Industrial Knee-jerk: In-Network Simultaneous Planning and Control on a TSN Switch

Zeyu Wang
School of Software and BNRist
Tsinghua University
Beijing, China
ycdfwzy@gmail.com

Jingao Xu
School of Software and BNRist
Tsinghua University
Beijing, China
xujingao13@gmail.com

Xu Wang
Global Innovation Exchange
Tsinghua University
Beijing, China
xu_wang@mail.thu.edu.cn

Xiangwen Zhuge
School of Software and BNRist
Tsinghua University
Beijing, China
zgxw18@gmail.com

Xiaowu He
School of Software and BNRist
Tsinghua University
Beijing, China
horacehxw@gmail.com

Zheng Yang[*]
School of Software and BNRist
Tsinghua University
Beijing, China
hmilyyz@gmail.com

ABSTRACT

Rapid advances in programmable network devices catalyzed the development of in-network computing, which is foreseen as a key enabler to empower the intelligence of production lines and mechanical arms in Industry 4.0. Various pioneering approaches have demonstrated the significant benefits of moving simple yet delay-sensitive industrial *control* tasks performed by servers to network switches. However, our detailed field study at a top-tier auto glass factory reveals that current practice fails to achieve a real-time and deterministic intelligent decision closure as leaving those complex yet essential *planning* tasks still on edge or cloud. In this paper, we design and implement a brand-new industrial switch, named Netopia, on a commercial Zynq platform through software and hardware co-design. Netopia enables *planning* and *control* to simultaneously perform on a network switch during communication. At the core of Netopia are three simple yet effective modules - a determinism guarantee mechanism, a computing acceleration scheme, and a packet deterministic forwarding framework that work hand-in-hand to ensure mechanical arms obtain intelligent control commands with low and deterministic latency. Comprehensive evaluations in industrial environments demonstrate that Netopia achieves an average end-to-end intelligent decision latency of 3.0ms with a jitter < 0.4ms, reduced by > 86% over existing works.

CCS CONCEPTS

• **Networks → In-network processing**; **Programmable networks**; • **Computer systems organization → Real-time system architecture**.

[*]Zheng Yang is the corresponding author.

KEYWORDS

Time-Sensitive Networking, In-Network Computing, Industrial Control

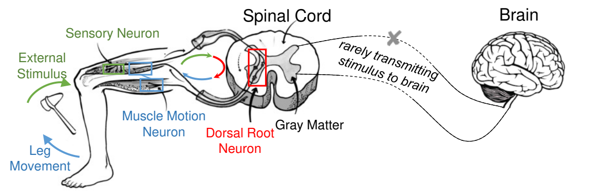
ACM Reference Format:
Zeyu Wang, Jingao Xu, Xu Wang, Xiangwen Zhuge, Xiaowu He, and Zheng Yang. 2023. Industrial Knee-jerk: In-Network Simultaneous Planning and Control on a TSN Switch. In *The 21st Annual International Conference on Mobile Systems, Applications and Services (MobiSys '23), June 18–22, 2023, Helsinki, Finland.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3581791.3596836

1 INTRODUCTION

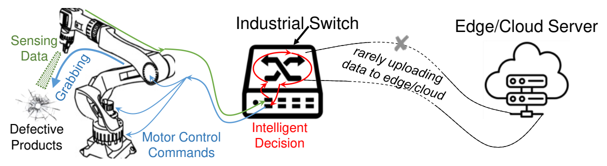
In the era of Industry 4.0 (*a.k.a.*, the 4th Industrial Revolution), the intelligence of production lines and the autonomy of mechanical arms have gradually become highly prized goals for manufacturing factories[29]. The tasks for mechanical arms are gradually evolving from simple relay logic to intelligent decisions[10, 46, 47]. Compared to mechanical arms with task-specific and pre-programmed operating instructions on traditional lines, autonomous arms could make intelligent and agile decisions according to production or environmental information, such as defective product detection and grabbing[8, 9, 17], obstacle avoidance and emergency braking[48, 52]. Such an intelligence lift will bring a production paradigm shift and significantly improve efficiency and safety.

According to our field study at a worldwide top-tier auto glass manufacturer (§2.1), a mechanical arm's intelligent decision closure can be abstracted as *planning* and *control* two modules as illustrated in Fig.2. The former exploits the arm's sensing data to infer some intelligent tasks (e.g., object classification, detection) using neural networks, and on this basis, plan the arm's subsequent high-level operation trajectory (e.g., 20cm forward, 30° rotation). The latter translates the planned trajectory into low-level motor commands based on the arm's status messages using control algorithms (e.g., Inverse Kinematics[58], PID[62]). Eventually, these commands will be distributed to each motor.

With the increasing interconnectivity of industrial devices, cloud or edge robotics have hit the mainstream in industrial robotics research[49]. Status-quo solutions either (i) offload both the planning and control modules to a centralized cloud or edge server[42] as illustrated in Fig.2a; or (ii) load the low-level control module onto



(a) Illustration of human knee-jerk reflex.



(b) Illustration of our proposed industrial knee-jerk.

Figure 1: An analogy between (a) human knee-jerk and (b) our proposed industrial knee-jerk. Our design aims to enable mechanical arms (like human arms and legs) to rapidly perform some urgent and intelligent tasks through the control of nearby industrial switches (analogous spinal cords), reducing the dependence on distant edge/cloud servers (as human brains).

a network switch but keeps the high-level yet complex planning module on cloud/edge (Fig.2b). In general, using a more powerful server accelerates neural network inference in planning, and moving control to switch further reduces the end-to-end latency[29].

Although the early signals are positive and the industrial wheels are in motion, we find current practice fails to enable mechanical arms to periodically obtain updated control commands with *low* and, equally important, *deterministic* latency (i.e., newer arms require $\leq 1ms$ update period), where real-time and determinism are crucial for a broader spectrum of industrial applications[44]. Grand challenges are three-fold:

• **Network heterogeneity exacerbates data propagation delay.** Unlike industrial networks (e.g., PROFINET[64], EtherCat[57]) deployed between mechanical arms and switches, cloud/edge servers and switches are typically connected by low-cost yet low reliability standard Ethernet (Fig.2). Transmitting sensing data (e.g., successive images, high-frequent radar point clouds and IMU samples) across those heterogeneous networks causes considerable delays (§2.2-C1).

• **Computational resource dynamics degrades latency determinism.** Despite the different network distances from industrial environments, edge and cloud infrastructures typically rely on operating systems (OS) or OS-level virtualization technologies[66] to facilitate hardware resource management. However, the computational resources allocated to a specific task are thus varied and influenced by OS or CPU scheduling, making it difficult to guarantee the task with deterministic processing latency[32] (§2.2-C2).

• **Background traffic overload impairs control commands dispatch.** In practical industrial environments, each network switch initially has to forward a large volume of background traffic (e.g., production information, surveillance videos), and transmitting sensing data to edge/cloud further burdens the switch. Such an overloaded data flow overwhelms those critical control packets for arms, affecting their real-time and deterministic distribution. (§2.2-C3). In a nutshell, leaving the planning module on cloud/edge sets a strong barrier to achieve a real-time and deterministic intelligent decision closure, spanning from data transmission, computation, and traffic forwarding.

Recently, two new opportunities have arisen in overcoming the above challenges: (i) With the advent of programmable network devices, switches can perform not only pure packet forwarding but computations as well. This trend led to the birth of a new computational paradigm called *in-network computing*[26]; and (ii) the use of novel embedded computing platforms with hierarchical arithmetic units (e.g., Xilinx Zynq[25], Nvidia Tegra[65]) enables lightweight devices to conduct relatively complex tasks.

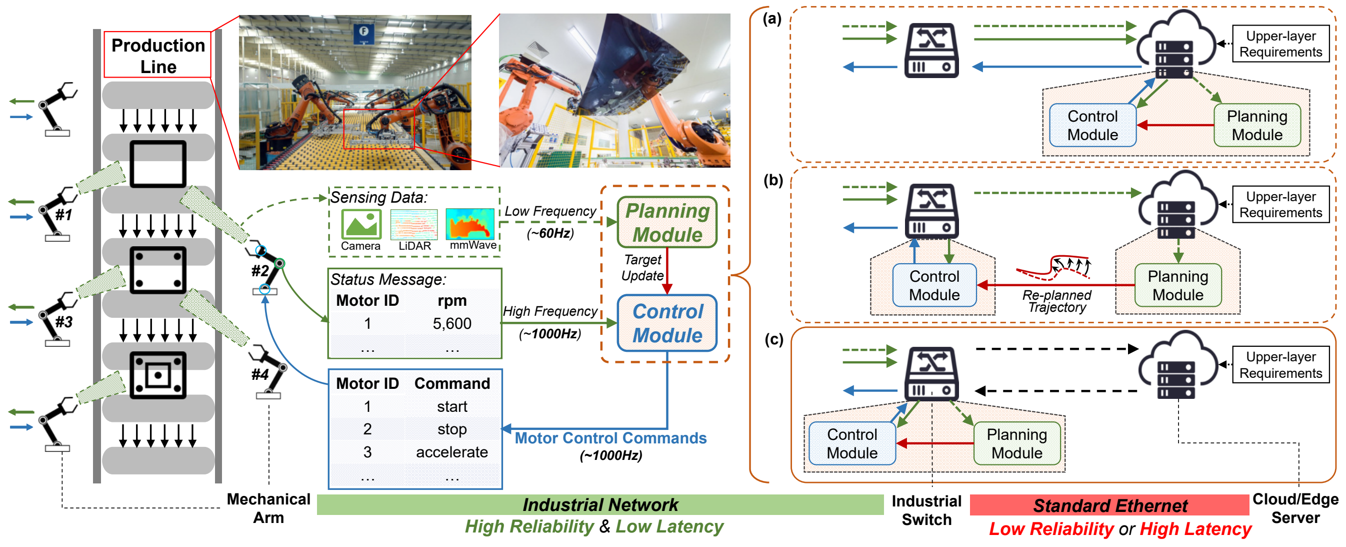
Motivated by the above challenges and opportunities, we design and implement a brand-new network switch, named **Netopia**, on a Zynq platform following a software and hardware co-design paradigm. Netopia enables an industrial switch to perform both the *planning* and *control* modules without offloading each of them to edge/cloud, reducing the dependence on powerful servers. Just like human physiological reflexes such as knee-jerk[61] and hand-retraction[45] as shown in Fig.1a, where the behavior of skeletal muscle, after stimulus, is controlled only by proximal neurons at *spinal cord* rather than distal central nervous at *brain*[61]; our work (Fig.1b), similarly, aims to splits the original long, distant closures into smaller ones and realizes low-latency and deterministic calculations for some simple yet urgent tasks during communication.

Simultaneously moving *planning* and *control* to industrial network poses many challenges that are addressed in this work: (i) how to ensure delay determinism of the end-to-end intelligent decision closure when diverse sub-tasks are diffused among heterogeneous arithmetic units; (ii) how to improve the real-time performance of those complex tasks in the planning module (i.e., neural network inference plus trajectory planning); and (iii) how to achieve deterministic forwarding of critical control commands. Overall, the design and implementation of Netopia excel in three aspects:

• **Delay Determinism Guarantee**. Before loading specific *planning* and *control* tasks, we first introduce a delay determinism guarantee mechanism to Netopia, spanning hardware resource isolation, software process management, and reliable data interaction, making any tasks running on Netopia delay deterministic (§4.1).

• **Task Computing Acceleration**. We design an integrated hardware and software neural network acceleration framework, which significantly reduces the delay of intelligent task inference in *planning* by effective neural network module decomposition and task re-assignments between low-level logic blocks and high-level computational cores (§4.2).

• **Packet Deterministic Forwarding**. We implement Time-Sensitive Networking (TSN) complying with IEEE 802.1AS [2] and IEEE 802.1Qbv[1] standards in the communication layer of Netopia, enabling Netopia to reserve dedicated time slots for critical industrial control data. Benefiting from this, mechanical arms could reliably receive updated control packets with deterministic latency. Additionally, as TSN is compatible with industrial networks and standard



Status Message:

Motor ID	rpm
1	5,600
…	…

Motor ID	Command
1	start
2	stop
3	accelerate
…	…

Figure 2: An illustration of a practical production line and comparison of current practice for intelligent decision tasks. (a) Traditional centralized solutions (e.g., Baseline-I[42] on IJCA'18), where both the *planning* and *control* modules are offloaded to a cloud or edge server. (b) A recent work with in-network *control* yet still leaves the *planning* on the server (i.e., Baseline-II[29] on NSDI'22). (c) Our work, Netopia, with in-network simultaneous *planning* and *control*.

Ethernet, the deterministic forwarding of control packets would not affect Netopia's plug-and-play property (§4.3).

We fully implement Netopia on the latest Zynq UltraScale+ platform[25] through software and hardware co-design. Comprehensive experiments are carried out on two public mechanical control datasets and in real industrial environments, using 28.11 GB sensing data and industrial network traffic with over 1,500 object grabbing tests. We compare Netopia with two state-of-the-art (SOTA) industrial control systems, Baseline-I (IJCA'18[42]) and Baseline-II (NSDI'22[29]), and the experiment results show that Netopia achieves an 100% defect detection and grabbing success rate, outperforming comparative approaches by 33% and 28%. The average end-to-end intelligent decision closure latency of Netopia is 3.0*ms* with a jitter < 0.4*ms*, reduced by >86% over related works.

In summary, this paper makes three contributions.

(1) We design and implement Netopia, as far as we are aware of, the first industrial switch that makes both the *planning* and *control* modules compatible with *in-network computing*. Netopia empowers mechanical arms to make agile decisions with determinism, thus driving the intelligence of production lines.

(2) We propose several technologies, spanning from delay determinism guarantee, task computing acceleration, and packet deterministic forwarding, in Netopia to enable mechanical arms to obtain intelligent control commands with low and deterministic latency.

(3) We extensively evaluate the performance of Netopia and two comparative systems on public datasets and in real industrial environments. The results demonstrate Netopia's superior performance.

Contribution to the community. First, we systematically study existing industrial planning-control systems and reveal their fundamental limitations based on our field study in a typical manufacturing industry, which would help to uncover new research issues on industrial networks for the community. Second, we make Netopia's prototype implementation publicly available[1]. Netopia can serve as a platform for the research about *in-network computing* and also a brand-new switch for the deployment of industrial intelligent decision systems.

2 BACKGROUND AND MOTIVATION

We first introduce the mechanical arm's intelligent decision problem according to our case study in a top-tier auto glass manufacturer. We then explain the limitations of current practice and present the system goals in designing Netopia.

2.1 Mechanical Arm's Intelligent Decision

We present a snapshot of a real glass production line in Fig.2. Take the glass defect detection, a highly prized task throughout the whole production process in the manufacturer, as an example; some mechanical arms on the line (i.e., arms #2 and #4) are expected to not only passively complete fixed operations (e.g., glass bending, painting, gluing), but autonomously leverage sensors (e.g., camera, radar) to detect defects on glass using neural networks. And once a piece of glass is defective (e.g., dimensional non-compliance, internal cracks), these arms will directly grab it onto a recycling line for re-producing rather than continue processing it.

Compared to traditional solutions where manually distinguish defective glass at the end of a production line, such an intelligent lift could save labor costs while greatly improving production efficiency and yield rate as numerous unnecessary operations on defective products are eliminated.

Problem statement. We abstract an intelligent decision closure as *planning* and *control* two modules. As depicted in the left part of Fig.2, an arm periodically uploads sensing data (e.g., images, point

[1]https://github.com/MobiSense/Netopia

clouds) to the *planning* module and its status messages to the *control* module, respectively.

Note that *control* and *planning* run in parallel due to the huge gap between their operating frequency: the *control* module continuously adjusts each motor's commands at motor response frequency (i.e., around 1,000Hz adopted by the factory's newer mechanical arms[13]) through control algorithms (e.g., PID), to make the arm's motion match a target trajectory. The *planning* module, at the meantime, leverages a neural network to detect defects and track products (e.g., at 60Hz camera rate), and further plans the arm's subsequent operation trajectory. The latest planned trajectory, thereafter, will be served as the new target for the *control* module.

What is a feasible solution? Let t_p and t_c be the delay of the *planning* and *control* module, respectively. Specifically, t_p consists of sensing data upload (⇢ in Fig.2), planning task computation, and target trajectory update (→) delay. And t_c is a also combined delay of status messages upload (→), algorithms operation, and control commands forwarding (→). In general, both low and deterministic t_p and t_c are crucial for an arm:

• Industrial arms' motors require $< 1ms$ update period (i.e., an ultra-fast 1000Hz response frequency), indicating t_c *should be deterministically less than 1ms* in any scenarios. A longer t_c causes a motor to repeatedly execute the cached wrong commands and even emergency braking[13], which impairs the *control* performance and affects production quality.

• As for the *planning* delay, a smaller t_p contributes a more accurate and effective planned trajectory as the conveyor belt moves during calculation. On the other hand, t_p *cannot exceed the inter-frame interval* (i.e., $16.67ms$ for 60Hz camera rate) to prevent blocking of computation requests or even frames input order errors[33].

2.2 Limitations of Current Practice

We build a production line testbed in the glass factory to conduct our field study. We then re-implement two SOTA industrial intelligent control systems, Baseline-I (offloading both *planning* and *control* to edge) and Baseline-II (offloading *planning* to edge and *control* to switch), and evaluate their performance by conducting over 200 defective glass detection and grabbing tests (setup detailed in §5). The overall performance is exhibited in Fig.3a. As seen, neither of the current practice could achieve low or deterministic t_p or t_c within an acceptable range under practical industrial network conditions (i.e., network load > 50%). We dig into the underlying reasons and find the challenges are three-fold:

C1: Considerable data transmission delay. Offloading *planning* to edge/cloud has to simultaneously upload mechanical arms' sensing data (e.g., continuously captured frames). However, the standard Ethernet deployed between industrial switch and edge/cloud server suffers from a lower reliable and higher delayed data transmission performance compared to those sophisticated industrial networks (e.g., PROFINET[64], EtherCat[57], Modbus[59]) on production lines. What's worse, there is also protocol isolation between industrial networks and standard Ethernet, making it difficult to directly transmit data packets from one to the other. Such a drawback further exacerbates the packet conversion (e.g., analysis, re-packing, and forwarding) delay on the switch.

To validate our analysis, we measure the average delay for transmitting a 1920×1080 frame captured by the arm to a server under different network loads. We further categorize the delay into packet conversion (i.e., from PROFINET to standard Ethernet in the factory) on the switch and data transmission on Ethernet two parts[2]. The results are shown in Fig.3b. We observe that both systems produce high (i.e., the sum of those two parts $> 35ms$) and fluctuating delays in all scenarios. The server will blindly wait for over two rounds of acceptable t_p ($16.67ms$) before computing, leaving huge room for improvement. Additionally, the packet conversion on the switch contributes nearly two-thirds of the total delay, which cannot be easily addressed by a simple network upgrade (e.g., adopting advanced 10-Gigabit Ethernet).

C2: Highly dynamic computation latency. Edge and cloud infrastructures typically rely on conventional time-sharing operating systems (TSOS, e.g., Linux, Unix) or OS-level virtualization technologies (e.g., Docker[56]) to facilitate hardware resource management and reduce development costs. Such an operation cannot guarantee the processing latency of a specific task is deterministic as the resources allocated to it are dynamically varied and influenced by OS or CPU scheduling[32]. Although existing solutions leverage real-time operating systems (RTOS) to eliminate the effects for industrial applications[30], it increases the average waiting delay for each task, and merely brings a minor boost to those machine learning models running on GPU[5].

We set up an edge server and measure the latency of each task in *planning*, including two optional network backbones (i.e., VGG16 and ResNet-50) for defect detection (on GPU) and one trajectory planning (TP) algorithm (on CPU), running on a TSOS (i.e., conventional CentOS[53]) and RTOS (i.e., upgraded CentOS with Xenomai[67] patch), respectively. The results are shown in Fig.3c. In accordance with our analysis, the task processing delay varies significantly on TSOS, leaving nearly half of the defect detection latency over acceptable t_p ($16.67ms$). Moreover, we find the leverage of RTOS barely influences lifting the latency determinism of those neural networks (on GPU), compared with the planning algorithm (on CPU). The above results also reveal that although one can additionally deploy costly industrial networks to connect switches and an edge server (e.g., around \$10,000 to function) for the lowest transmission latency, the highly dynamic computation latency will soon become a newer bottleneck.

C3: Unreliable control packet forwarding. In industrial network topology, each switch is exploited to forward a large volume of background traffic (e.g., production information, surveillance videos) among connected devices (e.g., arms, other switches, or edge/cloud servers). Additionally, transmitting sensing data or high-frequent status messages to cloud/edge would further burden the traffic forwarding on switch. Such an overloaded data flow overwhelms control packets for arms, resulting in excessive queuing delays, even inaccurate forwarding, of those critical control packets.

We evaluate the control packet forwarding performance in different network load settings. Specifically, we measure each control packet's forwarding delay (i.e., the time a packet spends traversing the switch) and forwarding success rate (i.e., the ratio a target arm could receive the packet within the $1ms$ motor response cycle). The

[2]We omit the $< 1ms$ propagation delay on PROFINET.

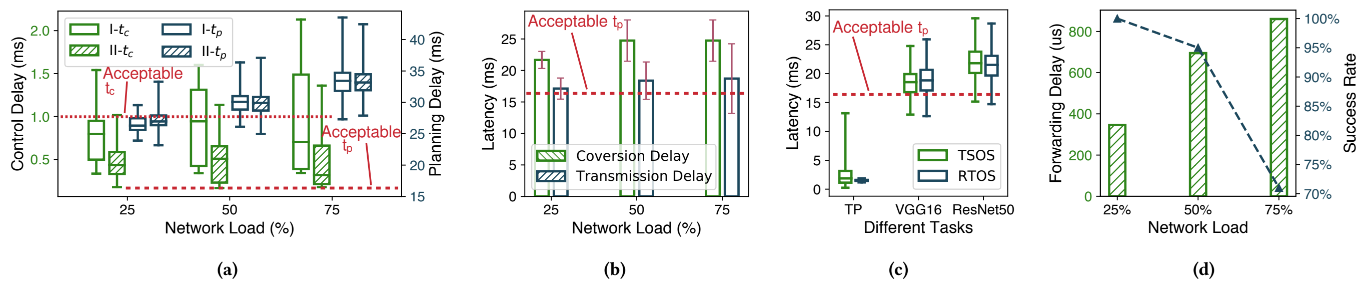


Figure 3: Limitations of current practice based on our field study. (a) Overall performance of two status-quo solutions. (b) Considerable data transmission delay. (c) Highly dynamic computation latency. (d) Unreliable control packet forwarding.

results are depicted in Fig.3d. As seen, the forwarding success rate drops below 75% when network loads > 75% with an average queuing delay spiking to 1.2ms, exceeding the acceptable t_c. Evidently, such a lower success rate and higher forwarding delay fail to meet the arm's ultra-fast response frequency.

Lessons Learned. To enable arms to obtain intelligent control commands with low and deterministic latency, we find three dimensions could be enhanced:

(i) On the architecture front, according to C1, we can design a brand-new industrial switch that simultaneously conducts both urgent *planning* and *control* on it, thus thoroughly eliminating the data transmission delay and uncertainty.

(ii) On the implementation front, motivated by C2, we should accelerate tasks computing and ensure deterministic latency through handcrafted software and hardware co-design, bypassing those uncertain OS- or CPU-level resource allocation and task scheduling.

(iii) On the network front, following C3, the switch needs to support a prioritized and deterministic forwarding of those critical control packets.

2.3 System Goals

Netopia takes a solid step forward in solving the above issues and thus enhancing the intelligence of production lines. We list the system goals below:

Goal 1: Plug and play. Netopia should be implemented as a general plug-in switch for industrial deployments, exposing well-packaged yet non-dedicated ports to end-devices for connection. This allows devices to benefit from Netopia without re-developing specific network protocols (§5.3).

Goal 2: Portability. Netopia should provide functionality and resource abstractions of the next-generation computing platform. This allows a broader scope of *planning* tasks (neural network with diverse backbones, layers, etc.) and *control* algorithms to take advantage of Netopia (§5.3).

Goal 3: Efficiency. Netopia should effectively reduce the end-to-end latency and ensure determinism, spanning data transmission, task calculation, and packet forwarding (§5.2).

3 SYSTEM OVERVIEW

We leverage the latest commercial Zynq UntralScale+ MPSoC (abbreviated as MPSoC), a heterogeneous computing platform launched by Xilinx[25], to implement Netopia through software and hardware co-design. We briefly introduce the MPSoC platform, and then present the system architecture.

3.1 Zynq Platform Primer

Fig.4 illustrates the hierarchical computing resources provided by MPSoC. As seen, MPSoC consists of a processing system (PS, for software development) and user-programmable logic (PL, for hardware design) two modules. The PS features a 64-bit Cortex-A53 quad-core processor (4*A-Core) and a Cortex-R5 dual-core real-time processor (2*R-Core). The four A-Cores are typically centralized and scheduled by a Linux OS, such as PetaLinux[20] and Debian[54]. The two R-Cores, designed for real-time application, are typically scheduled by a RTOS. The PL provides programmable logic blocks, digital signal processing (DSP), etc., for hardware design. Benefiting from the above versatile computational resources, MPSoC has been foreseen as a key enabler for next-generation applications such as 5G Wireless[3, 75], advanced driving assistance system (ADAS)[36], industrial Internet-of-Things[44, 71].

MPSoC is able to handle packet forwarding and complex task calculation simultaneously. However, we find simply migrating *planning* and *control* onto MPSoC without an elaborate software and hardware co-design could not achieve delightful performance, in terms of the end-to-end intelligent decision closure delay and determinism (§5.4). There are still three challenges that ought to be addressed: (i) how to ensure end-to-end delay determinism when hierarchical and heterogeneous units are all involved in the computation for both *planning* and *control*; (ii) how to realize real-time inference of neural networks in the *planning* module; and (iii) how to achieve deterministic forwarding of critical packets in the *control* module.

3.2 Netopia's Architecture

Netopia tackles the above challenges through software and hardware co-design. Fig.4 sketches the architecture of Netopia. Specifically, the design of Netopia can be abstracted as *communication* and *computation* two layers:

• On the network communication front, Netopia implements a complete network protocol stack and thus forwards industrial data traffic (i.e., background traffic) similar to a conventional switch, and an A-Core (i.e., #A4) is leveraged for the general switch configuration. In addition, Netopia receives arms' sensing data and

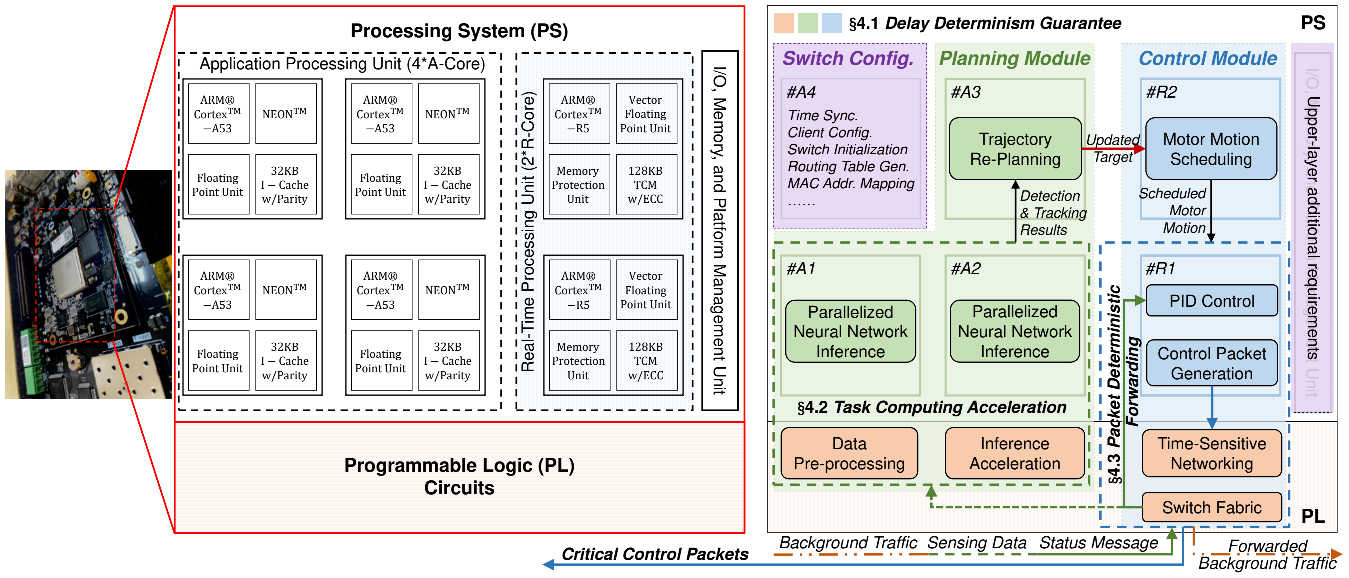


Figure 4: An exhibition of our proposed Netopia switch (left) based on a Xilinx Zynq UntraScale+ MPSoC (middle), and Netopia's architecture (right).

status messages, and distributes control packets back to each arm at motor response frequency (i.e., 1000Hz) after computation. Netopia implements a *Packet Deterministic Forwarding* framework, ensuring each arm to obtain control packets with deterministic latency (§4.3). The entire communication layer is implemented on the PL.

• On the computation front, both the *planning* and *control* modules are accomplished through the collaboration of PS and PL. Before loading specific tasks to the MPSoC platform, Netopia introduces a *Delay Determinism Guarantee* mechanism through computing resource abstraction, core isolation, and reliable data interaction among diverse heterogeneous units (§4.1). This allows (*i*) any tasks running on the platform to enjoy deterministic latency; and (*ii*) the sub-tasks on each arithmetic unit could be flexibly adjusted with respect to the modification of high-level tasks without re-designing the entire system architecture, improving Netopia's portability.

As for the specific workflow, Netopia first leverages a *Task Computing Acceleration* scheme to accelerate the complex neural network inference (e.g., defect detection) on logic blocks and two A-Cores (i.e.. #A1 and #A2) using sensing data (§4.2). Based on the inference results, the *planning* module leverages a re-planning algorithm to calculate an arm's subsequent working trajectory for grabbing a defective glass on an A-Core (i.e., #A3) and send it to the *control* module. Meanwhile, the *control* module (*i*) schedules each motor's motion according to the high-level working trajectory on an R-Core (i.e., #R2); and (*ii*) generates motor control packets through PID control on #R1 based on the scheduled motor motion and arm's current status. Those critical control packets, eventually, will be distributed back to the arm.

4 DESIGN AND IMPLEMENTATION

4.1 Delay Determinism Guarantee

Distinguished from commercial applications, industrial applications impose higher real-time and deterministic requirements (e.g., sub-millisecond *versus* tens of milliseconds latency and jitter). Although MPSoC's hierarchical computational resources empowers lightweight devices to perform some complex tasks, the heterogeneity of arithmetic units (i.e., logic blocks on PL, A-Cores and R-Cores on PS), and the intermediate data interaction among them, still challenge the real-time and deterministic manner of industrial systems. Simply running an OS on MPSoC for centralized resource management and scheduling will again leave Netopia with the same drawbacks as edge/cloud solutions (§2.2).

In Netopia, we tackle the above challenge and ensure the computation delay determinism of any tasks on Netopia through handcrafted and detailed resource and computation management. We categorize the end-to-end delay determinism into the determinism of intermediate data interaction and sub-task processing two parts, and propose *Tri DMA* to ensure the former while *Isolated A-Core* and *Bare-metal R-Core* for the latter, bypassing the uncertain OS scheduling.

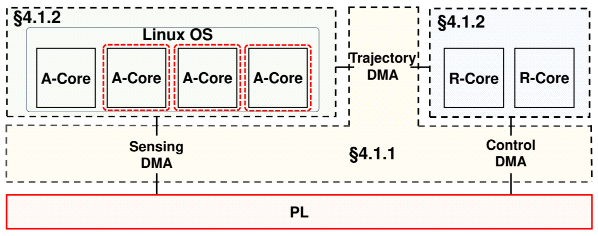


Figure 5: Illustration of Determinism Guarantee.

4.1.1 ***Tri DMA for Intermediate Data Interaction.*** We leverage the direct memory access (DMA) technique to take over the intermediate data interaction process. In general, DMA allows different arithmetic units to access the main system memory and transfer data in-between independently of the CPU (Cortex A53

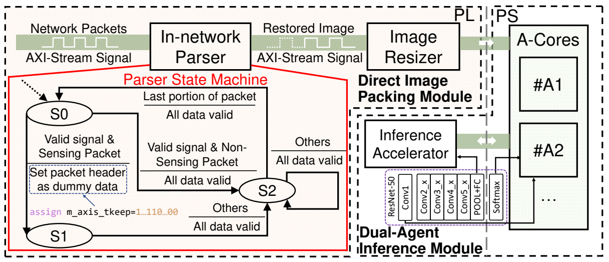


Figure 6: Workflow of Task Computation Acceleration

and R5). Taking the data interaction from an A-Core to an R-Core (and vice-versa) as an example, without DMA, when the R-Core is executing programmed input/output, it is typically fully occupied for the entire duration of the read/write operation, and thus receives the data from A-Core with frequent blocking. On the contrary, with DMA, the R-Core first initiates a data transfer channel with A-Core, then it does programmed operations while the transfer is still in progress.

Following the above insight, we propose three different types of DMA bridging all heterogeneous arithmetic units as illustrated in Fig.5: (*i*) the *Sensing DMA* bridges PL to A-Core to transfer sensing data in the *planning* module; (*ii*) the *Trajectory DMA* updates the planned trajectory from A-Core to R-Core for communications between the *planning* and *control* modules; and (*iii*) the *Control DMA* exchanges status messages and control commands between PL and R-Core. Generally, different from existing network-based solution such as PL-PS ethernet interface[23] and OpenAMP[37], our proposed Tri DMA enables each kind of critical data flow to own its exclusive transmission channel, ensuring the intermediate data interaction determinism throughout the whole task lifespan. On the other hand, leveraging DMA channels can transfer data to and from devices with much less CPU overhead than network-based solutions[55].

4.1.2 *Sub-task Processing Determinism Guarantee.*

We propose Isolated A-Core and Bare-metal R-Core to keep sub-tasks processing away from uncertain OS scheduling.

Isolated A-Core It's difficult to ensure the execution latency of a time-sensitive task running on an TSOS, i.e., Debian[54]. The rationale is that there are a number of background processes running on the OS at the same time, competing with the time-sensitive task for computing resources. To avoid the influence of these processes, we reserve three A-Cores (red dotted in Fig.5) for the execution of each time-sensitive task. In our implementation, we realize A-Core isolation by build Linux OS with boot parameter `isolcpus=<cpu number>`.

Bare-metal R-Core R-Core is dedicated for real-time applications, such as high frequency industrial machinery control. It provides a high level of reliability and determinism. It's common to running an RTOS on the R-Cores for better resource management and complex task scheduling. However, considering RTOS's elaborate scheduling algorithm could introduce extra delay (see §5.4), we finally adopts the R-Cores with no OS, i.e., bare-metal R-Core.

ALGORITHM 1: PL Load Balancing Algorithm

```
input :Resized Image AXI-Stream signal *axis-image*
1 S-A1 ← Get-#A1-Status;
2 S-A2 ← Get-#A2-Status;
3 if S-A2 is busy than S-A1 then
4     Send-DMA-to-#A1(axis-image);
5 else
6     Send-DMA-to-#A2(axis-image);
```

ALGORITHM 2: PS Receiving New Image Algorithm

```
1 while Receiving is uncompleted do
2     Receive-DMA-from-PL(new_image);
3 if Running on #A1 then
4     Push-into-#A1-Task-Queue(new_image);
5     Update-#A1-Status(+1);
6 else
7     Push–into-#A2-Task-Queue(new_image);
8     Update-#A2-Status(+1);
```

4.2 Task Computing Acceleration

The essential intelligent tasks (e.g., defect detection, object tracking) in *planning* rely on neural networks (e.g., VGG, ResNet) to function. Although MPSoC provides versatile computational resources, it's still non-trivial to make the real-time performance of neural network inference on Netopia meet industrial application requirements. The rationale is that the processors (i.e., the quad-core Cortex-A53 and dual-core Cortex-R5) on MPSoC's PS are designed for general applications without being optimized for those massive floating-point operations in neural networks as Nvidia Jetson lightweight AI devices do[19]. According to our measurements, running a typical ResNet-50 model on the Cortex-A53 first brings an average 12*ms* delay for resizing each input image (from 1920×1080 to 224×224), and then costs over 110*ms* to infer the model. Each of these two parts is unacceptable because t_p should be below 16.67*ms* (§5.4).

To address the above issue and accelerate task computing in Netopia, we propose a fresh neural network inference workflow on MPSoC through the collaboration of PL and PS. As shown in Fig.6, the workflow consists of a *Direct Image Packing* and a *Dual-Agent Inference* two modules. At a high-level, *Direct Image Packing* runs on PL, re-packing image clips from network packets and converting each image to a specific size and format that fits the neural network input. The *Dual-Agent Inference* schedules the computational tasks between PL and PS, leveraging logic blocks to cooperate and accelerate the neural network inference.

4.2.1 *Direct Image Packing.*

Netopia implements an in-network packet parser for restoring image data from the network link layer, and on this basis, directly executes data pre-processing (e.g., image resizing) on PL. In general, the leverage of elaborate logic blocks could accelerate the image resizing, and the integrated image packet parsing and pre-processing workflow further reduces the delay of transferring raw images from PL to PS. Specifically, the *Direct Image Packing* module accelerates the image pre-processing by around 500% compared to simply perform it on PS (§5.4).

ALGORITHM 3: PS Model Inference Algorithm

```
1  An ← Get-Core-Number();
2  while True do
3      if An-Task-Queue is empty then
4          continue;
5      new_image ← Pop-From-An-Task-Queue();
6      while Inference not finished do
7          Offload-Inference-to-PL();
8          Perform-Unsupported-Operations-On-An();
9      Send-Inference-Results-to-Follow-up-Modules();
10     Update-An-Status(-1);
```

Fig.6 (left) presents the workflow of the Direct Image Packing Module. As seen, sensing data packets will be first parsed and packed into raw images through a in-network parser state machine (the left bottom part). Specifically, the state machine fully explores the piplined propriety of task processing on PL; it continuously removes the header of sensing data packets from state S0 to state S1 (i.e., by set the packet header signals as dummy data, i.e., *assign m_axis_tkeep=1..110..00* in implementation), and further reconstructs raw images from S1 to S2. An reconstructed image will be directly fed into a hardware image resizing Intellectual Property (IP[22]). Eventually, PL sends it to PS (i.e., A-Cores) by *Sensing DMA* for neural network inference.

4.2.2 ***Dual-Agent Inference.*** Dual-Agent Inference utilizes PL and two A-cores to accelerate neural network model inference. Generally, we find there are two major complementary features between PL and PS, which guides our design for task decomposition and re-allocation to accelerate neural network inference.
(i) On the task execution front, PL is superior at parallelized tasks processing than PS as lots of logic designs in PL have the pipeline architecture[63], which motivates us to adopt pipeline architecture in the inference accelerator and exploit two A-Cores to improve the inference throughput;
(ii) On the computational ability front, PL has a greater appreciation for operations with large number of parameters than PS as a number of arithmetic units, such as DSP[21], allows ultra-fast parameters loading and efficient numerical calculation, which guides us to offload computation-intensive tasks to PL. Nevertheless, logic blocks are unable to handle exponential calculations, so we have to move the Softmax or SigMod layer back to A-Cores.

We translate the above analysis into a practical neural network acceleration workflow: two isolated A-Cores for low-computing layers inference and pipelined Inference Accelerator for computation-intensive layers inference. As an example, in the lower right of Fig.6, we split ResNet-50 into two portions: (i) the first convolutional layer and the Softmax layer before output are executed on an isolated A-Core. (ii) the main backbone is offloaded to PL for acceleration.

In order to improve the throughput, we balance the workload of two cores by introducing Dual-Agent Inference mechanism including three algorithms. Algorithm1 explains the logic how PL dispatches pre-processed data (i.e., resized images) to these two A-cores. PL first gets two A-cores' current state (Line 1, 2) from certain registers, then select one A-core with lower workloads (Line 3) and sends the image through the DMA (Line 4, 6). On each agent, there are two threads that running Algorithm2 and Algorithm3, respectively. Algorithm2 performs image receiving from PL (Line 1, 2). After receiving, the new image will be pushed into the task queue (Line 4, 7) waiting for inference, and update the core status[3], thus change the values of registers in PL, via PS-PL AXI-lite interface (Line 5, 8). Algorithm3 takes charge of model inference. It continuously takes images from task queues (Line 5) and performs the model inference by offloading it to the PL accelerator mentioned above (Line 7). Additionally, some special operations, which are not supported by PL, will be executed with the assistance of A-Cores (Line 8). After completing inference, it will send the inference results to the follow-up modules (Line 9), and update the current A-core's status because its workload decreases (Line 10).

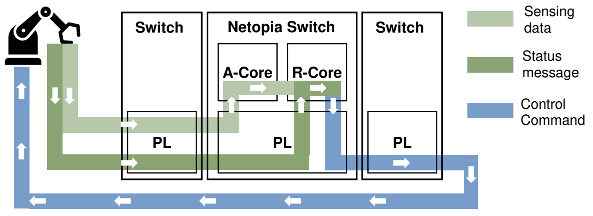


Figure 7: Different network traffic flow in Netopia.

4.3 Packet Deterministic Forwarding

Netopia realize deterministic forwarding of the following three kinds of critical data packets: (i) the status message packet; (ii) the calculated control command packet; and (iii) the sensing data packet. Among them, (i) and (ii) enjoy the highest priority compared to (iii) as they are directly related to an arm's operations. (iii) is also given a higher priority than background traffic (e.g., surveillance videos) since all intelligent decisions are made based on it.

In Netopia, the key insight behind achieving traffic deterministic forwarding is *reserving dedicated network bandwidth and time slots* for those critical packets. To this end, we make the communication layer of Netopia compatible with TSN, following the IEEE 802.1Qbv standard[1] to leverage a *Time-Aware Shaper* for resource reservation. To push forward the determinism, we additionally introduce a *Time Synchronization* mechanism referring to the IEEE 802.1AS standard as reserving dedicated time slots requires connected devices enjoy the same global timestamp (i.e., bias $<1ns$).

4.3.1 Time Synchronization. The naive software implementation of time synchronization, e.g., Network Time Protocol[60] (NTP), could achieve milliseconds accuracy, which does not meet the industrial requirements. In Netopia, we implement the time synchronization protocol defined in IEEE 802.1AS[2] through software and hardware co-design. The synchronization algorithm is developed in PS using C on #A4, while the real-time clock and timestamping module are implemented in PL using verilog for precise timing. The timestamp of each synchronization packet was added as soon as its pass through the PHY chip and TEMAC IP[24] in PL, so that the

[3]We define the core status as the size of current task queues

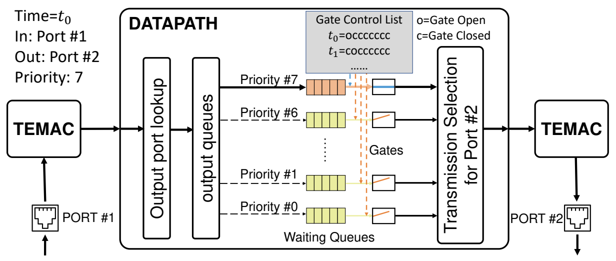


Figure 8: Netopia switch forwards a packet with priority 7 from port #1 to port #2

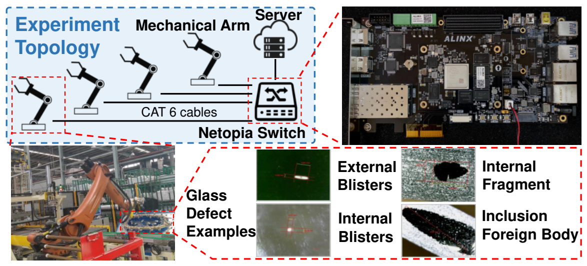


Figure 9: Experimental industrial network topology with four typical glass defect examples.

link delay, key to the synchronization algorithm, is more precise as PL logic hardly impairs the link delay.

4.3.2 *Time-Aware Shaper.* IEEE 801.1Qbv designs a Time-Aware Shaper (TAS), with predefined Gate Control List (GCL), to transmit periodic traffic streams. The concept of TAS is based on time-division multiple access (TDMA). It divides the network communication into fixed length, repeating time cycles and uses a timed flow table to control the traffic transmission, thus reserve the bandwidth for critical traffic. The timed flow table specifies when to open the gate within a cycle so that the critical traffic (see Fig.7) with the corresponding priority can being transmitted. The background traffic can only be transmitted when no critical traffic is under transmission.

Fig.8 shows how Netopia switch forwards a packet with priority 7 (top priority) from port #1 to port #2. The packet first passes through TEMAC IP for signal conversion, and then goes into DATAPATH module. In DATAPATH, this packet is pushed into a waiting queue according to it's priority (7) and destination port number (2). At time t_0, GCL opens the Gate 7 and this packet will be forwarded after passing through output TEMAC.

5 EVALUATION

5.1 Experimental Methodology

Field studies. Based on Netopia, we have developed a real-time defective glass detection and sorting system, and deployed it on a production line in the glass factory. As shown in Fig.9, the system consists of four mechanical arms connected to a Netopia switch through category 6 Ethernet cable.We select 522 panes of defective glass (refer to the bottom-right of Fig.9) and 246 panes of qualified glass to pass through the conveyor belt, and designate one mechanical arm (arm #4 in particular) to pick all defective ones. The whole field studies lasts over two hours with around 1500 defect detection and grabbing tests.

Dateset. In addition to our field studies, we also conduct comprehensive evaluations based on public robotic control datasets (grasping dataset[31] and push dataset[11]). In this part of evaluations, we leverage four ZYNQ-7021 SoC boards, simulating mechanical arms, to periodically (1000Hz) send sampling data to Netopia switch also in industrial environments.

Metrics. We first use the *grabbing success rate*, the rate arm #4 could successfully identify a defective glass and grab it onto a recycle line, to evaluate the system performance from a holistic perspective. We further measure the *control* latency t_c and *planning* latency t_p (defined in §2.1) to better understand the rationale behind the overall performance.

Baselines. We compare Netopia with two most relevant state-of-the-art industrial control systems, Baseline-I[29] and Baseline-II[42], to evaluate the system performance. Following their design, we additionally introduce an edge server to deploy them in industrial environments. Specifically, mechanical arms connect to the industrial switch through PROFINET, while the edge server connects to the industrial switch through 100Mbps standard Ethernet and is equipped with an Intel Core i9-10980XE CPU@3.00GHz and an NVIDIA GeForce RTX 2080 Ti GPU.

Network loads. To better evaluate the system robustness under practical industrial network conditions, we further plug multiple surveillance cameras into our testbed to send surveillance videos through the deployed network, simulating background traffic. By adjusting the quantity and quality of active video streams, we divide the network load into four levels of 0%, 25%, 50%, 75% Note that in real manufacturing scenarios, the network load typically exceeds 50%.

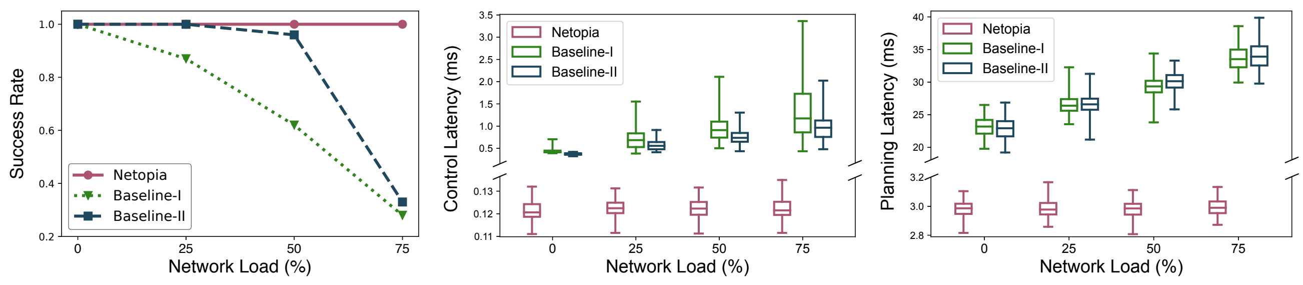
Workloads. In our experiments, the *planning* module first uses a neural network with ResNet-50 backbone to detect defects on each glass and track the defective ones. Afterwards, it leverages a trajectory planning algorithm to calculate the arm's subsequent working trajectory for grabbing each defective glass onto a recycle belt. Meanwhile, the *control* module exploits IK algorithm[58] to translate the planned trajectory into arm's configuration space (i.e., each motor's target state), and PID control[62] to gradually adjust each motor's motion.

5.2 Overall Performance

Fig.10 illustrates the overall performance of Netopia and two baselines. In a nutshell, Netopia achieves an order of magnitude lower control and planning latency, and succeeds in grabbing defective glass panes in all test cases.

Fig.10a displays the success rate results[4]. Netopia maintains 100% success rate under all network loads, outperforming two baseline systems. When the network is idle (network load = 0%), both baselines achieve 100% success rate. However, their success rates decrease significantly as the background traffic increases. When the network load is 75%, Baseline-I and Baseline-II exhibits 33% and 28% success rate, respectively. To make matters worse, the mechanical arm is shut down by the safety mechanism occasionally with the baselines.

[4]We excluded all unsuccessful cases caused by ResNet-50's misidentification



(a) Success Rate (b) Control Latency (c) Planning Latency

Figure 10: Overall performance

Furthermore, We measures the latency introduced by the *control* module and *planning* module respectively in Fig.10b and Fig.10c. As depicted in the bottom part of two figures, Netopia perfectly meets the requirements of industrial arms: (i). t_c is stable at around 120μs under all four network conditions, and $t_c \ll 1ms$ even in the worst cases. (ii). t_p is no more than 3.2ms in all test cases, far below the requirement of 16.67ms. Benefiting from (i) and (ii), Netopia never misses any grabbing in Fig.10a.

Nevertheless, the results of Baseline-I and Baseline-II are unsatisfactory. As depicted in Fig.10b, though t_c of two baselines is relatively small ($< 1ms$) with no background traffic, it increases significantly as the network load increases. At the 75% network load level, over half of the baseline cases show $t_c \geq 1ms$, which causes the unexpected shutdowns in Fig.10a. Besides, the jitter of t_c in Baseline-I is markedly lower than that in Baseline-II, mainly resulting from the additional link in network transmission.

As to t_p, Fig.10c demonstrates that offloading *planning* module to cloud/edge servers for real-time control is infeasible: on the one hand, heavy network load delays the arrival of images from the arm (around 29ms when NL = 75%); on the other hand, the latency jitter introduced by GPU computation (up to 10ms) further impairs its practicality.

5.3 Robustness Study

We conduct three experiments to demonstrate Netopia's portability and scalability. The latency of *control* and *planning* modules is recorded and shown in Fig.11.

Impact of different network models. We evaluate the performance of Netopia with different neural network models as the defect detector. Fig.11a shows the CDFs of t_p with ResNet-101, MobileNetv1 and YOLOv3. Compared to Fig.10c, t_p is obviously higher in this experiment, because all these three models have more FLOPs (Floating Point OPerations) than ResNet-50. For ResNet-101 and MobileNetv1, t_p distributes in the range of $[3.1ms, 4.2ms]$ and $[8.0ms, 10.2ms]$, respectively, satisfying the requirements of 16.67ms. Their t_p jitter is also small enough for deterministic requirements. However, the average t_p with YOLOv3 is around 19.5ms, slightly above 16.67ms, because YOLOv3 is a relatively heavier detector. As a result, YOLOv3 is more suitable for non-urgent but accuracy-sensitive tasks with powerful cloud/edge servers.

Impact of arm's DoF. We investigate the performance of Netopia on the mechanical arm with different degrees of freedom (DoF). As the DoF of arm is positively related to the computational complexity in *control* module, we compare the *control* latency t_c[5] with three most common DoFs of the commercial arm, i.e., 5, 6 and 7. As depicted in Fig.11c, t_c increases slightly as the DoF increases, with mean value of 100.8μs, 108.7μs and 122.1μs, respectively. Meanwhile, the jitter of t_c does not exceed 40μs across different DoFs. The result demonstrates that Netopia can maintain its reliability even if more axes are added to future mechanical arms.

Impact of number of connected arms. We delineate the performance of Netopia when multiple mechanical arms are connected. As depicted in Fig.11c, when the number of devices increases from 1 to 4, the control latency remains stable at around 125μs. Benefiting from the pipeline design in hardware acceleration and the load balancing of dual A-core inference, the total planning latency only increases slightly when more arms are present (from 3.01ms for one arm to 3.25ms for four arms). In summary, the result demonstrates that Netopia is scalable and enables the connection of more end devices.

5.4 Ablation Study

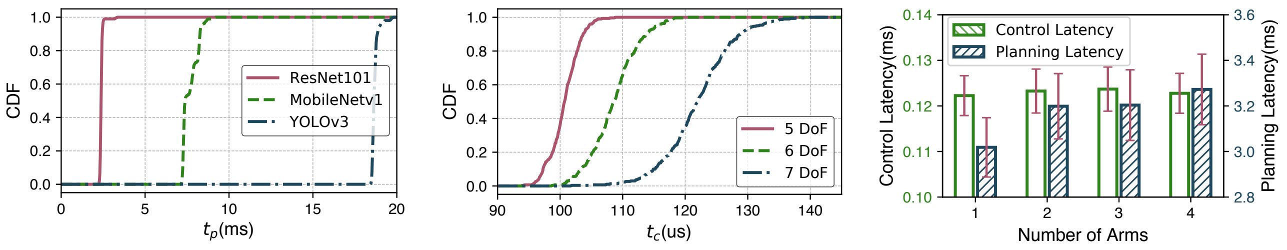
We conduct several experiments to understand the effectiveness of each module in Netopia.

Hardware acceleration. We remove Direct Image Packing (DIP) and Dual-Agent Inference (DAI), and demonstrate the respective planning latency in Fig.12a and Fig.12b. Fig.12a demonstrates that DIP does reduces the planning latency t_p by about 0.4ms and lower the jitter by about 0.18ms, ensuring Netopia's fast and deterministic planning. Furthermore, as depicted in Fig.12b, t_p without DAI is more than one order of magnitude higher than that with, indicating that inference acceleration is critical to Netopia.

Tri DMA. We replace the sensing DMA with a PS-PL Ethernet interface to investigate its influence on Netopia. Fig.12c shows the planning latency w/ and w/o the sensing DMA. Although the median of t_p only increases 0.54ms, in the worst case, t_p increases by over 130%. This indicates that DMA can reduces the t_p jitter by eliminating the bandwidth contention between critical and background traffic.

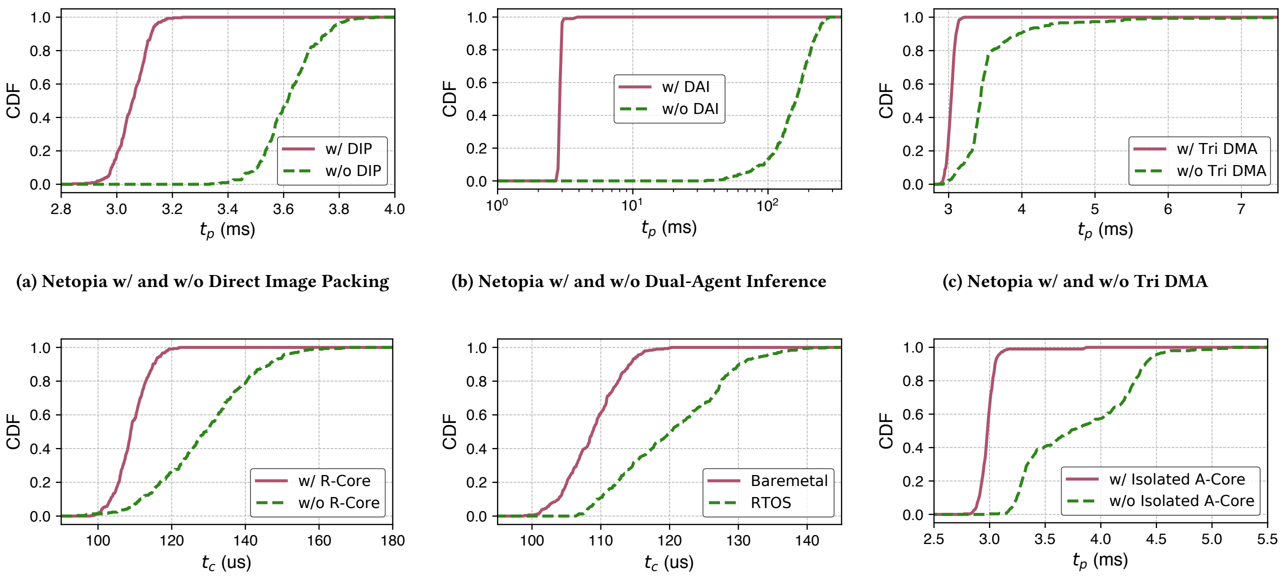
Baremetal RPU. In Netopia, the computation of the *control* module is deployed on the baremetal RPU. We conducted two experiments

[5]As TSN requires pre-scheduling[16], the computation latency in *control* module should be known in advance. For convenience, TSN's transmission latency is excluded from t_c in this experiment.



(a) Impact of different neural network model (b) Impact of arm's DoF (c) Impact of number of connected arms

Figure 11: Robustness Evaluation



(a) Netopia w/ and w/o Direct Image Packing (b) Netopia w/ and w/o Dual-Agent Inference (c) Netopia w/ and w/o Tri DMA

(d) Netopia w/ and w/o R-Core (e) Netopia with Bare-metal R-Core and RTOS (f) Netopia w/ and w/o Isolated A-Core

Figure 12: Ablation Study

to understand if it could utilize the ordinary APU and if the alternative ways to schedule RPU (e.g., RTOS) were practical. Fig.12d and Fig.12e display the corresponding results. If the *control* module is placed on the APU, the APU will simultaneously do the control and planning computation, resulting in an addition in the delay and jitter of control(Fig.12d). Similarly, when using RTOS on the RPU scheduling, the operating system adds unnecessary overhead and increase the control's delay and jitter (Fig.12e).

Isolated A-Core. We evaluate the performance of Netopia's *planning* module with and without core isolation on the APUs. The results are shown in Fig.12f. When core isolation is turned off, the planning computation may be interrupted by kernel-level processes or other high priority user-level processes. According to the experiment, core isolation can successfully reduce t_p by 0.32ms and the jitter by 4.77ms. As a result, it is crucial for Netopia to implement the core isolation technique.

6 RELATED WORK

We review the most related works in this section.

Industrial Control for Mechanical Arms. Generally speaking, repeatability and accuracy are two golden metrics for measuring the control effectiveness of an arm during the execution of its producing tasks. The former is defined as its ability to achieve repetition of the same task while the latter is the difference (i.e., the bias) between the requested task and the task actually achieved[43, 50]. The ultimate goal of an industrial control system is to make each arm have both - accurately hitting every target every time. Traditional industrial control systems rely on expensive hardware programmable logic controllers (PLC), and major PLC providers include Siemens, Rockwell Automation, Mitsubishi Electric, etc.. However, hardware PLC-based solutions cannot meet the high flexibility and intelligence requirements of production lines in the era of Industry 4.0 (e.g., the defective glass detection and grabbing case presented in §2.1), as the re-programming of PLCs typically takes a long time, e.g., days even weeks, and requires a lot of expertise[15].

Edge Computing in Industry. Recent years have witnessed the rapid developments of edge computing[4, 34, 38, 41, 68, 69, 74], especially for industrial applications such as industrial inspection[35, 51, 72], robotic perception[6]. Several studies have also shown the benefits of connecting mechanical arms to an edge server for flexible and intelligent industrial control, replacing traditional high-end PLC-based solutions and thus making the control tasks evolve from simple relay logic to complex machine learning models[39, 40]. Though edge-based processing has clear benefits in making the management processes simple and flexible, they cannot easily satisfy the low-latency and high-reliability network requirements of real-time industrial control as demonstrated in §2.2.

In-Network Industrial Computing and Control. With the advent of next-generation network devices (e.g., PISA switches[7], P4 language[12], and a wider scope of handcrafted white box switches based on diverse computing platforms[14]), a new era has begun in which programmable switches can not only perform pure packet forwarding but simple computations as well. This trend leads to the birth of in-network computing, a new computational paradigm, where edge/cloud-based computations (or a part of them) are loaded to switches or other programmable data planes[18, 26, 27]. This new way of using networking hardware opens up the opportunities for low-latency and reliable calculations during the communication. Recent studies adopt the paradigm for controlling mechanical arms on network switches[28, 70, 73]. The most relevant in-network control systems to our work is Baseline-II (published on NSDI'22[29]), which performs the real-time velocity control of robot arms on a P4-enabled programmable network switch and offloads the high-level planning module at an edge-based industrial controller. However, based on our extensive evaluations, we find leaving planning tasks on edge still faces fundamental limitations, spanning from data transmission, computation, and packet deterministic forwarding, for real-time industrial applications (detailed in §2.2), which motivates the design of Netopia.

7 DISCUSSION

Netopia is the first attempt to simultaneously offload the control module and planning module on an industrial switch. We briefly discuss some concerns, limitations and future works in this section.

• **Cost of Netopia.** Netopia is implemented on the development board MPSoC, priced at around $1,000 (with $800 dedicated to the core FPGA board), offers a more economical alternative to traditional industrial control platforms. As detailed in §2.2, deploying such platforms—including industrial Ethernet and edge computers—typically exceeds $10,000. Additionally, converting the design prototype into dedicated chips could significantly reduce Netopia's expenses. Consequently, this budget-friendly, high-performing solution positions Netopia for success in the competitive market.

• **Generalizability.** We concentrate on processing general and computationally-intensive visual tasks within the planning module. Owing to the *Direct Image Packing* and *Dual-Agent Inference* modules in §4.2, Netopia boasts wide applicability across various network architectures, with most inference acceleration strategies (e.g., pruning, quantization and early exit) ready for deployment with minimal adaptation. However, industrial environments also involve other types of sensing data, such as mmWave RADAR and IMU data. As a result, fusing multi-modal data on Netopia switches for precise environment perception and motion control is nontrivial and would be left as the future work.

• **Multi-arms Collaboration.** Netopia represents a pioneering effort to offload simple yet urgent tasks from cloud/edge servers to industrial network switches within the planning module. Still, some urgent tasks with high computational demands, like multi-arms collaboration, remain to be addressed. Future research will explore methods for offloading those complicated tasks onto Netopia switches. Potential avenues include investigating specific computation compression algorithms and devising strategies to divide complicated tasks into smaller components, distributing them across multiple switches. Ultimately, the key challenge lies in ensuring that mechanical arms receive intelligent control commands with low and deterministic latency.

8 CONCLUSION

We have presented the design and implementation of Netopia, a brand-new industrial switch that simultaneously supports in-network planning and control. Netopia proposes functionality and resource abstractions of a next-generation computing platform and features three vital services to ensure delay determinism in terms of task computation, data interaction, and packet forwarding. On this basis, Netopia enables mechanical arms to periodically obtain intelligent control commands with low and deterministic latency, empowering them to make agile and intelligent decisions. Extensive evaluations on public datasets and in real industrial environments demonstrate its superior performance.

9 ACKNOWLEDGMENTS

We sincerely thank the MobiSense Group, the anonymous reviewers and our shepherd for their constructive comments and feedback in improving this work. This work is supported in part by the National Key Research Plan under grant No. 2021YFB2900100, the NSFC under grant No. 61832010 and No. 62202263.

REFERENCES

[1] 2015. Enhancements for Scheduled Traffic.

[2] 2020. Timing and Synchronization for Time-Sensitive Applications.

[3] Mamta Agiwal, Abhishek Roy, and Navrati Saxena. 2016. Next generation 5G wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* (2016).

[4] Ali J Ben Ali, Marziye Kouroshli, Sofiya Semenova, Zakieh Sadat Hashemifar, Steven Y Ko, and Karthik Dantu. 2022. Edge-SLAM: edge-assisted visual simultaneous localization and mapping. *ACM Transactions on Embedded Computing Systems* (2022).

[5] Sara Alonso, Jesus Lazaro, Jaime Jimenez, Leire Muguira, and Unai Bidarte. 2021. Evaluating the OpenAMP framework in real-time embedded SoC platforms. In *2021 XXXVI Conference on Design of Circuits and Integrated Systems (DCIS)*.

[6] Andrea Bonci, Pangcheng David Cen Cheng, Marina Indri, Giacomo Nabissi, and Fiorella Sibona. 2021. Human-robot perception in industrial environments: A survey. *Sensors* (2021).

[7] Ismail Butun, Yusuf Kursat Tuncel, and Kasim Oztoprak. 2021. Application Layer Packet Processing Using PISA Switches. *Sensors* (2021).

[8] Tamás Czimmermann, Gastone Ciuti, Mario Milazzo, Marcello Chiurazzi, Stefano Roccella, Calogero Maria Oddo, and Paolo Dario. 2020. Visual-based defect detection and classification approaches for industrial applications—a survey. *Sensors* (2020).

[9] Hongwen Dong, Kechen Song, Yu He, Jing Xu, Yunhui Yan, and Qinggang Meng. 2020. PGA-Net: Pyramid Feature Fusion and Global Context Attention Network for Automated Surface Defect Detection. *IEEE Transactions on Industrial Informatics* (2020).

[10] Guoguang Du, Kai Wang, Shiguo Lian, and Kaiyong Zhao. 2021. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review* (2021).

[11] Chelsea Finn, Ian Goodfellow, and Sergey Levine. 2016. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems* (2016).

[12] Open Networking Foundation. 2022. *Programming Protocol-independent Packet Processors*. https://opennetworking.org/p4/

[13] FRANKA. 2022. *Minimum system and network requirements*. https://frankaemika.github.io/docs/requirements.html

[14] Nomios Group. 2022. *What is white box switching?* https://www.nomios.com/resources/white-box-switching/

[15] Vivek Hajarnavis and Ken Young. 2008. An assessment of PLC software structure suitability for the support of flexible manufacturing processes. *IEEE transactions on automation science and engineering* (2008).

[16] Xiaowu He, Xiangwen Zhuge, Fan Dang, Wang Xu, and Zheng Yang. 2023. Deep-Scheduler: Enabling Flow-Aware Scheduling in Time-Sensitive Networking. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE.

[17] Yu He, Kechen Song, Qinggang Meng, and Yunhui Yan. 2019. An end-to-end steel surface defect detection approach via fusing multiple hierarchical features. *IEEE Transactions on Instrumentation and Measurement* (2019).

[18] Ning Hu, Zhihong Tian, Xiaojiang Du, and Mohsen Guizani. 2021. An energy-efficient in-network computing paradigm for 6G. *IEEE Transactions on Green Communications and Networking* (2021).

[19] NVIDIA Inc. 2022. *Jetson TX2*. https://www.nvidia.com/en-sg/autonomous-machines/embedded-systems/jetson-tx2/

[20] Xilinx Inc. 2020. *Petalinux - Xilinx wiki*. https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842250/PetaLinux

[21] Xilinx Inc. 2022. *Flexible DSP Solutions*. https://www.xilinx.com/products/technology/dsp.html

[22] Xilinx Inc. 2022. *Intellectual Property*. https://www.xilinx.com/products/intellectual-property.html

[23] Xilinx Inc. 2022. *MPSoC PS and PL Ethernet Example Projects*. https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/478937213/MPSoC+PS+and+PL+Ethernet+Example+Projects

[24] Xilinx Inc. 2022. *Tri-Mode Ethernet Media Access Controller (TEMAC)*. https://www.xilinx.com/products/intellectual-property/temac.html

[25] Xilinx Inc. 2022. *Zynq® UltraScale+™ MPSoC*. https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html

[26] Somayeh Kianpisheh and Tarik Taleb. 2022. A Survey on In-network Computing: Programmable Data Plane And Technology Specific Applications. *IEEE Communications Surveys & Tutorials* (2022).

[27] Changhoon Kim. 2016. Programming the network dataplane. *ACM SIGCOMM: Florianopolis, Brazil* (2016).

[28] Ike Kunze, René Glebke, Jan Scheiper, Matthias Bodenbenner, Robert H Schmitt, and Klaus Wehrle. 2021. Investigating the applicability of in-network computing to industrial scenarios. In *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*.

[29] Sándor Laki, Csaba Györgyi, József Pető, Péter Vörös, and Géza Szabó. 2022. In-Network Velocity Control of Industrial Robot Arms. In *Proceedings of the USENIX NSDI*.

[30] Sander Lass and Norbert Gronau. 2020. A factory operating system for extending existing factories to Industry 4.0. *Computers in industry* (2020).

[31] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research* (2018).

[32] Haoran Li, Meng Xu, Chong Li, Chenyang Lu, Christopher Gill, Linh Phan, Insup Lee, and Oleg Sokolsky. 2021. Towards Virtualization-Agnostic Latency for Time-Sensitive Applications. In *29th International Conference on Real-Time Networks and Systems*.

[33] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge assisted real-time object detection for mobile augmented reality. In *The 25th annual international conference on mobile computing and networking*.

[34] Sicong Liu, Yingyan Lin, Zimu Zhou, Kaiming Nan, Hui Liu, and Junzhao Du. 2018. On-demand deep model compression for mobile devices: A usage-driven model selection framework. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*.

[35] Zheng Liu, Hiroyuki Ukida, Pradeep Ramuhalli, and Kurt Niel. 2015. Integrated Imaging and Vision Techniques for Industrial Inspection. *Advances in Computer Vision and Pattern Recognition* (2015).

[36] Ryosuke Okuda, Yuki Kajiwara, and Kazuaki Terashima. 2014. A survey of technical trend of ADAS and autonomous driving. In *Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test*.

[37] OpoenAMP. 2022. *OpenAMP Project*. https://www.openampproject.org/

[38] Arthi Padmanabhan, Neil Agarwal, Anand Iyer, Ganesh Ananthanarayanan, Yuanchao Shu, Nikolaos Karianakis, Guoqing Harry Xu, and Ravi Netravali. 2023. GEMEL: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

[39] Jorge Ribeiro, Rui Lima, Tiago Eckhardt, and Sara Paiva. 2021. Robotic process automation and artificial intelligence in industry 4.0–a literature review. *Procedia Computer Science* (2021).

[40] Gerasimos G Rigatos. 2011. Modelling and control for intelligent industrial systems. *adaptive algorithms in robotics and industrial engineering* (2011).

[41] Manasvini Sethuraman, Anirudh Sarma, Ashutosh Dhekne, and Umakishore Ramachandran. 2021. Foresight: planning for spatial and temporal variations in bandwidth for streaming services on mobile devices. In *Proceedings of the 12th ACM Multimedia Systems Conference*.

[42] Alaa Sheta, Nazeeh Ghatasheh, Hossam Faris, and Ali Rodan. 2018. Robotics Evolution: from Remote Brain to Cloud. *International Journal of Control and Automation* (2018).

[43] Arif Şirinterlikçi, Murat Tiryakioğlu, Adam Bird, Amie Harris, and Kevin Kweder. 2009. Repeatability and accuracy of an industrial robot: Laboratory experience for a design of experiments course. *The Technology Interface Journal* (2009).

[44] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. 2018. Industrial internet of things: Challenges, opportunities, and directions. *IEEE transactions on industrial informatics* (2018).

[45] PLOS ONE Staff. 2016. Retraction: biomechanical characteristics of hand coordination in grasping activities of daily living.

[46] Guangzhi Tang and Konstantinos P Michmizos. 2018. Gridbot: An autonomous robot controlled by a spiking neural network mimicking the brain's navigational system. In *Proceedings of the International Conference on Neuromorphic Systems*.

[47] Yunchao Tang, Mingyou Chen, Chenglin Wang, Lufeng Luo, Jinhui Li, Guoping Lian, and Xiangjun Zou. 2020. Recognition and localization methods for vision-based fruit picking robots: A review. *Frontiers in Plant Science* (2020).

[48] Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath. 2022. Safety-Critical Control and Planning for Obstacle Avoidance between Polytopes with Control Barrier Functions. In *IEEE International Conference on Robotics and Automation*.

[49] Axel Vick, Vojtěch Vonásek, Robert Pěnička, and Jörg Krüger. 2015. Robot control as a service—towards cloud-based motion planning and control for industrial robots. In *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*. IEEE, 33–39.

[50] Michal Vocetka, Róbert Huňady, Martin Hagara, Zdenko Bobovskỳ, Tomáš Kot, and Václav Krys. 2020. Influence of the Approach Direction on the Repeatability of an Industrial Robot. *Applied Sciences* (2020).

[51] Daniel Weimer, Bernd Scholz-Reiter, and Moshe Shpitalni. 2016. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP annals* (2016).

[52] Patrick Wenzel, Torsten Schön, Laura Leal-Taixé, and Daniel Cremers. 2021. Vision-based mobile robotics obstacle avoidance with deep reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*.

[53] Wikipedia. 2022. CentOS. https://en.wikipedia.org/w/index.php?title=CentOS&oldid=1118193821

[54] Wikipedia. 2022. Debian. https://en.wikipedia.org/w/index.php?title=Debian&oldid=1126272347

[55] Wikipedia. 2022. Direct memory access. https://en.wikipedia.org/w/index.php?title=Direct_memory_access&oldid=1123461808

[56] Wikipedia. 2022. Docker (software). https://en.wikipedia.org/w/index.php?title=Docker_(software)&oldid=1123196945

[57] Wikipedia. 2022. EtherCAT. https://en.wikipedia.org/w/index.php?title=EtherCAT&oldid=1120354882.

[58] Wikipedia. 2022. Inverse kinematics. https://en.wikipedia.org/w/index.php?title=Inverse_kinematics&oldid=1124543632.

[59] Wikipedia. 2022. Modbus. https://en.wikipedia.org/w/index.php?title=Modbus&oldid=1125273307

[60] Wikipedia. 2022. Network Time Protocol. https://en.wikipedia.org/w/index.php?title=Network_Time_Protocol&oldid=1125151433

[61] Wikipedia. 2022. Patellar reflex. https://en.wikipedia.org/w/index.php?title=Patellar_reflex&oldid=1069824082

[62] Wikipedia. 2022. PID controller. https://en.wikipedia.org/w/index.php?title=PID_controller&oldid=1121264197.

[63] Wikipedia. 2022. Pipelining (DSP implementation). https://en.wikipedia.org/w/index.php?title=Pipelining_(DSP_implementation)&oldid=1121275138

[64] Wikipedia. 2022. Profinet — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Profinet&oldid=1118142114

[65] Wikipedia. 2022. Tegra. https://en.wikipedia.org/w/index.php?title=Tegra&oldid=1123369665

[66] Wikipedia. 2022. Virtualization. https://en.wikipedia.org/w/index.php?title=Virtualization&oldid=1116535435

[67] Wikipedia. 2022. Xenomai. https://en.wikipedia.org/w/index.php?title=Xenomai&oldid=1109190278

[68] Jingao Xu, Hao Cao, Zheng Yang, Longfei Shangguan, Jialin Zhang, Xiaowu He, and Yunhao Liu. 2022. SwarmMap: Scaling Up Real-time Collaborative Visual SLAM at the Edge. In *Proceedings of the USENIX NSDI*. 977–993.

[69] Mengwei Xu, Zhe Fu, Xiao Ma, Li Zhang, Yanan Li, Feng Qian, Shangguang Wang, Ke Li, Jingyu Yang, and Xuanzhe Liu. 2021. From cloud to edge: a first look at public edge platforms. In *Proceedings of the 21st ACM Internet Measurement Conference.*

[70] Zheng Yang, Yi Zhao, Fan Dang, Xiaowu He, Jiahang Wu, Hao Cao, Zeyu Wang, and Yunhao Liu. 2023. CaaS: Enabling Control-as-a-Service for Time-Sensitive Networking. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE.

[71] Han Zhang, Abhijith Anilkumar, Matt Fredrikson, and Yuvraj Agarwal. 2021. Capture: Centralized Library Management for Heterogeneous {IoT} Devices. In *30th USENIX Security Symposium (USENIX Security 21)*.

[72] Jialin Zhang, Xiang Huang, Jingao Xu, Yue Wu, Qiang Ma, Xin Miao, Li Zhang, Pengpeng Chen, and Zheng Yang. 2022. Edge Assisted Real-time Instance Segmentation on Mobile Devices. In *Proceedings of the IEEE ICDCS*.

[73] Yi Zhao, Zheng Yang, Xiaowu He, Jiahang Wu, Hao Cao, Liang Dong, Fan Dang, and Yunhao Liu. 2022. E-TSN: Enabling Event-triggered Critical Traffic in Time-Sensitive Networking for Industrial Applications. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. 691–701. https://doi.org/10.1109/ICDCS54860.2022.00072

[74] Ruogu Zhou and Guoliang Xing. 2014. nshield: A noninvasive nfc security system for mobiledevices. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*.

[75] Fengyuan Zhu, Mingwei Ouyang, Luwei Feng, Yaoyu Liu, Xiaohua Tian, Meng Jin, Dongyao Chen, and Xinbing Wang. 2022. Enabling software-defined PHY for backscatter networks. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*.