文章编号:1672-058X(2014)11-0045-05

# 基于 NS2 的优先级队列管理算法设计*

潘丰旻, 江 明**, 周加文, 葛 愿
(安徽工程大学,安徽 芜湖 241000)

**摘 要**:网络控制系统的信号以分组的形式在网络中传输,引起了传输时延、丢包等问题;为了更好地研究网络控制系统的性能,利用 NS2 搭建了网络控制系统的网络传输模型。并以其中的优先级业务分组为研究对象,设计了优先级队列管理算法;在 DropTail 算法的基础上进行改进,继承了原有算法的优势,并加入了业务优先级识别和弃包选择机制;通过调用两种算法进行实验,从时延、丢包以及吞吐量方面进行分析与对比,验证了 PDropTail 算法的有效性。

**关键词**:网络控制系统;优先级;队列管理算法;时延;丢包;吞吐量;
**中图分类号**:TP307 **文献标志码**:A

## 0 引 言

网络控制系统(Networked Contorl Systems,NCS)于 1999 年出现在马里兰大学 Walsh 的论著中,该系统利用通信媒介使控制回路形成闭环[1]。与传统的控制系统不同,它采用多样的网络连接方式提高了系统组件的灵活性,在智能电网、智能交通等领域有着巨大的应用前景[2-3]。融合了计算机技术和通信技术的 NCS,除了控制算法影响其性能好坏以外,合理地调度计算机的网络资源也十分必要.NCS 的网络服务质量指标包括时延、丢包和吞吐量等。由于网络的不稳定性必然引入时延、丢包等问题[4-6]。因此,要在 NCS 的网络传输部分采用合适有效的队列管理算法。

进行实验模拟是研究分析 NCS 性能的高效方法。NCS 模拟工具主要包括 MATLAB、OPNET 以及 NS2. MATLAB 的 TrueTime 工具箱在控制策略方面的模拟功能强大,但其网络模块的功能相对薄弱,模拟 NCS 网络环境中的队列管理机制较为困难;OPNET 具有很强的网络功能,但这款商用软件使用费较为高昂;而 NS2 作为一款开源免费的软件,它采用分裂对象模型的方式,利用 Otcl 语言搭建网络拓扑结构,C++语言实现具体协议,达到了模拟配置灵活性和仿真运行高效性的统一[7-9],受到越来越多的重视。如高文宇等人提出的网络仿真软件 NS2 中队列调度算法的扩展[10]。杨故等提出的基于 NS2 的改进队列管理算法及其实现[11]。高鹏等人基于 NS2 的主动队列管理算法的仿真与分析[12]等。为此,本文首先设计 NCS 中的优先级队列管理算法;其次利用 NS2 搭建一个网络传输模型,并调用队列管理算法进行仿真实验;最后进行实验数据的分析和实验结果的总结。

**收稿日期**:2014-05-22;**修回日期**:2014-06-20.
* **基金项目**:国家自然科学基金项目(61271377,61203034);安徽省自然科学基金项目(1308085QF120);安徽省级优秀青年人才基金重点项目(2012SQRL086ZD);安徽工程大学国家级大学生创新创业训练计划项目(201310363049).
**作者简介**:潘丰旻(1990-),男,硕士,从事网络化控制系统研究.
** **通讯作者**:江明(1965-),男,教授,硕导,从事智能网络控制系统研究.

# 1 基于NS2的优先级队列管理算法实现

## 1.1 队列管理算法

网络业务流以数据分组的形式在网络中传输。它们流经不同网络节点时会采取队列缓存、延迟转发等服务方式,而队列管理的作用就在于此。能通过丢弃或者标记这些数据分组来管理网络传输节点中的队列缓冲资源。队列管理主要分为主动式队列管理及被动式队列管理。主动式队列管理算法通过预测分组的状态特性,在网络拥塞还未发生之前就按照预先的设定进行相应的弃包,从而有效控制队列缓冲区的长度,其中的典型算法为RED;而被动式队列管理算法在队列缓冲区设置了固定的数据分组存储上限,当队列中接收到的分组达到上限值,则启动弃包机制,其中典型算法有DropTail算法。验证队列管理算法性能的好坏,要从业务需求和算法复杂程度出发,结合时延、丢包、吞吐量等因素进行综合讨论,而两种典型算法的算法性能如表1所示。

主动队列管理算法结构较为复杂,增加设备开销,平均时延相对较高,网络吞吐量也相对较低。其参数设置敏感并且网络响应相对滞后,并没有在网络当中得到大量使用;相比之下,虽然被动式队列管理算法丢包率相对较高,但其相对较低的延时特性以及简单的算法结构,在实际网络中仍获得广泛使用。

## 1.2 PDropTail算法设计

NCS因为系统业务类型复杂,不同业务之间对于网络服务质量往往有着不同需求,需要在保持较低时延和较高吞吐量的基础上进行优先级区分服务。所以新的算法具有优先级识别机制,它在选择弃包方面不再是直接“弃尾”,而是综合考虑了队列中数据分组的优先级情况:当数据分组进入节点的队列缓冲区时,若缓冲区为未满状态,则仍然按照先进先出的原则实行队列管理;若缓冲区为已满状态,则启动优先级识别机制,选取队列缓冲区队头前两个数据分组进行优先级比较。若第一个数据分组的优先级较高,则选择丢弃队列尾的数据分组,否则丢弃队列头。PDropTail队列管理算法原理如图1所示。

表1 队列管理算法性能对比

| QoS指标 | CBR流 | |
|---|---|---|
| | DropTail | RED |
| 平均时延/ms | 51.757 | 209.792 |
| 丢包/550个 | 18 | 6 |
| 平均吞吐量/Kb | 976.980 773 | 940.139 963 |

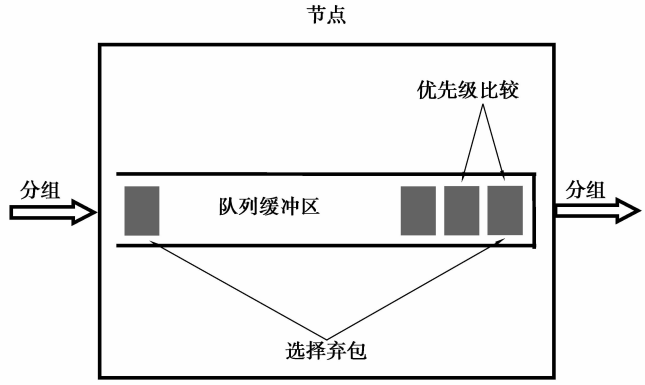


图1 PDropTail队列管理算法原理图

## 1.3 PDropTail算法在NS2中的实现

利用NS2仿真软件进行PDropTail算法的设计,首先需要在NS2中进行算法脚本的编译与添加。而PDropTail算法的优先级识别机制和弃包选择机制的部分核心程序代码如下:

```
void PDropTail::enque(Packet* p)
{
    ……
    int qlimBytes = qlim_ * mean_pktsize_;
    if ((! qib_ && (q_->length() + 1) >= qlim_) ||
    (qib_&&(q_->byteLength()+hdr_cmn::access(p)
    ->size()) >= qlimBytes))
    {
        int prio1,prio2;
        prio1=HDR_IP(q_->lookup(0))->prio();
        prio2=HDR_IP(q_->lookup(1))->prio();
        if(prio2>prio1) {
```

```
        q_->enque(p);
        Packet * pp=q_->deque();
        drop(pp);
    }
    else {
        drop(p);
    }
}
```

qib_&&(q_->byteLength()+hdr_cmn::access(p)函数部分是对缓冲区的状态进行判断,若队列为满状态,则启动优先级比较机制对队列头前两个数据分组的优先级 prio1 和 prio2 进行比较。该比较机制中若第二个数据分组的优先级更大则强制丢弃队头数据分组 drop(pp);若第一个数据分组优先级大则强制丢弃队尾分组 drop(p)。而当队列状态为未满时,数据分组将直接进入队列缓冲区 enque(p)。

## 2　基于 NS2 的网络模拟与算法性能分析

### 2.1　搭建网络拓扑结构

利用 NS2 仿真软件搭建如图 2 所示的 NCS 网络传输模型。搭建该模型的步骤包括节点设置以及链路配置两方面。从节点设置方面考虑,标号为 8、9 的两节点设为路由节点,0、1、2、3 节点设为发送节点,4、5、6、7 设置为接收节点;从链路配置方面考虑,两路由节点间的瓶颈链路带宽设为 1.7 M,时延设为 20 ms,其余链路带宽均设为 2 M,时延为设 10 ms。这样的拓扑结构设计和带宽配置主要为了让数据分组在传输过程中产生一定的网络拥塞情况。

在 0 到 4,1 到 5,2 到 6 以及 3 到 7 链路之间各建立一条 UDP 联机,并在 UDP 联机之上建立 CBR 流量发生器.为了模拟 4 个不同优先级业务,需要对数据分组的 IP 包头设置优先级。将 4 种业务的优先级分别设置为 1、2、3、4,其中数字越大为优先级越高,在网络中将获得优先服务。网络模型搭建完成后,分别调用两种算法进行模拟实验,模型正常工作后会出现如下的动画效果如图 2 所示。网络模型搭建的主要代码如下:

```
set n8 [MYMns node]
set n9 [MYMns node]     //路由节点设置
MYMns duplex-link MYMn8 MYMn9 1.7M 20ms 算法
MYMns queue-limit MYMn8 MYMn9 10
for {set i 0} {MYMi<3} {incr i} {
set s(MYMi) [MYMns node]
set d(MYMi) [MYMns node]
MYMns duplex-link MYMs (MYMi) MYMn8 2M 10ms 算法
MYMns duolex-link MYMn9 MYMd (MYMi) 2M 10ms 算法
//发送节点、接收节点设置以及链路参数配置
set udp (MYMi) [new Agent/UDP]
set null (MYMi) [new Agent/Null]
MYMns attach-agent MYMs (MYMi) MYMudp (MYMi)
MYMns attach-agent MYMd (MYMi) MYMnull (MYMi)
MYMns connect MYMudp (MYMi) MYMnull (MYMi)
MYMudp (MYMi) set fid_ MYMi
MYMudp (MYMi) set prio_ MYMi   //建立 udp 联机
set cbr (MYMi) [new Application/Traffic/CBR]
MYMcbr (MYMi) attach-agent MYMudp (MYMi)
}     //cbr 流量发生器设置
```

### 2.2　队列管理算法性能分析与比较

调用两种算法下的业务时延情况如图 3 所示,其中业务 cbr1 为获得最高优先级的业务,cbr4 为优先级最低的业务。调用两种算法下,当模拟时间在 0.3 s 之前,网络业务量都不断上升,节点缓冲区中的数据分组不断增多导致网络时延不断增大。但此时由于缓冲区为未满状态,队列管理机制仍采取“先进先出”的原则,此时,两种算法下的网络时延情况大体一致。当数据分组在队列缓冲区开始拥塞的情况下,DropTail 算法采取“弃尾”机制不停丢弃队尾分组。而 PDropTail 算法采取优先级识别和弃包选择机制后,高优先级业务获得优先服务,相比 DropTail 算法就得到了更低的时延。因此两种算法下的时延情况有较大不同。而表 2 为不同业务流的平均网络时延情况。

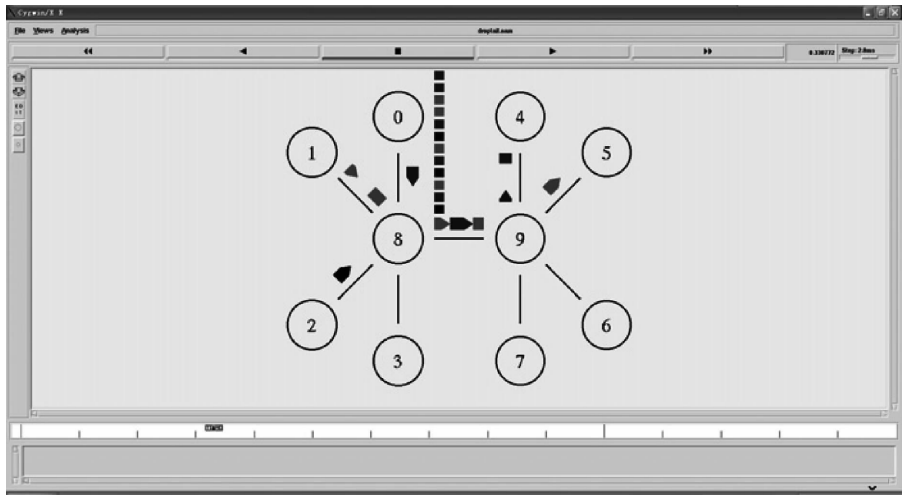


图 2 网络模拟动画效果

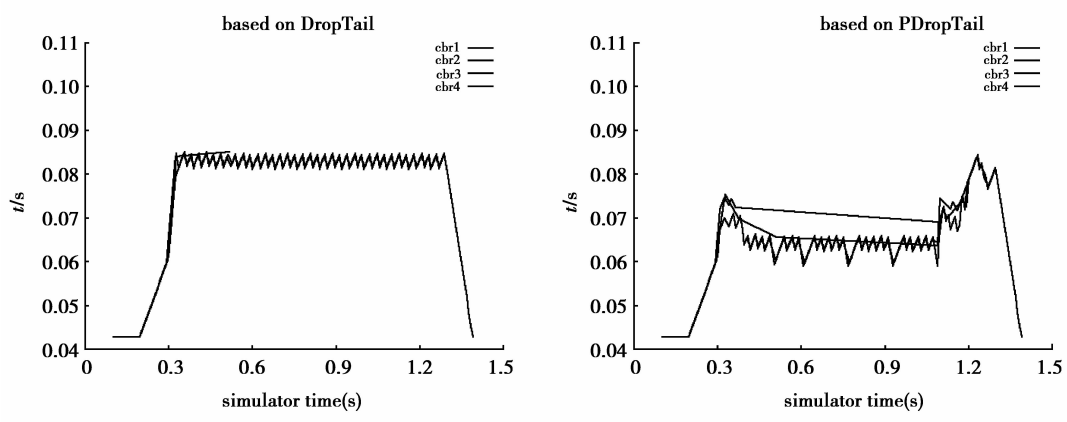


图 3 两种算法下的时延情况

表 2 两种算法下的业务流平均时延

| 算法 | 业务分组时延/s | | | |
|---|---|---|---|---|
| | cbr1 | cbr2 | cbr3 | cbr4 |
| DropTail | 0.052 086 | 0.064 097 | 0.082 484 | 0.080 544 |
| PDropTail | 0.053 087 | 0.060 726 | 0.066 66 | 0.065 628 |

明显可以看出在调用 PDropTail 算法时,不同业务的时延基本低于 DropTail 算法下的时延,事实证明新算法能保证较低的网络时延。而两种算法下的实时网络丢包率情况如图 4 所示。两种算法的实际网络丢包情况如表 3。

表 3 两种算法下的业务流丢包情况

| 算法 | 业务分组丢包(/125 个) | | | |
|---|---|---|---|---|
| | cbr1 | cbr2 | cbr3 | cbr4 |
| DropTail | 95 | 102 | 23 | 13 |
| PDropTail | 92 | 101 | 32 | 8 |

由图形和表格数据可知:两种算法下的实时丢包率均随着模拟时间逐步上升。虽然两种算法情况下的总丢包数保持一样的水平,但是调用 PDropTail 算法下确保了优先级较高的业务 cbr4 获得优先服务,保证了其较低的网络丢包情况。

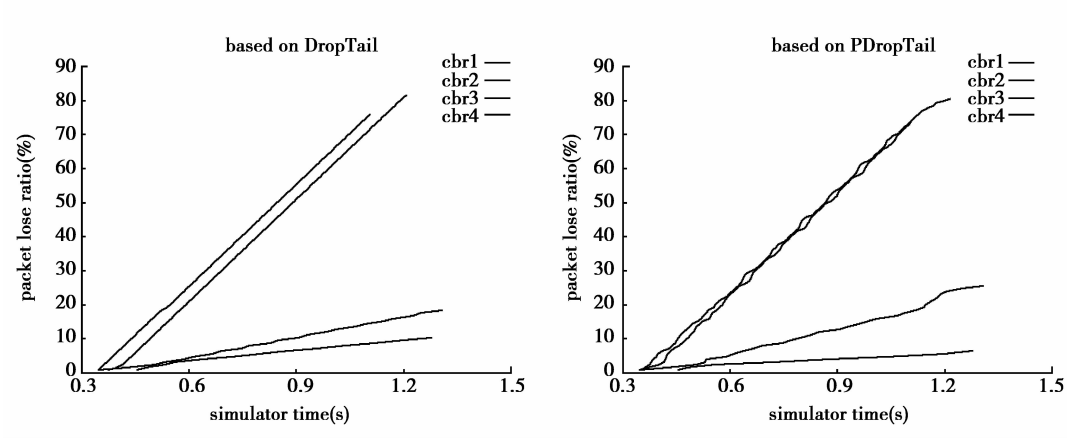


图 4　两种算法下的丢包率情况

调用两种算法的网络吞吐量情况如图 5 所示。图 5 反映了新算法能保持与 DropTail 算法几乎一致的高吞吐量。因此,通过时延、丢包和吞吐量等方面因素的实验对比分析,验证了新算法对于优先级业务具有一定区分服务的能力,能保证较高优先级业务较低的网络时延和丢包率,也能确保业务在网络传输过程中较高的吞吐量。

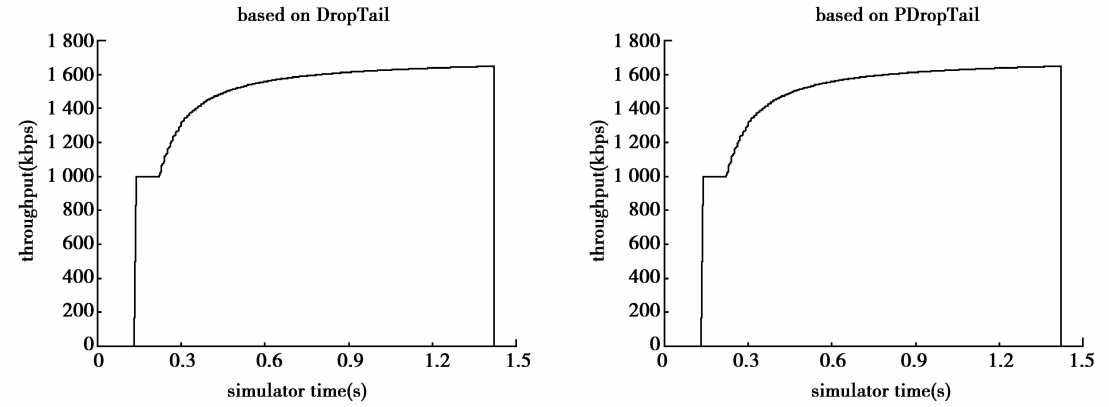


图 5　两种算法下的吞吐量情况

## 3　结束语

本文利用 NS2 对 NCS 的网络传输模型进行模拟仿真,设计添加了优先级队列管理算法,并在实验中对 DropTail 算法、PDropTail 算法的性能进行了分析比较。从实验数据中可以得出,PDropTail 算法能为 NCS 中具有不同优先级的业务提供更好的区分服务,保证较高优先级业务具有较低的网络传输延时、较低的网络低丢包率,同时也继承了 DropTail 算法较高吞吐量的优点。因此,实验模拟的结果达到了预期效果。

**参考文献:**

[1] WALSH G, HONG Y, BUSHNELL L. Stability analysis of networked control systems[J]. American Control Conference, 1999. Proceedings of the 1999 ,14(4):2876-2880

[2] IPACCHI A, ALBUYEH F. Grid of the future[J]. Power and Energy Magazine, IEEE ,2009,7(2):52-62

[3] WANG Feiyue. Parallel Control and Management for Intelligent Transportation Systems: Concepts, Architectures, and Applications[J]. Intelligent Transportation Systems, IEEE Transactions on,2010,11(3):630-638

[4] YUE Dong, HAN Qinglong, CHEN Peng. State feedback controller design of networked control systems[J]. Circuits and Systems II: Express Briefs, IEEE Transactions on,2004,51(11):640-644

(下转第 61 页)