{j+1}) - t_c(m_0, m_j) = nl + \frac{3}{2}(j-1)l + \frac{j+1}{2}p_s - (nl + \frac{3}{2}(j-2)l + \frac{j}{2}p_s) = \frac{3}{2}l + \frac{1}{2}p_s = 4l$. As shown in Figure C.22, the duration between the start time of the two conflicts equals

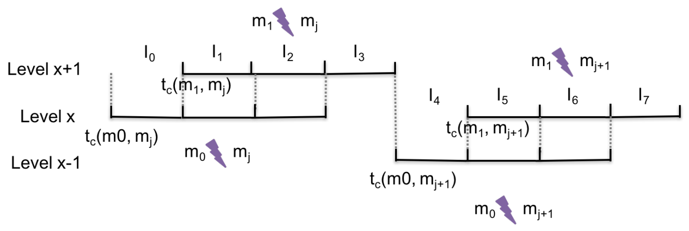


Figure C.22: The stall time for m_0 (lower segments) and m_1 (upper segments), when conflicting with m_j and m_{j+1}.

to the duration of the conflicts among m_0, m_1 and m_j, which means that the new conflict of m_0 and m_{j+1} starts when the conflicts among m_0, m_1 and m_j finishes. There is no "rest time" between the conflicts among m_0, m_1 and m_j and the conflicts among m_0, m_1 and m_{j+1}. So, the conflict of m_0 and m_1 always happen during the conflicts with other higher priority messages and will not induce more stall time alone.

For any two consecutive messages, m_i and m_{i+1}, we can show the message progress, similar to the process above. Conflicts always happen when the lower priority messages are going down (conflict situation 1). Even though the two messages going down conflict with each other (conflict situation 3), each gets a chance to make progress when the other one is stalled due to the conflicts with higher priority messages (the newer message m_{i+1} will never get ahead of the older message m_i); both messages finally can reach to the destination.

The proof above is for the worst case for odd separation ($\frac{p_s}{l} = 5$). Outside the worst case, the message density is lower, and therefore fewer conflicts and stalls will happen, which comes to the same conclusion.

$\square$

Appendix D. Proof of Lemma 7.4

Proof. Similar to odd value of $\lfloor \frac{p_s}{l} \rfloor \geq 5$, we first consider the worst case of the smallest separation of two consecutive messages when $\lfloor \frac{p_s}{l} \rfloor$ is even, $\frac{p_s}{l} = 6$. We show the lemma is true for the base case of m_0 and m_1, and then generalize to

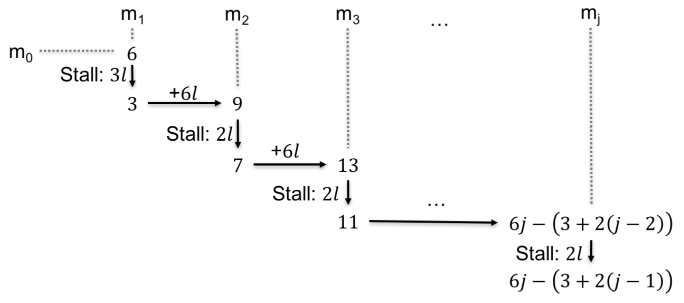


Figure D.23: The calculation process of level separations with higher priority messages for m_0 and m_1, when $\frac{p_s}{l} = 6$.

Table D.5: The total stalls of m_0 and m_1 (i.e., d_0 and d_1) when m_0 and m_1 conflict with higher priority messages ($\frac{p_s}{l} = 6$)

	m_1	m_2	m_3	$\dots$	m_j
m_0	$3l$	$(3+2)l$	$(3+2*2)l$		$3l+2(j-1)l$
m_1	-	$3l$	$(3+2)l$		$3l+2(j-2)l$

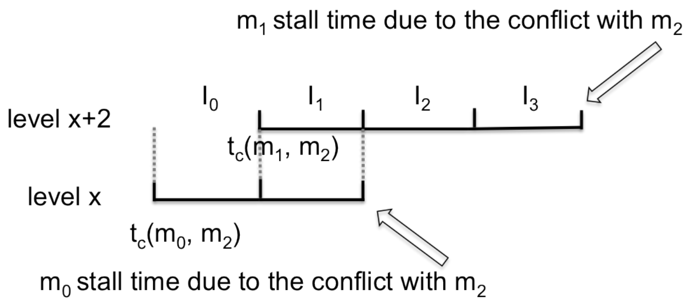
any two consecutive messages, m_i and m_{i+1}. There are three cases:

(1) When both m_0 and m_1 go up ($\lfloor \frac{t}{l} \rfloor < n$), $\Delta h(m_0, m_1) = \lfloor \frac{t}{l} \rfloor - \lfloor \frac{t-p_s}{l} \rfloor = \frac{p_s}{l} = 6 > 3$, there is no conflict.

(2) When m_0 goes down ($\lfloor \frac{t}{l} \rfloor \geq n$) and m_1 goes up ($\lfloor \frac{t-p_s}{l} \rfloor < n$), $\Delta h(m_0, m_1) = 2n - \lfloor \frac{t}{l} \rfloor - \lfloor \frac{t-p_s}{l} \rfloor = 2n - 2\lfloor \frac{t}{l} \rfloor + \frac{p_s}{l}$. The conflict only involves the conflict situation 1. Since $\frac{p_s}{l} = 6$ is even ($\Delta h(m_0, m_1)$ is even), the conflict happens with $\Delta h(m_0, m_1) = 2$ and can be resolved in $3l$ time slots. By solving $\Delta h(m_0, m_1) = 2n - 2\lfloor \frac{t}{l} \rfloor + \frac{p_s}{l} = 2$, we get $\lfloor \frac{t}{l} \rfloor = n + 2$. After this conflict, m_0 stays at the same level as the conflict before (stalled), $h(m_0) = 2n - \lfloor \frac{t}{l} \rfloor = n-2$; m_1 goes up 2 levels and $h(m_1) = \lfloor \frac{t-p_s}{l} \rfloor + 3 = n - 1$. Although the level difference is 1, m_0 and m_1 are in the situation shown in Figure 3d, there is no more conflict between m_0 and m_1. Figure D.23 shows that the level separation of m_0 and m_1 is 6 to start with (before conflict), going down to 3, after the conflict (because m_1 advances 3 levels while m_0 stalls).

(3) Similar to the case of $\frac{p_s}{l} = 5$, when both m_0 and m_1 go down, both

m_0 and m_1 will conflict with higher priority messages, m_2, m_3, ..., m_j. These conflicts involve the conflict situation 1, given that m_2, m_3, ..., m_j go up. For both m_0 and m_1, only the first conflict starts with an even level separation (for m_0 see case (2) above) and the rest of conflicts are all odd. Therefore, as shown in Figure D.23, conflicts after the first conflict are resolved in $2l$ time slots. A similar process can be followed for m_1. Table D.5 shows the total stalls in terms of the number of time slots when m_0 and m_1 conflict with m_2, m_3, ..., m_j under the condition $\frac{p_s}{l} = 6$. Below, we separate this into three subcases to show how these conflicts are resolved: (3A) m_0 and m_1 conflicting with m_2, (3B) m_0 and m_1 conflicting with m_3 and (3C) m_0 and m_1 conflicting with m_j.



m1 stall time due to the conflict with m2
l0
l1
l2
l3
level x+2
tc(m1, m2)
level x
tc(m0, m2)
m0 stall time due to the conflict with m2

Figure D.24: The stall time for m_0 (lower segments) and m_1 (upper segments), when conflicting with m_2.

Case 3A: m_0 and m_1 conflict with m_2. m_0 and m_1 will not be completely blocked during the conflicts with m_2: m_0 will go down for 2 levels, and m_1 will go down for 1 level. The level of m_0, m_1 and m_2 is $h(m_0) = 2n - \lfloor \frac{t-3l}{l} \rfloor$ (as shown in Table D.5, $d_0 = 3l$ due to the conflict with m_1), $h(m_1) = 2n - \lfloor \frac{t-p_s}{l} \rfloor$ and $h(m_2) = \lfloor \frac{t-2p_s}{l} \rfloor$, respectively. When m_0 starts conflicting with m_2, $\Delta h(m_0, m_2) = 2n - \lfloor \frac{t-3l}{l} \rfloor - \lfloor \frac{t-2p_s}{l} \rfloor = 1$, and we get $\lfloor \frac{t}{l} \rfloor = n + \lfloor \frac{p_s}{l} \rfloor + 1$, so $t_c(m_0, m_2) = nl + p_s + l$ and $h(m_0) = 2n - \lfloor \frac{t-3l}{l} \rfloor = 2n - \lfloor \frac{t}{l} \rfloor + 3 = n - \frac{p_s}{l} + 2$. When m_1 starts conflicting with m_2, $\Delta h(m_1, m_2) = 2n - \lfloor \frac{t-p_s}{l} \rfloor - \lfloor \frac{t-2p_s}{l} \rfloor = 2$, and we get $\lfloor \frac{t-p_s}{l} \rfloor = n + \frac{1}{2} \lfloor \frac{p_s}{l} \rfloor - 1$, so $t_c(m_1, m_2) = nl + \frac{3}{2}p - l$ and $h(m_1) = 2n - \lfloor \frac{t-p_s}{l} \rfloor = n - \frac{1}{2}\frac{p_s}{l} + 1$. $\Delta h(m_1, m_0) = \frac{1}{2}\frac{p_s}{l} - 1 = 2$, which means m_0 and m_1 will conflict again (conflict situation 3) with each other given that m_0 got

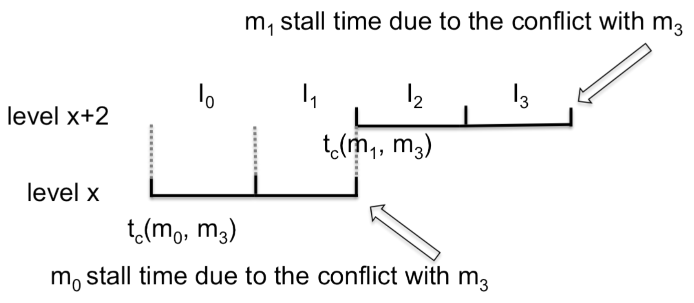

m₁ stall time due to the conflict with m₃
I0
I1
I2
I3
level x+2
tc(m1, m3)
level x
tc(m0, m3)
m0 stall time due to the conflict with m3

Figure D.25: The stall time for m_0 (lower segments) and m_1 (upper segments), when conflicting with m_3.

stalled before m_1 conflicts with m_2. $t_c(m_1, m_2) - t_c(m_0, m_2) = \frac{1}{2}p_s - 2l = l$. Figure D.24 represents the stall time for m_0 and m_1. During I_0, m_0 conflicts with m_2 (and stalls), while m_1 keeps going down for 1 level and m_2 goes up for 1 level. During I_1, m_0 conflicts with both m_1 and m_2; m_1 conflicts with m_2; only m_2 (the highest priority message) goes up 1 level. During I_2 to I_3, m_1 conflicts with m_2, allowing m_0 to go down 2 levels and m_2 to go up 2 levels.

m_0 and m_1 will not conflict with m_3, since during I_3 ($\lfloor \frac{t}{l} \rfloor = n + \lfloor \frac{p_s}{l} \rfloor + 4$), the level of m_0, m_1 and m_3 is $h(m_0) = n - \frac{p_s}{l} + 2$, $h(m_1) = n - \frac{1}{2}\frac{p_s}{l} + 1$ and $h(m_3) = \lfloor \frac{t-3p_s}{l} \rfloor = \lfloor \frac{t}{l} \rfloor - \frac{3p_s}{l} = n - \frac{2p_s}{l} + 4$, respectively, with $\Delta h(m_0, m_3) = 4$ and $\Delta h(m_1, m_3) = 6$ both greater than 3, m_3 will not conflict with m_0 and m_1. Thus, m_3 and its other higher priority messages will not conflict with m_0 and m_1 during I_0 to I_3 (see Figure D.24).

Case 3B: m_0 **and** m_1 **conflict with** m_3. m_0 and m_1 will not be blocked during the conflicts with m_3: m_0 will go down for 2 levels, and m_1 will go down for 2 levels. The level of m_0, m_1 and m_3 is $h(m_0) = 2n - \lfloor \frac{t-5l}{l} \rfloor$ (as shown in Table D.5, $d_0 = 5l$ due to the conflicts with m_1 and m_2), $h(m_1) = 2n - \left\lfloor \frac{t-p_s-3l}{l} \right\rfloor$ (as shown in Table D.5, $d_1 = 3l$ due to the conflict with m_2) and $h(m_3) = \lfloor \frac{t-3p_s}{l} \rfloor$, respectively. When m_0 starts conflicting with m_3, $\Delta h(m_0, m_3) = 2n - \lfloor \frac{t-5l}{l} \rfloor - \lfloor \frac{t-3p_s}{l} \rfloor = 1$, and we get $\lfloor \frac{t}{l} \rfloor = n + \frac{3}{2}\lfloor \frac{p_s}{l} \rfloor + 2$, so $t_c(m_0, m_3) = nl + \frac{3}{2}p_s + 2l$ and $h(m_0) = 2n - \lfloor \frac{t-5l}{l} \rfloor = 2n - \lfloor \frac{t}{l} \rfloor + 5 = n - \frac{3}{2}\frac{p_s}{l} + 3$. When m_1 starts conflicting with m_3, $\Delta h(m_1, m_3) = 2n - \left\lfloor \frac{t-p_s-3l}{l} \right\rfloor - \lfloor \frac{t-3p_s}{l} \rfloor = 1$

and get $\left\lfloor \frac{t-p_s}{l} \right\rfloor = n + \left\lfloor \frac{p_s}{l} \right\rfloor + 1$, so $t_c(m_1, m_3) = nl + 2p_s + l$ and $h(m_1) = 2n - \left\lfloor \frac{t-p_s-3l}{l} \right\rfloor = 2n - \left\lfloor \frac{t-p_s}{l} \right\rfloor + 3 = n - \frac{p_s}{l} + 2$. The level difference between m_1 and m_0 is $\Delta h(m_1, m_0) = \frac{1}{2}\frac{p_s}{l} - 1 = 2$, which means m_0 and m_1 conflict with each other again. The start conflicting time difference is $t_c(m_1, m_3) - t_c(m_0, m_3) = \frac{1}{2}p_s - l = 2l$. Figure D.25 illustrates stall intervals for m_0 and m_1. During I_0 to I_1, m_0 conflicts with m_3, allowing both m_1 and m_3 to go down and up for 2 levels, respectively. During I_2 to I_3, m_1 conflicts with m_0 and m_3, allowing both m_0 and m_3 to go down and up for 2 levels, respectively. Even though m_0 and m_1 conflict, each can move further by 2 levels when the other one conflicts with m_3.

Similar to case 3A, m_4 cannot conflict with m_0 and m_1 during the conflict from I_0 to I_3 in Figure D.25. Since during I_3 ($\left\lfloor \frac{t}{l} \right\rfloor = n + \frac{3}{2}\left\lfloor \frac{p_s}{l} \right\rfloor + 5$), the level of m_0, m_1 and m_4 is $h(m_0) = n - \frac{3}{2}\frac{p_s}{l} + 3$, $h(m_1) = n - \frac{p_s}{l} + 2$ and $h(m_4) = n - \frac{5}{2}\frac{p_s}{l} + 5$, respectively, with $\Delta h(m_0, m_4) = 4$ and $\Delta h(m_1, m_4) = 6$ both greater than 3, m_4 will not conflict with m_0 and m_1. Therefore, m_4 and its other higher priority messages will not conflict with any messages from I_0 to I_3.

Case 3C: m_0 and m_1 conflict with m_j ($j \geq 2$). m_0 and m_1 will not be blocked during the conflict and can go down by 2 levels. The level of m_0, m_1 and m_j is $h(m_0) = 2n - \left\lfloor \frac{t-(3+2(j-2))l}{l} \right\rfloor$ (as shown in Table D.5, $d_0 = (3+2(j-2))l$ due to the conflicts with m_1, m_2, ..., m_{j-1}), $h(m_1) = 2n - \left\lfloor \frac{t-p_s-(3+2(j-3))l}{l} \right\rfloor$ (as shown in Table D.5, $d_1 = (3+2(j-3))l$ due to the conflicts with m_2, ..., m_{j-1}) and $h(m_j) = \left\lfloor \frac{t-jp_s}{l} \right\rfloor$, respectively. In general, when m_0 starts conflicting with m_j, $\Delta h(m_0, m_j) = 2n - \left\lfloor \frac{t-(3+2(j-2))l}{l} \right\rfloor - \left\lfloor \frac{t-jp_s}{l} \right\rfloor = 1$, and we get $\left\lfloor \frac{t}{l} \right\rfloor = n + \frac{j}{2}\left\lfloor \frac{p_s}{l} \right\rfloor + j - 1$, so $t_c(m_0, m_j) = nl + \frac{j}{2}p_s + (j-1)l$ and $h(m_0) = 2n - \left\lfloor \frac{t-(3+2(j-2))l}{l} \right\rfloor = 2n - \left\lfloor \frac{t}{l} \right\rfloor + (3 + 2(j-2)) = n - \frac{j}{2}\frac{p_s}{l} + j$. When m_1 starts conflicting with m_j, $\Delta h(m_1, m_j) = 2n - \left\lfloor \frac{t-p_s-(3+2(j-3))l}{l} \right\rfloor - \left\lfloor \frac{t-jp}{l} \right\rfloor = 1$, and we get $\left\lfloor \frac{t-p_s}{l} \right\rfloor = n + \frac{j-1}{2}\left\lfloor \frac{p_s}{l} \right\rfloor + j - 2$, so $t_c(m_1, m_j) = nl + \frac{j+1}{2}p_s + (j-2)l$ and $h(m_1) = 2n - \left\lfloor \frac{t-p_s-(3+2(j-3))l}{l} \right\rfloor = 2n - \left\lfloor \frac{t-p_s}{l} \right\rfloor + (3+2(j-3)) = n - \frac{1}{2}(j-1)\frac{p_s}{l} + j - 1$. The level difference between m_1 and m_0 is $\Delta h(m_1, m_0) = \frac{1}{2}\frac{p_s}{l} - 1 = 2$, which means m_0 and m_1 will conflict again (conflict situation 3). The start conflict time difference is $t_c(m_1, m_j) - t_c(m_0, m_j) = \frac{1}{2}p_s - l = 2l$. The stall time for both m_0 and m_j is the same as Figure D.25: during the conflict, m_1 can go down for 2 levels during I_0 to I_1; and m_0 can go down for 2 levels during I_2 and I_3.

Also, m_{j+1} will not conflict with m_0 and m_1 from I_0 to I_3. Since during I_3 ($\lfloor \frac{t}{l} \rfloor = n + \frac{j}{2}\lfloor \frac{p_s}{l} \rfloor + j + 2$), the level of m_0, m_1 and m_4 is $h(m_0) = n - \frac{j}{2}\frac{p_s}{l} + j$, $h(m_1) = n - \frac{1}{2}(j-1)\frac{p_s}{l} + j - 1$ and $h(m_{j+1}) = n - (\frac{j}{2}+1)\frac{p_s}{l} + j + 2$, with $\Delta h(m_0, m_{j+1}) = 4$ and $\Delta h(m_1, m_{j+1}) = 6$, m_{j+1} and its higher priority messages will not conflict with m_0 and m_1 from I_0 to I_3. This pattern will repeat itself indefinitely in the worst case.

Similar to the reason of the general case of $\frac{p_s}{l} = 5$, there is no delay caused by the conflict of m_0 and m_1 for Case 3A, 3B and 3C above. For any two consecutive messages, m_i and m_{i+1}, even though they conflict with each other during the downside transmission, each gets a chance to make progress and finally reaches to the destination.

The proof above is for the worst case of $\frac{p_s}{l} = 6$. For the other even values of $\lfloor \frac{p_s}{l} \rfloor$ will obviously come to the same conclusion. $\square$