doi:10.11887/j.cn.202501014 http://journal.nudt.edu.cn

面向军事物联网任务低时延需求的网内协同计算方法

任继军[1],李瑞彪[1],马步云[2],任智源[2*]
(1. 西安邮电大学 通信与信息工程学院,陕西 西安 710121;
2. 西安电子科技大学 综合业务网理论及关键技术国家重点实验室,陕西 西安 710071)

摘 要:为了解决军事物联网中传感数据在往返服务器应用层时的高通信时延消耗问题,提出一种面向低时延任务需求的多设备网内协同计算方法。该方法依托以P4交换机为核心的网络架构展开研究,采用基于P4程序的数据面编程策略来完成交换机内的数据包处理。设计了一种任务映射策略,将任务集映射至交换机网络拓扑,实现任务在网络拓扑路径上边传输边计算的协同作业模式。之后构建时延优化模型以找到最佳映射结果,并通过异构最早完成时间算法进一步对任务进行了最佳调度。实验结果表明,当单个数据包的数据大小为1 000 Byte时,该方法的输出时延与本地服务和云服务相比分别降低约54.2%和72.1%。因此,所提出的方法有效降低了时延,为满足任务的低时延需求提供了切实可行的解决方案。

关键词:军事物联网;协同计算;P4交换机;任务映射
中图分类号:TN925+.1 **文献标志码**:A **文章编号**:1001-2486(2025)01-147-11

论文拓展

In-network collaborative computing method for low-latency demand of military IoT tasks

REN Jijun[1], *LI Ruibiao*[1], *MA Buyun*[2], *REN Zhiyuan*[2*]
(1. School of Communications and Information Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, China;
2. State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China)

Abstract: To solve the problem of high-communication latency consumption when sensor data during the round trip to the server application layer in the military IoT(Internet of Things), an in-network collaborative computing method for multiple devices with low-latency task demands was proposed. This method relied on a network architecture with P4 switches as the core and employed a data-plane programming strategy based on the P4 program to complete the packet processing within the switch. A task mapping strategy was designed to map the task set to a switch network topology, thus realizing a collaborative operation mode in which tasks were computed while being transferred on the network topology path. After that, a latency optimization model was built to find the best mapping result, and the task was further optimally scheduled through the heterogeneous earliest finish time algorithm. Experimental results show that when the data size of a single packet is 1 000 Byte, the output latency of this method is reduced by about 54.2% and 72.1% compared with the local service and cloud service, respectively. Therefore, the proposed method effectively reduces latency, offering a practical solution to meet the low-latency demands of tasks.

Keywords: military IoT; collaborative computing; P4 switches; task mapping

近年来,无线传感器网络的迅速发展,推动物联网(Internet of Things, IoT)技术在医疗、交通、电网及军事等重要领域的应用[1]。海量轻量级设备的分布特性为任务的远程现场监测和控制带来极大的便利[2]。在军事方面,各类作战设备通过传感系统与信息网络连接,建立军事物联网。通过获取武器装备、作战个体和战场环境的状态要素和特征数据,实现战场感知透明化、武器装备智能化、后勤保障灵敏化。传统的作战模式演变为信息化作战,高速度、高精度、高强度成为战争

收稿日期:2022-10-11
基金项目:陕西省重点研发计划资助项目(2021GY-100)
第一作者:任继军(1980—),男,陕西咸阳人,高级工程师,博士,硕士生导师,E-mail:renjijun@xupt.edu.cn
***通信作者**:任智源(1983—),男,山东德州人,教授,博士,硕士生导师,E-mail:zyren@xidian.edu.cn
引用格式:任继军,李瑞彪,马步云,等. 面向军事物联网任务低时延需求的网内协同计算方法[J]. 国防科技大学学报, 2025, 47(1): 147-157.
Citation: REN J J, LI R B, MA B Y, et al. In-network collaborative computing method for low-latency demand of military IoT tasks[J]. Journal of National University of Defense Technology, 2025, 47(1): 147-157.

化常态。为满足信息化作战要求,21 世纪出现了许多对时延要求较高的军事物联网任务,如目标定位打击、军事健康监测、可穿戴设备计算等,以实施全程透明化指挥,提高作战决策效率,保障作战人员生命安全[3-5]。目前,大规模物联网设备投放战场,使任务情报的获取、处理和传输过程易于实现,但大量数据的实时处理给通信网络造成巨大压力。一些学者曾提出将任务卸载到本地或云端服务器,通过对传感数据的高性能处理来减少任务延迟[6]。然而,尽管单个服务器集群的计算性能优越,但远离传感网络和作战单元将导致高的传输延迟。在高动态战场环境中,传感器频繁采集数据并更新参数,服务器间歇性接收处理数据包。一旦网络拥塞或传输延迟过大,服务器无法及时获取数据,可能导致指挥中心错误决策。同时,长距离链路受物理破坏或攻击威胁,影响任务包传输的可靠性。因此,需要解决任务包高效、低时延的传输问题,以提升军事行动灵活性和决策能力。

目前,网内计算技术[7-11]兴起,支持任务在网络设备间的路径上传递,以渐近方式高效地完成计算。值得关注的是,软件定义网络(software defined networking, SDN)的概念被提出,网络运营商自定义数据包处理行为,使得网络设备从垂直集成模型转变为可编程平台,即增强了网络行为的扩展能力[12-13]。其中 P4 交换机作为更便捷的网络计算元素,主要表现为低延迟、低功耗及高吞吐量,且具有分布范围广、部署成本低的特点。这表明任务可随时随地卸载到设备,设备可以接收分散的情报数据。因此,引入了基于可编程 P4 的数据面编程策略[14-18],以数据包拆包、处理、封包、传输的方式,在靠近数据源的网络层优先完成包计算包转发任务,与上传至服务器相比,缩短了任务包的接收和反馈时间。当前以 P4 承载计算的场景,如带内网络遥测[19]、卫星通信[20]、网内数据聚合[21]及多媒体网络[22]等被业界广泛研究,而在 P4 与包计算包转发任务融合上,尚未出现公开发表的技术或论文。

因此,为了避免军事包计算任务往返服务器应用层的时延消耗,本文将网内计算技术与可编程 P4 交换机结合,提出同区域交换机协同工作的网内计算方法。结合图映射原理将任务集映射至交换机网络,得到一组计算节点组合,继而可分步骤协作完成任务的包计算和转发。其中采用有向无环图(directed acyclic graph, DAG)表示分割后的任务集,无向图(undirected graph, UG)表示交换机网络拓扑。同时,进一步利用异构最早完成时间(heterogeneous earliest finish time, HEFT)算法的特性,在“边计算边传输”的过程中最大限度地使用计算和通信资源,从而显著提升所提网内协同计算方法的优势。

1 网络架构模型

本节构建一种如图 1 所示的网络架构。该架构由 4 部分组成,分别为物联网云平台、P4 处理层、感知层和指挥系统。在任务执行过程中:云平台将任务卸载至 P4 处理层,P4 交换机负责情报数据的采集、处理和传输,最终反馈指挥系统执行动作,达到打击目标或更新信息的目的。

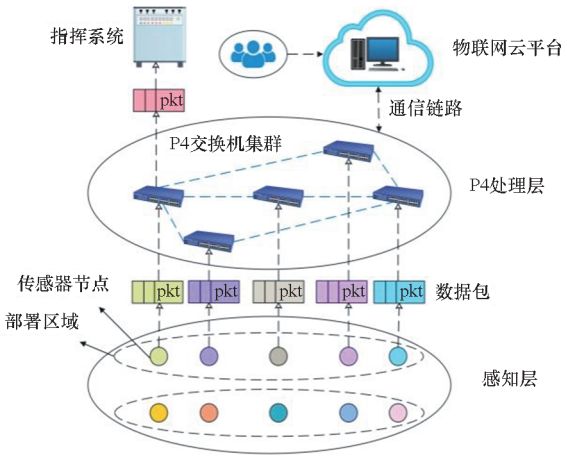


图 1 基于可编程 P4 的网络架构

Fig. 1 Network architecture based on programmable P4

1)物联网云平台:作为控制终端,其功能主要有两方面。一是对接指挥员,允许指挥员向平台下达满足自身需求的作战任务;二是实时获取处理层的网络拓扑信息和节点状态,通过平台上部署的任务映射策略将包计算任务委托给处理层。

2)P4 处理层:该部分由分布在作战要素附近的交换机组成,负责监控并接收感知层数据,同时为任务映射提供逻辑计算支持。出口交换机通过接收指挥系统反馈的数据,提供控制命令。

3)感知层:通过网络型传感器现场检测和收集数据,将数据序列化为固定格式的数据包流,上传至 P4 网络处理。每个传感器节点的 IP 数据包(pkt)包含路由信息和特定数据包类型,确保数据与指定交换机绑定。

4)指挥系统:当收到包指令时,控制作战单元发起定位、打击、管理和侦察预警等军事行动。

2　基于 P4 的数据面编程策略

在任务映射前,本节重点对交换机内部的工作原理、任务的逻辑处理流程,以及基于 P4 的整体开发流程等依次展开研究。

2.1　P4 交换机

对于 P4 交换机而言,其工作原理可使用其抽象转发模型来描述,主要包含解析器、逆解析器、匹配动作表和元数据总线,如图 2 所示。

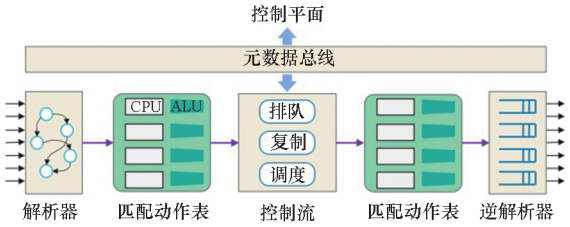


图 2　基于 P4 的抽象转发模型

Fig. 2　Abstract forwarding model based on P4

图 2 中,交换机内部组件之间形成一条分组流水线,这决定了数据包处理行为与组件的排列顺序有关,即依次进行了数据包的解析、多级流水线以及重组等操作。

数据包载入交换机端口后,首先通过解析器拦截,利用解析图将包头与载荷分开。与载荷不同,包头中定义了许多不同类型的数据,这些数据被提取出来并参与后续匹配。此外,程序员可以根据所需的任务数据,灵活定义包头结构和解析流程。解析完成后会经过多级流水线处理。多级流水线是多级匹配动作表在控制流(排队、复制以及调度)基础上的逻辑连接,记录数据在表之间的状态变化,需通过控制平面来驱动。匹配动作表包含关键字、动作和操作数据,匹配时触发动作,修改头部数据。当这一阶段完成后,逆解析器重新组装数据包,以便在逆解析阶段传输。

2.2　P4 程序

分组流水线使用 P4 语言进行交换机硬件实现,旨在将 P4 程序定义的抽象结构映射到设备的物理体系结构,以驱动其运行。主要有 5 个基本程序,即头部声明、匹配动作表、控制流、解析和逆解析。

1)头部声明数据源地址、目的地址、版本、协议以及服务类型等信息,同时定义了包头格式。其中头部的部分字段是匹配动作表的匹配字段,用来触发动作的响应。

2)匹配动作表又称查找表,被程序员定义为关键字与动作的多组合匹配形式。针对某一操作的处理过程定义在动作内部,且多个关键字的连续操作实现特定功能。

3)控制流是用于表示多个匹配动作表(顶点)之间依赖关系(边)的有向无环图,其依赖关系决定了数据包在不同表间的跳转路径,指示包头如何读/写、修改。

4)解析是在识别包头后提取相关字段。解析过程被抽象地表示为一个有限状态机,如果发生状态转换,解析器则改变解析数据包的方式。

5)逆解析需要两个输入数据源,分别为分组载荷和分组头部。将这两个输入源重构为一个数据包的过程为逆解析。

编写 P4 程序是在配置阶段,其目的是预先制定规则以定义数据平面行为。配置文件内容包括解析图、控制流程序、表配置和动作集。在运行阶段,数据平面会运行 P4 程序来处理数据包。

2.3　任务的逻辑处理流程

为进一步明确交换机如何基于分组流水线执行任务逻辑,图 3 给出以单个传感器、单个交换机以及指挥系统为主线的处理流程。其中,交换机的多级流水线(查表、动作和流控制)是任务处理的核心内容。

具体的处理流程为:当传感器发送原数据包 data1 时,交换机处理后生成现数据包 data2,并在任务结束前转发至指挥端。现数据包的 data2 格式需与指挥端预命令格式一致,否则指令不生效。交换机需要根据查找表对包格式进行判断和重塑。交换机的查找表中定义了格式和任务数据字段;动作集中定义了格式判断(If)、生成(Generate)和具体任务执行(Ops)等动作,且分别被存储在不同的表当中。其中,If 操作用来判断包格式是否满足一致要求;Generate 负责将不满足要求的包格式重塑为新数据包格式;Ops 中包含了具体的任务处理操作。

当交换机收到原数据包时,首先解析头部和载荷,匹配头部的格式字段与查找表中的格式字段,执行相应的 If 动作。若 If 条件不满足,继续匹配表中 Generate 字段并执行格式重塑;若条件满足,则匹配包头的其他字段与查找表中的任务数据字段,执行 Ops 操作。最后,经过逆解析后的现数据包被转发至指挥端或下一跳交换机。

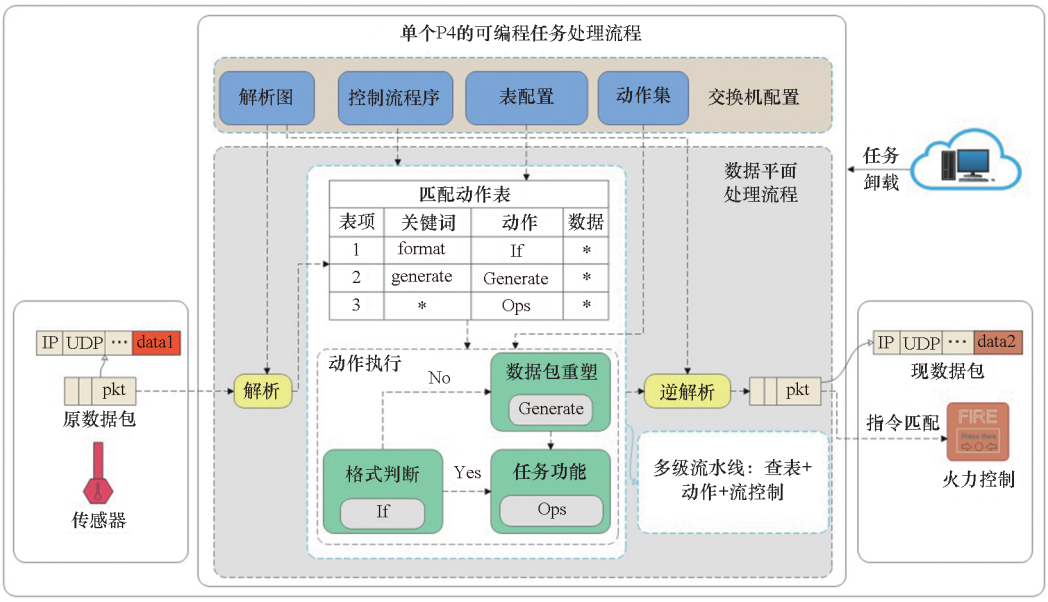


图 3 基于单个 P4 的任务功能实现

Fig. 3 Implementation of task function based on a single P4

2.4 基于 P4 的开发流程

下面概括 P4 交换机完整的开发流程。

步骤 1:根据 P4 标准,自定义相关的任务程序,包括数据帧解析、匹配动作表定义和包的流控制过程。特别地,任务处理行为被定义在匹配动作表中。

步骤 2:编译 P4 程序,得到交换机的二进制配置文件和运行时 API(交换机功能驱动)。

步骤 3:将配置文件加载到对应交换机芯片 Tofino,更新数据帧解析的状态和匹配动作表的内容。

步骤 4:交换机接收外部数据包,对更新后的匹配动作表遍历并执行指令动作。同时,在流控制下增加、删除、修改、查询表项。

开发流程如图 4 所示,图中 P4 架构提供可供任务开发的相关组件以及接口。编译器负责将指定的程序配置映射到目标设备。配置策略有两种:一种是解析器配置,规定了包头的顺序;另一种是表配置,实现逻辑表映射到交换机管道中的内存[23]。此外,交换机的控制平面允许表项通过

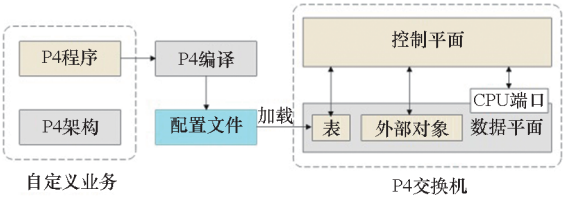


图 4 P4 的开发流程图

Fig. 4 Workflow diagram of P4

表依赖图(一种定义逻辑表之间依赖关系的 DAG)进行外联逻辑操作,该平面也可控制外部对象来执行附加功能。

3 任务映射策略

虽然单个交换机可以线速处理数据包,但其有限的通信范围使其难以正常接收来自较远位置的传感数据。因此,本节采用任务映射策略,通过交换机集群计算的方式来解决该问题。

3.1 DAG 表示的 IoT 任务

一般而言,一个完整的计算任务有多个相互独立的功能模块,每个模块相当于一个逻辑上完整的子任务。特别地,子任务之间的任务依赖约束,即前置和后置任务在时间上的执行优先级关系,可以用一个 DAG 来明确表示。

本小节将 DAG 驱动的军事监测任务抽象描述为 $G=(M,E)$,M 和 E 分别被定义为子任务集合和关联子任务的有向边集合。其中 $M=\{w_1,w_2,\cdots,w_s,w_{s+1},\cdots,w_{l-1},w_l \mid s\geqslant 1,l>s+1\}$,$w_1,w_2,\cdots,w_s$ 表征入口任务;$w_{s+1},\cdots,w_{l-1}$ 表征中继任务,w_l 表征出口(目标)任务。入口、中继、出口任务间相互独立,且每个子任务的处理逻辑不同于其他任务。此外,中继或目标任务的前向子任务集合表征为 $\Phi_{\uparrow}(w_i)=\{w_j \mid (w_j,w_i)\in E\}$。$E$ 中边的权重表示任务处理时延,边的方向表示数据包的转发方向。

3.2 UG 表示的 P4 网络拓扑

由一组 P4 交换机组网生成的网络拓扑图可

以用 UG 表示,即 $N=(V,K)$。其中,$V=\{v_1,v_2,\cdots,v_s,v_{s+1},\cdots,v_{t-1},v_t \mid s\geqslant 1,t>s+1\}$ 为所有 P4 处理节点集合,K 为 N 的边集合。为便于 DAG 任务映射,将 $v_1,v_2,\cdots,v_s$ 作为起始处理节点,$v_{s+1},\cdots,v_{t-1}$作为中继处理节点,v_t 作为目标处理节点。节点间的边为无向边,允许双向转发数据包。

若图 N 为连通图,其点割集用 $S\subseteq V$ 表示,其边割集用 $S'\subseteq K$ 表示,其连通分支数用 $\omega(N)$ 表示。当移除图 N 中的最小点割集或最小边割集使其不连通时,移除的顶点或边的数量分别称为点连通度或边连通度,它们通常被用来衡量网络拓扑的抗毁性能。其中,点连通度用 $\min\{|S|:S\subseteq V(N),\omega(N-S)>1\}$ 表示,边连通度用 $\min\{|S'|:S'\subseteq K(N),\omega(N-S')>1\}$表示。

给定从 v_i 到 v_j 传输单位量数据的最低时延,即 $d_{v_iv_j}$,并将 $p_{v_iv_j}$称为基于 $d_{v_iv_j}$获得的最短路径。因此,相应地得到最短路径集,即 $P=\{p_{v_iv_j} \mid v_i,v_j\in V\}$。考虑节点间直接或间接连通,利用其边权重和节点依赖关系,再依托 Dijkstra 算法可得到 $d_{v_iv_j}$和 $p_{v_iv_j}$。此外,起始处理节点到目标处理节点的最长路径称为关键路径。

3.3　DAG 至 UG 的映射规则

基于上述定义,在云平台部署一种以处理层交换机为目标设备的任务映射策略,包括节点映射和边映射规则。

定义 1　任务节点映射,即军事任务在 P4 集群的部署策略。定义 $\zeta:M\rightarrow V$,其中 ζ 表示任务部署节点集。旨在将入口任务 $w_1,w_2,\cdots,w_s$ 映射到 P4 的起始处理节点 $v_1,v_2,\cdots,v_s$;中继任务 $w_{s+1},\cdots,w_{l-1}$映射到中继处理节点 $v_{s+1},\cdots,v_{t-1}$;出口任务 w_l 映射到目标处理节点 v_t。映射结果表示为

$$\zeta(w_i)=\begin{cases}v_i, i\in\{1,2,\cdots,s\}\\ v_t, i=l\\ v_j, v_j\in V/\{v_1,v_2,\cdots,v_s,v_t\}\end{cases}\tag{1}$$

定义 2　边映射,即 DAG 表示任务的有向边映射到 UG 的无向边。定义 $\psi:E\rightarrow K$,其中 ψ 表示边映射集合,满足如下关系:

$$\psi(w_i,w_j)=p_{\zeta(w_i)\zeta(w_j)}\tag{2}$$

式中:ψ 将 E 中的所有有向边映射到节点的最短路径。

下面给出简单的映射示例,如图 5 所示。图 5中,在 DAG 任务至 UG 的映射基础上,考虑了 UG 中映射节点对每个不同格式数据包的处理操作,如 If、Generate、Ops 等动作,同时包的转发路径与任务流的方向一致,以确保任务处理的准确性。

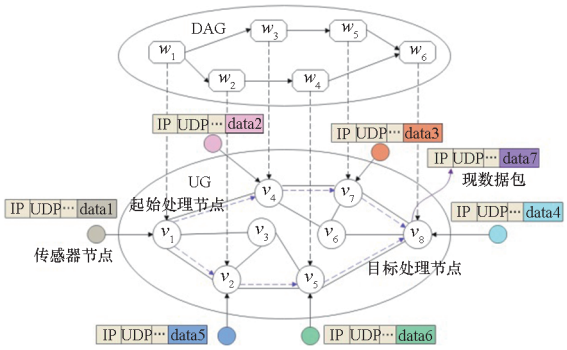


图 5　DAG 至 UG 映射示例

Fig. 5　Example of DAG to UG mapping

4　时延优化模型

由于不合理任务映射而带来的波动延迟需要优化,本节将构建一个以时延为优化目标的数学模型。在任务处理期间,某个 P4 节点执行到下一个节点或目标处理节点的总时延是其最短路径上的最大时间跨度,具体包含了处理时延、累积时延。其时延模型表示为

$$T_{\text{sum}}(w_i)=T_{\text{proc}}(w_i)+T_{\text{accu}}(w_i)\tag{3}$$

式中:处理时延 $T_{\text{proc}}(w_i)$是设备处理任务 w_i 时所消耗的时间,表示为

$$T_{\text{proc}}(w_i)=\frac{D_{w_i}\alpha}{C_{\zeta(w_i)}}\tag{4}$$

其中 D_{w_i}为处理的数据包大小,α 为任务的逻辑复杂度,$C_{\zeta(w_i)}$为设备的处理能力。

累积时延 $T_{\text{accu}}(w_i)$表示子任务在当前节点开始处理前所消耗的时间,具体为前向子任务的总时延、传输时延、队列时延之和,表示为

$$T_{\text{accu}}(w_i)=\max_{w_j\in\Phi\uparrow(w_i)}\{T_{\text{sum}}(w_j)+T_{\text{trans}}(w_i)+T_{\text{que}}(w_i)\}\tag{5}$$

式中:$T_{\text{trans}}(w_i)$表示传输时延,可表示为

$$T_{\text{trans}}(w_i)=d_{\zeta(w_j)\zeta(w_i)}\times D_{w_jw_i}\tag{6}$$

其中 $d_{\zeta(w_j)\zeta(w_i)}$表示两映射节点间传输单位数据量的最小延迟,$D_{w_jw_i}$表示当前传输数据包的数据量大小;队列时延 $T_{\text{que}}(w_i)$指数据包在传送前在队列中等待的时间,可表示为

$$T_{\text{que}}(w_i)=\frac{L\times I_{\text{que}}}{B}\tag{7}$$

其中 L 为数据包长度,I_{que}为队列平均长度,B 为传输速率。由于多个设备协同作业时的通信距离较短,因此传输时延的影响可忽略不计。

通过上述时延模型可知,目标处理节点的输出时延即为任务图 G 所消耗的总时延,可表示为

$$T_{\text{sum}}(G) = T_{\text{sum}}(w_l) \tag{8}$$

考虑 DAG 至 UG 映射结果的随机因素,定义映射矩阵来统一表示所有的映射关系,映射矩阵表示为

$$\boldsymbol{X} = \begin{bmatrix} x_{w_1v_1} & \cdots & x_{w_1v_s} & \cdots & x_{w_1v_t} \\ x_{w_2v_1} & \cdots & x_{w_2v_s} & \cdots & x_{w_2v_t} \\ \vdots & & \vdots & & \vdots \\ x_{w_lv_1} & \cdots & x_{w_lv_s} & \cdots & x_{w_lv_t} \end{bmatrix} \tag{9}$$

式中:$x_{w_pv_q} \in \boldsymbol{X}$ 表征全部子任务与 P4 节点的映射结果,若其值为 1,则该子任务 w_p 被部署到 P4 处理层的 v_q 节点;若其值为 0,则子任务 w_p 未被部署到 v_q 节点,即

$$x_{w_pv_q} \in \{0,1\}, \forall w_p \in M, \forall v_q \in V \tag{10}$$

结合映射原理,时延模型的表示形式更新为

$$T_{\text{sum}}(w_i) = \max_{w_j \in \xi(w_i)} \left\{ T_{\text{sum}}(w_j) + \sum_{v_p, v_q \in V} d_{v_iv_j} D_{w_jw_i} x_{w_jv_q} x_{w_iv_p} \right\} + \sum_{v_p \in V} \frac{D_{w_i}\alpha}{C_{v_p}} x_{w_iv_p} \tag{11}$$

因此,$T_{\text{sum}}(G)$ 可表示为 $\boldsymbol{X}$ 的函数,即

$$T_{\text{sum}}(G) = F(\boldsymbol{X}) \tag{12}$$

综上所述,为了从映射矩阵中找到最佳的映射结果,建立如下时延优化模型:

$$\begin{cases} \boldsymbol{X} = \arg\min_{x_{w_pv_q} \in X}(F(\boldsymbol{X})) \\ \text{s.t.} \begin{cases} x_{w_pv_q} \in \{0,1\}, \forall w_p \in M, \forall v_q \in V \\ \sum_{v_q \in V} x_{w_pv_q} = 1, \forall w_p \in M \\ x_{w_pv_q} = 1, p = q, \forall p \in [1,s] \\ x_{w_pv_q} = 1, w_p = w_l, v_q = v_t \\ x_{w_pv_q} = 0, q \in [1,s], \forall p \in [s+1, l-1] \\ x_{w_pv_q} = 0, v_q = v_t, \forall p \in [s+1, l-1] \\ T_{\text{sum}}(w_p) = 0, \forall p \in [1,s] \end{cases} \end{cases} \tag{13}$$

5 HEFT 算法

在映射优化的基础上,为了进一步完成优化目标,本节采用一种基于 HEFT 的表调度算法[24]。该算法适用于解决多设备协同工作场景中具有任务依赖性约束和低复杂度的 DAG 多任务调度问题,能够预先评估所有设备处理数据包和通信的能力,在满足整个任务依赖的同时,将每个任务映射到计算和通信资源集中的路径设备上,从而减少连续任务的总完成时间。其设计思想分为两个阶段:同级任务的优先级排序和任务的调度。

在任务排序阶段,需要参考的两个指标分别由式(14)和式(15)得到。

$$\overline{e_i} = \frac{1}{m} \sum_{j=1}^{m} e_{i,j} \tag{14}$$

式中,$\overline{e_i}$ 表示任务 w_i 的平均处理时间,m 为设备总数,$e_{i,j}$ 为在设备 v_j 上运行任务 w_i 的时间。

$$\overline{c_{i,j}} = \overline{h} + D_{w_iw_j}/\overline{B} \tag{15}$$

式中,$\overline{c_{i,j}}$ 表示子任务 w_i 执行到 w_j 时数据包传输所花费的平均通信成本,$\overline{h}$ 为设备的平均启动时间,$\overline{B}$ 为设备间的平均传输速率。

任务的排序值可通过下式得到:

$$r(w_i) = \overline{e_i} + \max_{w_j \in s(w_i)} \{\overline{c_{i,j}} + r(w_j)\} \tag{16}$$

式中:$r(w_i)$ 表示子任务 w_i 与出口任务间的关键路径长度,其中具有高排序值的任务优先被调度;$s(w_i)$ 表示子任务 w_i 的一组直接后继子任务。对于出口任务而言,其排名值可以表示为

$$r(w_{\text{exit}}) = \overline{e_{\text{exit}}} \tag{17}$$

在任务调度阶段,通过映射优化模型(13)和式(11),得到任务 w_i 在每个设备执行时的最早完成时间 e_{ft},并选择最小 e_{ft} 所对应的设备进行任务调度。HEFT 算法步骤如算法 1 所示。

算法 1 HEFT 算法

Alg. 1 HEFT algorithm

输入:DAG 和 P4 网络设备集

输出:最佳调度结果

1. 通过式(14)得到每项任务的平均计算开销,通过式(15)得到两个相关设备间数据包的平均通信开销
2. 根据式(16),遍历计算所有任务(从出口任务开始,到入口任务结束)的排序值 r,并以非递增的方式排序后存储在任务排序列表中
3. 初始化:$i = 1$,代表列表中的第 1 项任务
4. **while** 列表中存在未安排的任务 **do**
5. 对列表中的任务 w_i 进行调度
6. **for each** $v_j \in V$ **do**
7. 由式(13)得到任务 w_i 在设备 v_j 的最早完成时间 e_{ft},即 $e_{\text{ft}}(w_i, v_j)$
8. **end for each**
9. 找到最小化的 e_{ft},并将任务 w_i 安排至设备 v_j
10. $i = i + 1$
11. **end while**

6 仿真测试

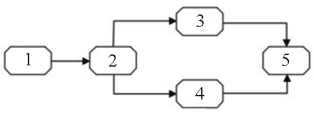
本次仿真在基于 Python 的软件平台上进行

模拟,旨在验证网内协同计算方法的可行性。测试平台配置为 i5 - 7200U CPU 和 8 GByte 内存。首先,评估了网内协同计算、本地计算和云计算三种计算范式在包计算与转发任务中的时延表现。然后,分析了包格式重塑和网络拓扑边连通度对时延的影响。最后,为展示 HEFT 任务调度算法的性能优势,比较了 HEFT 与其他调度算法在任务完成时间上的差异。

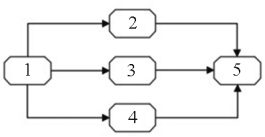
6.1　准备工作

实际应用场景中,不同作战任务内部的逻辑处理方式多样化。为不失一般性,对照真实的任务范例,设计了 3 种具有不同依赖关系的 DAG 任务模型,如图 6 所示。

(a) 任务模型 1
(a) Task model 1



(b) 任务模型 2
(b) Task model 2

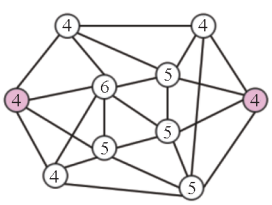


(c) 任务模型 3
(c) Task model 3

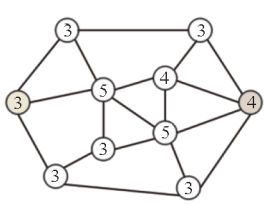
图 6　用于仿真的任务模型
Fig. 6　Task model for simulation

同时,为符合作战场景中交换机随机分布的特点,利用 Python 平台产生如图 7 所示的 4 种不规则网络拓扑(按照设备的边连通度划分),指代 10 个组网的 5 交换机代理节点。每个节点上标记了与其关联的边的条数。由 3.2 节中边连通度的定义可知,网络拓扑Ⅰ、Ⅱ、Ⅲ、Ⅳ的边连通度依次为 4、3、2、1。

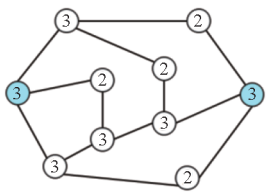
此外,测试所用参数参照文献[25 - 26]进行



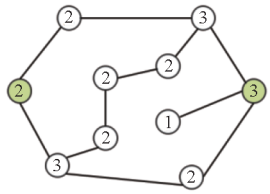
(a) 网络拓扑结构Ⅰ
(a) Network topology Ⅰ



(b) 网络拓扑结构Ⅱ
(b) Network topology Ⅱ



(c) 网络拓扑结构Ⅲ
(c) Network topology Ⅲ



(d) 网络拓扑结构Ⅳ
(d) Network topology Ⅳ

图 7　用于仿真的 P4 网络拓扑
Fig. 7　P4 network topology for simulation

设置,如链路传输速率和设备处理能力等参数基本满足交换机在 4G(10 ~ 100 Mbit/s)网络下工作的实际需求,但没有考虑传输过程中丢包后重传的时延消耗。实验参数的具体设置如表 1 所示。

表 1　仿真参数
Tab. 1　Simulation parameters

参数	值
单个交换机的处理能力/GHz	[2,5]
P4 网络的数据链路传输速率/(Mbit/s)	[25,50]
本地服务器的处理能力/GHz	10
本地数据链路传输速率/(Mbit/s)	8
云服务器的处理能力/GHz	10
云数据链路传输速率/(Mbit/s)	4
任务的逻辑复杂度/(cycles/bit)	100
数据包字段长度/Byte	[46,1 480]

6.2　仿真结果分析

6.2.1　三种计算范式的时延比较

本小节采用图 6 的任务模型 2 和图 7 的网络拓扑结构Ⅱ,对基于 HEFT 算法的 P4 网内协同计算、本地计算和云计算的时延进行评估。设置每个 P4 节点接收数据包大小相同,测试结果如图 8 所示。

图 8 的结果表明,相同负载下,网内协同计算的延迟最小,且链路时延的降低幅度最大。以包负载为 1 000 Byte 为例,网内协同计算的链路时延分别比本地计算和云计算降低约 28.27 ms 和 69.91 ms。当负载较小时,由于本地计算和云计算具备更强的计算能力,能够在一定程度上弥补较长传输距离带来的时延劣势。但随着包负载的增大,这些计算模式的计算优势逐渐减弱,传输时延逐步增加。相比之下,网内协同计算的时延变化较为平稳,当包负载从 200 Byte 增加到 1 000 Byte时,链路时延仅增加了 3.27 ms。因此,

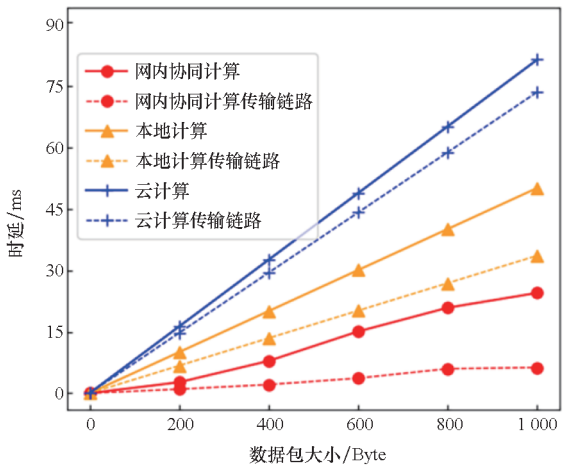


图 8　三种计算范式的时延比较

Fig. 8　Comparison of the latency of three computing paradigms

提出的网内协同计算方法有效降低了包转发过程中的传输时延,更加适应军事任务对低时延的苛刻需求。

6.2.2　任务分支的数据包重塑数目对时延的影响

本小节使用图 6 中任务模型 3 和图 7 中的网络拓扑结构Ⅱ,研究任务分支中最大包重塑数量与本文方法性能之间的关系。假设每个节点负责单一数据包,发生包重塑的总数为 3。以“{分支最大重塑数,{重塑任务编号}}”的形式设计 a、b、c、d、e、f、g、h 共 8 种方案,分别为{1,{2,3,4}}、{2,{1,2,3}}、{2,{1,3,4}}、{2,{2,3,5}}、{2,{3,4,5}}、{3,{1,2,5}}、{3,{1,3,5}}、{3,{1,4,5}},设置数据包大小范围在[200 Byte,1 000 Byte],单个数据包的重塑时间在[0.2 ms,1 ms]。测试结果如图 9 所示,箱线图记录了每个方案的最小值、中间值以及最大值。

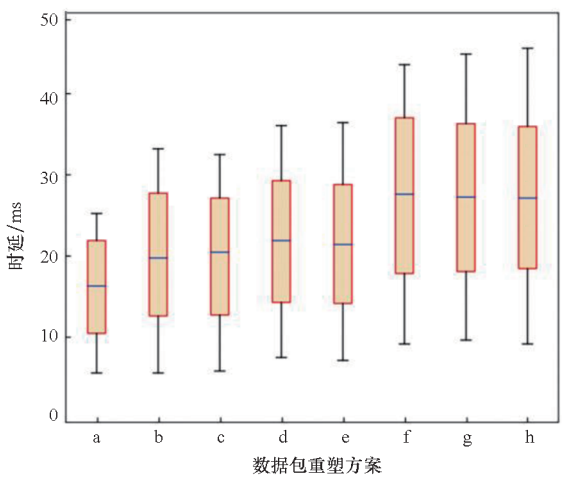


图 9　8 种重塑方案对时延的影响

Fig. 9　Impact of eight reshaping schemes on latency

通常情况下,包格式重塑会滞后任务的完成时间。本小节的目的是在包重塑数量已知的情况下,将此类数据包安排到合理的执行任务节点(具有多种功能的节点)以减少额外延迟。图 9 的结果显示,当 3 个格式重塑操作都发生在任务执行过程的同一条分支路径上,即分支最大重塑数为 3 时,任务时延的最小值、中间值、最大值均较大,如方案 f、g、h。相反地,分支上的最大重塑操作越少,时间越短,如方案 a 所需时间最短。主要原因是网内协同计算的任务时延实际上是分支路径的最大时间跨度,当包重塑操作在一条路径上累积时,最大时间跨度明显增加。考虑到这种情况,将数据包重塑任务合理地调度到不同的路径分支,可以有效地降低时延。

6.2.3　P4 网络的边连通度对时延的影响

本小节应用任务模型 2,进一步研究 4 种网络拓扑边连通度对本文网内协同计算时延的影响。

图 10 给出了网络拓扑的边连通度对时延的影响,其结果表明,网络拓扑的边连通度越高,则网内协同计算时延越低,如数据包大小为 1 000 Byte时,在拓扑Ⅰ下的时延比拓扑Ⅳ降低约 38%。其原因是边连通度会显著影响端到端的最小通信时延,导致 Dijkstra 算法得到的最短路径随边连通度的改变而发生变化。与拓扑Ⅱ、拓扑Ⅲ和拓扑Ⅳ相比,在拓扑Ⅰ中,起点到达终点的备选路线更多,因此应用拓扑Ⅰ得到的最短路径最小。从而得出结论,在边连通度高的 P4 网络环境中,网内协同计算的时延开销较低。

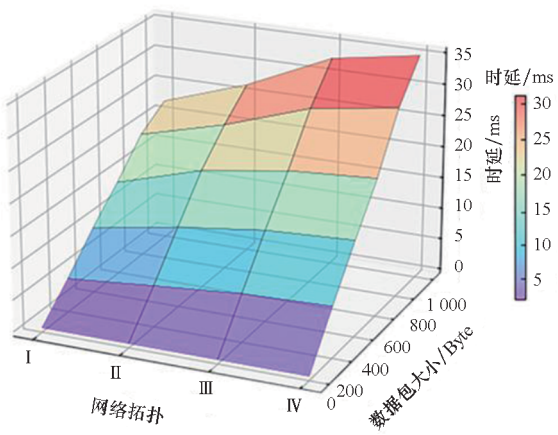
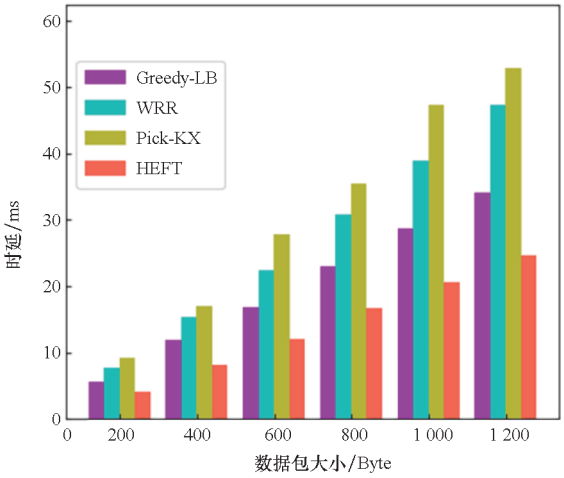


图 10　网络拓扑的边连通度对时延的影响

Fig. 10　Impact of edge connectivity of network topology on latency

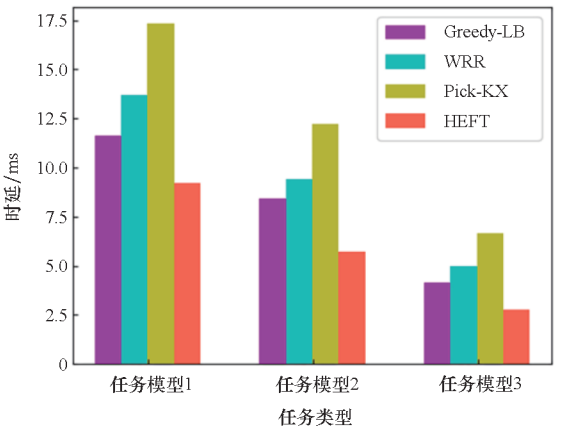
6.2.4　不同算法的时延比较

为了证明 HEFT 任务调度算法的适应性,从两方面进行了时延评估:一是处理的数据包大小,二是不同的任务类型。其他调度算法,如加权循环(weighted round robin, WRR)算法、贪婪负载均衡(greedy load balancing, Greedy-LB)算法、Pick-KX 负载均衡算法等被作为 HEFT 算法的比较对象。采用网络拓扑结构Ⅲ进行仿真,结果如图 11 所示。其中图 11(a)是采用任务模型 2 所得结果。



(a) 基于不同数据包大小的时延比较

(a) Latency comparison based on different packet sizes



(b) 基于不同任务的时延比较

(b) Latency comparison based on different tasks

图 11　不同算法的时延对比

Fig. 11　Comparison of latency of different algorithms

图 11(a)的结果表明,无论数据包多大,HEFT 算法的时延始终要低于其他 3 种调度算法。如包大小为 1 200 Byte 时,HEFT 算法的延迟性能比 Greedy-LB 算法、WRR 算法和 Pick-KX 算法分别改善 25.6%、47.9% 和 54.3%。同时,为了证明 HEFT 算法针对不同任务模型均适用,在图 11(b)中给出对应仿真结果,可以看出,4 种算法应用到 3 种任务模型(子任务数量相同)下的时延有一定差距,其中任务模型 3 的时延最低,任务模型 1 的时延最高。该结果由关键路径上的子任务数量所导致:与任务模型 1 和任务模型 2 相比,任务模型 3 在该路径下的子任务数量最少,因此所消耗的时间最短。此外,无论采用何种任务模型,HEFT 算法相对于其他算法均具有更低的延迟。主要原因是 Pick-KX 算法将子任务调度至节点时,未考虑设备的包转发能力;而 WRR 算法和 Greedy-LB 算法尽管在该问题上有所提升,但却忽略了设备网络链路的通信开销问题。与其相反,HEFT 算法以设备的计算和通信开销为优化指标,综合提高了资源的利用率,从而弥补了上述缺陷。

7　结论

在军事物联网中,包计算转发任务在往返服务器处理过程中存在着明显的缺陷,即无法有效解决从网络层到应用层的数据包传输时延开销问题。为此,本文提出了一种低时延的网内协同计算方法。文章前部分详细介绍了 P4 交换机及其实现包计算的过程;后半部分重点介绍了任务映射、时延优化建模,以及通过 HEFT 算法对本文方法进行优化。通过 Python 仿真测试的结果表明,与本地和云服务相比,该方法可保持在更短时间内完成任务。然后,进一步研究和讨论了如何合理地调度重塑数据包到指定执行节点、如何设置网络拓扑的边连通度等来提高该方法的效率。此外,也验证了所提算法比其他算法更显著地降低了任务时延。为进一步证明所提计算方法处理任务时的低时延特性,将在后续的相关工作中使用 P4 专用模拟器(BMv2)对其进行研究。

参考文献(References)

[1] 蒋永彬. 2G/3G 转网关键技术分析与物联网业务发展建议[J]. 通信世界, 2022(1): 34 – 37.
JIANG Y B. Analysis of key technologies for 2G/3G migration and suggestions for IoT business development [J]. Communications World, 2022(1): 34 – 37. (in Chinese)

[2] GAI R L, DU X Y, MA S Y, et al. A summary of 5G applications and prospects of 5G in the Internet of Things[C]// Proceedings of the IEEE 2nd International

Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), 2021: 858 – 863.

[3] BHOMBLE B, KATKAR S, DHERE D, et al. IoT based smart sniper[C]//Proceedings of the International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), 2017: 557 – 561.

[4] REYES CH R P, VACA H P, CALDERÓN M P, et al. MilNova: an approach to the IoT solution based on model-driven engineering for the military health monitoring[C]//Proceedings of the CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 2017: 1 – 5.

[5] 任智源, 肖尧, 郭凯, 等. 适应低时延业务需求的分布式可穿戴单兵作战信息系统[J]. 国防科技大学学报, 2018, 40(4): 159 – 165.
REN Z Y, XIAO Y, GUO K, et al. Distributed wearable individual soldier combat information system for low-latency business[J]. Journal of National University of Defense Technology, 2018, 40(4): 159 – 165. (in Chinese)

[6] CHEN Z G, ZHAN Z H, LIN Y, et al. Multiobjective cloud workflow scheduling: a multiple populations ant colony system approach[J]. IEEE Transactions on Cybernetics, 2019, 49(8): 2912 – 2926.

[7] 李瑞彪, 任继军, 任智源. 面向车联网低时延需求的分布式路径计算方案[J]. 西安交通大学学报, 2022, 56(10): 91 – 100.
LI R B, REN J J, REN Z Y. Distributed path computing scheme for low-latency demand of Internet of vehicles[J]. Journal of Xi'an Jiaotong University, 2022, 56(10): 91 – 100. (in Chinese)

[8] AHMED M, MUMTAZ R, ZAIDI S M H, et al. Distributed fog computing for Internet of Things (IoT) based ambient data processing and analysis[J]. Electronics, 2020, 9(11): 1756.

[9] 张夏童, 任智源, 胡锦涛, 等. 面向医疗大数据任务低时延需求的路径计算方案[J]. 西安交通大学学报, 2020, 54(2): 119 – 126.
ZHANG X T, REN Z Y, HU J T, et al. A path computing scheme for low-latency requirement of medical big data task[J]. Journal of Xi'an Jiaotong University, 2020, 54(2): 119 – 126. (in Chinese)

[10] WU H Z, SHEN Y B, XIAO X, et al. Accelerating industrial IoT acoustic data separation with in-network computing[J]. IEEE Internet of Things Journal, 2023, 10(5): 3901 – 3916.

[11] 何秀丽, 任智源, 史晨华, 等. 面向医疗大数据的云雾网络及其分布式计算方案[J]. 西安交通大学学报, 2016, 50(10): 71 – 77.
HE X L, REN Z Y, SHI C H, et al. A cloud and fog network architecture for medical big data and its distributed computing scheme[J]. Journal of Xi'an Jiaotong University, 2016, 50(10): 71 – 77. (in Chinese)

[12] XING J R, HSU K F, KADOSH M, et al. Runtime programmable switches[C]//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation, 2022: 651 – 665.

[13] 刘莹, 曹畅, 杨建军, 等. P4技术的边缘计算相关应用研究[J]. 信息通信技术, 2020, 14(4): 45 – 50.
LIU Y, CAO C, YANG J J, et al. Research on the application of P4 technology in edge computing[J]. Information and Communications Technologies, 2020, 14(4): 45 – 50. (in Chinese)

[14] WANG S Y, WU C M, LIN Y B, et al. High-speed data-plane packet aggregation and disaggregation by P4 switches[J]. Journal of Network and Computer Applications, 2019, 142: 98 – 110.

[15] ZHANG C, BI J, ZHOU Y, et al. HyperV: a high performance hypervisor for virtualization of the programmable data plane[C]//Proceedings of the 26th International Conference on Computer Communication and Networks (ICCCN), 2017: 1 – 9.

[16] CASTANHEIRA L, PARIZOTTO R, SCHAEFFER-FILHO A E. FlowStalker: comprehensive traffic flow monitoring on the data plane using P4[C]//Proceedings of 2019 IEEE International Conference on Communications (ICC), 2019: 1 – 6.

[17] WU Y J, HWANG W S, SHEN C Y, et al. Network slicing for mMTC and URLLC using software-defined networking with P4 switches[J]. Electronics, 2022, 11(14): 2111.

[18] PAOLUCCI F, SCANO D, CASTOLDI P, et al. Latency control in service chaining using P4-based data plane programmability[J]. Computer Networks, 2022, 216: 109227.

[19] CUI M W, LI X, WANG Y, et al. SPT: sketch-based polling in-band network telemetry[C]//Proceedings of 2022 IEEE/IFIP Network Operations and Management Symposium, 2022: 1 – 7.

[20] 杨爱玲, 邹乾友, 付松涛. P4交换机在天地一体化网络中的应用[J]. 信息工程大学学报, 2020, 21(4): 453 – 458.
YANG A L, ZOU Q Y, FU S T. Implementation with P4 switch in space and terrestrial integrated network[J]. Journal of Information Engineering University, 2020, 21(4): 453 – 458. (in Chinese)

[21] SAPIO A, ABDELAZIZ I, CANINI M, et al. DAIET: a system for data aggregation inside the network[C]//Proceedings of the 2017 Symposium on Cloud Computing, 2017: 626.

[22] EDWARDS T G, CIARLEGLIO N. Timestamp-aware RTP video switching using programmable data plan[J]. Proceedings of the ACM SIGCOMM Industrial Demo, 2017: 1 – 2.

[23] JOSE L, YAN L, VARGHESE G, et al. Compiling packet programs to reconfigurable switches[C]//Proceedings of the 12th USENIX Symposium on Networked Systems Design and

Implementation, 2015: 103 – 115.

[24] ZHANG H L, WU Y H, SUN Z X. EHEFT-R: multi-objective task scheduling scheme in cloud computing[J]. Complex & Intelligent Systems, 2022, 8(6): 4475 – 4482.

[25] HASSAN M A, XIAO M B, WEI Q, et al. Help your mobile applications with fog computing[C]//Proceedings of the 12th Annual IEEE International Conference on Sensing, Communication, and Networking-Workshops (SECON Workshops), 2015: 1 – 6.

[26] MUÑOZ O, PASCUAL-ISERTE A, VIDAL J. Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading [J]. IEEE Transactions on Vehicular Technology, 2015, 64(10): 4738 – 4755.