

# 基于节点分层和延迟敏感的服务放置策略

鲍春林<sup>1</sup>, 宋丽华<sup>2</sup>, 余航<sup>1</sup>

(1. 陆军工程大学研究生院, 江苏南京 210000;

2. 陆军工程大学指挥控制工程学院, 江苏南京 210000)

**摘要:**在灾难应急响应时,为了提高救援效率,使用无人机作为灾区的临时通信基站提供中继服务,由无人机以及边缘设备组成雾节点网络共同提供服务。但是由于雾节点设备的计算存储资源和传输带宽有限,无法部署大型应用程序。为了避免单一雾节点负载过高,将原应用拆分为若干微服务并且分散部署到雾节点上。然而服务数量的激增导致了服务请求时间过长。针对该问题,提出了基于节点分层和延迟敏感的服务放置算法。将雾节点划分为三层,约束微服务的部署位置。对每个雾节点的等待时间建模,尽最大努力使每个雾节点的服务等待延迟最小。在仿真实验中,通过与 Edge-ward 策略进行对比,表明该方法能够在资源受限的情况下有效减少服务等待时间。

**关键词:**灾难响应;雾计算;微服务;服务部署;无人机

中图分类号:TP39

文献标识码:A

文章编号:1673-629X(2022)10-0014-07

doi:10.3969/j.issn.1673-629X.2022.10.003

## Service Placement Policies Based on Node Tiering and Latency Sensitivity

BAO Chun-lin<sup>1</sup>, SONG Li-hua<sup>2</sup>, YU Hang<sup>1</sup>

(1. School of Graduate, Army Engineering University of PLA, Nanjing 210000, China;

2. School of Command Control and Engineering, Army Engineering University of PLA, Nanjing 210000, China)

**Abstract:** In disaster-response operations, UAV are used as temporary communication base stations in disaster areas to provide relay services to improve the efficiency for rescue requests. Fog node network is formed by UAV and edge-device to offer services. Meanwhile, due to the limited computational storage resources and transmission bandwidth of the fog node, it is impossible to deploy huge applications. The original application is split into several microservices and deployed to the fog nodes in a decentralized manner to avoid excessive load on a single fog node. However, the proliferation of the number of services leads to more service request. To address this problem, a node-based hierarchical and latency-sensitive service placement algorithm is proposed. These fog nodes are divided into three layers to constrain the deployment location of microservices. The request time of each fog node is modeled, and the best effort is made to minimize the service waiting delay of each fog node. In simulation experiments, by comparing with the Edge-ward strategy, it is shown that the proposed method can effectively reduce the service waiting time under the resource constraint.

**Key words:** disaster response; fog computing; microservice; service deployment; UAV

## 0 引言

许多毁灭性灾难,例如地震、洪水、暴雨和海啸等,可能对基础设施造成巨大破坏并对群众生命和财产造成威胁。由于灾难所导致的灾区基础实施损坏,在受灾地区组织有效的灾难响应是一项具有挑战性的任务。通信或计算基础设施的损坏,使救援团队无法及时感知、共享以及处理收集的数据。在传统的应急响应中,大多使用对讲机、电台以及卫星来建立紧急传输服务。然而这些解决方案存在着带宽不足以及高延迟

的挑战,救援人员能够获取和分享的数据形式也存在局限性。

近年来无人机技术的兴起为解决上述问题提供一种新的方案,在雾计算或边缘计算的范式下,无人机可以通过作为雾节点提供中继服务以及运行应用程序<sup>[1]</sup>。通过将多个无人机部署在灾区中来组织成飞行自组网(FANET),使其互相之间协同工作,共同提供计算和通信服务。

为了在灾难响应中提供有效帮助,应用程序中越

收稿日期:2021-08-06

修回日期:2021-12-08

基金项目:国家自然科学基金资助项目(62172432)

作者简介:鲍春林(1996-),男,硕士,通讯作者,研究方向为微服务、韧性;宋丽华,博士,教授,CCF会员(H1448M),研究方向为形式化验证。

来越多地加入机器学习算法以实现音视频识别、伤害诊断、路线规划等功能。它们通常要求更多的资源开销,超出地面传感器等雾节点的供给能力。现有的工作大多集中在如何提供安全可靠的传输服务上,对于这些拥有较高计算需求的任务,如何将它们部署在网络中的研究较少。

由于一些应用程序对计算资源需求较高,为了克服雾节点资源不足的问题,可以将其拆分为独立的、松耦合和轻量的微服务<sup>[2]</sup>,分散部署在雾节点上。在接收到服务请求时,临时网络会根据请求的类型选择不同的微服务来组成完整的应用程序以对请求做出响应。以伤员诊断为例,由图 1 所示,无人机节点  $U_1$  通过生命探测器发现一名伤员(步骤 1)并且位置信息被无人机节点  $U_2$  接收(步骤 2),  $U_2$  通知距离最近的救援小组前往营救(步骤 3)同时保持在原位置等待其他任务,救援路线由  $U_1$  和  $U_2$  共同计算(步骤 4)。救援小组到达现场后,通过便携式设备收集测量伤员的医疗数据(步骤 5)并进行分析,形成初步的医疗诊断(步骤 6)。为了得到更专业的治疗,救援小组将伤员的相关情况上传到  $U_1$ (步骤 7)。  $U_1$  通过  $U_2$  以及  $U_3$  连接到最近的医疗中心(步骤 8),在专业的反馈治疗后可以帮助救援小组进一步治疗伤员(步骤 9)。在得到更专业的救助后,由  $U_1$ 、 $U_2$  和  $U_3$  共同计算出伤员至医疗中心最短无障碍道路,方便救援中心派出救护车接收伤员(步骤 10)。

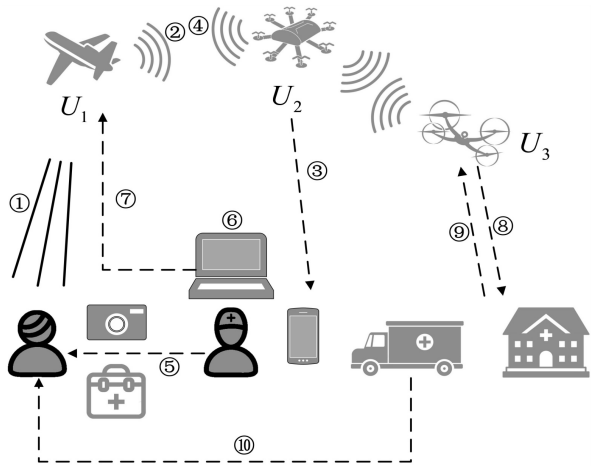


图 1 灾区伤员救助案例

上述救援过程的每个步骤都可以视为一个独立的微服务。当收到请求时,一个完整的响应在经过若干步骤后完成。在这些可用服务中,路线规划、医疗诊断以及人脸识别等都需要通过机器学习算法实现,对计算资源有一定的需求。并且考虑到临时网络的带宽有限,不合理的放置策略可能会导致过高的网络时延,耽误宝贵的救援时间。

从平衡资源和带宽需求出发,该文主要提出了一

种基于雾节点分层和延迟敏感的服务放置算法,以减少带宽不足导致的应用程序的延迟,并且物理设备能够满足应用程序的资源需求。

## 1 背景和相关工作

雾计算常被用于解决 IOT 网络中云数据中心无法提供高质量服务的问题。通过将部分服务从云数据中心迁移到边缘设备中,缩短用户和服务提供者的距离,减少用户和云之间长距离传输导致的延迟,对云计算起辅助作用。在雾计算系统中,任何具有计算电源、存储和网络功能的资源都可以被称为雾节点,如智能手机、个人计算机、笔记本电脑<sup>[1,3]</sup>。在网络中雾节点处于用户和云中心之间,通过在网络边缘放置服务以减少通信延迟并更快地执行服务。常见的雾计算架构大体被分为三层,最下层为边缘网络,主要由用户终端设备和传感器组成。中间为雾层,主要由一些具有一定计算能力的设备(例如服务器、网关等)组成。最上层为云数据中心,主要负责分析和存储大量数据。

结合边缘计算和无人机,已经有很多相关方面的工作。文献[4]呈现了最佳的部分卸载方案,称为 POSMU(部分卸载策略最大化用户任务编号),以获得每个用户端设备的最佳卸载比、本地计算频率、传输功率和边缘服务器计算频率。文献[5]旨在通过联合优化无人机轨迹、卸载任务比例和用户调度变量,在离散二进约束、能源消耗约束和无人机轨迹约束下,使每个时隙中所有用户的最大延迟之和最小。

整体式架构转换至微服务架构可能面临的挑战:

对于常见边缘系统的服务放置问题,文献[6]研究了在多个 Cloudlets 上的服务放置联合优化,并为最终用户的请求进行负载调度。文献[7]基于延迟约束下的贪婪和阈值策略,提出了 SAE 和 CEC 两相算法。文献[8]提出了一种基于上下文感知的放置策略,该策略将物联网设备层面的情境与雾节点的能力相协调,以尽量减少各种 IOT 应用的服务交付时间。文献[9]提出了基于容器的任务调度算法,用于基于雾的高并发的延迟敏感的智能组合。文献[10]提出了一种改良的遗传算法,将三个由多个微服务组成的石油工厂维护服务放置到边缘云基础设施中,以最大限度地减少资源消耗和整体服务延迟。笔者的工作更多的是关注服务之间的依赖关系对延迟的影响。

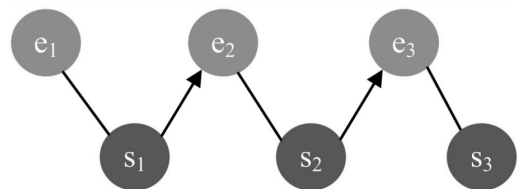


图 2 服务依赖链定义

在微服务架构下,每个微服务都会提供 API 接口,用于调用其他微服务。如图 2 所示,每个微服务  $s$  都会提供一个或多个服务功能  $e$ ,实线表示微服务能够提供的服务功能。箭头表示调用关系,微服务  $s_1$  需要调用微服务  $s_2$  的服务功能  $e_2$ 。微服务的调用使得它们之间具有依赖关系,多个微服务之间的服务调用序列定义为服务依赖链<sup>[11]</sup>。

以引言中伤员救助为例,整个过程划分为多个步骤,每个步骤都可以视为一个独立的微服务,这些微服务以一种依赖关系组成。除了起始的微服务,其他微服务都需要上一个微服务的计算结果。在得到可达路线之前系统内部需要进行 4 次服务请求。因此在向外提供服务时,虽然与原本完整的服务相比,系统和外部的服务请求者之间的通信量并不会有明显的差距,但系统内部的各个微服务之间将会进行频繁的通信请求,从而导致通信量激增。特别是由于灾区组建的临时网络带宽有限,即使通过无人机进行中继传输仍然可能会导致较高的延迟,对于一些延迟敏感的应用程序,保证这些任务能够在限制时间内完成至关重要。在一些紧急情况下,延迟过高可能会导致严重的后果。通过调整服务的放置策略,能够减少部分通信流量。

## 2 模 型

### 2.1 雾节点网络架构

与常见的 IOT 网络不同,在灾区中由于基础设施的损坏,几乎无法依靠云数据中心提供充分的资源<sup>[12-13]</sup>。对于灾区的雾节点网络,在传统的雾计算架构的基础上删除了云数据中心。并且以引言中救助场景为例,结合灾区网络的特点对雾节点架构进行调整,以更精确地描述雾节点设备的延迟。

根据上文的叙述,由于雾节点的计算资源有限,无法部署过多的应用程序。针对这种情况,将原应用程序拆分为若干个微服务,并且这些微服务可以部署在不同的雾节点上。在这种架构下,每个完整的应用程序不再部署在单一的设备上而是由多个设备共同承载。每个应用程序由若干微服务以一种依赖关系组成,由于原本的应用程序中存在功能相似的模块,在拆分为微服务后,某些微服务可能属于不止一条服务依赖链,最终形成一个更加庞大的服务关系网。目标是将依赖关系网部署到一组雾节点上,在满足所有资源需求的前提下,尽可能地减少服务延迟<sup>[14-15]</sup>。

由于灾难区域网络中物理设备的资源容量和带宽有限,作为一般原则,将对资源需求低的微服务尽可能部署在资源相对较少的传感器上,一些计算任务繁重的微服务部署在资源相对丰富的智能终端或无人机上。由于服务之间的依赖关系,服务间的业务路由和

服务分配是一个联合问题。需要尽可能地将不同依赖链产生的流分配到不同的路径上以避免同一路径承载过大流量<sup>[16]</sup>。

根据计算能力和网络中的位置,将灾区临时网络中的雾节点分为三层(见图 3)。

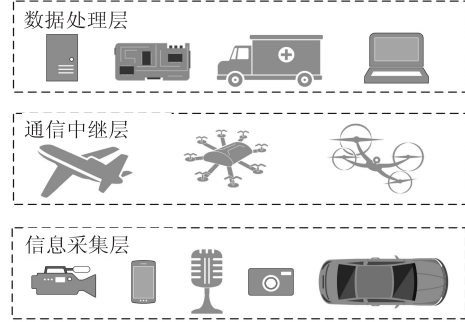


图 3 灾区雾节点架构

(1) 信息采集层。在这层中的雾节点多为传感器设备(如相机、具有拍照功能的手机、生命探测器等),主要负责采集外界的环境信息。这些设备普遍拥有较少的资源,每个节点的功能也较为单一。

(2) 通信中继层。将网络中所有的无人机分配到中继层,主要承担两种任务:(a) 为灾区网络提供可靠传输服务;(b) 承载一部分微服务。处于服务链中部的微服务更适合于部署在无人机上,这类微服务的主要任务多为整合并筛选有用信息。它们通常具有更好的可替代性。

(3) 数据处理层。在这一层中的雾节点主要是一些计算终端或存储设备,拥有更多的计算资源(如个人笔记本、救援车辆、微型服务器等),然而更多的计算任务也导致了更长的程序执行时间。

### 2.2 模型架构

将灾难响应网络中的所有设备及其通信链路视为有向图  $G(V, E)$ , 其中  $V$  是所有设备节点的集合,  $E$  则是所有链路的集合。

$$\delta: E \rightarrow (V \times V)$$

$$\delta_e = (p, q)$$

$$p, q \in V$$

(1)

对于每个设备  $v$ , 总共有  $k$  种类型的资源  $R_v = (r_v^1, r_v^2, \dots, r_v^k)$  (例如 CPU、RAM 等), 其中  $r_v^k$  是设备  $v$  的第  $k$  种资源量。

所有微服务组合成的完整的服务集合记为  $S = \{S_1, S_2, \dots, S_I\}$ , 总共有  $I$  个服务, 由于微服务的可替代性, 这些完整的服务中有一些具有相同的功能。每个服务  $S_i = (N_i, Z_i)$  是由若干个微服务所组成的依赖链,  $N_i$  表示该服务所包含的微服务的集合,  $Z_i$  表示服务中依赖项的集合。将系统中的所有微服务表示为:

$$N = \bigcup_{i=1}^I N_i \quad (2)$$



对于每个微服务  $n \in N$  所需要的资源表示为  $d_n = \{d_n^1, d_n^2, \dots, d_n^k\}$ 。

应该保证每个微服务都至少被部署在一个节点上,并且一些特定的服务只能运行在特定的节点上,应该满足以下约束:

$$\sum_{v \in V} x_{vn} y_{vn} \geq 1, \forall n \in N \quad (3)$$

其中,  $y_{vn}$  是一个二进制变量,用于确定微服务  $n$  是否部署在  $v$  上。 $x_{vn}$  是一个二进制变量,用于指定节点  $v$  是否允许承载微服务  $n$ 。

资源约束。每个节点上的资源消耗量不应超过当前节点的资源量:

$$\sum_{n \in N} x_{vn} y_{vn} d_n^k \leq r_v^k, \forall v \in V \quad (4)$$

在计算延迟时,对每个雾节点的服务请求等待时间进行建模,考虑三种情况:

①当  $v$  没有上行链路时,此时节点  $v$  的延迟  $l_v$  就是当前节点所部署的服务执行时间:

$$l_v = c_v \quad (5)$$

$c_v$  是节点  $v$  处理数据的时间。为了便于计算服务等待时间,不考虑排队机制,单个雾节点上服务的等待时间定义为所有服务运行时间线性相加。

②当  $v$  只有一条上行链路时,此时节点  $v$  的延迟  $l_v$  定义为:

$$l_v = l_u + b_{uv} + c_v \quad (6)$$

其中,  $u$  是  $v$  的父节点,  $l_u$  是  $u$  的总延迟,  $b_{uv}$  是父节点与子节点之间链路的传输时延。

③当  $v$  有多条上行链路时,此时节点  $v$  的延迟  $l_v$  定义为:

$$l_v = \max_{u \in U_v} \{l_u + b_{uv}\} + c_v \quad (7)$$

其中,  $U_v$  是  $v$  的所有父节点的集合。

优化的目标是在满足资源约束(3)(4)的条件下,使整体网络的总延迟最小,即所有雾节点的延迟总和最小。

$$\min \sum_{v \in V} l_v \quad (8)$$

考虑传感器与无人机进行通信时,由于消息的长度相比于带宽要小很多,链路中的延迟变化少,可以视为链路固有属性。无人机之间的通信,消息类型比较类似,一些关键节点的数据量可能过多,导致延迟稍长,但整体上可视为固有属性。无人机与边缘设备的通信,虽然带宽有所提高,但由于任务数据较大导致延迟相比于前两种更高,具体延迟主要取决于任务量大小,以各设备所能承受任务的阈值,确定各设备和无人机之间的链路的固有延迟。

综上,将每条链路的传输延迟  $b_{pq}$  作为链路的固有属性并且是已知的。

### 3 算法设计

根据第 2 节中的约束条件,设计了启发式服务放置算法。依据上一节中的模型,将服务依赖网络和雾节点网络简化为图 4。其中服务依赖网络内的每个节点代表一个微服务,雾节点网络中深色节点代表雾节点中的数据处理层设备和信息采集层设备,浅色节点代表无人机节点。在服务依赖网络中,调用关系是单向的并且不存在循环调用。在雾节点网络中,信息的传递是双向的。微服务数量远多于雾节点数量,每个雾节点上可能部署多个微服务,但不应超过该节点的资源上限。对于每个微服务,可选择部署的雾节点的范围有限。

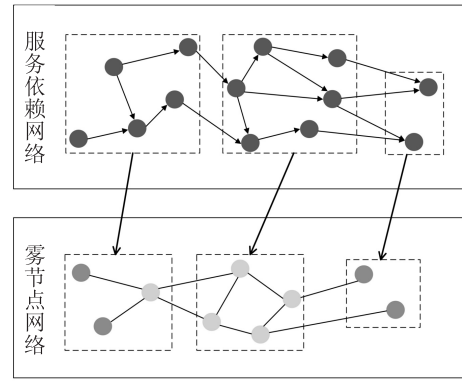


图 4 服务覆盖拓扑

为了减少雾节点设备总体的延迟,根据雾节点的资源 and 微服务之间的依赖关系,算法的思想为:

将微服务分散部署到整个网络中,避免服务集中导致个别链路拥塞过于严重。

由于灾区临时网络设备资源不足,网络的扩展难度大,因此对资源限制的考虑应该高于对网络延迟性能的考虑。

尽可能地将服务依赖链覆盖到结构相近的物理网络路径上,使存在调用关系的两个微服务能够部署在相邻的雾节点上,以此减少无用的转发次数。

优先部署在依赖关系前端的微服务。

对于  $c$ , 由于个别微服务在功能上具有一定的通用性以及可替代性,可能会同时属于多条依赖链。当一条链路更契合物理链路时,与其存在相交微服务的其他依赖链在进行分配部署时必然会受到影响(存在较多的转发次数),因此这里追求的是所有雾节点的整体延迟。

Algorithm 1:

输入:

$V$ : 设备拓扑图

$N$ : 服务链拓扑图

$l_c$ : 每个微服务在拥有充分资源条件下的运行时间集合

$l_w$ : 每个微服务在不同雾节点上独立运行的时间集合,并且满足(3)的条件

$B_{pw}$ : 每条链路的传输延迟集合

输出: 部署策略

- 1 依据  $l_c$  降序对  $N$  进行 BFS 排序
- 2 for each  $n \in N$ :
- 3 if  $n$  没有部署
- 4  $l \rightarrow \infty$
- 5  $V_{able} \leftarrow$  在  $l_{nw}$  中筛选出可部署服务的雾节点
- 6 for each  $v \in V$  in  $V_{able}$ :
- 7  $v$  资源充足的条件下
- 8 if 设备没有父节点
- 9  $l \leftarrow$  部署  $n$  后所有该节点的服务运行时间和
- 10 elif 设备只有一个父节点
- 11  $l$  根据情况②计算
- 12 else
- 13  $l$  根据情况③计算
- 14 选择  $l$  最少的  $v$  部署  $n$
- 15 对  $v$  的可用资源更新

算法的整体思想是优先部署处在服务链前端的服务,因此采用广度优先搜索根据每个微服务在拥有充分资源下的运行时间以降序对微服务进行排序。为满足约束(3),在算法第 5 步,根据每个微服务的功能,对允许部署的设备进行筛选。为满足约束(4),在部署微服务前,先检查所部署的设备是否拥有足够的资源。在计算每个雾节点的服务运行时间时,其所部属

的微服务之间不存在优先级,假设以最坏的情况考虑即所有的微服务同时在运行中,将服务运行时间以线性相加的方式计算。假设当请求到达时,该雾节点的等待延迟为其所部署的所有微服务的运行时间总和。算法默认物理设备拥有足够多的资源用于部署服务,不存在服务未被部署的情况。

## 4 实验结果与分析

在本节中,对所提算法和 Edge-ward<sup>[17]</sup>策略分别进行了仿真。仿真结果表明,所提算法平均的服务延迟相比于 Edge-ward 策略降低了 62.8%。

Edge-ward 策略是 iFogSim 模拟器中默认的部署策略之一。iFogSim 是一个开源的用于对物联网、雾和边缘计算环境建模的模拟器。在 Edge-ward 策略中,服务放置在边缘附近,不考虑应用和任务的优先级,无人机仅提供中继服务。

### 4.1 仿真配置

在仿真实验中,分别建立了雾节点网络和微服务网络的仿真拓扑。在雾节点的构建中,总共加入了 16 个雾节点,根据一些以往文献<sup>[17-18]</sup>的数据对设备的资源配置进行了分配。具体情况如表 1 所示。

表 1 雾节点设备的参数配置

设备层级	数量	CPU/MIPS	RAM/MB	相邻层级之间的 传输延迟/ms
采集层	4	500	256	36
中继层	8	4 000	2 048	124(层内延迟)
处理层	4	10 000	8 192	829

对需要被部署的微服务主要分为 3 类,如表 2 所示。其中 A 类微服务主要提供信息采集的功能,对资源需求不高,一般作为一条服务链的起始服务。这类微服务多部署在传感器设备上。B 类微服务多为服务链中部的微服务,主要执行资源整合为汇总、对数据进

行预先的处理和筛选等任务。C 类微服务多为计算密集型微服务,对物理资源的需求相比于 B 类更高。B 类和 C 类微服务更偏向于部署在无人机和边缘设备上。

表 2 微服务的参数配置

微服务类型	数量	CPU 需求/MIPS	RAM 需求/MB	拥有充分资源时的 运行时间/ms
A	9	150	32	1
B	15	1 507	128	20
C	7	2 006	256	53

在实际部署时,相同类型的微服务的资源需求不可能完全相同。为了得到更好的仿真效果,实际对微服务进行参数配置时,在表 2 给出的数据基础上添加适当的浮动区间。不同雾节点之间的延迟参数配置同理。

### 4.2 仿真结果

为了比较算法的性能,验证降低延迟的有效性,对整个服务网络的每个起始服务,使用深度优先搜索的方式,分别选出了其所有的不相交的服务链。根据第 2 节中所提出的模型以及两种算法所得出的部署方

案,分别计算这些服务链的总延迟。所计算出的链路延迟相比于实际网络运行时显著增加。原因是假设的模型以最坏的情况考虑,不考虑排队的问题,所有服务都同时运行,整个网络近似满负荷运转,网络的利用率非常高,因此网络环境非常糟糕。当所实施的服务部署方案非常糟糕时,服务链的延迟会非常大。在仿真结果中个别服务的延迟已经超过了 10 秒。

仿真结果如图 5 所示,横轴代表了所选出的每一条服务链,纵轴代表各服务链计算出的延迟总和。其中,三角形为实施算法 1 得出的部署策略的结果,圆形为根据 Edge-ward 策略所得到的结果。从结果中可以看出,算法 1 要明显优于 Edge-ward。根据算法 1 计算出的部署策略,服务链的平均延迟为 3 259.31 ms,

而根据 Edge-ward 得出的策略,服务链的平均延迟为 8 761.45 ms,是算法 1 的 2.67 倍。在图 6 中,对每个雾节点的延迟根据第 2 节中的计算方法进行对比,横坐标为每个雾节点,其中 0-3 号为采集层节点,4-11 号为中继层节点,12-15 号为处理层节点。纵坐标为节点延迟。从结果中可以看出,采集层节点的延迟两种算法几乎没有差距;中继层节点中的延迟 Edge-ward 的平均延迟为算法 1 的平均延迟的两倍;处理层的节点中,由于 Edge-ward 不计算节点的服务处理时间而只计算资源是否满足,导致个别节点部署过多服务。在仿真结果中表现为个别节点的延迟在 Edge-ward 策略下要明显高于算法 1。

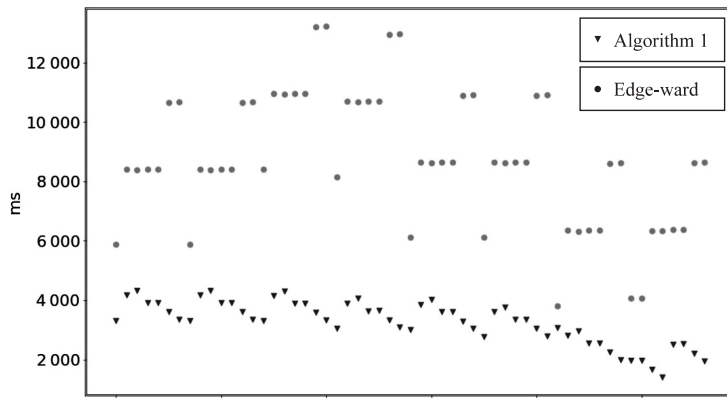


图 5 算法 1 与 Edge-ward 服务依赖链延迟结果对比

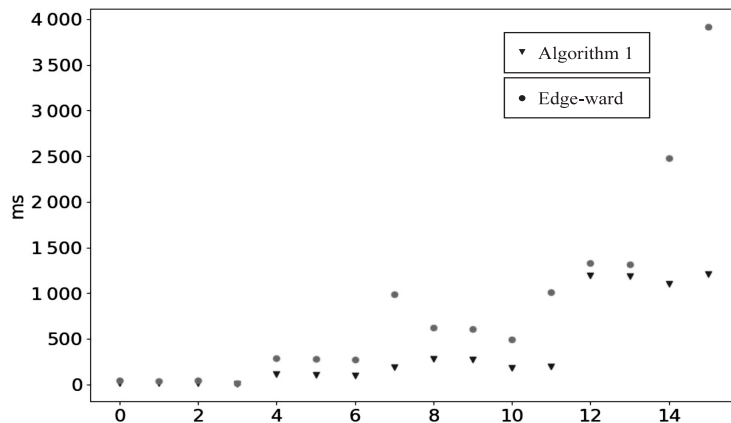


图 6 算法 1 与 Edge-ward 雾节点延迟结果对比

### 4.3 结果分析

算法 1 令服务链网络的其中一部分服务部署在无人机上,一定程度上减少了服务链中相邻两个服务在交互时,真实网络中的跳转次数,进而避免了链路中的等待延迟。使用 Edge-ward 部署服务时,微服务都被部署在边缘设备上,流在经过一条服务链时,需要在网络中进行若干次跳转,导致较多的链路等待延迟。并且由于边缘设备和无人机之间的链路延迟明显高于无人机节点间的延迟,服务链中除了起始服务外,每一对

相邻的两个微服务在交互时,至少经过两次边缘设备的链路。而在算法 1 中,服务链仅在尾部的少数微服务在交互式时经过边缘设备链路。前者的等待延迟在累积后会成倍高于后者。

## 5 结束语

提出了一种基于节点分层和延迟敏感的服务放置算法,用于支持高效的灾难响应操作。在灾难响应中,服务延迟对开展救援的效率起到至关重要的作用。必

须保证服务能够在规定时间内对请求者做出响应。在所提出的算法中,通过将原有的整体式服务拆分微服务,并将服务的部署位置从边缘设备向雾环境网络的中间部分转移。为了有效缩短延迟,建立了不同雾节点设备的延迟模型,根据微服务之间的依赖关系,根据每个设备的延迟变化对服务进行了重新部署并进行了仿真实验和结果评估。将算法与 Edge-ward 策略进行比较,仿真结果表明提出的算法能够有效缩短延迟,平均时延减少了 62.8%。

未来打算对该算法进行扩展,并考虑网络中设备的移动性。在后续工作中,假设所有设备都不具有移动性即整个网络的拓扑结构不会发生变化。然而在现实世界中,雾节点的移动性的概率非常高,需要对部署策略进行适当的调整以适应网络拓扑结构的变化。此外,还打算在之后的工作中考虑灾区网络的韧性。由于雾节点设备的资源有限,少数设备几乎不可避免地会出现资源耗尽的情况,特别是在一些关键的链路上。此外受灾区的影响,所搭建的灾区临时网络可能会遭到破坏,个别雾节点因此失去响应。由于微服务具有可替代性,当一条服务链中的某些服务无法响应时,能够在网络中寻找其他微服务替代无法响应的微服务从而使服务不中断,因此考虑在雾环境中通过增加冗余路径以增强网络的韧性。

#### 参考文献:

- [1] WEI X, LI L, TANG C, et al. UAV fog-assisted data-driven disaster response: architecture, use case, and challenges [C]//The Netherlands: web information systems engineering-WISE 2020. [s.l.]: Springer International Publishing, 2020: 591-606.
- [2] 辛园园, 钮俊, 谢志军, 等. 微服务体系结构实现框架综述[J]. 计算机工程与应用, 2018, 54(19): 10-17.
- [3] 段雪城, 范平清. 基于边缘计算的物联网网关监控系统的研究[J]. 电子器件, 2019, 42(6): 1569-1573.
- [4] TANG Q, CHANG L, YANG K, et al. Task number maximization offloading strategy seamlessly adapted to UAV scenario[J]. Computer Communications, 2020, 151: 19-30.
- [5] HU Q, CAI Y, YU G, et al. Joint offloading and trajectory design for UAV-enabled mobile edge computing systems[J]. IEEE Internet of Things Journal, 2018, 6(2): 1879-1892.
- [6] ERDELJ M, NATALIZIO E, CHOWDHURY K R, et al. Help from the sky: leveraging UAVs for disaster management[J]. IEEE Pervasive Computing, 2017, 16(1): 24-32.
- [7] HUA M, WANG Y, LI C, et al. UAV-aided mobile edge computing systems with one by one access scheme[J]. IEEE Transactions on Green Communications and Networking, 2019, 3(3): 664-678.
- [8] ZHANG L, ZHAO Z, WU Q, et al. Energy-aware dynamic resource allocation in UAV assisted mobile edge computing over social Internet of vehicles[J]. IEEE Access, 2018, 6: 56700-56715.
- [9] YIN L, LUO J, LUO H. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing[J]. IEEE Transactions on Industrial Informatics, 2018, 14(10): 4712-4721.
- [10] RAHMAN A, JIN J, CRICENTI A L, et al. Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance[J]. IEEE Transactions on Industrial Informatics, 2018, 15(5): 2500-2511.
- [11] MA S P, FAN C Y, CHUANG Y, et al. Using service dependency graph to analyze and test microservices[C]//2018 IEEE 42nd annual computer software and applications conference (COMPSAC). Tokyo: IEEE, 2018: 81-86.
- [12] 易发胜. 基于服务的网络端系统 QoS 的研究[D]. 成都: 电子科技大学, 2008.
- [13] 马希琳. 基于 Kubernetes 容器集群资源调度策略研究[D]. 西安: 西安科技大学, 2019.
- [14] 郝庭毅, 吴恒, 吴国全, 等. 面向微服务架构的容器级弹性资源供给方法[J]. 计算机研究与发展, 2017, 54(3): 597-608.
- [15] 冯显力, 韦化, 韦洪波, 等. 含微服务的调度自动化系统分布式实时数据库[J]. 电力系统保护与控制, 2018, 46(21): 138-144.
- [16] 万书鹏, 易强, 张凯, 等. 基于微服务架构的新一代调控系统服务编排技术[J]. 电力系统自动化, 2019, 43(22): 116-121.
- [17] GUPTA H, VAHID DASTJERDI A, GHOSH S K, et al. iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of things, edge and fog computing environments[J]. Software: Practice and Experience, 2017, 47(9): 1275-1296.
- [18] KHOSROABADI F, FOTOUHI-GHAZVINI F, FOTOUHI H. SCATTER: service placement in real-time fog-assisted IoT networks[J]. Journal of Sensor and Actuator Networks, 2021, 10(2): 26.