Monaas：面向基于 TSCH 的工业物联网网络的移动节点即服务

Jinting Liu、Jingwei Li、Tengfei Chang*

arXiv:2601.03917v1 [cs.NI] 7 Jan 2026

摘要—**IEEE 802.15.4 标准的时隙信道跳频（TSCH）模式可为工业物联网（IIoT）应用提供超高的端到端可靠性和低功耗。随着工业 4.0 的发展，具有不同服务质量（QoS）要求的动态突发任务，以及如何有效管理和利用不断增加的移动设备，成为网络解决方案面临的两大挑战。现有基于 TSCH 的网络缺乏能够应对这些挑战的系统框架设计。本文提出一种新颖、面向服务的分层物联网网络架构，名为移动节点即服务（Monaas）。Monaas 旨在通过全新的架构设计和协议机制，系统地管理并调度移动节点，将其作为按需提供的弹性资源。其核心特性包括：兼顾全局协调与本地自治的分层架构、用于主动资源分配的任务驱动调度，以及按需移动资源集成机制。我们在 nRF52840 硬件测试平台上进行实现和性能评估，验证了 Monaas 链路层机制的可行性与潜力，并展现了其在特定场景中的优势。在实体 nRF52840 测试平台上，面对突发流量和链路退化，Monaas 对高优先级任务的任务完成率（TCR）始终超过 98%；而所有具有代表性的基线方案（静态 TSCH、6TiSCH Minimal、OST、FTS-SDN）均低于 40%。此外，其按需移动资源集成机制可在 1.2 秒内激活服务，至少比软件定义网络（3.5 秒）和 OST/6TiSCH（> 5.8 秒）快 65%。**

索引词—**物联网（IoT）、无线传感器网络（WSN）、TSCH、媒体访问控制、动态调度、移动节点、网络架构。**

I. 引言

因其能够在保持超低功耗的同时提供高通信可靠性，TSCH 已被应用于工业物联网（IIoT）应用中 [1]。然而，迈向工业4.0的进程带来了截然不同的系统需求，包括高度由事件驱动的流量模式、更强的空间和时间相关性，以及对低时延按需通信日益增长的需求 [2]。与此同时，现代工业环境已不再仅由静态低功耗传感器节点构成，而是越来越多地集成了自动导引车（AGV）和空中无人机等移动代理；这些设备配备了先进的《传感器》、执行器以及强大的机载计算能力 [3]。这些移动且资源丰富的设备构成了大量尚未得到利用、可按需调用的系统资源而现有的 TSCH 协议和调度研究在很大程度上忽视了这些资源。

在当今仓库和工厂的工业管理中，例如资产跟踪和环境状况监测，资源受限、成本低廉的感知设备被广泛部署，用于持续监测基本运行状态。同时，自动导引车（AGV）和空中无人机等移动平台日益被用于执行日常巡检、库存核验和维护相关任务。现有的 TSCH 研究主要关注在节点移动的情况下维持网络连接性和调度稳定性 [4], [5],，通常将移动实体视为性能扰动的来源，并认为必须尽量减小或补偿其影响。然而，这些方法在很大程度上缺乏语义层面的理解，无法将高层任务需求与可用移动资源异构的感知、执行、通信和计算能力联系起来。因此，当前基于TSCH的系统错失了重要机遇，未能实现灵活、任务驱动的资源分配，充分发挥移动工业代理按需提供能力的潜力。

为弥补这一空白，我们提出 MONAAS，这是一种面向基于TSCH的工业网络的移动节点即服务框架。MONAAS 引入了一种面向服务的调度抽象，将移动节点从被动的网络参与者提升为主动的服务提供者。通过向调度层明确公开节点能力，该框架能够基于网络中可用的感知、执行、通信和计算资源做出调度决策。与传统的 TSCH 设计相比，MONAAS 纳入了一种招募机制，使网络能够动态请求移动设备提供特定服务。该机制使移动代理（如自动导引车和无人机）能够根据不断变化的系统需求灵活参与。因此，移动资源可以按需激活并进行协调，而非静态地嵌入调度方案中。此外，MONAAS 支持根据检测到的事件发生时间和位置进行按需调度。因此，调度决策能够与服务所需的时间和地点相匹配。这种事件感知设计能够及时、高效地利用资源做出响应，同时保持 TSCH 通信的可靠性保证。

本研究聚焦于 Monaas 框架的 MAC层实现，提供基础的调度与资源编排机制，衔接高层任务语义与底层 TSCH通信原语。本文的主要贡献总结如下：

作者现供职于中国广东省香港科技大学（广州）信息枢纽（电子邮箱：jliu738@connect.hkust-gz.edu.cn；jli186@connect.hkust-gz.edu.cn；tengfeichang@hkust-gz.edu.cn）。
*通讯作者

如下：

- 我们提出了一种**面向服务的调度框架**，将移动节点建模为弹性服务资源，并通过执行区域感知调度实现任务驱动的能力匹配。
- 我们设计了一种**按需移动资源集成机制**，根据实时任务需求主动发现、招募并临时集成移动节点，而非被动处理连接问题。
- 我们为 Monaas 开发了一种**分层网络架构**，在全局协调与区域自治之间取得平衡，使本地领导节点能够及时做出决策，同时保持整个系统的可扩展性。
- 我们在基于 nRF52840-DK 的实体测试平台上，于**MAC层实现并评估了 Monaas 的核心逻辑**，结果表明任务完成率显著提升（超过 98%，而对比方案低于 40%），移动资源集成延迟也显著降低（1.2 秒，而对比方案超过 3.5 秒）。

本文其余部分安排如下。第二节回顾 TSCH 网络中现有的移动性管理方法，并指出 Monaas 所解决的根本范式差距。第三节介绍 Monaas 系统架构，包括其分层设计、核心机制以及任务驱动的资源编排策略。第四节介绍实验方法、基于 nRF52840-DK 设备的硬件测试平台、基准配置和评估场景。第五节分析实验结果，展示 Monaas 在动态环境和移动资源集成方面的有效性。第六节讨论当前局限并展望未来。

II. 相关工作

本节回顾 TSCH 调度相关工作，尤其关注工业网络中的移动性支持。我们首先简要概述 TSCH 调度的基础知识，包括时隙帧结构和基于矩阵的调度模型。随后，我们回顾现有针对移动场景提出的 TSCH 调度机制。综观这些工作，大多数方法都将移动性主要视为需要缓解的网络中断源，且缺乏对移动设备能力的语义解释。

A. 背景技术：*TSCH* 的工作原理

IEEE 802.15.4 TSCH 是一种面向高可靠、低功耗工业物联网应用的 MAC 层协议。其基本原理是将时间和频率域虚拟化为结构化的**调度矩阵**，由该矩阵管理所有网络通信。这个矩阵是一个网格，其中时间被划分为**时间槽**，无线电频谱则被划分为多个**信道**。矩阵中的特定（时间槽、信道）组合称为**单元**，它是用于通信的**网络资源**的基本单位。如图 1 所示，节点会被分配特定单元用于发送或接收数据，从而确保无竞争通信。例如，领导者可以使用专用广播单元发送命令，而成员节点则使用单播单元向其领导者报告数据。

在本文中，我们严格区分两类资源：1）**网络资源**，指用于通信的 TSCH 单元；2）**服务资源**，指节点的物理或计算能力（例如，感知、执行和数据处理）。这一区分对于理解现有方法为何存在不足至关重要：它们擅长分配网络资源，却缺乏根据具有空间和时间约束的任务需求来发现、匹配和编排服务资源的机制。

在标准的 6TiSCH 网络中，运行流程通常从新节点开始：它会监听现有节点发送的**增强信标（EB）**。这些信标包含网络同步和调度信息。新节点收到增强信标后，便会完成同步并加入网络。随后，它会与父节点协商（例如使用 6P 协议），以获取用于上行数据传输的专用时隙。此过程会建立一条通往网络根节点的可靠多跳路径。这种调度方式有两项主要优势：

- **高能效：**节点仅在分配给自己的单元内启用无线电，其他时间则保持低功耗休眠状态，从而大幅降低能耗。
- **高可靠性：** TSCH 在连续时隙中切换不同信道，有效缓解持续干扰和多径衰落，确保通信链路稳健可靠。

在介绍 TSCH 的基本原理后，本节其余部分将从若干关键维度回顾相关工作，以阐明 Monaas 架构的背景和创新之处。

B. 对 *TSCH* 移动性处理的评析：以连接为中心的视角

对现有 TSCH 文献的回顾表明，主流设计范式将移动性主要视为需要缓解的连接挑战，而非值得利用的服务机遇。这一视角直接反映了 TSCH 协议最初的问题域：这些协议旨在为基本静态的工业传感器网络提供可靠且能效高的通信。

1) 将移动性视为对连接的干扰： 基础 TSCH 调度协议，包括 TASA [6] 等集中式方案，以及 Orchestra [7] 和 ALICE [8]，等分布式方案，都是为低功耗、准静态的 TSCH 无线传感器网络设计的。它们的首要目标是在稳定的网络条件下确保极高的可靠性和能效。在这一设计背景下，节点移动性本质上被视为一种会威胁既有通信路径和调度一致性的破坏性事件。因此，这些系统所采用的移动性处理机制从根本上说具有防御性质。

基于软件定义网络的集中式方法 [9], [10]体现了这一观点，重点是在移动性下保持端到端路径的完整性。中央控制器跟踪移动节点并重新计算调度，以确保其数据能够

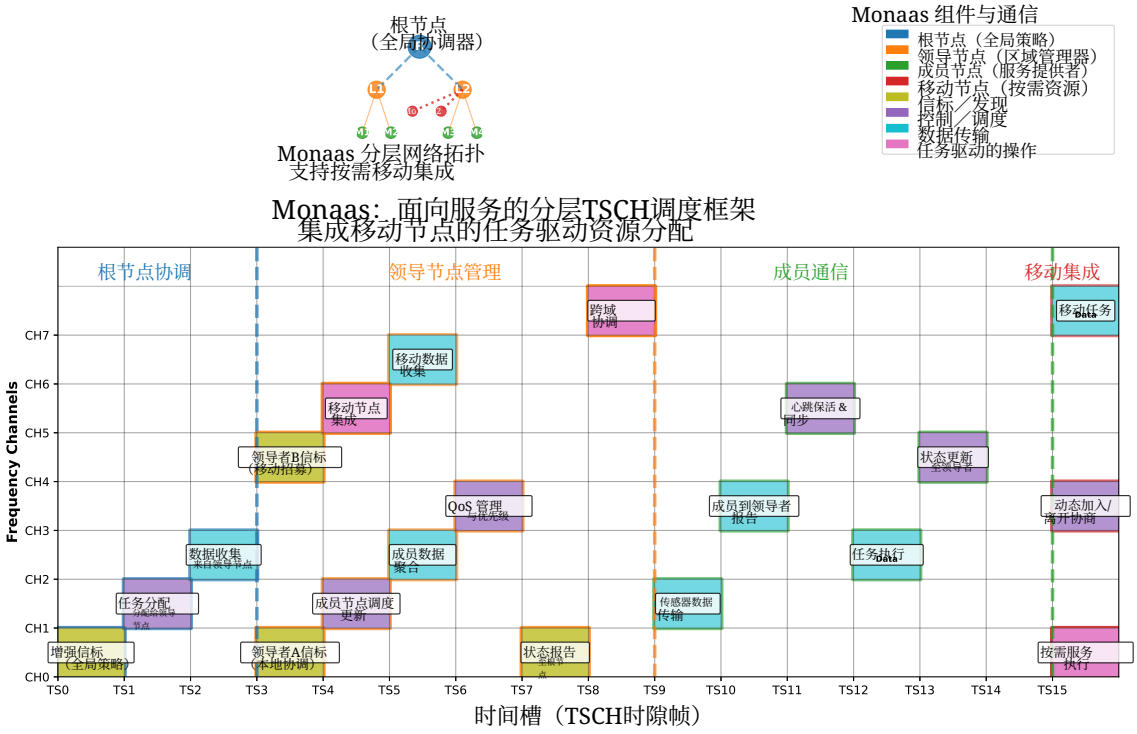


图 1。 TSCH 时序示意图，　　展示不同节点（根节点、领导者、成员节点、移动节点）在不同　　t
时间槽和信道上执行的通信任务。

继续到达网络根节点。其根本目标是保持移动节点的连接。类似地，OST 等分布式方案 [11] 通过检测链路故障并发起邻居发现来响应移动性，从而在本地修复拓扑。尽管控制结构不同，但这两种方法追求的都是反应式链路维护，而非主动式资源利用。它们解决的是如何让移动节点保持连接这一问题，而不是该节点能够提供什么功能。

除了这些基准设计之外，还有一些研究明确探讨了 TSCH 和 6TiSCH 网络中的移动性与服务质量。Al-Nidawi 和 Kemp 提出的移动性感知框架 [12]通过考虑链路动态性来提升切换鲁棒性，但仍侧重于在切换过程中维持连接。MSU-TSCH 系列方案 [13], [14] 会根据检测到的移动性事件更新本地调度计划，通过事件发生后的适应性调整，加快路由恢复和时隙单元恢复。Tavallaie 等人 [15] 提出了一种面向移动 6TiSCH 网络的流量感知调度器，改善了移动环境下的资源分配，但其驱动因素仍是流量指标，而非任务语义。从控制平面的角度来看，SDN-TSCH [16] 通过引入流量隔离来保障逐流保证，而多目标服务质量优化 [17] 则在时延、可靠性和能效之间进行权衡。尽管这些方法提升了移动性鲁棒性和服务质量保障能力，但其根本上仍以连接和流为中心。移动节点仍被视为必须维持连接的移动端点，而非能够主动编排其能力的弹性资源。

2）移动性感知中的语义鸿沟：这种以连接为中心的设计理念造成了根本性的语义鸿沟，存在于 MAC 层事件与其在应用层的意义之间。设想这样一个工业场景：一辆配备气体传感器和高清摄像机的自动导引车进入了发生化学品泄漏的区域。在 MAC 层，这一事件仅被视为通过分组接收检测到一个新的邻近节点。然而，在应用层，系统在迫切需要感知和巡检能力的地点，恰好获得了这些关键能力。现有 TSCH协议缺乏弥合这一鸿沟的机制，因为它们从地址和链路的角度理解移动性，而非从服务和能力的角度理解。

当前的移动性管理方案 [4], [13] 完全在网络层运行，无法提供将节点身份与其功能能力相关联的抽象。因此，网络能够检测到新的 MAC 地址，却无法识别这些地址所代表的资源。缺乏这种语义转换，系统便无法进行任务驱动决策，也无法主动招募合适的移动代理。从协议层面来看，自动导引车与任何其他节点都无法区分。低层网络事件与高层服务需求之间的这种脱节，从根本上限制了移动节点在 TSCH 系统中的作用。这使得移动节点无法从不稳定的端点转变为按需提供服务的服务提供者 [5]。Monaas 的设计目标正是弥合这一语义鸿沟。

C. 总结

基于文献综述，现有的 TSCH 调度方法存在三项根本性局限。首先，它们采用以连接为中心的世界观，将移动

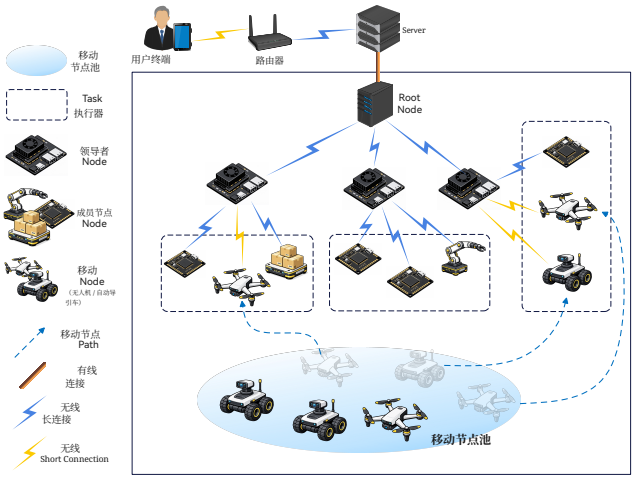


图 2. Monaas 分层架构，展示了四个关键组件：根节点（全局协调器）、领导节点（区域管理器）、成员节点（静态服务提供者）和移动节点（弹性资源）。

节点主要视为需要管理的连接负担，而非可加以利用的服务提供者。其次，它们缺乏任务语义感知，因此无法将高层任务需求（例如所需能力、执行区域或时间窗口以及服务质量约束）有效映射为具体的调度决策。第三，它们没有明确的服务资源编排机制，无法根据动态任务需求发现、匹配并协调异构节点能力。

要系统地解决这些局限性，必须回答三个关键研究问题。

- **架构设计：** 如何设计混合网络架构，在保持全局策略一致性的同时实现区域自治，从而快速响应本地动态变化，尤其是移动节点的加入和离开?
- **任务-资源匹配：** 网络调度决策如何超越传统的流量驱动模型，将任务语义（包括服务类型、优先级和服务质量要求）与异构节点能力纳入考量?
- **移动资源集成：** 如何设计协议机制，基于实时任务需求主动发现、招募、整合和调度移动节点，从而实现网络服务容量的弹性、按需扩展?

所提出的 Monaas 框架通过分层架构、语义感知任务调度机制以及移动资源集成协议设计来回答这些问题。

III. Monaas：面向移动资源的系统框架

A. 系统架构概述

图 2 展示了 Monaas 框架的整体架构及其与异构网络的交互实体。该系统采用分层设计，由一个根节点、多个领导节点、静态成员节点以及具有多样化能力的移动节点组成。

根节点（R）位于层级结构的顶端，负责维护全局策略、全网任务定义以及长期调度约束。它通过可靠的骨干链路与领导节点通信，但不直接参与细粒度的任务执行或本地移动性处理。

领导节点（L）充当区域控制器，管理局部网络区段。每个领导节点都会掌握本地成员节点（M）和动态可用移动节点（Mo）的情况，包括它们的能力、状态和近期活动。领导节点负责将从根节点接收的高层任务需求转化为特定区域的调度决策和招募操作。

成员节点代表静态、资源受限的 TSCH 设备，执行持续的感知与报告任务。这些节点按照确定性的 TSCH 调度计划运行，并构成网络的稳定骨干。

移动节点（Mo）构成一个按需提供服务资源的弹性资源池。它们可能包括自动导引车、无人机或其他配备先进感知、执行、通信和计算能力的移动平台。移动节点在移动过程中可动态加入或离开领导者区域，并沿着图中所示的虚线移动路径行进。

当任务事件发生时，负责的领导者会评估任务语义，例如所需的服务类型、执行位置和 QoS 约束。基于这些信息，领导者会从本地或邻近的移动节点池中有选择地招募合适的移动节点，并将其整合到 TSCH 调度计划中（第 III-B 节）。这一招募与整合过程通过明确的协议消息交换实现，从而支持移动资源以临时且由任务驱动的方式参与（第 III-C 节）。

通过将全局协调与根节点分离，并在领导节点处进行局部决策，该架构同时实现了策略一致性，以及对移动性和动态任务需求的快速响应。这种分层的面向服务设计使 Monaas 能够将移动节点从不稳定的端点转变为主动编排的服务提供者。

为形式化图 2 所示的架构，我们将 Monaas 网络建模为节点集合 N，该集合由单个根节点 R、领导节点集合 L、静态成员节点集合 M 以及移动节点池 Mo 组成：

$$N = \{R\} \cup L \cup M \cup Mo \tag{1}$$

每个领导节点 $l \in L$ 管理一个动态本地域$D_l(t)$，其中包括其下属的静态成员节点和当前与其关联的移动节点：

$$D_l(t) = \{m \in M \mid \text{parent}(m) = l\} \cup \{mo \in Mo \mid \text{associated}(mo, l, t)\} \tag{2}$$

这里，parent(m) 表示负责静态节点 m 的领导节点，而 associated(mo, l, t) 表示移动节点 mo 在时间 t 与领导者 l 相关联。

整体架构和领导节点的部署位置根据工业环境的实际部署情况确定。在实践中，领导者角色和区域边界可以在部署时间静态配置，也可以通过位置感知方法 [18] 动态确定。静态成员节点通常根据其部署区域分配给特定领导者。这种关联可以预先配置，也可以通过轻量级发现与关联机制建立，例如采用与移动节点类似的增强信标（EB）交换。分配完成后，静态成员节点通常会在整个部署生命周期内始终与同一领导者相关联。

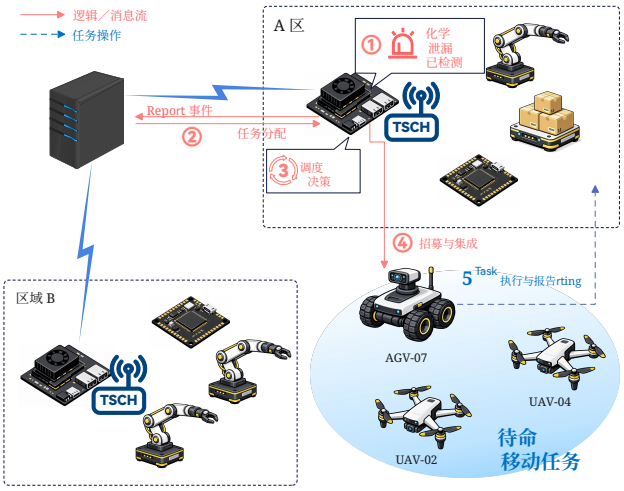


图 3. Monaas 在工厂环境中的示意性部署。根节点充当中央工厂控制器。领导节点管理特定区域（A 区、B 区），负责监管静态成员节点（例如环境传感器），并与在各区域间移动的移动节点（例如自动导引车）交互。

B. 任务驱动的能力感知调度

Monaas采用任务驱动、能力感知的调度策略，根据任务需求和节点能力预留网络带宽。在 Monaas 中，**任务** T 被正式定义为一个元组，其中封装了调度和执行所需的所有信息：

$$T = \{id, P, Q, C_{req}, Z_{target}, W_{target}\} \tag{3}$$

其中，id 唯一标识该任务。P 表示任务优先级，例如 $P \in \{1, 2, 3, 4\}$，取值包括低、中、高和关键。Q 表示服务质量（QoS）要求，包括最大可容忍时延 lat_{max} 和最低分组投递率pdr_{min}等约束。C_{req} 指定执行任务所需的能力集合，例如 {气体传感器，高速报告}。Z_{target} 定义目标执行区域，限定任务必须执行的位置。W_{target} 定义执行时间窗口，限定任务应在何时执行以及何时可以释放分配的资源。

除了定义任务外，Monaas 还使用一组运行时属性来描述每个网络节点 $n \in N$。C_n表示节点 n提供的能力集合。S_n 记录节点的实时运行状态，例如剩余电池能量 b_n。$load_n$ 表示节点 n当前的流量负载，以排队等待传输的数据包数量来衡量。

任务需求到具体 TSCH 时隙数量的转换由可配置策略函数 EstimateSlots (T) 处理。该函数将高层任务语义转换为 MAC层资源需求。

对于任务 T，每个时隙帧所需的时隙数估算如下：

$$req_slots = pkt_sf \cdot retx \cdot f_P \tag{4}$$

其中，每个项均根据任务属性推导如下。

数据包 时隙帧 表示一个时隙帧周期内生成的数据包数量，其定义为

$$pkt_sf = r_{T_{id}} \cdot T_{sf} \tag{5}$$

其中，$r_{T_{id}}$ 是任务特定数据包生成率，T_{sf} 是时隙帧周期。

重传因子 表示满足任务可靠性约束所需的重传因子，计算方式为

$$retx = \frac{Q.pdr_{\min}}{L_{est}} \tag{6}$$

其中，L_{est} 表示估计链路交付概率。通常可以通过 TSCH 网络中的保活数据包进行监控，记录传输次数和收到确认（ACK）的次数。当不要求严格可靠性时，较低的 $Q.$ 分组投递率$_{\min}$ 可以减少重传次数，从而降低时延和资源消耗。

f_P 是根据任务优先级 P得出的优先级相关缩放因子。优先级较高的任务会被分配较大的 f_P值，以提供调度余量并在发生竞争时降低时延。

最终的 请求时隙数随后转换为以下公式

$$req_slots = \left\lceil r_{T_{id}} \cdot T_{sf} \cdot \frac{Q.pdr_{\min}}{L_{est}} \cdot P \right\rceil \tag{7}$$

所得的 请求时隙数会四舍五入为整数，并由领导者用于分配 TSCH 单元。每个领导者的时隙资源由根节点预先分配，并可根据运行时的任务需求动态调整。这种方法避免了节点之间进行完全分布式时隙协商所带来的复杂性和信令开销，同时仍能实现弹性资源分配。

在全局层面，根节点维护一个时隙资源池 R_{global}，并根据历史流量模式和预期任务需求，预先为每个领导节点分配基线资源。每个领导节点 l 控制一个本地资源池 $R_l \subseteq R_{global}$，并根据任务需求和本地网络状况，在其管理域内独立为任务分配时隙。当本地资源池不足以满足新到达的任务时，领导节点会直接向根节点请求额外的时隙资源，而无需与其他领导节点进行协商。

算法 1 领导节点任务处理与资源分配 1：输入：任务 T、领导节点 l、域 $D_l(t)$、资源 $R_l(t)$ 2：输出：资源分配决策和执行计划 — 阶段 1：通信资源检查 — 3：请求时隙 ← ESTIMATESLOTS(T) 4：如果 $|R_l(t)|$ <请求时隙, 则 5：添加所需的 ← 请求时隙 $-|R_l(t)|$6：SENDTOROOT(REQ SLOTS(l, add needed)) 7：WAITFORRESPONSE(timeout) 8：如果已授予时隙, 则 9：$R_l(t)$ ← $R_l(t)$ ∪已授予时隙 10：否则 11：返回失败□请求被拒绝 12：结束 if 13：结束 if — 阶段 2：服务能力检查与招募 — 14：能力节点 ← FINDCAPABLENODES($D_l(t)$ $T.C_{req}$) 15：如果 |能力节点| < MINNODES(T), 则 16：所需数量 ← MINNODES(T) $-$ |能力节点|17：缺失能力 ← $T.C_{req} \setminus (\bigcup\{C_n \mid n \in$ 能力节点$\})$ 18：BROADCASTBEACON(T, 缺失能力) 19：候选节点 ← COLLECTREQUESTS(timeout) 20：已选节点 ← SELECTBEST(候选节点) 21：对于所有 $mo \in$已选节点, 执行 22：GRANTACCESS(mo, l) 23：$D_l(t+1)$ ← $D_l(t)$ $\cup\{mo\}$24：能力节点 ← 能力节点 $\cup\{mo\}$25：结束 for 26：结束 if — 阶段 3：最终任务分配 — 27：如果 |能力节点| ≥ MINNODES(T), 则 28：选定节点 ← SELECTOPTIMAL(能力节点 T) 29：时隙 ← ALLOCATESLOTS(选定节点 T, $R_l(t)$) 30：SENDASSIGNMENT(选定节点 T, 时隙) 31：返回成功 32：否则 33：返回失败□资源不足 34：结束 if

形式化地，领导者 l 在时间 t 的资源状态定义为

$$R_l(t) = R_{base}(l) \cup R_{alloc}(l,t) \tag{8}$$

其中，$R_{base}(l)$ 表示由根节点预先分配的基线资源，$R_{alloc}(l,\ t)$ 表示在时间 t动态授予的额外时隙。

算法 1 描述了领导者节点处理任务请求并在 Monaas 框架下执行任务驱动的资源分配的过程。每当新任务 T 到达领导者 l 时，该算法便会触发，并使用领导者当前的域 $D_l(t)$ 和本地时隙资源池 $R_l(t)$执行操作。

在第 1 阶段，领导者首先通过 ESTIMATESLOTS 计算所需的时隙数 所需时隙数 ，将任务需求转换为具体的媒体访问控制层需求。函数。如果本地可用的时隙资源不足，领导者会向根节点请求缺少的时隙。此步骤利用分层资源管理，在无需分布式协商的情况下弹性扩展领导者容量。如果根节点拒绝请求，或未能在预定义的超时时间内响应，算法将提前终止，以避免产生部分分配或不可行的分配结果。

在阶段 2，领导者会评估其当前域中是否包含足够数量的节点，以满足任务所需的能力 $T.C_{req}$。如果具备相应能力的节点数量低于任务要求的最低数量，领导者便会启动移动资源的按需招募。领导者会识别缺失的能力，并广播针对特定任务的信标，以吸引能够提供所需服务的移动节点。在招募窗口内响应的候选移动节点将被收集并评估。选定的移动节点将获准访问权限，并动态整合到领导者的域中，从而扩展可用的服务池。

在阶段 3 中，领导者根据任务语义、节点状态和调度影响，选择一组最优的、具备相应能力的节点。随后，从本地资源池分配时隙并分派给选定的节点。任务执行指令和调度信息将传达给参与节点。如果招募后仍无法获得足够的通信资源或服务能力，算法将以失败告终。

算法 1 通过将通信资源供给与服务能力招募解耦，实现了 Monaas 的核心设计。它优先考虑任务语义，而非保持连通性，并支持在领导者级控制下弹性整合移动资源。

C. 移动资源感知协议的消息流

图 4 展示了 Monaas 协议端到端的消息交换工作流程。该流程与算法 1 中给出的领导者端决策逻辑，以及由时隙需求估算模型给出的时隙需求估算直接对应。

工作流程始于根节点将任务 T 分配给指定的领导者。收到任务后，领导者使用任务驱动调度模型中定义的 ESTIMATESLOTS 函数计算所需时隙数 时隙需求估算 。通过这一计算，领导者可以确定当前本地资源池 $R_l(t)$能否满足任务需求。

如果通信资源和服务能力都充足（对应算法 1 阶段 3 中的成功路径），领导者便直接向选定的成员节点下发任务命令和分配的 TSCH 资源。执行过程中，成员节点会定期向领导者报告任务进度和感知数据。

如果服务能力充足，但本地时隙资源不足，领导者将遵循算法 1 阶段 1 中的资源扩展路径。在这种情况下，领导者根据所需时隙数 与 $|R_l(t)|$ 之间的差额，向根节点请求额外时隙。收到分配的资源后，领导者会更新其本地资源池，并继续进行任务分配。

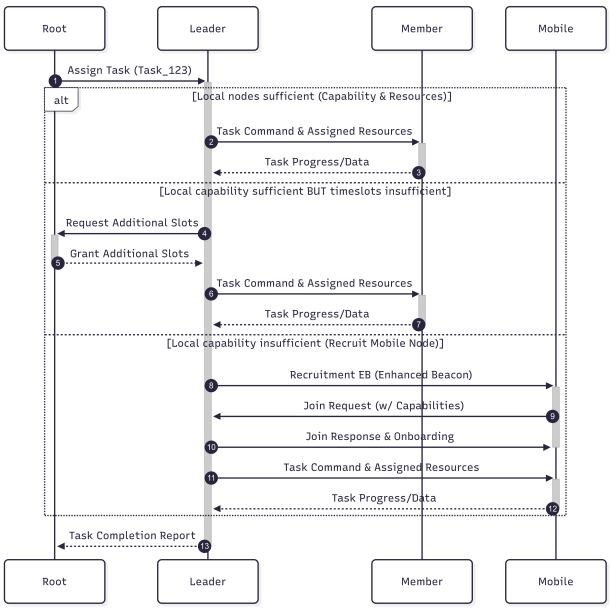


图 4. Monaas 协议的消息工作流程，展示了任务分发、分层时隙配置以及按需移动资源招募。该工作流程对应于算法 1 中领导者侧的决策逻辑。

如果本地服务能力不足，领导者将进入算法 1 阶段 2 中描述的招募路径。领导者使用增强信标广播招募消息，其中编码了任务需求以及根据 $T.C_{req}$ 得出的缺失能力。移动节点会根据任务需求评估自身的能力集，并发送包含能力描述的加入请求。选定的移动节点将完成接入，并动态整合到领导者的域 $D_l(t)$ 中。

集成后，移动节点被视为常规任务执行器，并从更新后的资源池 $R_l(t)$ 中分配 TSCH 资源。任务执行过程中，静态节点和移动节点都会向领导者报告进度和数据。

任务完成后，领导者汇总执行结果，并向根节点提交任务完成报告。通过调度逻辑、资源管理和协议消息传递之间的紧密协作，Monaas 确保在媒体访问控制层始终贯彻高层任务语义。

Monaas 的消息格式如表 I 所示。在消息工作流程中，根节点发送给领导者的分配任务消息编码为命令帧 (0x02)，子类型为 0x10（任务请求），其中包含任务标识符和高级任务参数。随后，领导者发送给成员节点或移动节点的任务命令与已分配资源消息也使用子类型为 0x10 的命令帧，其中嵌入了执行参数和已分配的时隙信息。在执行期间，任务进度/数据消息通过数据帧 (0x01) 报告，子类型分别对应传感器数据 (0x01)、节点状态 (0x02) 或事件数据 (0x03)。当本地通信资源不足时，领导者会使用子类型为 0x14（资源请求）的命令帧向根节点发送请求额外时隙消息，根节点则使用子类型为 0x15（资源响应）的命令帧进行响应，以分配额外时隙。如果本地服务能力不足，领导者会广播招募增强信标，并将其编码为信标帧 (0x00)，其中包含招募信息元素，用于指定任务标识符、所需能力、优先级、QoS 约束、执行区域、时间窗口、访问凭证和资源估算。移动节点通过加入请求消息进行响应，该消息编码为子类型为 0x02 的命令帧，其中包含其能力描述符；随后通过子类型为 0x05（确认）的命令帧确认成功入网。最后，任务完成后，领导者使用子类型为 0x13（任务完成）的命令帧向根节点报告执行结果，从而完成表 I 中定义的协议消息交换。

D. 使用 *Monaas* 的实际示例

为说明 Monaas 的端到端运行过程，我们将通过图 3 所示工厂环境中的一个实际示例进行介绍。

1) 阶段 *1*：关键任务生成：假设在 A 区检测到突发化学品泄漏，触发了由工厂控制中心发出的紧急警报，该控制中心充当根节点。根节点制定了一项高优先级任务 T_{leak}，以评估情况并控制潜在风险。

任务定义如下。

- id =泄漏扫描 A 01.
- P =关键。
- $Q = \{$延迟$_{\text{最大}} = 200$ 毫秒, 分组投递率$_{\text{最小}} = 0.9\}$。
- $C_{req} = \{$气体传感器，高清摄像机$\}$。
- Z_{target} =A 区。
- $W_{target} = [t_{now}, t_{now} + 300\,\text{s}]$。

根节点使用 CMD AssignTask 消息，将此任务分配给负责 A 区的领导者，记为 Leader A。

2) 阶段 *2*：领导节点侧调度决策：收到 T_{leak} 后，Leader A 执行算法 1 中定义的决策逻辑。

Leader A 首先使用 EstimateSlots 函数计算所需时隙数所需时隙数。在此场景中，该任务生成一个数据流，数据包速率为 $r_{T_{id}}$ =2 个数据包/秒。TSCH 时隙帧时长为 $T_{sf} = 2.02$ 秒。估计链路交付概率为 $L_{est} = 0.8$。任务优先级为关键，对应的优先级缩放因子为$f_P = 4$。

根据第三节 B 中定义的模型，计算每个时隙帧所需的时隙数如下：

$$req_slots = \left\lceil 2 \cdot 2.02 \cdot \frac{0.9}{0.8} \cdot 4 \right\rceil = \lceil 18.18 \rceil = 19.$$

Leader A 将该需求与其本地资源池进行比较，该资源池有 8 个可用时隙。按照算法 1 的阶段 1，Leader A 向根节点请求 11 个额外时隙。根节点从全局资源池中分配所请求的资源，并相应扩展 $R_l(t)$。

随后，Leader A 会评估其本地域内的服务能力可用性。ZoneA 中所有静态成员节点

表 I 帧类型及其对应子类型

帧类型	值	子类型语义
信标	0x00	[16位]任务标识符， [8位]所需能力（位向量映射）， [2 位] 优先级， [8 位]QoS 约束（百分比）， [8 位]执行区域， [16 位] 时间窗口（以时隙计）， [64 位] 访问凭证， [16 位]资源估算（以时隙计）
Data	0x01	0x01：传感器数据；0x02：节点状态；0x03：事件数据；0x04：位置数据
命令	0x02	0x01：日程更新；0x02：加入请求；0x05：确认；0x06：激活； 0x08：MAC 测试；0x09：测试响应； 0x10：任务请求；0x11：任务响应；0x12：任务进度；0x13：任务完成； 0x14：资源请求；0x15：资源响应
确认	0x03	无子类型
事件	0x04	0x01：状态变化；0x02：错误；0x03：拓扑变化

均仅能提供基本的环境感知，缺乏所需的气体感测和高清成像能力。这触发了算法 1 阶段2中所述的移动资源招募流程。

3) 第 *3* 阶段：移动节点招募与集成：Leader A 将缺失的能力需求 C_{req}嵌入其增强信标所携带的信息元素中。附近的一辆移动自动导引车（记为 AGV–07）在监听同步信号时收到该信标。AGV–07 广播其能力集 C_{AGV07} = {气体传感器、高清摄像机、机械臂}。AGV 的运行状态表明其剩余能源充足，电池电量为 85%。

AGV–07 向 Leader A 发送 REQ Join 消息以请求关联。Leader A 评估该请求，确认能力兼容性和节点健康状态后，通过 RSP Join 消息允许 AGV–07 加入其域。因此，AGV–07 成为 $D_l(\bar{t})$的临时成员，并纳入后续调度决策。

4) 第 *4* 阶段：任务执行与报告：在确保通信资源和服务能力后，Leader A 使用 CMD ExecuteSubTask 消息向 AGV–07 分配一项具体子任务。从更新后的资源池 $R_l(t)$中为 AGV–07 分配专用 TSCH 时隙。AGV–07 前往 A 区的目标位置，执行气体感测和视觉检查，并在分配给自己的传输机会期间报告测量数据。

完成后，Leader A 汇总报告的数据，并确定任务已成功执行。随后向根节点发送最终的 REPORT TaskStatus 消息，完成任务生命周期。该示例展示了 Monaas 如何将任务语义、分层资源管理和按需移动服务整合到统一且确定性的调度工作流中。

在传统的 6TiSCH 网络中，同一场景的处理方式会有根本性的不同。自动导引车 07 首先会通过标准的邻居发现和基于 6P 的协商加入网络，随后依靠应用层机制公开其感测能力，最后采用流量驱动的资源分配进行调度，而不会明确考虑任务优先级或服务语义。这一多阶段流程会带来显著时延，也无法保证自动导引车的专用能力（如气体感测和高清成像）能够被识别或

表 II Monaas 框架的符号摘要

符号	描述
N	Monaas 网络中的所有节点集合
R	负责全局协调的根节点
L	领导节点集合
M	静态成员节点集合
Mo	移动节点集合
n	一个通用节点， $n \in N$
l	一个领导节点， $l \in L$
$D_l(t)$	领导者 l 在时间 t 管理的本地域
T	Monaas 中的任务实例
id	任务的唯一标识符
P	任务优先级
Q	任务服务质量要求
lat_{max}	任务最大允许时延
pdr_{min}	最低所需分组投递率
C_{req}	任务所需的能力集
Z_{target}	任务的目标执行区域
W_{target}	任务的执行时间窗口
C_n	节点 n 提供的能力集
S_n	节点 n 的实时运行状态
b_n	节点 n 的剩余电池能量
$load_n$	节点 n 的数据包队列长度
$r_{T_{id}}$	任务特定数据包生成率（数据包/秒）
T_{sf}	TSCH时隙帧时长
pkt_sf	每时隙帧生成的数据包数（$r_{T_{id}} \cdot T_{sf}$）
L_{est}	估计链路交付概率
f_P	根据任务优先级 P 得出的优先级缩放因子
req_slots	执行任务 T 所需的时隙数
R_{global}	由根节点管理的全局时隙资源池
$R_l(t)$	时间 t 时领导者 l 的本地时隙资源池
$R_{base}(l)$	预先分配给领导者的基准时隙 l
$R_{alloc}(l,\ t)$	在时间 t动态授予的额外时隙

在紧急情况下得到有效利用。相比之下，Monaas 通过明确的能力匹配、区域感知招募和任务优先级驱动调度，实现快速集成，使移动节点从被动的连接端点转变为主动的按需服务提供者。

*E. Monaas*符号说明

表 II 展示了 Monaas 中定义的所有符号说明，供参考。

IV. 实验设置与配置

为了在真实硬件上严格评估 Monaas 架构的 MAC层性能，我们使用 Nordic nRF52840 DK 开发套件作为无线节点进行实验。每个节点运行基于

OpenWSN [1] 协议栈的固件，并采用其成熟的 TSCH MAC 层实现。Monaas 和所有基线方案的控制逻辑均直接构建在 TSCH 之上，并禁用标准网络层协议，以隔离 MAC 层调度的影响。TSCH 配置的时隙帧长度设为 11，时隙时长为 20 ms，因此时隙帧周期为 2.02s。默认情况下，每个节点的静态发送速率为每秒 1 个数据包。在不同实验场景下，实际发送速率会有所变化，下一节将对此进行说明。

测试平台部署在典型的室内实验室环境中，覆盖约 80 m^2，，并受到周围设备产生的真实射频干扰。这些节点的默认发射功率设为 0 dBm。实验网络由 11 个 nRF52840 DK 节点组成，配置精简，足以验证所提架构的核心机制。

实验期间，根节点通过串行接口将收到的所有数据包转发至主机 PC。在 PC 上运行的基于 Python 的数据收集与分析工具会实时解析日志并计算所有评估指标。

网络拓扑和移动模式经过设计，旨在突出以服务为中心和以连接为中心的集成范式之间的差异。测试网络包含 11 个节点，采用双域拓扑：根节点位于中央，两个领导节点构成部分重叠的通信域。在这些域内部署了五个静态成员节点，在成员节点、领导节点和根节点之间建立两级关联结构。三个移动节点最初处于空闲状态，预留用于按需服务招募。

每个移动节点均配置了独特的服务能力配置，以明确构造服务缺口。Mobile-1 提供用于应急响应的环境感知能力，Mobile-2 支持视觉巡检和监控，Mobile-3 则支持维护和执行任务。相比之下，静态成员节点仅具备基本的感知与通信功能，因此高级任务的执行依赖于移动招募。这些能力是逻辑定义的抽象，用于模拟异构服务以进行协议评估，而不是连接到设备上的实际物理传感器或执行器。

在移动场景中，当发布需要气体与温度感知的任务时，Mobile-1 从网络覆盖范围之外移动到领导者的通信范围内，从而触发服务驱动集成。

V. 实验分析

本节通过将 Monaas 与四种具有代表性的 TSCH 基线方案进行比较，评估其性能。这些基线方案涵盖静态调度、分布式自适应和集中式控制。它们体现了不同的设计假设，并突出了以连接为中心的调度与以服务为中心的调度之间的根本差异。

- *静态 TSCH*：一种理想化参考方案，针对固定拓扑和流量预先制定无冲突调度模式．它提供最大的确定性，但无法适应拓扑、流量或移动性变化。
- *6TiSCH 最小配置 [19]*：一种标准的动态基线方案，使用 6TOP 协议以反应式方式协商单元，从而恢复连接。该方案仍以连接为驱动，且不了解任务语义或移动节点的服务能力。
- *OST [11]*：一种流量自适应调度器，根据本地拥塞和队列状态分配资源。与 6TiSCH Minimal 相比，它提升了灵活性，但仍不考虑任务。
- *FTS-SDN [9]*：一种集中式、基于流的方法，由基于 PC 的软件定义网络控制器计算并部署全局调度。与这种单体式控制回路不同，Monaas 采用分层的控制器–领导者架构，既保留本地敏捷性，又将移动性作为一种服务资源加以利用。

这些基线为评估确定性、适应性和控制开销提供了全面的衡量标准。以下分析将观察到的性能趋势与这些架构差异直接关联起来。

A. 场景 1：动态服务能力的最低成本

此基线场景探讨了一个根本性问题：*我们是否负担得起面向服务的能力？* 在稳定条件下，静态 TSCH 自然实现了最佳性能（分组投递率 98.5%、时延 270 毫秒、无线电占空比 0.8%），这也是僵化方案的理论上限。关键发现是，Monaas 在获得完整服务编排能力的同时，性能几乎与之相同（分组投递率 98.2%、时延 350 毫秒），而能耗溢价仅为 0.3%（无线电占空比为 1.1%，而非 0.8%）。这项几乎可忽略不计的成本证明，无需牺牲基线效率，也能实现分层服务协调，因此动态能力显然值得作为一项架构投资。

能耗分析揭示了五种方法在协调策略上的显著差异。Monaas 通过高效的本地化领导者信标和成员状态同步，仅产生 1.1% 的无线电占空比；相比之下，其他动态方案的能耗高得难以承受：SDN 为维持持久的控制器连接和全局流一致性，能耗达到 2.3%（高出 2.9×）；OST 因持续进行由流量驱动的协商，能耗达到 1.9%（高出 2.4×）；而 6TiSCH 则因分布式时隙管理需要 1.4%（高出 1.8×）。Monaas 仅产生适度开销，却实现了 5.7 packets/s 的吞吐量，与静态方案的 5.8 packets/s 相近；而 SDN 的吞吐量为 5.6 packets/s，能耗却高得多。这一效率优势为 Monaas 的高级能力提供了至关重要的能耗预算，包括任务语义处理、移动资源集成和能力匹配；这些能力在动态压力下至关重要。控制平面信令开销也呈现相同的排序（静态 ¡ Monaas ¡ SDN ¡ OST ¡ 6TiSCH），这表明本地化的分层协调可避免持续的控制器心跳和点对点协商风暴（见图 5）。

[1] http://www.openwsn.org/

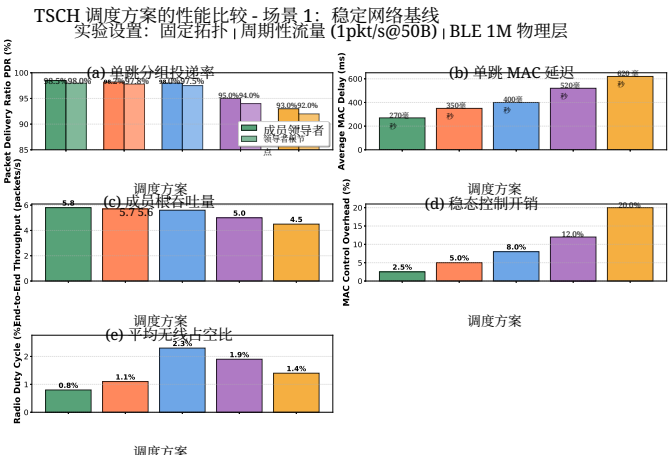


图 5. 场景 1：基线性能和开销比较。(a) 逐跳分组投递率。(b) 平均逐跳 MAC 延迟。(c) 端到端吞吐量。(d) MAC控制开销。(e) 无线电占空比。

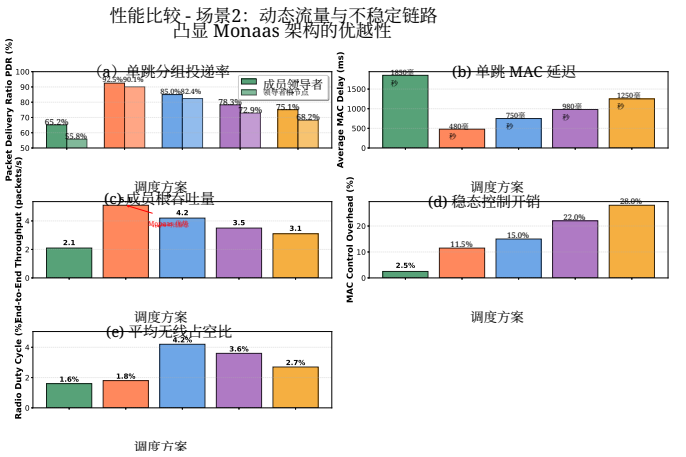


图6. 场景2：动态条件下的整体网络性能。(a) 逐跳分组投递率。(b) 逐跳媒体访问控制延迟。(c) 端到端吞吐量。(d) 控制开销。(e) 无线电占空比。

B. 场景　图 *2*：任务语义能力的高价值　　性

基于场景 1 中关于面向服务的能力只需极低成本这一发现，本场景探讨了一个相关问题：*为什么选择面向服务的方法？* 在严苛条件下——突发性高优先级任务与链路退化并存——我们在测试平台上的结果有力地证明了 Monaas 的任务语义感知能力。我们观察到，Monaas 的任务完成率（TCR）达到 99.5%，而以连通性为中心的方法则降至 $\leq$ 35%，这表明任务驱动范式在压力情境下至关重要。

性能层次进一步揭示了在应对动态变化时的架构差异：Monaas（TCR 为 99.5%，高优先级时延为 85 ms，分组投递率为 92.5%）表现最佳，因为它能将任务需求转化为即时分配无竞争路径（与第 III 节中的 EstimateSlots(T)机制一致）。SDN 的 TCR 为 96.0%，时延为 120 ms，但将优先级视为通用流；OST 和 6TiSCH 虽具备流量感知能力，却对任务语义盲感知（分别为 88.0%/180 ms 和 85.0%/250ms）；静态方案则严重退化（TCR 为 35.0%，时延为 2200ms），这可能是因为其固定容量无法适应 5$\times$ 流量突发。

从经典指标来看，我们还观察到 Monaas 在压力下仍能保持韧性：其分组投递率维持在 92.5%（静态方案为 65.2%，其他方案为 75–82%），截止期限性能近乎理想（DMR $\approx$ 98%；流量驱动基线为 $\leq$ 80%）；平均 MAC 延迟保持在 480ms（静态方案为 1850 ms）；端到端吞吐量达到 5.1 pkts/s（静态方案为 2.1 pkts/s）；无线电占空比维持在 1.8%（SDN 为 4.2%，6TiSCH 为 2.7%）；协调开销则控制在 11.5%（6TiSCH 为 28.0%）。总体而言，这些结果表明，在场景 1 中进行适度投入，能在网络承受压力时带来显著收益。

综合来看，　28$\times$ 时延差距和任务完成率（TCR）的显著差异，展现了在测试平台上面临压力时，具备任务语义感知能力的潜在优势，同时也指明了进一步开展更广泛验证的可行方向。

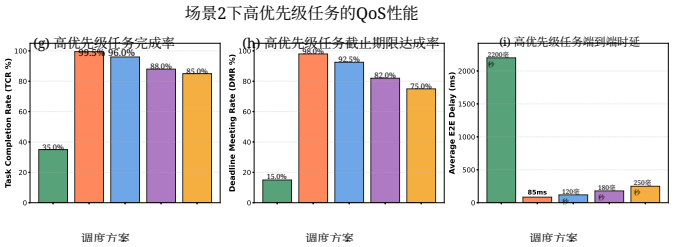


图7. 场景2：高优先级任务的QoS分析。(g) 任务完成率。(h) DMR。(i) 高优先级任务的端到端时延。

C. 场景 *3*：按需移动资源集成

此场景测试了 thirdcore 创新：以服务为中心的移动资源集成与以连接为中心的网络接入。在我们的测试平台上，Monaas 的服务激活延迟为 1.2 秒，而软件定义网络（SDN）为 3.5 秒，OST/6TiSCH 为 $\geq$5.8 秒。更重要的是，Monaas 实现了面向任务的集成（成功率为 95%），能够指挥移动节点执行特定任务；而 OST/6TiSCH 仅实现了面向连接的接入（成功率约为 30%），不具备特定任务命令能力。

观察到的性能差异似乎反映了面向服务的招募与以连接为中心的协议之间的范式差异。Monaas 的面向服务的模型利用领导节点广播包含明确任务需求的增强信标，使移动节点能够在尝试集成前评估能力匹配。这种本地化、任务感知的方法似乎能够高效地形成从能力差距检测到服务资源到达的闭环。相比之下，以连接为中心的方法采用通用协议：软件定义网络遵循边缘—云—边缘路径，这或许可以解释 3.5 秒的时延；而 OST/6TiSCH 则提供标准的网络加入机制，却不理解任务语义——它们可以建立连接，但无法引导具体的应用行为。结果表明，将移动节点视为服务资源而非连接端点，或许能实现更有效的动态集成（见图 8）。

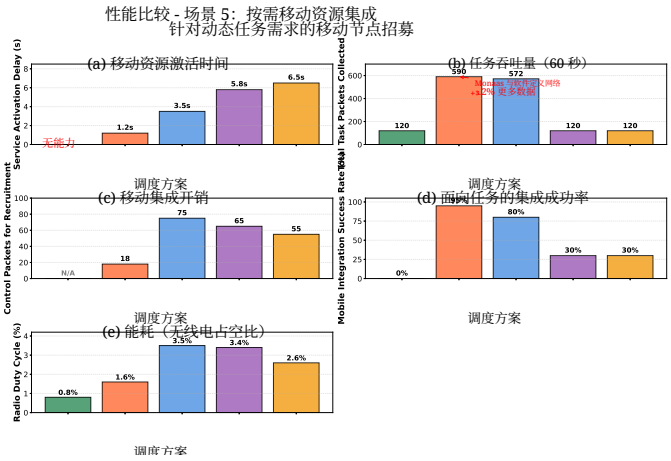


图 8。场景 3：移动资源集成的性能。

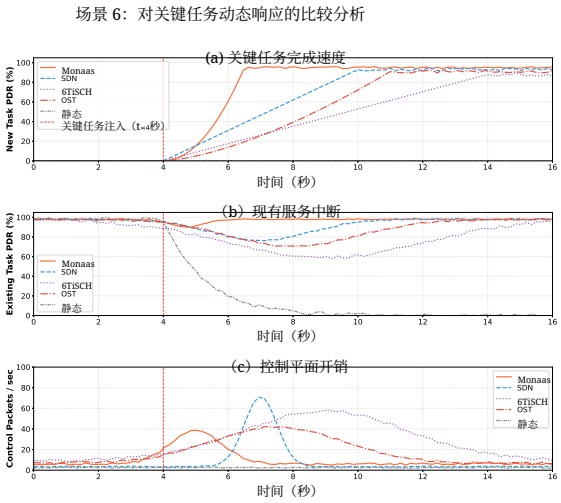


图 9. 场景 4：动态响应与架构韧性的综合比较。(a) 对新关键任务的响应速度。(b) 对现有后台服务的干扰。(c) 控制平面协调成本。

D. 场景 *4*：关键任务注入下的架构韧性

此场景测试了 Monaas 三项核心创新在最大压力下的综合集成能力：在 t=4s 时注入一项关键任务，该任务要求立即重新分配资源。结果表明，与以连接为中心的架构相比，Monaas 的面向服务的协调模型可能具有更强的韧性。Monaas 在 2.5 秒内完成任务激活，且服务中断极小（现有任务的分组投递率仅下降 8%）；而以连接为中心的方法似乎难以应对：软件定义网络耗时 6.0 秒，并造成显著的全网中断（分组投递率降至 ∼76%）；OST/6TiSCH 耗时 7.0–10.0 秒，且对现有服务造成严重影响（分组投递率降至 59–71%）。

观察到的性能模式似乎反映了根本性的架构差异。Monaas 的面向服务的模型采用本地化的领导者-成员协调机制，产生了集中的控制开销脉冲（峰值为 37 pkts/s），表明其能够高效地进行任务语义资源编排。相比之下，以连通性为中心的方法表现出不同的协调成本：软件定义网络的集中式模型产生了尖锐的信令峰值（71 pkts/s），这可能解释了整个网络范围内的中断；而 OST/6TiSCH 的分布式协商则造成了持续时间更长的开销，这可能解释了其持续的服务干扰。结果表明，与将移动节点视为网络连接端点相比，将其视为弹性服务资源，或许能够实现更具韧性的动态适应（见图 9）。

E. 实验总结

场景 1 至 5 的实验结果表明，Monaas 在多项指标上的表现均达到或超过了所有基准方案，从静态效率（场景 1 ）到综合鲁棒性（场景 5 ）皆是如此。这表明，其分层任务驱动架构在应对工业物联网动态且异构的应用需求方面具备一定的技术优势和实际可行性。

VI. 局限性与未来工作

尽管本研究验证了 Monaas 分层架构的基本可行性，但对于移动节点即服务这一概念所蕴含的潜力，我们目前才刚刚触及表面。当前实现采用单层领导者-成员结构，能够有效解决动态资源整合这一基本问题，但距离实现真正智能的物联网资源编排仍有相当大的差距。

在当前设计中，网络初始化后节点角色基本固定，这限制了系统在复杂动态环境中的适应能力。实际上，物联网节点通常具备多种能力，既可作为某些任务的执行器，也可能作为其他任务的协调器。此外，当前的任务-资源匹配机制较为粗略，主要依赖简单的服务类型标识符。真正的智能要求系统能够理解任务的深层语义、资源的多维能力，以及两者之间复杂的映射关系。

我们对下一代 Monaas 的愿景是通过实现 **动态递归层级管理**，突破这些当前角色限制。其核心理念是，任何成员节点都可以同时充当本地领导者，招募并管理其下属资源，而整个过程对其上级领导者透明。

这一设计将带来质的飞跃：

- **递归任务分解：** 复杂任务可以自动分解为多个层级，由每个子领导者根据本地信息做出最优决策。

- **邻近资源整合：** 节点可以主动招募其通信范围内最合适的资源，从而无需远程协调。

- **管理复杂性解耦：** 上级领导者无需操心下属的内部

组织，从而能够专注于高层战略。

尽管递归层次架构前景广阔，但它也带来了新的技术挑战：**多级资源冲突仲裁**、**跨层优先级传播**以及**递归性能优化**。解决这些问题需要采取跨学科方法，融合分布式系统、博弈论和人工智能的相关理论。更深层的挑战在于实现真正的涌现智能——在保持系统高效运行的同时，使整个网络展现出超越各个节点能力之和的集体智慧。

递归式 Monaas 有望在智能制造、智慧城市和精准农业等领域实现突破性应用。想象一下这样的工厂车间：每个智能工作站不仅是生产单元，也是能够自主招募自动导引车、《传感器》和机械臂的微型车间管理器。整个系统将展现前所未有的适应性和协作效率。

我们计划在未来的工作中实现这一愿景：首先建立递归层次架构的理论框架，然后开发相应的协议机制，最后在真实世界的应用场景中验证其有效性。

当前的 Monaas 仅仅是这一宏伟蓝图的第一块基石。移动节点即服务的真正时代或许才刚刚开始。

参考文献

[1] D. Dujovne、T. Watteyne、X. Vilajosana 和 P. Thubert，“6TiSCH：支持确定性 IP 的工业互联网（物联网）”，《*IEEE*通信杂志》，第 36–41 页，2014年12月。[2] Y. Lu，“工业4.0：技术、应用与开放研究问题综述”，《工业信息集成杂志》，第 6 卷，第 1–10 页，2017年。[3] H. Park、H. Kim、S.-T. Kim 和 P. Mah，“基于多智能体强化学习的时隙信道跳频媒体访问控制调度方案”，*IEEE Access*，第 8 卷，第 139 727–139 736 页，2020年。[4] S. Duquennoy、A. Elsts、B. Al Nahas 和 G. Oikonomou，“面向 Contiki 的 TSCH 和 6TiSCH：挑战、设计与评估”，载于分布式传感器系统国际会议（*DCOSS*）论文集，加拿大安大略省渥太华，2017年，第 11–18 页。[5] M. V. Ngo、Q. D. La、D. Leong、T. Q. S. Quek 和 H. Shin，“体域传感器网络中用户行为驱动的MAC调度：一种跨层方法”，*IEEE*传感器期刊，第 19 卷，2019年，第 17 期，第 7765、7755 页。[6] Y. Jin、P. Kulkarni、J. Wilcox 和 M. Sooriyabandara，“面向基于 IEEE 802.15.4e TSCH 的工业低功耗无线网络的集中式调度算法”，载于*IEEE*无线通信与网络会议（*WCNC*）论文集，卡塔尔多哈，2016年，第 1–6 页。[7] S. Duquennoy、B. Al Nahas、O. Landsiedel 和 T. Watteyne，“Orchestra：通过自主调度的 TSCH 实现稳健的网状网络”，载于*ACM*嵌入式网络传感器系统会议（*SenSys*）论文集，韩国首尔，2015年。[8] S. Kim、H.-S. Kim 和 C. Kim，“ALICE：面向 TSCH 的基于自主链路的小区调度”，载于*ACM/IEEE*传感器网络信息处理国际会议（*IPSN*）论文集，加拿大魁北克省蒙特利尔，2019年，第 121–132 页。[9] L. L. Bello、A. Lombardo、S. Milardo、G. Patti 和 M. Reno，“对用于在工业无线传感器网络中集成移动性管理的 SDN 框架进行实验评估与分析”，*IEEE*工业信息学汇刊，第 16 卷，2020年，第 8 期，第 5595、5586 页。

[10] M. Pettorali、F. Righetti、C. Vallati、S. K. Das 和 G. Anastasi，“基于 TSCH 的工业无线网络中的移动性管理”，*IEEE*移动计算汇刊，第 23 卷，2024年，第 9 期，第 8728、8710 页。[11] S. Jeong、H.-S. Kim、J. Paek 和 S. Bahk，“OST：具备流量感知能力的按需TSCH调度”，载于*IEEE*计算机通信会议（*INFOCOM*）论文集，加拿大安大略省多伦多，2020年，第 69–78 页。[12] Y. Al-Nidawi 和 A. H. Kemp，“面向时隙信道跳频 IEEE 802.15.4e 传感器网络的移动性感知框架”，*IEEE*传感器期刊，第 15 卷，第 12 期，第 7112–7125 页，2015年。[13] W. Jerbi、O. Cheikhrouhou、A. Guermazi 和 H. Trabelsi，“MSU-TSCH：物联网中 TSCH 的移动调度更新算法”，*IEEE*工业信息学汇刊，第 19 卷，第 7 期，第 7978–7985 页，2023年。[14] ——，“面向工业无线传感器网络的增强型MSU-TSCH调度算法”，《并发与计算：实践与经验》，第 36 卷，第 5 期，2024年。[15] O. Tavallaie、J. Taheri 和 A. Y. Zomaya，“面向移动 6TiSCH 网络的流量感知TSCH调度设计与优化”，载于*ACM/IEEE*物联网设计与实现国际会议（*IoTDI*）论文集。ACM，2021年，第 234–246 页。[16] F. Veisi、J. Montavont 和 F. Theoleyre，“SDN-TSCH：为具有流量隔离功能的调度式无线网络实现软件定义网络”，载于*IEEE*计算机与通信研讨会（*ISCC*）论文集，希腊罗得岛，2022年，第 1–7 页。[17] A. Vatankhah 和 R. Liscano，“面向异构物联网传感器网络时隙信道跳频调度的服务质量感知多目标增强型差分进化优化”，《传感器》，第 24 卷，第 18 期，文章编号 5987，2024年。[18] M. Pettorali、F. Righetti、C. Vallati、S. K. Das 和 G. Anastasi，“Lasa-r：面向含移动节点的工业物联网网络的位置感知调度与重新调度算法”，*IEEE*物联网期刊，第 11 卷，第 9 期，第 15 735–15 749 页，2024年。[19] X. Vilajosana、K. Pister 和 T. Watteyne，“IEEE 802.15.4e TSCH模式上的最简 IPv6（6TiSCH）配置”，RFC 8180，2017年5月。