学术观点

支持控制即服务的实时工业网络体系架构

杨 铮 赵 毅 贺骁武 等
清华大学

关键词：工业互联网 柔性制造 时间敏感网络

背景

计算机网络技术目前正处在从消费互联网向工业互联网转变的关键发展时期。在这个转变过程中，一些工业场景需要使用开放、标准的计算机网络协议，代替现有封闭、私有的工业网络控制协议。这不仅是一个美好的愿望，也是来自工业生产发展的内在需求。

在物资相对短缺的时代，制造业主要解决的是“有没有”的问题，通过生产流程标准化，把从原料一步步地变成最终成品这一过程固定下来，用过程控制的方法将生产流程转化为自动生产的程序，然后将程序输入设备进行大批量的生产。这种“产量为王”模式的缺点是生产的产品千篇一律。如果要生产不同的产品，那么每个产品都需要一条生产线或者分时段使用一条生产线。

如图1所示，随着人们物质生活水平的不断提高，生产订单呈现出定制化、多品类、小批量的特点。与此同时，工业设备数字化率和联网率快速提升；工业设备的计算能力大幅增强，可以完成更加复杂的计算任务。这些变化催生出**柔性制造**这一全新的生产理念。柔性制造既要求每件产品都能定制，又要求高产能。这对目前的工业控制系统提出了新挑战。

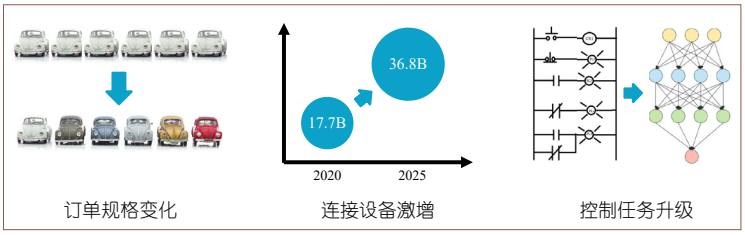


图1 工业互联网发展趋势

柔性制造带来的新挑战

柔性制造带来生产线的频繁切换，包括控制任务的转移、增加和升级。**控制任务转移**指控制任务从网络中的一台控制器转移到另一台控制器；**控制任务增加**指由新设备、新传感器引入的新控制任务；**控制任务升级**指简单控制任务（由梯形图描述的可编程逻辑控制器（Programmable Logic Controller，PLC）控制算法）升级为复杂控制任务（例如基于视觉的工业缺陷检查）。

以汽车玻璃生产为例，我们调研并分析了生产厂家在定制化生产中面临的问题及其根源[1, 2]。目前，汽车玻璃订单已呈现出品类多、批量小等特点。一个订单玻璃片数量可以从几片到几万片不等，并且小批量的订单占比很大。为了生产不同品类的玻璃，生产线需要频繁切换。据统计，每次切换需要

10 分钟更换模具、40 分钟进行配置及试生产，生产线平均每天要切换 6~8 次。假设生产线 24 小时运转，切换造成的产能损失可达 24%。另外，随着汽车产品升级和汽车主机厂商的要求不断提高，近一段时间，玻璃生产线在前期处理环节增加了 3 台设备、3 项新任务（表面检测、边缘检测、尺寸检测）和 3 类计算方式（数值运算、计算机视觉、比例积分微分（Proportional Integral Derivative, PID）控制），在印刷环节增加了 1 台设备、1 项新任务（印刷检测）及 1 类计算方式（计算机视觉），在质量控制环节增加了 2 台设备、2 项新任务（曲面检测、外观检测）及 2 类计算方式（数值运算、计算机视觉）。

为了应对上述生产模式的变化，生产企业持续磨炼内功、加班加点、改进管理，把生产线切换的损失弥补回来。但整体来讲，这些努力离柔性制造的愿景还存在较大差距。具体来说，传统工业控制系统不适应柔性生产的要求，主要存在三大问题。（1）**配置过程繁琐**：生产线切换需要先停机再重新配置 PLC 和设备间的物理连接，造成生产效率下降；（2）**规模扩展受限**：PLC 物理接口数量限制了生产线可以增加的设备数量，难以满足厂商的升级需求；（3）**升级成本高昂**：随着控制任务复杂性逐渐增加，算力不足的 PLC 需要更新换代，产生额外的停机成本。造成上述问题的根本原因在于，控制任务与执行该任务的控制器之间深度绑定，阻碍了控制任务在网络中的灵活调度。

控制器虚拟化

近年来，部分研究工作尝试使用虚拟可编程逻辑控制器（Virtual PLC）取代工业中的硬件 PLC。例如，Givehchi 等人 [3] 建立了云服务器中的虚拟 PLC，以增强工业自动化系统的敏捷性。Hegazy 等人 [4] 也将控制任务部署到云服务器，并提出了一种自适应延迟补偿器和分布式容错方法来提高及时性和可靠性。然而，由于云服务器的网络延迟和抖动，即使在私有云中，虚拟 PLC 的确定性也是有限的，它们只能满足软实时应用的要求 [3]。因此，关键的控制任务仍然依赖硬件 PLC。

为了提升网络确定性、降低网络延迟，IEEE 802.1 工作组正在推进时间敏感网络（Time-Sensitive Networking，TSN）技术的研发和标准化工作。TSN 是一系列 IEEE 标准的集合，它以时间同步为基础，设计了多种流量整形方式，最终实现低延迟、具有确定性的数据通信服务。此外，为了进一步提高可靠性，TSN 还提供了包括帧复制和消除在内的多项技术，增加冗余性。TSN 还支持统一的配置协议，实现标准化的网络管理。然而，IEEE TSN 是局域网技术，基于以太网协议研发，无法在更大范围实现网络和控制系统的实时性与可靠性。

针对云化 PLC 及 TSN 的局限性，**我们选择了一条不同于云化 PLC 的路径，将控制器虚拟化从云服务器下沉到网络中，通过在网络交换机上灵活部署控制任务，让控制功能作为一种服务，内生于网络中。**

CaaS网络架构

总体架构

我们提出了**控制即服务**（Control as a Service，CaaS）技术，**将控制任务与专用控制器解耦，协同调度控制任务与网络流量，将控制任务灵活部署到网络任意交换机中** [1]。如图 2 所示，CaaS 技术将整个工业控制网络**虚拟为一个通用控制器**，实现**控制功能虚拟化**，支持生产线灵活切换与升级。

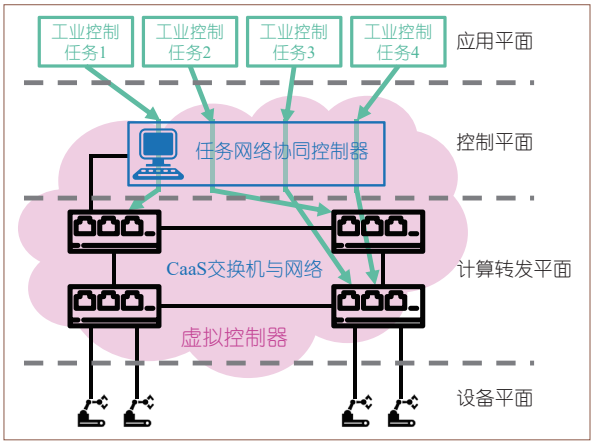


图2　控制即服务网络架构

具体来讲，支持控制即服务的实时工业网络体系结构自底向上主要分为四层：

1. 网络终端设备平面，包括大量的传感器和执行器。

2. 计算转发平面，负责完成计算和转发两项任务。传统网络中交换机只负责转发任务，新型CaaS交换机还可以完成计算任务——作为虚拟控制器，完成工业控制任务。

3. 控制平面，其核心是“任务－网络协同调度器”，协同调度网络中的控制任务分配（控制任务何时在哪个交换机上进行计算）与数据流量传输（数据流量何时由哪条路径传输）。

4. 应用平面，包括各类工业控制任务。作为一个整体，CaaS网络根据工业控制任务的自身需求和网络能力，将控制任务调度到网络中的任意交换机上执行计算，并将计算结果（即控制命令）实时发送到执行设备。

CaaS的核心是计算转发平面与控制平面。计算转发平面的核心是CaaS交换机，负责执行控制任务与转发任务。控制平面的核心是任务－网络协同调度器，统一协同调度控制任务与数据转发。

计算转发平面——CaaS交换机

CaaS交换机采用软硬件协同设计，在可编程硬件逻辑（Programable Logic，PL）部分，用硬件实现高速传输与精确计时；在通用处理器系统（Processing System，PS）部分，用软件实现复杂计算任务，包括PLC运行时、配置客户端以及时间同步。CaaS交换机的设计难点是要保证数据从输入设备（如传感器）到输出设备（如执行器）的端到端确定性。如图3所示，我们将这个确定性细分为三方面：**网络传输确定性、软硬件数据交换确定性、计算确定性。**

我们使用**时间敏感网络**保障网络传输确定性，**设计双直接内存访问**（Direct Memory Access，DMA）机制保障软硬件数据交换确定性，采用**核隔离**和**基于全局时间感知的任务计算**保障计算的确定性。其中，**双DMA**机制将CaaS元数据和时间敏感的控制数据通过不同信道分别传输，避免软硬件之间的关键数据交换受到非时间敏感数据的干扰。**核隔离**机制将配置客户端、时间同步、其他交换机功能与PLC进程隔离，通过专用CPU核心确保任务执行时长的确定性。**基于全局时间感知的任务计算**从PL的实时时钟（Real-Time Clock，RTC）模块获取到高精度全局同步时间，按照每个周期的指定时间控制任务的启动时刻，避免不同设备间CPU时间不同步带来的影响。

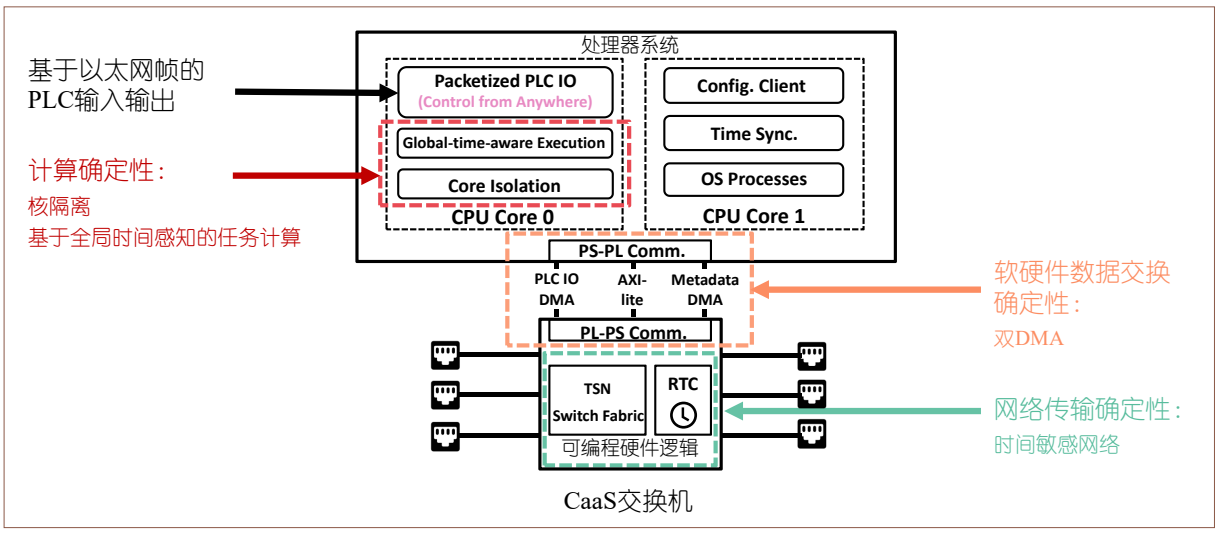


图3　CaaS交换机设计

控制平面——CaaS任务与流量调度

任务–网络协同调度器

在控制平面，任务 – 网络协同调度器需要对网络流量和任务进行合理调度，以保证数据传输和计算的确定性。

CaaS 的协同调度与 TSN 的流量调度不一样。TSN 只调度网络流量，而 CaaS 协同调度控制任务与网络流量，调度目标更复杂、调度参数更多。

调度算法需要指明每个计算任务在哪个交换机上执行、每个任务执行的开始和结束时间，以及数据流何时经过哪条网络链路。我们把任务和流量的协同调度建模成可满足模理论（Satisfiability Modulo Theories, SMT）问题，进而用 SMT 求解器进行求解。

图 4 给出了一个任务调度的例子，在这个例子中有两个计算任务（Task 1&Task 2），任务 1 以 D1 为输入设备，以 D4 和 D5 为输出设备。右侧给出了任务 1 的具体调度结果，任务 1 被分配到交换机 SW2 上执行，任务 1 的输入数据流，即从 D1 到 SW2 的数据流，经过了链路（D1, SW1）和（SW1, SW2）。该数据流在这两条链路上预留的时间槽如图 4 中标记所示。当输入数据到达 SW2 后，任务 1 开始执行。当任务 1 的计算完成后，链路（SW2, D4）和（SW2, D5）上分别预留了一个时间槽，用来传输任务 1 对应的两条输出数据流。

基于深度强化学习的流量调度

快速且高效的全局调度算法能够提升 CaaS 对关键流量和关键任务的支撑能力。虽然可以将 TSN/CaaS 协同调度转化为 SMT 问题进行求解，但是该问题属于 NP 难问题，通用的 z3、Gurobi 等求解器仅能处理较小规模的数据流调度问题。当数据流规模超过 1000 时，通用求解器需要数小时甚至数天进行求解，这对追求效率的工业场景来说是难以接受的。有些研究工作提出了一些基于启发式方法的调度方案，试图利用专家经验，在固定的时间预算内尽可能找出 TSN 调度问题的可行解 [5, 6]。这些方法虽然能够加快问题求解速度，但是依赖许多人工设计的简单规则，不一定能找到最有效的搜索策略，且通用性较差。我们观察到，尽管调度问题很难，但确认潜在调度的正确性或可满足性要简单得多。这为利用强化学习（Reinforcement Learning，RL）框架从隐含的数据分布中进行一系列试错并推导出更好的搜索策略提供了机会。

我们提出了 DeepScheduler，一种基于深度强化学习的快速可扩展的 TSN/CaaS 调度方法 [2]。与之前严重依赖专家知识或问题特定假设的方法不同，DeepScheduler 可以自动从数据流之间的复杂依赖关系中学习有效的调度策略。如图 5 所示，DeepScheduler 将 TSN/CaaS 调度当作一个多步决策问题，其中基于神经网络的代理将问题的当前状态作为输

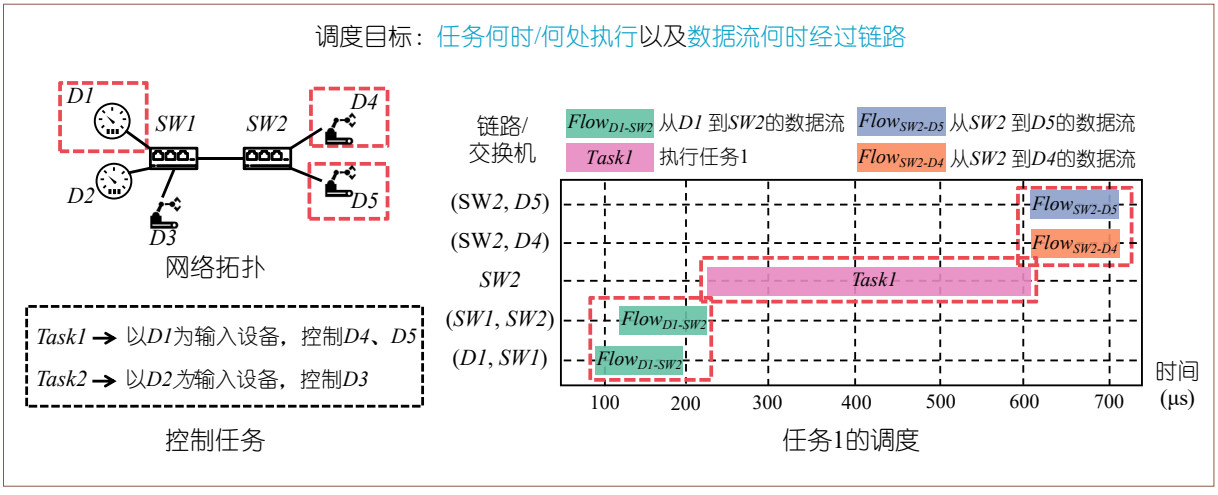


图4　任务–网络协同调度

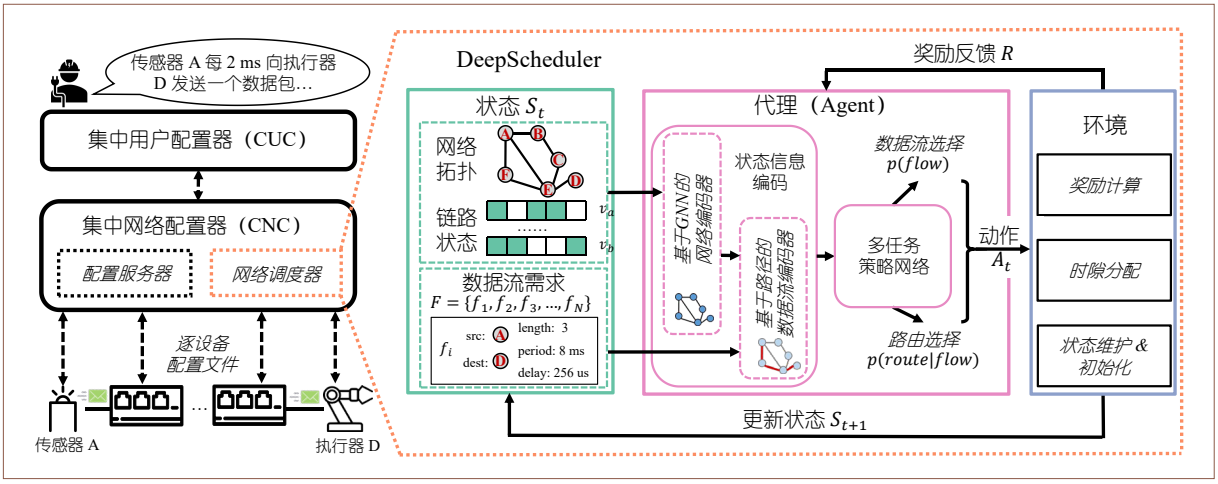


图5　基于深度强化学习的网络调度器

入并输出调度动作。环境模型会基于代理的输出维护当前状态，并向代理反馈奖励指导模型训练。为解决深度强化学习在 TSN/CaaS 调度中面临的挑战，我们提出了一种**基于图神经网络（GNN）的可扩展状态信息编码器**，利用图结构适应性地对网络拓扑和链路状态之间的复杂关系建模；设计了**基于路径的流感知编码器**，从可能路由的角度结合当前网络状态和流量需求；将调度的**决策空间分解**为两个相互依存的子任务，减小每个 RL 代理的问题规模，使模型能够有效地探索整个搜索空间。此外，我们还提出了硬样本挖掘和增量训练技术，以稳定模型训练过程。

我们将基于深度强化学习的 DeepScheduler 与多种已有调度方法在仿真和物理测试平台上进行了测试。实验结果表明，与现有的基于启发式搜索（禁忌搜索、遗传算法）和专家知识（数据流优先级、时隙分配方式）的调度方法相比，DeepScheduler 的运行速度分别快了超过 150 倍和 5 倍，能够在数秒内完成所有测试用例求解；可调度性分别提高了 36% 和 39%。DeepScheduler 带来的高效性和高成功率使流量调度问题不再是柔性制造的障碍。

CaaS交换机的设计与实现

为了实现 CaaS 网络架构，我们自主研发了“知更 Ziggo”CaaS 工业交换机和测试分析工具（http://tns.thss.tsinghua.edu.cn/ziggo/）。该交换机采用基于 FPGA 的软硬件协同设计方式，**全面支持 IEEE 802.1AS、Qav、Qbv、Qcc 等 TSN 协议[7]，支持 IT 流量与 OT 流量的共网传输，实现关键数据流量的确定性转发与超低时延传输，达到纳秒级的时间同步与微秒级的每跳时延抖动。**

“知更 Ziggo”CaaS 工业交换机基于 Xilinx ZYNQ-7000 片上系统（SoC）实现。如图 6 所示，该片上系统结合了 FPGA 的硬件可编程性和基于 ARM 的处理器的软件可编程性，与 CaaS 的设计理念相匹配。其中，交换逻辑结构在硬件中使用 Verilog 语言实现，每个端口在交换机结构中有三个 PS-PL 交互信道，分别用于控制任务的 I/O 数据、CaaS 元数据和后台流量。流量整形器使用 BRAM 实现，根据控制平面下发的调度结果控制每个队列的发送状态。我们根据 IEEE 802.1AS 实现了时间同步协议，时间戳和高精度时钟在 PL 运行，同步相关算法在 PS 部分运行。为了支持确定性计算，我们将开源项目 OpenPLC[8] 的硬件 IO 层替换为我们的数据包化 PLC IO 层。我们还通过 AXI-lite 接口将 PL 的实时时钟模块连接到 PS，实现了一个 Linux 驱动程序访问全局同步的时间。同时，我们在 CaaS 的 Linux 系统中进行了核心隔离，保障计算任务的执行。

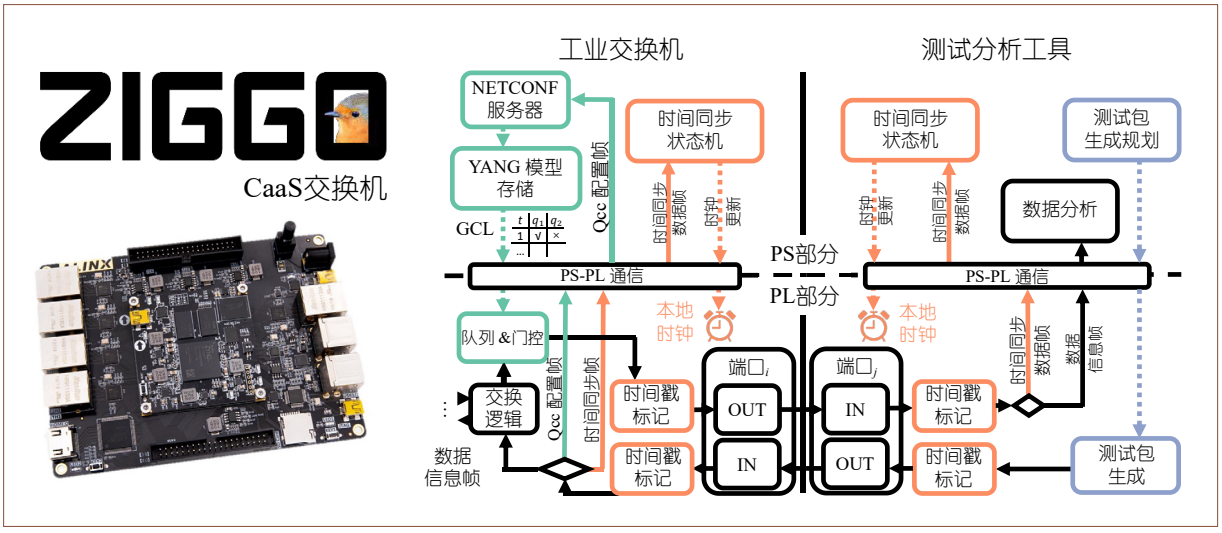


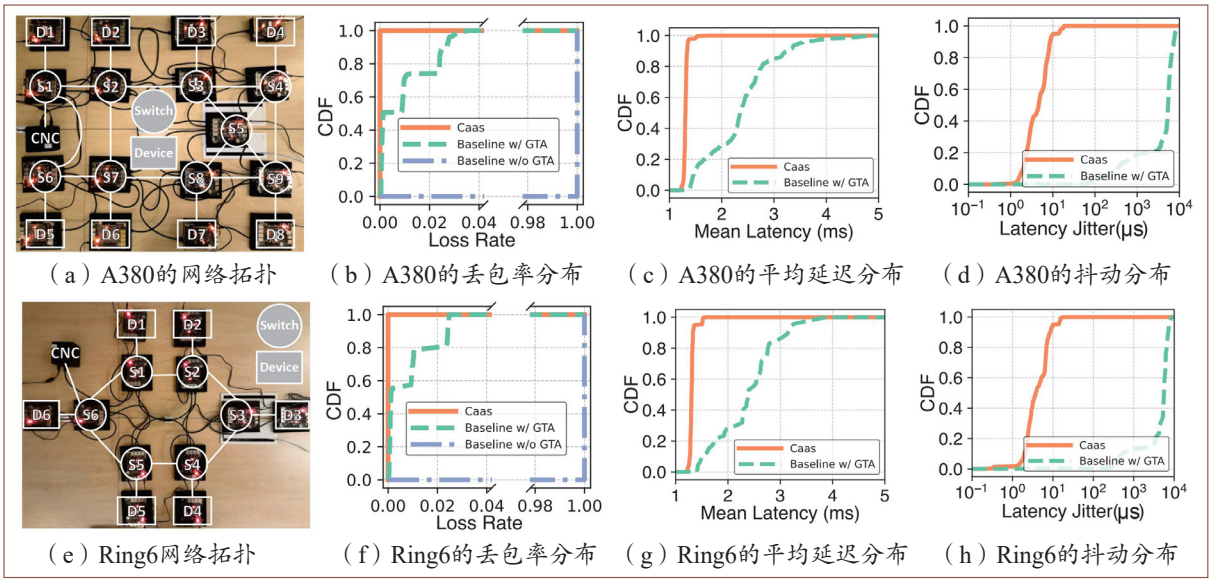
图6 “知更Ziggo”CaaS工业交换机

CaaS 系统性能测试

我们在不同的网络拓扑和控制任务需求下，对 CaaS 交换机进行组网和系统性能测试。CaaS 的目标是实现一个灵活的工业控制系统，能够满足关键控制任务的确定性要求。由于通用 PLC 和工业交换机的系统是封闭和专有的，无法与 CaaS 进行直接对比。因此，我们基于 TSN 和 OpenPLC 构建了两种单 DMA 基线方法。其中，Baseline w/ GTA 方法还加入了基于全局时间感知的任务计算能力。

如图 7 所示，我们在 A380 和环状网络（Ring6）拓扑下分别进行了 50 次实验和上万次采样。总的来说，CaaS 实现了零丢包率，并将延迟降低了 42%~45%，网络传输抖动降低了三个数量级。

图 7（b）和 7（f）展示了丢包率结果。在两个拓扑上，Baseline w/o GTA 的丢包率等于 1，即无法按计划传输任何数据包。这是因为 PLC 无法获取



（a）A380的网络拓扑 （b）A380的丢包率分布 （c）A380的平均延迟分布 （d）A380的抖动分布

（e）Ring6网络拓扑 （f）Ring6的丢包率分布 （g）Ring6的平均延迟分布 （h）Ring6的抖动分布

图7 CaaS系统性能测试

全局同步的时间。因此，我们引入了第二个基线方法 Baseline w/GTA。累积分布函数（CDF）表明，CaaS 在两个测试床均实现了 0 数据包丢失，而对于 Baseline w/ GTA，99 百分位丢包率为 3.18%（A380）和 2.47%（Ring6）。

平均延迟的分布如图 7（c）和 7（g）所示。CaaS 显示出更集中的延迟分布和比基准线更低的平均延迟。以图 7（c）为例，大多数任务的延迟在 1.31 ms 左右的狭窄范围内，而 Baseline w/ GTA 的延迟在 1.46 ms 到 3.24 ms 之间均匀分布。平均而言，CaaS 的延迟比基准线低 42%~45%。这种改进有两个原因。首先，任务 – 网络联合调度算法可以通过同时调整任务调度和流量调度全局优化调度，而两步调度方法只能找到局部最小值。其次，计算转发平面技术确保了 PL-PS 通信和任务执行的确定性。因此，CaaS 可以完美地遵循数据流规划，而基线方法总是偏离规划。

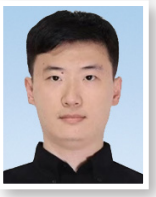
如图 7（d）和 7（h）所示，与基线方法相比，CaaS 的抖动几乎可以忽略不计。具体来说，CaaS 在 A380 和 Ring6 上的中位数抖动分别为 4.6 µs 和 3.7 µs，而基线方法的抖动分别为 5.4 ms 和 5.6 ms，相差三个数量级。主要原因是基准线中缺乏核隔离，导致计算任务执行时间不确定。

总结

本文提出支持 CaaS 的实时工业网络体系架构，将控制任务与专用控制器解耦，根据控制任务与网络流量的协同调度，灵活部署到网络的任意交换机中，并将整个工业控制网络虚拟为一个通用控制器，实现控制功能虚拟化。本文采用软硬件协同设计的方法，研发了支持 CaaS 的网络交换机及协议栈，设计了一系列保障网络确定性、计算确定性、软硬件数据交互确定性的机制，从而最终保障了工业控制任务从输入到输出的端到端确定性。CaaS 将推动工业控制网络向着分布化、虚拟化和服务化的方向迈进。 ■

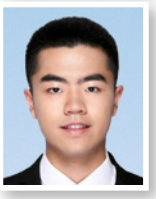
杨　铮

CCF 高级会员。清华大学副教授、博士生导师。IEEE Fellow。主要研究方向为物联网与工业互联网。
yangzheng@tsinghua.edu.cn



赵　毅

CCF 学生会员。清华大学软件学院博士生。主要研究方向为物联网、边缘计算、时间敏感网络。



贺骁武

CCF 学生会员。清华大学软件学院博士生。主要研究方向为工业互联网、边缘计算。

其他作者：党　凡

（本文责任编委：**郭得科**）

参考文献

[1] Yang Z, Zhao Y, Dang F, et al. CaaS: Enabling Control-as-a-Service for Time-Sensitive Networking[C]//*IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. 2023.

[2] He X W, Zhuge X W, Dang F, et al. DeepScheduler: Enabling Flow-Aware Scheduling in Time-Sensitive Networking[C]//*IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. 2023.

[3] Givehchi O, Imtiaz J, Trsek H, et al. Control-as-a-service from the cloud: A case study for using virtualized PLCs[C]// *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*. 2014: 1-4.

[4] Hegazy T, Hefeeda M. Industrial Automation as a Cloud Service[J]. *IEEE Transactions on Parallel and Distributed Systems*. 2013, 26(10): 2750-2763.

[5] Yan J, Quan W, Jiang X, et al. Injection Time Planning: Making CQF Practical in Time-Sensitive Networking[C]// *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 2020: 616-625.

[6] Dürr F and Nayak N G. No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN)[C]// Proceedings of the 24th International Conference on Real-Time Networks and Systems. 2016: 203–212.
[7] Institute of Electrical and Electronics Engineers, Inc. Time-Sensitive Networking (TSN) Task Group[EB/OL]. 2023. https://1.ieee802.org/tsn/.
[8] OpenPLC, Inc. Meet OpenPLC: A multi-hardware Programmable Logic Controller Suite[EB/OL]. 2023. https://openplcproject.com/.