计算机科学
COMPUTER SCIENCE

边缘计算中面向数据流的实时任务调度算法

张翀宇，陈彦明，李炜

引用本文

张翀宇，陈彦明，李炜. 边缘计算中面向数据流的实时任务调度算法[J]. 计算机科学, 2022, 49(7): 263-270.

ZHANG Chong-yu, CHEN Yan-ming, LI Wei. Task Offloading Online Algorithm for Data Stream Edge Computing [J]. Computer Science, 2022, 49(7): 263-270.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210300195

边缘计算中面向数据流的实时任务调度算法

张翀宇　陈彦明　李　炜
安徽大学计算机科学与技术学院　合肥 230601
(1403932783@qq.com)

摘　要　近年来,随着物联网(Internet of Things,IoT)技术的发展,其应用场景呈爆炸式增长,这类应用一般具有时延敏感性和资源受限性。如何在有限的资源环境下实现任务的实时分配是当前的一个研究热点,而将这些有限的计算资源动态分配给实时任务,一般来说是一个 NP-hard 的组合优化问题。为解决此问题,设计了一种基于李雅普诺夫优化的实时调度算法,在保持虚拟队列稳定的情况下优化长期平均总能耗和总效用。首先在计算资源和通信资源约束下建立联合总能耗和加权总效用的优化模型,该模型包含两层虚拟缓冲队列,通过端到端(Device-to-Device,D2D)的调度方式进行任务卸载;然后基于李雅普诺夫优化,将长期平均总能耗和总效用的联合优化问题转化为一系列实时优化问题,为此还设计了一种基于贪心的设备匹配算法。数值实验的结果显示,该算法的效果比随机法所能达到的最好情况提升了 8.6%,并且在不同连接概率下其效果逼近穷举法。

关键词:物联网;计算卸载;李雅普诺夫优化;贪心算法
中图法分类号　TP393

Task Offloading Online Algorithm for Data Stream Edge Computing

ZHANG Chong-yu,CHEN Yan-ming and LI Wei
School of Computer Science and Technology,Anhui University,Hefei 230601,China

Abstract　With the development of Internet of Things (IoT) technology,its application scenarios have exploded recently,and such applications are generally delay-sensitive and resource-constrained. It is a focused issue in the way of offloading the real-time tasks under the condition of limited resource. Besides,it is a NP-hard combinatorial optimization problem to allocate limited computational resources for the real-time tasks. To solve this problem,this paper proposes a real-time resource management algorithm based on Lyapunov optimization,aiming at stabilizing the virtual queues while optimizing the total power consumption and total utility. Firstly,the optimization model for the total power consumption and weighted total utility is proposed under the constraint of computation and communication resources. This model contains of two virtual buffer queues,and tasks are unloaded in a device-to-device (D2D) scheduling model. Then,an optimization algorithm is proposed based on Lyapunov optimization to decompose the joint long-term average sum energy consumption and sum utility optimization problem into a series of real-time optimization problems. To solve these problems,a greedy-based matching algorithm is proposed. Experimental results demonstrate that the performance of the proposed algorithm is 8.6% better than the best result of random method and can approximate the exhaustive attack method under different connection degrees.

Keywords　Internet of Things,Computation offloading,Lyapunov approximation,Greedy algorithm

1　引言

近年来,随着芯片技术和信息通信技术的快速发展,IoT设备及对应的应用逐渐涌现出来[1]。由于这些应用的高实时性,在处理该类任务时,设备端和处理节点之间应是高带宽和低干扰的。IoT设备不定期随机触发并生成时延敏感和计算密集的数据流任务,虽然有些设备具有一定的计算能力,但远远不能满足任务的需求,通常情况下需要将产生的数据传输到云端去计算。尽管云端的计算资源丰富,但由于传输过程的高延时性,因此不适合处理实时的数据流任务。

而边缘计算[2]和任务卸载技术[3]的提出给上述问题带来了新的解决思路。边缘计算将云计算的计算能力和存储能力从网络核心推到网络边缘,与移动设备和用户紧密相连,其可以解决计算密集型应用和资源有限的移动设备之间的矛盾,

到稿日期:2021-03-18　返修日期:2021-07-09
基金项目:国家自然科学基金(61802001)
This work was supported by the National Natural Science Foundation of China(61802001).
通信作者:陈彦明(cym@ahu.edu.cn)

将无法在设备端处理的任务卸载到边缘计算节点进行处理，以有效地减少堆积在设备端的任务缓存，从而达到降低时延的目的。然而，在物联网中将海量数据传输至移动边缘计算服务器的过程中存在一定延时，并且伴随着巨大的传输能量消耗，仍然不能有效解决实时且密集的计算类问题。例如，在智慧城市[4]场景中，任务设备执行特征提取，在边缘服务器中进行危险事件识别，由于需要运行人脸识别算法、目标跟踪算法以及预测算法等，因此任务对计算资源和时延的要求较高，边缘服务器往往不能满足这些需求。一种可行的解决办法是通过云-边-端[5-7]的联合处理并回传结果。然而在三端传输过程中往往时延较长，不能保证实时回传处理信息，因此本文考虑更靠近设备端资源的综合利用，以期解决实时性问题。上述工作忽视了对任务的操作集的优化。在实时系统中，数据流任务的结果往往需要经过多次迭代才能得到，每次迭代产生的结果逐步逼近最终结果，并且每一步产生一定的效用价值。当系统处在高负载的情况下，系统应当舍弃一部分操作来降低计算的复杂度，虽然不如最终结果精确，但一定程度上满足了计算的需求。

在 IoT 环境中，由于在设备空闲时仍然留有大量的计算资源未被利用，因此本文设计了一种新型的实时调度边缘资源池的模型，它可以在边端设备之间进行直接的资源调用，将无法处理的数据流缓存在任务设备的两层虚拟缓存队列，通过端到端的方式将任务传输给资源较为丰富的设备。另外，由于实时的负载变化，调度器可以根据当前缓存状态，通过对任务操作集的优化，来控制输入到缓存的负载。

在时延和能耗方面，相比传统的蜂窝传输方式，D2D 连接方式是复用蜂窝 D2D、WIFI 直连和蓝牙等，它们具有较短的传输时延和能耗。因此，本文利用这种实时调度边缘资源池模型可以使计算资源更加靠近任务设备，在处理实时性任务时具有很大的优势。然而，在实时调度边缘资源池的模型中，数据流任务的调度并非易事，一是在没有任何先验信息下决策需要调度的数据流任务比较困难，二是网络波动会影响设备端的传输能耗。

综上所述，本文做出了以下几点贡献：1)在稳定两层任务缓存队列的情况下，构造最小化长期平均总能量消耗和最大化加权总效用的联合问题，为此提出了实时调度算法，对任务操作集和 D2D 卸载策略进行了优化；2)应用李雅普诺夫优化方法，将长期问题分解为每个时段内的一系列小问题。在此基础上，采用基于贪心的设备匹配算法求解每个时槽内的近似最优解。

2 相关工作

以往研究证明了任务卸载确实可以减少能量的消耗和缩短时延[8-9]。虽然任务卸载对设备的性能提升具有决定性的作用，但由于无线信道存在不稳定性，有效的传输方式对卸载效率有着重要影响。为了解决传输过程中的干扰问题，Chen 等提出利用博弈论来集中式优化卸载获益的移动设备的数量[10]；Mao 等利用李雅普诺夫优化进行动态资源管理，并对计算服务器接入的移动设备的网络资源和系统长期总能耗进行联合优化[11]，但未考虑到设备与设备之间的资源利用。然而，文献[10,12-14]提出的框架或策略，都是在边缘计算服务器或数据中心进行优化，如果设备的通信受阻，算法将会放弃设备中的任务卸载进程。因此，依靠边缘服务器的任务卸载不是一个理想的解决方案。

早期的 IOT 数据流处理平台主要用于采集和显示实时的原始传感器测量数据，这些数据常常需要经过几个处理阶段才能形成可操作的自动或人工决策[15]。现阶段的 IOT 设备具有一定的计算资源，并且可以通过 D2D 连接和其他设备建立流数据传输信道，进行数据流任务卸载[16]。Zhang 等[17]提出了一种联合优化传输速率选择和功率控制的算法，该算法最小化中继辅助 D2D 链路和传统蜂窝链路的总传输能耗，以有效地利用通信链路的通信资源。在数据流传输的解决方案中，Baccarelli 等[18]提出并开发了一个在 TCP/IP SaaS 雾计算中基于李雅普诺夫优化的动态调度器框架，该框架利用一组队列来增加运行在虚拟机上的允许工作负载。其提出的算法优化了最大化数据中心的平均工作负载和最小化平均总能耗，但是动态调度器框架是存在于数据中心的，在 IOT 设备端上建立工作负载会增长由等待队列带来的时延。近年来，为了实现实时数据流任务的计算卸载，Liu 等[19]提出了一种实时调度边缘资源池的模型。在该模型中，位于网络边缘的设备利用 D2D 协作来实现计算资源的池化和共享。本文模型与文献[19]提出的模型有一些相似之处，同样是将部分设备端的资源看作一个资源池，但是考虑了设备与设备之间数据流的传输能耗。

本文设计了一种边缘计算中面向数据流任务的实时调度算法，该算法将在一定范围内具有良好通信的设备或者基站当作调度节点，通过该节点控制其一跳或多跳范围内的所有设备，并将那些长期处在空闲状态节点的集合看作一个资源池。调度节点内运行基于李雅普诺夫优化的实时调度算法，调度当前资源紧缺设备中的数据流至资源丰富的设备端。

3 实时数据流调度模型

如图 1 所示，在 IoT 环境中存在着 $g \triangleq \{1,2,\cdots,G\}$ 个设备，选择其中一个设备或者边缘服务器作为一个集中式调度器，用于控制这 g 个设备之间的通信。各设备之间通过蜂窝 D2D(或 WIFI 直连/蓝牙/Zigbee)互连，并形成复杂的网络拓扑结构。本文将时间分割为时间帧，帧序号 $t \in \{0,1,2,\cdots,T\}$，每个时间帧之间的时槽长为 ε。

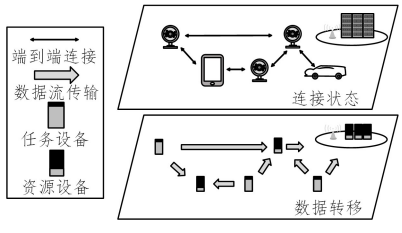

图 1 IoT 实例

Fig. 1 Illustration of IoT

为了有效地缓存任务,本文在设备端设置两层虚拟任务缓存队列,如图 2 所示。

(1)数据流准许输入层(Streaming Admission Input Layer,SAIL),任务设备直接执行缓存在该层的任务,并且传输合适大小的任务到任务积压层(Task Backlog Layer,TBL),其主要作用是充当积压层的缓冲层。

(2)任务积压层。该层负责接受来自数据流准许输入层的数据流任务,其主要作用是将数据流传输到合适的资源设备。

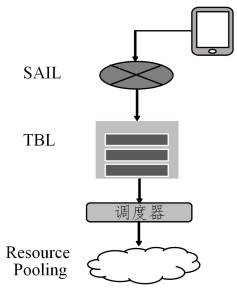

图 2 系统模型

Fig. 2 System model

根据长期资源占用率,将这 g 个设备划分成 N 个任务设备(长期处于资源紧缺状态的设备)和 M 个资源设备(长期处于资源丰富状态的设备)。假设任务设备 $i\in N$ 与资源设备 $j\in M$ 之间存在连接,记为 $E_{ij}=1$。那么,与设备 i 存在连接的设备集合为 $\Delta_i=\{j\mid E_{ij}=1,j\in M\}$,同理与设备 j 存在连接的设备集合为 $\gamma_j=\{i\mid E_{ij}=1,i\in N\}$。任务设备 i 自身的处理能力为 c_i(CPU 转数每秒),资源设备 j 上长期空闲的资源量记作 f_j(CPU 转数每秒)。

在实时数据流任务调度系统中,数据流任务的处理需要多次迭代完成,每次迭代的结果都会使计算结果更加精确,并且对系统会产生一定的执行效用,处理密度越高,任务的效用就越大。在处理原数据的过程中,设备端可以选择不同原数据的清晰度,以适应当前的系统负载状态,其分辨率越高,任务设备收集的数据量就越多,任务的效用也就越大。本文使用 u 表示原任务的操作集,包括迭代次数和选择的视频分辨率,u 取自于操作集的集合 ϑ,$u\in\vartheta$,另外,$w_{i,u}(t)$ 表示以操作集 u 执行在设备 i 上的数据流任务的效用值。在时槽 t 开始时,设备 i 随机生成随机数据流任务,$b_{i,u}(t)$(CPU 转数)为以操作集 u 处理该数据流所需要的资源量,$b_{\max}$ 为数据流任务所需的资源量的上限。如图 3 所示,操作集 u 的取值为 1~5,其所需资源量依次增加。任务所需的资源量越多,其处理效用就越高。

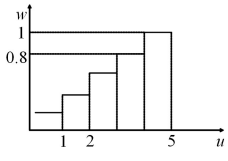

图 3 效用和操作集的离散关系

Fig. 3 Discrete relationship between utility and opration set

对于 TBL,它负责缓存数据流任务,并且发送数据流给资源设备。在 t 时槽内该层收到来自 SAIL 的数据流任务,并且通过调度器控制,传输合适大小的数据流任务至资源设备。在 TBL 中的数据流任务可以表示成三元组 $\langle S_i(t),A_i(t),\rho_i(t)\rangle$。当调度器选择调度缓存在任务设备上的数据流任务时,设备端会收集任务所需的数据流,并将其传输至资源设备 j 上,其中 $K_{ij}(t)$ 为任务设备 i 到资源设备 j 的传输速率,$S_i(t)$/(bits/s)为在 t 时槽内任务设备 i 的数据收集速率。根据数据流处理的并行性和实时性,TBL 中被调度的任务所需的资源量越多,设备端收集的数据量就越大,即 $K_{ij}(t)\geqslant S_i(t)$。$A_i(t)$为在 t 时槽内对应的处理密度(CPU 转数每比特)。以往的工作已经证实[19],对于单位数据量的不同任务需要不同的 CPU 转数。$\rho_i(t)$ 为数据流任务输出数据量和输入数据量的比值。数据流任务在任意资源设备执行时,都会产生一定量的输出数据流,大小为数据流任务的输入数据量乘以 $\rho_i(t)$,并且 $0<\rho_i(t)<1$。

在 t 时槽开始时,任务设备 i 会先以 $c_i*\varepsilon$ 的能力处理该数据流任务。由于物理容量的限制,当设备自身的能力不足以处理该数据流任务时,即 $c_i*\varepsilon<b_{i,u}(t)$,设备端需要的资源量大小为 $b_{i,u}(t)-c_i*\varepsilon$,设备 i 将需要协助的部分数据流任务缓存在 SAIL 中。当设备 i 不需要其他设备协助时,即 $c_i*\varepsilon>b_{i,u}(t)$,则设备端会处理缓存在 SAIL 中的数据流,大小为 $c_i*\varepsilon-b_{i,u}(t)$。在 t 时槽开始时,SAIL 的缓存队列长记为 $Z_i(t)$,其大小是缓存在 SAIL 数据流中的所需资源量。在 t 时槽内,SAIL 将部分数据流任务传输到 TBL,传输的任务所需资源量大小为 $r_i(t)$。不失一般性,开始时队列长为 0,即 $Z_i(0)=0$。SAIL 的缓存队列长 $Z_i(t)$ 遵循以下公式:

$$Z_i(t+1)=\max\{Z_i(t)+b_{i,u}(t)-c_i*\varepsilon-r_i(t),0\} \tag{1}$$

$$\bar{b}_i-c_i*\varepsilon\leqslant\bar{r}_i\leqslant\bar{b}_i \tag{2}$$

$$0\leqslant r_i(t)\leqslant b_{i,u}(t)\leqslant b_{\max} \tag{3}$$

其中,$\bar{b}_i\triangleq\lim\limits_{T\to\infty}\dfrac{1}{T}\sum\limits_{t=0}^{T-1}E\{b_{i,u}(t)\}$ 为平均时间期望输入负载,$\bar{r}_i\triangleq\lim\limits_{T\to\infty}\dfrac{1}{T}\sum\limits_{t=0}^{T-1}E\{r_i(t)\}$ 为平均时间传输到 TBL 的期望负载。那么,$\bar{b}_i-c_i*\varepsilon$ 为输入到 SAIL 的期望负载。式(2)中,$\bar{r}_i\leqslant\bar{b}_i$ 成立是由于 $r_i(t)$ 取自输入到 SAIL 的部分数据量,平均时间传输到 TBL 的期望负载小于平均时间期望输入负载,即 $\bar{r}_i\leqslant\bar{b}_i$。

任务积压层负责接受来自 SAIL 的数据流任务,并将数据流任务传输到资源设备。通过建立虚拟队列来统计在时槽 t 开始时 TBL 上数据流任务的资源占用量,记为 $Q_i(t)$。调度部分数据流任务,并传输对应的数据流给与设备 i 互连的资源设备集 Δ_i。$y_{ij}(t)$ 为任务设备 i 调度资源设备 j 的计算资源量,大小不超过当前时槽可以调度的资源量 $Q_i(t)$,$y_{ij}(t)=K_{ij}(t)A_i(t)x_{ij}(t)\varepsilon$。其中,$x_{ij}(t)$ 为决策变量,指在 t 时槽内任务设备 i 是否调度资源设备 j 的资源。若 $x_{ij}(t)=1$,则在 t 时槽内任务设备 i 调度了资源设备 j 的资源,否则 $x_{ij}(t)=0$。那么,设备 i 在时槽 t 开始时的队列长 $Q_i(t)$ 遵循以下公式:

$$Q_i(t+1)=Q_i(t)-\sum_{j\in\Delta_i}y_{ij}(t)+r_i(t) \tag{4}$$

不失一般性，队列长开始时为 0，即$Q_i(0)=0$。

根据网络状态，在 t 时槽内调度器建立任务设备到资源设备的调度通道，并传输数据流。来自任务设备 i 的传输过程产生的总能耗$P_{i,\mathrm{sum}}(t)$为：

$$P_{i,\mathrm{sum}}(t)=\sum_{j\in\Delta_i}(x_{ij}(t)P_{ij}(t)+x_{ij}(t)P_{ji}(t)) \tag{5}$$

其中，$P_{ij}(t)$为从任务设备 i 到资源设备 j 的传输能耗，$P_{ji}(t)$为从资源设备 j 到任务设备 i 的传输能耗。由于本文采用的是蜂窝 D2D 通信，根据文献[17,20-21]，任务设备 i 到资源设备 j 的传输速率可以表示为：

$$K_{ij}(t)=w_{ij}\log\left(1+\frac{P_{ij}(t)\cdot g_{ij}(t)}{\sigma(t)}\right) \tag{6}$$

$$L_{ji}(t)=w_{ij}\log\left(1+\frac{P_{ji}(t)\cdot g_{ji}(t)}{\sigma(t)}\right) \tag{7}$$

其中，本文假设复用的信道为上行带宽，并且系统通过正交的方式给设备之间分配一段连续的频谱，w_{ij}为分配给设备 i 和 j 之间复用子信道的带宽。根据文献[20]，$g_{ij}(t)$为设备 i 到设备 j 的信道增益，$g_{ji}(t)$为设备 j 到设备 i 的信道增益，$\sigma(t)$为背景干扰(如背景噪声)。由于干扰主要来自于复用子信道，本文假设系统中干扰为固定值。式(7)为资源设备 j 回传结果的传输速率公式，$L_{ji}(t)=K_{ij}(t)*\rho_i(t)$。为了简便，记 $F()$为任务设备中传输速率到传输能耗的映射，即$P_{ij}(t)=F(K_{ij}(t))$。

本文关注在稳定长期平均队列下的长期平均总能耗和长期平均总效用，其表达式为：

$$\bar{P}=\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in N}P_{i,\mathrm{sum}}(t) \tag{8}$$

$$\bar{w}=\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in N}w_{i,u}(t) \tag{9}$$

综上，本文的优化问题模型 P1 可以表示为：

$$\min_{u,r(t),x(t),p(t)}(\bar{P}-\beta\bar{w}) \tag{10}$$

s. t. (2)(3)

$$\sum_{j\in\Delta_i}x_{ij}(t)\leqslant 1,i\in N \tag{11}$$

$$0\leqslant\sum_{i\in\gamma_j}y_{ij}(t)\leqslant f_j,j\in M \tag{12}$$

$$0\leqslant P_{ji}(t)\leqslant P_{j,\max},j\in M,i\in\gamma_j \tag{13}$$

$$0\leqslant P_{ij}(t)\leqslant\min\{P_{i,\max},F(Q_i(t)/\varepsilon)\},i\in N,j\in\Delta_i \tag{14}$$

$$x_{ij}(t)\in\{0,1\} \tag{15}$$

$$\lim_{T\to\infty}\frac{1}{T}E\{Z_i(t)\}=0,\lim_{T\to\infty}\frac{1}{T}E\{Q_i(t)\}=0,i\in N \tag{16}$$

其中，β为非负控制参数，用于调整效用的比重。式(11)表明，在时槽 t 内，任务设备 i 的调度对象有且只有一个或者零个；式(12)代表资源设备 j 中被调度的资源量应小于自身的能力；式(13)为资源设备传输能量应小于其传输能耗的最大值；同样，式(14)是任务设备 i 的传输能量应小于其传输能耗的最大值，并且传输量应小于 TBL 队列长；式(16)表示队列长应在平均速率下保持稳定[22]。

4 实时调度算法

为了解决 P1，本文设计了基于李雅普诺夫优化的实时调度算法，其子问题 SP2 是一个 NP-hard 的组合优化问题。为此，本文设计了一种基于贪心的设备匹配算法，以较低的复杂度在每个时槽上求解一个近似最优解。

4.1 李雅普诺夫优化

由于数据流任务具有随机性和动态性(如行人追踪)，很难预测下一个时槽任务的生成情况，因此，在当前时槽内应通过分配现有的资源使系统队列长期保持稳定。为了描述系统的总队列长，本文首先定义 $I(t)\triangleq\{Q_i(t)+Z_i(t)\mid i\in N\}$为当前系统负载状态集合，并且定义李雅普诺夫方程为：

$$L(t)=\frac{1}{2}\sum_{i\in N}(Q_i^2(t)+Z_i^2(t)) \tag{17}$$

式(17)代表系统的队列积压程度。在每个时槽内，调度器给出的决策使得李雅普诺夫方程始终保持在一个界限下，那么该系统始终保持稳定。因此，本文通过减少李雅普诺夫漂移(Lyapunov-drift)来使得李雅普诺夫方程始终保持在一个拥塞程度较低的区域内。

$$\Delta(I(t))=E[L(t+1)-L(t)\mid I(t)] \tag{18}$$

李雅普诺夫漂移 $\Delta(I(t))$代表经过一个时槽后在李雅普诺夫方程中总队列长度的变化程度，此项变化程度取决于当前策略的选择，一个有效的策略可以使 $\Delta(I(t))$的变动较小。因此，将队列稳定性加入能耗优化中，李雅普诺夫漂移加罚函数(Lyapunov drift-plus-penalty) $\Delta v(I(t))$表示为：

$$\Delta v(I(t))=\Delta(I(t))+VE[\sum_{i\in N}(P_{i,\mathrm{sum}}(t)-\beta w_{i,u}(t))\mid I(t)] \tag{19}$$

其中，V 是非负控制变量，用于调整队列长和总能耗加权总效用的权衡值。

定理 1 对于任意 $r_i(t)\in[0,b_{\max}]$，$\sum_{j\in\Delta_i}y_{ij}(t)\in[0,\max[f_j^2\mid j\in\Delta_i]]$，$\Delta v(I(t))$存在上限：

$$\begin{aligned}\Delta v(I(t))\leqslant B&+E[\sum_{i\in N}r_i(t)(Q_i(t)-Z_i(t))\mid I(t)]+\\&E[\sum_{i\in N}Z_i(t)(b_i(t)-c_i*\varepsilon)\mid I(t)]-\\&E[\sum_{i\in N}Q_i(t)\sum_{j\in\Delta_i}y_{ij}(t)\mid I(t)]+\\&V[\sum_{i\in N}P_{i,\mathrm{sum}}(t)-\beta w_{i,u}(t)\mid I(t)]\end{aligned} \tag{20}$$

其中，$B\triangleq\sum_{i\in N}[\max[b_{\max}^2,\max[f_j^2\mid j\in\Delta_i]]+b_{\max}^2+(c_i*\varepsilon+b_{\max})^2]$是在所有时槽上的一个常量。

证明：$\Delta(I(t))$可以表示为以下形式：

$$\begin{aligned}\Delta(I(t))=&\frac{1}{2}\sum_{i\in N}E[(Q_i(t)-\sum_{j\in\Delta_i}y_{ij}(t)+r_i(t))^2-Q_i^2(t)+\\&(\max\{Z_i(t)+b_{i,u}(t)-c_i*\varepsilon-r_i(t),0\})^2-\\&Z_i^2(t)\mid I(t)]\\=&\frac{1}{2}\sum_{i\in N}E[2Q_i(t)(r_i(t)-\sum_{j\in\Delta_i}y_{ij}(t))+(r_i(t)-\\&\sum_{j\in\Delta_i}y_{ij}(t))^2+(\max\{Z_i(t)+b_{i,u}(t)-c_i*\varepsilon-\\&r_i(t),0\})^2-Z_i^2(t)\mid I(t)]\end{aligned}$$

显然，对于任意 $a,b,c\geqslant 0$，存在$(\max\{a+b-c,0\})^2\leqslant a^2+b^2+c^2+2a(b-c)$。那么公式 $\Delta(I(t))$中，其中一项可以推导为：

$$\begin{aligned}&(\max\{Z_i(t)+b_{i,u}(t)-c_i*\varepsilon-r_i(t),0\})^2-Z_i^2(t)\\&\leqslant b_{i,u}^2(t)+(c_i*\varepsilon+r_i(t))^2+2Z_i(t)(b_{i,u}(t)-c_i*\varepsilon-\end{aligned}$$

$r_i(t))$

又因为：

$$\sum_{i\in N}[(r_i(t)-\sum_{j\in\Delta_i}y_{ij}(t))^2+b_{i,u}^2(t)+(c_i*\varepsilon+r_i(t))^2]$$
$$\leqslant\sum_{i\in N}\{\max[b_{\max}^2,\max[f_j^2\mid j\in\Delta_i]]+b_{\max}^2+(c_i*\varepsilon+b_{\max})^2\}\triangleq B$$

所以，$\Delta(I(t))\leqslant\sum_{i\in N}E[Q_i(t)(r_i(t)-\sum_{j\in\Delta_i}y_{ij}(t))+Z_i(t)(b_{i,u}(t)-c_i*\varepsilon-r_i(t))\mid I(t)]+B$ 在上式两边加上加权总能耗和加权总效用，定理 1 得证。

通过李雅普诺夫优化后，原问题可以转化为实时优化问题，目标是在决策集合域内找到一个策略$(u,r(t),x(t),p(t))$，使得 $\Delta v(I(t))$的上限值最小化，因此李雅普诺夫方程 $L(t)$的值始终靠近一个界定值附近，并且加权总能耗可以最小化。

4.2 实时调度算法设计

本节提出了实时调度算法，其主要思想是将 P1 转化为最小化 $\Delta v(I(t))$的上限问题，也就是对式(20)小于等于号右边式子在每个时槽内进行最小化。因此，P1 问题转化为 P2 问题，表达式为：

$$\underset{u,r(t),x(t),p(t)}{\text{Min}}\ \sum_{i\in N}[r_i(t)(Q_i(t)-Z_i(t))-Q_i(t)\sum_{j\in\Delta_i}y_{ij}(t)+VP_{i,\text{sum}}(t)-V\beta w_{i,u}(t)]$$
s. t. (2)(3),(11)－(16) (21)

为了简化公式，本文分别使用$U_1(u,r,t)$，$U_2(x,p,t)$去替换目标函数，u 表示所有可行的任务操作集，r 表示所有可行的传输量集合，x 和 p 表示所有可行的匹配策略与能耗集合。算法 1 中，本文通过对每个时槽计算式(21)，来使缓存在 SAIL 和 TBL 中的数据流任务保持稳定，同时总能耗加权总效用可以被优化。另外，算法 1 根据式(1)、式 (4)更新两层队列长。

算法 1 实时调度算法

1. 初始化设置$Z_i(0)=0$，$Q_i(0)=0$，并初始化计算资源和通信资源等信息；
2. 在每个时槽开始时，调度器获得任务初始信息和$\{Z_i(t)\}$，$\{Q_i(t)\}$，$\{g_{ij}(t)\}$；
3. 解决 P2 优化公式得到 u，r(t)，x(t)，p(t)；
4. 时槽 t 加 1 并重复步骤 2 和步骤 3。

后文将进一步阐述 P2 的近似最优解，包括最优两层间传输的任务量、最优任务操作集、近似最优发射传输功率以及近似最优匹配策略。本文观察 P2，通过解决子问题 SP1 得到最优两层间传输的任务量和最优任务操作集。

$$\underset{u,r(t)}{\text{Min}}\sum_{i\in N}[r_i(t)(Q_i(t)-Z_i(t))+b_{i,u}(t)Z_i(t)-V\beta w_{i,u}(t)] \quad (22)$$
s. t (2)(3)

两层间传输的任务量的更新方式如下：

$$r_i(t)=\begin{cases}b_{i,u}(t), & \text{for } Q_i(t)<Z_i(t)\\ 0, & \text{otherwise}\end{cases} \quad (23)$$

通过式(23)的准入控制可以看到，它实现了基于阈值的数据流控制，仅当$Q_i(t)-Z_i(t)$为负时才允许所有当前输入负载进入 TBL，这意味着当 TBL 中缓存的任务量比 SAIL 中缓存的任务量少时，传输的任务量大小为当前输入负载；当 TBL 中缓存的任务量比 SAIL 中缓存的任务量多时，此时设备不再传输数据流任务到 TBL 中。

任务的操作集分为收集任务所需的数据量以及选取的计算退出点，不同的组合方式对应不同的效用，即在操作集 $u\in\vartheta$ 内设备 i 输入负载$b_{i,u}(t)$映射于唯一效用值$w_{i,u}(t)$。因此，最优任务操作集 u 可以根据以下公式得到：

$$u=\arg\min\{Q_i(t)b_{i,u}(t)-V\beta w_{i,u}(t),Z_i(t)b_{i,u}(t)-V\beta w_{i,u}(t)\} \quad (24)$$

其次，本文观察 P2，通过解决 SP2 得到近似最优发射传输功率和近似最优匹配策略：

$$\underset{x(t),p(t)}{\text{Min}}\ \sum_{i\in N}[-Q_i(t)\sum_{j\in\Delta_i}y_{ij}(t)+VP_{i,\text{sum}}(t)] \quad (25)$$
s. t. (11)－(16)

每个通道内最优能耗值$P_{ij}^*(t)$为：

$$P_{ij}^*(t)=\arg\min\{U_{ij}(P_{ij}^-(t)),U_{ij}(0),U_{ij}(\min\{P_{i,\max},F(Q_i(t)),d_{ij}(t),e_{ij}(t)\})\} \quad (26)$$

其中，$d_{ij}(t)=\frac{\sigma}{g_{ij}(t)}\left(\sqrt[p_i(t)]{\frac{g_{ji}(t)}{\sigma(t)}P_{j,\max}+1}-1\right)$，$e_{ij}(t)=\frac{\sigma}{g_{ij}(t)}(2^{\frac{f_j}{w_{ij}A_i(t)\varepsilon}}-1)$，$P_{ij}^-(t)$可以通过牛顿法求得[23]。很明显，在每个任务设备到资源设备的决策通道上，计算最优发射功率$P_{ij}^*(t)$时，任务设备是相互独立的，但是在联合计算$U_2(x,p,t)$时，由于限制式(12)和式(13)中不同任务设备竞争相同的资源设备，因此任务设备之间又是相互竞争的。为了解决这类问题，本文提出了基于贪心的设备匹配算法。

算法 2 基于贪心的设备匹配算法

1. 初始化信道信息并计算每个通道最优发射功率等信息；
2. 将每个通道内的最优发射功率(包括发射功率为 0)带入 SP2，获得并记录每个设备通道值$U_{ij}(t)=-Q_i(t)\sum_{j\in\Delta_i}y_{ij}(t)+VP_{i,\text{sum}}(t)$，根据通道排序将任务设备号分别记入 rank 队列；
3. 调度器优先选择使得 SP2 最小化的通道，并将被选择的资源设备和任务设备进行加锁操作；
4. 重复步骤 2、步骤 3，直到每个任务设备都确定其最优发射功率和最优通道，解锁所有的资源设备和任务设备。

5 理论分析

本节提供了性能分析，然后在以下定理中进一步说明了所提方案的性能界限。

定理 2 假设数据流任务的到达率严格在可行域∂内，则在算法 1 下，长期平均总能耗和总效用的性能边界表示为：

$$\bar{P}-\beta\bar{w}\leqslant P^{\text{opt}}+\frac{B}{V}-\beta w^{\text{opt}} \quad (27)$$

其中，P^{opt}和w^{opt}是 P2 问题的最优值。

证明：如果 P2 问题是可解决的，那么对于任意 $\delta>0$，存在一个平稳随机的策略 π 满足所有瞬时限制：

$$\begin{cases}E[P_\Sigma^\pi(t)]\leqslant P^{\text{opt}}+\delta\\ E[w_\Sigma^\pi(t)]\leqslant w^{\text{opt}}+\delta\\ E[b_{i,u^\pi}(t)]-c_i*\varepsilon\leqslant E[r_i^\pi(t)]+\delta\\ E[r_i^\pi(t)]\leqslant E[\sum_{j\in\Delta_i}y_{ij}^\pi(t)]+\delta\end{cases} \quad (28)$$

其中，$P_{\Sigma}^{\pi}(t)=\sum_{i\in N}P_{i,\mathrm{sum}}^{\pi}(t)$，$w_{\Sigma}^{\pi}(t)=\sum_{i\in N}w_{i,u}^{\pi}(t)$。假设每个时槽内的最优策略为$\{u^*,r^*,x^*,p^*\}$，则：

$$\begin{aligned}\Delta v(I(t))\leqslant & B+E[\sum_{i\in N}r_i^*(t)(Q_i(t)-Z_i(t))\mid I(t)]+\\ & E[\sum_{i\in N}Z_i(t)(b_{i,u^*}(t)-c_i*\varepsilon)\mid I(t)]-\\ & E[\sum_{i\in N}Q_i(t)\sum_{j\in\Delta_i}y_{ij}^*(t)\mid I(t)]+V[\sum_{i\in N}(P_{i,\mathrm{sum}}^*(t)-\\ & \beta w_{i,u}(t))\mid I(t)\\ \leqslant & \sum_{i\in N}\delta(Q_i(t)+Z_i(t))+V(P^{\mathrm{opt}}-\beta w^{\mathrm{opt}}+\delta-\\ & \beta\delta)+B\end{aligned}\tag{29}$$

对于 P2 问题，由于平稳随机的策略 π 是次优的，因此不等式(29)成立。令 $\delta\to 0$ 可以得到式(27)，得证。

定理 3 假设存在 $\varepsilon>0$ 且 $\psi(\varepsilon)(P^{\mathrm{opt}}<\psi(\varepsilon))$ 满足 Slater 条件[23]，则两个缓冲区的平均队列长度之和 $\bar{q}$ 满足：

$$\bar{q}\leqslant\frac{B+V(\psi(\varepsilon)-P^{\mathrm{opt}}+\beta w^{\mathrm{opt}})}{\epsilon}\tag{30}$$

$$E[P_{\Sigma}^{\pi}(t)-\beta w_{\Sigma}^{\pi}(t)]=\psi(\epsilon)\tag{31}$$

其中，$\bar{q}\triangleq\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in N}[Z_i(t)+Q_i(t)]$，$\epsilon$表示数据流任务到达率与可行域$\partial$边界之间的距离的一种衡量。

证明：类似的证明可以在文献[22]中找到。根据以上分析，可以从式(27)和式(30)中找到关于优化目标和队列长的一个$[O(1/V),O(V)]$权衡。因此，网络管理员可以通过调整 V 来平衡长期平均队列长。例如，对于延迟敏感型的应用，可以将 V 的值调整得更小；而对于能量敏感网络和容迟应用，可以将 V 的值调整得更大。

6 仿真与评估

6.1 实验设置

为了验证实时调度算法的有效性和稳定性，本文利用 MATLAB 进行仿真实验，并与随机法和穷举法的结果进行对比分析，同时分析关键参数对系统的影响。本文采用不同的连接概率去仿真物联网环境。

如图 4 所示，实验环境是随机网络(Random Network)，灰色的点为任务设备，黑色的点为资源设备。随机网络中通过设定固定的连接概率，对每个资源设备到任务设备随机建立连接。类比于一般的传感器网络，设备之间的连接都是随机的，这样避免了单个设备连接大量设备的情况，符合本文研究的物联网环境。实验中，设定连接概率为 0.3～1，任务设备和资源设备的数量 N 和 M 分别为 5 和 10。

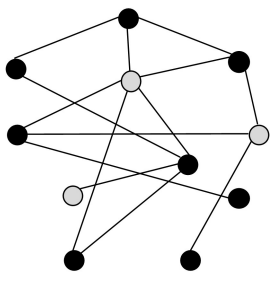

图 4 随机网络

Fig. 4 Random network

6.1.1 计算能力和通信能力的限制

本文模型中存在长期资源紧张的任务设备和资源丰富的资源设备。这些资源可以是长期资源空闲的设备，也可以是边缘计算服务器。假设资源丰富的设备的可用 CPU 周期数 f_j 为 12～20 GHz，资源匮乏的任务设备的可用 CPU 周期数 c_i 为 3～4 GHz。系统分配给蜂窝 D2D 连接的上行带宽 w_{ij} 和下行带宽 w_{ji} 均分配为 1～10 Mb/s。时槽长度 ϵ 设置为 1s，最高传输功率 $P_{i,\max}$ 和回传功率 $P_{j,\max}$ 均设置为 5 W。信道干扰 $\sigma(t)$ 设置为 10^{-13} W，信道增益 $g(t)$ 是关于 $g_0(d_0/d)^4$ 的指数分布，其中信道损失限制 $g_0=-40$ dB，基准距离 $d_0=10$ m，两种设备之间的平均距离为 $d=50$ m。

6.1.2 任务参数

参考文献[19]，任务的种类大致可以分为 3 种：1)文字处理任务，这类任务需要低计算频率和低传输速率，具体数值为表 1 中的任务一和任务二；2)视频压缩任务，这类任务需要低计算频率和高传输速率，具体数值为表 1 中的任务三；3)目标跟踪，这类任务需要高计算频率和高传输速率，具体数值为表 1 中的任务四。每种任务的持续时间平均设置为 35 个时槽，输出比率平均设置为 0.28，任务生成的概率为 0.5。因此，任务设备上的任意时槽内可能存在多种任务的随机组合，那么每个时槽上输入到任务设备的任务负载量存在唯一的最大值，因此最大输入任务所需资源量为 $b_{\max}=2.6*10^{10}$ cycles。设备的操作集 u 如表 2 所列，以任务一为例。

表 1 任务设置

Table 1 Set of tasks

	计算频率/(c/bits)	传输速率/Mbps
任务一	$0.6\sim1.0*10^3$	0.6～1.0
任务二	$0.9\sim1.3*10^3$	1.3～2.0
任务三	$0.9\sim1.3*10^3$	3.5～4.0
任务四	$1.3\sim1.6*10^3$	6.5～7.0

表 2 任务操作集

Table 2 Set of operation

操作	计算频率/(c/bits)	传输速率/Mbps	效用
一	$1.0*10^3$	1.0	1.0
二	$0.8*10^3$	1.0	0.9
三	$0.8*10^3$	0.8	0.8
四	$0.6*10^3$	0.8	0.7

6.2 实验结果

首先显示本文算法在随机网络中的性能，并且展示参数对算法性能的影响。如图 5 所示，本文设置控制变量 β 为 0.1～1，$V=0.5*10^{19}$。图 5 中的点是总效用收敛至平衡点时的位置。从图 5 可以看出，在固定 V 时，系统总效用正比于 β 的大小。如式(10)所示，β 的增加使得在式(27)左式中增大了效用参数的比例，从而获得了更高的总效用。设备长期平均队列长度随着 β 的增大而增大。当系统需要视频质量更好和处理结果更佳时，可以适当提高参数 β 的值。本文同样分析了另一个参数 V 对设备长期队列长和长期平均系统总能耗的影响，本文设置 $\beta=0.7$。

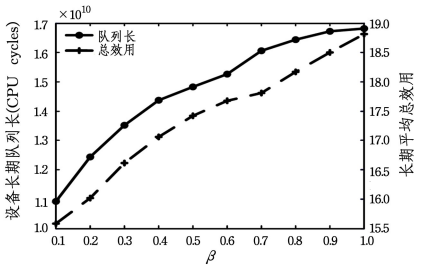

图 5 设备长期平均队列长 $\bar{q}/N$ 和长期平均总效用 $\bar{w}$ 与控制变量 β 的关系

Fig. 5 Average queue length per task device and long-term average sum utility vs. β

图 6 给出了系统总能耗和总队列长度分别与权重 V 成 $[O(\frac{1}{V}),O(V)]$相关,验证了式(27)和式(30)。当控制变量 V 足够大时,系统的能耗会减少得更少。然而,当控制变量 V 足够大时,长期平均系统总能耗也会减少得更少。因此,选择合适的控制变量,可以有效地平衡总能耗和队列长度的大小。针对延时敏感的数据流任务时,可以选择小数值的控制变量 V;同样地,针对能耗敏感的数据流任务时,可以选择大数值的控制变量 V。图 7 进一步给出了随着时槽增长系统中长期平均队列长度随着时槽的变化,以及在不同控制变量下系统队列收敛点的位置。横向看,图像显示出当系统运行 1000 次迭代后,平均队列长度收敛到一个平衡点,并上下浮动,这是因为在李雅普诺夫优化中,平均队列长度会逐渐逼近平衡点,直至达到系统稳定。纵向看,当控制变量 V,β 增长时,收敛点也随之增大,从 $1.18*10^{10}$ cycles 变化到 $1.81*10^{10}$ cycles。

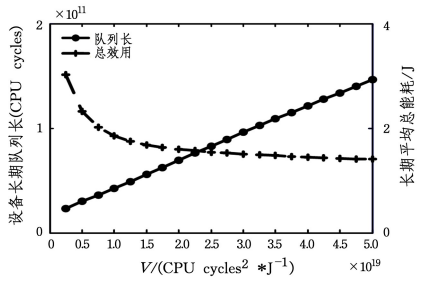

图 6 设备长期平均队列长 $\bar{q}/N$ 和长期平均总能耗 $\bar{P}$ 与控制变量 V 的关系

Fig. 6 Average queue length per task device $\bar{q}/N$ and long-term average sum energy consumption vs. V

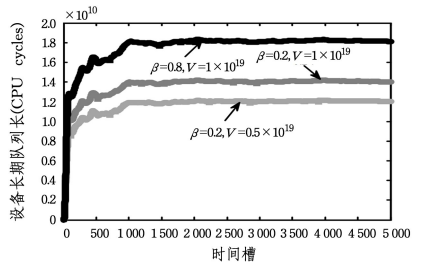

图 7 设备长期平均队列长 $\bar{q}/N$ 与控制变量 V,β 的关系

Fig. 7 Average queue length per task device $\bar{q}/N$ vs. V,β

如图 8 所示,在不同控制变量 V 下将最优操作集和未被优化的原始操作集进行对比,总体上看,优化过的操作集能使设备长期平均队列长下降 3.78%,虽然代价是牺牲一部分总效用,但是使得系统总体负载有所下降。为了体现本文提出的匹配算法的性能,本文将其和枚举法进行对比。如图 9 所示,本文将枚举法作为基准算法,使用性能比作为评价标准,分别用待测评的算法减去枚举法数值再除以枚举法数值,用 1 减去得到的数据得到性能比。性能比可以体现当前算法与枚举法结果的差距,当差距小时,性能比接近 1;当差距过大时,性能比接近负数。总体上,本文算法在不同场景中和不同连接概率下可以到达 98.3%的比率,比随机匹配策略最好的情况高 8.6%。这是因为系统中的调度方式采用的是集中式,并且在随机法最好的情况是在连接概率为 0.3 时,此时随机法命中最优调度通道的概率较高。然而,随机匹配策略在连接概率增长的过程中效果却逐渐下降。

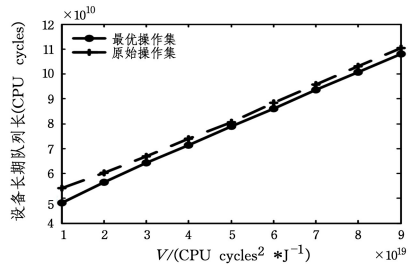

图 8 不同控制变量 V 下最优操作集和原始操作集的结果对比

Fig. 8 Performance of the best operation set and original operation set vs. V

结束语 本文研究了在 IoT 环境中面向数据流任务的调度问题,针对多设备之间的资源分配提出了基于李雅普诺夫优化的实时调度算法。该算法在稳定系统队列的基础上优化长期平均总能耗和加权总效用。此外,本文还提出了基于贪心的设备匹配算法。仿真实验的结果表明,该算法达到了 98.3%的性能。接着实验验证了控制变量 V,β 和设备长期平均队列长的关系,明确描述了控制变量关于长期平均总能耗和加权总效用的权衡,证实了优化操作集的重要性。由于设备本身具有移动性,在本文模型中缺乏设备移动性的讨论,将来的工作着重解决模型在移动场景中的应用。

参 考 文 献

[1] WANG B Y. Review on internet of things[J]. Journal of Electronic Measurement and Instrumentation, 2009, 23(12): 1-7.

[2] SHI W S, SUN H, CAO J, et al. Edge Computing-An Emerging Computing Model for the Internet of Everything Era[J]. Journal of Computer Research and Development, 2017, 54(5): 907-924.

[3] XIE R C, LIAN X F, JIA Q M, et al. Survey on computation offloading in mobile edge computing[J]. Journal on Communications, 2018, 39(11): 138-155.

[4] BEIGI N K, PARTOV B, FAROKHI S. Real-time cloud robotics in practical smart city applications[C]// IEEE 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mo-

bile Radio Communications (PIMRC). Montreal, Canada, 2017: 1-5.

[5] GUO M, LI L, GUAN Q. Energy-Efficient and Delay-Guaranteed Workload Allocation in IoT-Edge-Cloud Computing Systems[J]. IEEE Access, 2019, 7(99): 78685-78697.

[6] CHEN Y, YANG T, LI C, et al. A Binarized Segmented ResNet Based on Edge Computing for Re-Identification [J]. Sensors, 2020, 20(23): 1-19.

[7] LI Z, ZHANG X. Resource Allocation and Offloading Decision of Edge Computing for Reducing Core Network Congestion[J]. Computer Science, 2021, 48(3): 281-288.

[8] KUMAR K, LU Y H. Cloud computing for mobile users: Can offloading computation save energy? [J]. Computer, 2010(4): 51-56.

[9] BARBERA M V, KOSTA S, MEI A, et al. To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing[C]// Proceedings of IEEE INFOCOM. Turin, Italy, 2013.

[10] CHEN X, JIAO L, LI W, et al. Efficient multi-user computation offloading for mobile-edge cloud computing [J]. IEEE/ACM Transactions on Networking, 2015, 24(5): 2795-2808.

[11] MAO Y, ZHANG J, SONG S H, et al. Stochastic Joint Radio and Computational Resource Management for Multi-User Mobile-Edge Computing Systems[J]. IEEE Transactions on Wireless Communications 2017, 16(9): 5994-6009.

[12] LI G S, WANG J P, WU J H, et al. Data processing delay optimization in mobile edge computing[J/OL]. Wireless Communications and Mobile Computing, 2018: 1-9. https://www.hindawi.com/journals/wcmc/2018/6897523/.

[13] LI N, MARTINEZ-ORTEGA J F, DIAZ V H J I A. Distributed power control for interference-aware multi-user mobile edge computing: A game theory approach[J]. IEEE Access, 2018, 6: 36105-36114.

[14] SAHNI Y, CAO J, ZHANG S, et al. Edge Mesh: A new paradigm to enable distributed intelligence in Internet of Things[J]. IEEE Access, 2017, 5: 16441-16458.

[15] FORTUNA C, GALE T. Streaming Data Processing for IoT [M]. John Wiley & Sons, Ltd, 2020.

[16] ARRIBAS E, MANCUSO V. Achieving Per-Flow Satisfaction with Multi-Path D2D[J]. Ad Hoc Networks, 2020(106): 1-16.

[17] ZHANG Z, WU Y, CHU X, et al. Energy-Efficient Transmission Rate Selection and Power Control for Relay-Assisted Device-to-Device Communications Underlaying Cellular Networks [J]. IEEE Wireless Communication Letters, 2020, 9(8): 1133-1136.

[18] BACCARELLI E, NARANJO P G V, SHOJAFAR M, et al. Energy and delay-efficie nt dynamic queue management in TCP/IP virtualized data centers [J]. Computer Communications, 2016, 102: 1-37.

[19] LIU J, LUO K, ZHOU Z, et al. ERP: Edge Resource Pooling for Data Stream Mobile Computing [J]. IEEE Internet of Things Journal, 2019, 6(3): 4355-4368.

[20] JIA Q, XIE R, TANG Q, et al. Energy-Efficient Computation Offloading in 5G Cellular Networks with Edge Computing and D2D Communications[J]. IET Communications, 2019, 13(8): 1122-1130.

[21] PU L, CHEN X, XU J, et al. D2D Fogging: An Energy-Efficient and Incentive-Aware Task Offloading Framework via Network-Assisted D2D Collaboration[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3887-3901.

[22] MICHAEL N. Stochastic Network Optimization with Application to Communication and Queueing Systems [M]. Morgan & Claypool, 2010.

[23] WIMP J. Pi and the AGM: A study in analytic number theory and computational complexity (Jonathan M. Borwein and Peter B. Borwein)[J]. SIAM Review, 1988, 30(3): 530-533.

ZHANG Zhong-yu, born in 1996, postgraduate. His main research interests include edge computing and Internet of Things.

CHEN Yan-ming, born in 1983, Ph. D. His main research interests include distributed algorithms, edge computing, neural networks and model compression.

(责任编辑:喻藜)