分 类 号_____ 学号 D201177565

学校代码 10487 密级_____

華中科技大學

博 士 学 位 论 文

面向群智感知的

边缘计算资源调度机制研究

学位申请人： 侯海翔

学 科 专 业： 计算机系统结构

指 导 教 师： 金海 教授

答 辩 日 期： 2019 年 1 月 5 日

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Philosophy

EN Title

Ph.D. Candidate : Haixiang Hou

Major : Computer Architecture

Supervisor : Prof. Hai Jin

Huazhong University of Science & Technology

Wuhan 430074, P. R. China

January 5, 2019

独创性声明

本人声明所呈交的学位论文是我个人在导师的指导下进行的研究工作及取得的研究成果。尽我所知，除文中已标明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于　保密 □，在 ____ 年解密后适用本授权书。

　　不保密 □。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

摘 要

中文摘要。

关键词：

Abstract

English abstract.

Key words:

目 录

一 绪论参考资料

随着 5G 通信标准的确立，物联网时代正式拉开了序幕。种类繁多的移动设备、高度集成的传感器、以及万物互联的网络空间，促使群智感知应用对存储资源、网络资源和计算资源的需求达到了更高的境界。云计算作为集中式服务模型，已经难以支持覆盖区域广、时效性强的群智感知应用。边缘计算作为一种与云计算互补的分布式计算模型，如何利用其特性为群智感知服务提供基础支撑，是一个亟待解决的研究课题。本章首先介绍群智感知场景下边缘计算中的资源调度问题；然后简要介绍当前国内外研究现状和现有工作的不足；接着阐述了本文研究的目的与意义；最后介绍论文的组织结构和层次关系。

1.1 研究背景

1.1.1 边缘计算

边缘计算是与云计算[1] 相辅相成的一种新型计算模型。云计算作为集中式处理模型，需要将用户数据收集至部署有云平台的数据中心[2]，才能有效地服务于用户。但在面临万物互联的新型网络空间时，云计算服务将面临以下挑战：

1) **实时性不足**。在物联网场景中，存在着许多强实时性的应用。在云计算中，数据需要不断在客户端和云端之间往返，导致响应时延增加、用户体验下降。例如在智能驾驶应用中，云计算无法达到毫秒级的数据处理时延[3]。移动终端上的虚拟视觉（Virtual Reality，VR）框架 Furion[4] 在探索过程中发现，仅靠云计算服务无法帮助移动终端获得高质量的实时 VR 服务。而利用边缘计算将渲染服务卸载到边缘服务器中，MUVR[5] 成功实现了一个低通信且稳定的多用户 VR 框架。

2) **带宽不足**。在网络空间中，将边缘设备产生的数据传送至云计算中心，需要消耗大量的带宽资源。Intel 在 2016 年的报告[6] 指出，一辆智能驾驶汽车工作一天可以产生 4 TB 的数据。而一架波音 787 飞机在飞行途中，其数据产生速率达到了 5 GB 每

秒[7]。如此巨大的数据量,利用云服务进行保存和处理,不仅需要耗费大量的带宽,且网络传输导致的时延也将导致计算服务毫无意义。

3) **能耗过高**。云计算进入实践以来,研究者们更多关注数据中心中的能耗问题。据《中国数据中心能效研究报告》[8] 显示『白皮书引用格式』,我国 2015 年度数据中心能源消耗已经超越 1000 亿千瓦时。在美国,2013 年度数据中心总能耗就已达到 910 亿千瓦时,Sverdlik 在调研 [1]中预测,2020 年度全美数据中心能耗会到达 1400 亿千瓦时。随着用户、应用数据的增加,全球数据中心的能耗仍在进一步上升。

4) **用户数据安全和隐私**。在外物互联的网络空间中,边缘设备已经走入用户的私人生活空间。例如家用摄像头、便携式移动设备、智能网联汽车,这些设备中都拥有大量的私人隐私数据。一旦这些数据被上传至云端,用户隐私的泄露风险也会成倍增加。目前,欧盟已经强制实施"通用数据保护条例"(General Data Protection Regulation, GDPR)来保护用户的隐私。对于云计算公司,如何保护用户的数据安全和隐私安全,也更为重要。

为了避免云计算模型在物联网场景下的不足之处,2013 年西北太平洋国家实验室的 Ryan LaMothe 提出一种新性计算模型概念——边缘计算(Edge Computing)。2015 年,欧洲电信标准协会 [2](European Telecommunications Standards Institute, ETSI)在白皮书[9] 中正式定义了边缘计算的概念。边缘计算就是将应用任务在靠近数据源(例如移动设备、传感器、最终用户等)的资源上进行处理。"边缘"主要相对于云计算中心而言,指代数据产生源到附近的任意计算、存储、网络资源。同年,思科也推出了雾计算白皮书[10]。雾计算这一概念最早于 2011 年由 Bonomi 首次提出[11]。雾计算通过虚拟化架构将远端的云端服务迁移至本地节点,让高高在上的云服务更贴近用户,从而提高应用服务的访问效率和服务质量。『虽然边缘计算和雾计算在指导思路上有相似处,都是将云端服务落地到更接近用户的网络边缘侧。但是雾计算更多探讨的是实际应用的落地;边缘计算更倾向于研究边缘计算体系结构中的问题。所以学术界更愿意探讨边缘计算下的本质问题。』

[1]https://www.datacenterknowledge.com/archives/2016/06/27/heres-how-much-energy-all-us-data-centers-consume/

[2]https://www.etsi.org

『典型的边缘设备

缺图』

相较于云计算模型,边缘计算具有 3 个明显的优势:

1)**大量临时数据不需要上传至云端服务器**。在边缘计算中,应用程序可以利用边缘节点中的资源完成数据的存储、计算工作。避免用户数据上传至云端服务器,为主干网络节省了大量的带宽。

2)**计算任务不再需要云计算中心的响应**。在边缘计算中,云端服务可以卸载至资源丰富的边缘节点上。更短的网络路径让应用服务的响应延时大大减少,不仅增加了响应能力,也提高了用户的使用体验。

3)**隐私数据保存在边缘设备,无需上传**。由于用户隐私数据可以存储在边缘设备中而不是云端服务器,减少了隐私数据的传输路径,在一定程度上规避了隐私泄露的风险。

得益于这些优势,近年来边缘计算得到了突飞猛进的发展。利用边缘计算,不仅可以在广袤的无线网络接入范围内提供更好、更快、更准确的信息技术服务和云计算能力;还可以在数据的边缘,利用富余的资源快速完成应用服务。目前,边缘计算已经被采用在以下典型应用场景中:公共安全中的实时数据处理、智能网联车和自动驾驶、虚拟现实、工业物联网、智能家居以及智慧城市。这些真实的应用场景与人类的生活、生产、娱乐息息相关。并且这些应用还有一个共同的特点,它们的需要从大量的边缘设备中收集大规模的传感数据作为『数据分析』的基础。这种应用范式,也叫群智感知。

1.1.2 群智感知

无线传感网络 + 为什么要有群智感知

群智感知是一种基于物联网的“以人为本”的感知模式。其构想起源于 2006 年《连线》(Wired)杂志提出的众包一词,旨在利用分布式解决方案将感知工作分配出去来共同完成应用任务或提供服务。利用这一特性,群智感知可以完成个体无法实现的复杂环境下大规模动态社会感知任务,例如交通拥堵状态、城市空气质量监测等。在

早期,这些社会感知任务可以利用无线传感网络[12] 完成。由于不同类型的感知任务需要部署不同功能的传感器甚至不同架构的传感网络,随着社会感知任务需求的增多和变更,静态的传感网络在部署和维护上都会耗费大量的人力成本和物力成本[13]。

『**缺图**』

群智感知利用众包的方式,将感知任务众包给用户的移动设备(例如手机、平板、智能手表等)作为基本感知设备。这些移动设备集成了丰富的传感器,可以获取大量与设备所处环境相关的数字信息,例如环境光(光学传感器)、噪声(麦克风)、地理位置(GPS 传感器)、移动状态(陀螺仪、加速计)等。除此之外,这些移动设备还可以利用自身优秀的通信能力快速交付数据,甚至利用本地计算资源进行数据处理。将对于无线传感网络而言,群智感知应用的部署成本少、灵活性高,更适合复杂网络环境下的大规模动态社会感知任务。

除了移动设备内置了大量传感器之外,大量的现代交通工具中,雷达、摄像头、GPS、陀螺仪、加速计等设备也已经成为保障安全驾驶的必备传感器。而物联网的飞速普适,也让更多的传感设备具备了网络连接功能和简单数据处理逻辑。因此,群智感知可以收集到种类更多、信息更全的感知数据,从而实现更多创新型研究与应用。西北工业大学利用校内学生的智能手机,在校区内实现了基于群智感知的噪声监测系统[14]。通过众包方式收集不同位置和不是时间的噪声污染数据,该系统利用离散数据重建出高精度的城市噪声时空分布地图,为城市噪声治理提供了可视化的监控平台。论文[15] 利用车内传感器和道路行人手机内的传感器,将多个信息源混合在一起。在不借助额外传感器和通信网络的前提下,搭建智能交通拥堵检测系统。Waze [1]是一款基于群智感知的导航服务。在传统的允许用户帮助编辑图资的基础上,Waze 开创性的引入用户的 Facebook 和 Twitter 消息,实时更新周边商户信息和道路交通状态。通过整合用户数据,Waze 比传统的导航服务内容更加丰富、信息也更精准。

在物联网环境下,人类社会和网络空间中已经包含了海量的、具备多元化传感功能的智能设备。众多的潜在参与者、饱满的空间覆盖、便利的数字信息提取,让群智感知成为了环境研究[16]、人类社会关系研究[17]、智慧城市建设研究[18] 的基础工具。

[1]www.waze.com

1.1.3 边缘计算与群智感知应用

边缘计算在群智感知中的作用

资源管理分配与任务调度

传感(数据采集)、通信(数据传输)、计算(数据处理)

总括：面向群智感知的边缘计算资源调度,现有研究有哪些不足

群智感知应用的工作流程可以简单划分为四步[19]:任务分发,传感器感知,数据上传和数据处理。其中,任务分发是指根据群智感知任务的覆盖区域、执行任务的用户数量、以及任务的持续时间,快速将感知任务发送到合适的移动设备上。移动设备收到感知任务后,通知传感器工作并产生感知数据。然后,移动设备利用自身的通信能力,将传感数据上传到等待数据处理的地方(例如云服务器)。最后,对这些数据进行去重、映射、分析,从而推断出符合共同利益的结果[20]。

『插图』

早期,群智感知应用通常借助云上资源来进行感知数据的保存和处理。随着群智感知应用的需求增加,覆盖范围扩大,所收集的感知数据『越来越多』,对云端资源的需求也成倍递增。利用边缘计算可以将服务卸载到数据产生源附近的特征,可以有效缓解群智感知应用对云计算资源的压力。**举例：MCS+Edge Computing** CrowdITS Crowdsourcing in Intelligent Transportation Systems 阿里巴巴的城市大脑 2.0 [1]

在边缘计算模型中,为了保障群智感知服务的稳定性和服务质量,必须针对群智感知的工作阶段对边缘网络中的各种资源进行调度和管理。这也意味在群智感知的数据采集阶段、数据传输阶段、以及数据处理阶段,需要对边缘网络中的网络资源、计算资源、存储资源进行合理的划分和调度。

前文已经阐明边缘计算旨在将云上服务迁移至靠近移动设备的网络边缘侧。而群智感知需要汇聚大量移动设备收集海量的传感数据,为具体应用提供服务支撑。这两种技术的有机结合,已经成为智慧城市建设的前沿研究热点。

[1]https://damo.alibaba.com/labs/city-brain

1.1.4 面向群智感知的边缘计算研究及其挑战

目前,边缘网络正在快速发展阶段。

编程模型应用、服务的功能划分。亚马逊的 Lambda 计算服务,可实现任务的自适应卸载。

虚拟化技术

代码卸载 (code offloading)

硬件标准、软件标准

动态调度数据可分布(任务分发 + 数据采集)

网络可分布(SDN 中网络管理和路径规划)

资源可分布(边缘服务节点弹性部署)

1.2 国内外研究现状

针对群智感知应用,边缘计算的实现具有三大挑战:

1)利用边缘计算提高群智感知的覆盖率和感知质量

2)边缘计算下大规模群智感知数据的链路调度

3)边缘网络中的群智感知服务卸载调度

包含『关键技术』和『应用举例』

1.3 研究意义与目的

1.4 论文组织结构

二 绪论

本章首先介绍群智感知技术和边缘计算的相关研究背景,然后简要介绍当前国内外研究现状和现有工作的不足。通过分析群智感知应用在大规模场景下所存在的缺陷,明确了设计面向智能感知的边缘计算资源调度机制的目的和意义。在讨论国内外针对大规模群智感知的研究现状及不足之处之后,进一步介绍论文的研究内容和主要贡献。最后介绍论文的组织结构。

2.1 研究背景

2.1.1 群智感知技术

群智感知技术也被称为移动群智感知技术。群智感知技术通过租用大量具备感知能力和计算能力的移动设备(例如智能手机、平板电脑、可穿戴设备、智能车辆等)共同提取、共享数据,并对这些数据进行去重、映射、分析,从而估计、推断出符合共同利益的结果[20]。在现实世界中应用群智感知技术,需要大量的传感器在不同空间中产生传感数据,并利用网络将传感数据汇总并交付给计算资源进行处理。

如今,配有各种传感器的智能设备已经无处不在。常见的智能手机已经集成了大量的传感器,它们不仅可以获取与设备所处环境相关的数字信息,例如环境光(光学传感器)、噪声(麦克风)、位置(全球定位系统 GPS)、移动状态(陀螺仪、加速计)等,甚至还可以获取与使用者自身相关的数字信息,例如运动状态、心跳频率等。在大量的现代交通工具中,雷达、摄像头、GPS、陀螺仪、加速计等设备也已经成为保障安全驾驶的必备传感器。因此在人类的生活空间中,已经遍布了大量具备多元化传感功能的智能设备,并且这些智能设备的使用者大多数为自然人。所以,群智感知技术的潜在参与者多、空间覆盖广,让其成为了环境研究[16]、人类社会关系研究[17]、智慧城市建设研究[18] 的基础工具。利用群智感知技术,研究者们已经完成了很多有启发性的工作。近年来,移动智能设备的能力得到了极大的发展,对于在不进行大规模投资的情况下收

集数据的企业来说，群智感知已经成为一种极具吸引力的方案。许多技术公司都愿意使用群智感知收集大量的数据以提供额外的高品质服务。

群智感知应用的工作流程可以简单划分为四步[19]：任务分发，传感器感知，数据上传，数据处理。在任务分发过程中，需要决策群智感知任务的覆盖区域，执行任务的用户数量，以及任务的持续时间。根据实际需求将群智感知任务部署到合适的智能设备上。传感器感知是指智能设备在收到任务之后，通知传感器工作并产生感知数据。数据上传则是在传感数据产生后，由智能设备利用自身的通信能力，将传感数据上传到等待数据处理的地方（例如云服务器）。待大部分传感数据收集完成后，由云端服务器进行去重、映射等操作，并利用数据挖掘、数据分析得出可用的结论。

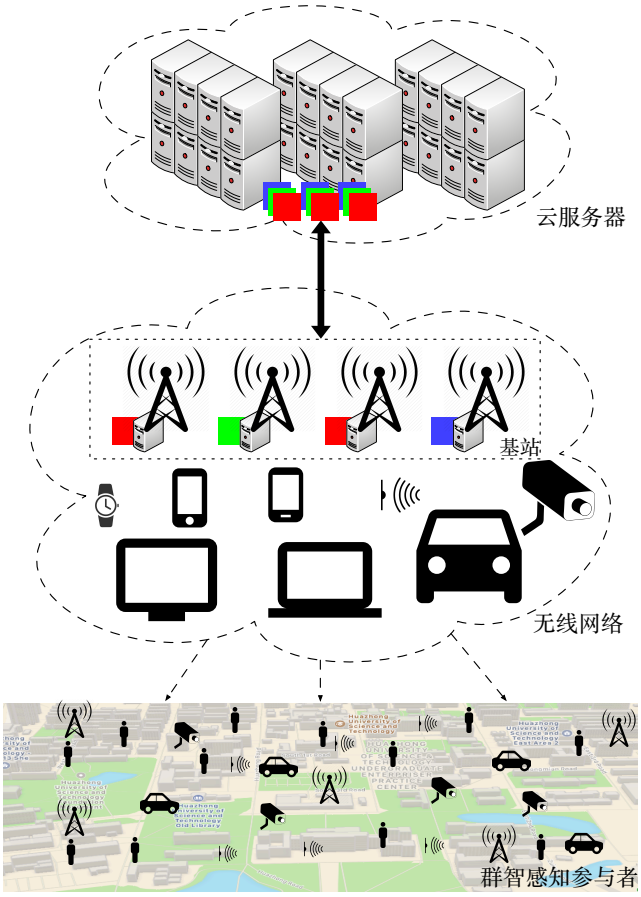


图 2-1　群智感知应用典型场景

图 2-1 描述了典型的群智感知应用场景。图中，智能手机、可穿戴设备、笔记本跟随自然人的位置分布在不同的地理位置。在该区域中，还有行驶中的车辆、监控摄像头、以及其它无线传感器散布在四周。利用网络将传感器的数据收集之后，交付给云

端服务器。云端服务器通过对传感数据的处理和分析,得到该区域中的相关的环境数据、以及区域中的自然人所相关的数据。因此,群智感知技术在实际使用过程中,往往需要部署借助云计算服务以完成数据的后台处理工作。

2.1.2 边缘计算

近年来,物联网和 5G 通讯技术取得了突飞猛进的发展,万物互联的时代也正在加速到来。如今,不仅网络边缘的设备数量在急剧增长,这些设备产生的数据量也呈现爆炸式增长的趋势。据 IDC 预测[19], 2020 年全球数据生产总量将超过 40 泽字节(zettabyte, ZB),而物联网中产生的数据占比达到了 40% 以上。届时,集中式处理模型,例如传统的云计算模型,将会面临带宽不足、延时较高、能耗效率低、安全隐私风险高等种种问题。为了避免集中式处理模型所即将面临的问题,边缘计算模型应运而生。

边缘计算是一种面向大量边缘设备所产生的海量数据的计算模型[21]。它是一种将数据处理和数据存储放在网络的边缘节点的分布式计算形式。在靠近终端设备、或者数据源头的网络边缘侧,融合通信、计算、存储、应用能力而构建分布式计算平台,是边缘计算的核心思想。在边缘计算中,可以高效结合 WiFi 网络、5G 网络、端到端通信、软件定义网络等通信手段,将云端服务本地化,提供“智慧化”服务的基础。边缘计算利用分布在距离终端最近的基础设施,为网络边缘设备提供具有针对性的服务与算力。这些算力可以在数据源端完成部分的数据预处理任务,『数据预处理包括但不限于』。在数据预处理完成后,再将剩下的数据传回云端,根据不同的业务进行最终的处理。因此,边缘计算将网络结构和云端资源进行协同配合,为个人或企业提供满足技术需求的弹性计算服务。

『缺图』

图 **??**描述了典型云计算场景和边缘计算场景的不同之处。相较于云计算框架,边缘计算模型的优势主要有 3 点。1)边缘计算中,产生于网路边缘的数据并不用全部上传至云端,这一特性能够有效减轻骨干网络中的带宽占用;2)在靠近数据生产者处完成大部分的数据预处理工作,减少对云计算资源的依赖,不仅可以减少服务的延时,

还能提高服务的响应能力;3)用户数据不用上传至云端服务器,而是暂存在网络边缘设备上,减少了用户隐私泄露的风险。

边缘计算利用自身的优势弥补了云计算模型中海量数据传输延时高,隐私安全处理敏感等问题,使得边缘计算更适用于新兴的物联网应用场景。因此,近年来边缘计算得到了快速的发展和完善。将云计算能力扩展至距离终端最近的边缘侧,以满足"大连接,低延时,大带宽"的新需求。目前,边缘计算已经逐步应用在和人类生活息息相关的各类场景中,例如『公共安全中的实时数据处理、智能车联网、自动驾驶、虚拟现实、工业互联网、智能家居和智能城市』

2.1.3 面向群智感知的边缘计算研究及其挑战

近年来,业界和学术界已经着手将边缘计算结合到实际生产与应用中。这些应用,大多数与智慧城市建设密切相关。例如,论文[20]在车辆上部署 GPS 和加速度计,可以在车辆行驶过程中定位城市中的坑洼路面;科特迪瓦的一个环境非政府组织在非洲阿比让市电信公司的帮助下,在阿比让市利用 25 个蜂窝基站和群智感知应用,向阿比让市民即时提供空气质量感应结果[];『举例』。

这些研究工作与实际应用,和群智感知以及边缘计算都密不可分,这也揭示了面向群智感知的边缘计算模型在未来城市建设中的重要地位。随着群智感知应用的应用领域和感知范围逐步扩大,如何将边缘设备有机地组织起来,合理调度边缘网络中的计算资源、网络资源,以及云端资源的合理分配,这些研究内容都为边缘计算技术实际落地带来了新的挑战。目前,面向群智感知的边缘计算中主要研究挑战有三点。

第一个挑战是**边缘计算中面向群智感知的任务调度问题**。在群智感知应用中,需要大量的参与者和边缘设备为群智感知服务提供输入数据。然而,边缘设备自身并不具备群智感知服务的发现能力。因此,群智感知服务需要将数据采集任务利用网络分发到合适的边缘设备上,雇佣这些智能设备收集必要的传感数据,并反馈给附近的边缘计算节点。在真实场景下,群智感知的参与者多为边缘设备的拥有者,且大多数参与群智感知的边缘设备为移动设备,所以这些边缘设备通常具有一定的移动特性。

但是由于群智感知服务往往具备基本的实时性要求[],所以以上四个步骤在实际

执行中是连续且不可分割的。因此,在边缘网络中,边缘设备

近年来,参与群智感知应用的边缘设备集成度越来越高,功能越来越强大,其可使用的网络接入技术也逐渐多样化。在早期的群智感知研究工作中,群智感知的参与设备多使用蜂窝通信[22,23] 来上传感知数据。随后,Karaliopoulos 等人的研究[24] 发现,在传感数据上传过程中利用端到端通信,可以减少边缘设备的能耗成本的网络通信成本,并以此来为群智感知服务雇佣更多的参与者。同时,论文[25] 通过研究机会式通信网络[] 中的数据包转发规律,发现在边缘网络中利用数据包融合技术可以有效减少数据传输延时。随着 5G 和物联网技术的飞速发展,边缘设备可以利用蓝牙技术、WiFi 技术、进场通信技术、端到端通信技术在小范围内快速交换数据信息。

在群智感知应用中,服务所需要的原始数据基本来源于边缘设备的传感数据。在该场景下,用户和边缘设备多处于运动状态中[],例如智能车联网中的车辆、群智感知应用中的参与者。这些设备和参与者随着地理位置和所处网络环境的改变,都会在网络中引起网络拓扑结构的变化。然而,群智感知服务往往以连续的方式收集参与者的信息并提供反馈结果。所以在边缘计算实际应用中,随着参与者或者边缘设备状态的改变,边缘网络需要时刻对边缘设备的网络状态进行维护并合理调度边缘网络中的数据流[]。

第二个挑战是**边缘计算中面向群智感知的服务资源资源调度问题**。

第三个挑战是**边缘网络中的『』调度问题**。

1)网络自组织结构:在群智感知应用中,有大量的边缘设备参与其中。首先,边缘计算中,服务端从云端转变到网络边缘侧,如何让这些边缘设备快算获取自身周边存在的服务,是边缘计算在网络层面的一个重要问题。其次,在边缘计算场景中,用户和边缘设备的参与方式均为动态过程,例如车联网、群智感知应用中的参与者。这些设备和参与者随着地理位置和网络环境的改变,都会在网络中引起拓扑结构的变化。如何针对变化拓扑部署并迁移边缘服务,也是边缘计算在网络层需要考虑的难题。再者,边缘设备会产生大量的传感数据,这些数据虽然不会全部上传至主干网络,但如何平衡边缘网络中的数据负载,合理调度边缘网络中的数据流量,也是边缘计算网络层需要解决的问题。最后,随着无线通信技术的发展,在边缘网络中可以利用的通信

技术也变得多样化。针对不同业务场景合理使用蓝牙、WiFi、5G、或者蜂窝网络也成为边缘网络中的热门研究方向。

2) 数据处理框架: 在群智感知应用中, 边缘设备无时不刻提供的传感数据构成了海量数据场景。并且, 这些海量数据具备多样性、体量大、时间相关性、地域相关性、以及冗余性等特点。如何对这些海量数据进行实时处理, 对边缘计算提出了新的数据处理需求。因此, 构建一个针对边缘数据进行管理、处理、分析以及共享的框架更显得格外重要。与此同时, 随着人工智能技术[] 的快速发展, 很多边缘设备还需要结合自身数据运行多种智能算法。例如自然语言的处理[]、实时语言翻译[]、智能车辆导航[] 等应用, 都需要机器学习相关算法的参与。因此在边缘计算的数据处理框架中, 不仅需要针对传感数据流进行管理, 还需要兼容常见的的机器学习框架, 例如谷歌的 TensorFlow[]、开源的 Caffe[] 等。

3) 安全和隐私: 由于群智感知应用发生在最靠近用户的网络边缘侧, 部分数据可以避免被上传到云端, 在一定程度上降低了用户隐私数据泄露的风险[26]。然而, 相较于云计算中心, 边缘网络中的某些节点依然潜伏着安全隐患, 例如伪造的、或者被不法人员操控的无线接入点等。因此, 参与边缘计算的设备或者用户, 其隐私和安全一样受到安全隐患的威胁。由于边缘网络中, 边缘计算节点的分布式架构和异构性也让计算节点难以进行统一化管理, 从而导致新的安全问题和隐私泄露隐患。针对这一系列安全隐患, 科研人员将其归类为应用安全、信息安全、网络安全和系统安全。在传统分布式计算架构中, 这些安全隐患可以利用可信执行环境[27,28] 进行有效的规避。例如, Intel 软件防护管理技术[29]、Intel 内存加密技术[]、AMD 平台安全防范技术[]、AMD 内存加密技术[] 等, 这些技术不仅可以保障执行平台和环境的安全性, 还可以对存数数据进项加密防止数据泄露。通过这些技术进行修改, 适配到边缘计算中, 可以保障在边缘计算节点出现隐患时, 依然保障应用和数据的可靠性和安全性。

在面向群智感知的边缘计算框架中, 论文针对

利用 GPS 和麦克风, 可以绘制一个区域的噪声污染情况。[20] 总结出群智感知的三类应用场景: 1) 环境监测; 2) 基础设施监测; 3) 社交状态跟踪。

通过用户的参与状态, 群智感知又可以分为两类。第一类是参与式群智感知, 用

户作为志愿者自愿参与群智感知并提供数据信息。第二类是机会式群体感知,智能设备能够自动地感知、收集、并共享数据,整个过程并不需要用户的主动干预。

其中一些著名的公司就包括 FaceBook、Google、和 Uber。

2.2 国内外研究现状

如今,群智感知技术已经和人类的生活密不可分。作为分布式感应数据收集的基础框架,群智感知技术已经普遍应用于智慧城市的建设当中。智慧城市是指把新一代信息技术、网络技术充分应用在城市管理与建设过程中的城市信息化高级形态。而城市中的各类信息收集,则是智慧城市落地的奠基石。利用群智感知技术,能够产生大量基于自然环境、人类社会活动等多维度的海量数据。通过对这些数据进行数据挖掘、学习和分析,不仅可以了解居民的生活习惯、城市环境状态、城市能源消耗等多维状态,更能为实现以人为本的全面可持续发展道路提供基础支撑。

2.2.1 边缘计算中面向群智感知的任务调度机制

2.2.2 边缘计算中面向群智感知的服务资源资源调度

2.3 研究内容和意义

ddd

2.4 论文组织结构

第一章为绪论。首先给出面向群智感知的边缘计算技术的的研究背景和面临的挑战,然后介绍现阶段国内外研究现状,最后给出研究内容和主要贡献以及论文组织结构。

第二章为『』

第三章为边缘网络中基于移动模型的群智感知边缘服务调度机制。首先基于随机移动模型建立群智感知任务分发和时间的关系,并依据传染病模型对群智感知任务

分发的过程建立数学模型。接着分析群智感知任务的持续时间和边缘节点的移动模型对群智感知任务分发过程以及数据回收过程的影响,并提出群智感知任务的覆盖率。再提出提高群智感知任务覆盖率的优化算法。最后结合仿真模拟验证优化算法对群智感知任务覆盖率的效果。

第四章为边缘网络中基于社交模型的群智感知边缘服务调度机制。首先基于城市公交路线建立群智感知边缘服务部署框架。通过分析城市居民在公共交通中的行为,建立群智感知数据回收和成本的关系模型。基于此种模型,提出基于公交路线的边缘服务部署决策算法。最后采用仿真模拟验证算法效率以及成本节省比例。

第五章总结全文,概括文中的主要贡献并展望未来的研究工作。

三　边缘计算中面向群智感知的任务调度机制

边缘计算是一种新的云端服务模型,它为许多移动计算应用提供了新的范式,尤其是群智感知应用。在群智感知应用场景中,任务分发服务和数据收集服务均可以部署在边缘代理节点(例如基站等)上。利用边缘节点之间的机会式通信进行数据交互,可以将群智感知任务分发到更多的边缘设备上,并避免过多的成本开销。所以,边缘计算会对群智感知应用的服务部署和服务质量造成『可见的影响』。为此,『本章』针对群智感知应用在边缘网络中的数据传输行为进行『随机性分析』。通过引入常微分方程组来描述任务的传播过程和数据收集过程,并将群智感知应用可实现的覆盖范围推到为带有参数的函数。『其中参数包含边缘代理节点数量,边缘设备数量,边缘设备的相遇率等』。利用数学分析具有截止期限的群智感知应用,『本章』提出了可以找出群智感知应用中任务分发和数据收集的最佳时间分配算法。借助模拟测试和评估,验证了『本章』分析方法的正确性,时间分配算法的可行性。评估结果表明,『本章』分析方法的结果和模拟数据的平均误差小于 9.6%。

3.1　研究背景

在传统的云计算模型中,大量的用户数据被收集并发送至云端服务器,由云端的计算资源对海量数据进行处理。边缘计算模型和云计算模型相背而驰,通过探索网络边缘中的边缘节点(例如蜂窝网络中的基站)的可利用资源,将云端服务器的数据处理逻辑部署到网络的边缘侧,使计算、数据处理资源更加贴近数据生产源。由于边缘网络中边缘设备繁多、地域分布广、且靠近用户,研究者们普遍认为边缘计算可以显著提升网络应用的服务质量[30]。近年来,学术界和工业界都在不断挖掘边缘计算的巨大潜力,研究者们都致力于将边缘计算应用到不同领域的开创性工作中[31,32]。

目前,移动计算领域和边缘计算正日益紧密的结合在一起。鉴于边缘计算的地域分布特性以及数据处理更靠近数据源,移动计算应用可以利用边缘计算实现高带宽、

低延时的实时服务。群智感知是移动计算领域中的一种典型应用。它利用无处不在的无线网络和众多移动设备中的内置传感器来完成『环境』感知。不同于传统的专用型无线传感器网络，群智感知可以更方便、快捷地收集与人类生活、自然环境息息相关的各种信息[33]。不仅如此，群智感知还被广泛应用于其它不同领域，例如 WiFi 性能测量[34]、天气预报[25]、空气质量监测[35]、城市噪声监测[36] 和城市智能交通建设[37]。

当使用云计算技术支撑群智感知应用时，如论文[38,39] 中所讨论，所有的感知数据都会被上传到云端。由于产生感知数据的移动设备(例如智能手机、智能手表)和位于数据中心的云端设备距离遥远，感知数据在传输过程中可能会遇到较高的网络延时和不可预测的网络抖动。通过边缘计算的支援，群智感知应用可以探索边缘节点中的可利用资源(例如蜂窝网络的基站或者其它大型无线网络接入点)，进行任务转发和数据收集工作。『为了进一步扩大群智感知应用的适用范围，任务分发工作和数据收集工作都可以作为服务被部署到合适的边缘节点中』。如图 3-1所描述的场景中，当移动节点

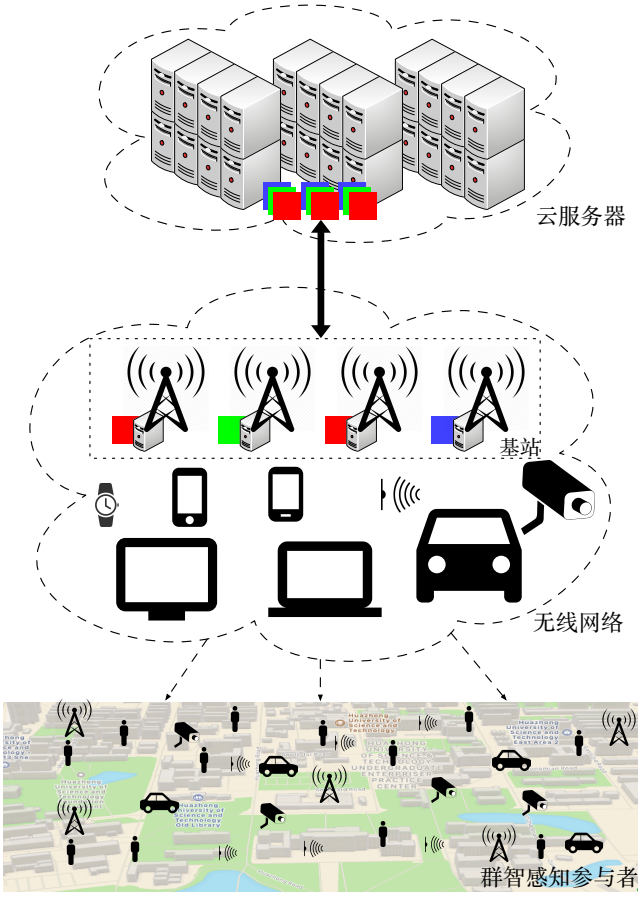


图 3-1　群智感知应用典型场景

收到来自边缘节点的群智感知任务是,这些移动节点可以执行相应的传感操作并将感知数据反馈给边缘节点。介于蜂窝通信的成本较高,减退了许多潜在群智感知参与者的热情。为了解决这一问题,许多学者研究了群智感知中参与者的激励机制,也有部分学者发现将端到端通信机制引入群智感知应用,同样可以缓解这一问题[40]。

在本章节中,边缘计算被用以辅佐群智感知应用中的任务分发过程和数据收集过程,以提高群智感知应用的覆盖范围和服务质量。在任务分发过程中,已部署任务分发服务的边缘节点作为任务分发的源头,不断地将群智感知任务分发至移动节点。在数据收集过程中,移动节点作为感知数据的源头,将数据回传至已经部署数据收集服务的边缘节点。任务的传播和感知数据的转发不仅可以通过蜂窝数据进行发送,也可以通过端到端通信的方式传输。现有的研究[41–43] 已经分析了端到端机会式通信网络中的信息传递延时。这些研究工作表明,端到端通信机制中消息传递延时的影响因素主要有移动节点的数量和移动节点的相遇率。然而,这些研究工作中通常在分布式节点中选取一个源节点和一个目标节点。但是在边缘网络强化的群智感知场景下,存在着复数个对等的源节点和目标节点。这些工作并不适合用来分析边缘计算下的『群智感知』。因此,为了提高群智感知服务的质量,必须用新的方法量化边缘网络中服务资源部署对群智感知应用的影响。在该述求下,第一个挑战就是探索服务的部署方案影响群智感知应用的覆盖范围和服务质量的机理。另外,群智感知应用中的设备随着时间的推移会改变自身的位置。所以第二个挑战是探索移动节点的移动模型对群智感知应用服务质量的影响。

为了解决这两个问题,本章节在边缘计算下的群智感知中引入端到端通信,并对数据传输过程进行理论分析性能分析。本章节的主要贡献如下:

1)本章利用常微分方程组,以描述边缘计算环境下的群智感知中的任务分发阶段和数据收集阶段。通过解常微分方程组进行分析和求解,可以量化群智感知应用的覆盖范围并推导出可实现的感知质量。

2) 由于群智感知服务的具有时效性,因此本章设计了一种算法时间划分算法,以找出任务分发阶段和数据收集阶段的最佳时间分配,帮助群智感知应用获得更好的感知质量。

3) 本章经过大量的模拟实验分析, 验证了模型的正确性和准确性。同时, 基于仿真的测试还验证了时间分配算法的效率。

3.2 系统架构

本章节中, 采用了和论文[25] 相似的群智感知应用范式, 主要研究对象为群智感知应用中的两个重要过程: 任务分发过程和数据收集过程。其中, 任务分发过程是指已部署任务分发服务的边缘节点将任务分发至边缘网络中的移动节点, 数据收集过程是指已经接受到任务的移动节点将感知数据反馈给已部署数据收集服务的边缘节点。考虑到群智感知参与者本身具备移动性, 本章也将移动节点的运动特征纳入考虑范围。本小节重点介绍分析模型中的参数定义和模型建立。

3.3 Preliminaries

在边缘计算场景下, 群智感知中的任务分发服务和数据收集服务都可以部署在边缘节点中。在不失普适性的情况下, 本章假设蜂窝网络的基站作为能够承载这些服务的边缘节点。此时群智感知服务的流程大体分为四步: 1) 群智感知的服务发起者选择合适的基站将任务分发服务和数据收集服务部署到这些基站上; 2) 在任务分发阶段, 部署有任务分发服务的基站利用蜂窝网络和端到端通信将群智感知任务发送到附近的移动节点上; 3) 移动节点在收到群智感知任务之后, 一边处理任务, 一边利用端到

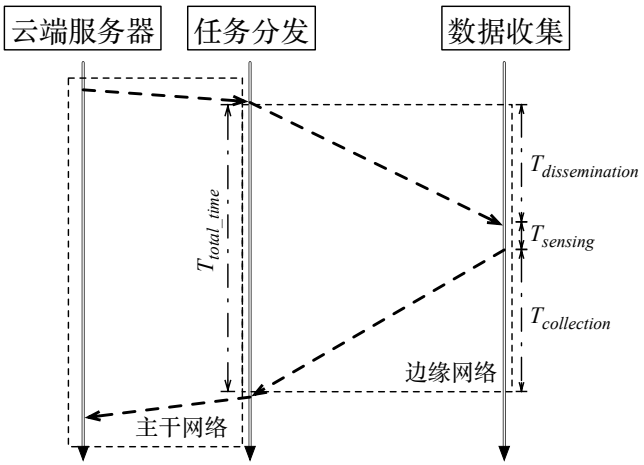


图 3-2 『群智感知的时间』

端通信将任务广播给附近的其它移动节点;4) 已经完成任务的移动节点,可以利用蜂窝网络或者端到端通信,将感知数据反馈给已经部署有数据收集服务的边缘节点。由于本章节的重心并不是激励移动节点收集数据,所以在整个群智感知应用执行期间,论文假设所有的移动节点都是群智感知应用的志愿者。图 3-2描述了边缘计算环境中,群智感知应用中各阶段的时间开销。在云端服务器将任务分发服务和数据收集服务部署至边缘节点之后,$T_{dissemination}$ 表示群智感知任务从边缘服务节点发送到移动节点的时间开销;$T_{sensing}$ 表示移动节点执行任务需要的时间;$T_{collection}$ 表示移动节点将感知数据反馈回边缘服务节点的时间;T_{total_time} 则为三者之和。

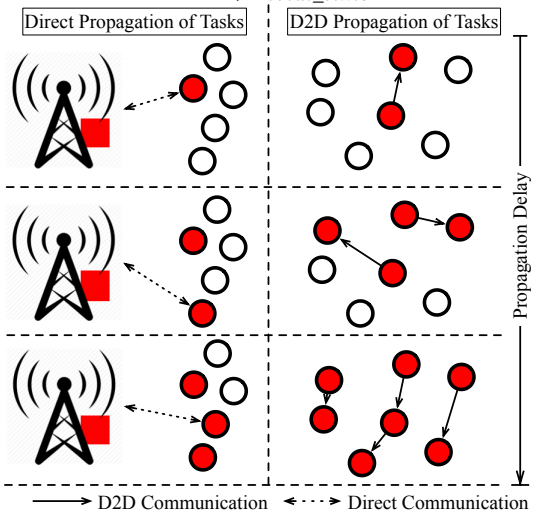


图 3-3 The task dissemination phase

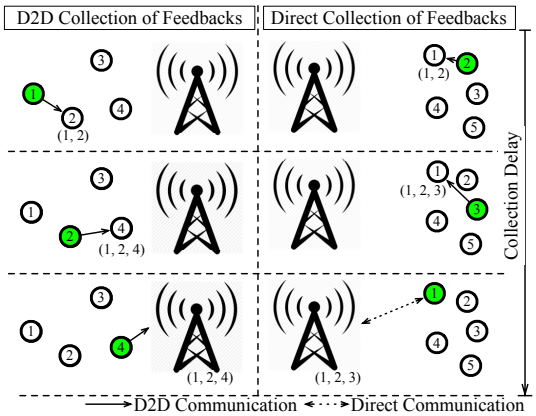


图 3-4 The data collection phase

图 3-3展示了边缘计算下的群智感知服务利用蜂窝网络和端到端通信完成任务分发。首先,部署有任务分发服务的基站利用蜂窝网络将任务发送至移动设备。已经收到任务的移动设备,在其移动过程中,会有一定的概率遇到没有接收到任务的移动

节点。此时,已经收到任务的设备利用端到端通信,将任务分发至还没有收到任务的设备。待任务执行完成后,移动设备可以利用蜂窝网络将感知数据传送给基站,也可以利用端到端通信将结果委托给在移动过程中相遇的其它设备,并传送给基站。这一过程,如图 3-4所示。

3.3.1 系统建模

本章将边缘网络中的移动设备分为两类:一类是『部署有任务分发服务和数据收集服务的边缘服务节点』;另一类是『执行群智感知任务的边缘设备』。其中,边缘服务节点上同时部署任务分发服务和数据收集服务。本章中,边缘服务节点的数量用 N_f 表示。在群智感知应用的目标覆盖区域内,边缘设备的数量记为 N,且这些设备都可认为是群智感知服务中的志愿者。由于边缘设备在移动的过程中,会与其它边缘设备相遇,利用相遇的过程,可以利用端到端通信完成任务分发和数据转发。这里假设在边缘设备的接触时间内数据交换工作能够顺利完成。

由于边缘终端的移动特性,每个移动设备都有一定的概率遇到相邻设备并进行端到端数据交互。此处用 λ_f 表示边缘服务节点和边缘设备之间的相遇概率;用 λ_n 表示任意边缘设备之间的相遇概率。在边缘网络中,边缘设备的相遇过程可以利用不同的移动模型进行推导。在 Groenevelt 等人的论文[44] 中,随机路径移动模型中的相遇概率可以用式 3.1进行计算。其中,d 是设备的通信距离,$\mathbb{E}[V^*]$ 指的是特定区域内所有节点的平均速度,A 表示所有节点通信半径覆盖的区域面积,w 是常数,其近似值为 1.3683。

$$\lambda = \frac{2wd\mathbb{E}[V^*]}{A} \tag{3.1}$$

鉴于群智感知应用的时效性,图 3-2中的 T_{total_time} 可以被认为是群智感知服务的生命周期。同时,$T_{dissemination}$ 代表了任务分发的周期;$T_{sensing}$ 代表了数据感知的周期;$T_{collection}$ 代表了数据收集的周期。相较于任务分发周期和数据收集周期,数据感知的时间周期非常小,所以重点考虑任务分发周期和数据收集周期,对于数据感知周期则忽略不计。

3.4 分析模型

本小节主要阐述任务分发过程和数据收集过程的理论分析方法。表 3.1 给出了本章中使用的符号定义。

表 3.1 『符号和定义』

符号	定义
N	移动设备的数量
N_f	边缘服务节点的数量
λ_n	移动设备之间的相遇概率
λ_f	边缘服务节点和移动设备的相遇概率
$I(t)$	t 时刻已经收到任务的移动设备的数量
$I'(t)$	t 时刻 $I(t)$ 的增量
$P_{rcv}(t)$	t 时刻边缘服务节点收到感知数据的概率

3.4.1 任务分发过程分析

在任务分发过程中,当边缘设备从边缘服务节点收到任务之后,在其移动过程中利用端到端通信对群智感知任务进行二次分发。这种基于端到端的任务分发过程和人群中传染病的扩散原理类似。因此可以借助 SIR(Susceptible Infective Removal)模型来描述边缘设备之间的任务分发过程。但是和传统 SIR 模型不同的是,边缘服务节点可以利用蜂窝网络直接将任务分发至移动设备。所以任务的分发速度也和边缘服务节点的数量 N_f 高度相关。同时,$I(t)$ 作为已经收到任务的移动设备的数量,其大小也会直接已经任务分发的速度。将这两种任务分发的途径结合考虑,任务分发的速率如式 (3.2)所示。其中,$I'(t)$ 是 $I(t)$ 的导数。

$$I'(t) = N_f\lambda_f(N - I(t)) + I(t)\lambda_n(N - I(t)) \tag{3.2}$$

上式是黎卡提(Riccati)微分方程的。考虑到初始条件 $I(0) = 0$,即在群智感知开始时刻($t = 0$)没有移动节点被感染,则时刻 t 已接收到任务的移动设备的数量可由

式(3.3)表示。

$$I(t)=\frac{N(e^{(\lambda_n N+\lambda_f N_f)t}-1)}{e^{(\lambda_n N+\lambda_f N_f)t}+\frac{\lambda_n N}{\lambda_f N_f}},\forall\,\lambda_n\in[0,1),\ \lambda_f N_f\in[0,1) \tag{3.3}$$

当移动节点接收到群智感知任务之后,根据任务内容执行感知操作并将感知数据存储在其本地缓存中,然后等待机会将数据传送回边缘服务节点。为了真实了解参与到群智感知任务的移动节点数量,式 (3.4) 中的 $C_T(t)$ 表示所有移动设备中已经收到群智感知任务的设备覆盖率。当 $C_T(t)=1$ 时,表示区域内所有的移动节点都参与了群智感知服务。

$$C_T(t)=\frac{I(t)}{N}=\frac{e^{(\lambda_n N+\lambda_f N_f)t}-1}{e^{(\lambda_n N+\lambda_f N_f)t}+\frac{\lambda_n N}{\lambda_f N_f}} \tag{3.4}$$

3.4.2 数据收集过程分析

已经收到任务的移动节点按照要求获取感知数据后,需要及时地将感知数据传送回边缘服务节点。而群智感知应用在其生命周期内,收集到的数据越多,数据覆盖范围越大,则群智感知服务能够达到的服务质量就越高。这也意味着在规定的时间范围内,边缘服务节点必须尽可能收集来自所有移动设备上的感知数据。因此,在时域上分析数据收集过程也是非常重要的工作。

和任务分发时数据流的途径一样,感知数据在收集过程中,移动设备也可以使用蜂窝网络或者端到端通信的方式将发送感知数据。在这一过程中,任何边缘服务节点都可以作为感知数据的目的地。在使用端到端通信时,移动设备可以相互交换感知数据,并将收到的感知数据进行打包处理并未下一次转发做好准备。一个感知数据包一旦到达任何一个边缘服务节点,则该组感知数据被认为已经收集完成。在时刻 t,一个数据包被边缘服务节点收到的概率记为 $P_{rcv}(t)$。在边缘网络中,数据包可以利用蜂窝网络或者端到端通信传输,而这两种传播方式都是相互独立的。因此,$P_{rcv}(t)$ 的计算

方法如式 (3.5)。

$$P_{rcv}(t) = 1 - P_{nD2D}(t)P_{nDirect}(t) \tag{3.5}$$

上式中, $P_{nD2D}(t)$ 和 $P_{nDirect}(t)$ 都可以利用式 (3.2) 推导得出。其中, $P_{nD2D}(t)$ 指的是数据包以端到端通信未送达的概率, 计算方法如式 (3.6); $P_{nDirect}(t)$ 指的是数据包以蜂窝网络通信未送达的概率, 计算方法如式 (3.7)。

$$P_{nD2D}(t) = (\frac{N - S(t)}{N})^{\lambda_f N_f t}, \forall\ \lambda_f N_f \in [0, 1) \tag{3.6}$$

$$P_{nDirect}(t) = (\frac{N}{N + N_f})^{S(t)-1} \tag{3.7}$$

在上式中, $S(t)$ 表示一组感知数据包在传输交换之后, 网络中该数据包的拷贝总数。其计算方法如式 (3.8)。

$$S(t) = \frac{Ne^{\lambda_n N t}}{e^{\lambda_n N t} + N - 1}, \forall\ \lambda_n \in [0, 1) \tag{3.8}$$

3.4.3 传感质量分析

群智感知的应用, 往往需要覆盖防范的地域范围。其传感数据所涵盖的地域范围越广, 则群智感知服务的普适性越强。例如, 一个十字路口的交通状况需要通过附近所有道路上的车辆数量来推理。为了准确计算群智感知应用的覆盖率, 群智感知应用的目标区域如图 3-5被切割成一组网格单元: $G = \{g_1, g_2, g_3, \ldots, g_m\}$。网格单元的大小意味着地域空间上的感知密度, 该参数由群智感知应用的开发者所决定。在分析传感质量是, 这里用 $L_{ij}(t)$ 表示在时刻 t 时, 移动设备 j 是否在网格 $g(i)$ 中。其定义如式 (3.9)表示。

$$L_{ij}(t) = \begin{cases} 1, \text{ mobile node } j \text{ is in } g(i) \text{ at time } t. \\ 0, \text{ otherwise.} \end{cases} \tag{3.9}$$

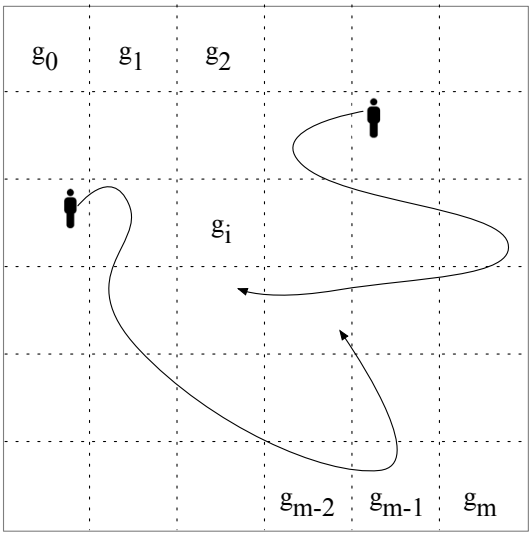


图 3-5 Spatial granularity in MCS

利用 $L_{ij}(t)$,移动设备 j 在网格 $g(i)$ 中的总停留时间 C_{ij} 可以用式 (3.10)表示。

$$C_{ij}(t) = \int_0^t L_{ij}(\varepsilon)d\varepsilon, \forall\, i \in [0, m], j \in [0, N] \tag{3.10}$$

因此,在 t 时刻,群智感知服务的感知质量 $C(t)$ 可以利用来式 (3.11)来计算。

$$C(t) = \sum_{i=0}^{m}\sum_{j=0}^{N} g_i \times C_{ij}(t), \forall\, t > 0 \tag{3.11}$$

3.5 Applications of the ODE-based Analysis

基于上一小节的模型,本节利用『stochastic analysis』的方法对边缘计算下的群智感知应用进行理论分析。由于群智感知应用对感知数据的时效性非常敏感,不具备时效性的感知数据可能会导致错误的计算结果。因此在使用机会式端到端通信交换感知数据时,必须考虑群智感知应用的截止时间限制。前文已经阐述群智感知应用中,足够多的参与设备和大量的反馈数据才能保证群智感知的时效性。所以在固定生命周期内,合理分配这两个过程的时间配额才能最大程度的提高群智感知应用的感知质量。

通过前文的分析,当任务分发过程的时间配额为 t 时,在周期 T_{total_time} 中群智感知应用可以收集到的感知数据总数 $D(t)$ 可以利用式 (3.12)计算。该式中,实际参与群

智感知应用的移动设备数量为 $N \times C_T(t)$。$T_{total\_time} - t$ 是数据收集阶段的时间配额，在该时间段类可以收到的『数据比例为 $C_F(T_{total\_time} - t)$』。

$$D(t) = N \cdot C_T(t) \cdot C_F(T_{total\_time} - t), \forall\, t \in [0, T_{total\_time}] \tag{3.12}$$

在式 (3.12)中，t 是唯一的自变量。因此，不同阶段的时间配额会直接影响到群智感知应用的感知质量。当 $t = 0$ 或 $t = t_{total\_time}$ 时，$D(t)$ 恒为零。且当 $0 \le t \le t_{total\_time}$ 时，$C_T(t) \ge 0$ 且 $C_T(t)$ 为单调递增函数；$C_F(T_{total\_time} - t) \ge 0$ 且 $C_F(T_{total\_time} - t)$ 为单调递减函数。因此，当 $0 \le t \le t_{total\_time}$ 时，$D(t) \ge 0$ 且对于 t 必定存在一个值使得 $D(t)$ 取到最大值，即群智感知应用的感知质量达到最优。为了找到最佳的时间划分，本文设计了一个枚举算法来找出 t 的最优解。该算法伪代码如『算法 3.1』所示。

算法 3.1: 找出让 $D(t)$ 最大的时间配额划分 t

Input: N : 移动设备的总数量
Input: N_f : 边缘服务节点的数量
Input: λ_n: 移动设备之间的相遇率
Input: λ_f: 边缘服务节点和移动设备之间的相遇率
Input: T_{total_time} : 群智感知服务的生命周期
Data:『输入数据』

```
let t = T_total_time / N_slots
let t_list = [t, 2t, 3t, 4t, ..., N_slots × t]
let res_list = [0] × N_slots
let index = 0; maximum = 0; best_t = 0
for item ∈ t_list do
    res_list[index] = D(item)    (Formula 3.12)
    if D(item) > maximum then
        maximum = D(item)
        best_t = t_list(index)
    index ++
```

Output: $maximum, best\_t$: $D(t)$ 的最大值，和对应 t 的值

在算法 3.1中，群智感知应用的生命周期被切分成 N_slots 份。通过对不同的时间配额进行枚举计算，求出让 $D(t)$ 取值最大的时间划分。该算法的时间复杂度为

$O(N\_slots)$。

3.6 系统测试

为了验证本文分析方法的正确性和准确度,本节使用模拟器和前文提出的理论分析方法进行对比。同时,本文的算法在寻找最优时间分配方面的效率也得到了证明。

3.6.1 模拟器设置

在测试过程中,本文使用 ONE 模拟器[45] 对边缘计算模型下的群智感知应用进行模拟仿真。在仿真实验中,主要的输入参数有: N、N_f、λ_n、λ_f,其具体含义如表 3.1所示。对仿真过程产生的日志信息进行处理,可以获得参与群智感知应用的移动设备数量以及每个数据包所经过的具体路径。结合时间戳信息进一步分析,还可以获得边缘服务节点在任意时刻收到的感知数据数量以及已收集的感知数据所覆盖的地域范围。

利用本文中的随机分析方法,可以利用参数 N、N_f、λ_n 和 λ_f 推导出同样的信息。将模拟仿真和随机分析得到的结果进行对比,可以验证本文中『随机分析』的正确性。在对比过程中,先分别验证任务分发过程和数据收集过程的覆盖率,在针对群智感知全过程,验证其范围覆盖和感知质量。

3.6.2 On the Accuracy of our Stochastic Analysis

首先通过使用不同的参数 N, N_f, λ_n, and λ_f 来验证本文对任务分发过程的分析。图 3-6反应了不同参数取值对群智感知效率的影响。图中横坐标为时间轴,纵坐标为群智感知应用收到的『感知数据的数量』,其结果可通过式 (3.4)计算得到。通过曲线对比仿真结果和模拟结果,两者的趋势是非常接近的。在 5000 组不同参数的实验对比中,任务分发分析的平均误差为 5.7%。对图像进行观察可以发现,群智感知应用执行初期,『』增长率偏低,这是因为任务只能通过边缘服务节点分发。在获得任务的移动设备增多之后,区域内的任务分发源也越来越多,因此能够执行感知任务的移动设备数量呈指数级上涨。当任务覆盖率接近饱和时,『』增长速度逐渐降低,获得任务的移动设备数量逐步趋近于目标区域内的移动设备总数。

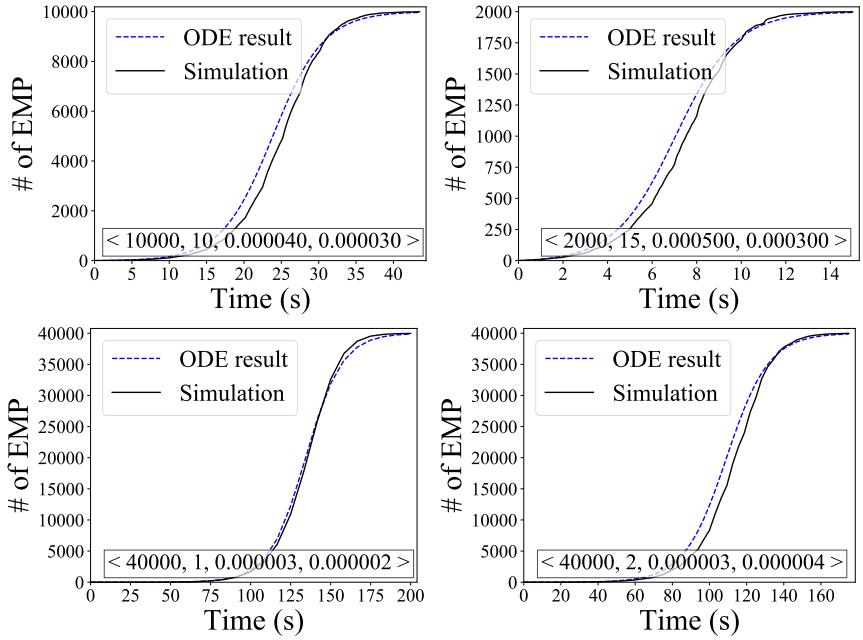


图 3-6 The accuracy of the analysis for the number of effective MCS participants

然后，用同样的方法来验证本文对数据收集过程的分析。假设每个获得感知任务的移动设备在收集到所需数据后，用一个数据包将结果反馈给边缘服务节点。此时，在目标区域内，已经部署有多个边缘服务节点。当数据包抵达任何一个边缘服务节点时，则可以认为数据反馈成功。利用式 (3-7)，可以计算被成功接收的数据包数量。图 3-8展示了模拟试验和理论分析的对比结果。图中，两者的变化趋势依然保持一致。在 5000 组不同参数的实验对比中，数据收集理论分析的平均误差为 9.6%。且变化趋势和实际情况相符。

最后，对边缘计算下的群智感知应用全过程进行分析验证。结合任务分发和数据收集两个重要阶段，验证本文中时间划分的合理性。在本次实验中，群智感知应用的生命周期 T_{total_time} 被设置为 100 秒。针对场景定义参数 N, N_f, λ_n, and λ_f 设计了四种不同的场景，分析群智感知应用可以收到的数据量和时间的关系。测试结果如图 3-8所示，理论测试值通过式 (3.12)计算得出。图示结果直观的反映出本文的理论分析犯法和模拟实验结果非常吻合，在任何时间分配方案下，分析结果总是接近模拟结果。这也验证了本文对任务分发和数据收集两个阶段的分析也是准确无误的。此外，在图中还能观察到的一个有趣现象：接收到的感知数据总量首先随着任务分发的时间配额的增加而增加，然后随着达到最大值而减少。最初，随着分配给任务分发阶段的

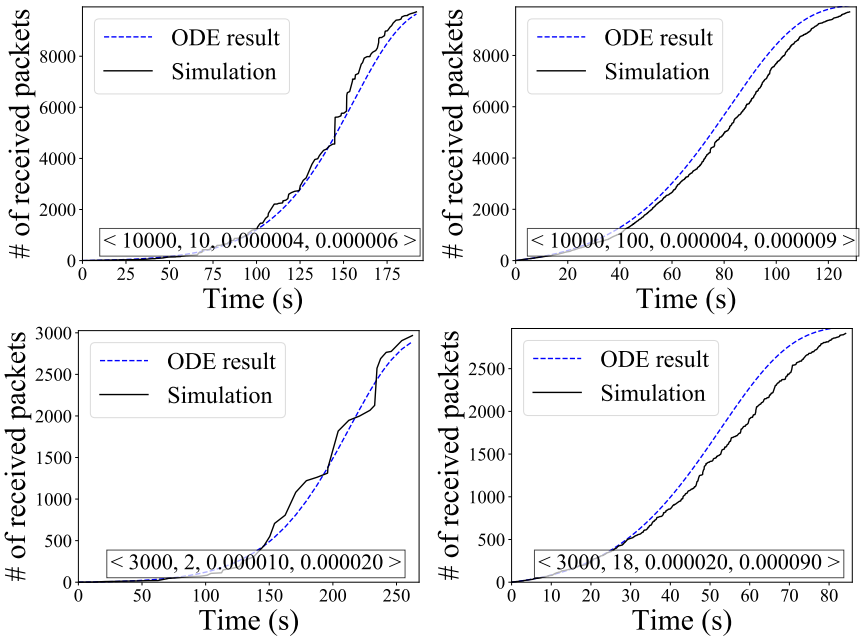


图 3-7 The accuracy of the analysis for the number of received packets

时间越来越多，越来越多的移动节点可以有效地参与 MCS 过程。因此，接收到的数据包数量会增加。达到最大值后，进一步的增加将减少分配给数据收集阶段的时间配额。而在固定的声明周期内，不仅需要任务顺利发送到移动设备，也需要感知数据能准确的传送回边缘服务节点。因此，当数据收集时间减少时，群智感知应用的感知质量会随之下降。这意味着存在一个最佳时间分配，可以最大限度地提高感知质量，找到这样一个最佳设置非常重要。幸运的是，我们提出的算法 1 能够处理这个问题。

为了验证时间配额划分算法的准确性，测试中使用参数 $N = 6000, N_f = 4, \lambda_n = 0.000027, \lambda_f = 0.00018$ 定义的模拟场景。群智感知应用的生命周期 T_{total_time} 设置为 100 秒。任务分发和数据收集的最佳时间分配，接收到的数据包数量应该是最大的。假设每个群智感知参与者需要必须返回一个数据包，则应该收集的数据包总数和群智感应参与者的数量应当相同，即为 N。因此可以使用被收集的数据包总数与群智感知参与者数量的比例来表示群智感知应用的效率。图 3-9展示了任务分发过程的时间配额分别为 25 秒、50 秒、75 秒以及 58.6 秒时的群智感知应用执行效率。其中，58.6 秒是利用时间配额划分算法找出的最佳任务分发时间配额。对比实验结果可以看出，最佳分配的结果比其他分配方法导致的执行效率要高出 20% 以上。在算法实验中，N_slots 被设置为 10^7，算法执行时间不超过 2500 毫秒。

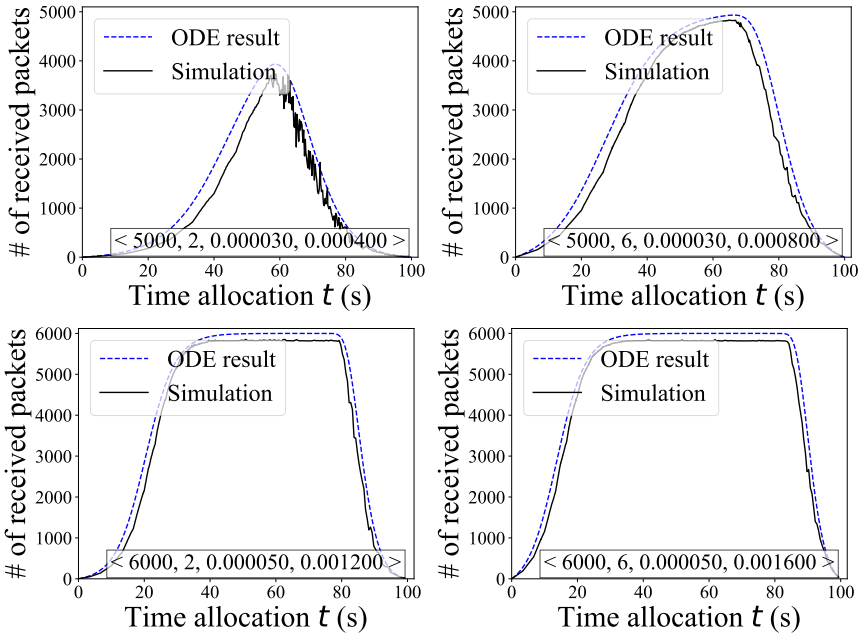


图 3-8 The accuracy of the analysis for the different time allocations

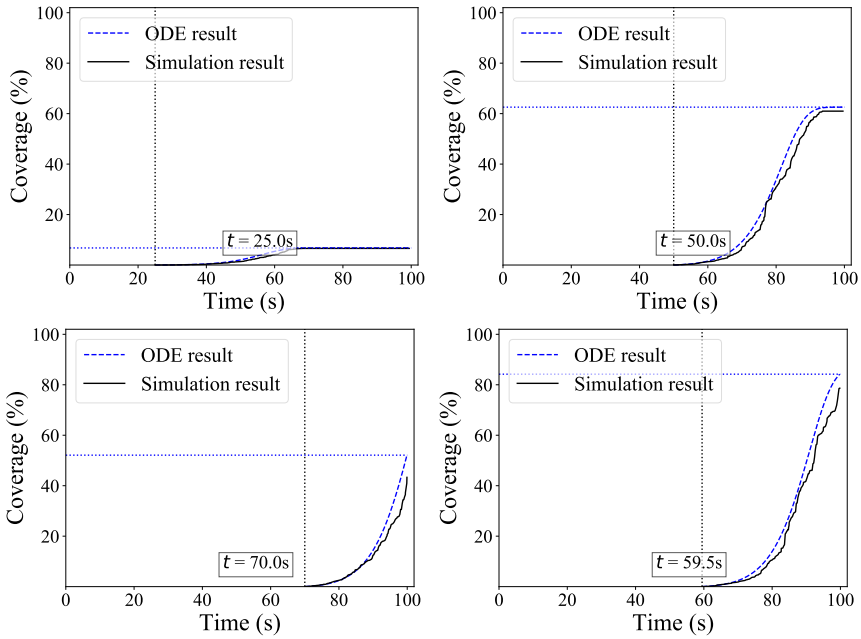


图 3-9 The results of the analysis and simulation for the best time allocation

3.6.3 The Coverage under Different Deployment Settings

该测试将时间配额划分算法应用到不同的网络场景下，获得在群智感知应用的生命周期内可以接受到的数据包总量。其中，基准群智感知的网络环境定义为 $N = 5000, N_f = 2, \lambda_n = 0.00003, \lambda_f = 0.0004$。在每组实验中，四个环境变量值只改变

其中一个，剩余三个环境变量保持不见。以此可观察出一种环境变量的变化对群智感知应用的感知质量的影响。也从侧面反映出边缘计算对群智感知应用的影响。测试结果如图 3-10所示。通过仿真数据和理论数据的对比，验证了本文的『Stochastic Analysis』方法准确性。同时，测试结果也展示了不同类型的边缘网络环境变量对群智感知应用的感知质量的影响效果。测试中，边缘服务节点的数量 N_f 以倍数关系增长，移动设备的数量 N、相遇率 λ_n 和 λ_f 均以 10% 等比例增长。对比发现，在『边缘计算下的群智感知应用』中，移动设备的数量 N 对感知效率的影响最大，其次是移动设备的相遇率 λ_n，然后是边缘服务节点的数量 N_f，最后是边缘服务节点和移动设备的相遇率 λ_f。

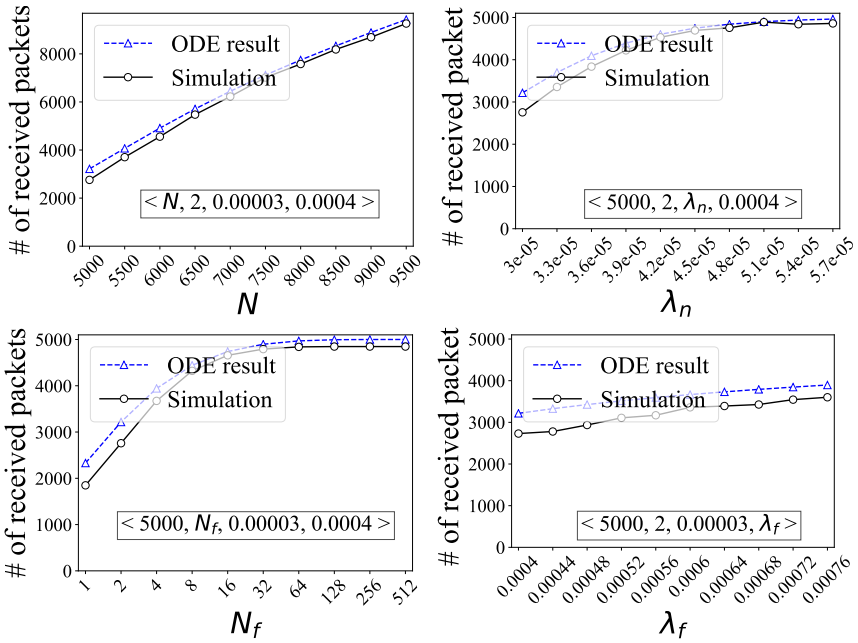


图 3-10 The number of packets received under different deployment settings

3.7 本章小结

本章节研究了一个实用的和潜在的用于具有 D2D 通信的 MCS 的雾计算应用，其中雾节点可以同时充当任务传播者和数据收集器。特别是，我们考虑机会性 D2D 通信来转发任务和传感数据。基于 D2D 的任务分发阶段和数据收集阶段都由 ODS 描述。通过求解这些方程，我们提出了有效 MCS 参与者和接收分组的封闭形式表达式，

作为雾节点数量、移动节点数量、移动节点之间的相遇率以及雾节点和移动节点之间的相遇率的函数。然后，我们通过共同探索任务传播阶段和数据收集阶段的分析，进一步得出可实现的感知质量。我们还运用我们的分析来找出两个主要阶段的最佳时间分配，从而最大限度地提高截止日期受限的 MCS 应用的感知质量。通过广泛的基于仿真的研究，我们验证了我们分析的正确性和准确性。与此同时，我们的随机分析总是能够揭示使用 D2D 通信的雾计算授权 MCS 的真实行为。

四 Multi-pathRouting for Energy Efficient Mobile Offloading in Software Defined Networks

在软件定义网络 (SDN) 中,规则空间是由三进制内容可寻址存储器 (TCAM) 引起的不可避免的限制。当负载变得更受欢迎时,负载数量的急剧增加会影响链路调度和客户的服务质量 (QoS)。因此,SDN 中的链路调度必须考虑规则空间、带宽和 QoS 的约束。为了优化移动计算中的能耗,我们构建了一个模型来理解负载决策、延迟和能耗之间的关系。我们实现了一个两阶段算法,通过使用交换机中的规则空间和链路中的带宽来调度链路。通过对比评估,验证了算法的可行性。当 SDN 网络中规则空间的使用不充分时,与最佳解决方案相比,该算法可以将超过 90 % 的能量效率存档。

4.1 研究背景

随着移动网络技术的发展,人们在日常生活中越来越离不开移动设备,如手机、平板电脑、笔记本电脑。手机可以帮助人们完成许多任务,比如导航、购物、拍照、分享等等。移动计算和云计算技术的进步[46,47] 允许移动设备更容易、更灵活地访问云中的各种资源。如今,用户可以使用各种移动设备将个人数据存储到云存储服务中,例如 Dropbox、OneDrive 或 Simple Storage Service(S3)。大多数云服务的目标是为客户有效地完成复杂的任务,并降低成本。为了灵活地从更多的云服务器中受益,一个名为“卸载”[48] 的解决方案通过将任务从本地设备迁移到云服务器来提供移动设备来增强能力。通过将计算交付给云,移动设备可以完成各种任务,如 GPS 导航、环境感知、监控、增强现实等。此外,能耗显著降低,因此移动设备的待机时间大大提高。因此,卸载在移动网络环境中被广泛使用。

网络中的设备数量和流量类型正在急剧增长。为了更有效地管理网络,新出现的网络架构 SDN 被发明出来。SDN 的目标是分离控制平面和数据平面[49]。SDN 采用可编程网络,并利用虚拟化技术,这与其他网络体系结构不同。同时,SDN 还可以共

享网络基础设施,使网络功能实现软件化。

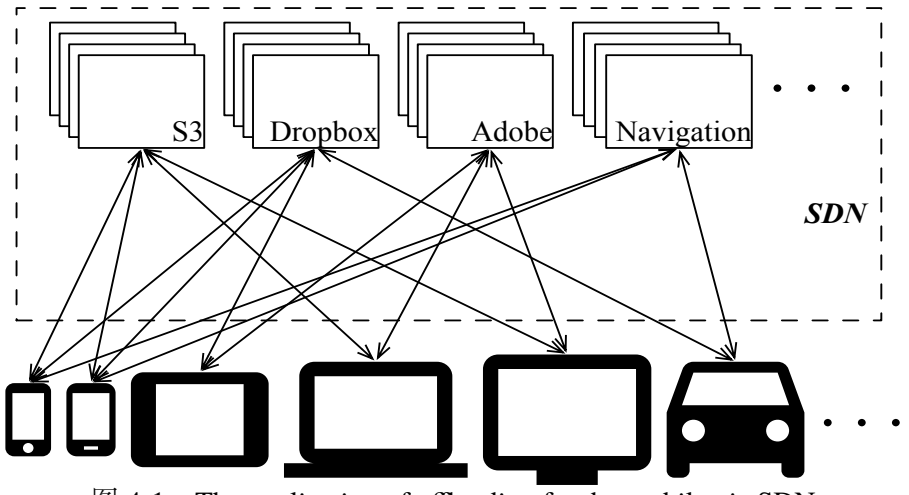


图 4-1 The application of offloading for the mobile via SDN

图 4-1显示了通过 SDN 卸载货物的应用程序,SDN 通常部署在数据中心。Lantz 等人[50] 为笔记本电脑中的 SDN 构建一个快速原型。SDN 还可用于中间件的有效政策执行[51]。在 APIs 的支持下,SDN 可以部署和管理统一的[52]。为了广泛提供 SDN,研究人员将重点放在 SDN [10] 的安全性和可扩展性上。2014 年,SDN 扩展到互联网,并创造了软件定义的互联网交换节点概念[53]。现在,SDN 用于在数据来自物联网[54] 的情况下管理服务功能链。

尽管 SDN 为管理网络提供了一种更方便、更灵活的解决方案,但仍存在一些缺点,特别是 TCAM 的限制。TCAM 是 SDN 交换机中昂贵且耗电的芯片,用于存储多路径路由规则。TCAM 的容量被命名为规则空间,它代表了可以放在流表[55] 中的转发规则的数量。随着网络规模的扩大,TCAM 花费大量的精力和金钱来增加 SDN 交换机中的规则空间。因此,规则空间限制成为现有 SDN 交换机的限制。通常,SDN 交换机传递的最大链路数在 10000 到 15000 之间[56]。

随着代码卸载的趋势,现有 SDN 在移动网络场景中面临新的挑战。有必要考虑规则空间、带宽、卸载率、能耗和成本之间的关系。然而,SDN 交换机的容量从未被视为受限资源。本文旨在提出一种新的链路调度算法来平衡 SDN 中的带宽和规则空间。最终目标是帮助用户最大限度地降低能耗,确保服务质量。据我们所知,这是第一次同时研究移动卸载决策和 SDN 中的约束条件。我们的主要贡献如下。

1)我们建立了一个网络模型来制定 SDN 中的限制和移动卸载中的能耗。

2)为了解决求解迭代学习算法的计算复杂性,我们提出了一种两阶段算法,通过大量基于仿真的研究验证了该算法的正确性和有效性。

4.2 System Model and Problem Statement

图 4-2显示了我们的模型所指向的整个场景。整个模型由三部分组成: 1)用户的手机,2)选择用户到云的链接的 SDN 加速器,3)云计算资源池。为了模拟通过 SDN 进行移动卸载的场景,我们在以下假设下设计了我们的系统: 1)每个用户的任务是独立的;2)所有交换机之间的链路都是稳定的;3)SDN 管理所有网络连接。

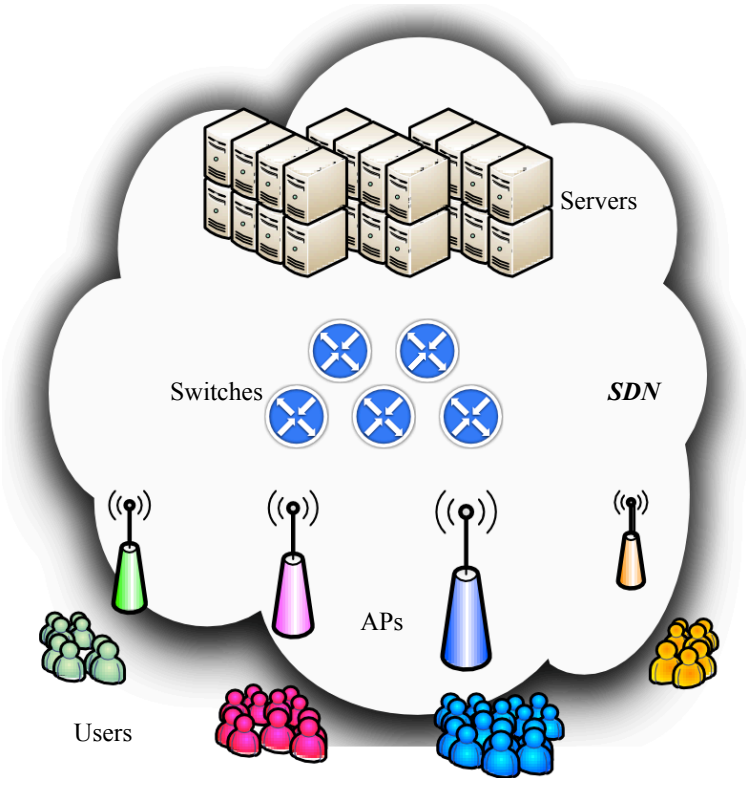


图 4-2 Offloading for mobile users

如图 4-2所示,每个用户通过固定 AP 访问网络资源。用户和云服务器之间的所有链接都由 SDN 分配。SDN 的主要职责是为所有用户安排链路,以提高他们的 QoS。

同时, SDN 的工作对用户是透明的。为了满足更多用户, 我们可以租用云计算资源, 如虚拟专用服务器(VPS)、Docker 等。你想要的资源越多, 你就需要付出越多。

在这项工作中, 我们使用集合 U 来代表所有用户。对于每个用户 u 来说, 属于集合 U ($u \in U$)。在云中, 有许多服务器可以为用户提供各种服务。让 s 表示数据中心中的服务器。这些服务器的集合被记录为 S ($s \in S$)。考虑到 SDN 支持的链接, l 代表一个链接, $L(l \in L)$ 代表所有链接。然后让 B 表示带宽。为了抽象云中的资源, 可以假设计算资源的计算能力为 μ。然后, μ_s 表示云中服务器 s 的计算能力, μ_u 表示用户 u 的本地设备的计算能力。用户 u 请求服务的频率记录为 λ_u。

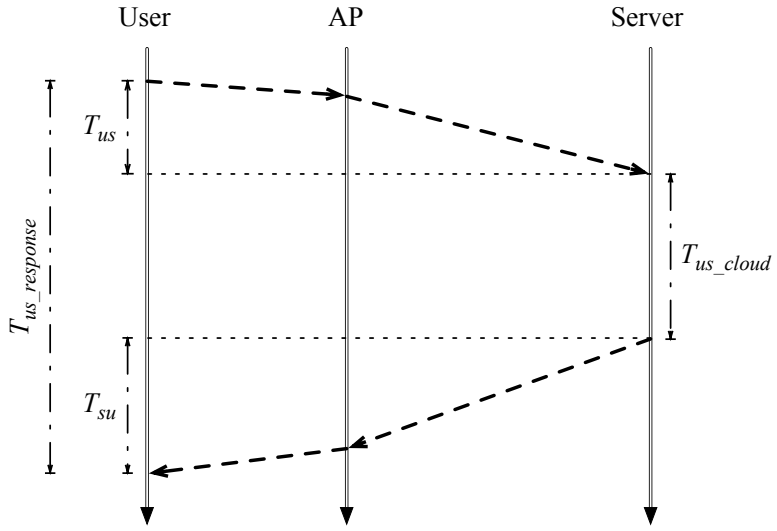


图 4-3 Timestamps taken in each step

图 4-3用于理解每个步骤中的时延。在图中, T_{us} 是从用户 u 到服务器 s 的时延。T_{su} 是从服务器 s 到用户 u 的延迟。T_{us_cloud} 是用户 u 在服务器 s 中的执行时间。$T_{us\_response}$ 是将任务从用户 u 卸载到服务器 s 的总时间。鉴于假设条件(2), 用户和服务器之间的时间成本是恒定的。同时, SDN 网络没有为用户指定 AP, 因为用户更喜欢最稳定的网络服务。由于用户和接入点之间的等待时间可以被视为常数, 因此不需要将等待时间分成两部分。然而, 当用户发起任务时, SDN 网络可以帮助用户选择不同的链接进行上传和下载。因此, T_{us} 和 T_{su} 适合用作用户和服务器之间延迟的参数。

$$T_{us\_response} = T_{us} + T_{su} + T_{us\_cloud},\ \forall\, u \in U, s \in S \tag{4.1}$$

在此模型中, 云服务的响应时间 $T_{us\_response}$ 可以通过这些延迟的总和来理算, 如

式 (4.1) 。按照假设条件(1),每个任务都可以在本地或云中执行。因此,我们需要一个二进制变量来指示一个任务是在云中执行还是在本地执行。对于用户 u,让 x_{us} 表示卸载到服务器 s 或是在本地执行的任务的卸载决策。基于上述系统模型,我们提出了一个排队模型来描述本文研究的问题。符号 α_{us}、x_{us} 和 α_{us} 之间的关系为式 (4.2),x_u 的定义为式 (4.3)。

$$\begin{gathered}\frac{\alpha_{us}}{A} \leq x_{us} \leq A \cdot \alpha_{us} \quad (A \to \infty) \\ \forall\, u \in U, s \in S,\ 0 \leq \alpha_{us} \leq 1\end{gathered} \tag{4.2}$$

$$\begin{gathered}\frac{1-\sum\limits_{s\in S}\alpha_{us}}{A} \leq x_u \leq A \cdot (1-\sum_{s\in S}\alpha_{us}) \quad (A \to \infty) \\ \forall\, u \in U, s \in S,\ 0 \leq \sum_{s\in S}\alpha_{us} \leq 1\end{gathered} \tag{4.3}$$

根据排队论原理[57],在本地执行的服务的时间 T_{u_local} 和在云中执行的服务的时间 T_{s_cloud} 的平均执行时间可以用 $M/M/1$ 模型通过式 (4.4)和式 (4.5)来测量。

$$T_{u\_local} = \frac{x_u}{\mu_u - (1-\sum\limits_{s\in S}\alpha_{us})\cdot\lambda_u}, \quad \forall\, u \in U \tag{4.4}$$

$$T_{s\_cloud} = \frac{x_{us}}{\mu_s - \sum\limits_{u\in U}\alpha_{us}\cdot\lambda_u}, \quad \forall\, s \in S \tag{4.5}$$

使用公式 (4.1)、(4.4)和(4.5),可以计算用户和云服务器之间的响应时间,以预测卸载决策。通过调整卸载决策和网络路径,可以影响用户的响应时间和能源成本。这项工作的目标是最大限度地降低用户的能源成本。

4.3 Problem Formulation

为了用数学方法更好地描述这个模型,我们在表 4.1中定义了这些符号。这些符号的细节将在后面解释。

表 4.1 Notations used in system model

Notation	Description
usl	The specific link from user u to server s ($usl \in L$)
L_{us}	The link set from user u to server s ($usl \in L_{us}$)
x_{usl}	The uplink decision of link usl
y_{usl}	The downlink decision of link usl
T_{usl}	The latency of the link usl
B_{usl}	The bandwidth of the link usl
B^u	The requirement of uplink bandwidth for user u
B^d	The requirement of downlink bandwidth for user u
x_r^{usl}	Whether the link usl passes through the switch r
e_u	The energy cost of execution locally for user u
e_s	The energy cost of execution on cloud server s

4.3.1 Constraints

如上所述, SDN 负责为用户选择最佳链接。用 usl 表示用户 u 和服务器 s 的链路, 用 L_{us} ($usl \in L_{us}$) 表示可以连接用户 u 和服务器 s 的链路集合。由于上行链路和下行链路可能是不同的链路, 我们使用二进制变量 x_{usl} 和 y_{usl} 来表示是否使用 usl 作为上行链路和下行链路。x_{usl} 和 y_{usl} 的值与公式 (4.6) 中的 x_{us} 相关。x_{usl} 和 y_{usl} 的值应该满足公式 (4.7)。

$$\begin{aligned} 0 \le x_{usl} \le x_{us}, \quad \forall\, usl \in L_{us}, u \in U, s \in S \\ 0 \le y_{usl} \le x_{us}, \quad \forall\, usl \in L_{us}, u \in U, s \in S \end{aligned} \tag{4.6}$$

$$\begin{aligned}\sum_{usl\in L_{us}} x_{usl} = 1, \quad \forall\, u \in U, s \in S \\ \sum_{usl\in L_{us}} y_{usl} = 1, \quad \forall\, u \in U, s \in S\end{aligned} \tag{4.7}$$

然后,我们使用 T_{usl} 来表示链路 usl 的延迟。借助 x_{usl} 和 y_{usl},可以通过式 (4.8)计算 T_{us} 和 T_{su} 的数值。

$$\begin{aligned}T_{us} = x_{usl} \cdot T_{usl}, \quad \forall\, usl \in L_{us}, u \in U, s \in S \\ T_{su} = y_{usl} \cdot T_{usl}, \quad \forall\, usl \in L_{us}, u \in U, s \in S\end{aligned} \tag{4.8}$$

QoS Constraints: 当用户请求服务时,该过程可以理解为用户提供输入并等待相应的输出。尽管任务可以在本地或云中执行,但是用户总是希望他们能够尽快得到反馈,对于任务在哪里执行,用户并不关心。所以,任务在调度时应尽可能让用户的等待时间减少。因此,平均服务时间是一个非常重要的服务质量指标。通过以上工作,我们可以利用式 4.9精确地计算平均服务时间。

$$\begin{aligned}T_{us\_response} = x_{usl}T_{usl} + \frac{x_{us}}{\mu_s - \sum\limits_{u\in U} \alpha_{us} \cdot \lambda_u} + y_{usl}T_{usl} \\ \forall\, usl \in L_{us}, u \in U, s \in S\end{aligned} \tag{4.9}$$

遵循代码卸载的原则,任务被分配给执行时间最短的执行环境。因此,用户 u 的平均执行时间必须满足式 (4.10)所示的限制条件。

$$\begin{aligned}x_{usl}T_{usl} + \frac{x_{us}}{\mu_s - \sum\limits_{u\in U} \alpha_{us} \cdot \lambda_u} + y_{usl}T_{usl} \leq T_{QoS} \\ \mu_s - \sum_{u\in U} \alpha_{us} \cdot \lambda_u > 0 \\ \forall\, usl \in L_{us}, u \in U, s \in S\end{aligned} \tag{4.10}$$

此时,任务在本地执行的响应时间应满足式 (4.11)所示的限制条件。

$$\begin{gathered}\frac{x_u}{\mu_u-(1-\sum\limits_{s\in S}\alpha_{us})\cdot\lambda_u}\leq T_{QoS}\\ \mu_u-(1-\sum_{s\in S}\alpha_{us})\cdot\lambda_u>0\\ \forall\, u\in U, s\in S\end{gathered}\tag{4.11}$$

Bandwidth Constraints: 与传统网络一样,SDN 中的每个链路都有自己的最大带宽。我们假设用户 u 通过链路 usl 与服务器 s 连接。因此,链路 usl 的带宽必须满足所有用户的需求。假设 usl 的最大带宽是 B_{usl},并且上行链路和下行链路的带宽需求分别是 B^u 和 B^d。那么约束条件可以描述为式 (4.12)。

$$\begin{gathered}\sum_{u\in U}\sum_{s\in S}\lambda_u\alpha_{us}(x_{usl}\cdot B^u+y_{usl}\cdot B^d)\leq B_{usl}\\ \forall\, usl\in L_{us}, u\in U, s\in S\end{gathered}\tag{4.12}$$

Rule Space Constraints: 除了带宽限制之外,SDN 中的交换机的可用规则容量也有一定的上限。规则空间表示交换机上允许的最大连接数。因此,SDN 交换机提供的最大连接数是有限的。我们用 r 来表示 SDN 交换机。$R(r\in R)$ 表示交换机的集合。然后将 C_r 表示为交换机 r 的规则空间。然而,交换机 r 只能承载部分链接。在此二进制变量 x_r^{usl} 表示链路 usl 是否通过交换机 r,其定义如式 (4.13)所示。

$$\begin{gathered}x_r^{usl}=\begin{cases}0, & if\ usl \text{ does not pass through } r\\ 1, & if\ usl \text{ passes through } r\end{cases}\\ \forall\, u\in U, s\in S, usl\in L_{us}\end{gathered}\tag{4.13}$$

因此,规则空间的约束条件可以利用 x_r^{usl} 来计算,其表达式如式 (4.14)。

$$\begin{gathered}\sum_{u\in U}\sum_{s\in S}\sum_{l\in L}(x_{usl}+y_{usl})\cdot x_r^{usl}\leq C_r\\ \forall\, u\in U, s\in S, usl\in L_{us}, r\in R\end{gathered}\tag{4.14}$$

4.3.2 ILP Formulation

如前文所述,本章建立模型的目标是最小化用户的成本。我们将本地执行的成本定义为 e_u,将云服务器中的执行成本定义为 e_s,因为任务可以在本地或远程执行。总能源成本 E 可通过式 (4.15)计算。

$$E = \sum_{u \in U}(\lambda_u(1 - \sum_{s \in S}\alpha_{us})e_u + \lambda_u \sum_{s \in S}\alpha_{us}e_s) \quad \forall\, u \in U, s \in S \tag{4.15}$$

整理上述讨论中的所有约束条件,可以将『能耗』转化为最大-最小公平问题,其描述如式 (4.16)所示。

$$\begin{aligned} &\min : E \\ &\text{s.t. : (6)-(7), (10)-(12), (14)} \end{aligned} \tag{4.16}$$

4.4 A Two-Phase Algorithm

为了使这项工作高效, 该算法主要负责两个阶段的工作。首先, 该算法确定 $\sum_{s \in S}\alpha_{us}$ 和 $\sum_{u \in U}\alpha_{us}$,然后创建一组首选链接。其次,该算法为用户 u 和服务器 s 分配最合适的链接,并调整 α_{us}。

Phase 1 (Algorithm 1): 在式 (4.15),λ_u,e_s,和 e_u 是常数。变量 α_{us} 是确定 E 值的唯一变量。为了降低总能耗, 必须在 e_s 和 e_u 之间调整系数。因此, 当 e_s 大于 e_u 时, $\sum_{s \in S}\alpha_{us}$ 中必须取最大值, 否则取最小值。为了优化上述模型的解决方案,最好的方法是找出 $\sum_{s \in S}\alpha_{us}$ 的取值范围。

利用约束条件(4.10)和(4.11), 我们可以得到 $\sum_{s \in S}\alpha_{us}$ 的范围和 $\sum_{u \in U}\alpha_{us}$ 的范围。

算法 4.1: Determine α_{us} and create a set of links

Input: λ_u :『解释说明』
Input: μ_u :『解释说明』
Input: μ_s:『解释说明』
Input: e_u:『解释说明』
Input: e_s :『解释说明』
Data:『输入数据』

```
if e_u ≤ e_s then
    for u ∈ U do
        Σ_{s∈S} α_us = max(0, 1 − μ_u/λ_u)
else
    for s ∈ S do
        Σ_{u∈U} α_us = min(1, μ_s/λ_u)
for u ∈ U do
    for s ∈ S do
        build L_us, a set with all links from u to s
        sort L_us ascending order by latency
α_s = Σ_{s∈S} α_us,  α_u = Σ_{u∈U} α_us
```

Output: $\alpha_s, \alpha_u, L_{us}$

$$\sum_{s \in S} \alpha_{us} \in [max(0, 1 - \frac{\mu_u}{\lambda_u}), 1], \quad \forall\, u \in U, s \in S$$

$$\sum_{u \in U} \alpha_{us} \in [0, \frac{\mu_s}{\lambda_u}], \quad \forall\, u \in U, s \in S$$

在确定 $\sum_{s \in S} \alpha_{us}$ 和 $\sum_{u \in U} \alpha_{us}$ 的范围后,可以根据约束条件找出一组从用户 u 到服务器 s 的链接。在这组链路中,存储了关于延迟、带宽和每个链路中交换机的信息。这些信息能够作为『算法二中』的输入,帮助用户 u 选择连接服务器 s 的最佳链路。

Phase 2 (Algorithm 2): 在『算法一』中,可以得到 $\sum_{s \in S} \alpha_{us}$ 的最优解。因此,链路调度算法负责找出所有 α_{us},这使得『总和』逼近最佳值。由于规则空间、带宽、延迟和 QoS 的限制,我们将贪婪算法和动态规划算法相结合来达到目标。

首先,我们按 λ_u 的升序对用户进行排序,并按 μ_s 的升序对服务器进行排序。这

算法 4.2: Choose the appropriate link for user $u \in U$

Input: α_s : from Algorithm 1
Input: α_u : from Algorithm 1
Input: $_{us}$: from Algorithm 1
Input: R:『解释说明』
Input: $rule\ space$ of $r(r \in R)$:『解释说明』
Data:『输入数据』

all weights initialized as 0 all α_{us} initialized as 0 U in ascending order of λ_u S in ascending order of μ_s **for** $u \in U$ **do**
  if $T_{u\_local} < T_{Qos}$ **then**
    $\alpha_{us} = 0, s \in S$
  else
    for $s \in S$ **do**
      if $1 \leq \frac{\lambda_u}{\mu_s}$ ***and*** $T_{s\_cloud} < T_{QoS}$ **then**
        for $l \in L_{us}$ **do**
          if $l.weight$ *is not exceed* 10% **then**
            $\alpha_{us} = 1$
            $x_{usl}[u, s, l] = y_{usl}[u, s, l] = 1$
            next u
      if $T_{s\_cloud} < T_{QoS}$ ***and*** $\alpha_u < 1$ **then**
        for $l \in L_{us}$ **do**
          if $l.weight$ *is not exceed* 10% **then**
            $\alpha_{us} = \frac{\lambda_u}{\mu_s}$
            $x_{usl}[u, s, l] = y_{usl}[u, s, l] = 1$

Output: x_{usl}

里的策略是优先从速度较慢的手机上卸载任务。一般来说,云服务器的成本和性能是正相关的。因此,云服务器的最佳选择是能耗最小且能满足 QoS 的服务器。然后,可以在第一阶段使用分类链接集(L_{us}),并遵循贪婪算法来选择用户 u 和服务器 s 之间最快的链接。但是,最快的链路不一定是最好的。『原因』。因此,我们使用动态规划算法来挑选最合适的链接。在这一步中,我们尝试收集关于规则空间使用和带宽使用的统计数据作为权重。

链路调度算法旨在使带宽使用率和链路使用率同时增长。在规则空间和带宽限制下保障用户的 QoS 而不产生额外开销是链路选择的重要依据。因此,本文的链路调度策略为每个链路设定权重,并且使链路的权重尽可能相近。所以,两种资源的占用率的权重比例相同。其中,所有链路的最大权重差值不能超过 10%。当某个链接的权重比其他链接的权重高出 10% 时,该链接会从可选择链接中屏蔽。这个策略是为了防止高 QoS 链接被快速占满。一旦高质量的链路和服务器被占用完全,其余用户必须降低卸载比率以确保 QoS,这将导致能耗成本的增加。

4.5 Performance Evaluation

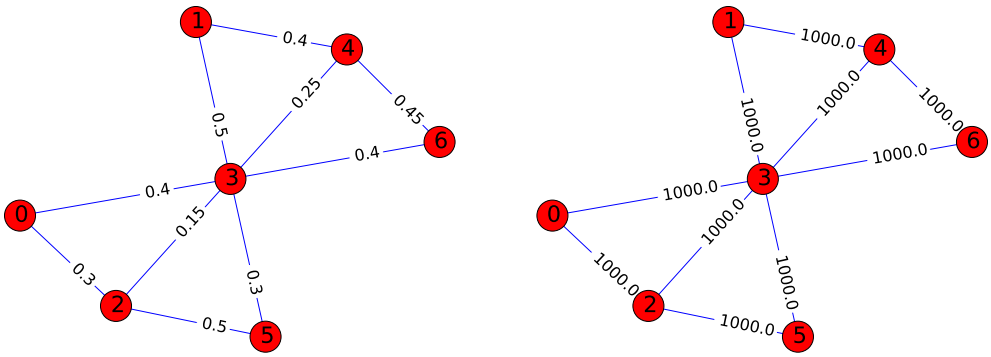
为了评估依据权重的链路调度算法(LSW)的性能,本文给出了与传统的带权重的链路调度算法(SPF)以及 Gurobi [1]的最优解进行性能比较。在实验中,使用了两种不同的 SDN 网络拓扑结构。第一个是一个小型网络,用于验证『算法』的可行性。第二个网络拓扑是一个有 100 个交换机的网络,用来比较传统的 SPF 方法和我们的 LSW 算法。

4.5.1 Verification of Feasibility

第一个小型网络的拓扑结构如图 4-4所示。在此图中,定义了每条链路的延迟和带宽。每个交换机的规则空间定义为 100,100,80,80,80,100,100。在网络中,节点 0 和节点 1 是接入点,节点 5 和节点 6 连接到不同的服务器。连接到节点 6 的服务器比连接到节点 5 的服务器快 10%。

链路调度算法 LSW 的目标是最小化总能耗 E。因此,我们比较了三种方法计算出的不同的 E 值。如图 4-5所示,不同的链路调度策略会影响卸载场景中的总能耗。当用户数量较少时,链路调度算法不会对能耗产生太大影响。随着用户数量的增加,能源成本之间的差距急剧扩大。当网络中有 100 个用户时,LSW 方法的总能耗比最优解(Gurobi 计算得出)高 12.6 %,比 SPF 方法的最优解高 36.4%。同时,100 名用户使 SDN 网络基本上处于满负荷状态。因此,当 SDN 网络的负载接近满负荷时,LSW

[1]https://www.gurobi.com



(a) Weight of latency (s) (b) Weight of bandwidth (MB)

图 4-4 The total energy consumption via different methods

方法此时存在局限性。从这个图中，当 SDN 网络负载不满时，LSW 方法和最优解之间的差异可以减少到小于 10%。

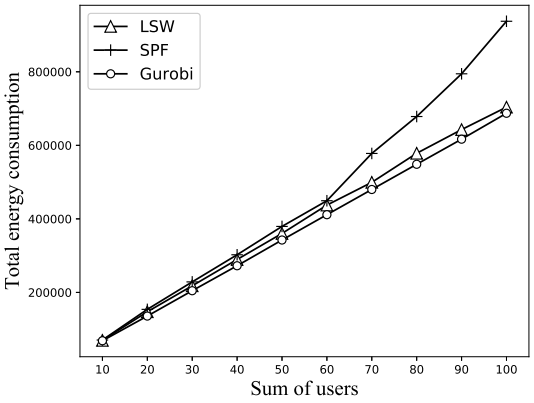


图 4-5 The total energy consumption via different methods

在图 4-6中，任务卸载的比率随着用户数量的增加而增加。通过前文的分析，可以得知卸载决策会影响能耗。然而，链路调度的策略会影响任务卸载的决策。在该图中，从侧面反映出更好的链路调度方法可以使得任务卸载到云端的平均比率更高。尤其是当 QoS 和 SDN 贷款受到限制时，没有足够的链路来来传输被卸载的任务。

图 4-7显示了每个交换机中规则空间的使用情况。对于最优解（Gurobi），不同用户总数和规则空间占用率并没有直接关系。这与 SPF 方法相同。在 LSW 算法的设计中，我们有意保持每条链路使用率的平衡性。因此，当 SDN 网络中的开销变得更大时，仍然留有高质量的链路预留给新用户使用。在 SPF 方法中，我们可以知道规则空间在交换机中的使用可能会发生剧烈变化。这意味着为了满足更多的用户，链路调度

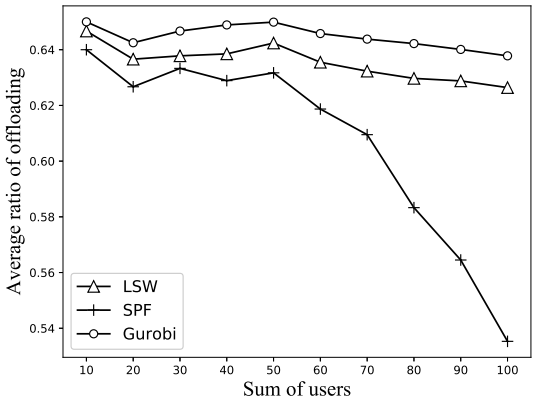
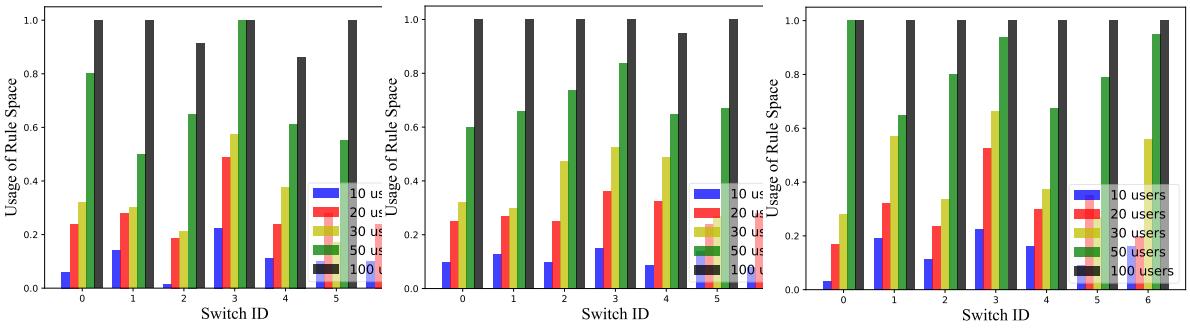


图 4-6 The average ratio of offloading via different methods

的次数会有很大的变化。这也表明 SDN 网络中的交换机可能会频繁地改变其状态，这将导致额外的能耗成本。



(a) Rule usage via Gurobi (b) Rule usage via LSW (c) Rule usage via SPF

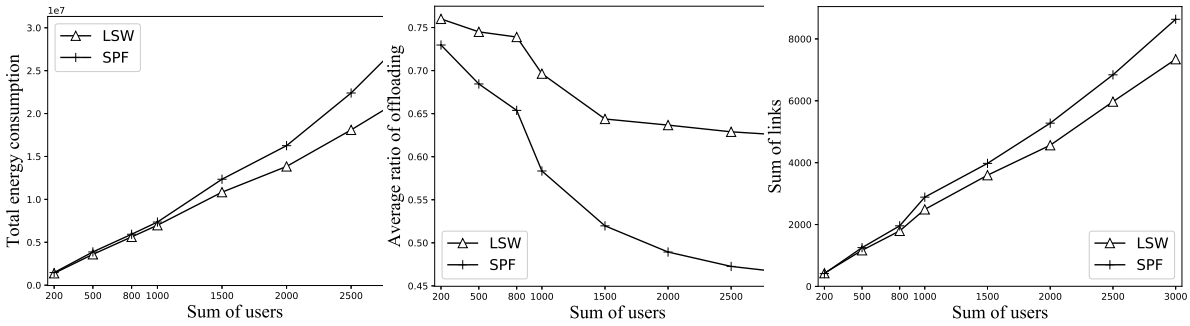
图 4-7 The rule usages in switches via different methods

4.5.2 性能测试

为了比较 LSW 算法和 SPF 算法，本文设计了一个拥有 100 个交换机的 SDN 网络。网络中有 10 个接入点和 40 台服务器，用户数量从 200 到 3000 浮动。这个网络中每个交换机的规则空间是 1000。在这个 SDN 网络中，有 30% 的链路不能满足 QoS，由此来模拟网络拓扑中的不稳定因素。

图 4-8(a) 显示了这两种方法的总能耗。通过 LSW 算法，总能耗和用户数量之间的关系几乎是线性的。然而，当有 3000 个用户时，SPF 算法得到的结果比 LSW 算法多 31.9%。图 4-8(b) 显示了两种方法中卸载决策的比率。当用户从 800 增加到 1500 时，这一数字急剧下降。原因是一些链路不能满足 QoS。这两种算法都受到这些链路

的影响, 但是 LSW 算法的抗干扰能力更好。图 4-8显示了这两种方法中使用的链接数。在同样的情况下, LSW 方法中的链路数比 SPF 算法中的链路数少 10%。这意味着 LSW 方法可以使用更少的链路来帮助降低总体能源成本。



(a) 能量总开销　　(b) 平均任务卸载率　　(c) 网络中链路使用数量

图 4-8　The performances via different methods

4.6　本章小结

本文建立了一个 SDN 网络公式模型, 以了解 SDN 中总能量成本和限制之间的关系。为了最小化总能耗, 我们将这个问题表述为线性规划问题。在对模型进行研究之后, 我们提出了一种链路调度算法, 根据每个链路的权重来选择最合适的链路。权重由交换机中规则空间的使用和链路中带宽的使用组成。通过大量实验, 我们证明了算法的有效性。通过对比实验, 该算法比 SPF 算法显示出显著的优势, 极大地提高了最优解的性能。

五　边缘计算中面向群智感知的服务资源资源调度

建议使用边缘计算来支持移动群体感应 (MCS) 应用程序进行传感数据处理。在这篇论文中, 我们考虑了公交乘客手机支持的 MCS 应用程序, 他们在装有边缘服务器的不同公交车站之间进行换乘。部署有相应 MCS 服务的边缘服务器可以通过设备间通信直接从参与者的传感器获取和处理传感数据。因此, 希望部署更多的 MCS 服务来探索更多的 D2D 通信, 而不会导致蜂窝通信成本。但是这与服务部署成本相矛盾。考虑到公共汽车乘客的流动性特征, 我们将在通信成本和服务部署成本之间寻求一个平衡, 以追求整体成本效益。我们首先将问题表达成混合整数线性规划模型, 然后设计一个低复杂度的启发式算法。实验结果验证了我们算法的高效率, 因为它能更接近最优解。

5.1　概述

移动群体感应 (MCS) [58,59] 已经成为一个吸引人的范例, 有助于天气预报[60]、空气质量监测[35] 和交通监测[37]。传统上, 这种应用是通过专门部署的传感器进行的, 具有很高的建造和维护成本。由于传感器丰富的智能设备的流行, MCS 利用无处不在的无线连接和众多智能设备的内置传感器来执行参与式和机会式感测。借助具有移动性的地理分布式智能设备, MCS 可以通过扩展传感覆盖区域与专用传感系统互补或甚至替代。

对于智能城市应用, MCS 应用通常需要覆盖整个城市的传感数据。这对于当前 MCS 架构在大量额外的移动网络流量以及相应的数据处理任务方面提出了新的挑战。针对这种现象, 边缘计算作为将云服务扩展到网络边缘的新型计算范例, 可能是 MCS 的一种有前景的候选计算资源提供手段, 如图 5-1所示。作为地理分布式计算范例, 边缘计算使计算、控制和存储更接近移动设备。同时, 数据传输延迟可以显著减少, 因为数据不再需要通过长距离完全上传到云服务器。这些特点引起了学术界的极

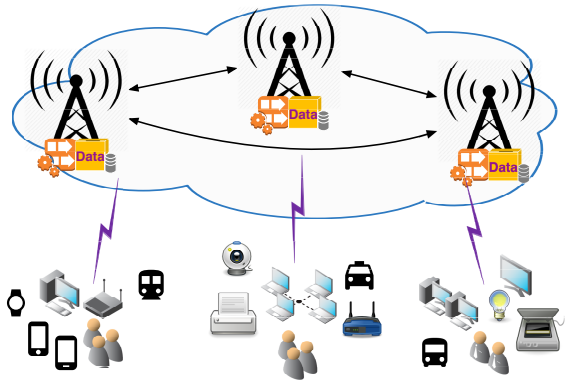


图 5-1 Architecture of edge computing empowered MCS

大兴趣。例如，唐等人。[61] 创建一个分层分布式边缘网络体系结构来执行数据表示和特征提取，并展示城市范围内服务提供的可行性。在论文[62] 中，作者发现，考虑到人类社会关系和设备对设备（D2D）通信，边缘计算可以提高网络运营的效率。詹诺夫·艾尔[63] 最近证明 D2D 通信在提高 MCS 效率方面是有效. 上述研究表明，随着 D2D 通信的引入，边缘计算对 MCS 应用产生积极影响。

对于面向智能城市的 MCS 应用来说，居民必须从他们配备有各种传感器的智能设备中提供传感数据。许多居民通常乘坐公共交通系统，例如公共汽车，在城市里旅行。一辆公共汽车将按照计划的路线参观多个不同的公共汽车站。最近许多研究提倡将公交车站作为部署边缘服务器以托管各种边缘服务的兴趣点（PoI），MCS 服务也不例外，用于感知数据采集和处理[63]。与此同时，由于公共汽车通常会在每个汽车站停一会儿，这为乘客通过 D2D 通信上传感测数据提供了一个很好的机会。这进一步意味着，位于每个公交车站的边缘服务器有望承载用于感测数据处理的 MCS 服务，并探索用于数据采集的 D2D 通信。

在本文中，我们考虑了一个延迟容忍但截止时间受限的 MCS 应用。也就是说，一旦应用程序被调用并招募了一些参与乘客，传感数据将在预定的期限内被完全处理，该期限通常不是短暂的。例如，我们对城市每小时的平均噪音水平感兴趣，可能会招募乘客通过智能手机感知噪音水平，并在一小时内将他们的感应数据报告给 MCS 服务。从 MCS 应用管理器的角度来看，降低运营成本总是有意义的，运营成本主要由两部分组成，即 MCS 服务部署成本和蜂窝通信成本。请注意，我们认为 D2D 通信是免费的合理假设。如果不能在预定期限内及时报告数据，蜂窝通信将用于报告数据，

这不可避免地会导致蜂窝通信成本。因此,人们可以尝试部署大量 MCS 服务来探索 D2D 通信机会。不幸的是,服务部署成本是另一个不可忽视的问题。这迫使我们寻求一种适当的解决方案,能够平衡蜂窝通信成本和服务部署成本,以最小化总成本。为此,除了在所有公交车站部署 MCS 服务外,我们还将在适当的公交车站部署适当数量的 MCS 服务。我们注意到,由于公共汽车的常规和人们的生活习惯,存在着一种移动模式。移动性模式可以使用公交车站网络来描述,在该网络中,乘客以不同的速率到达、离开、在不同的顶点 (即公交车站) 之间转移。通过各种数据分析手段可以很容易地获得这些统计数据。这正是我们可以用来寻求 MCS 服务放置解决方案的特性。本文的主要贡献总结如下:

- 据我们所知,我们是第一个针对基于公共汽车乘客的 MCS 应用研究成本-效益高的 MCS 服务布局问题。特别是,我们考虑了整个城市的乘客流动模式。

- 我们将问题正式表述为混合整数线性规划 (MILP),以提供对问题的深刻理解。因此,提出了一种低计算复杂度的启发式算法。

- 我们进行了广泛的基于仿真的研究来评估我们提出的启发式算法的性能。实验结果表明,该算法能够很好地逼近最优解,优于其他竞争对手。

本文的其余部分组织如下。第 2 节介绍了边缘服务放置的相关工作。我们在第三节中介绍了系统模型。在第四节中,我们给出了成本最小化问题的 MILP 公式,并在第五节中提出了我们的启发式算法。第 6 节报告了绩效评估结果。最后,第 7 节结束了这项工作。

5.2 系统模型

在这一部分,我们首先介绍了本文所研究问题的系统模型,主要包括公交乘客移动模型和基于边缘服务的拥挤数据处理模型。之后,我们陈述了本文要研究的问题。

5.2.1 背景和系统模型

在本文中,我们考虑了公交车乘客参与拥挤的情况,他们可能会被转移到城市的不同地方。在基于公共汽车乘客的人群中,乘客负责收集从家里到公共汽车站的道路

上的监控数据。我们不对传感数据采集施加任何限制。乘客可以通过从道路上专门部署的传感器收集感测数据或者通过嵌入智能手机中的传感器生成数据来获取数据。为了减轻基于基础设施的通信 (例如蜂窝通信) 的负担, 我们提倡参与者尽可能使用设备对设备 (D2D) 通信。因此, D2D 通信可能发生在乘客的智能手机和传感阶段专门部署的传感器之间。为了便于处理, 我们假设乘客在感知阶段可以获得相同的数据量, 用 s 表示。很容易将其扩展到不同乘客可以获取需要上传处理的大量数据的情况。

乘客在不同的公交车站上车, 形成乘客换乘图。因此, 我们使用有向图 $G =< V, E >$ 来描述乘客的流动性。顶点 $v \in V$ 代表乘客乘坐的公交车站。可以大致估计公交车站上的乘客到达率和离开率 $v \in V$, 分别表示为 λ_v 和 d_v。按照公交车习惯, 乘客将在不同的公交车站之间换乘。我们还可以获得不同公交车站之间换乘速度和时间的统计估计。从公交车站 $u \in V$ 到 $v \in V$ 的换乘速度和时间分别由 λ_{uv} 和 t_{uv} 表示。注意, 由于考虑了公交车线路规划和交通条件, 在实际中 t_{uv} 不一定等于 t_{vu}。

为了确保传感数据的及时性, 采集的传感数据必须及时上传和处理。通常, 众包应用程序具有一定的生命周期 T, 在此周期内, 必须完成整个过程, 即感测、通信和处理。乘客可以通过 D2D 通信将感测数据上传到部署有数据处理服务的公交车站, 或者通过蜂窝通信上传到云。我们假设在公交车站持续时间内, 所有感测数据都可以通过 D2D 通信成功上传。通过 D2D 通信不会产生通信费用, 在蜂窝通信的情况下, 每数据单元收取 c_c。尽管在公交车站启用数据处理时, 公交车站必须部署相应的边缘计算服务, 这与云中的服务相同。在网络边缘部署众包数据处理服务并不是免费的, 边缘服务每单位时间收费 c_v。

5.2.2 用于群体数据处理的边缘服务

由于大多数移动设备具有多种通信手段, 这些移动设备可以通过 D2D 无线通信技术 (例如蓝牙、WiFi) 或长途无线通信 (例如蜂窝网络) 与其他移动设备交换数据。利用 D2D 无线通信, 数据交换和能耗的成本大大低于长途无线通信。在公交车站部署边缘服务后, 乘客可以在这些车站等候时将传感数据发送给这些服务器。如果一些车站没有部署边缘服务, 则从这些车站出发的乘客可以在通过带有边缘服务的车站时

上传他们的感测数据。一般来说,来自一名乘客的这些感测数据量并不太大,当公交车停在有边缘服务的车站时,上传的时间就足够了。

另一方面,公交车站容纳了大量的人。这意味着部署在公交车站的边缘服务可以获取大量传感数据。这些站点是预处理传感数据的最佳场所,例如冗余消除、数据挖掘、数据学习。借助边缘服务,主干网和云服务的过载可以大大减少。与此同时,由于这些边缘服务在地理上更接近乘客,因此自然可以减少服务的延迟。但是,在所有公交车站部署边缘服务是不现实的。如何部署边缘服务是一个紧迫的挑战。

例如,武汉市有 2000 多个汽车站。很难想象在每个站点部署边缘服务的成本。考虑到每个站点的人员流量可能存在巨大的差异,部署边缘服务后每个站点产生的收入也会有很大的不同。因此,对于不同的站点来说,部署边缘服务的权重将取决于许多实际因素,如人员流量、位置、连通性等。

5.2.3 基于动机示例的问题陈述

让我们用图 2 5-2来解释成本效率高的 MCS 服务放置问题的动机。我们考虑有五个站点的情况,2100 人参与了交通,这表明有 2100 个数据单元需要上传和处理。对于每个车站,绿色箭头上的数字表示从车站上车的人数,蓝色箭头上的数字表示在这个车站乘坐公交车的人数。直观地说,我们将在连接最紧密的公交车站部署服务。例如,如果只部署一个边缘服务,最好的选择必须是站 1。该服务可以获取 1300 单位的传感数据,剩下 800 单位通过蜂窝通信。如果要部署两个边缘服务,最好的解决方案是在站点 2 和站点 3 上部署它们,并且可以通过 D2D 通信直接获取 1900 个传感数据单元。同时,我们注意到,我们可以在 5 号站、3 号站和 1 号站部署 3 项服务,使用 D2D 通信获取所有数据。然而,这些站点的排名并不是最高的连接度。我们将这种现象归因于这样一个事实,即乘客可能在应用寿命内通过多个不同的公交车站。仔细利用移动模式以最大限度地利用 D2D 通信,而不招致过多的服务部署成本,这一点至关重要。

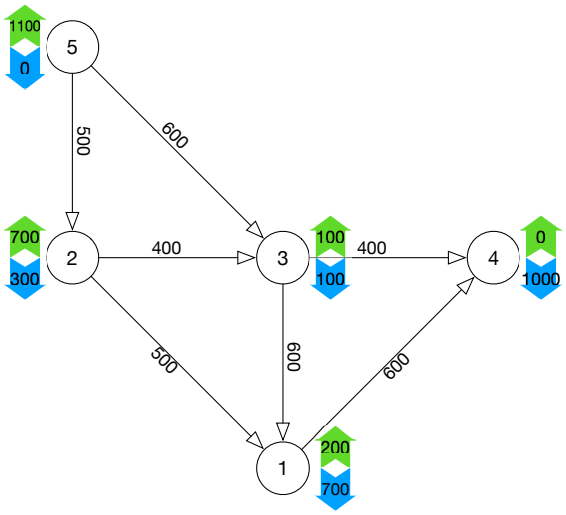


图 5-2 An example of traffic statistics

5.3 Problem Formulation

为了表示我们是否在公交车站 $v \in V$ 上提供边缘服务，我们定义了一个二进制变量为

$$x_v = \begin{cases} 1, \text{ the service is placed on } v \in V, \\ 0, \text{ otherwise.} \end{cases}$$

对于群体感知生命周期 T，边缘服务部署总成本 C_v 可以计算为

$$C_v = \sum_{v \in V} x_v c_v T \tag{5.1}$$

我们将从 u 站出发的乘客在 v 站的出发率定义为 d_v^u。不同于乘客的出发率，f_v^u 用于表示从 u 站出发的乘客在 v 站的数据出发率。对于每个乘客，他/她可以通过具有边缘服务的站，并通过 D2D 通信上传数据。因此，我们可以得到:

$$f_v^u \leq d_v^u s, \forall u, v \in V(u \neq v) \tag{5.2}$$

来自公交车站 u 的乘客，从公交车站 w 到公交车站 v 的到达率为 $\lambda\lambda_{wv}^u$。

来自公交车站 u 的乘客，从公共汽车站 w 上传到 v 的有效数据速率为 f_{wv}^u。显然，

我们有:

$$f_{wv}^{u} \leq \lambda_{wv}^{u} s, \forall u, v, w \in V \tag{5.3}$$

每当乘客经过部署有数据处理边缘服务的汽车站时,所有携带的传感数据都将通过 D2D 通信上传。一旦数据被上传,无需冗余上传。因此,来自公交车站 u 的乘客的有效数据率应满足

$$\sum_{x \in V} f_{xv}^{u}(1 - x_v) = f_{v}^{u} + \sum_{w \in V} f_{vw}^{u}, \forall u \in V, v \in V \tag{5.4}$$

如果公交车站 v 部署有数据处理边缘服务,即 $x_v = 1$,所有乘客携带的数据将通过 D2D 通信上传。

对于每个公交车站,有效数据率应满足

$$\lambda_u s(1 - x_u) = \sum_{w \in V} f_{uw}^{u}, \forall u \in V \tag{5.5}$$

显然,如果一个公交车站,比如 $u \in V$,部署了边缘服务,所有到达车站的乘客都应该直接上传他们的感应数据。在这种情况下,来自公交车站 u 的乘客的有效数据率应为 0

可以通过 D2D 通信上传的总数据可以计算为

$$\sum_{v \in V}(\sum_{u \in V} \sum_{x \in V} f_{xv}^{u} x_v + \lambda_v s x_v) \tag{5.6}$$

现在,我们可以将成本最小化问题表述为

$$\min : \sum_{v \in V} x_v c_v T + \sum_{u \in U} c_c(\lambda_u - \max_{v \in V} x_v \lambda'_{uv} s - \sum_{v \in V} x_v d_{uv} s)T \tag{5.7}$$

接下来,我们将上述优化问题线性化为混合整数线性规划 (MILP),该规划通过引入辅助变量 $f_u \leq \lambda'_{uv}, \forall u \in U, v \in V$ 形成。通过这样的定义,$f_u s$ 可以用来代替 $\max_{v \in V} x_v \lambda'_{uv} s$,以表示 D2D 可为首次到达公交车站 u 的乘客上传的最大数据速率。

因此,(8) 中的优化问题可以等效地转换成以下形式:

$$\begin{aligned}
\text{min}: \quad & \sum_{v\in V} x_v c_v T + \\
& \sum_{u\in V} c_c(\lambda_u s - \lambda_u s x_u - \sum_{x\in V}\sum_{v\in V} f_{xv}^u x_v))T \\
\text{s.t.:} \quad & f_v^u \le d_v^u s, \forall u, v \in V, u \ne v \\
& f_{vw}^u \le \lambda_{vw}^u s, \forall u, v, w \in V \\
& \lambda_u s(1 - x_u) = \sum_{w\in V} f_{uw}^u, \forall u \in V \\
& \sum_{x\in V} f_{xv}^u (1 - x_v) = f_v^u + \sum_{w\in V} f_{vw}^u, \forall u \in V, v \in V
\end{aligned} \tag{5.8}$$

5.4 算法设计

式 (5.8)是 MILP 形式,被广泛认为是 NP 难,因此不可能在多项式时间内解决。当问题规模较大时,使用通过求解式 (5.7)得到的解是不可行的,尽管我们可以通过各种 MILP 求解器获得最优解,例如 Gurobi。为了使该算法实用,我们将设计一种低计算复杂度的算法,该算法能够以可接受的计算复杂度逼近最优解。为此,我们设计了一个两阶段算法,介绍如下。

阶段 1 (算法 1) : 由于公共交通中的乘客数量非常大,不可能直接解决最佳边缘服务安置位置。因为公交路线是乘客的一种社交图,我们使用乘客的路径来确定边缘服务的位置。从交通统计数据中,我们可以得到每个乘客的轨迹。跟踪路径由乘客经过的车站的顺序列表表示。然后,通过交通统计数据可以找到具有相同轨迹的乘客。如果跟踪路径中有一个车站部署了边缘服务来获取数据,所有沿着该跟踪路径行驶的乘客都可以更新他们的传感数据

在算法 1 中,我们假设公共交通中的乘客总数是 n。乘客 i 的轨迹路径被表示为 p_i。然后,所有路径的集合是 P,并且 $P = \sum_{i=1}^{n} p_i$。因为一些乘客可能有相同的轨迹路径,因此可以找出独立的轨迹路径以及每个路径上的乘客数量。独立的跟踪路径在集合 IP 中存储为 $(count, p)$,$count$ 是沿着跟踪路径 p 行进的乘客数量。借助数据集

IP,如果在跟踪路径中有包含边缘服务的站点,则可以方便地计算跟踪路径中的数据采集。

另一方面,如果一个站已经部署了边缘服务,这个站的数据采集可以通过 IP 获得。这意味着可以计算站点的预期产量。剔除收入不足的公交车站是有帮助的,可以避免成本的浪费。

算法 5.1: Build independent paths set IP

Input: P :『解释说明』
Input: n :『解释说明』
Data:『输入数据』

```
for p ∈ P do
    if p ∉ IP then
        add (1,p) in IP
    else
        use p ∈ P to find (count, p) in IP
        count++
for s ∈ Stations do
    for p ∈ IP do
        if sinp then
            s.data += p.count
    if s.data > c_v/c_c then
        add s to ChoiceSet
```

Output: $ChoiceSet$

阶段 2 (算法 2) : 在算法 1 中,具有足够收入的一组站点被表示为 $ChoiceSet$。算法 2 是找出合理的站点组合,以最小的成本收集所有感测数据。由于具有边缘服务的站的总数有限,并且只能在集合 $ChoiceSet$ 中选择这些站,所以选择的站的组合被限制在小范围内。在 IP 的帮助下, 通过 $ChoiceSet$ 中选定站点获取的数据非常有效。在知道通过 D2D 通信上传的数据量后,我们可以得到不同放置解决方案的总体数据获取成本。最后,与最低成本相对应的站点组合是最佳的边缘服务放置解决方案。当选择多个站点部署边缘服务时,冗余数据也可以通过 IP 来计算。由于蜂窝通信的成

算法 5.2: Find out the best combination of stations

Input: IP :『解释说明』
Input: $ChoiceSet$:『解释说明』
Input: c_c:『解释说明』
Input: c_v:『解释说明』
Data:『输入数据』

```
for s_sum from 0 to min(ChoiceSet.size(), n × c_c/c_v) do
    for Stations with s_sum stations in ChoiceSet do
        Data = 0 for p ∈ IP do
            for ∀station ∈ p (station ∈ ChoiceSet) do
                Data = Data + p.count
        cost = s_sum * c_v + (n − Data) × c_c
        add (cost, Stations) in Res
```

Output: $\alpha_s, \alpha_u, L_{us}$

本被称为 c_c,在一个站部署边缘服务的成本被称为 c_v,具有边缘服务的站的数量不超过 $n \times c_c/c_v$。同时,在数据采集量小于 c_v/c_c 的站上部署边缘服务也是不值得的。这两个条件指定了具有边缘服务的站点数量的上限。

5.5 实验测试

为了验证问题公式和两阶段算法,我们在本节中报告了基于模拟的结果。此外,我们比较了不同边缘服务放置解决方案,以证明在智能城市应用中,两阶段算法是在公交车站部署边缘服务的最佳方法。

5.5.1 模拟器 BTSP

为了回放乘客等候、乘车、到达和上传数据的行为,我们建立了一个公交车模拟平台 (BTSP) 来模拟一个城市的交通行为。BTSP 的主要输入可分为两部分。

输入的第一部分是城市的公共汽车时刻表,第二部分是每个乘客的路线。从公共汽车时刻表中,我们可以得到所有公交汽车的过往路线、到达时间和出发间隔。此外,可以立即生成交通有向图。输入的第二部分是每个乘客的路线,其中包含所有乘客的

信息,每个乘客的起点站和终点站。对于每个乘客,都有几种可用的路径来满足乘客的需求。然后,我们按照可用路径构建一个乘车计划列表,然后按时间成本对列表进行排序。因此乘客可以优先选择时间成本最小的乘坐计划。

通过输入的两个部分数据,BTSP 知道一辆公共汽车是否停在车站。如果一辆公交车在车站,公交车上的乘客可以按照他们的乘车计划下车,车站的乘客可以决定是否上车。与此同时,如果这个站点已经部署了边缘服务,公交车上或站点上没有上传传感数据的乘客可以在公交车在站点停车时上传数据。总之,BTSP 可以实时完整地重放交通过程和数据上传过程。

5.5.2 The Cost under Different Placement Strategy

在 BTSP 中,我们按照实际公交时间表在武汉市试验了 30 条公交线路。BTSP 共有 273 个站点,241 个公共汽车在 60 分钟内从 7:00 到 8:00 运行。公交线路图如图 **??**所示,所有公交线路用绿色高亮显示。在这种情况下,我们假设众包应用程序有 60 分钟的特定寿命,边缘服务只能部署在公交车站。在车站部署边缘服务的成本是每小时 500($c_v = 500$)。通过蜂窝网络上传一段数据的成本是 1 ($c_c = 1$)。

遵循前文描述的系统模型,我们使用 Gurobi 找出边缘服务的最佳放置解决方案。与本文中的两阶段算法相比,我们构建了另外两种不同的布局策略。第一种算法是根据人员流动 (FoP) 选择站点来部署边缘服务。第二种算法是选择站点,在站点连接后部署边缘服务。由于交通的繁忙程度随着时间的推移而变化,我们在 BTSP 中使用不同数量的乘客进行模拟。

当乘客量设置为 5000、10000 和 50000 时, BTSP 中的结果如图 5-4、图 5-5和图 5-6所示。由于 c_c 设置为 1, 最大成本等于乘客人数。在图 5-4(a)、图 5-5(a) 和图 5-6(a) 中,结果表明,我们的两阶段算法可以找到一个更好的布局解决方案,以最小化成本,这类似于 Gurobi 找到的最佳解决方案。然而,FoP 和 CoS 放置策略的性能不令人满意,它们的部署成本比最佳放置策略高出 20%。比较 FoP 和 CoS 的放置策略,它们的成本非常接近。因此,按照人员流动或站点连接部署边缘服务不是节省成本的有效方法。

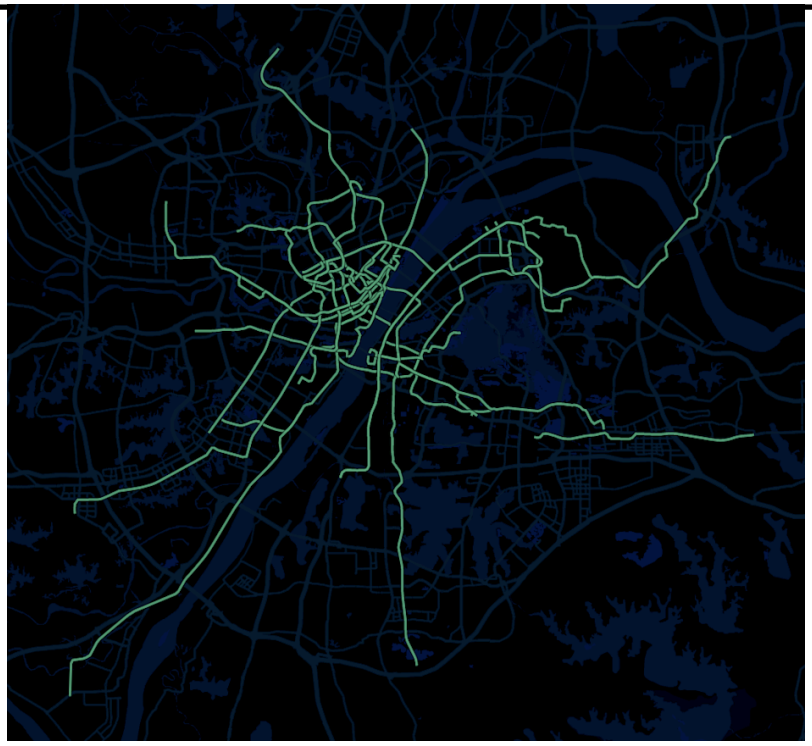
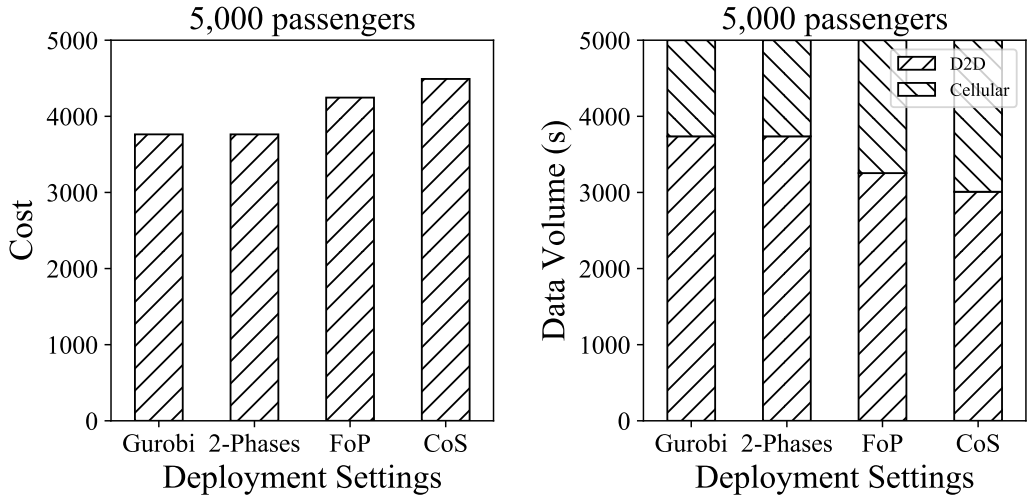


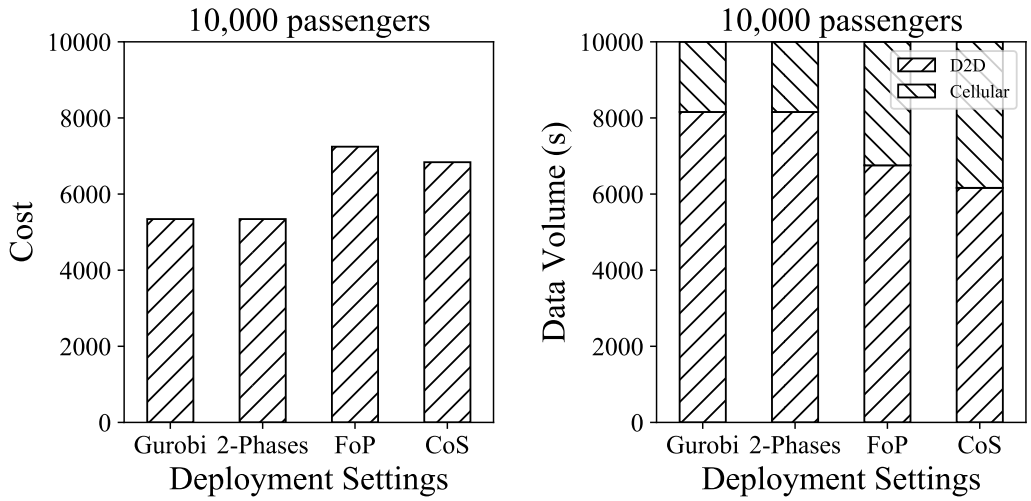
图 5-3　Bus lines in BSTP



(a) Cost　　(b) Data volume

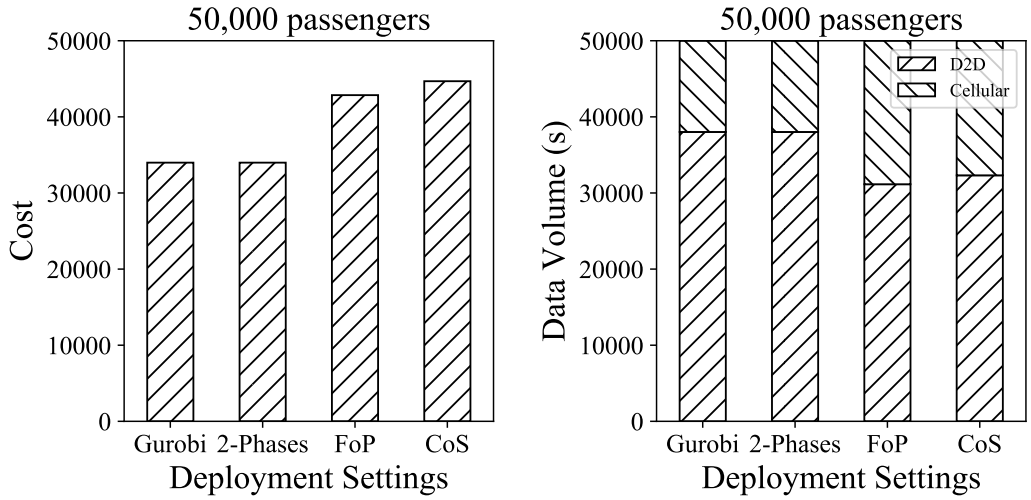
图 5-4　5000 passengers in BTSP

在数据采集中，每条数据都有固定的体积，等于 s。由于所有乘客都必须上传他们的传感数据，所以收集的数据总量是乘客人数乘以 s。如图 5-4(b)、图 5-5(b)、图 5-6(b) 所示，当成本较低时，D2D 通信上传的数据变得更多。当边缘服务按照 FoP 策略和



(a) Cost (b) Data volume

图 5-5 10000 passengers in BTSP



(a) Cost (b) Data volume

图 5-6 50000 passengers in BTSP

CoS 策略部署时，D2D 通信上传的数据显著减少。

在评估之前，我们已经告知 60 分钟内有 241 辆公共汽车在行驶。公共汽车的最大载客量设定为 50 人。在这个模拟场景中，5000 名乘客造成较轻的交通负载，10000 名乘客几乎交通满载，50000 名乘客造成交通过载。结果显示，当交通负载接近满时，

通过 D2D 通信上传的数据比例变得最大。相反,当交通负载太重或太轻时,通过 D2D 通信上传的数据量将变得更小。当交通负载较轻时,交通流量的减少会导致不经济的布局决策。当交通负载很重时,大量乘客将滞留在起点站或中转站。如果这个车站没有部署边缘服务,严重阻碍乘客通过 D2D 通信上传数据。

5.6 本章小结

在这篇论文中,我们研究了智慧城市的拥挤感知。通过在公交车站部署边缘服务,众包应用程序可以灵活地归档数据采集。由于城市交通构成了一个巨大的网络拓扑,人类的社交活动允许他们的智能设备在这个网络中流动。由于公共汽车将在公共汽车站短暂停留,它为公共汽车上的智能设备和车站的边缘服务之间的数据交互提供了机会。为了深入分析边缘服务的位置,我们建立了一个数学模型,并将问题转化为 MILP, MILP 的最优解可以通过现有的工具如 Gurobi 来计算。然后我们创建了一个公交车交通模拟平台,名为 BTSP。它用于重放和模拟交通行为。评估结果证明,我们的两个阶段能够以最低的成本找到最佳的放置解决方案。

六 总结

致 谢

感谢我的老师

参考文献

[1] Armbrust M, Fox A, Griffith R, et al. A View of Cloud Computing. Communications of the ACM, 2010, 53(4):50–58.

[2] Montresor A. Reflecting on the Past, Preparing for the Future: From Peer-to-Peer to Edge-Centric Computing. in: Proceedings of the IEEE International Conference on Distributed Computing Systems, 2016, 22–23.

[3] Geiger A, Lenz P, Urtasun R. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012, 3354–3361.

[4] Lai Z, Hu Y C, Cui Y, et al. Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices. in: Proceedings of the Annual International Conference on Mobile Computing and Networking, 2017, 409–421.

[5] Li Y, Gao W. MUVR: Supporting Multi-User Mobile Virtual Reality with Resource Constrained Edge Cloud. in: Proceedings of the IEEE/ACM Symposium on Edge Computing, 2018, 1–16.

[6] Kato S, Takeuchi E, Ishiguro Y, et al. An Open Approach to Autonomous Vehicles. IEEE Micro, 2015, 35(6):60–68.

[7] 施巍松, 孙辉, 曹杰, et al. 边缘计算: 万物互联时代新型计算模型. 计算机研究与发展, 2017, 54(05):907–924.

[8] 国家智能制造标准化总体组. 中国数据中心能效研究报告(白皮书), 2015.

[9] Hu Y C, Patel M, Sabella D, et al. Mobile Edge Computing—A Key Technology towards 5G. ETSI white paper, 2015, 11(11):1–16.

[10] Computing F. The Internet of Things: Extend the Cloud to Where the Things are. Cisco White Paper, 2015.

[11] Bonomi F. Connected Vehicles, the Internet of Things, and Fog Computing. in: Proceedings of the ACM International Workshop on Vehicular Inter-Networking, 2011, 13–15.

[12] Akyildiz I F, Su W, Sankarasubramaniam Y, et al. Wireless Sensor Networks: A Survey. Computer Networks, 2002, 38(4):393–422.

[13] 任彦, 张思东, 张宏科. 无线传感器网络中覆盖控制理论与算法. 软件学报, 2006, 03:422–433.

[14] 吴文乐, 郭斌, 於志文. 基于群智感知的城市噪声检测与时空规律分析. 计算机辅助设计与图形学学报, 2014, 26(04):638–643.

[15] Ali K, Al-Yaseen D, Ejaz A, et al. CrowdITS: Crowdsourcing in intelligent transportation systems. in: Proceedings of the IEEE Wireless Communications and Networking Conference, 2012, 3307–

3311.

[16] Dutta P, Aoki P M, Kumar N, et al. Common Sense: Participatory Urban Sensing Using A Network of Handheld Air Quality Monitors. in: Proceedings of the IEEE International Conference on Embedded Networked Sensor Systems, 2009, 349–350.

[17] Asl H Z, Iera A, Atzori L, et al. How Often Social Objects Meet Each Other? Analysis of the Properties of A Social Network of IoT Devices Based on Real Data. in: Proceedings of the IEEE Global Communications Conference, 2013, 2804–2809.

[18] Wang L, Zhang D, Wang Y, et al. Sparse Mobile Crowdsensing: Challenges and Opportunities. IEEE Communications Magazine, 2016, 54(7):161–167.

[19] 施巍松, 张星洲, 王一帆, et al. 边缘计算:现状与展望. 计算机研究与发展, 2019, 56(01):69–89.

[20] Ganti R K, Ye F, Lei H. Mobile Crowdsensing: Current State and Future Challenges. IEEE Communications Magazine, 2011, 49(11):32–39.

[21] Sun X, Ansari N. EdgeIoT: Mobile Edge Computing for the Internet of Things. IEEE Communications Magazine, 2016, 54(12):22–29.

[22] Zhang F, Jin B, Liu H, et al. Minimum-Cost Recruitment of Mobile Crowdsensing in Cellular Networks. in: Proceedings of the IEEE Global Communications Conference, 2016, 1-7.

[23] Xiao M, Wu J, Huang H, et al. Deadline-Sensitive User Recruitment for Probabilistically Collaborative Mobile Crowdsensing. in: Proceedings of the IEEE International Conference on Distributed Computing Systems, 2016, 721-722.

[24] Karaliopoulos M, Telelis O, Koutsopoulos I. User Recruitment for Mobile Crowdsensing over Opportunistic Networks. in: Proceedings of the IEEE Conference on Computer Communications, 2015, 2254-2262.

[25] Zhao D, Ma H, Tang S, et al. COUPON: A Cooperative Framework for Building Sensing Maps in Mobile Opportunistic Networks. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(2):392–402.

[26] Yi S, Qin Z, Li Q. Security and Privacy Issues of Fog Computing: A Survey. in: Proceedings of the IEEE International Conference on Wireless Algorithms Systems and Applications, 2015, 685–695.

[27] Sabt M, Achemlal M, Bouabdallah A. Trusted Execution Environment: What It is, and What It is Not. in: Proceedings of the IEEE International Conference on Trust Security and Privacy in Computing and Communications, 2015, 57–64.

[28] Ning Z, Zhang F, Shi W, et al. Position Paper: Challenges Towards Securing Hardware-assisted Execution Environments. in: Proceedings of the IEEE International Conference on Hardware and Architectural Support for Security and Privacy, 2017, 6:1–6:8.

[29] Hoekstra M, Lal R, Pappachan P, et al. Using Innovative Instructions to Create Trustworthy Software Solutions. in: Proceedings of the IEEE International Conference Workshop on Hardware and

Architectural Support for Security and Privacy, 2013, 11.

[30] Bonomi F, Milito R A, Zhu J, et al. Fog Computing and Its Role in the Internet of Things. in: Proceedings of the IEEE International Conference on Mobile Cloud Computing (MCC), 2012, 13-16.

[31] Marjanovic M, Antonic A, Zarko I P. Edge Computing Architecture for Mobile Crowdsensing. IEEE Access, 2018, 6:10662–10674.

[32] Chiang M, Zhang T. Fog and IoT: an Overview of Research Opportunities. IEEE Internet of Things Journal, 2016, 3(6):854–864.

[33] Guo B, Chen C, Zhang D, et al. Mobile Crowd Sensing and Computing: When Participatory Sensing Meets Participatory Social Media. IEEE Communications Magazine, 2016, 54(2):131–137.

[34] Rosen S, Lee S, Lee J, et al. MCnet: Crowdsourcing Wireless Performance Measurements through the Eyes of Mobile Devices. IEEE Communications Magazine, 2014, 52(10):86–91.

[35] Zhang D, Xiong H, Wang L, et al. Crowdrecruiter: Selecting Participants for Piggyback Crowdsensing under Probabilistic Coverage Constraint. in: Proceedings of the ACM Conference on Ubiquitous Computing, 2014, 703-714.

[36] Zheng Y, Liu T, Wang Y, et al. Diagnosing New York City's Noises with Ubiquitous Data. in: Proceedings of the ACM Conference on Ubiquitous Computing, 2014, 715-725.

[37] Zhou P, Jiang S, Li M. Urban Traffic Monitoring with the Help of Bus Riders. in: Proceedings of the IEEE International Conference on Distributed Computing Systems, 2015, 21–30.

[38] Antonic A, Marjanovic M, Pripuzic K, et al. A Mobile Crowd Sensing Ecosystem enabled by CUPUS: Cloud-based Publish/Subscribe Middleware for the Internet of Things. Future Generation Computing System, 2016, 56:607–622.

[39] Messaoud R B, Rejiba Z, Ghamri-Doudane Y. An Energy-Aware End-to-End Crowdsensing Platform: Sensarena. in: Proceedings of the IEEE Annual Consumer Communications & Networking Conference, 2016, 284-285.

[40] Wang Y, Li H, Li T. Participant Selection for Data Collection through Device-To-Device Communications in Mobile Sensing. Personal and Ubiquitous Computing, 2017, 21(1):31–41.

[41] Qin S, Feng G. Performance Modeling of Network Coding based Epidemic Routing in DTNs. in: Proceedings of the IEEE Wireless Communications and Networking Conference, 2013, 2057-2062.

[42] Li Y, Wang W. Message Dissemination in Intermittently Connected D2D Communication Networks. IEEE Transactions on Wireless Communications, 2014, 13(7):3978–3990.

[43] Zhao D, Ma H, Li Q, et al. A Unified Delay Analysis Framework for Opportunistic Data Collection. Wireless Networks, 2018, 24(4):1313–1325.

[44] Groenevelt R, Nain P, Koole G. The Message Delay in Mobile Ad Hoc Networks. Performance Evaluation, 2005, 62(4):210–228.

[45] Keränen A, Ott J, Kärkkäinen T. The ONE Simulator for DTN Protocol Evaluation. in: Proceedings of the International Conference on Simulation Tools and Techniques for Communications, 2009, 55:1-55:10.

[46] Lee K, Lee J, Yi Y, et al. Mobile Data Offloading: How Much can WiFi Deliver? IEEE/ACM Transactions on Networking, 2013, 21(2):536–550.

[47] Linthicum D S. Cloud Computing Changes Data Integration Forever: What's Needed Right Now. IEEE Cloud Computing, 2017, 4(3):50–53.

[48] Kumar K, Liu J, Lu Y H, et al. A Survey of Computation Offloading for Mobile Systems. Mobile Networks and Applications, 2013, 18(1):129–140.

[49] Open Networking Foundation. Software-defined networking: The new norm for networks. 2012.

[50] Lantz B, Heller B, McKeown N. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. in: Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networks, 2010, 19:1-19:6.

[51] Qazi Z A, Tu C C, Chiang L, et al. SIMPLE-fying Middlebox Policy Enforcement Using SDN. in: Proceedings of the ACM Conference on SIGCOMM, 2013, 27-38.

[52] Ferguson A D, Guha A, Liang C, et al. Participatory Networking: An API for Application Control of SDNs. in: Proceedings of the ACM Conference on SIGCOMM, 2013, 327-338.

[53] Gupta A, Vanbever L, Shahbaz M, et al. SDX: A Software Defined Internet Exchange. in: Proceedings of the ACM Conference on SIGCOMM, 2014, 551-562.

[54] Morabito R, Beijar N. A Framework Based on SDN and Containers for Dynamic Service Chains on IoT Gateways. in: Proceedings of the ACM Workshop on Hot Topics in Container Networking and Networked Systems, 2017, 42-47.

[55] Dasgupta M, Biswas G P. Design of Multi-path Data Routing Algorithm Based on Network Reliability. Computers & Electrical Engineering, 2012, 38(6):1433–1443.

[56] Katta N, Alipourfard O, Rexford J, et al. Infinite CacheFlow in Software-Defined Networks. in: Proceedings of the ACM Workshop on Hot Topics in Software Defined Networking, 2014, 175-180.

[57] Kleinrock L. Queueing systems, volume 2: Computer applications. John Wiley & Sons, 1976.

[58] Ganti R K, Ye F, Lei H. Mobile Crowdsensing: Current State and Future Challenges. IEEE Communications Magazine, 2011, 49(11):32–39.

[59] Guo B, Wang Z, Yu Z, et al. Mobile Crowd Sensing and Computing: The Review of an Emerging Human-Powered Sensing Paradigm. ACM Computing Surveys, 2015, 48(1):1–31.

[60] Yu X, Zhao H, Zhang L, et al. Cooperative Sensing and Compression in Vehicular Sensor Networks for Urban Monitoring. in: Proceedings of the IEEE International Conference on Communications, 2010, 1-5.

[61] Tang B, Chen Z, Hefferman G, et al. Incorporating Intelligence in Fog Computing for Big Data

Analysis in Smart Cities. IEEE Trans. Industrial Informatics, 2017, 13(5):2140–2150.

[62] Yan J, Wu D, Wang H, et al. Social D2D Communications Based on Fog Computing for IoT Applications. in: Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications, 2017, 314-325.

[63] Zhan Y, Xia Y, Zhang J, et al. Incentive Mechanism Design in Mobile Opportunistic Data Collection with Time Sensitivity. IEEE J. Internet of Things, 2018, 5(1):246–256.

附录 A 攻读学位期间发表的学术论文

[1] Liao X, Li H, Jin H, **Hou H**, Jiang Y, Liu H. VMStore: Distributed storage system for multiple virtual machines. SCIENCE CHINA Information Sciences, 2011, 54(6):1104-1118. DOI 10.1007/s11432-011-4273-0

[2] **Hou H**, Jin H, Liao X, Zeng D. Multi-path Routing for Energy Efficient Mobile Offloading in Software Defined Networks. in Proceedings of The IEEE International Symposium on Parallel and Distributed Processing with Applications, 2017, pp. 360-367. DOI 10.1109/ISPA/IUCC.2017.00058

[3] **Hou H**, Jin H, Liao X, Zeng D. Stochastic Analysis on Fog Computing Empowered Mobile Crowdsensing with D2D Communications. in Proceedings of The IEEE International Conference on Ubiquitous Intelligence and Computing, 2018, pp. 656-663. DOI 10.1109/SmartWorld.2018.00131

[4] **Hou H**, Jin H, Liao X, Zeng D. Cost Efficient Edge Service Placement for Crowdsensing via Bus Passengers.