闫明月, 裘莹. 基于 TSCH 的工业无线网络流量自适应传输调度机制研究[J]. 智能计算机与应用,2024,14(5):164-171.
DOI:10.20169/j.issn.2095-2163.240522

基于 TSCH 的工业无线网络流量自适应传输调度机制研究

闫明月, 裘 莹
(浙江理工大学 信息学院, 杭州 310018)

摘 要: 当今 LLN 网络设备要满足低功耗要求的同时必须支持多种业务数据的传输,所以需要一种动态调度算法提高网络的可靠性。虽然基于 IEEE 802.15.4 标准的时隙信道跳变(TSCH)技术可以满足 LLN 的需求,然而现有的 TSCH 调度算法缺少灵活性。为了解决这一问题,本文提出了基于 TSCH 的流量感知自适应调度改进算法(Enhanced Adaptive Transmission Scheduling Mechanism,EATSM)。本文通过自相似性采样方法采集各链路流量信息,在准确获取各个邻居链路流量之后进行按权按序为每个邻居链路分配时隙数量与位置,保证传输可靠性的同时又能做到调度灵活性。通过理论分析与实验数据对比,本文提出的 EATSM 调度算法设计相比较于主流自适应调度算法,数据包投递率为 99.57%,提高了 0.26 个百分点,节点运行占空比为 0.34%,减少了 0.88 个百分点。

关键词: LLN; IEEE 802.15.4; TSCH; 动态时隙调度; 无线网络协议
中图分类号: TN919.3 **文献标志码**: A **文章编号**: 2095-2163(2024)05-0164-08

Research on adaptive transmission scheduling mechanism of industrial wireless network traffic based on TSCH

YAN Mingyue, QIU Ying
(School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: Today's LLN network devices must support the transmission of multiple service data while meeting the requirements of low power consumption, so a dynamic scheduling algorithm is required to improve the reliability of the network. Although the Time-Slot Channel Hopping (TSCH) technology based on IEEE 802.15.4 standard can meet the needs of LLN, the existing TSCH scheduling algorithm lacks of flexibility. To solve this problem, this paper proposes the Enhanced Adaptive Transmission Scheduling Mechanism (EATSM) algorithm based on TSCH. In this paper, the self-similarity sampling method is used to collect the traffic information of each link, and after accurately obtaining the traffic of each neighbor link, the number and position of time slots are allocated to each neighbor link in order according to the weight, so as to ensure the reliability of transmission and achieve scheduling flexibility. Through theoretical analysis and experimental data comparison, compared with the mainstream adaptive scheduling algorithm, the proposed EATSM scheduling algorithm shows that the packet delivery rate is 99.57%, an increase of 0.26 percentage points, and the node operation duty ratio is 0.34%, a decrease of 0.43 percentage points.

Key words: Low-power and Lossy Network; IEEE 802.15.4; TSCH; dynamic slot scheduling; wireless network protocol

0 引 言

随着“工业 4.0”的推进,无线传感网在构建智能网络方面发挥着不可替代的作用。人工智能、大数据、云计算等技术的发展,推动物联网、移动互联网、无线通信技术等领域不断进步,使信息技术与工业融合越来越紧密[1]。无线传感器网络不受布线限制,可在复杂的工业环境中应用,对提升生产效率、降低成本、保障产品质量有着非常重要的作用,同时也能通过检测环境卫生,从而改善并减少能源消耗等方面产生积极影响[2]。

IEEE 802.15.4[3] 是一种低速、低功耗、低数据率的无线局域网(WLAN)标准。其中,时隙信道跳频(TSCH)技术[4] 是 IEEE 802.15.4 标准中最重要

基金项目: 国家自然科学基金青年科学基金项目(62003307)。
作者简介: 闫明月(1997-),女,硕士研究生,主要研究方向:嵌入式与物联网技术。
收稿日期: 2023-04-15

的调度方式之一，采用了时间分隔和频道跳转技术，使得网络在低功耗的情况下实现高可靠性、低延迟和高可扩展性。TSCH 协议具备高容量和可靠性、一定的灵活性和低能耗性。

低功耗局域网（LLN）是一种基于无线传感器网络技术实现的传输网络，主要应用于智能物联网、远程监控、智能城市等领域。在 LLN 网络中，流量控制是一个至关重要的问题[5-8]，因此需要有效的流量自适应技术来确保网络数据传输的稳定性和质量。

目前，研究人员已经开发出了许多基于 TSCH 的流量自适应技术，并在不同的应用场景中进行了测试。其中，一些技术包括基于改进的控制理论的自适应网络控制技术运用于林火检测[9]、基于优化算法的网络资源分配技术用来优化 5G 异构网络中的机器型通信[10]以及基于云交互信息创建数据安全管理库以确保所有医疗记录的安全和私密安全[11]等等。

本文的主要研究工作有：阐述工业无线网络传输调度策略的相关知识点，引入当下主流网络调度策略。就流量感知自适应调度方法对 OST 进行重点分析，基于 OST 的模型思想提出自己的流量感知自适应调度策略：EATSM。EATSM 的设计整体框架是由 3 个重要的模块组成。这里给出阐释分述如下。

（1）拓扑处理器：主要工作是通过路由感知检测来自各个设备的链路流量 N；

（2）时隙帧选择器：将获取的不同邻居设备的链路流量 N_i 最小公倍数算法处理，选择等长的时隙帧作为公共时隙帧。

（3）时隙插槽分配器：根据流量 N_i 算出此定向链路的数据包量 $Pkts$，在公共时隙帧内对应分配等量的时隙插槽数，再根据访问邻居先后顺序排列来自节点设备 i 的定向链路上的数据包等量插槽。

最后，利用 Arch Linux 安装 Contiki-OS 小型操作系统，并利用 Cooja 仿真平台构建无线传感网络模型，各个节点使用 EATSM 调度策略，根据仿真数据分析投递率和网络稳定性。并与主流调度算法 OST 做对比，得出结论：本文的策略在提高投递率和减少网络能耗方面更出色。

1　流量感知自适应算法设计

算法设计整体框架如图 1 所示。图 1 中，EATSM 调度算法介于 RPL 和 TSCH 之间，能够与路由网络层和应用层以及链路层进行直接交互。EATSM 算法的输入参数为本节点的 RPL[12]路由信息与应用层用来模拟流量突变的变化周期流；输出数据为 TSCH 运行时的传输调度信息。运行的每个节点设备使用本地调度生成各自的传输调度表，且不需要做全局信息比较，因此不产生额外的信令开销。

调度算法的 3 个模块分别为：节点拓扑信息、时隙帧选择器和时隙插槽分配器。

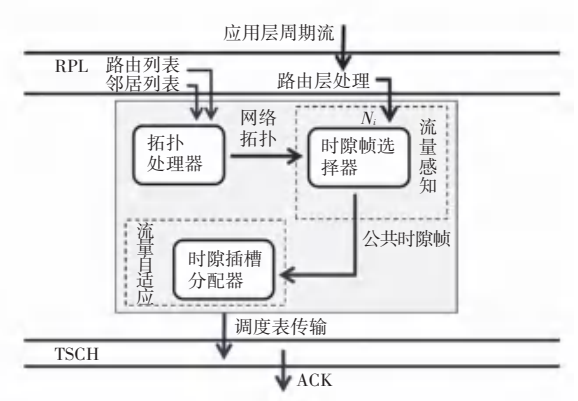

图 1　算法设计整体框架

Fig. 1　Overall framework of algorithm design

1.1　拓扑处理器

目前的基本采样方法有周期采样和随机采样，但是对于无线通信传输特点而言，缺点也比较突出：采样的周期性让网络状态具有很强的可预测性，这会使被测网络陷入一种同步状态；而随机采样的抽样间隔不固定造成频域分析复杂化。

目前主流的关于流量采样在无线网络中的研究[13-15]证明，自相似性采样方法能够最大程度地还原网络状态。甚至在文献[16]中应用自相似性设计了网络流量生成器，进行网络建模并用于测试网络攻击行为、拥塞控制和异常检测系统。

自相似抽样法是通过相邻抽样间隔时间内的 Hurst 参数差值来确定网络流量状态变化程度。基于自相似采样思想，本文通过相隔采样时间内的流量增率 ΔN 作为 Hurst 参数，根据 ΔN 的大小决定下一次采样间隔时间的增减量。此时间段采集的流量保留，既可以计算流量增率，也是下一跳节点自适应调度表的参考值，流量 N 表示时间间隔（Δt）内平均传送一个数据包（Pkt）的时隙间隔数量，流量 N 的计算公式为：

$$N = Pkts/(\Delta t/10) \tag{1}$$

各个节点设备利用自相似采样得到的流量值 N 将作为流量自适应设计部分的参考数据。

1.2 时隙帧选择器

在网络调度策略中,时隙帧作为节点的调度单位,不同的时隙帧长度对网络性能的影响是不可小觑的:选择时隙帧长度较大的网络存在数据包延迟和较差的数据传输可靠性;而盲目地减少时隙帧长度不仅不会提高传输可靠性,在时隙帧长度过短时会使节点处于高频的唤醒状态,从而导致不必要的能量损耗,过短的时隙帧提高了链路冲突。EATSM 的槽帧选择器旨在根据各节点设备流量传输状态选取长度适合的时隙帧,使网络能够以低能耗实现高投递率。

参考 EATSM 调度算法框架图,每个节点设备在使用时隙帧选择器时,将路由拓扑信息和经路由网络层处理过的应用层信息 N_i 作为时隙帧选择器的输入值,这里 N_i 表示节点设备 i 在流量感知设计部分获取到的平均流量参数表示。

考虑由 n 个终端节点设备和 1 个根节点设备组成的网络。首先,时隙帧选择器根据节点设备 i 在持续一段时间内感知统计的数据包数量 $Pkts$,令这段持续时间长度为 D,单位为一个时隙值(10 ms),在 EATSM 中时间长度 D 被定义为网络中 $n+1$ 个节点设备的所有链路上的数据包间隔时隙数的最小公倍数。所以 D 值作为这 $n+1$ 个节点设备所组成的拓扑网络中的公共参数,同时,也表示这个网络的流量特性。在 D 期间内定向链路 j 的数据包承载量为 P_j,可由式(2)来求值:

$$P_j = \sum_{i=1}^{m+1} \frac{D}{N_i} \tag{2}$$

其中,N_i 表示节点设备 i 的数据包间隔时隙数,m 表示与链路 j 相关联的终端设备的数量。每个终端设备可能是链路 j 的发送器的各子节点,也可能是接收器的各子节点。接下来,还将对本文研究工作展开探讨论述如下。

(1)链路最大负载。当 $m=n-1$ 时,在根节点设备和其他 $n-1$ 个终端节点设备之间的所有数据转发流量包都将由定向链路 j 承载,在这样的极端拓扑情况下,期间 D 内定向链路 j 的数据包最大转发量为 P_j^m,表示为:

$$P_j^m = \sum_{i=1}^{n} \frac{D}{N_i} \tag{3}$$

(2)节点设备最大负载。同样地,在定向链路 j 的负载量为 P_j^m 时,讨论终端节点设备的最大负载量。在极端拓扑情况下,终端节点设备负责在 D 期间内传输负载链路的数据包,那么此时的工作内容将会是上行链路与下行链路的全部流量负载的接收与发送。设极端拓扑情况下终端节点设备的最大流量负载为 $P_{\max}$,表示为:

$$P_{\max} \simeq 2 \times \left(\sum_{i=1}^{n} \frac{D}{N_i^u} + \sum_{i=1}^{n} \frac{D}{N_i^d} \right) \tag{4}$$

其中,N_i^u 表示从节点设备 i 到根节点设备的上行数据包间隔时隙数,N_i^d 表示从根节点设备到节点设备 i 的下行数据包间隔时隙数。

由于给定的 D 是时隙帧的时隙数,只有当 D 的值大于等于 $P_{\max}$ 时,也就是网络时隙数量要能够容纳网络最大负载的数据包数量时,才能保证即使在极端拓扑情况下也能使用自适应调度算法。这也是 EATSM 接下来的选取时隙帧的条件之一。

(3)时隙帧满足条件。时隙帧长度大于等于 $P_{\max}$ 之后,将对具体的时隙帧做子时隙帧选取与时隙分配处理。在选取时隙帧时以节点各单向链路承载数据包量作为参考依据,当节点作为接收设备时,对来自于各个相邻节点的链路流量 N_i 做最小公倍数算法处理,为 $lcm\{N_1, N_2, N_3, \cdots\cdots, N_i\}$,求出的最小公倍数作为公共时隙帧 L 的时隙个数。

子时隙帧 handle 与长度设计例图如图 2 所示。图 2 中,子时隙帧的 handle 作为时隙帧的标识符,EATSM 设计为每个链路的流量值累加后的生成数,所以子时隙帧 handle 不会重复,再就是子时隙帧的长度设计是与链路流量 N 所匹配的。根据遍历邻居链路顺序,按序安排各个子时隙帧在公共时隙帧的位置。

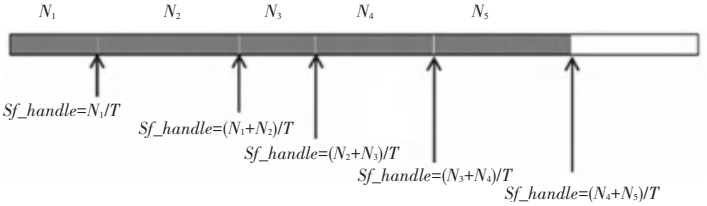

图 2 子时隙帧 handle 与长度设计例图

Fig. 2 Example diagram of handle and length design of sub-slot frame

1.3　时隙插槽分配器

子时隙帧的 handle 与长度确定下来后，紧接着就是将各节点间链路信息内添加时隙插槽位置与对应的子时隙帧 handle。分配时隙插槽时根据流量 N_i 算出此定向链路的数据包量 $Pkts$，对应分配等量的时隙插槽数，再根据访问邻居先后顺序排列来自节点设备 i 的定向链路上的数据包等量插槽。

邻居链路流量的时隙分配如图 3 所示。图 3 中，节点收到 3 个相邻节点的定向链路，其流量参数分别为 3、4、6，公共时隙帧为 12，为邻居 1 分配的时隙插槽为{1,2,3,4}，为邻居 2 分配的时隙插槽为{5,6,7}，为邻居 3 分配的时隙插槽为{8,9}。

图 3　邻居链路流量的时隙分配表

Fig. 3　Slot allocation table of neighbor link traffic

1.4　多跳信道设计

在网络拓扑结构变化频繁的情况下，路由邻居之间的路由可能会失效，对网络传输性能会产生不良影响。所以为了弥补这一缺陷，扩大网络容量，本文采用了多信道跳频技术，基于链路信息的信道偏移计算公式为：

$$offset_{\text{channel}} = h(ASFN + MAC)\%(N_{\text{Listc}} - 2) \quad (5)$$

其中，$offset_{\text{channel}}$ 使用 $N_{\text{Listc}} - 2$ 的模运算符，而不是 N_{Listc}，因为 EB 和 RPL 共享槽帧占用 2 个固定的信道偏移，保证了基本的 TSCH 和 RPL 操作。

2　流量感知自适应算法实现

2.1　验证自相似采样可行性

验证自相似采样优于周期与随机采样：首先设计了简单的模拟网络采样，通过实验数据来验证自相似采样优于周期与随机采样。算法 1 描述了周期采样与随机采样算法设计；算法 2 和算法 3 描述了自相似采样的算法设计。

算法 1　周期采样与随机采样算法设计

```
1.  while(1){
2.      int i = i + 1;
3.      int t = 1 < < i;
4.      if(周期采样){
5.          etimer_set (&et, CLOCK_SECOND * random_rand());
6.      }else if(随机采样){
7.          etimer_set (&et, CLOCK_SECOND * old_T);
8.      printf("time:%u\n", t);
9.      PROCESS_YIELD_UNTIL(etimer_expired(&et));
10.     etimer_reset(&et);
11.  }}
```

算法 2　自相似采样间隔的算法设计

```
1.  void select_T(void)
2.  {
3.   int old_T = 10; int old_N = 10;
4.   int new_N = (T * 10)/(nbr -> num_tx);
5.   If((new_N - old_N)/old_T) < 0
// 流量减小
6.      old_T = old_T + 2;
// 减少采样间隔
7.   elseIf((new_N - old_N)/old_T) > 0
// 流量增大
8.      old_T = old_T - 2;   // 增大采样间隔
9.  if(old_T < 0)
10.        return 0;    //流量异常
11.  old_N = new_N;
12.  }
```

算法 3　自相似采样的算法设计

```
1.  while(1){
2.      int i = i + 1;
3.      int t = 1 < < i;
4.        etimer_set (&et, CLOCK_SECOND * old_T);
5.      printf("time:%u\n", t);
6.      PROCESS_YIELD_UNTIL(etimer_expired(&et));
7.      select_T();
8.      etimer_reset(&et);
9.        etimer_set (&et, CLOCK_SECOND * old_T);
10.  }
```

将以上代码在 Contiki 系统上进行编译，并把代码生成的数据记录下来后用点线图展示。其中，对比实验设置如下：周期采样时间间隔为 2 s，采样数据截取到 30 s；随机采样的随机时间间隔控制在 0~4 s时间，采样数据截取到 30 s 左右；自相似采样时

间初始值为 4 s,采样间隔为流量增率的倒数、即 $1/\Delta N$, 采样数据截取到 30 s。

周期采样点线如图 4 所示,随机采样点线如图 5 所示,自相似采样点线如图 6 所示。由于代码设计部分是为了反映幂函数模型图,图 4、图 5 和图 6 对比显示,自相似采样生成的点线图更加贴近幂函数模型,接下来的流量采样将沿用此自相似算法。

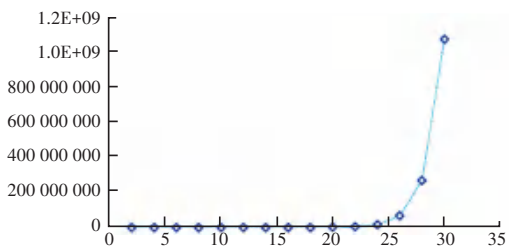

图 4 周期采样点线图

Fig. 4 Diagram of periodic sampling points

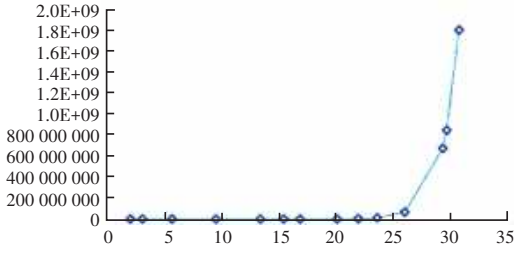

图 5 随机采样点线图

Fig. 5 Diagram of random sampling points

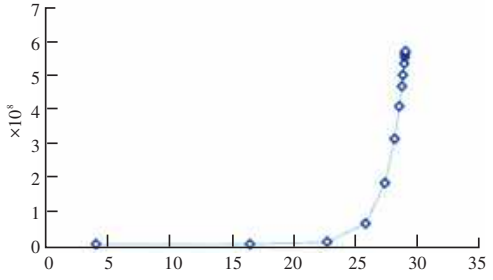

图 6 自相似采样点线图

Fig. 6 Diagram of self-similar sampling point

2.2 流量感知设计

应用层的报文发送条件是:当路由邻居下一跳地址是链路层目的地址时就发送报文,所以链路层在检测到路由邻居下一跳地址是链路层目的地址时就记录数据包发送数量;再用定时器 T 计算此时间段流量值 N,保留 N 与 T,在下一次测量时作为对比数据;最后根据 N 的变化调整 T 的大小,算法 4 和算法 5 描述了自相似性采样法的流量感知过程。

算法 4 EATSM 流量感知算法 1

```
//时刻记录采集的数据包数量
1.  uint16_t dest_id = TSCH_LOG_ID_FROM_LINKADDR(addr);
2.  uip_ds6_nbr_t * nbr = nbr_table_head(ds6_neighbors);
3   while (nbr ! = NULL) {
4.    uint16_t nbr_id = ((nbr -> ipaddr.u8[14]) < < 8) | (nbr -> ipaddr.u8[15]);
5     if (dest_id == nbr_id) {
6.      nbr -> num_tx ++;
7.  }
8.   nbr = nbr_table_next (ds6 _ neighbors, nbr);}
```

算法 5 EATSM 流量感知算法 2

```
1.  void select_N (void * ptr)
2.  {   uip_ds6_nbr_t * nbr;
3     uint16_t nbr_id;
4.    nbr = nbr_table_head (ds6_neighbors);
5.     while (nbr ! = NULL) {
6.          nbr_id = ID_FROM_IPADDR(&(nbr ->ipaddr));
7.          change_queue_N_update(nbr_id, nbr -> num_tx);
8.    }
9.     nbr = nbr_table_next (ds6 _ neighbors, nbr);
       //采集过流量值以后将 nbr->num_tx 置零
10.      nbr = nbr_table_head (ds6_neighbors);
11.     while (nbr ! = NULL) {
12.       nbr_id = ID _FROM _IPADDR (&(nbr ->ipaddr));
13.        nbr -> num_tx = 0;
14.    }nbr = nbr_table_next (ds6_neighbors, nbr);
```

2.3 流量自适应设计

(1)传输调度策略生成。接收方根据各邻居节点流量值信息设计调度表,由于帧解析流量值过程与时隙调度分配过程是分开处理的,EATSM 将设置一个共同结构体 tsch_neighbor 来存储调用相关数据信息。算法 6 描述了帧解析流量值设计,算法 7 描述了时隙调度分配设计。

算法 6 EATSM 帧解析流量数据算法

```
1.  frame802154_t frame;
    //获取帧 N 信息
2.  {uip_ds6_nbr_t * nbr = nbr_table_head(ds6_neighbors);
```

```
3.  while (nbr ! = NULL) {
4   uint16_t old_N = 0;
5.  uint16_t new_N = frame->pigg1;
6.  uint16_t nbr -> nbr_sf_handle = old_N + new_N;
7.   old_N = new_N;
8.  uint16_t nbr -> new_N = new_N;
9.    }
10.     nbr = nbr_table_next (ds6_neighbors, nbr);
```

算法 7　EATSM 时隙调度分配算法

```
1.  void post_process_rx_N(void)
2.  uip_ds6_nbr_t *nbr = nbr_table_head (ds6_neighbors);
3.  while (nbr ! = NULL) {
4.      uint16_t nbr_id = ((nbr -> ipaddr.u8[14]) << 8) | (nbr -> ipaddr.u8[15]);
5.      uint16_t i;
6.      struct tsch_slotframe * sf;
7.      struct tsch_link *l;
        //流量值 N 确定 slotframe 的 handle 和 size
8.      sf = tsch_schedule_add_slotframe
(nbr -> nbr_sf_handle, nbr -> nbr_N);
9.      if (sf! = NULL) {
10.       for (i = (nbr -> nbr_sf_handle - nbr -> nbr_N);
                i < nbr -> nbr_N;  i ++) {
11.       l = tsch_schedule_add_link(
12.        sf, LINK_OPTION_RX,
                LINK_TYPE_NORMAL,
                  &tsch_broadcast_address, i,
                    channel_offset);
13.  } }nbr = nbr_table_next (ds6_neighbors, nbr);
14.  }
```

(2)传输调度策略使用。新的调度策略产生，接收方需通过 ACK 帧告知发送方。发送方在发送数据包前将对 ACK 帧解析并根据新的调度表发送链路数据。算法 8 描述了帧解析调度表设计，算法 9 描述了更改自身调度表并发送链路数据的设计。

算法 8　帧解析调度数据算法

```
1.  frame802154_t frame;
2.  tsch_packet_parse_eack(&frame);
3.  uint16_t nbr -> my_sf_handle = frame -> pigg2;
4.  uint16_t nbr -> my_N = frame->pigg3;
```

算法 9　更改调度表并发送链路数据算法

```
1.  void post_process_rx_t_offset(void)
2.  { uint16_t channel_offset = 3;
3.      uip_ds6_nbr_t *nbr = nbr_table_head (ds6_neighbors);
4.      while (nbr ! = NULL) {
5.          uint16_t id = ((prt_nbr -> ipaddr.u8[14]) << 8) | (prt_nbr -> ipaddr.u8[15]));
6.          uint16_t nbr_id = ((nbr -> ipaddr.u8[14]) << 8) | (nbr -> ipaddr.u8[15]));
7.          uint16_t i;
8.          uint16_t channel_offset = 3;
//设置传输信道
9.          struct tsch_slotframe * sf;
10.         struct tsch_link *l;
11.          sf = tsch_schedule_add_slotframe (nbr -> my_sf_handle, nbr ->my_N);
12.          for (i = (nbr -> my_sf_handle - nbr ->my_N); i < size; i ++) {
13.     l = tsch_schedule_add_link(
            sf, LINK_OPTION_TX, LINK_TYPE_NORMAL, &tsch_broadcast_address, i, channel_offset);
14.          change_queue_select_packet(nbr_id, nbr -> my_sf_handle, i);
15.          } }nbr = nbr_table_next (ds6_neighbors, nbr);
16.  }
```

3　仿真设计与结果分析

3.1　仿真设计

Cooja 是 Contiki[17] 自带的节点和网络仿真工具，主要用于验证和评估 Contiki 系统在不同网络场景下的性能。且具有直接使用编译后的 Contiki 系统固件进行软件模拟运行的能力，通过创建多节点的网络场景来实现仿真。

在 EATSM 的测试和仿真中，为满足项目实际需求，使用了第一种信道模型 UDGM Distance Loss。这种信道模型的主要特点是通信信号的强度随距离的指数衰减，在使用时只需要指定通信范围和干扰范围两个参数即可。

仿真过程中的各项数据将通过以下插件获得：

通过 Network 插件展现网络的拓扑结构并实时显示数据流向;通过 Mote output 插件实时显示所有节点的标准输出信息;通过 Timeline 插件在时间轴上观察每个节点无线设备的开关状态和数据收发情况;通过 Radio messages 插件对无线介质上传输的信息进行抓取并解析。观察运行过程是否正常,再从多种数据进行仿真结果分析。

EATSM 选取 2 个节点进行通信,使用 Cooja 仿真器,运行时间长达 72 h,对生成的上行路由的数据包收发信息进行统计分析。

3.2 结果分析

本文将选取同为流量感知自适应调度技术的主流算法 OST[18] 作为对比对象。

数据包投递率对比如图 7 所示。由图 7 可知,EATSM 设计的传递调度投递率高达 99.57%。并且与 OST 实验数据信息进行统计对比发现,相比较 OST 的 99.31% 的投递率,EATSM 的投递率在 OST 基础上增加了 0.26 个百分点。

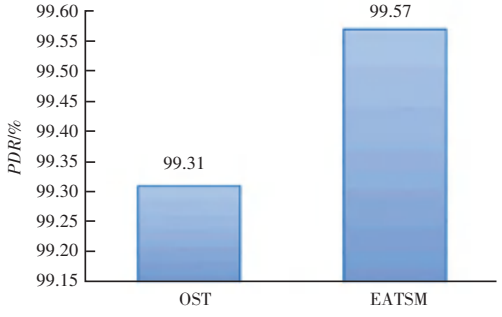

图 7 数据包投递率对比图

Fig. 7 Comparison of packet delivery rates

节点运行总占空比对比如图 8 所示。由图 8 可知,EATSM 设计的节点运行占空比低至 0.34%。并且 EATSM 的设计相比较 OST,占空比减少了 0.88 个百分点。证明了本文提出的 EATSM 设计节点降低能耗方面的性能做到了优化。

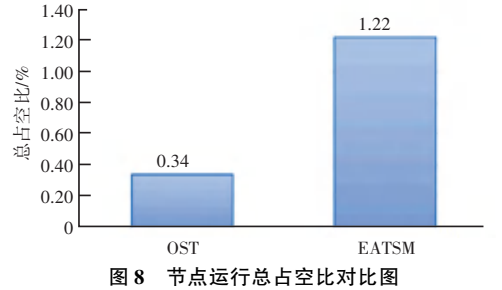

图 8 节点运行总占空比对比图

Fig. 8 Comparison of duty cycle of nodes

节点在时隙操作过程中发送进程与接收进程的占空比对比如图 9 所示。图 9 中,显示 EATSM 在时隙接收过程占空比高一些,这是由于 EATSM 的调度设计部分主要工作在接收节点,所以接收数据时无线电开的时间稍长点,但是结合图 8 可以看出,EATSM 在节约能耗上还是更加优秀的。

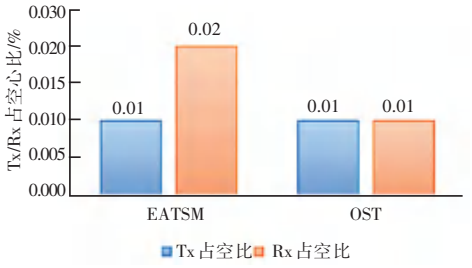

图 9 节点 Tx/Rx 占空比对比图

Fig. 9 Comparison of Tx/Rx duty ratios of nodes

4 结束语

本文基于 IEEE802.15.4 标准的 TSCH 的信道跳频技术展开工作内容。首先介绍了 LLN 网络中 TSCH 技术的应用不够灵活,基于此问题,本文提出的 EATSM 流量感知自适应调度算法通过自相似采样对获取各个链路流量信息,再对这些流量信息值设计公共时隙帧并分配独立的时隙插槽。传输调度过程的流量信息 N 和调度表信息分别加载到数据报文与 ACK 帧上,不会产生额外的信令开销。最后在实验部分,本文设计的 EATSM 算法的投递率高达99.57%,EATSM 比主流算法 OST 也要高 0.26 个百分点。EATSM 的占空比为 0.34%,且相较于 OST 减少了 0.43 个百分点。实验结果表明,本文的基于 TSCH 的流量感知自适应调度法(EATSM)能在保持低能耗的同时提高投递率。

参考文献

[1] 冉凌宇. “物联网+人工智能”:Web3.0 时代的数字传媒发展初探[J]. 出版广角,2021(7):70-72.

[2] 李丰,伍彩虹. 无线传感器网络在环境监测中的运用分析[J]. 皮革制作与环保科技,2022,3(23):55-56,66.

[3] IEEE Standards Association. IEEE standard for low-rate wireless networks[S]. New York, USA: IEEE Std 802.15.4™-2015, 2015.

[4] HAMMOUDI S, BENTALEB A, HAROUS S, et al. Scheduling in IEEE 802.15.4e time slotted channel hopping: A survey[C]// 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). New York, USA :IEEE, 2020: 331-336.

[5] JUNG D, ZHANG Zhenjie, WINSLETT M. Vibration analysis for iot enabled predictive maintenance [C]//2017 IEEE 33rd International Conference on Data Engineering (icde). San Diego, USA :IEEE, 2017: 1271-1282.

[6] KUMAR S, ANDERSEN M P, KIM H S, et al. Bringing full-scale TCP to low-power networks[C]//Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems. New

York，USA：ACM，2018：386-387.
[7] KIM H S，KO J G，BAHK S. Smarter markets for smarter life：applications，challenges，and deployment experiences[J]. IEEE Communications Magazine，2017，55(5)：34-41.
[8] REINA D G，ASKALANI M，TORAL S L，et al. A survey on multihop ad hoc networks for disaster response scenarios[J]. International Journal of Distributed Sensor Networks，2015，11(10)：647037.
[9] AL-HABASHNEH A A Y，AHMED M H，HUSAIN T. Adaptive MAC protocols for forest fire detection using wireless sensor networks[C]//2009 Canadian Conference on Electrical and Computer Engineering. Newfoundland，Canada：IEEE，2009：329-333.
[10] LI Qiyue，TANG Haocheng，SUN Wei，et al. Optimal resource allocation of 5G machine-type communications for situation awareness in active distribution networks[J]. IEEE Systems Journal，2021，16(3)：4187-4197.
[11] MAKKA S，SREENIVASULU K，RAWAT B S，et al. Application of blockchain and Internet of Things (IoT) for ensuring privacy and security of health records and medical services[C]//2022 5th International Conference on Contemporary Computing and Informatics (IC3I). Cairo，Egypt：IEEE，2022：84-88.
[12] WINTER T，THUBERT P，BRANDT A，et al. RPL：IPv6 routing protocol for low-power and lossy networks[EB/OL]. [2011-04-28]. https://doi.org/10.17487/RFC6550.
[13] 董慧颖. 基于自相似性的网络流量抽样方法研究与应用[D]. 哈尔滨：哈尔滨理工大学，2008.
[14] ADRIAN S J，ALEXANDRU I，STOLOJESCU-CRISAN C，et al. Network self-similar traffic generator with variable hurst parameter[C]//2020 International Symposium on Electronics and Telecommunications (ISETC). Timisoara，Romania：IEEE，2020：1-4.
[15] IVANISENKO I，KIRICHENKO L，RADIVILOVA T. Investigation of self-similar properties of additive data traffic[C]//2015 Xth International Scientific and Technical Conference " Computer Sciences and Information Technologies"(CSIT). Lviv，Ukraine：IEEE，2015：169-171.
[16] HIRCHOREN G A，PORREZ N，SALA L B，et al. Quality of service in networks with self-similar traffic[C]//2017 XVII Workshop on Information Processing and Control (RPIC). Resistencia，Argentina：IEEE，2017：1-5.
[17] DUNKELS A，GRONVALL B，VOIGT T. Contiki-a lightweight and flexible operating system for tiny networked sensors[C]//29th Annual IEEE International Conference on Local Computer Networks. Washington DC，USA：IEEE，2004：455-462.
[18] JEONG S，KIM H S，PAEK J，et al. OST：On-demand TSCH scheduling with traffic-awareness[C]//IEEE INFOCOM 2020-IEEE Conference on Computer Communications. Toronto，Canada：IEEE，2020：69-78.