尚坤豪,符琦,巴冰. 一种基于路边单元协作的任务卸载策略[J]. 湖南科技大学学报(自然科学版), 2022, 37(1):100-108. doi:10.13582/j.cnki.1672-9102.2022.01.014

SHANG K H,FU Q,BA B. A Task Offloading Strategy Based on Rode Side Unit Cooperation [J]. Journal of Hunan University of Science and Technology (Natural Science Edition) , 2022, 37(1):100-108. doi:10.13582/j.cnki.1672-9102.2022.01.014

一种基于路边单元协作的任务卸载策略

尚坤豪,符琦*,巴冰

(湖南科技大学 计算机科学与工程学院,湖南 湘潭 411201)

摘 要:在车辆边缘计算框架中,当车辆处于行驶状态时,需要和不同的路边单元(Rode Side Unit,RSU)进行连接,此时必然存在 RSU 的切换.文章着力于减少车辆在 RSU 切换过程中任务所需的完成时间.针对这个问题,提出相邻 RSU 协同工作的通信结构,并基于车辆和 RSU 之间的距离设计了一种求解车辆在 RSU 通信覆盖范围内剩余行驶时间的算法.当前 RSU 利用车辆的剩余行驶时间决定是否将计算任务进行卸载.若需要卸载,当前 RSU 就将任务卸载给车辆将要连接的下一个 RSU,利用下一个 RSU 进行任务的计算,最终将计算结果返回给目标车辆.最后,利用 Veins 框架将 OMNet++和 SUMO 结合在一起进行仿真实验.对仿真结果进行分析后可知:相较于相邻 RSU 没有进行协同工作的场景,文章提出的 RSU 协作通信结构和算法在任务的完成时间上改善了 25%.

关键词:RSU 协作;消息转发;任务卸载;车辆边缘计算;车联网

中图分类号:TP29 **文献标志码**:A **文章编号**:1672-9102(2022)01-0100-09

A Task Offloading Strategy Based on Rode Side Unit Cooperation

SHANG Kunhao, FU Qi, BA Bing

(School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China)

Abstract: In the framework of vehicle edge computing, when the vehicle is in the state of driving, it needs to connect with different roadside units (RSU), and there must be a switch of RSU at this time. This paper focuses on reducing the task completion time of the vehicle in the process of RSU handover. For this problem, this paper proposes a communication structure of adjacent RSU to work together, and designs an algorithm based on the distance between the vehicle and the RSU to solve the remaining travel time of the vehicle within the communication coverage of the RSU. The current RSU uses the remaining travel time of the vehicle to decide whether to forward the calculation task. If unloading is required, the current RSU will unload the task to the next RSU to which the vehicle will be connected, and use the next RSU to calculate the task and return the calculation result to the target vehicle.Finally, this paper uses the Veins framework to combine OMNet++ and SUMO for simulation experiments. After analyzing the simulation results, it can be known that compared with the scene of adjacent RSU non-cooperation, the communication structure and algorithm of RSU cooperation proposed in this paper improves the task completion time by 25%.

Keywords: RSU collaboration; message forwarding; task offloading; vehicle edge computing; internet of vehicles

收稿日期:2021-10-21

基金项目:湖南省自然科学基金资助项目(2017JJ4036;2018JJ2139);湖南省教育厅科学研究项目资助(17K033;19A174)

* **通信作者**,E-mail:fuqi@ hnust.edu.cn

移动边缘计算的发展越来越受到重视.特别是随着 5G 的发展,使得汽车功能变得更加丰富.为了让人们的出行更加安全和方便,汽车也像智能手机一样拥有越来越多的应用[1].但是某些车载应用对于延时的要求和手机应用不相同,如自然语言处理、自动驾驶[2]和协同交通管理系统等,对延时的要求很高.车联网的发展是随着云计算技术的成熟而发展起来的,特别是近几年车联网得到越来越多人的关注,相关的技术也慢慢开始成熟[3].云计算拥有大量的计算资源,但是它部署在云端,车辆需要将消息发送到云端才能进行通信.由于存在较高的延时,对于实时性的车载应用就不能使用云计算[4].虽然车辆本身拥有车载计算(On-Board Computers,OBCs)资源,但是由于车载计算的硬件成本很高,增加 OBCs 资源会增大车辆制造成本,这不利于智能车的普及[5].边缘计算(Edge Computing,EC)的出现解决了这个问题,EC 是指允许在网络的边缘进行数据处理然后返回计算结果的技术.例如,智能手机是可穿戴联网设备的边缘,这些可穿戴设备的计算需求可以利用智能手机来完成[6].针对车联网,从边缘计算又引出了移动边缘计算(Mobile Edge Computing,MEC)和车辆边缘计算(Vehicle Edge Computing,VEC).MEC 的提出是为了使应用程序可以在网络边缘运行,它的特点是低延迟,高带宽[7].对于车辆网络来说,网络的边缘是 RSU.由于 RSU 的计算资源是有限的,我们在 VEC 中采用 RSU 和云服务器协同合作的方式可以有效地解决 RSU 计算资源不足的问题.当 RSU 的计算资源不充足的时候,将计算任务交给云服务器进行计算.车辆使用卸载调度算法决定计算任务是否需要卸载,如果车辆决定不进行任务卸载,计算任务由 OBCs 来完成.如果车辆进行任务卸载,就把任务卸载给 RSU,RSU 接收到后决定是要在本地执行还是将任务卸载到云服务器去执行[8].

在 VEC 架构中,车辆卸载计算任务给所通信的 RSU 后,在没有接收到返回结果前就驶离了当前 RSU 的通信覆盖范围,并到达下一个 RSU 的通信覆盖范围.此时由于相邻 RSU 之间没有进行通信,车辆就接收不到当前 RSU 的计算结果.车辆只能等待此任务请求超时之后,才能再次将这个任务卸载给其他 RSU,这个过程会使任务的完成时间快速升高,而且车辆在道路上行驶时会频繁地进行 RSU 的切换,为了车辆安全稳定的行驶,这类任务的延时是不可忽略的.

1　相关工作

由于云计算的技术相对比较成熟,近年来人们为了将云计算的技术扩展到移动设备中,提出了移动云计算,也有很多移动云计算的研究,这些研究[9-11]都在致力于使用不同的方法去解决计算卸载的问题.如果太多的移动设备用户同时通过无线访问将计算任务卸载到云中,它们之间很有可能产生严重的干扰,这对数据传输的速率会有很大的影响,从而使数据的传输时间增加.文献[9]提出将博弈论的方法用于移动云计算中来进行计算卸载,用户通过博弈可以自组织成互相满意的计算卸载决策.文献[10]表明移动计算是在动态环境下进行的过程且移动用户卸载计算的无线信道随机变化.文章提出将动态环境下的移动用户的卸载决策过程表述为随机博弈.

虽然云计算有大量的计算资源,但是它远距离传输产生的延时是不可忽略的,而且由于车辆的移动性使其和云的连接变得不稳定.为了解决这个问题,边缘计算开始得到研究人员的关注.边缘计算将计算设备安装在网络的边缘,从而减少大量数据传输的时间,从而达到那些实时车载应用的需求.对于边缘计算的研究主要是为了减少计算卸载的等待时间和最大化使用边缘计算的资源.文献[12]使用非正交多路访问的计算卸载方案,其中一组车辆可以将部分的计算工作量卸载到边缘服务器,以实现部分计算卸载从而减少应用的延时和能源的消耗.文献[13]提出了一种分布式算法,包括时钟频率配置、传输功率分配、信道速率调度和卸载策略.该算法对本地执行的时钟频率和边缘云执行的传输功率进行了联合优化.

边缘设备虽然可以快速地处理车辆产生的数据,但是它的计算资源有限而且远不如云计算.当需要计算的数据比较大时,边缘设备的计算资源就会被全部占用,此时其他的计算需求只能等待.为了追求整个系统的最大利益,车辆不能随意申请自己所需的资源.文献[14]提出在边缘车辆网络中,车辆具有高机动性,车辆和相关基础设施的连接具有间歇性,缺乏连通性.为了解决这类问题,作者在车辆边缘计算中对计

算卸载任务进行建模,还提出移动性预测检索(Mobility Prediction Retrieval,MPR)数据检索协议.为了更加有效地使用移动边缘网络中的基础设施,避免单个节点超负荷运行,文献[15]建议将负载均衡和卸载集成在一起,并在车辆边缘计算系统中研究多用户、多服务器的资源分配,提出一种低复杂度的算法来共同选择 VEC 服务器,并优化卸载率和计算资源.文献[16]设计基于匹配的服务器选择决策算法和基于 Adam 梯度优化法的计算任务卸载比例与资源分配联合优化算法,并对上述 2 种算法进行联合迭代,直至收敛,从而得到近似最优解以达到负载均衡.

为了满足日益增长的车辆应用需求,文献[17]提出将 MEC 和云计算结合在一起,进行协同计算即云辅助移动边缘.车辆可以将自己的计算任务留在本地进行计算,也可以卸载到 MEC 服务器或者云服务器上进行计算.对于延时要求不高的任务可以优先卸载到云服务器,对于延时要求高的任务优先卸载到 MEC 服务器.文献[18]为了提高云辅助移动边缘系统的性能,提出在线卸载调度和资源分配算法.将车辆请求计算资源的过程模拟成一个博弈的过程,在博弈中每辆车都会追求自己的最大利益,因此,每辆车都会在请求资源时计算收益,通过计算请求这个计算资源是否会受益,从而决定是否去请求这个计算资源.这个博弈过程最终会达到平衡即纳什平衡.文献[19]为了降低计算卸载的延迟和传输成本,提出了一种车载网络中基于云的移动边缘计算卸载框架.作者考虑到计算任务的时间消耗和车辆的移动性,提出一种有效的预测组合模式降级方案,任务可以通过直接上传或预测中继传输自适应的卸载到 MEC 服务器上.

2 通信结构和计算方法

2.1 RSU 协作通信结构

采用云服务器、边缘服务器和车辆节点构成车载通信网络架构.其中,RSU 作为边缘服务器部署在路边,以便车辆节点申请 RSU 的计算资源来完成相关任务的计算与卸载.当 RSU 的计算资源不充足时,RSU 则将计算任务卸载到云服务器进行远程处理.考虑到相邻 RSU 之间的协作,特别是在高速公路的场景中,本文提出的算法利用相邻 RSU 协作通信来减少任务完成的时间.同时约定,每个 RSU 有固定的通信范围 r,且任意 2 个 RSU 的通信覆盖范围都不重叠.相邻 RSU 之间能够利用基础设施对基础设施(Infrastructure to Infrastructure,I2I)的通信协议来传输消息.车辆节点会产生计算任务,计算任务可以通过 OBCs 来完成,若任务需要的计算资源较大,车辆可以将计算任务卸载到 RSU 或云端来完成.车辆和 RSU 通过车对基础设施(Vehicle to Infrastructure,V2I)的通信协议进行数据传输.相邻 RSU 协同工作过程如图 1 所示.

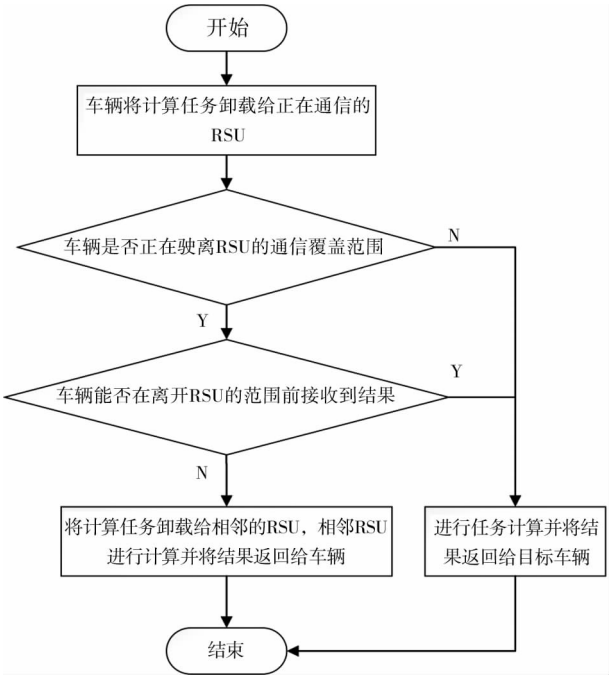

图 1 RSU 协作通信流程

在图 1 中,RSU 在接收到车辆卸载任务时,通过车辆和 RSU 之间的距离变化来判断车辆的行驶趋势,当距离变小时,表明车辆正在向着靠近 RSU 的方向行驶,不会很快离开当前 RSU 的通信范围,确定可以收到任务的返回结果;当距离变大时,表明车辆向着远离 RSU 的方向行驶,随时有可能驶出当前 RSU 的通信范围.此时,要判断车辆在离开通信覆盖范围前能否接收到返回结果,分为 2 种情况:第一种情况是在路边单元 R 的通信范围内,车辆将计算任务卸载给 R,在计算结果返回前,车辆驶离 R 的通信覆盖范围到达相邻的 RSU 的通信覆盖范围,导致车辆接收不到返回结果;第二种情况是车辆在离开 R 的通信覆盖范围前,可以接收到返回结果.当出现第一种情况时,R 计算出结果后,由于找不到接收的车辆节点只能将结

果抛弃,而车辆由于没有接收到任务的结果,只能在等待一个超时重传时间后将任务重新卸载,这会使任务完成所消耗的时间急剧增加.利用相邻 RSU 进行协同工作的方法来解决车辆接收不到计算结果的问题,具体方法是 R 将计算任务卸载到车辆的下一个 RSU,由这个 RSU 进行任务的计算并将计算结果返回给车辆.此时不需要考虑 RSU 计算资源不足的问题,当前 RSU 可以直接将计算任务卸载给相邻的 RSU,因为当 RSU 的计算资源不足时,RSU 会申请云服务器的计算资源来辅助自己进行计算.当出现第二种情况时,R 将计算结果直接返回给目标车辆.

2.2　资源队列模型

每个 RSU 具有有限的通信带宽,为了提高其带宽资源利用率,我们为每个 RSU 设计了动态计算的消息队列.每个 RSU 都设计了 2 个动态队列,分别保存来自车辆的消息和来自相邻 RSU 的消息.消息队列主要取决于消息任务的到达率和 RSU 的计算能力.RSU 的 2 个动态队列都采用 M/M/s 的排队模型,2 个队列模型中的 s 分别用于表示云服务器的个数和相邻 RSU 的个数.M/M/s 系统的服务强度如式(1)所示[20].

$$\rho_{mi}=\frac{\lambda_m}{i\mu_i}. \tag{1}$$

式中:λ_m 为消息 m 平均到达率的泊松分布;μ_i ($i=1,2,\cdots,s$)为 RSU 计算能力大小的指数分布.

为保持系统稳定 ρ_{mi} 必须小于 1,相应的平衡分布如式(2)所示.

$$B_K=\begin{cases}\dfrac{(i\rho_{mk})^k}{k!}B_0, 0\leqslant k\leqslant i-1;\\ \dfrac{\rho_{mi}i^i}{i!}B_0, \quad k\geqslant i.\end{cases} \tag{2}$$

其中,

$$B_0=\left[\sum_{k=0}^{i-1}\frac{(k\rho_{mj})^k}{k!}+\frac{i^i\rho_{mi}^i}{i!\ (1-\rho_{mi})}\right]^{-1}. \tag{3}$$

消息 m 转发给相邻路边单元 i 的等待时间如式(4)所示.

$$t_{im}=\frac{\rho_{mi}}{\lambda_m(1-\rho_{mi})^2}B_i. \tag{4}$$

2.3　RSU 负载模型

在 RSU 将消息卸载给相邻 RSU 的时候,需要考虑相邻 RSU 的工作负载,当前 RSU 根据负载状态判断相邻 RSU 是否适合接收任务,若适合就直接进行卸载,若不适合,则等待一段时间后进行再次判断,直到将任务卸载后才结束.RSU 工作负载的计算公式[21]如式(5)所示.

$$K=\gamma/\theta. \tag{5}$$

式中:K 为 RSU 的工作负载;γ 为单位时间内数据包的到达率;θ 为处理和计算的服务时间.

2.4　剩余行驶时间算法

在实际情况中并不是所有任务都需要卸载给相邻的 RSU,只有在车辆离开当前 RSU 通信覆盖范围前,接收不到计算结果的任务才需要卸载.为了判断出需要卸载给相邻 RSU 的任务,基于车辆和 RSU 之间的距离设计了一个计算车辆在当前 RSU 通信覆盖范围剩余行驶时间的算法.根据车辆的剩余行驶时间,RSU 可以决定车辆的哪些任务需要卸载给相邻的 RSU,因为这些任务在当前 RSU 内没有充分的时间将计算结果返回给目标车辆,只能由相邻的 RSU 进行计算并将结果返回给目标车辆.

该算法重点是用来计算车辆在当前 RSU 通信覆盖范围的剩余行驶距离,利用剩余行驶距离和车辆速度的比求出剩余行驶时间.我们定义在 RSU 通信范围内的不同时刻车辆 u 与 RSU 之间的距离为 $D_{\mathrm{VR}}^u=\{D_{\mathrm{VR1}}^u,D_{\mathrm{VR2}}^u,\cdots,D_{\mathrm{VRn}}^u\}$. 当车辆和 RSU 之间的距离缩短时,说明车辆向进入 RSU 通信覆盖范围的方向行驶,并逐渐靠近 RSU;当车辆和 RSU 之间的距离变大时,说明车辆向驶离 RSU 通信覆盖范围的方向行驶,并逐渐远离 RSU.设置 RSU 通信半径为 500 m.由于在 RSU 通信覆盖范围内的道路有很大概率是弯曲的,

很难直接求出道路的实际长度,所以当车辆在向着离开 RSU 通信覆盖范围的方向行驶时,根据车辆坐标可以在道路上确定一个点,然后计算出道路在此点的切线,根据切线来计算车辆在 RSU 通信覆盖范围的剩余行驶距离,从而求出剩余行驶时间.由于车辆在进行计算任务卸载时会频繁地向 RSU 发送消息,包括当前的坐标和上一次发送消息的坐标,此时这 2 个点表示的直线可以近似看成道路 S 的一条切线.

车辆在行驶过程中,我们使用 (X^u, Y^u) 表示车辆 u 本次向 RSU 发送消息时的坐标,使用 $(\text{before}X^u, \text{before}Y^u)$ 来记录车辆 u 上一次向 RSU 发送消息时的坐标,然后根据两点坐标可以得到一个直线方程 L.由于前后 2 次发送消息的间隔很小,则 L 可以看作当前道路 S 在点 (X^u, Y^u) 处的切线方程.用 D_{RS} 代表 RSU 到这个切线方程的垂直距离,RSU 的通信圆会截取切线 L 的一部分作为弦,用 L_s 代表这条弦长的一半.各个变量的直观表示如图 2 所示.

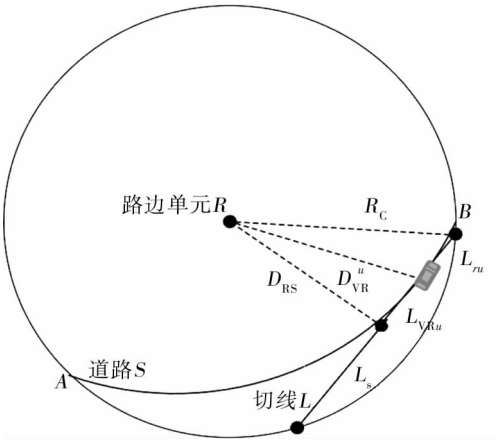

图 2 变量的直观表示

图 2 中圆心代表路边单元 R 的坐标,RSU 通信覆盖范围内的道路 S 用一条曲线表示,小车的位置表示车辆发送消息时的坐标,图 2 中车辆沿道路方向向 B 点行驶,$L_{\text{VR}u}$ 代表车辆 u 到切线和 RSU 垂直交点的距离;L_{ru} 代表车辆 u 在 RSU 通信覆盖范围的剩余行驶距离;R_{C} 代表 RSU 的通信半径,此时 RSU 的通信半径为 500 m;D^u_{VR} 代表车辆 u 和 RSU 之间的距离.

由于本算法只考虑车辆在 RSU 切换过程中任务完成所需要的延时,所以只需要计算车辆要离开 RSU 通信覆盖范围时的剩余行驶时间.车辆在 RSU 通信覆盖范围的剩余行驶时间的计算方法如式(6)所示.

$$t_{ru} = \frac{L_{ru}}{V_u}. \tag{6}$$

式中:t_{ru} 为车辆 u 在 RSU 通信覆盖范围的剩余行驶时间;V_u 为车辆 u 的当前速度;L_{ru} 的计算方法如式(7)所示.

$$L_{ru} = L_s - L_{\text{VR}u}. \tag{7}$$

$D^u_{\text{VR}i}$ 为车辆 u 当前位置与当前所通信的 RSU 的距离;D^u_{VRi-1} 为车辆 u 上一个位置与当前所通信的 RSU 的距离.当车辆和 RSU 之间的距离变大时,表明车辆正向着离开 RSU 通信覆盖范围的方向行驶,此时需要计算车辆在 RSU 通信覆盖范围内的剩余行驶距离并计算剩余行驶时间.其中,车辆和 RSU 之间距离的计算方法如式(8)所示.

$$D^u_{\text{VR}i} = \sqrt{(X^u - X^R)^2 + (Y^u - Y^R)^2}. \tag{8}$$

式中:(X^R, Y^R) 为 RSU 的坐标.

设 L 的方程为 $Ax + By + C = 0$,则 D_{RS} 的计算方法如式(9)所示.

$$D_{\text{RS}} = \left| \frac{AX^R + BY^R + C}{\sqrt{A^2 + B^2}} \right|. \tag{9}$$

L_s 的计算方法如式(10)所示.

$$L_s = \sqrt{R_{\text{C}}^2 - D_{\text{RS}}^2}. \tag{10}$$

$L_{\text{VR}u}$ 计算方法如式(11)所示.

$$L_{\text{VR}u} = \sqrt{(D^u_{\text{VR}i})^2 - D_{\text{RS}}^2}. \tag{11}$$

算法首先获得车辆和 RSU 的当前坐标,计算车辆和 RSU 之间的距离,通过比较前后 2 次距离的大小判断出车辆是在向 RSU 靠近还是远离.若是远离,说明车辆在向着离开 RSU 通信覆盖范围的方向驶去,则计算车辆在 RSU 的剩余行驶距离,并通过剩余行驶距离来计算剩余行驶时间;若是靠近,说明车辆向着 RSU 通信覆盖范围中心的方向驶去,车辆还会在 RSU 通信覆盖范围内行驶一段时间,此时不需要计算剩余行驶距离.剩余行驶时间算法的伪代码如算法 1 所示.

算法 1.短文本特征权值计算算法

输入:车辆坐标 (X^u, Y^u) 和速度 V_u, RSU 的坐标 (X^R, Y^R)

输出:车辆的剩余行驶时间 t_{ru}

① 初始化 $D^u_{\mathrm{VR0}} = 0$, $i = 1$;

② while(TRUE)

③ $D^u_{\mathrm{VR}i} = \sqrt{(X^u - X^R)^2 + (Y^u - Y^R)^2}$;

④ if $(D^u_{\mathrm{VR}i} > D^u_{\mathrm{VR}i-1})$

⑤ $t_{ru} = \dfrac{L_s - L_{\mathrm{VR}u}}{V_u}$;

⑥ end if

⑦ end while

最后,根据得出的剩余行驶时间 t_{ru}, RSU 可以判断任务的类型,决定任务是在本地进行计算,还是卸载给车辆的下一个 RSU 进行计算.我们用 t^{e2e} 代表车辆和 RSU 稳定连接时消息的点到点延时.当 $t_{ru} > t^{e2e}$ 时,说明目标车辆在离开此 RSU 通信覆盖范围前能接收到返回的计算结果,则此任务在本地进行计算并将计算结果发送给目标车辆;当 $t_{ru} < t^{e2e}$ 时,说明车辆在离开此 RSU 通信覆盖范围前不能接收到返回的计算结果,则将此任务卸载给相邻的 RSU,再由相邻的 RSU 进行计算并将计算结果发送给目标车辆.通过上述步骤就可以判断出需要卸载给相邻 RSU 的任务,从而减少这类任务的完成时间,提高车辆在行驶过程中的安全性,使车载应用更加稳定.RSU 进行判断的伪代码如算法 2 所示.

算法 2. RSU 判断任务类型的算法

输入:车辆的剩余行驶时间 t_{ru}

输出:TRUE or FALSE

①while(TRUE)

② if($t_{ru} > t^{e2e}$)

③Forward_To_CPU(TASK);/ * 任务在本地执行 * /

④ return(FALSE);

⑤ else

⑥Forward_To_Adjacent_RSU(TASK);/ * 任务卸载给相邻 RSU * /

⑦ return(TRUE);

⑧ end if

⑨end while

2.5 延时计算方法

根据基本的网络传输模型可以得出 RSU 协作通信结构中任务的通信延时包括处理延时 t_{proc}、传输延时 t_{trans}、排队延时 t_{queue} 和传播延时 t_{prop}, 排队延时和处理延时是由网络环境决定的,传输时延是由传输的包的大小和传输速率决定的.车辆 u 的任务 m 的通信延时计算方法如式(12)所示.

$$t^c_{um} = t_{\mathrm{proc}} + t_{\mathrm{trans}} + t_{\mathrm{queue}} + t_{\mathrm{prop}} + mt'_{\mathrm{prop}}. \tag{12}$$

式中:t'_{prop} 为消息从当前 RSU 传播到相邻 RSU 所产生的延时.当消息从当前 RSU 传输到相邻 RSU 时,m 的值等于 1,当消息只在当前 RSU 内进行传播时,m 的值等于 0.

若相邻的 RSU 没有进行通信,则当车辆在当前 RSU 通信覆盖范围发送任务计算请求后离开它的通信覆盖范围,车辆只能等到此条请求的时间耗尽,再进行超时重传.此时这个任务计算请求从第 1 次发送到完成的总时间包括点到点延时和超时重传时间 Timeout 两部分,计算方法如式(13)所示.

$$t_{\mathrm{unsuc}} = t^{e2e}_u + \mathrm{Timeout}. \tag{13}$$

式中:t^{e2e}_{ui} 为车辆 u 任务的点到点延时.

3 仿真建立和分析

3.1 仿真建立

仿真使用城市交通模拟(Simulated of Urban Mobility,SUMO)作为交通模拟器,使用 OMNet++来进行网络的仿真.使用 veins 框架将 2 个平台结合在一起使用,该框架 MAC 层基于 802.11 协议且开发完善.本研究分 2 个场景进行分析,一个场景中相邻的 RSU 不进行连接,另一个场景中将相邻的 RSU 通过光纤连接且可以互相发送消息,2 个场景中 RSU 的分布满足相邻 RSU 的通信覆盖范围不重叠且相切.表 1 列出了仿真设置的重要参数.

表 1 仿真参数设置

参数	取值
RSU 最大通信距离/m	500
车的数量	20/40/80
车辆速度/(km/h)	60/80/100/120
RSU 数量	5
云服务器的数量	1
仿真时间/s	300

3.2 结果分析

我们分别对 2 个场景进行多次仿真并对仿真结果进行分析.首先,针对 RSU 没有连接的场景,在仿真场景中设置 40 辆汽车,并设置车辆的速度为 100 km/h.根据仿真结果可以明显地看出车辆在进行 RSU 的切换时,会发生延时增加的情况(如图 3 所示).车辆和 RSU 有稳定连接时消息的延时在 0.02 s 左右.当车辆将计算任务卸载给 RSU 后,在接收到计算结果前就驶离了此 RSU 的通信覆盖范围时,任务的延时会增加.这是由于此时车辆没有得到任务计算的结果,导致车辆等待一个超时重传时间后再次发送此任务的计算请求.此任务从第 1 次发送请求到得到计算结果所花费的时间会更多,本算法的主要目的是减少这类任务的完成时间.

为了计算车辆进行 RSU 切换时上述情况发生的概率,分别设置了不同数目的车辆在 100 s 有效的仿真时间内进行仿真,仿真结果如图 4 所示.从图 4 中可以看出:在相同的仿真时间内,不同的车流量和不同的车速对结果的影响是不同的.车辆的速度越快需要相邻 RSU 协作的任务就越多.当车辆维持在相同的速度时,在相同时间内经过的车辆越多,需要 RSU 协作的任务就越多.其中可以看出车辆速度在 80~100 km/h 时,需要 RSU 协作的任务数量有一个增加的趋势.说明在速度较高的路段更需要相邻 RSU 协作来减少这类任务的延时,从而减少事故发生的概率.

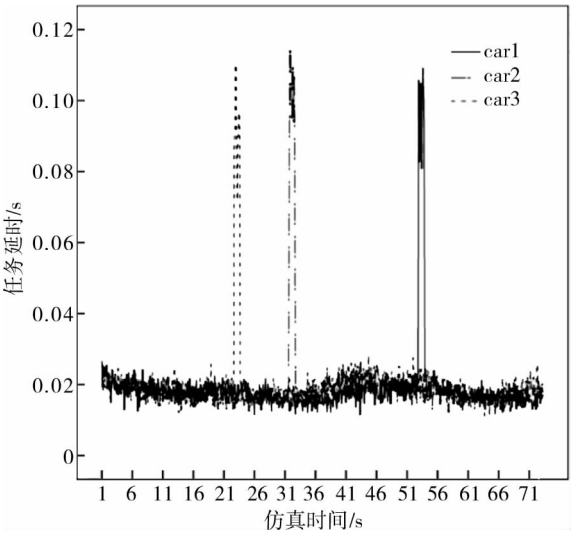

图 3 车辆任务延时

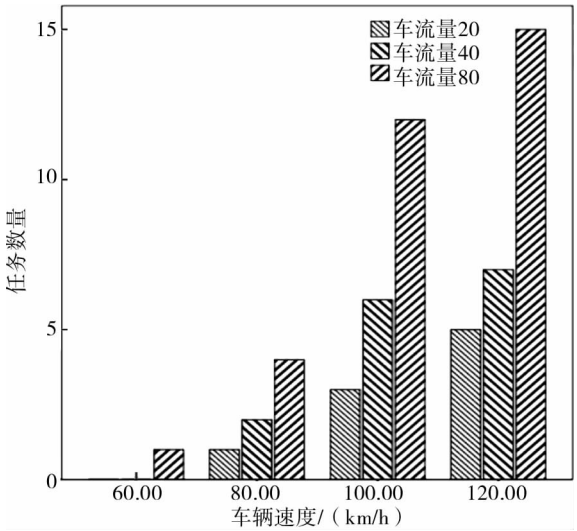

图 4 不同条件时需要相邻 RSU 转发的消息数量

在仿真实验中,针对这类任务比较了 RSU 连接时和 RSU 未连接时这 2 种情况下,这类任务的完成时间.当相邻 RSU 没有进行连接时,由于车辆接收不到任务的计算结果,导致任务的完成时间增加.当相邻 RSU 进行连接时,利用相邻 RSU 之间的相互协作,车辆可以在下一个 RSU 中接收计算结果,可以有效地减少任务的完成时间.其中,当 RSU 没有进行协作时,任务完成时间包括超时重传时间和消息点到点的延时两部分.为了比较本文提出的这类任务的完成延时,截取有效的数据进行对比,对比结果如图 5 所示.图 5 是这类任务 2 种场景下的完成延时对比.

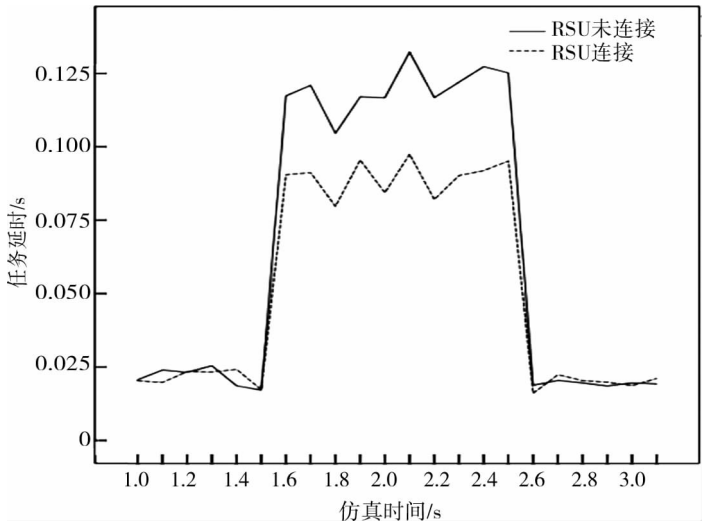

图 5　平均任务完成时间的对比

从仿真结果可以看出:相邻 RSU 之间进行协作通信,可以减少这类任务完成的时间.基于车辆和 RSU 之间的距离来计算车辆在当前 RSU 通信范围剩余时间的算法是有效的.相较于 RSU 没有进行协同工作的场景,本文提出的通信模型和算法在延时上改善了 25%,这可以很大的提高车载应用的性能.

4　结论

1)提出了一种基于 M/M/S 排队模型的 RSU 协作通信策略,为任务分类与卸载提供数据传输支持.

2)提出了一种车联网短文本特征权值计算算法,为任务分类提供依据.

3)在上述基础上,提出了一种基于剩余行驶距离和时延估计任务卸载算法.

参考文献:

[1] Zhang K, Leng S P, He Y J, et al. Mobile Edge Computing and Networking for Green and Low-Latency Internet of Things[J]. IEEE Communications Magazine, 2018, 56(5): 39-45.

[2] Wang C, Li Y, Jin D, et al. On the Serviceability of Mobile Vehicular Cloudlets in a Large-Scale Urban Environment[J]. IEEE Transactions on Intelligent Transportation Systems, 2016, 17(10): 2960-2970.

[3] 胡海洋, 刘润华, 胡华. 移动云计算环境下任务调度的多目标优化方法[J]. 计算机研究与发展, 2017, 54(9): 1909-1919.

[4] 周悦芝, 张迪. 近端云计算:后云计算时代的机遇与挑战[J]. 计算机学报, 2019, 42(4): 677-700.

[5] Johnston S J, Apetroaie-Cristea M, Scott M, et al. Cox. Applicability of commodity, low cost, single board computers for Internet of Things devices[C]// IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, USA, 2016: 141-146.

[6] Lin L, Liao X, Jin H, et al. Computation Offloading Toward Edge Computing[J]. Proceedings of the IEEE, 2019, 107(8): 1584-1607.

[7] Ahmed E, Rehmani M H. Mobile Edge Computing: Opportunities, solutions, and challenges[J]. Future Generation Computer Systems,2017, 70: 59-63.

[8] Zhao J, Li Q, Gong Y, et al. Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(8): 7944-7956.

[9] Yi C, Cai J, Wang R, et al. Computation Offloading Game for Edge Computing with Strategic Local Pre-Processing Time-Length[C]// 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall).Victoria, BC, Canada, 2020: 1-5.

[10] Zheng J, Cai Y, Wu Y, et al. Stochastic computation offloading game for mobile cloud computing[C]// 2016 IEEE/CIC International Conference on Communications in China (ICCC). Chengdu, China, 2016: 1-6.

[11] 齐彦丽, 周一青, 刘玲, 等. 融合移动边缘计算的未来 5G 移动通信网络[J]. 计算机研究与发展, 2018, 55(3): 478-486.

[12] Wu Y, Qian L, Ni K, et al. Delay-Minimization Nonorthogonal Multiple Access Enabled Multi-User Mobile Edge Computation Offloading[J]. IEEE Journal of Selected Topics in Signal Processing, 2019, 13(3): 392-407.

[13] Wang Q, Guo S, Liu J, et al. Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing[J]. Sustainable Computing: Informatics and Systems, 2019, 21: 154-164.

[14] Boukerche A, Soto V. An Efficient Mobility-Oriented Retrieval Protocol for Computation Offloading in Vehicular Edge Multi-Access Network[J]. IEEE Transactions on Intelligent Transportation Systems, 2020, 21(6):2675-2688.

[15] Dai Y, Xu D, Maharjan S, et al. Joint Load Balancing and Offloading in Vehicular Edge Computing and Networks[J]. IEEE Internet of Things Journal, 2019, 6(3): 4377-4387.

[16] 杨紫淇, 蔡英, 张皓晨, 等. 基于负载均衡的 VEC 服务器联合计算任务卸载方案[J]. 计算机科学, 2021 ,48(1): 81-88.

[17] Yuan H, Zhou M. Profit-Maximized Collaborative Computation Offloading and Resource Allocation in Distributed Cloud and Edge Computing Systems[J]. IEEE Transactions on Automation Science and Engineering, 2020: 1-11.

[18] Wang Z, Zheng S, Ge Q, et al. Online Offloading Scheduling and Resource Allocation Algorithms for Vehicular Edge Computing System[J]. IEEE Access, 2020, 8: 52428-52442.

[19] Zhang K, Mao Y, Leng S, et al. Mobile-Edge Computing for Vehicular Networks: A Promising Network Paradigm with Predictive Off-Loading[J].IEEE Vehicular Technology Magazine, 2017,12(2): 36-44.

[20] Zhao H, Zhu Y, Ding Y, et al. Research on Content-aware Classification Offloading Algorithm Based on Mobile Edge Calculation in the Internet of Vehicles[J]. Journal of Electronics & Information Technology, 2020,42(1):20-27.

[21] Wu T Y, Obaidat M S, Chan H L. Quality Scan scheme for load balancing efficiency in vehicular ad hoc networks (VANETs)[J]. Journal of Systems and Software, 2015, 104: 60-68.