doi:10.3772/j.issn.1002-0470.2013.02.003

无线传感器网络的高能效的任务实时处理方法①

韩光洁②[*,**] 张 娜[***] 董玉慧[*] 徐勇军[****] 王 鹏[****]

([*]河海大学计算机与信息学院 常州 213022)
([**]常州市传感网与环境感知重点实验室 常州 213022)
([***]江苏省软件产品检测中心 南京 210008)
([****]中国科学院计算技术研究所 北京 100190)

摘 要 针对无线传感器网络的任务协作处理的需求,提出了一种基于协作的复杂任务求解方法(CTSC),该方法首先根据任务图将划分后的任务模块进行分组,从而能够缩短任务响应延迟和有效减少通信能耗;其次基于一种节点等级域模型招标选取协作节点和确保任务分配的最优化,从而保证任务的高效处理和实现节点能耗均衡。通过仿真实验进行的与现有方法的比较的结果表明,CTSC 方法更适合传感器网络处理复杂任务,该方法可以有效缩短节点的任务响应时间,并且能量消耗较少。

关键词 无线传感器网络,协作,任务,等级域,招标协议

0 引 言

对于无线传感器网络来说,一个给定的复杂任务需要被分解成多个任务模块并分配给不同的传感器节点进行处理,直到整个复杂任务处理完成。合理的分布式计算可以保证算法的可靠性,并且避免将大量任务处理工作分配给单一节点而使其能量耗竭,因此节点能耗的均衡得到了广泛关注和研究[1-6]。Yang 与 Prasanna[7]提出了一种能量均衡的任务分配算法,该算法在满足最小化能量消耗的同时达到能量均衡的目的。Sekhar 等人[8]提出一种基于 A* 的算法,该算法将任务分配给大量传感器节点,同时考虑到节点的能量受限问题,提出了贪婪 A*(Greedy A*)算法来减少 A* 算法最优分配方案的复杂度。李志刚等人[9]针对同构节点组成的网络中提出了一种高能源效率的任务分配算法,且将处理任务分配作为二次 0-1 规划问题,提出了分布式逐层优化分配算法,实现了任务处理的能量优化。Abdelhak 等人[10]提出了一种任务分组方法,该方法可以保证任务的并行处理。以上的研究大都在同构网络中采用协作式任务处理来减少能量消耗,大都没有考虑在实时性要求较高的应用中如何实现任务的实时响应,在异构网络环境下也未能很好地最优化任务处理时间和系统能量消耗。本研究将传感器网络的任务处理问题定义为减少任务的响应时间、均衡节点能量消耗的最优化问题,针对一跳异构网络环境,提出了一种基于协作的复杂任务求解方案(complicated task solution scheme based on cooperation, CTSC)。CTSC 是一种基于任务模块分组计算和节点等级域的处理方案,它能够确保系统内节点能量均衡消耗和缩短任务响应时间。该方案包含两个部分:任务分组和节点等级划分。任务分组是将划分后的各个任务模块,参照其关系图划分到若干小组,以满足实时性要求;等级划分是定量节点的任务处理性能,提高协作节点选取效率,保持节点能量消耗的平衡性,有效延长系统的生命周期。

1 网络模型与问题描述

假设网络中任意两个节点能够互相通信。各个节点的能量和计算能力各不相同,这些节点构成一个异构网络。当网络中某个节点检测到一个任务,根据该任务信息量的大小判断其是否为复杂任务,若是,则采用 CTSC 方案处理该复杂任务,使得复杂

① 国家重大科技专项(2010ZX03006-002),国家自然科学基金(61173132),教育部留学回国人员科研启动基金,江苏省输配电装备技术重点实验室开放基金(2010JSSPD04)和常州市传感网与环境感知重点实验室开放基金(CZSN201101)资助项目。
② 男,1972 年生,博士,副教授;研究方向:传感器网络,计算机网络;联系人,E-mail: hanguangjie@gmail.com
(收稿日期:2012-04-16)

任务能够快速响应并且保证各个节点在任务处理的过程中实现能量均衡消耗，从而达到延长系统生命周期的目的。相关的模型具体定义如下：

1.1 节点模型

用图 $A=(S,E)$ 来表示传感器节点模型，S 表示节点的集合，E 是边的集合。每个传感器节点 $S_i \in S(0 \leqslant i \leqslant n)$ 有三个属性 $RE[i]$、$e_{comp}[i]$ 和 $PROC[i]$，它们分别表示该节点的剩余能量，单位比特数据的计算能耗和处理速率。任意两个节点 s_i 和 s_j 之间的边 $e_{ij} \in E(0 \leqslant i, j \leqslant n)$ 有一个属性 WE_{ij}，代表节点 s_i 和 s_j 之间的单位比特通信所消耗的通信能量。

1.2 任务模型

在某节点感知到任务 T 后，将该任务划分为 m 个任务模块 $t_0, t_1, \cdots, t_{m-1}$。用图 $B=(T,W)$ 来表示任务模型，其中 B 是任务模块的集合，W 是任意两个任务模块之间加权边的集合。每个任务模块 $t_x \in T$ 都有一个属性 $DC[i]$，代表每个任务模块所需处理信息的比特数。每个加权边 W_{xy} 表示任意两个任务模块 t_x 和 t_y 相互通信的信息量。

1.3 相关定义

定义 1 感知节点：传感器网络中检测到任务到达的节点，用符号 S_0 来表示。

定义 2 控制节点：由感知节点从邻居节点中挑选出来负责节点等级划分与协作节点选取等工作的节点，用符号 S_c 来表示。

定义 3 协作节点：由控制节点挑选出来处理任务的节点，用符号 S_b 来表示。

定义 4 复杂任务：单个节点的能量不足以用来处理的任务，或是由单个节点处理会消耗过多时间和能量的任务，用符号 T 来表示。

定义 5 信息量：任务需要处理的信息大小，这里的信息是节点采集到的系统感兴趣的目标特性的相关数据，可以量化地看作是目标信息数据包的大小，用符号 M 来表示，单位为比特。

定义 6 等级域：节点的任务处理性能根据其任务处理能耗、任务处理成功率和处理速率的不同而各不相同。将节点划分为不同的等级，同一等级的节点的处理性能大致相同，这样的一个等级称为一个等级域。用符号 N 来表示。

2 复杂任务协作处理算法

复杂任务协作处理算法的主要流程，如图 1 所示。

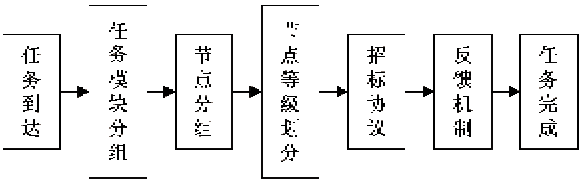

图 1 CTSC 算法流程图

2.1 任务模块分组

为了降低系统通信能耗，将相互联系紧密的任务模块分配给相同的节点处理，同时将相互之间没有联系的任务模块划分给不同的节点来处理，可以减少任务处理的等待时间，保证任务的及时处理。

参照任务模块关系图来对任务模块进行分组，同一父任务的多个子任务被划分到不同组内以保证任务的并行处理，缩短响应时间；在没有被分组的子任务中挑选权重最大的划分到与其父任务同一小组可以减少通信消耗。依据以上两个分组原则，我们将没有父任务仅有子任务的模块划分到第一组 G_0，并将其权值最大的子任务划分到同一组，将其他子任务各划分到不同组内。这些新进入分组的子任务进入分组以后自动从子任务升级为父任务，并要求其权值最大的子任务划分到同一组内，此阶段存在同一个子任务被要求加入不同分组的情况，因此在加入分组前，子任务进入与自身权重最大的父任务同一组内，而被淘汰掉的父任务则重新要求其权重次大的子任务加入小组。依据分组原则将所有任务模块划分到组内，再根据分组原则判断是否产生冗余的分组，若有则将其归并到与其权值最大的组内。为了计算方便，我们假设最后将任务模块划分为 g 个小组 $G_0, G_1, \cdots, G_k, \cdots, G_{g-1}$。

每一个任务小组里的任务模块由同一个节点来处理，g 个任务小组就由 g 个节点来处理。为了避免感知节点能量耗竭，能量最大的 g 个节点被挑选出来作为控制节点，分别负责 g 个协作节点的挑选工作，从而来分担感知节点的工作负担。每个控制节点拥有一个节点列表 N_c，用来存放可供挑选的节点号，并且保证任意两个控制节点的节点列表里没有重复的节点号，而且除了感知节点和控制节点以外的所有节点号都存在这些节点列表里。此外，为了使得节点能量均衡消耗，每个控制节点的节点列表里节点的个数和节点总能量应保持均衡。最后通过判断节点能量是否足够用来处理任务来决定任务

分组是否合理。如果控制列表中每个节点的能量都不足够用来处理任务,则表示该任务小组包含的任务模块过多,再根据任务分组原则将该任务小组中最后加入的任务模块调整到其它任务小组中,直到存在能量足够的节点为止。

2.2 节点等级划分

节点等级划分阶段通过预先挑选出能耗合理且正确率高的节点,以便任务模块的分配,从而达到均衡各节点能量消耗,延长网络生命周期,提高任务处理正确率的目的。影响节点性能的因素包括很多方面,其中两个主要的因素是节点的剩余能量和节点任务处理的成功率。本文算法优化了文献[8]中的方法,公式化各个节点任务处理的能量消耗,构造一个变量 P 来表示节点处理任务的性能。m 个任务模块被划分为 g 个任务小组,用 $m \times g$ 阶矩阵 X

$$\boldsymbol{X}[x][k] = \begin{cases} 1, t_x \text{ 被划分到 } G_k \text{ 中} \\ 0, \text{其他} \end{cases} \tag{1}$$

来表示任务模块的分组情况,用 $g \times n$ 的矩阵 Y

$$\boldsymbol{Y}[k][i] = \begin{cases} 1, G_k \text{ 被分配给 } S_i \\ 0, \text{其他} \end{cases} \tag{2}$$

表示任务小组的分配情况,这样,矩阵 $\boldsymbol{Z} = \boldsymbol{X} \times \boldsymbol{Y}$ 是表示任务模块分配情况的 $m \times n$ 阶矩阵。

用 $n \times n$ 阶矩阵 $\boldsymbol{Q}$ 表示任意两个任务模块之间的相互通信量,其中 $Q_{ij} = W_{xy}$。任务小组 k 分配给节点 i 处理,节点 i 消耗的计算能量为

$$\boldsymbol{E}_{\text{comp}}[i] = \sum_{x=0}^{m} \boldsymbol{X}[x][k] \cdot \boldsymbol{DC}[x] \cdot e_{\text{comp}}[i] \tag{3}$$

任务小组 k 分配给节点 i 处理,节点 i 消耗的通信能量为

$$\boldsymbol{E}_{\text{comm}}[i] = \sum_{y=0}^{m-1} \sum_{x=0}^{m-1} A_{xy} \tag{4}$$

其中

$$\boldsymbol{A} = \sum_{j=0, j\neq i}^{n-1} \begin{bmatrix} Z_{0i} \times Z_{0j} & Z_{0i} \times Z_{1j} & \cdots \\ Z_{1i} \times Z_{0j} & Z_{1i} \times Z_{1j} & \cdots \\ \vdots & \vdots & \ddots \\ Z_{(m-1)i} \times Z_{0j} & Z_{(m-1)i} \times Z_{1j} & \cdots \end{bmatrix} \begin{bmatrix} Z_{0i} \times Z_{(m-1)j} \\ Z_{1i} \times Z_{(m-1)j} \\ \vdots \\ Z_{(m-1)i} \times Z_{(m-1)j} \end{bmatrix} \times \boldsymbol{Q} \times e_{ij} \tag{5}$$

任一节点能够处理这个任务小组的前提条件是节点当前阶段的剩余能量大于等于处理任务消耗的能量,用下式表示:

$$\boldsymbol{E}_{\text{comp}}[i] + \boldsymbol{E}_{\text{comm}}[i] \leqslant \boldsymbol{RE}[i] \tag{6}$$

为了实现节点能量均衡消耗,在任务分配的过程中不仅要考虑节点处理任务消耗的能量,还要考虑节点的当前剩余能量。除此之外,在挑选节点时还需要考虑节点的可靠性,节点处理任务的成功率越高则其越可靠。将定量分析节点任务处理性能问题看作为能耗和成功率的最优化问题,可以用变量 P 的大小来表示节点 s_i 任务处理性能的优劣。

用 $1 \times n$ 的一维数组 $n_{\text{success}}[i]$ 表示节点 s_i 的任务处理的成功率,用 $1 \times n$ 的一维数组 $n_{\text{total}}[i]$ 表示节点 s_i 累积处理任务的总次数。综上,令

$$P = \frac{1}{2}\alpha \frac{n_{\text{success}}}{n_{\text{total}}} + \frac{1}{2}\beta \frac{RE[i] - E_{\text{comp}}[i] - E_{\text{comm}}[i]}{RE[i]} \quad (0 \leqslant \alpha, \beta \leqslant 1) \tag{7}$$

其中 α, β 是权重值,可以通过调节 α, β 的大小来协调能量与可靠性这两个性能指标,用来保持系统的稳定性。如果假设节点能量充足不需要考虑能量因素,则可令 β 为零;如果节点都是可靠的,只需考虑能量因素,则可令 α 为零。

由于 $\dfrac{RE[i] - E_{\text{comp}}[i] - E_{\text{comm}}[i]}{RE[i]} \in [0,1]$, $\dfrac{n_{\text{success}}}{n_{\text{total}}} \in [0,1]$, 可知 $P \in [0,1]$, 因此 p 值越大,节点被挑选的概率越大。计算节点任务处理性能参数之前,先判断节点剩余能量是否足够用来处理任务,并将能量不足够用来处理任务的节点划分为第三等级,其次根据 P 值的大小将节点划分为两个等级,其中 $P \in [0.5, 1]$ 的节点被划分为第一等级, $P \in [0, 0.5)$ 的节点划分为第二等级。节点的等级并不是固定不变的,而是随着 P 值的大小而变化的。节点的等级划分完成后,任务的分配总是优先挑选第一等级的节点,所以在第一等级中节点足够的情况下,分配方案不会考虑第二等级与第三等级中的节点,从而降低了通信与计算成本。

2.3 招标协议

CTSC 采用招标的方式来确定最终的协作处理任务节点,选取任务完成时间最短的节点作为协作节点。不同于传统意义上的泛洪广播标书的招标协议,本方案中标书的招标范围被限制在第一等级的节点中,当且仅当第一等级中的所有节点都被无法成功完成任务时,招标范围会转换为第二等级的节点。通过这种缩小招标范围的方法,可以节省通信能耗。协议中所涉及的相关概念定义如下:

定义 7 任务完成时间(Execution Time):控制节点将任务相关描述发送给节点的时间、节点计算任务所用的时间以及节点将计算结果返回给控制节点的时间总和称为任务完成时间,用符号 t_{exc} 来表示,中标节点的任务完成时间,用符号 t_{exc_b} 来表示。

定义 8 任务失败:控制节点在 t_{exc_b} 时间内未收到中标节点返回的任务处理结果或者收到的任务处理结果不完整,则视为任务处理失败。

定义 9 候补节点:第一等级和第二等级节点中没有被挑选出来作为协作节点的其他节点都被视为候补节点,在中标节点任务处理失败后,控制节点从候补节点中重新选择节点作为协作节点来完成任务。

通过招标的方式,挑选第一等级中任务完成时间最短的节点作为协作处理节点。当然,在某段时间 t_{wait} 内存在节点未能及时返回处理结果,此时控制节点向该部分节点重新投放标书并等待返回信息。由于时间和能量的限制,不能无限地重复投递标书,否则会影响任务处理的效率,产生较大时延和能耗。具体招标协议的伪代码,如图 2 所示。

```
① N=N_1 ∩ N_C; I=∅
② a=0
③ INFORMATION={S_C_ID, bit of t}   //招标信息构数据包
④ MULTICAST(INFORMATION)   //组播招标信息
⑤ if (S_i∈N and it received INFORMATION)
       Insert S_i to I
       Compute the t_exc   //节点收到信息并投标
⑥ wait t_wait
⑦ if (I=N)
       S_k = S_i(S_i has the minimum t_exc)   //确定中标节点
⑧ MULTICAST(CONFIRM)
⑨ else
       REPEAT UNICAST(INFORMATION)
       a=a+1   //重新发送招标信息
⑩ end while (a=k or I=N)
       Receive(CONFIRM)
       END
```

图 2 招标协议伪代码

N_1 表示第一等级节点的集合,N_C 表示控制节点邻居列表内节点的集合。控制节点通过组播的方式发送招标信息,节点收到后将各自的标书返回给控制节点。当控制节点收到所有邻居节点的返回信息或招标次数达到最大招标上限时,控制节点挑选任务完成时间最短的节点作为协作节点,并广播节点确认信息。

2.4 反馈机制

控制节点在 t_{exe_b} 内未收到中标节点的任务反馈信息,则任务处理失败,该中标节点成为失败节点,不能参加本轮任务处理但是可以参加下一次任务处理的招标。失败节点的协作任务处理总次数累加一次,成功任务处理次数保持不变。

任务失败后,控制节点要求与失败节点等级相同且任务完成时间仅次于失败节点的候补节点升级为中标节点,由当前时刻的中标节点重新处理任务。如果任务成功处理,则该中标节点的协作任务处理总次数和成功任务处理次数各累加一次,否则重新升级下一个候补节点来处理任务,直到成功处理任务。

3 仿真实验及性能评价

本文对 CTSC 算法和基于文献[10]中 Greedy A * 算法在协作系统任务处理效率以及系统性能等方面进行了有针对性的比较。采用 NS2 仿真平台,设定网络环境为 100m × 100m 的区域,网络内均匀分布若干异构传感器节点,这些节点的任务处理速度和能量消耗不同并且服从随机分布,节点两两之间能够相互通信,即为一个一跳网络。设定坐标(50,50)处的节点为感知节点,系统随机产生一个任务模块关系图作为待处理的复杂任务,并由感知节点读取该任务关系图开始协作系统的仿真实验。具体的仿真参数,如表 1 所示。

表 1 仿真参数

参数	数值	参数	数值
MAC 层协议	802.11	单位计算时间消耗	[0,1]/s
无线传播模型	TwoRay Ground	节点初始能量	[0,50]/J
无线路由协议	动态源路由协议	任务通信数据量	[0,100]/B
感知节点坐标	(50,50)	单位计算能耗	[0,1]/J
单位通信时间消耗	[0,3]/s	单位通信能耗	[0,3]/J

当网络中有 5 个节点时,我们对文献[10]中的 Greedy A * 和 CTSC 进行仿真分析,它们处理不同大小的复杂任务的时间消耗情况,如图 3 所示。

图 3 中当处理一个相对较小的任务,这个任务被划分成 3 个大小相等的任务模块时,两种方法的任务处理时间相同;处理另一个被划分成 4 个大小相等的任务模块时,两种方法的任务处理时间依旧

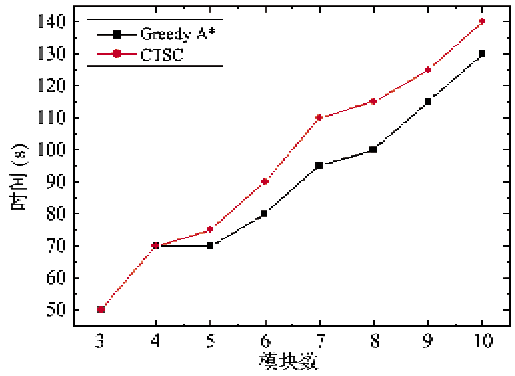

图 3　5 个节点的任务协作完成时间

相同,表示在以上两种情况下,具体的任务分配结果一样,这时任务处理时间相同。当较大的复杂任务划分后的任务模块较多而系统中只有 5 个节点时,CTSC 的任务处理时间比 Greedy A* 稍长,这是因为当只有 5 个节点,CTSC 在选择协作节点前,先选中若干个节点作为控制节点来分担感知节点的复杂任务,共同来承担计算各个节点处理任务的能耗、节点的等级划分以及任务分配的工作,这样实际用来处理任务的节点个数相应减少,容易出现排队等候处理的现象,从而导致响应延迟。

当网络中有 10 个节点时,这两种方法处理复杂任务的时间消耗情况,如图 4 所示。

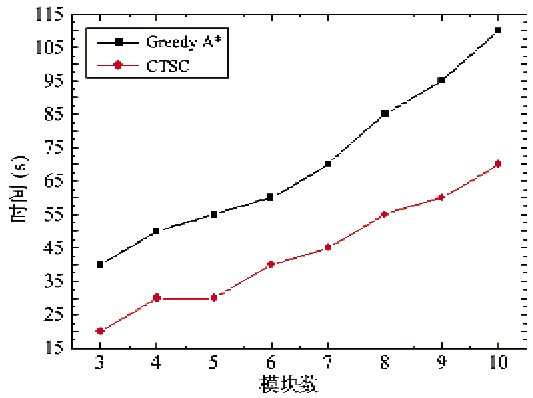

图 4　10 个节点的任务协作完成时间

由图 4 可以看出 CTSC 的处理速度总是比 Greedy A* 快。这是因为 CTSC 在模块分组时将互不关联的任务模块划分到不同的任务小组,任务模块的并行处理能够保证任务的及时响应,从而缩短任务的处理时间。通过比较图 3 和图 4,可以看出系统中节点越多,CTSC 处理任务的效率越高,在减少任务响应时间方面取得明显进步,更适合在大规模网络中实时处理复杂任务。

当网络中有 5 个节点时,这两种算法在处理复杂任务每个节点的能量消耗情况如图 5 所示。

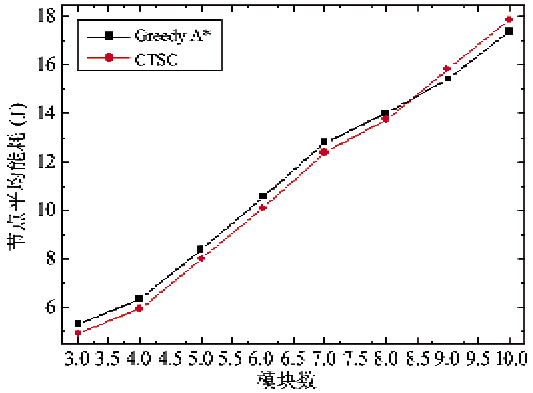

图 5　5 个节点协作处理任务的节点平均能耗

由图 5 中可以看出,当任务模块小于 9 时,CTSC 的节点平均能耗比 Greedy A* 小,这是因为 CTSC 将节点进行等级划分,每次任务分配算法总是在第一等级域中选择,这样减少了任务分配方案的个数,在减小算法计算消耗的同时减少了任务分配阶段的通信消耗。然而当任务变大,划分的任务模块个数增多时,CTSC 任务排队等待现象出现导致处理时间延长,节点的空闲时间所消耗的能量增多,导致节点平均能耗增大。

当网络中有 10 个节点,用这两种方法处理复杂任务时,每个节点的能量消耗情况,如图 6 所示。

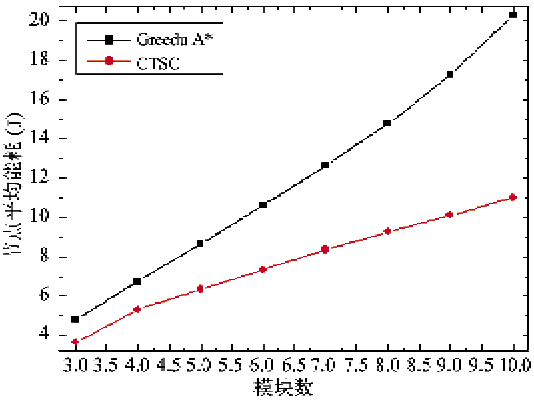

图 6　10 个节点协作处理任务的节点平均能耗

由图 6 中可以看出 Greedy A* 任务处理的节点平均能耗总是比 CTSC 的能耗大,且随着任务的增大和任务模块的增多,CTSC 的优势越来越明显,这是因为 CTSC 在任务模块分组阶段将通信代价较大的任务划分至同一个任务小组,由同一个节点处理这样的任务模块,避免了节点间过多的通信能耗。

同时 CTSC 任务分配算法总是在第一等级域中选择,减少任务分配方案的个数。当处理较大任务时,随着任务模块的个数增多,它们之间的通信量越来越大,通过等级域的划分能够避免不必要的通信消耗。

当利用 CTSC 和 Greedy A* 分别处理 8、15 和 20 个不同任务时,每个算法的迭代次数及每次迭代的能量消耗的比较结果如图 7 所示。

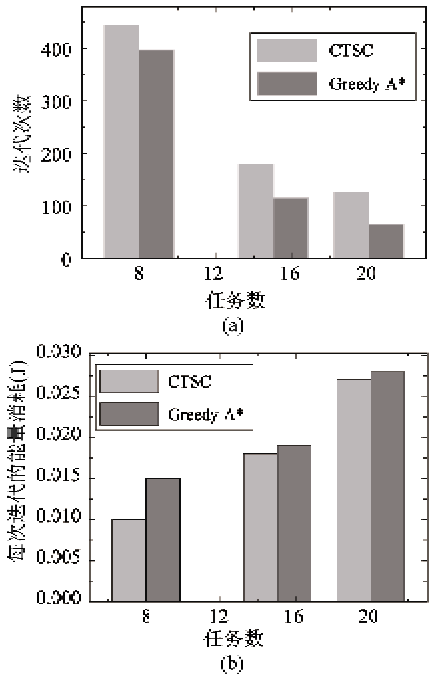

图 7 CTSC 和 Greedy A* 的完成任务的性能比较

从图 7 中可以看出,CTSC 算法能提高 55% 的网络寿命,并且随着任务的增加 CTSC 算法表现的更优越,这是由于 Greedy A* 中感知节点在任务分配过程中需要进行大量的计算和通信,来挑选出最优的节点协作处理任务,因此感知节点承担了更多的工作。而 CTSC 算法采用了等级域模型和任务图的分配方式。从图 7(b) 中可以看出 CTSC 算法比 Greedy A * 算法每次迭代可以节省大概 1.34% 的能量消耗。

当节点数目为 30、35 和 40 时,CTSC 和 Greedy A* 两个算法的迭代次数及每次迭代的能量消耗的比较,如图 8 所示。

由图 8 可见,CTSC 算法比 Greedy A * 算法能提高 45% 的网络生命周期,并且每次迭代可以节省大概 4.62% 的能量消耗。这意味着 CTSC 在任务协作处理的过程中较好地保证了所有节点能量的均衡消耗,这是因为CTSC在任务分配之前由感知节点挑

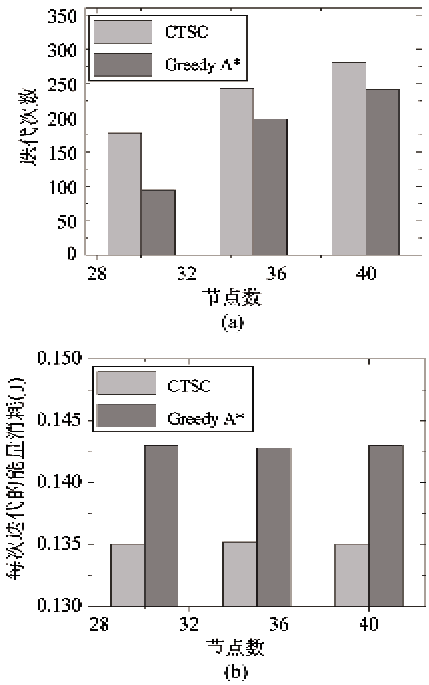

图 8 CTSC 和 Greedy A * 中节点数不同时的性能比较

选出几个控制节点,由这些节点负责计算节点的任务处理性能并与其他节点通信,从而分担了感知节点的工作负担,避免感知节点任务过重而能量耗竭。

通过以上仿真比较,证实 CTSC 相对于 Greedy A * 在任务处理时间和节点能量消耗两方面都取得明显进步,CTSC 尤其适合在大规模网络环境中处理复杂度较高的任务。通过合理的并行处理策略和缩小任务分配范围,实现了任务实时响应,减小了通信消耗。感知节点的工作也均匀分配给其它节点,保持了节点能量的均衡消耗。

4 结 论

本文提出了基于协作的复杂任务处理方案 CTSC,该方案在保证任务高效处理的同时使得节点能耗均衡,延长网络生命周期,实现系统性能优化。首先,通过任务模块的分组处理,将通信紧密的任务模块分配给同一个节点处理,减少节点之间的通信能耗,把可以并行处理的任务模块分配给不同的节点来处理,尽量缩短任务的响应时间。同时通过对节点等级的划分与缩小招标范围来优化任务分配方案,实现节能和均衡节点能耗的目的。并与 Greedy A* 算法比较,证明 CTSC 算法的可以有效减少任务响应时间和延长系统的生命周期。

参考文献

[1] Yuan T, Boangoat J, Ekici E, et al. Real-time task mapping and scheduling for collaborative in-network processing in dvs-enabled wireless sensor networks. In: Proceedings of IEEE International Parallel and Distributed Processing Symposium, Rhodes Island, USA, 2006. 25-29

[2] Kartik S, Siva R M C. Task allocation algortithms for maximizing reliability of distributed computing systems. *IEEE Transactions on Computers*, 1997, 46(6): 719-724

[3] Younis M, Akkaya K, Kunjithapatham A. Optimization of task allocation in a cluster - based sensor network. In: Proceedings of the 8th IEEE International Symposium on Computers and Communication (ISCC 2003), Kiris-Kemer, Turkey, 2003. 329-334

[4] Qishi W, Yi G. Supporting distributed application workflows in heterogeneous computing environments. In: Proceedings of the 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2008), Melbourne, Australia, 2008. 3-10

[5] Karimi H, Kargahi M, Yazdani N. Energy-efficient cluster-based scheme for handling node failure in real-time sensor networks. In: Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing, Chengdu, China, 2009. 12-14

[6] Kang H, Li X L. Power-Aware Sensor Selection in Wireless Sensor Networks. In: Proceedings of the 5th International Conference on Information Processing in Sensor Networks, Work-in- Progress, Nashville, USA, 2006

[7] Yang Y, Prasanna V K. Energy-balanced task allocation for collaborative processing in wireless sensor networks. *Mobile Networks and Applications*, 2005, 10(1): 115-131

[8] Manoj B S, Sekhar A, Siva Ram Murthy C. A state-space search approach for optimizing reliability and cost of execution in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 2009, 69: 12-19

[9] 李志刚,周兴社,李士宁等. 传感器网络能源有效任务分配算法. 计算机研究与发展,2009,46(12): 1994-2002

[10] Abdelhak S, Gurram C S, Ghosh S, et al. Energy-balancing task allocation on wireless sensor networks for extending the lifetime. In: Proceedings of the 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), Seattle, USA, 2010. 781-784

An energy-efficient method for real-time processing tasks for wireless sensor networks

Han Guangjie[*,**], Zhang Na[***], Dong Yuhui[*], Xu Yongjun[****], Wang Peng[****]

([*] College of Computer and Information, Hohai University, Changzhou 213022)

([**] Key Laboratory of Sensor Networks and Environmental Sensing, Changzhou 213022)

([***] Software Testing Center of Jiangsu Province, Nanjing 210008)

([****] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

Abstract

To meet the needs of wireless sensor networks for cooperative task processing, a complicated task solution scheme based cooperation (CTSC) is proposed in this paper. The CTSC firstly divides task modules into different groups based on the task graph to reduce the response time of task processing and minimize the communication energy consumption effectively. And then, the collaboration nodes are chosen by means of bid invitation based on a grade field model of sensor nodes. The grade field mechanism can ensure the optimal allocation of tasks to realize the balance of energy consumption and achieve the performance optimization. The comparison results obtained by a simulation experiment show that the CTSC is more suitable for large scale of wireless sensor networks, and it can effectively reduce the response time of task processing with less energy consumption.

Key words: wireless sensor networks, cooperation, task, grade field, bidding protocol