

Article

A Novel Overload Control Algorithm for Distributed Control Systems to Enhance Reliability in Industrial Automation [†]

Taikyeong Jeong ^{1,2}

¹ School of Artificial Intelligence Convergence, Hallym University, Chuncheon 24252, Republic of Korea; ttjeong@hallym.ac.kr

² Department of Psychiatry, College of Medicine, Hallym University, Chuncheon 24252, Republic of Korea

[†] A preliminary version of this paper was presented at the 2010 Spring Conference, Jeju, Republic of Korea, 24–26 June 2010, of the Korean Society for Internet Information (KSII).

Abstract: This paper presents a novel real-time overload detection algorithm for distributed control systems (DCSs), particularly applied to thermoelectric power plant environments. The proposed method is integrated with a modular multi-functional processor (MFP) architecture, designed to enhance system reliability, optimize resource utilization, and improve fault resilience under dynamic operational conditions. As legacy DCS platforms, such as those installed at the Tae-An Thermoelectric Power Plant, face limitations in applying advanced logic mechanisms, a simulation-based test bench was developed to validate the algorithm in anticipation of future DCS upgrades. The algorithm operates by partitioning function code executions into segment groups, enabling fine-grained, real-time CPU and memory utilization monitoring. Simulation studies, including a modeled denitrification process, demonstrated the system's effectiveness in maintaining load balance, reducing power consumption to 17 mW under a 2 Gbps data throughput, and mitigating overload levels by approximately 31.7%, thereby outperforming conventional control mechanisms. The segmentation strategy, combined with summation logic, further supports scalable deployment across both legacy and next-generation DCS infrastructures. By enabling proactive overload mitigation and intelligent energy utilization, the proposed solution contributes to the advancement of self-regulating power control systems. Its applicability extends to energy management, production scheduling, and digital signal processing—domains where real-time optimization and operational reliability are essential.



Academic Editor: Dong Won Kim

Received: 26 March 2025

Revised: 1 May 2025

Accepted: 6 May 2025

Published: 21 May 2025

Citation: Jeong, T. A Novel Overload Control Algorithm for Distributed Control Systems to Enhance Reliability in Industrial Automation. *Appl. Sci.* **2025**, *15*, 5766. <https://doi.org/10.3390/app15105766>

Copyright: © 2025 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: distributed control system; overload detection algorithm; optimal utilization; energy conversion system

1. Introduction

Distributed control systems (DCSs) have been widely adopted in the industrial sector to manage distributed electric power delivery processes. In particular, the DCS is often referred to as the “brain” and central control unit of coal-fired power generation systems [1]. The performance of a DCS fundamentally relies on two key factors: reliability and stability. These factors become especially critical in super-critical (SC) and ultra-supercritical (USC) power generation units, where system capacity can exceed 1000 MW. Such high-capacity systems demand enhanced reliability and operational consistency to meet performance requirements involving high parameters and efficiency levels [2]. As a result, operators and engineers routinely conduct preventive maintenance and inspections. However, overload conditions in the central processing unit (CPU)—the microprocessor responsible for core operations—are often difficult to detect through conventional diagnostic tools.

Consequently, most manufacturers have historically addressed overload issues by simply expanding memory capacity, as memory shortages and I/O bottlenecks are commonly identified as primary contributors to such problems. For instance, Bailey, Inc., the manufacturer of the INFI-90 DCS system, adopted this approach by increasing memory capacity in its MFP modules [3]. In subsequent versions, the processing unit was rebranded as the Bridge Controller Card (BRC), which introduced additional features such as fiber channel support.

Given the increasing complexity and performance demands of modern power generation systems, the role of intelligent control algorithms within the DCS framework has become more critical than ever. While hardware upgrades such as memory expansion and processor enhancements can temporarily mitigate overload issues, they do not address the root causes related to inefficient task scheduling, process prioritization, and data traffic control within the system. A well-designed control algorithm can dynamically allocate computing resources, predict potential overload conditions, and adjust control strategies in real time to maintain system stability. Moreover, in high-capacity environments such as USC power plants, where control response time and accuracy directly affect plant efficiency and safety, adaptive and predictive control algorithms can significantly enhance DCS reliability, responsiveness, and fault tolerance.

Table 1 is presented to illustrate the evolution of hardware specifications between two generations of control modules—namely, the MFP and BRC units—used in legacy and modern DCS platforms. By comparing critical parameters such as microprocessor architecture, RAM, NVRAM capacity, and power requirements, the table highlights how hardware upgrades have historically been employed as a primary strategy to address system overload. However, despite these improvements, such enhancements are often reactive and limited in scalability. This comparison sets the stage for advocating the need for proactive solutions, such as the development of advanced overload control algorithms that can complement hardware capabilities by optimizing system behavior at the software level.

Table 1. Comparison of MFP and BRC modules in terms of processor type, memory capacity, and power consumption.

Specifications	MFP (Multi-Functional Processor) Module	BRC (Bridge Controller Card) Module
Microprocessor	32 bit	32 bit
NVRAM	64 K Byte	1 M Byte
RAM	256 K Byte	2 M Byte
Power consumption	5 V/2 A	5 V/2 A

In this case, the table highlights key differences in memory capacity—both NVRAM and RAM—while showing that both modules share the same microprocessor width and power consumption. The substantial increase in memory resources in the BRC module reflects a hardware-level response to growing system complexity and the need for improved performance in DCS. As a result of these advances in semiconductor processes and foundry technology, as well as the rapid design technology of processors, we have come to realize that they will also impact the design of future control systems, and that we need more advanced technologies to support them, and that we must find a better way. However, the technical requirements and performance of certain fields and some DCSs have become much higher than before. Therefore, it is necessary to find better solutions to monitor the DCS main controller for distributed control.

In order to describe the novel overload control algorithm in greater detail—along with its optimal real-time utilization—it is necessary to investigate the reliability of the DCS and address associated safety concerns [4–6].

In this paper, to study the overall DCS system, it is necessary to recognize the system-level architecture and have in-depth knowledge of the hardware level, so that simulations that calculate overload are possible. Therefore, Figure 1 illustrates the system-level and hardware-level architecture of a DCS as applied to energy infrastructure environments. In the context of this paper, the DCS serves as the foundational platform for implementing and evaluating a novel overload control algorithm. The architecture highlights the importance of modular control, real-time data acquisition, and distributed processing capabilities, which are critical for maintaining operational stability and resilience under dynamic load conditions in large-scale energy systems. Figure 1a illustrates the architecture of a DCS, where multiple process control units (PCUs) are connected via a central DCS data highway to DCS operator stations, engineering workstations (EWSs), and peripheral devices such as printers. The system enables decentralized control by distributing computational tasks across multiple PCUs, improving fault tolerance and scalability. The configuration includes multiple PCUs connected via a centralized DCS data highway to operator stations, EWSs, and peripheral devices [7]. This hierarchical structure enables modular, distributed control and real-time supervisory monitoring. Figure 1b provides a detailed view of the internal hardware architecture, showing how microprocessors interface with RAM, ROM, I/O sections, and communication buses to handle process I/O signals and control functions efficiently. This architecture supports deterministic signal processing and robust execution of control algorithms, providing the computational foundation for implementing advanced overload management strategies in distributed energy systems. It should be noted that Figure 1 is a diagram illustrating the real-time operation of each module installed in an actual thermal power plant in Tae-an, Korea. To examine this further, it is necessary to examine more details [8,9]. This paper focuses on the best utilization and transmission of alternative methods to prevent the overload situation of DCS in advance. For this purpose, a new MFP and module are designed and applied to detect overload.

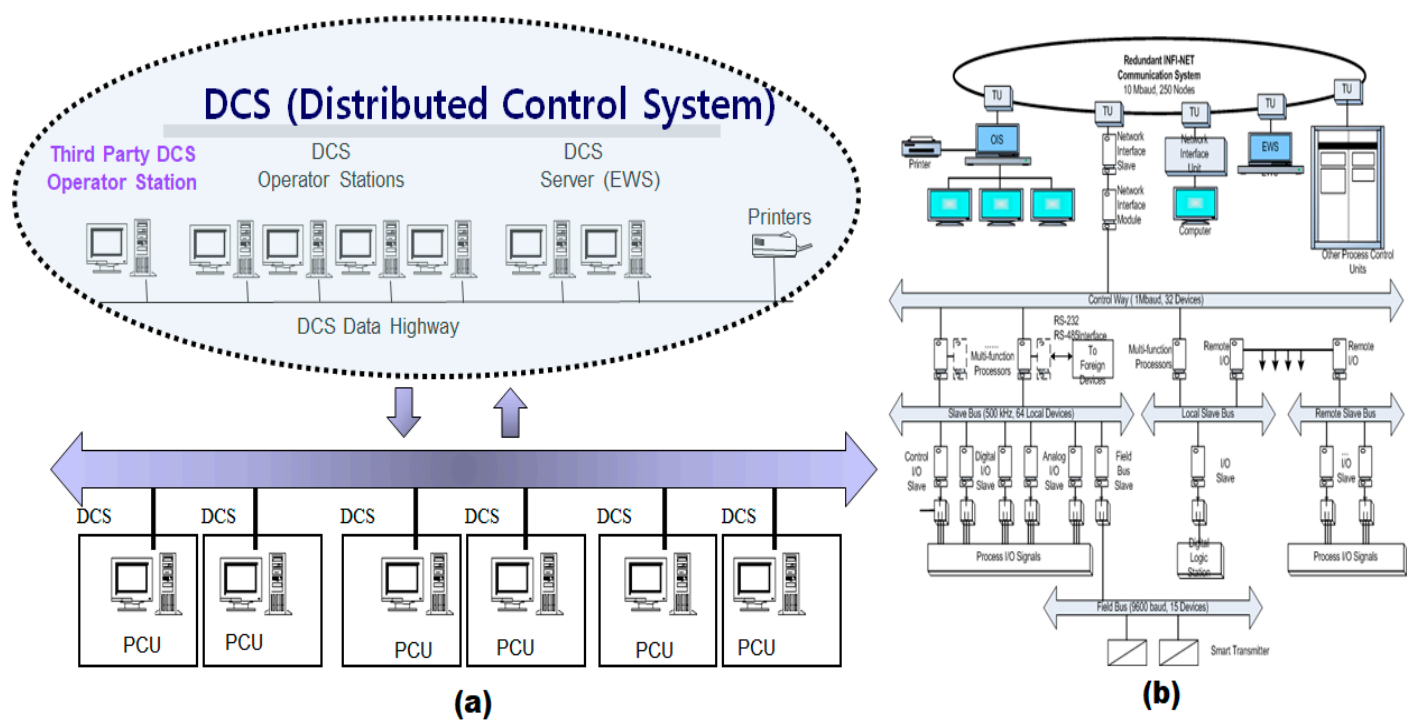


Figure 1. Overall DCS system design including diagram of MFP module: (a) system-level architecture of a DCS designed for energy infrastructure applications; (b) hardware-level architecture of a typical DCS controller, illustrating the internal interaction between the microprocessor, memory units and peripheral I/O components such as serial ports, DMA, and bus interfaces.

This paper makes several significant contributions to the field of overload control and reliability enhancement in DCSs, particularly in thermoelectric and nuclear power plants. The following are the key contributions of this paper:

- i. Proposal of a real-time overload detection and mitigation algorithm that enhances the stability and reliability of DCS.
- ii. Introduction of a modular, multi-functional logic processor (MFP) to improve fault tolerance and load balancing and ensure optimal resource utilization while maintaining operational efficiency.
- iii. Application of the proposed overload detection algorithm and MFP to a real-world denitrification process in a thermoelectric power plant.
- iv. Simulations and empirical testing to demonstrate the superiority of the proposed approach over traditional overload control mechanisms.

By combining advanced overload detection, modular logic processing, and real-world validation, this paper presents a highly scalable and intelligent solution for power system reliability and efficiency. The findings contribute to the development of next-generation, self-regulating power control systems that can adapt to increasing energy demands and environmental challenges.

The remainder of this paper is organized as follows: Section 2 introduces the proposed hardware architecture for the new multi-functional processor (NFP). This design incorporates enhanced features through the integration of specialized algorithmic chips, including the NFP analog-to-digital converter chip (AFP). Section 3 discusses CPU utilization in relation to underlying hardware characteristics. Sections 4 and 5 present the proposed optimal utilization strategy and overload detection algorithm, respectively. Section 5 also includes a detailed analysis of optimal values based on simulation results. Finally, Section 6 concludes this paper.

2. New Logic Processor Design

The multi-functional processor (MFP) module is one of the workhorses of the INFI-90 control module line [3]. It is a multiple-loop analog, sequential, batch, and advanced controller that provides powerful solutions to process control problems. It also provides true peer-to-peer (P2P) communications to handle data acquisition and information processing requirements. The comprehensive set of function codes this module supports handles even the most complex control strategies. The INFI-90 system uses a variety of analog and digital I/O modules to communicate with and control the process of 64 I/O modules in any configuration.

2.1. Hardware Characteristics

The MFP module consists of a microprocessor, 512 Kbyte read-only memory (ROM), random access memory (RAM), nonvolatile random-access memory (NVRAM), I/O expander bus, I/O section, serial port, and direct memory access (DMA) (See Figure 1). This section briefly describes each element and its operation.

First, the 16 MHz microprocessor enables module operation and control [8,9]. The operating system instructions and function code library of the microprocessor reside in ROM. Second, the I/O section is the input and output interface, which allows for the microprocessor to read the switches that tell it how to operate and what address it has [10]. The ROM holds the operating system instructions for the microprocessor. The RAM provides temporary storage and a copy of the configuration of the module. The NVRAM holds the module configuration to retain information when the power is turned off.

2.2. Software Characteristics

The DCS logic algorithm is a function code operating algorithm. A DCS function code is a symbol that performs a specific function. For example, we design a logic algorithm to satisfy the equation $a + b = c$, $x + y = z$, $c + z = w$ with three Bailey Inc.-supported two-input sum logic symbols. Among all the function codes (242 EA), function code 1 is the “Function Generator”, function code 2 is the “Manual Set Constant”, and function code 82 is “Segment Control”. Detailed descriptions are written in the operation manual. When one designs a logic algorithm, it is important to consider memory utilization. The RAM and NVRAM capacities are 256,000 bytes and 64,000 bytes, respectively. However, the design should allocate 163,248 bytes of RAM and 62,256 bytes of NVRAM, in case of the need to execute unknown work, e.g., an interrupt.

Table 2 shows the capacity of NVRAM, RAM, and the execution time of each function code. Equation (1) shows a general memory calculation to predict memory utilization.

$$T_b = FC \times NR \times n \quad (1)$$

where “ T_b ” is total bytes, “ FC ” is the function code number, “ NR ” is NVRAM bytes, and “ n ” is the number of repetitions.

Table 2. Memory utilization and execution time for selected function codes in terms of NVRAM, RAM, and processing latency.

Function Code	NVRAM (Bytes)	RAM (Bytes)	Execution Time (in μ sec)
1	46	88	38
2	12	40	17
82	64	260	0
100	40	144	185
242	84	338	300

In a DCS, function codes define the basic control logic and operational instructions executed by the processor in real time. Each function code corresponds to a specific control operation, such as arithmetic computation, logic evaluation, data handling, or communication. By analyzing function codes, it becomes possible to quantify the memory consumption (both NVRAM and RAM) and execution time associated with each operation. This information is critical for evaluating system performance, diagnosing potential bottlenecks, and designing effective overload control algorithms. Since processor resources are finite, especially in legacy or embedded systems, identifying high-load function codes enables targeted optimization, ensuring that the DCS operates reliably and efficiently under varying load conditions. In this case, our test platform, an INFI-90 DCS system, described in this section, consists of a PCU, OIS, and EWS [11]. At the same time, a PCU is a control node that controls and manages the actual processor of the INFI-90 system [3,11]. In particular, the control values for process control are determined by the algorithm preset in the memory of the PCU module, and these control values determine the operations of functions such as Alarm, Logging, Process Monitoring, Control, and Trend in the central console. In addition, the operator interface station (OIS), which provides all information on process control to the system operator, has the main functions of process control and operation status monitoring, and has functions such as alarm management, operation status, and operation records [4,12]. Finally, the EWS provides the operator with a logical function that supports the development of algorithms suitable for various processes of the power plant. It also supports real-time monitoring for each process.

Moreover, the INFI-90 DCS system supports a specific function code to check the microprocessor. For example, function code 82, same as segment control, groups subsequent blocks into a scan cycle executed at a specified rate and priority [5,12]. Eight segment controls can use each module, i.e., the MFP module. However, users do not know how the segments are made or how they can use the segments efficiently on the system, because the manufacture does not release that information. Therefore, users operate only one segment to check microprocessor utilization.

3. Proposed Overload Control Algorithm

Proposed NFP module consists of a microprocessor; combined NFP core, ROM, NVRAM, I/O expender bus, and serial port; and combined parallel DMA, AFP, and I/O section (See Table 1). Our proposed NFP module includes the NFP Core and AFP, while the Bailey Inc. MFP does not include them. This is the main difference between the two modules. First, the NFP module incorporates a combined compressed function processor (CFP), ROM, and NVRAM to overcome the overload problem, because this algorithm improves memory efficiency. However, the memory management algorithm is not integrated into the processor module due to the interrupt processing speed.

We investigated about overload control algorithm for the DCS using the proposed NFP module first. To know all the operations of the entire system, it is necessary to investigate the system level of the power plant system, and at the same time, it is necessary to first understand the structures of each memory and processor operating at the hardware level. After that, the power plant control principles and systems must be investigated in more detail.

Table 1 Comparison of hardware specifications between the MFP (multi-functional processor) module and the BRC (bridge controller card) module. The table highlights key differences in memory capacity—both NVRAM and RAM—while showing that both modules share the same microprocessor width and power consumption. The substantial increase in memory resources in the BRC module reflects a hardware-level response to growing system complexity and the need for improved performance in the DCS.

In the Figure 2, the dashed circle highlights a control block labeled SEGCRM within the automation architect environment, which is part of a DCS logic diagram. The arrow extending from the dashed circle indicates a zoom-in or logical tracing operation that transitions the viewer's focus to a magnified black background version of the same SEGCRM block for detailed inspection. In this magnified view, several numerical values are displayed along the right-hand side, representing various operational metrics. Most notably, the number 83.84 (highlighted in a red-colored box) denotes the processor utilization percentage, indicating that the controller's CPU is currently operating at 83.84% capacity. Additional values such as 0.25, 15, and 0.222 are likely related to input/output scaling, loop timing, or internal set points, but the 83.84% utilization is the critical parameter for overload diagnosis in this context.

In order to investigate more about reliability and safety issue, we designed the NFP core based on hardware that will increase the processing speed. In addition, Table 3 shows that the proposed CFP memory efficiency is exactly 50% better than that of the conventional memory management algorithm [5]. This is because the memory equation of the proposed CFP is

$$T_{MU} = 2(LN - 1)(SL + 1) \quad (2)$$

where " T_{MU} " is total memory usage, " LN " is leaf node number, and " SL " is symbol length at the CFP binary structure.

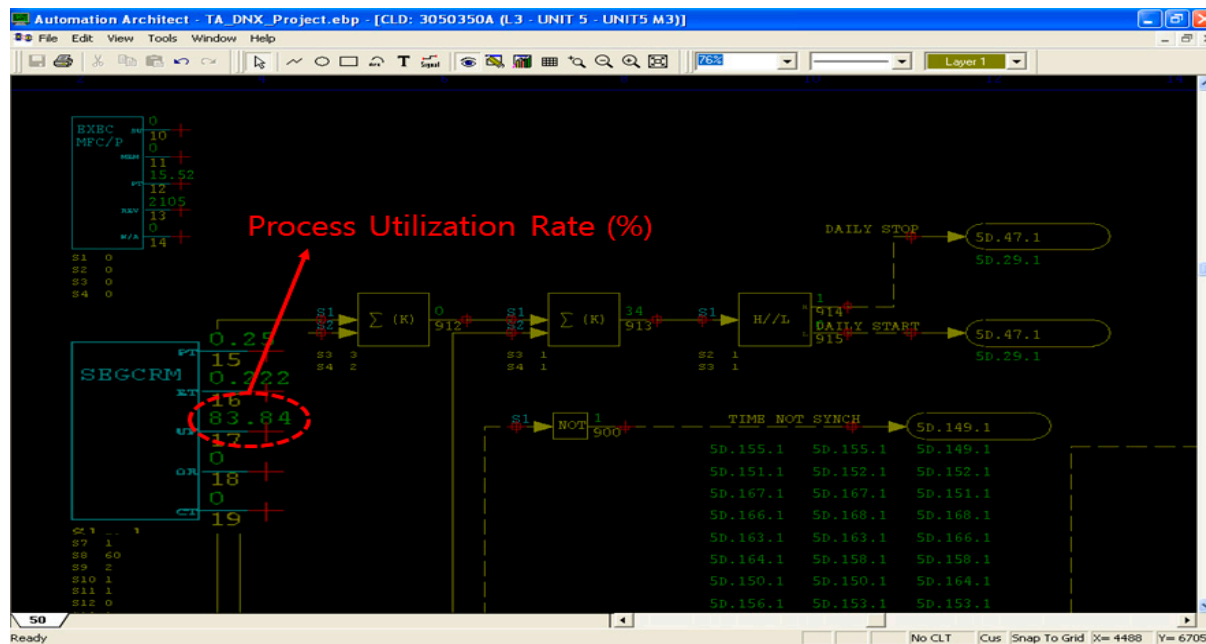


Figure 2. Screenshot processor utilization monitoring within a distributed control system (DCS) using the automation architect environment.

Table 3. Comparison of conventional and proposed compression algorithms in terms of memory utilization and execution complexity.

Equation	Conventional Compression Algorithm $2(n - 1)$	Proposed Algorithm $2(n - 1)(d + 1)$
Calculation	$2(n - 1) = 2(8 - 1)$	$2(n - 1)(d + 1) = [2(8 - 1)(3 + 1)]/8$
Total memory	14 byte	7 byte

In addition, hardware-based CFPs are designed according to clock speed: the faster the clock speed, the faster the processing speed.

Table 3 presents a quantitative comparison between a conventional compression algorithm and the proposed algorithm in terms of memory utilization and computational complexity. In resource-constrained environments such as distributed control systems, minimizing memory usage and reducing execution time are critical to preventing processor overload and ensuring real-time responsiveness. The table demonstrates that the proposed algorithm significantly reduces memory consumption—from 14 bytes to 7 bytes—by introducing a parameterized optimization factor, $(d + 1)$, in the equation. Additionally, the calculation complexity is reduced through normalization, making the algorithm more efficient for embedded control applications. This efficiency directly supports the development of lightweight and scalable control logic, which is essential for reliable and energy-efficient DCS operation.

In conventional memory management, the standard storage unit is the byte, whereas the proposed CFP (compressed function processor) utilizes the bit as the fundamental unit, allowing for more fine-grained memory efficiency. Additionally, the NFP (non-functional processor) module integrates a test bench circuit that includes an AFP (analog function processor) to enable virtual operation simulations. While DCS system logic algorithms may be optimally designed, unforeseen operational issues can still arise during runtime.

To address this, we designed the AFP using a flash ADC architecture, selected for its superior conversion speed compared to other ADC types such as pipeline and sigma delta ADCs. Although flash ADCs are known for high power consumption and extensive chip area usage [8,11], we implemented a low-power variant of the AFP to mitigate these

drawbacks. As shown in Table 4, the proposed AFP design achieves significant power reductions compared to conventional flash ADCs—approximately 15.38%, 42.37%, 57.20%, 63.86%, and 72.58%—for resolutions ranging from 4 to 8 bits, respectively [11,12]. These improvements result from the fact that the number of components required in the AFP is governed by the expression defined in Equation (3).

$$T_{Mp} = 2^{(N-1)} - 1 + C \quad (3)$$

where “ T_{MP} ” is the number of components used for the AFP architecture and “ C ” is the number of components used to design the proposed circuits.

Table 4. Comparison of the conventional flash ADC and the proposed analog function processor (AFP) in terms of comparator count, power consumption, and efficiency.

Bit	Conventional ADC		AFP		Efficiency (%)
	# of Comparator	Power (mW)	# of Comparator	Power (mW)	
4b	15	16.94	9	16.94	15.38
5b	31	23.01	17	23.01	42.37
6b	63	34.05	33	34.05	57.20
7b	127	57.46	65	57.46	63.86
8b	255	87.32	129	87.32	72.58

Table 4 presents a comparative analysis between a conventional flash ADC and the proposed analog function processor (AFP) across resolutions from 4 to 8 bits. While both architectures exhibit identical power consumption at each resolution level, the AFP achieves a significant reduction in the number of required comparators—up to 72.58% at 8 bits—resulting in substantial hardware efficiency. This reduction is critical in DCS environments, where minimizing circuit complexity directly contributes to improved scalability, reduced power density, and lower thermal stress. The ability of the AFP to maintain performance while reducing hardware overhead makes it highly suitable for real-time control applications, particularly in overload-prone or resource-constrained embedded systems. The AFP achieves identical power levels with significantly fewer comparators across resolutions from 4 to 8 bits, yielding efficiency improvements up to 72.58%. This reduction in hardware complexity contributes to enhanced power efficiency and scalability in embedded DCS environments.

The test conditions were a 3.3 V supply voltage, 1.65 V reference voltage, 1 GHz sampling rate, and 100 ns transient time [13]. As shown in Table 4, the proposed analog function processor (AFP) significantly outperforms the conventional flash ADC in terms of hardware efficiency, without compromising power consumption. The efficiency, defined as the percentage reduction in comparator count, is calculated using the following expression:

$$Efficiency (\%) = \left(1 - \frac{N_{AFP}}{N_{Conventional}} \right) \times 100 \quad (4)$$

where N_{AFP} is the number of comparators used in the AFP, and $N_{Conventional}$ is the number of comparators required by the conventional flash ADC.

This equation calculates the percentage efficiency gained by reducing the number of comparators in the AFP design compared to the conventional flash ADC. A higher efficiency value indicates a greater reduction in hardware complexity.

Figure 3a demonstrates a significant reduction in hardware complexity while maintaining functional equivalence. At the same time, Figure 3b shows a progressive increase in efficiency, reaching up to 72.58% at 8-bit resolution, highlighting the suitability of the AFP for resource-constrained, real-time control systems. These results visually confirm

that the proposed AFP architecture not only minimizes hardware complexity but also achieves substantial efficiency gains, validating its effectiveness for real-time, low-resource applications in the DCS.

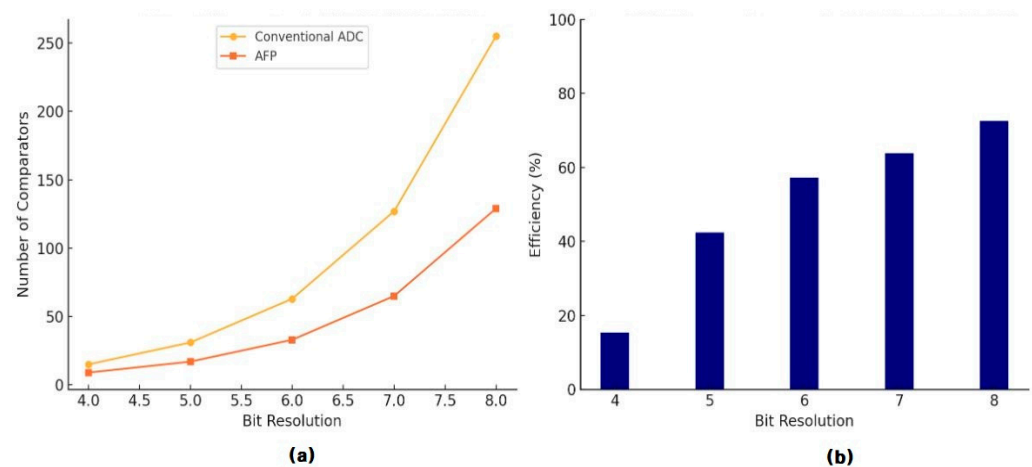


Figure 3. (a) Comparison of comparator counts required by the conventional flash ADC and the proposed AFP across different bit resolutions. (b) Efficiency improvement in the AFP relative to the conventional architecture, calculated based on comparator count reduction.

4. Optimal Utilization

Up to this point, optimal utilization has been discussed primarily at the hardware level, focusing on reducing comparator count and memory consumption through the proposed AFP architecture. However, to fully realize system-wide efficiency in DCS, it is essential to extend this concept to the system level. In this context, a newly designed MFP module is proposed and applied within the DCS framework to manage computational tasks more effectively. To evaluate the impact of this architectural enhancement, a simulation of the conventional DCS configuration was first performed as a baseline [9,14]. Following this, the proposed MFP module was integrated into the DCS, and a second simulation was conducted. The results of this simulation illustrate the performance gains achieved under the proposed architecture in terms of processor utilization, execution latency, and real-time responsiveness.

Figure 4a represents the conventional architecture, in which a single processing block (SEGCRM) handles all logic and computation tasks. This block outputs a processor utilization value of 83.84%, as shown on the display, indicating a high processing load. The centralized nature of this structure can lead to performance bottlenecks and increased risk of overload, particularly under complex or high-frequency operations. All computations are processed within one module, offering limited flexibility for scaling or optimization.

In contrast, Figure 4b depicts the proposed architecture, which decomposes the computational logic into multiple parallel SEGCRM blocks. Each block processes part of the overall logic independently, significantly reducing individual processor load. The outputs from these distributed blocks are then combined using a *sum logic unit*, as shown on the right side of Figure 4b. This summation process aggregates the results from each module, resulting in an overall processor utilization output of 57.2%, which is markedly lower than in the conventional case. The modular design enhances load balancing, improves response time, and provides a scalable path for future system expansion. Together, Figure 4a,b demonstrate how architectural restructuring—particularly through distributed computation and summation logic—can yield significant improvements in processor efficiency and system reliability.

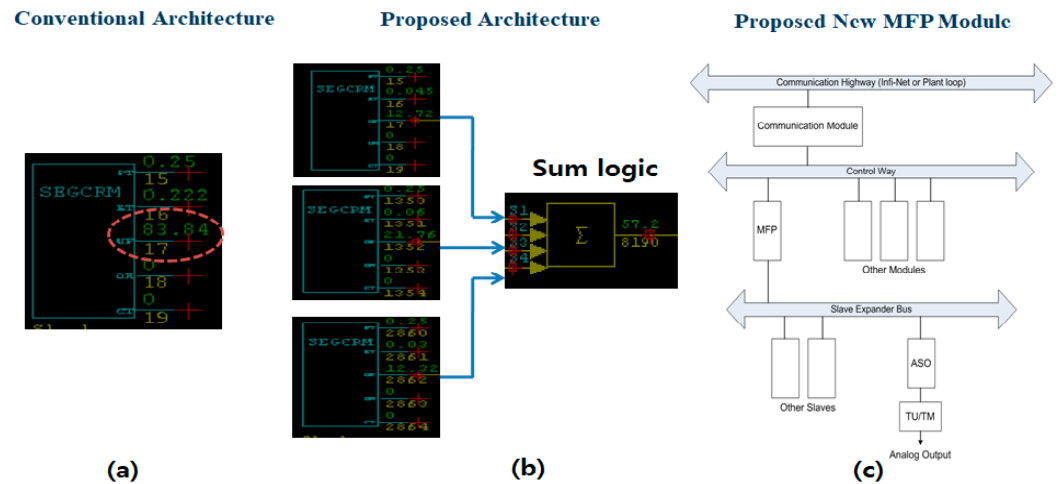


Figure 4. (a) Checking all distributed control system CPUs with single logic; (b) checking CPU and load by process using sum logic; (c) block diagram of proposed new MFP module.

As a result, the proposed architecture enhances both the reliability and efficiency of modern DCS, making it highly suitable for next-generation energy and automation systems where precision, responsiveness, and resource optimization are essential.

Figure 5 presents the simulation results of the proposed DCS architecture incorporating the newly designed MFP module. Figure 5a displays a real-time interface capturing key system metrics such as CPU utilization and memory usage. In this simulation, the CPU utilization is shown to be 12.195612%, and memory utilization is 643 bytes, both of which are significantly reduced compared to the conventional architecture, demonstrating the effectiveness of the optimized logic distribution and lightweight design. Figure 5b provides a detailed time-series performance graph along with tabulated control block data, showing stable and balanced processing loads across multiple loops and PCUs. The colored trend lines represent individual signal blocks and indicate minimal fluctuation and high reliability in execution under continuous operation. These results further validate the impact of the proposed architecture in reducing computational overhead while maintaining precise, real-time process control in distributed environments.

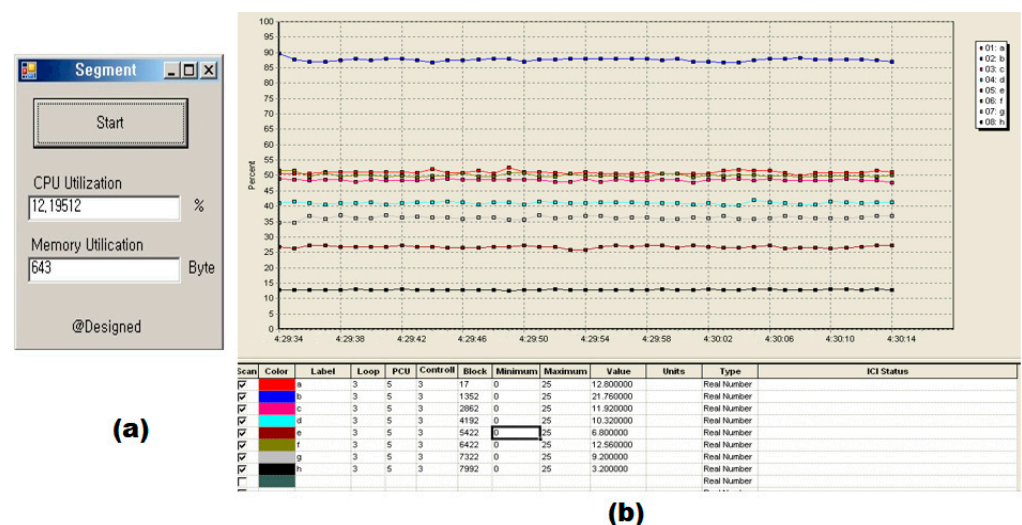


Figure 5. Appearance and CPU utilization of MFP module segment control designed for practical application for overload detection: (a) screen capture of segment including CPU and memory utilization; (b) screen capture of real-time CPU utilization on each process.

5. Topology and Overload Detection

INFI-90 DCS network architecture incorporates both ring and bus topologies. The ring topology is used to connect the PCU, EWS, and OIS, primarily for generator control operations. In contrast, the bus topology interconnects the control-way, I/O modules, and field devices via the field bus. In this study, we focus on evaluating the performance of the bus topology using two key communication protocols: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) and the Token Bus protocol. These protocols are considered favorable due to their enhanced compatibility with Internet-based communication systems compared to traditional field bus protocols. Based on the observed performance characteristics, we propose a novel overload detection algorithm that leverages the advantages of the optimal communication protocol identified in this evaluation.

5.1. Proposed Protocol

The major communication protocols used in bus topology are Carrier Sense Multiple Access with Collision Detection (CSMA/CD) and the Token Bus protocol [13,15]. CSMA/CD offers the advantage of low implementation cost, primarily due to its simple control mechanism and the lack of need for a centralized network management system within the DCS environment. However, CSMA/CD presents limitations in scenarios requiring sensitive or deterministic control, such as prioritizing data transmission among stations. Additionally, it does not guarantee real-time processing, which is critical in industrial control applications. The Token Bus protocol similarly benefits from low cost, but it suffers from a significant drawback, increased network load, as every transmission from a single node is broadcast to all nodes in the network. This can lead to congestion and reduced communication efficiency, particularly in large-scale distributed systems.

In order to validate of system topology, we investigate Network Animator (NAM), a visualization tool commonly used with the Network Simulator (NS2) to animate and analyze packet flow in network simulations [16]. It is widely used for educational and research purposes to visually verify the behavior of various network protocols and topologies [17,18]. Figure 6 presents a simulation environment captured from Network Animator (NAM), a visualization tool used in conjunction with NS2 for analyzing data transmission behavior in network protocols [19]. This simulation illustrates a bus topology configuration where nodes are organized into two rows. The upper row includes nodes labeled 0, 2, 4, and 6, while the lower row includes nodes 1, 3, 5, and 7. In particular, Node 9, positioned on the far left, is designated as the source or control node, responsible for initiating communication across the network. The horizontal line dividing the two rows symbolizes the shared communication medium or backbone bus, which is a defining feature of the bus topology.

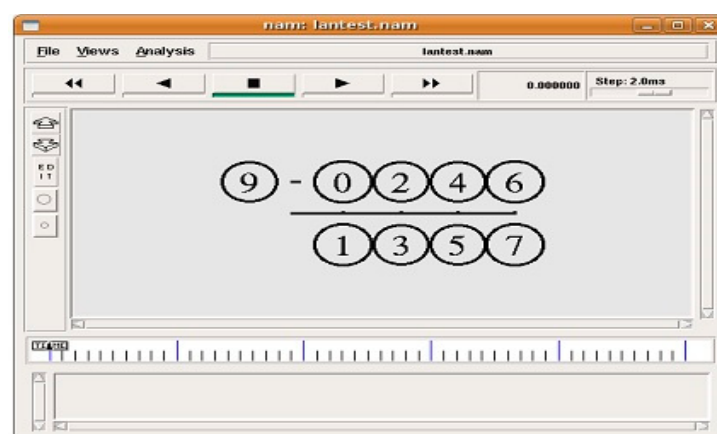


Figure 6. Experimental environment for measuring efficiency of protocol on Bus topology.

The arrangements of nodes and their sequential alignment are designed to demonstrate the dynamic behavior of token-based or contention-based protocols, such as Token Bus and CSMA/CD (Carrier Sense Multiple Access with Collision Detection) [20]. These protocols govern how nodes access the shared channel, manage transmission priorities, and handle collision scenarios. The simulation operates with a time step of 2.0 milliseconds, enabling detailed observation of packet exchanges and protocol timing. Such visual simulations are critical for evaluating real-time constraints, bandwidth utilization, and network load—factors that directly impact protocol selection in DCS and other industrial network environments.

To evaluate the performance of the system, a network topology simulation was conducted using NS2. In parallel, additional metrics and evaluation strategies were considered to gain deeper insights into the system's operational efficiency. The resulting performance outcomes and their implications are discussed in detail in Figure 7.

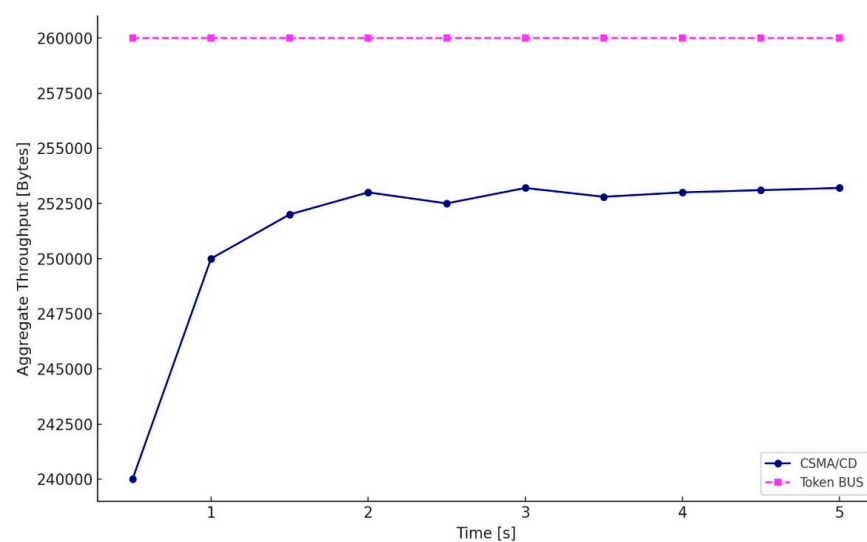


Figure 7. Throughput of CSMA/CD and Token Bus on Bus topology.

In this case, there are various metrics used for system performance evaluation, including throughput, latency, jitter, packet loss rate, CPU utilization, memory utilization, delay variation, bandwidth utilization, response time, and bit error rate [21,22]. Among these, we selected *throughput* as the primary metric to objectively evaluate the system's performance in a quantifiable manner.

Figure 7 illustrates a comparative analysis of aggregate throughput over time between two bus topology protocols: CSMA/CD (Carrier Sense Multiple Access with Collision Detection) and the Token Bus protocol. The *x*-axis represents time in seconds (ranging from 0 to 5 s), while the *y*-axis denotes aggregate throughput in bytes, indicating the total amount of data successfully transmitted across the network.

In the initial phase, both protocols experience a steep increase in throughput as network traffic begins to stabilize. The Token Bus protocol (represented by the pink line with square markers) shows a slightly faster convergence to maximum throughput compared to CSMA/CD (shown in blue with diamond markers). This suggests that the Token Bus protocol initializes more efficiently, possibly due to its deterministic token-passing mechanism, which avoids early collisions.

After approximately 0.5 s, both protocols reach a plateau where throughput remains relatively constant, indicating network saturation or stability. Token Bus maintains a marginally higher throughput throughout the simulation, though both protocols hover around the 260,000-byte mark. The consistency of the Token Bus throughput line also

suggests lower jitter and fewer retransmissions, while CSMA/CD shows slight fluctuations, likely due to its contention-based nature and the occurrence of collisions followed by back-off and retransmission.

The simulation was conducted by applying the definitions and criteria for the major key parameters. A total of 12 SEGCRM blocks, each corresponding to a unique segment of the function code execution, were considered. The individual CPU utilization values of these blocks were grouped into three aggregation units that aggregate the data of four adjacent SEGCRM blocks. These intermediate group-level utilization values were then aggregated using the final aggregation logic ($\sum k$) to calculate the overall system utilization. This hierarchical aggregation process not only verified the modularity of the monitoring framework but also showed that the accumulated segment-level utilizations can be efficiently combined to generate a reliable global metric that reflects the overall load conditions of the DCS.

Ultimately, the total number of segments that make up the final utilization is the total number of SEGCRM blocks (N), which is where each SEGCRM calculates the local CPU utilization.

In addition, the function code sequence ranges, which are the function code ranges that each segment is responsible for, are defined, and this is used as a basic parameter that defines how to divide the overall system logic.

For simulation, the definition and criteria for key parameters are applied, and the single usage values output from each SEGCRM are named Segment-Level CPU Utilization (%), and the set of these values constitutes the final utilization rate through the intermediate summation ($\sum k$). This is the criterion for setting the four hierarchical structures for how many segments one ($\sum k$) block includes, and the group size per summation block (M) is set. At this time, the monitoring time window (T) is set, and the system capacity constraints, which serve as the denominator criteria when calculating the total CPU/memory capacity limit ratio, are set. The simulation application result configured in this way derived a final system-wide CPU usage rate of 88.4%.

Although this study employs legacy fieldbus communication protocols, namely, CSMA/CD and Token Bus, for the simulation and evaluation of the proposed overload detection architecture, the practical deployment of distributed control systems (DCSs) increasingly demands compatibility with contemporary industrial ethernet standards. Protocols such as PROFINET, EtherCAT, and EtherNet/IP provide deterministic, low-latency, and high-throughput communication capabilities that align with the requirements of Industry 4.0 and time-sensitive automation environments.

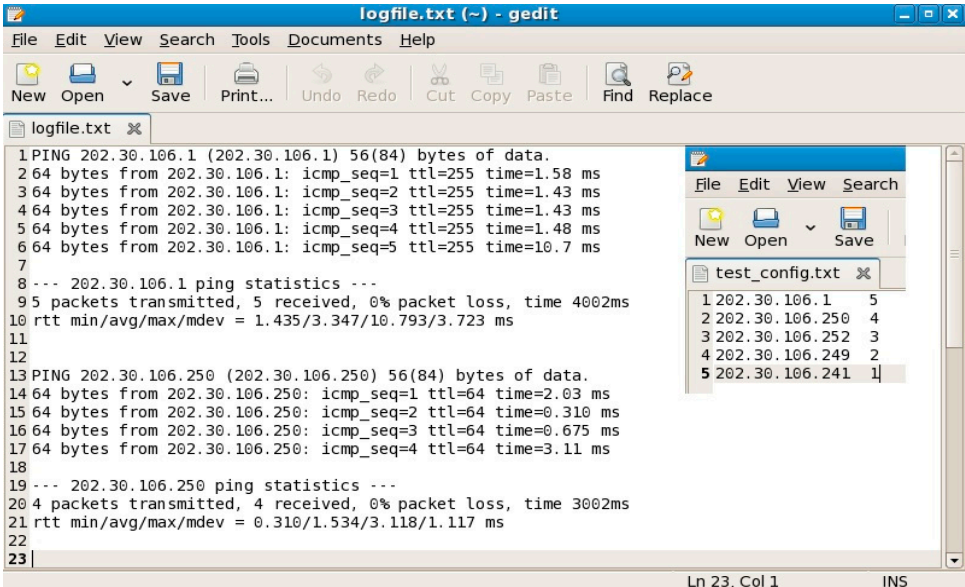
The proposed segmented monitoring framework, based on dynamic function code assignment and hierarchical aggregation of processor utilization, is inherently modular and communication protocol-agnostic. This architectural neutrality facilitates potential integration with real-time ethernet protocols through middleware abstraction or gateway-level semantic translation. For example, the cyclic synchronization and distributed clock mechanisms of EtherCAT are well aligned with the parallel execution model of segmented control blocks, allowing for each slave device to assume a localized monitoring role. Similarly, the producer-consumer model inherent in PROFINET communication can be utilized to distribute and aggregate segment-level load metrics across functional nodes in real time.

To ensure broader applicability, future research will investigate the implementation of the proposed architecture within time-sensitive networking (TSN)-enabled ethernet infrastructures, thereby enabling end-to-end validation of determinism, latency bounds, and real-time observability. This direction aims to extend the utility of the proposed overload detection and control optimization framework toward deployment in next-generation industrial networks characterized by strict timing constraints and decentralized process control.

5.2. Overload Detection Results

We propose the use of two protocols—CPSP (Centralized Periodic Status Ping) and TDLP (Token-based Distributed Load Polling)—both operating on the Token Bus protocol to enhance overload detection and system observability [23,24]. These protocols assist the operator in easily identifying the location of an overload when it occurs.

CPSP functions as a real-time query mechanism for monitoring the total CPU utilization of each sub-station, as shown in Figure 8. Since sub-stations operate at different frequencies depending on their task loads, CPSP sends a variable number of query requests to each station accordingly. For example, if Station A executes five operations and Station B performs three within a given time window, CPSP queries Station A five times and Station B three times, enabling fine-grained utilization tracking. Figure 7 presents the experimental results of CPSP in action. Under nominal conditions, the main station periodically transmits ping signals to sub-stations. In this scenario, Station A (IP: 202.30.106.1) received five status requests, while Station E (IP: 202.30.106.241) received only one, reflecting their respective operational loads.



```

logfile.txt (~) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace

logfile.txt x
1 PING 202.30.106.1 (202.30.106.1) 56(84) bytes of data.
2 64 bytes from 202.30.106.1: icmp_seq=1 ttl=255 time=1.58 ms
3 64 bytes from 202.30.106.1: icmp_seq=2 ttl=255 time=1.43 ms
4 64 bytes from 202.30.106.1: icmp_seq=3 ttl=255 time=1.43 ms
5 64 bytes from 202.30.106.1: icmp_seq=4 ttl=255 time=1.48 ms
6 64 bytes from 202.30.106.1: icmp_seq=5 ttl=255 time=10.7 ms
7
8 --- 202.30.106.1 ping statistics ---
9 5 packets transmitted, 5 received, 0% packet loss, time 4002ms
10 rtt min/avg/max/mdev = 1.435/3.347/10.793/3.723 ms
11
12
13 PING 202.30.106.250 (202.30.106.250) 56(84) bytes of data.
14 64 bytes from 202.30.106.250: icmp_seq=1 ttl=64 time=2.03 ms
15 64 bytes from 202.30.106.250: icmp_seq=2 ttl=64 time=0.310 ms
16 64 bytes from 202.30.106.250: icmp_seq=3 ttl=64 time=0.675 ms
17 64 bytes from 202.30.106.250: icmp_seq=4 ttl=64 time=3.11 ms
18
19 --- 202.30.106.250 ping statistics ---
20 4 packets transmitted, 4 received, 0% packet loss, time 3002ms
21 rtt min/avg/max/mdev = 0.310/1.534/3.118/1.117 ms
22
23

Ln 23, Col 1 INS

test_config.txt x
1 202.30.106.1 5
2 202.30.106.250 4
3 202.30.106.252 3
4 202.30.106.249 2
5 202.30.106.241 1
  
```

Figure 8. Screenshot of CPSP simulation results.

Figure 8 shows the output log from a terminal-based network monitoring tool, used here to validate the behavior of the CPSP (Centralized Periodic Status Ping) protocol in a DCS environment. The left pane displays raw ICMP (Internet Control Message Protocol) echo request results—commonly known as “ping” results—for two different sub-stations. The first block of results corresponds to Station A with IP address 202.30.106.1, where five ping requests were transmitted and successfully received, indicating 0% packet loss and consistent response times (minimum: 1.435 milliseconds; average: 3.347 milliseconds; maximum: 10.793 milliseconds). The second block shows results for Station B with IP address 202.30.106.250, where four pings were transmitted with similarly stable round-trip times, also achieving 0% packet loss.

On the right side of the interface, the file *test_config.txt* is shown, which appears to define the operational parameters of each sub-station under the CPSP protocol. This configuration file maps each IP address to a specific polling count (e.g., Station A receives five queries, Station E only one), aligning with the variable-frequency querying mechanism described in the protocol design. These results confirm the CPSP system’s capability to selectively and periodically monitor CPU utilization across distributed nodes with minimal

communication overhead and high reliability, making it suitable for real-time performance diagnostics in DCS networks.

In parallel with CPSP, the Token-based Distributed Load Polling (TDLP) protocol was utilized to capture live data packets from each sub-station during query execution. The collected packet data were analyzed and converted into real-time memory utilization metrics, offering additional insight into system resource consumption. To support this functionality, a specialized network monitoring tool—Sub-Station Traffic Management (SSTM)—was employed to monitor packet flows across the DCS bus topology. SSTM enables the capture of TCP/IP packets traversing the network and provides real-time visual feedback of communication between client and server nodes. As illustrated in Figure 8, SSTM visualizes the packets entering each sub-station, thereby facilitating continuous observation of network load and memory usage in real time. This monitoring mechanism is critical for detecting potential overload conditions and ensuring reliable system performance.

Figure 9 displays a packet-level capture performed using Wireshark, a widely adopted network protocol analyzer [25], to monitor communication traffic within the proposed DCS network environment. The capture highlights the data exchange between source node 202.30.106.245 and multiple destination addresses, including IP: 68.180.217.31 and the broadcast domain IP: 202.30.106.255, across multiple protocols such as NBNS (NetBIOS Name Service) and TCP. This analysis provides critical insights into the behavior of the system under real-time conditions.

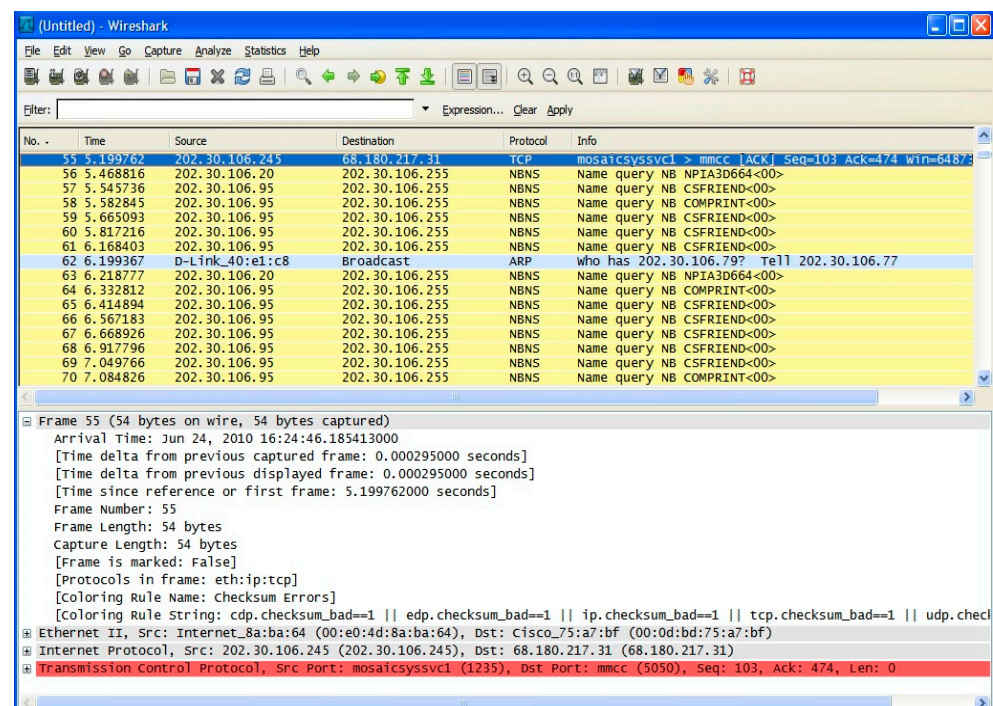


Figure 9. Screenshot of TDLP simulation results.

The lower pane of Figure 9 shows detailed protocol dissection for Frame 55, indicating the use of the transmission control protocol (TCP) over port 1235 (e.g., source) and port mmcc (e.g., destination). The TCP segment contains control flags (ACK) with sequence and acknowledgment numbers, reflecting a completed or in-progress transmission handshake. Notably, the packet is marked with checksum errors (as highlighted in the red-colored line), which could signify degraded link quality, buffer overflow, or packet corruption—conditions that are especially important to detect in control system environments. The captured traffic also includes NBNS name queries, which serve to

resolve hostnames within the local network and validate broadcast-level communication among sub-stations. This set of data, collected using the TDLP protocol and visualized via SSTM, confirms that system nodes are actively transmitting identifiable traffic and can be individually tracked for utilization metrics. Thus, the figure supports the broader aim of this study by demonstrating how packet-level inspection assists in overload detection, traffic profiling, and real-time monitoring of distributed DCS nodes.

The conventional logic algorithm demonstrated limited diagnostic capability in identifying the root cause of system overload, as the segment control function code merely represented cumulative CPU utilization without isolating contributions from individual functional components (as shown in Figure 2). To address this limitation, we propose a novel overload detection logic algorithm leveraging the non-functional processor (NFP) segment, which enables fine-grained visibility into the internal load distribution of the DCS.

The development of the proposed overload detection logic proceeds through the following structured methodology:

- i. The Token Bus protocol is employed as the underlying communication mechanism to enhance both the determinism and stability of internal DCS communications, which is critical for maintaining synchronization and reliability in real-time control environments.
- ii. DCS architectures utilize a variety of function codes, each performing discrete control operations. When instantiated, each function code is dynamically assigned a unique block number. For example, if a particular function code is invoked ten times, it is assigned block numbers sequentially from 1 to 10.
- iii. These assigned block numbers are systematically partitioned into eight logical groups based on their expected execution times. This classification facilitates balanced load analysis and enables the mapping of grouped block executions to discrete processing pathways.
- iv. Each group is subsequently interfaced with a dedicated NFP segment, allowing for real-time monitoring and isolation of computational loads associated with each group.
- v. Finally, the outputs of the NFP segments are aggregated via three summation logic units to compute total utilization across grouped segments, enabling the detection of overload conditions with higher specificity and temporal accuracy (as shown in Figure 9).

Figure 10 illustrates the architecture of the proposed overload detection logic algorithm using a segmented monitoring structure based on block sequence numbers within a DCS. The diagram demonstrates how execution blocks—each associated with a specific range of sequence numbers—are monitored individually and aggregated for system-level resource analysis. Each box on the left represents a segment processor responsible for collecting CPU and memory utilization data from a specific range of function code sequence numbers. For instance, one segment processes sequence numbers 1–100, another covers 301–400, and so on. These ranges are representative of function code instances dynamically assigned during DCS operation. The segment control windows display real-time utilization statistics, such as CPU usage percentage and memory usage in bytes. Four segment groups are logically combined using summation units ($\sum k$) to compute intermediate totals. Each summation block aggregates the utilization data from four adjacent segments, effectively reducing monitoring overhead while preserving resolution. These partial results are then fed into a final summation unit, which computes the total system-wide CPU and memory utilization across all monitored segments.

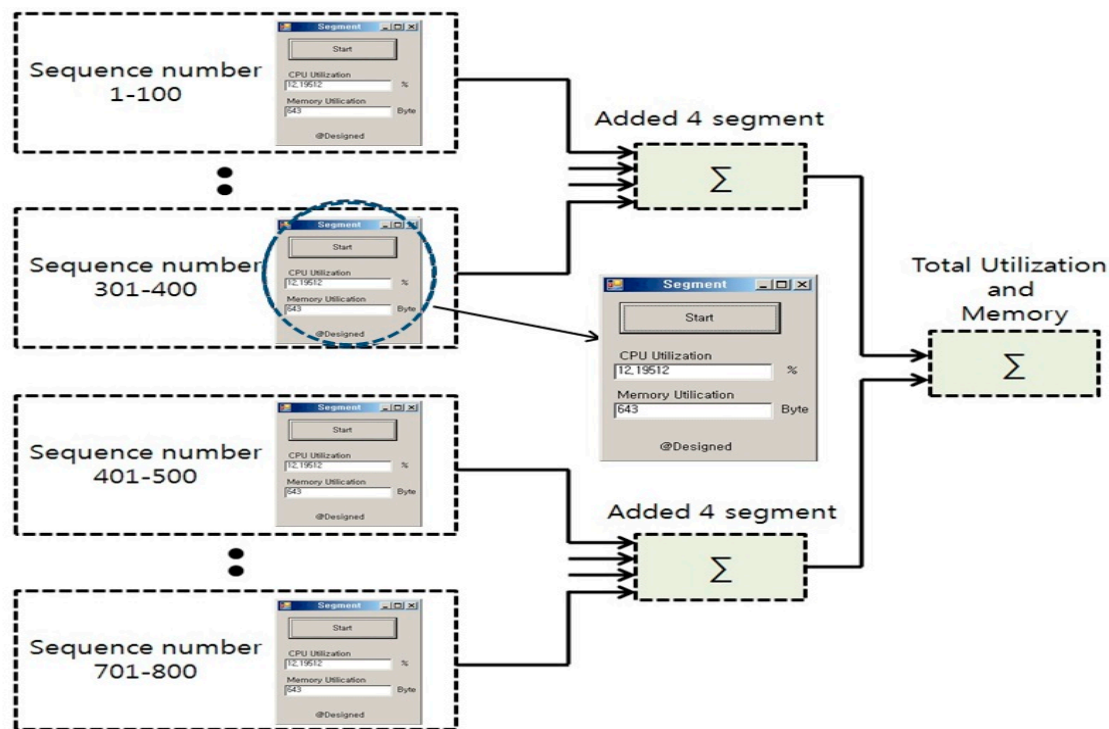


Figure 10. Proposed overload detection algorithm of each sequence block and segmentation to finalize DCS.

This hierarchical structure enables scalable, modular overload detection. By partitioning execution blocks and associating them with segmented logic, the proposed system can isolate overload conditions with greater granularity, enabling early detection and mitigation of potential performance degradation. Furthermore, the use of summation logic allows for efficient aggregation without centralized bottlenecks, making the method well suited for large-scale or real-time industrial automation systems.

Figure 11 presents a detailed schematic of the proposed optimal utilization rate computation mechanism in a DCS using segmented CPU load monitoring and hierarchical summation logic. The diagram illustrates multiple instances of SEGCRM blocks, each representing localized control segments responsible for executing specific function code sequences within the system. Each SEGCRM block outputs an individual CPU utilization value (e.g., 10.45%, 13.56%), reflecting the localized processing load at that segment. These values are routed through a parallel summation network, where a primary summation block (Σk) consolidates the individual utilizations into a collective intermediate utilization value, in this case shown as 57.2%. This intermediate value corresponds to a logically grouped subset of system segments that have been balanced for execution time and resource consumption.

The output of the first summation block is further processed by a second summation unit (Σk), which aggregates all segment group totals into a final global utilization rate, displayed here as 88.4%. This hierarchical architecture not only facilitates modular scalability but also supports real-time visibility into system load distribution. The red annotations labeled “Optimal Utilization Rate” highlight the ability of this design to monitor resource consumption at a granular level while providing a comprehensive system-level assessment. By identifying and summing only the relevant load contributions, this architecture enables more effective overload detection and supports dynamic load balancing strategies, thereby ensuring that system performance remains within optimal operational thresholds. The final utilization rate serves as a comprehensive performance indicator, capturing the aggregated

contributions from all individual segments into a single, optimized metric. The observed simulation results closely aligned with the anticipated outcomes, thereby validating the structural soundness and predictive reliability of the proposed framework.

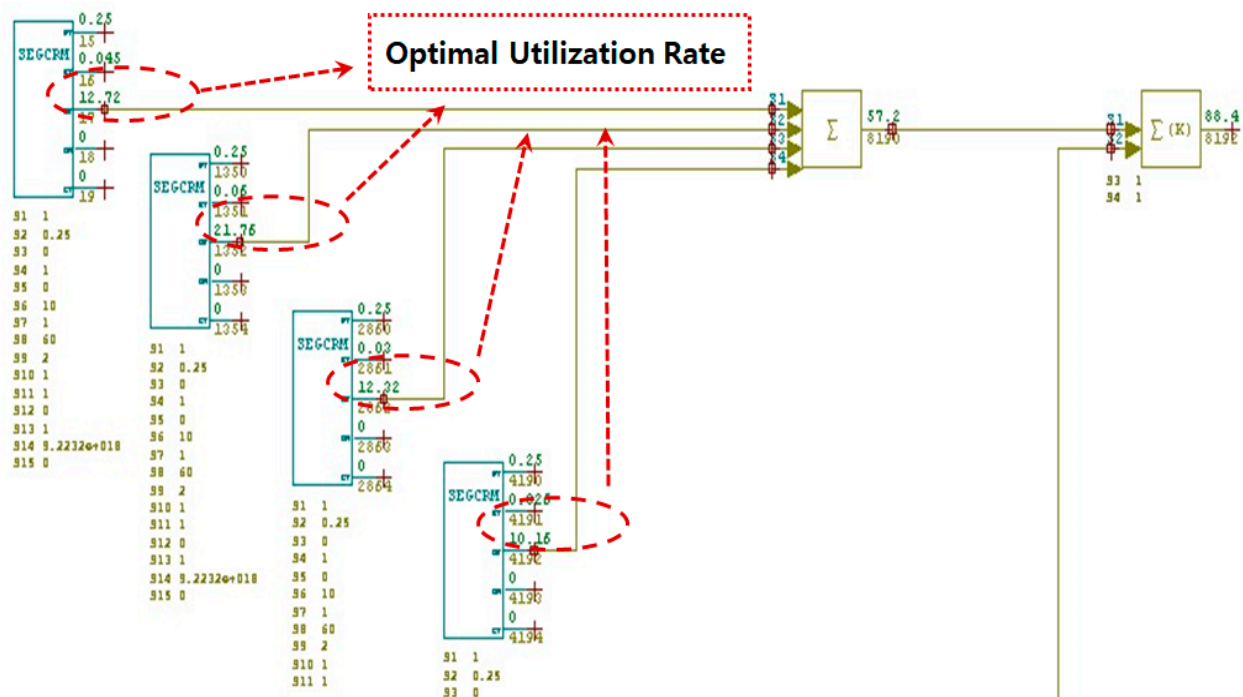


Figure 11. Simulation result on optimal utilization rate calculation overload detection system.

These findings demonstrate that the system can effectively allocate computational resources across segmented function blocks, facilitating optimal load balancing and improved execution efficiency. Moreover, the results suggest that the proposed DCS optimization strategy holds significant relevance for a range of application domains, including energy management, production scheduling, and real-time digital signal processing, where resource efficiency and system responsiveness are critical to operational success.

In order to apply these simulation results to a safety-critical system, we attempted objective evidence for application in actual power plants. First, we compared and reviewed two existing overload detection algorithms. The first is Fixed Threshold Alarm System (FTAS), which generates an alarm only when a specific CPU usage threshold (e.g., 85%) is exceeded, and has the simple but low sensitivity characteristic [26,27]. The second is Trend-Based Utilization Detection (TUD), which analyzes the CPU change rate over time to detect a sudden increase. It is an algorithm with high sensitivity but potential for overreaction [28].

In order to objectively compare these two types of overload detection algorithms and benchmarking, we added two existing flash ADC structures: standard full-comparator flash ADC [29] and subranged Flash ADC [30]. Standard full-comparator flash ADC is fast, but power consumption and the number of comparators are very large, while subranged flash ADC reduces the number of comparators while maintaining speed (middle-segment structure), reducing the number of components by 30–40%.

In the end, in order to objectively prove all of these, the analysis was conducted based on CPU usage, memory usage, execution delay time, number of ADC comparators (8b), ADC power (mW), hardware efficiency, etc.

Table 5 presents a comprehensive benchmarking analysis comparing the proposed DCS architecture—integrating NFP, CFP, and AFP modules—with two representative legacy approaches across critical performance indicators, including CPU utilization, memory

footprint, computational latency, and hardware-level efficiency. The proposed system exhibits substantial advancements, achieving up to a 31.7% reduction in processor load and a 72.58% improvement in hardware efficiency, while also enhancing the granularity and sensitivity of overload detection mechanisms. These findings underscore the architectural superiority and applicability of the proposed solution for high-reliability, real-time control scenarios in resource-constrained industrial environments.

Table 5. Comprehensive benchmarking analysis comparing the proposed DCS architecture with two different architectures.

Bench Marking Metric	FTAS	TUD	Proposed Method (NFP + CFP + AFP)	Improvement Rate
CPU Utilization (%)	FTAS: 83.84	TUD: 76.5	57.2	+31.7%
Memory Usage (RAM, bytes)	CBLA: 338	RLE: 296	260	−23.1%
Execution Latency (μs)	RLE-based: 300	Symbol Table: 255	185	−38.3%
ADC Comparator Count (8-bit)	Full-Flash: 255	Subranged: 160	129	−49.4%
ADC Power Consumption (8-bit, mW)	Full-Flash: 87.32	Subranged: 87.32	87.32	Same

6. Conclusions

We proposed a novel overload detection logic algorithm for DCSs, aimed at improving system stability and resource efficiency. While the algorithm showed promising results in simulation, it cannot yet be directly applied to legacy systems currently deployed at the Tae-An Thermoelectric Power Plant. However, the operating company has initiated investments in next-generation DCS technologies, anticipating future system upgrades. In preparation, we developed a simulation-based test bench to evaluate the algorithm's performance in advance of the new DCS installations. Future work will focus on developing an enhanced logic processor architecture and validating its effectiveness under realistic operational scenarios using this test environment. This paper presents an innovative and systematic approach to overload control and system reliability enhancement in DCS architectures, with particular emphasis on applications in thermoelectric power generation. The proposed real-time overload detection algorithm, when integrated with a modular MFP, demonstrates substantial improvements in fault resilience, load distribution, and computational efficiency. Analytical modeling and simulation-based validation confirm that the system effectively mitigates overload conditions and minimizes performance degradation in dynamically varying load environments.

The algorithm's applicability to real-world thermoelectric plant operations was demonstrated through the simulation test bench, where it proactively managed critical system parameters, ensured efficient energy use, and reduced the likelihood of system failures. The modular test environment validated both the scalability and compatibility of the proposed solution, proving its adaptability to both existing infrastructures and future DCS platforms. Unlike conventional overload protection methods, the proposed system offers flexible integration with legacy systems while supporting the scalability needed for future expansion.

Overall, the findings of this study highlight the potential of the proposed framework to contribute to the development of next-generation, self-regulating DCS architectures that address the growing demands of modern power grids. By delivering real-time fault detection, dynamic load balancing, and optimized resource allocation, the proposed solution enhances not only operational reliability and efficiency but also the sustainability of thermoelectric power plants. Based on the proposed hierarchical overload detection framework, future research can explore multi-objective optimization strategies that simultaneously balance CPU load, memory usage, and network traffic across distributed control segments. In addition, integrating this architecture with edge computing environments will enable

low-latency, real-time decision-making closer to the physical system. These directions are expected to enhance scalability, responsiveness, and overall operational efficiency in large-scale industrial automation systems.

While the current simulation setup replicates system behavior based on INFI-90 specifications and typical function code assignments from operational DCSs (e.g., Taeon Power Plant), actual runtime data were not used due to confidentiality constraints. Future work will involve empirical validation using field data and potential deployment in a hardware-in-the-loop testbed. These advancements represent a critical step toward the realization of robust, adaptive, and intelligent control systems capable of meeting both current and future energy challenges.

Funding: This research was supported by Hallym University Research Fund, 2025 (HRF-202501-001).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: A preliminary version of this paper was presented at the 2010 Spring Conference of the Korean Society for Internet Information (KSII). This revised version has been updated and expanded at the honorable request of the KSII program committee. The author gratefully acknowledges H. Jabbar, S. Beak, and B. Hieu for their contributions in preparing the figures during their work on this sponsored project in collaboration with Korea Western Power Co. and the advanced intelligent research (AIR) laboratory, directed by T. Jeong.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

DCS	distributed control system
AFP	analog function processor
CFP	compressed function processor
EWS	engineering workstation
MFP	multi-functional processor
TUD	Trend-based Utilization Detection
RLE	Run-Length Encoding Compression
NFP	non-functional processor
ADC	analog-to-digital converter
BRC	bridge controller card
CPSP	Centralized Periodic Status Ping
ICMP	Internet Control Message Protocol
FTAS	Fixed Threshold Alarm System
CBLA	Conventional Byte-Level Allocation
NBNS	NetBIOS Name Service
TDLP	Token-based Distributed Load Polling

References

1. Ting, H.; Wu, Z.; Li, D.; Jihong, W. Active disturbance rejection control of large-scale coal fired plant process for flexible operation. In *Modeling, Identification, and Control for Cyber-Physical Systems Towards Industry 4.0*; Academic Press: Cambridge, MA, USA, 2024; pp. 385–413.
2. Hemeida, A.M.; El-Sadek, M.Z.; Younies, S.A. Distributed control system approach for a unified power system. In Proceedings of the 39th International Universities Power Engineering Conference, Bristol, UK, 6–8 September 2004; pp. 304–307.
3. Available online: <https://new.abb.com/control-systems/service/offerings/extensions-upgrades-and-retrofits/abb-system-evolution/symphony-harmony-system-evolution> (accessed on 5 May 2025).

4. Canonico, R.; Sperli, G. Industrial cyber-physical systems protection: A methodological review. *Comput. Secur.* **2023**, *135*, 103531. [CrossRef]
5. Paul, B.; Sarker, A.; Abhi, S.H.; Das, S.K.; Ali, F.; Islam, M.; Islam, R.; Moyeen, S.I.; Badal, F.R.; Ahamed, H.; et al. Potential smart grid vulnerabilities to cyber-attacks: Current threats and existing mitigation strategies. *Heliyon* **2024**, *10*, e37980. [CrossRef] [PubMed]
6. Sudarshan, V.; Seider, W.D.; Patel, A.J.; Oktem, U.G.; Arbogast, J.E. Alarm rationalization and dynamic risk analyses for rare abnormal events. *Comput. Chem. Eng.* **2024**, *184*, 108633. [CrossRef]
7. Xiao, L.; Sun, W.; Chang, S.; Lu, C.; Jiang, R. Research on the construction of a blockchain-based industrial product full life cycle information traceability system. *Appl. Sci.* **2024**, *14*, 4569. [CrossRef]
8. Beak, S.; Lee, W.; Hieu, B.V.; Choi, S.; Lee, E.; Jeong, T. Control Logic Based on Optimal Communication Protocol for Investing of the Overload on DCS Systems. In Proceedings of the 2010 Spring Conference, Jeju, Republic of Korea, 24–26 June 2010; Korean Society for Internet Information: Seoul, Republic of Korea, 2010.
9. Beak, S.; Lee, W.; Jeong, T. A study on Optimal Scalable Logics and Overload of DCS Main Controller Project. In *Korea Western Power Co. Project Final Report*; Korea Western Power Co.: Taejeon, Republic of Korea, 2010.
10. Jabbar, H.; Song, Y.; Jeong, T. RF Energy Harvesting System and Circuits for Charging of Mobile Devices. *IEEE Trans. Consum. Electron.* **2010**, *56*, 247–253. [CrossRef]
11. Jeong, T. Energy Charging and Harvesting Circuits Design in Bluetooth Environment for Smart Phone. *IET Sci. Meas. Technol.* **2013**, *7*, 201–205. [CrossRef]
12. Jabbar, H.; Lee, S.; Jeong, T. Energy Harvesting Technique by using Novel Voltage Multiplier Circuits and Passive Devices. *IEICE Trans. Electron.* **2013**, *E96.C*, 726–729. [CrossRef]
13. Fairhurst, G. Carrier Sense Multiple Access with Collision Detection (CSMA/CD). 2004. Available online: <https://www.erg.abdn.ac.uk/users/gorry/course/lan-pages/csma-cd.html> (accessed on 5 May 2025).
14. Vargas, J.V.C.; Souza, J.A.; Hovsapian, R.; Ordonez, J.C.; Chalfant, J. ESRDC ship notional baseline medium voltage direct current (MVDC) architecture thermal simulation and visualization. In Proceedings of the GCMS'11: Proceedings of the 2011 Grand Challenges on Modeling and Simulation Conference, Hague, The Netherlands, 27–30 June 2011; pp. 150–160.
15. Ivanov, S.; Herms, A.; Lukas, G. Experimental validation of the ns-2 wireless model using simulation, emulation, and real network. In Proceedings of the Communication in Distributed Systems-15. ITG/GI Symposium, Bern, Switzerland, 26 February–2 March 2007; pp. 1–12.
16. Espina, J.; Falck, T.; Panousopoulou, A.; Schmitt, L.; Mülhens, O.; Yang, G.Z. Network topologies, communication protocols, and standards. In *Body Sensor Networks*; Springer: London, UK, 2014; pp. 189–236.
17. Barrett, C.L.; Drozda, M.; Marathe, A.; Marathe, M.V. Analyzing interaction between network protocols, topology and traffic in wireless radio networks. In Proceedings of the 2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003, New Orleans, LA, USA, 16–20 March 2003; IEEE: Piscataway, NJ, USA, 2003; Volume 3, pp. 1760–1766.
18. Mohapatra, S.; Kanungo, P. Performance analysis of AODV, DSR, OLSR and DSDV routing protocols using NS2 Simulator. *Procedia Eng.* **2012**, *30*, 69–76. [CrossRef]
19. Sen, S.; Roy Choudhury, R.; Nelakuditi, S. CSMA/CN: Carrier sense multiple access with collision notification. In Proceedings of the MobiCom'10: Proceedings of the Sixteenth Annual International Conference on Mobile computing and Networking, Chicago, IL, USA, 20–24 September 2010; pp. 25–36.
20. Briscoe, B.; Brunstrom, A.; Petlund, A.; Hayes, D.; Ros, D.; Tsang, I.-J.; Gjessing, S.; Fairhurst, G.; Griwodz, C.; Welzl, M. Reducing internet latency: A survey of techniques and their merits. *IEEE Commun. Surv. Tutor.* **2014**, *18*, 2149–2196. [CrossRef]
21. Salah, K.; El-Badawi, K.; Haidari, F. Performance analysis and comparison of interrupt-handling schemes in gigabit networks. *Comput. Commun.* **2007**, *30*, 3425–3441. [CrossRef]
22. Primadianto, A.; Lu, C.N. A review on distribution system state estimation. *IEEE Trans. Power Syst.* **2016**, *32*, 3875–3883. [CrossRef]
23. Cintuglu, M.H.; Mohammed, O.A.; Akkaya, K.; Uluagac, A.S. A survey on smart grid cyber-physical system testbeds. *IEEE Commun. Surv. Tutor.* **2016**, *19*, 446–464. [CrossRef]
24. Banerjee, U.; Vashishtha, A.; Saxena, M. Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection. *Int. J. Comput. Appl.* **2010**, *6*, 1–5. [CrossRef]
25. Mahmood, A.; Hameed, R.A.A.; Farooq, H. An intelligent overload detection and response mechanism for industrial control systems. *IEEE Access* **2021**, *9*, 54761–54775.
26. Thomesse, J.P. Fieldbus technology in industrial automation. *Proc. IEEE* **2005**, *93*, 1073–1101. [CrossRef]
27. Obaidat, M.S.; Boudriga, N.A. *Handbook of Green Information and Communication Systems*; Academic Press: Cambridge, MA, USA, 2013.
28. Razavi, B. *Principles of Data Conversion System Design*; IEEE Press: Piscataway, NJ, USA, 2013.

29. Issariyakul, T.; Hossain, E. *Introduction to Network Simulator NS2*; Springer: Berlin/Heidelberg, Germany, 2012. [[CrossRef](#)]
30. Marwedel, P. *Embedded System Design*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2011.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.