信息物理融合系统的动态多优先级调度

刘纯尧 张立臣
(华东师范大学软件学院 上海 200062)

摘 要 信息物理融合系统(Cyber-physical Systems,CPS)的复杂和异构性给设计者带来了不少挑战,其中任务的多样性使得传统的调度策略不能满足CPS的性能需求。提出了专门针对基于大规模传感器网络的CPS的动态多优先级调度策略。根据任务类型分配4级缓存队列:第1级是来自控制器待处理的实时任务,拥有最高的可抢占式优先级;第2级是来自控制器待转发的实时任务,拥有次高的可抢占式优先级;第3级是来自其他节点待转发的非实时任务,拥有第三高的非抢占式优先级;第4级是来自本地待发送的非实时任务,拥有最低的非抢占式优先级。设计了抢占与非抢占混合的动态调度策略来减少任务的平均等待时间,加入了等待时间阈值机制来保证第4级任务的公平性。通过理论分析和仿真实验对调度策略的性能做了评价。仿真结果显示,动态多优先级调度策略在提高系统性能和稳定性上要优于传统优先级调度。

关键词 信息物理融合系统,动态多优先级,实时/非实时任务,抢占/非抢占式
中图法分类号 TP316 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.006

Dynamic Multi-priority Scheduling for Cyber-physical Systems

LIU Chun-yao ZHANG Li-chen
(Software Engineering Institute, East China Normal University, Shanghai 200062, China)

Abstract The complexity and heterogeneity of Cyber-physical systems (CPS) bring big challenges to system designers. Conventional task scheduling scheme will not satisfy performance requirements of CPS due to the diversity of tasks. A dynamic multi-priority scheduling scheme for a class of CPS based on large-scale sensor network was proposed. In the proposed scheme, tasks reaching each node in the system are divided into four categories: real time tasks to be sent from local node which have the highest preemptive priority, real time tasks to be transmitted from other nodes which have the second highest preemptive priority, non-real time tasks to be processed from controller which have the third highest non-preemptive priority, non-real time tasks to be transmitted from other nodes which have the lowest non-preemptive priority. A mixed preemptive and non-preemptive priority scheduling method was designed to reduce the average waiting time of tasks in each queue. A waiting-time threshold mechanism was added to ensure the fairness of tasks. We analyzed task transmission delay of our scheduling algorithm. Simulation results show that the proposed dynamic multi-priority scheduling scheme outperforms conventional priority queue scheduling scheme.

Keywords CPS, Dynamic multi-priority, Real/non-real time task, Preemptive/non-preemptive

1 引言

信息物理融合系统(CPS)[1-5]是近年来新兴的热点研究课题,它是一种物理过程的计算综合,利用嵌入式计算机和网络来监视和控制物理过程,是物理世界与信息世界的交集,而不是并集[6]。CPS跟嵌入式实时控制系统、传感器网络以及传统的集中或分布式控制系统有很大区别。传统嵌入式实时控制系统是封闭的系统,它的计算能力不对外界开放,然而CPS是网络化的,各子系统协作运行以提高系统的整体性能;传感器网络主要用来监视环境,而CPS除了监视环境还能控制环境;CPS跟传统的集中或分布式控制系统的主要区别在于前者更加强调了实时性[7]。文献[8]给出了一个比较公认的一般化的CPS原型架构,它主要包含了3大部分:控制逻辑、传感单元和制动单元。这种架构体现出了CPS最为核心的性质,包括时效性、分布性、可靠性、容错性、安全性、可扩展性和自治性。CPS的研究领域有不少挑战,主要是因为物理世界的安全和可靠性要求不同于一般的计算,并且物理世界的连续与信息世界的离散很难无缝地结合[9,10]。硬件平台的多样性和应用程序本身及需求的多样性也给CPS软件系统的设计带来了很大挑战[11]。这就要求人们设计良好的资源调度策略来降低由异构问题产生的对系统性能的影响。CPS中的任务是多样性的,可能是实时任务,也可能是非实时任务,可能是周期性任务,也可能是非周期性任务,可能是事件驱动的,也有可能是时间驱动的。这使得传统的调度策略不能满足CPS的性能需求。目前,专门针对这种多学科交叉的CPS资源调度和优化方面的研究还比较少。其中一部分集中

到稿日期:2014-03-20 返修日期:2014-06-11 本文受国家自然科学基金项目(61370082,61173046,91318301)资助。
刘纯尧(1989-),男,硕士,主要研究方向为信息物理融合系统、实时系统,E-mail:lcy7260291@gmail.com;张立臣(1962-),男,教授,主要研究方向为信息物理融合系统和实时系统,E-mail:lczhang@sei.ecnu.edu.cn。

在将控制逻辑和调度相结合，给出改进的控制算法来提高系统的性能和稳定性[12-14]，这种策略只能针对系统规模比较小且物理部分的数学模型较简单的 CPS。对于像基于大规模传感器网络的 CPS，一方面控制算法的设计会变得异常困难，另一方面复杂的算法会严重降低系统的性能，比如能耗的控制。文献[15]将研究对象放到了分布式 CPS(DCPS)上，但是系统并没有控制模块，它只考虑了系统对物理环境(温度)的影响，而不是对物理部分的控制，调度的目的是为了降低温度，或者说是减少能耗，对于实时性要求较高的系统，这种策略就无法满足了。文献[16]给出了一类大规模 CPS 的分布式调度，并且将其应用于无线传感器分组调度、电力系统的车辆充电及数据中心的工作量调度上。单一的基于事件驱动的调度虽然能保证系统的稳定和成本的控制，但是无法提高它的性能。同样，单一的基于时间驱动的调度在保证系统稳定的前提下会浪费很多的资源。CPS 调度的设计涉及到不同领域的知识，我们的研究方向应当是如何搭建起它们之间的桥梁并且能降低耦合性，既能充分发挥各领域专家的特长又能将他们的工作无缝地整合到一块。对于大规模的基于传感器网络的 CPS，任务传输时延对控制算法的影响比较大，这时就要着重考虑设计良好的调度策略来减少传输时延，如果能将时延减小并且使其稳定，那么再让控制领域专家设计相应的控制算法就会变得比较简单。

目前，大多数基于传感器网络的应用都是采用 FCFS 调度策略来传输数据。这种策略实现比较简单，只需要设计一个缓存队列，对传感器的处理能力要求不高。它在传感器的执行效率、成本、节点内存和能耗控制上做到了最优化，适用于实时性要求较低的应用，如环境监测。对于实时 CPS 系统，这种策略难以保证系统的稳定性和安全性。如果采用多优先级队列，势必会造成计算量、节点内存和能耗的增加，如何在两者之间做出一个权衡是个难题。本文为了保证 CPS 的整体性能，仍然设计了 4 级缓存队列，它对传感器的性能要求较高，面对的甚至是下一代高性能传感器网络。节点任务包含了多种类型：实时任务和非实时任务、周期任务和非周期任务。本文根据不同的任务类型设定了 4 级优先级缓存队列，在优先级调度策略的基础上提出了动态多优先级调度，这种策略混合了抢占式调度和非抢占式调度，加入了等待时间阈值机制，可以较好地提高系统的性能和稳定性。

本文第 2 节主要介绍 CPS 的系统框架、优先级队列、任务模型和假设；第 3 节是本文的核心内容，提出了动态多优先级调度策略，并用伪代码表示出来；第 4 节对调度策略做了理论分析；第 5 节做了数值模拟仿真，将动态多优先级调度策略与传统的优先级调度策略做了比较；最后总结全文。

2 CPS 架构与任务队列模型

2.1 CPS 架构

本文对文献[8]提出的 CPS 原形架构做了简化，主要考虑 4 个部分：传感器(Sensors)、计算中心(Computer)、控制器(Controller)和制动器(Actuators)。其中传感器和制动器同时分布在节点上，构成了层次化多跳网络。将与控制器的单跳次数相同的节点分在同一层。离控制器越远的节点等级越高。任务只在相邻层的节点间传输。传感器采集本地数据并通过网络传输到计算中心。计算中心接收数据并作相应的处理和运算，如果发现某些数据不在预期的可接受范围之内，就将此数据发送控制器，控制器通过控制算法计算相应的控制参数，然后再通过网络传输到目的节点，目的节点的传感器接收此实时任务并交给本地的制动器来执行，如图 1 所示。

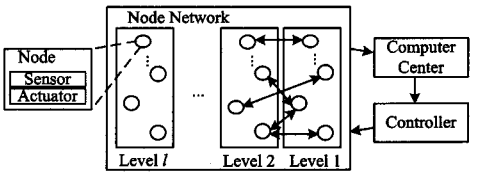

图 1 CPS 架构

2.2 多优先级任务队列模型

网络中的节点与物理环境紧密结合，监视并控制物理过程，但是节点的处理能力包括存储和电能都有限，所以必须将采集到的数据通过网络传输到计算中心，再通过控制器计算控制参数并传输到目的节点；把节点产生的数据定义为非实时的，把控制器传输过来的数据定义为实时的。因此除了第 k 层节点外的每个节点需要做以下 4 种事情：1)将来自控制器的实时数据交给本地制动器执行；2)接收并转发来自控制器的实时数据；3)接收并转发来自其他节点的非实时数据；4)采集本地非实时数据并发送。第 k 层节点由于处在网络边缘，因此不转发数据。假设数据的大小是一样的，并且能一次性传输。为了便于分析，后面用任务来代替前面提到的数据。鉴于有 4 类任务，因此采用四层优先级缓存队列(如图 2 所示)，分别用 Q_1，Q_2，Q_3，Q_4 表示，它们的优先级关系为：$Q_1 > Q_2 > Q_3 > Q_4$。任务到来后根据不同的类型进入不同的缓存区。假设 $Q_1 - Q_3$ 任务的到达服从泊松分布，λ_i 为 Q_i 任务的到达率，$1 \leqslant i \leqslant 3$；$Q_4$ 任务的到达是周期性的，周期为 p；相同队列的任务处理时间是固定的，为 $t_{pro,i}$，$1 \leqslant i \leqslant 4$；任务抢占花费的时间也是固定的，为 t_{switch}。非实时任务的信息包含节点 ID、节点所在的层级和节点监测到的数据。实时任务的信息包含节点 ID、节点所在的层级、控制器计算的控制参数和截止期。

采用这种多优先级任务队列模型有如下好处：实时任务队列 Q_1 和 Q_2 拥有较高优先级，能保证系统的整体性能，如稳定性、安全性；队列 Q_3 和 Q_4 能保证任务的最低平均等待时间，从而减少了传输时延，并且保证了公平性。这些将会在后面的调度策略中有所体现。

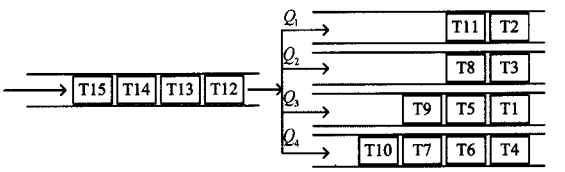

图 2 4 层优先级任务队列

3 动态多优先级调度

CPS 任务的多样性使得传统的调度策略无法满足系统的性能需求。传统的优先级调度分为抢占式和非抢占式两种，如果采用非抢占式调度，一旦某实时任务到达节点时有非实时任务在运行，那么实时任务必须要等非实时任务传输完毕才能执行。如果 $t_{switch} < t_{pro,i}$，非抢占式调度会增加实时任务的等待时间，对于实时行要求较高的 CPS 是不可取的。如果采用抢占式调度，一旦有持续的高优先级任务到达队列，低优先级任务就会处于饥饿的状态。另外，任务抢占所产生的资源交换和上下文切换耗费较大，所以抢占不应该时常发生。

为了解决上述两类调度中存在的问题，本文提出了动态多优先级调度策略，它是抢占与非抢占混合式调度。

系统的稳定和安全是由制动器来实现的，基于上一节的任务模型及假设，实时任务比非实时任务发生的频率小很多，不抢占所造成的等待时间要比抢占所造成的时延大很多，所以我们规定实时任务是可以无条件抢占非实时任务的。实时任务队列内的任务一般很少发生任务竞争的情况，我们采用最简单的先来先服务(FCFS)策略。具体来说，Q_1 和 Q_2 中的实时任务在执行的过程中，如果处理器正在被 Q_3 和 Q_4 中的任务占用，那么直接抢占此任务，进行资源交换，并保存上下文信息。另外，Q_1 比 Q_2 优先级高，可以抢占 Q_2 中的任务，但是这种情况发生得比较少，这在实时任务的到达率较低的前提下是合理的。Q_3 中的非实时任务发生得比较频繁，很容易出现资源竞争现象，一旦出现这种情况，我们比较任务产生的源节点的等级，如果等级较低，即离控制器距离较远，我们让这种任务先传输；如果两者的等级相等，那么随机选取一个先传输。这样可以减少低等级节点任务的传输时延，保证了公平性。Q_4 中的非实时任务是周期性到达的，周期取决于传感器的采样周期，这些任务拥有最低的优先级。一方面，它们可能被来自 Q_1 和 Q_2 的实时任务所抢占；另一方面，如果有来自 Q_3 的任务，那么等任务传输完成后再处理 Q_3 的任务，而不是直接发生抢占，之所以这样做是因为这种情况发生得比较频繁，如果都发生抢占，所耗费的资源交换和上下文切换时间将会很大。如果有来自 Q_4 中的连续任务，那么 Q_3 中的任务会处于饥饿状态。为了避免这种情况，我们设定时间阈值 δ，一旦等待时间超过 δ，并且期间没有来自 Q_1 和 Q_2 的实时任务的抢占，此任务就会抢占 Q_3 中的任务而被执行。需要说明的是，Q_4 中的任务是周期性产生的，它不会导致 Q_3 中的任务处于饥饿状态，因此不必设定等待时间阈值。

本文给出了一个调度实例(如图 3 所示)，Q_4 的任务周期为 5，Q_3 的到来服从泊松分布，到达率为 0.2，即平均每 5 个时间戳到达一个任务，所有任务执行时间都为 2，抢占花费时间 1。实例中 Q_3 的任务分别在时间点 2、5、14、21、22 处到达，Q_2 的任务在时间点 11 处到达，Q_1 的任务在时间点 26 处到达。在时间点 5，Q_3 任务和 Q_4 任务同时到达，Q_3 由于任务优先级高而先执行，因此 Q_4 任务等待了 2 个时间戳。在时间点 11，Q_2 任务抢占 Q_4 任务，抢占花费了时间 1，执行花费了时间 2，Q_4 因此等待了时间 3。在时间点 21，Q_3 任务到来了，但是由于处理器正在处理 Q_4 任务，Q_3 任务等待 Q_4 任务执行完毕，因此等待了时间 1。在时间点 26，Q_1 任务先于 Q_4 任务执行，没有发生抢占，因此 Q_4 任务被延时时间 2。此例总任务等待时间为 14。

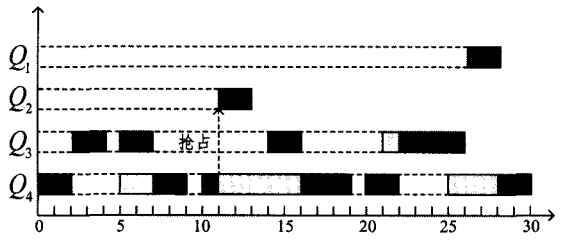

图 3　调度示例

伪代码表示：

```
While scan Q1～Q4 do
If (Q1≠null) then
  Switch (current task)
    Case Q1 task
      Wait;
    Case not Q1 task
      Stop processing current task and transmit top task of Q1;
    Case null
      Transmit top task of Q1;
  End switch;Goto Label;
Else if (Q2≠null) then
  Switch (current task)
    Case Q1 or Q2 task
      Wait;
    Case Q3 or Q4 task
      Stop processing current task and transmit top task of Q2;
    Case null
      Transmit top task of Q2;
  End switch;Goto Label;
Else if (Q3≠null) then
  Switch (current task)
    Case not null
      Wait;
    Case null
      Transmit top task of Q3;
  End switch;Goto Label;
Else if (Q4≠null) then
  Switch (current task)
    Case Q1 or Q2 or Q4 task
      Wait;
      twait=0;
    Case Q3 task
      If (twait<δ×ttimeslot) then
        twait=twait+ttimeslot
      Else Transmit top task of Q4; twait=0;
      End if
    Case null
      Transmit top task of Q4;
  End switch
End if
End while
```

4　性能分析

下面分析系统节点中各队列任务的平均等待时间，通过前两节描述的队列模型和调度策略，本文的任务模型可以看作是一个基于 M/D/1 的抢占非抢占混合优先制排队模型。根据 Agner Krarup Erlang[17] 的理论结果，M/D/1 排队模型的任务平均等待时间为：

$$t_{wait}=\frac{1}{2\mu}\times\frac{2-\rho}{1-\rho} \tag{1}$$

其中，$\mu=\frac{1}{D}$，D 为服务时间，$\rho=\frac{\lambda}{\mu}$，λ 为到达率。

对于 Q_1 中的任务，它拥有最高的优先级，并且可以抢占所有其他队列的任务，所以它的执行不受其他队列的影响，如果发生了抢占，任务的执行时间会变长，所以任务的平均等待时间可以看作发生抢占的任务和没有发生抢占的任务的平均等待时间的期望，假设发生抢占的比例为 α_1，利用式(1)求得

的平均等待时间为：

$$t_{aut,1}=\frac{\alpha_1}{2\mu_1}\times\frac{2-\rho_1}{1-\rho_1}+\frac{1-\alpha_1}{2\mu_2}\times\frac{2-\rho_2}{1-\rho_2} \tag{2}$$

其中，$\mu_1=\frac{1}{t_{pro,1}+t_{switch}}$，$\rho_1=\frac{\lambda_1}{\mu_1}$，$\mu_2=\frac{1}{t_{pro,1}}$，$\rho_2=\frac{\lambda_1}{\mu_2}$。整理得：

$$t_{aut,1}=\frac{\alpha\times\lambda_1\times(t_{pro,1}+t_{switch})^2}{2\times(1-\lambda_1\times(t_{pro,1}+t_{switch}))}+\frac{(1-\alpha)\lambda_1\times t_{pro,1}^2}{2\times(1-\lambda_1\times t_{pro,1})} \tag{3}$$

对于 Q_2 中的任务，它的执行只受到 Q_1 任务的影响，设其平均等待时间为 $t_{aut,2}$，易知：

$$(\lambda_1+\lambda_2)\times t_{aut,1\sim2}=\lambda_1\times t_{aut,1}+\lambda_2\times t_{aut,2} \tag{4}$$

其中，$t_{aut,1\sim2}$ 为整体的平均等待时间，可按 M/D/1 模型求出。即：

$$t_{aut,1\sim2}=\frac{1}{2\mu_{1,2}}\times\frac{2-\lambda_{1,2}/\mu_{1,2}}{1-\lambda_{1,2}/\mu_{1,2}} \tag{5}$$

其中，$\lambda_{1,2}=\lambda_1+\lambda_2$，$\mu_{1,2}=\frac{1}{\sum_{i=1}^{2}\frac{\lambda_i}{\lambda_{1,2}}\times(\alpha_i\times(t_{pro,i}+t_{switch})+(1-\alpha_i)\times t_{pro,i})}$。

则 Q_2 任务的平均等待时间为：

$$t_{aut,2}=(1+\frac{\lambda_1}{\lambda_2})\times t_{aut,1\sim2}-\frac{\lambda_1}{\lambda_2}\times t_{aut,1} \tag{6}$$

对于 Q_3 中的任务，它的执行一方面会遭到 Q_1 和 Q_2 任务的抢占，另一方面如果有正在执行的 Q_4 任务，由于不发生抢占，因此它会因等待而产生时延。其平均等待时间由两部分组成：正在等待的 Q_1、Q_2 和 Q_3 任务服务总时间 T_1 和等待处理器空闲的平均时间 T_2。设 L_i 为正在排队的平均队长，$1\leqslant i\leqslant 3$。则

$$T_1=\sum_{i=1}^{2}L_i\times(\alpha_i\times(t_{pro,i}+t_{switch})+(1-\alpha_i)\times t_{pro,i})+L_3\times t_{pro,i} \tag{7}$$

根据文献[18]的结果：

$$T_2=\rho\times\overline{S}_e \tag{8}$$

其中，$\overline{S}_e$ 为处理器剩余处理时间的均值，根据系统的服务时间分布来计算：

$$\overline{S}_e=\frac{\sigma^2\times\lambda^2+\rho^2}{2\rho^2} \tag{9}$$

其中，σ 为平均服务时间的方差，$\lambda=\sum_{i=1}^{3}\lambda_i+\frac{1}{p}$，$\rho=\frac{\lambda}{\mu}$，$\mu=\frac{1}{\sum_{i=1}^{2}\frac{\lambda_i}{\lambda}\times(\alpha_i\times(t_{pro,i}+t_{switch})+(1-\alpha_i)\times t_{pro,i})+\frac{\lambda_3+1/p}{\lambda}\times t_{pro,i}}$

则：

$$t_{aut,3}=T_1+T_2 \tag{10}$$

结合式(7)－式(10)，整理得：

$$t_{aut,3}=\frac{\sum_{i=1}^{2}L_i\times(\alpha_i\times(t_{pro,i}+t_{switch})+(1-\alpha_i)\times t_{pro,i})+\rho\times\overline{S}_e}{1-\rho_3} \tag{11}$$

对于 Q_4 中的任务，其优先级最低。为了防止因其他任务连续性抢占导致饥饿，我们设定了时间阈值 δ，一旦被 Q_3 的任务连续抢占超过 δ，则处理器开始打断正在执行的 Q_3 任务转而执行 Q_4 的任务。这种策略下的任务平均等待时间难以算出准确的理论值，我们通过下面一节的仿真实验给出它的实验结果。

令 $t_{aut,i}(l)$ 为第 l 层节点任务平均等待时间，假设相邻节点之间的网络传输时间相同，均为 $t_{s\to s}$，第 1 层节点任务传输到计算中心的时间为 $t_{s\to c}$，于是第 l 层节点的第 i 类任务从产生到传输至计算中心的总时延为：

$$delay_l=\sum_{k=1}^{l}t_{aut,i}(k)+(k-1)\times t_{s\to s}+t_{s\to c} \tag{12}$$

5 模拟仿真

为了比较动态多优先级调度策略与传统的抢占式优先级调度和非抢占式优先级调度策略的性能优劣，我们将 3 种算法分别用 java 编程实现。

5.1 任务平均等待时间在 3 种调度策略下的对比

我们在两种参数假设下分别做了仿真实验。

1）$t_{switch}<t_{pro,i}$，参数如表 1 所列。

表 1

参数	值
λ_1	0.04
λ_2	0.05
λ_3	0.20
p	8
t_{switch}	1.0
$t_{pro,i}$	2.0
$t_{s\to s}$	0.02
$t_{s\to c}$	0.02
k	4
仿真时间	10000

这种情况下的仿真结果（如图 4 所示）表明：当 $t_{switch}<t_{pro,i}$ 时，在动态多优先级调度下，Q_1 和 Q_2 的任务平均等待时间和在传统抢占式优先级调度下是一样的，小于传统非抢占式优先级调度。而 Q_3 的任务平均等待时间稍微优于传统抢占式优先级调度（原因在于任务抢占的次数变少了），而稍微大于传统的非抢占式优先级调度。Q_4 的任务平均等待时间明显优于传统的抢占式优先级调度（原因在于我们加入了等待时间阈值机制），而稍微大于传统的非抢占式优先级调度。综合考虑系统的整体性能，可以看出动态多优先级调度下的性能要优于传统的抢占式和非抢占式优先级调度。

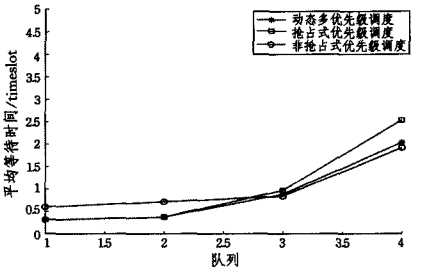

图 4 $t_{switch}<t_{pro,i}$ 时 3 种调度下的任务平均等待时间对比

2）$t_{switch}>t_{pro,i}$，参数如表 2 所列。

表 2

参数	值
λ_1	0.04
λ_2	0.05
λ_3	0.20
p	8
t_{switch}	3.0
$t_{pro,i}$	2.0
$t_{s\to s}$	0.02
$t_{s\to c}$	0.02
k	4
仿真时间	10000

这种情况下的仿真结果(如图5所示)表明:当 $t_{switch} > t_{pro,i}$ 时,动态多优先级调度的整体性能优于抢占式优先级调度,但是不如非抢占式优先级调度。原因在于抢占花费的时间太多。因此我们应当根据不同的应用需求采取不同的调度策略。

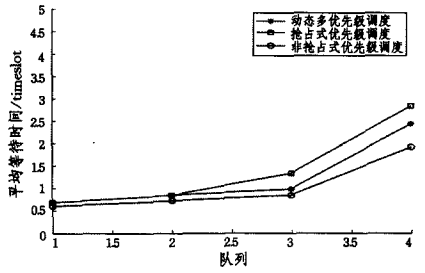

图5 $t_{switch} > t_{pro,i}$ 时3种调度下的任务平均等待时间对比

5.2 任务总时延在3种调度策略下的对比

令 $\theta = \frac{t_{switch}}{t_{pro,i}}$,我们通过多次仿真实验给出了任务总时延随 θ 的变化情况(如图6所示),结果表明:当 $\theta < 1$ 时,采用动态多优先级调度的总时延最小;当 $\theta > 1$ 时,采用非抢占式调度的总时延较小。

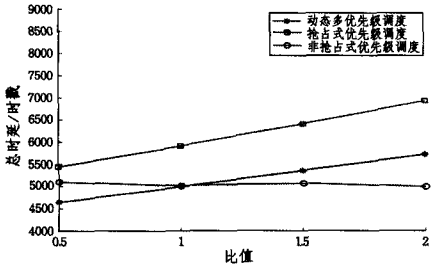

图6 θ 变化时3种调度下总时延对比

5.3 队列的最大长度在3种调度策略下的对比

队列的最大长度能作为设计缓存区大小的参照,由于节点存储能力有限,我们希望能尽量减小缓存区的大小。通过仿真实验,我们比较了3种调度策略下的各队列最大长度。此处采用了表1的参数。结果(如图7所示)表明:动态多优先级调度在 Q_1、Q_2 和 Q_4 上都能达到最小的最大队长。因此它的综合性能要优于传统的优先级调度。

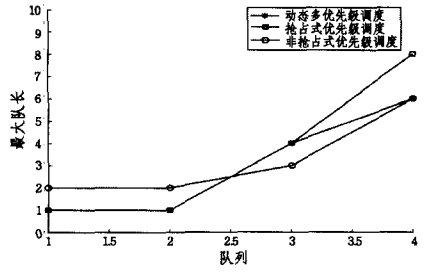

图7 3种调度下队列最大长度对比

结束语 本文针对多任务类型下的基于传感器网络的CPS提出了抢占与非抢占混合的动态多优先级调度策略,每个节点设计4层优先级缓存队列,任务的到来根据类型进入不同的队列,其中实时任务比非实时任务的优先级要高,并且可以无条件抢占非实时任务,转发的任务比本地的任务优先级要高,但是它们之间不发生抢占。为了保证低优先级队列的公平性,加入了等待时间阈值机制。最后通过仿真实验将我们的动态多优先级调度策略与传统的抢占式和非抢占式优先级调度做了比较,结果显示,在抢占耗费的时间小于任务处理时间的情况下,动态调度策略的整体性能要优于传统的优先级调度。

参 考 文 献

[1] NSF Workshop on Cyber-Physical Systems[OL]. http://varma.ece.cmu.edu/cps/,Oct. 2006

[2] COMPUTING,EMBEDDED. Cyber-physical systems[OL]. http://en.wikipedia.org/wiki/cyber-physical-system

[3] Baheti, Radhakisan, Gill H. Cyber-physical systems[C] // The Impact of Control Technology. 2011:161-166

[4] Rajkumar,Raj R,et al. Cyber-physical systems:the next computing revolution[C]//Proceedings of the 47th Design Automation Conference. ACM,2010

[5] Lui Sha, et al. Cyber-physical systems: A new frontier. " Machine Learning in Cyber Trust[M]. Springer US,2009:3-13

[6] Lee E A,Seshia S A. Introduction to Embedded Systems,A Cyber-Physical Systems Approach[OL]. http://LeeSeshia.org

[7] Wang Xiao-feng. Event-triggering in cyber-physical systems[M]. Diss. University of Notre Dame,2009

[8] Tan Ying, Goddard S, Perez L C. A Prototype Architecture for Cyber-Physical Systems[J]. ACM SIGBED Review, Special issues on RTSS forum on deeply embedded real-time computing, 2008,5(1)

[9] Lee E A. Cyber-physical systems-are computing foundations adequate[C]//Position Paper for NSF Workshop on Cyber-Physical Systems: Research Motivation, Techniques and Roadmap. 2006,2

[10] Lee E A. Cyber Physical Systems: Design Challenges [C]// International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC). May 2008

[11] Richard W,Parmer G. A software architecture for next-generation cyber-physical systems[C]//Position Paper at the NSF Cyber-Physical Systems Workshop. 2006

[12] Zhang F, Szwaykowska K, Wolf W, et al. Task Scheduling for control oriented requirements for cyber-physical systems[C]// Proc. IEEE Real-Time Systems Symposium. 2008:47-56

[13] Zhang Fu-min,Shi Zhen-wu,Wolf W. A dynamic battery model for co-design in cyber-physical systems[C]//29th IEEE International Conference on Distributed Computing Systems Workshops,2009. ICDCS Workshops'09. IEEE,2009:51-56

[14] Goswami D, Schneider R, Chakraborty S. Co-design of cyber-physical systems via controllers with flexible delay constraints [C]// Proceedings of the 16th Asia and South Pacific Design Automation Conference (ASPDAC). 2011:225-230

[15] Tang Q,Gupta S K S,Varsamopoulos G. A Unified Methodology for Scheduling in Distributed Cyber-Physical Systems[J]. ACM Transactions on Embedded Computing Systems (TECS), 2012,11(S2):57

[16] Li Qiao. Scheduling in Cyber-Physical Systems[OL]. http://www.cyphylab.ee.ucla.edu/Home/project/control-and-real-time-scheduling-co-design

[17] Olivier B,Garcia J-M. Analytical solution of finite capacity M/D/1 queues[J]. Journal of Applied Probability,2000,37(4):1092-1098

[18] 陆传赉. 排队论(第二版)[M]. 北京:邮电大学出版社