WSN 中基于最小延时的数据汇集树构建与传输调度算法

高 蕾[1] 胡玉鹏[2]
(惠州学院计算机科学系 惠州 516007)[1] (湖南大学软件学院 长沙 410082)[2]

摘 要 针对现有的无线传感器网络数据汇集算法延时较大的不足,对最小延时数据汇集树和传输调度问题进行了研究。提出一种基于度约束的汇集树构建算法(DCAT)。该算法按照 BFS 方式遍历图,当遍历到每个节点时,通过确定哪些节点与汇点更近来确定潜在母节点集合。然后,选择图中度数最小的潜在母节点作为当前被遍历节点的母节点。此外,为了在给定的汇集树上进行高效的数据汇集,还提出两种新的基于贪婪的 TDMA 传输调度算法:WIRES-G 和 DCAT-Greedy。利用随机生成的不同规模的传感器网络,参照当前最新算法,对所提方法的性能进行了全面评估。结果表明,与当前最优算法相比,将所提调度算法与所提汇集树构建算法结合起来,可显著降低数据汇集的延时。

关键词 无线传感器网络,数据汇集,最小延时,度约束,传输调度
中图法分类号 TP393 **文献标识码** A

Data Aggregation Tree Construction and Transmission Scheduling Algorithm Based on Minimum Latency in Wireless Sensor Networks

GAO Lei[1] HU Yu-peng[2]
(Computer Science Department, Huizhou University, Huizhou 516007, China)[1]
(Software College, Hunan University, Changsha 410082, China)[2]

Abstract Aiming at the shortcomings of the larger delay at the existing data aggregation algorithms in wireless sensor networks, we studied the problem of the minimum latency data aggregation tree and transmission scheduling. An aggregation tree construction algorithm based on degree constraint(DCAT) was proposed. It works by traversing the graph in a BFS manner. As it traverses each node, the set of potential parents is determined by identifying the nodes that are one-hop closer to the sink. The potential parent with the lowest degree in the graph is selected as the parent for the currently traversed node. Furthermore, we proposed two new approaches based on greedy for building a TDMA transmission schedule to perform efficient aggregation on a given tree: WIRES-G and DCAT-Greedy. We evaluated the performance of our algorithms through extensive simulations on randomly generated sensor networks of different sizes and we compared them to the previous state of the art. The results show that new scheduling algorithms combining with our new tree-building algorithm obtain significantly lower latencies than that of the previous best algorithm.

Keywords Wireless sensor networks, Data aggregation, Minimum latency, Degree constraint, Transmission scheduling

在无线传感器网络的多种应用中,数据由传感器节点采集后发往汇点(即 Sink)处,这种通信模式称为汇集模式[1-3]。该模式构建以汇点为根并通往汇点的树,然后沿着树向汇点传输报文,进而完成数据汇集。在部分应用中,汇集树上的部分节点接收到子节点的数据后,首先对数据进行汇集,然后再发往母节点,以降低需要传输的报文数量。数据汇集技术可将汇集操作时传输的报文数量从 $\Omega(n^2)$ 下降到 $O(n)$,极大地节约了网络能耗[4-6]。文献[7]研究了用单位圆盘图表示的传感器网络中的最小延时汇集调度问题(Minimum Latency Aggregation Scheduling, MLAS),提出了一种集中式($\Delta-1$)近似算法,称为最短数据汇集算法(Shortest Data Aggregation, SDA),其中 Δ 表示图中节点的最大度。然而,该算法性能的优劣依赖于网络拓扑结构,可扩展性较差。Huang 等人[8]提出一种基于 MIS 的集中式算法来求解 MLAS 问题,延时为 $23R+\Delta-18$,其中 R 表示汇点和其他任意节点的最大距离。然而,该算法在求解 MIS 的过程中需要节点多次交换信息,时间复杂度较高。

此外,文献[9]提出了被称为 BSPT(均衡式最短路径树)的构建算法以及被称为 WIRES(基于加权增量排序的汇集调度)的算法来实现数据汇集。BSPT 算法给出了数据汇集延时的范围为 $\max\{\xi_i+h_i: i=1,2,\cdots,n\}$ 的下界,其中 ξ_i 和 h_i 分别为指定树中节点 i 从根节点开始的子节点数量和跳数。它采取宽度优先搜索方式遍历图,然后采用双枝半匹配算法[10]来构建可使延时最小的最短路径树。而 WIRES 调度算

本文受国家自然科学基金(61300218)资助。
高 蕾(1976-),女,硕士,副教授,主要研究方向为无线传感器网络、信息检索;胡玉鹏(1981-),男,博士,副教授,CCF 会员,主要研究方向为无线传感网、云计算。

法则将汇集树作为输入，并将树中所有叶节点作为可在单位时间内被调度的合格节点。为每个合格节点计算一个权重，权重越高，表明该节点在当前时隙内被调度的优先级越高。然后依次考察合格节点，对在传输时不与先前节点发生干扰的所有节点进行调度。所有节点考虑完毕后，一个轮次完毕，通过删除已被调度的节点，增加从各个子节点接收到数据的母节点来更新合格节点集合。重复上述步骤，直到所有节点被调度一次。然而该方法使用汇集树中非叶相邻节点的数量来进行权重计算，因为节点被调度后从汇集树中删除，所以每一轮次均需重新计算权重，导致数据汇集延时增大，且额外耗费了能量。

针对以上方法的不足，本文提出一种新的汇集树构建算法，称为度约束汇集树（Degree-Constrained Aggregation Tree，DCAT）。此外，我们还提出两种新的调度算法，称为 WIRES-G 和 DCAT-Greedy。通过全面的仿真实验评估了本文汇集树构建算法和调度算法的性能，并与当前最优算法 BSPT-WIRES[9] 进行了性能比较。结果表明，DCAT 算法与 WIRES 调度算法相结合可将延时性能提升 21%。如果将本文调度算法 WIRES-G 与 DCAT 算法相结合，可实现进一步的性能提升，将这种融合算法称为 DCAT-WIRES-G。此外，DCAT-Greedy 的性能比 BSPR-WIRES 高出 32%～40%，具体取决于网络规模的大小。

1　网络模型和问题描述

本文利用单位圆盘图来模拟无线传感器网络中的数据汇集过程，然后对 TDMA 调度问题进行研究。如果两个节点互相位于对方的传输范围内，则认为这两个节点连通。沿着图的生成树进行数据汇集。在传输时隙期间必须对链路进行调度，以便使可能发生干扰的链路在不同时隙内传输数据，同时使每个节点在其所有子节点传输完毕后再传输数据，保证数据汇集的有效性。本文采用基于图的干扰模型[11]：如果 v_1 在 u_2 的传输范围内，则认为链路 (u_1,v_1) 和 (u_2,v_2) 对接收器 v_1 产生干扰。一次汇集操作的延时定义为汇点接收到所有节点的数据所需要的时间。

假设节点同步，且共享相同的无线信道。假设所有节点固定布置，传输范围相同且恒定。同时假设干扰半径等于传输半径[12]。时间经过时隙处理，每个节点经过调度后在指定时隙内传输数据。如果在同一时隙内传输数据时不会发生干扰，则两个节点可在同一时隙内传输数据。我们还假设节点具有求取最小值、最大值、求和以及计数的功能，将 n 个数据元素作为输入，产生的一个元素作为输出。

已知一组传感器节点 $S=\{s_0,s_1,\cdots,s_{n-1}\}$，其中 s_{n-1} 表示汇点，每个节点均有一个数据需要传输给汇点。我们希望找到一种传输调度策略，使所有节点在各自子节点传输完毕后自己只需传输一次，便可将所有融合数据发往汇点，同时不发生干扰。用图 $G=(V,E)$ 表示一个无线传感器网络，$s\in V$ 表示汇点，我们为图 G 定义一个生成树 T 作为它的有效调度，该树以汇点 s 为根并通往汇点，对于数据传输任务 $A:V\rightarrow Z^+$，要求：

(1) $v\in children(u)\Rightarrow A(u)>A(v)$；

(2) $(u,v)\in T$ 且 $(\omega,v)\in G\Rightarrow A(u)\neq A(\omega)$。

条件(1)可保证每个节点在其子节点传输完毕后才开始传输，保证数据经过融合；条件(2)可保证传输过程未被干扰。图 G 有效调度 A 的延时可表示为 $L(G,A)$，并定义 $L(G,A)=\max_{v\in V}\{A(v)\}$。于是，MLAS 问题可表示如下：已知图 $G=(V,E)$，为图 G 寻找一种可使延时最小化的有效调度。很显然，这一问题可分为两个阶段：汇集树构建过程和基于树的调度策略搜索过程。下面对这两个问题进行研究。

2　基于度约束的汇集树（DCAT）

本节给出了基于度约束的汇集树构建算法。当对节点的潜在母节点进行选择时，本文算法的主要思想就是选择图中度数最小的节点。原因是潜在母节点的度非常关键，只有对节点进行调度，才能避免受到其母节点在图中其他子节点的干扰，同时避免受到母节点在图中其他相邻节点的干扰。本文算法与 BSPT 算法不同，BSPT 算法通过将节点均匀分布于它们的潜在母节点之间以尽量提高并行化水平，即该算法尽量降低树中节点的度，并忽略潜在母节点在图中的相邻节点。而 DCAT 的具体内容见算法 1。首先，按照 BFS 方式遍历图，当遍历到每个节点时，通过确定哪些节点与汇点更近（少一跳距离）来确定潜在母节点集合。然后，选择图中度数最小的潜在母节点作为当前被遍历节点的母节点。与基于 BFS 的算法类似，本文 DCAT 算法的时间复杂度为 $O(|V|+|E|)$，因为每个顶点和每条边均被探索一次。

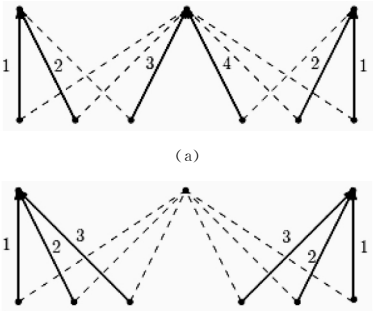

图 1　DCAT 算法和 BSPT 算法构建的汇集树比较

图 1 中，实线表示汇集树的边，虚线表示图中未作为树边的边。图 1(a)示出了 BSPT 算法构建的汇集树，可以看出，该树的最优调度需要 4 个时隙。图 1(b)示出了 DCAT 算法针对同一图形构建的汇集树，此时，最优调度只需要 3 个时隙。图 1 表明了 DCAT 中采用的方法的性能优于 BSPT 的原因。图 1 中，子节点被均匀分布于潜在母节点之间，且与它们在图中的度无关，于是调度算法受到的约束更多。从图 1 可以清晰地看出，考虑图中节点的度数，其性能要优于使汇集树的度最小化。这表明，DCAT 更适合构建高效的汇集树，至少当不同节点具有不同度数时如此。

算法 1　DCAT

输入：G=(V,E)，s：汇点

输出：图 G 以 s 为根的生成树，其中 v.p 表示 v∈V 的母节点

1. procedure DCAT(G,s)

```
2.  for each u∈G.V do
3.    u.d=-1
4.    u.p=nil
5.  end for
6.  s.d=0
7.  Q=Ø
8.  ENQUEUE(Q,s)
9.  while Q≠Ø do
10.   u=Dequeue(Q)
11.   for each v∈N(u)do
12.     if v.d<0 then
13.       v.d=u.d+1
14.       ENQUEUE(Q,v)
15.     end if
16.     if v.d>u.d then
17.       if v.p==nil or |N(v.p)|>|N(u)| then
18.         v.p=u
19.       end if
20.     end if
21.   end for
22. end while
23. end procedure
```

3 调度算法

本节给出了 2 种用于数据汇集的调度算法。第一种算法称为 WIRES-G，是对文献[9]中 WIRES 算法的一种改进，我们在 WIRES 算法中增加一个步骤，以便在每个时隙期间调度更多个节点。每一轮次中，新添步骤需要为原算法无法调度的合格节点匹配新的母节点。WIRES-G 的具体内容请见算法 2(G 表示贪婪)。工作流程如下：第 5－9 行表示原 WIRES 算法每一轮次的步骤，第 9 行结束时，S 包含经过调度将在时间 j 发送数据的节点，R 包含从 S 中节点接收数据的节点集合。此时，如果继续保持原来的树，则其他所有节点经过调度后再传输数据总会与已被调度的节点发生干扰。

算法 2 WIRES-G

```
输入:G=(V,E),s:汇点,v.p:树中 v∈V 的母节点
输出:G 的一个有效调度,其中 v.t 表示 v∈V 的传输时间
1. procedure WIRES-G(G;s)
2.   ∀v∈G.V v.t=0 //对时隙初始化
3.   j=1
4.   while s.t=0 do
5.     L=GETELIGIBLENODES(G)
6.     COMPUTEWEIGHTS(L)
7.     SORTDECREASING(L)
8.     S=R=Ø//发送节点和接收节点集合开始时为空
9.     SCHEDULENODES(L;S;R)
10.      L=L\S //将发送节点从合格节点中删除
11.      GREEDY-SCHEDULING(L;S;R)
12.    j=j+1
13.  end while
14. end procedure
15. procedure SCHEDULENODES(L;S;R)
16.  for each u∈L do
17.    if u∉N(R)且 u.p∉N(S) then //如果 u 在传输时不发生冲突
18.      u.t=j //通过调度使 u 在时间 j 传输
19.      S=S∪{u}
20.      R=R∪{u.p}
21.    end if
22.  end for
23. end procedure
24. procedure GREEDY-SCHEDULING(L;S;R)
25.  for each u∈L do
26.    if u∉R 且 u∈N(R) then
27.      r=nil//未发现母节点
28.      for each p∈N(u) do //寻找母节点
29.        if p.t==0 且 p∉N(S)且(r=nil 或|N(p)|<|N(r)|)
           then
30.          r=p
31.        end if
32.      end for
33.      if r≠nil then //如果找到新的母节点
34.        p=u.p
35.        u.p=r //分配新的母节点
36.        if ISELIGIBLE(p) then
37.          L=L∪{p}
38.      end if
39.      u.t=j
40.      S=S∪{u}
41.      R=R∪{u.p}
42.        end if
43.    end if
44.  end for
45. end procedure
```

在算法 2 的 GREEDY-SCHEDULING 中，我们对之前未被顺利调度的所有合格节点进行迭代。首先，考查当前节点 p 是否已经不再作为接收节点，同时考察该节点在传输数据时是否会对 R 中的接收节点造成干扰。如果如此，则为 p 选择一个未被调度且可在时间 j 传输数据并不发生干扰的相邻节点，同时该相邻节点的度在图中所有类似相邻节点间最小。找到相邻节点后，便将其作为当前节点在汇集树中新的母节点。如果先前母节点未遗留下未被调度的子节点，并且在当前轮次内不作为接收节点，则将其添加到合格节点列表中，原因是它可在时间 j 被调度(第 36－38 行)。

在算法 2 中，GETELIGIBLENODES 的时间复杂度为 $O(|V|)$，COMPUTEWEIGHTS 的时间复杂度为 $O(|V|+|E|)$。第 7 行在最坏情况下的时间复杂度为 $O(|V|\log|V||)$。SCHEDULENODES 和 GREEDY-SCHEDULING 算法在每次调用时均遍历所有节点和边，时间复杂度为 $O(|V|+|E|)$。第 4 行的外层循环最多执行 $|V|$ 次。因此，WIRES-G 和 WIRES 类似，其时间复杂度为 $O(|V|^2\log|V|+|V||E|)$。

本文提出的第二种调度算法称为 DCAT-Greedy，具体内容见算法 3。该算法融合了本文的汇集树构建算法 DCAT 及 GREEDY-SCHEDULING 算法，其目的是进一步降低延时。

DCAT-Greedy 算法首先采用 DCAT 构建一个汇集树。每次迭代时，确定哪些节点有资格在此轮接受调度。只有所有子节点均被分配了一个时隙的节点才会成为有资格被调度的节点（合格节点）。DCAT-Greedy 采用与 WIRES 相同的方法计算权重（非叶相邻节点的数量），并根据权重对节点降序排列。然后，GREEDY-SCHEDULING 对所有合格节点进行迭代，在不发生干扰的前提下使尽可能多的节点被调度。

算法 3　DCAT-Greedy

输入：G=(V,E)，s：汇点

输出：G 以 s 为根的生成树，及 G 的一个有效调度，其中 v.t 和 v.p 分别表示 v∈V 的传输时间和母节点

```
1. procedure DCAT-GREEDY(G)
2.   DCAT(G;s)
3.   ∀v∈G.V v.t=0//时隙初始化
4.   j=1
5.   while s.t=0 do
6.     L=GETELIGIBLENODES(G)
7.     COMPUTEWEIGHTS(L)
8.     SORTDECREASING(L)
9.     S=R=Ø
10.    GREEDY-SCHEDULING(L;S;R)
11.    j=j+1
12.  end while
13. end procedure
```

DCAT-Greedy 算法与 WIRES-G 有两点不同：1）它利用 DCAT 来构建树；2）没有调用 SCHEDULENODES。从先前讨论可以看出，DCAT-Greedy 的时间复杂度为 $O(|V|^2\log|V|+|V||E|)$。

4　仿真结果

本文结合小型、中型和大型无线传感器网络对汇集树构建算法 DCAT 及传输调度算法进行性能评估，通过将节点随机均匀分布于 5*5，10*10 和 20*20 的地理区域上生成小型、中型和大型网络。节点的传输范围为 1，节点的密度范围为 8～200，其中密度定义为图中节点的平均度。如果节点间的欧氏距离小于或等于传输范围，则认为节点连通。对每个被选密度值共生成 100 个连通图，对这 100 个图求取平均作为最终结果。

首先，单独评估汇集树构建算法。为此，我们比较了 DCAT-WIRES 和 BSPT-WIRES 的性能，同时衡量了所生成调度方案的延时。如图 2 所示，网络尺寸为 20*20，采用 WIRES 调度。DCAT 树无论在哪种密度设置下，延时均较低。鉴于篇幅有限，这里只给出大型网络的运行结果，对小型和中型网络具有相同结论。

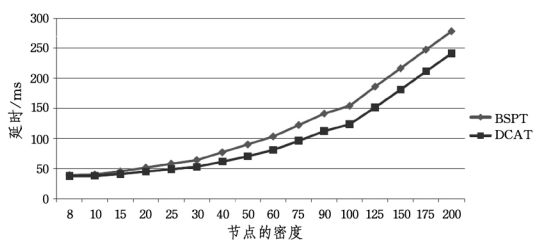

图 2　DCAT 和 BSPT 树的平均汇集延时性能比较

图 3 在一定程度上表明了 DCAT 算法的性能优于复杂性更高的 BSPT 算法的原因。该图给出了图中相邻节点数量与汇集树上子节点平均数量之间的关系。可以看出，与 DCAT 相比，BSPT 分配给度数更高节点（高度节点）的子节点更多，而 DCAT 倾向于为度数低于网络密度的节点分配较多子节点。如果以度数较高的节点作为母节点，则调度算法可在同一时隙内调度更多个节点，有助于降低调度的总体延时。

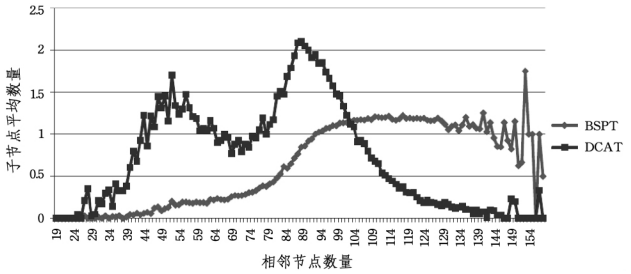

图 3　图的度数与汇集树子节点平均数量之间的关系

下面评估 WIRES-G 和 DCAT-Greedy 2 种调度算法的性能。图 4 给出了 WIRES-G，DCAT-Greedy 和 WIRES 在小型传感器网络上的性能比较。很显然，DCAT-Greedy 对小型网络的性能最优。可以看出，DCAT-Greedy 所生成的调度方案的平均延时要远低于其他算法，当密度设置较高时更是如此。DCAT-Greedy 甚至远低于 BSPT 生成的汇集树的下界。当密度为 200 时，DCAT-Greedy 的性能比排名第二的算法 DCAT-WIRES-G 高出 25%，比先前的最优算法 BSPTWIRES 高出近 40%。此外，用百分比表示的性能增益随着密度稳定上升。鉴于篇幅所限，对中型网络的运行结果类似，此处略。可以看出，无论在哪种情况下，WIRES-G 均可降低调度方案的延时，且与采用的树构建算法无关；但 WIRES 无法实现这一性能。BSPTWIRES-G 和 DCAT-WIRES 的性能比较接近，前者略优于后者。因此，无论是采用本文新提出的调度算法 WIRES-G，还是新提出的树构建算法 DCAT-Greedy，均可实现性能提升。DCAT-WIRES-G 的性能优于其他算法，但低于 DCAT-Greedy，且密度越大，性能增益越大。

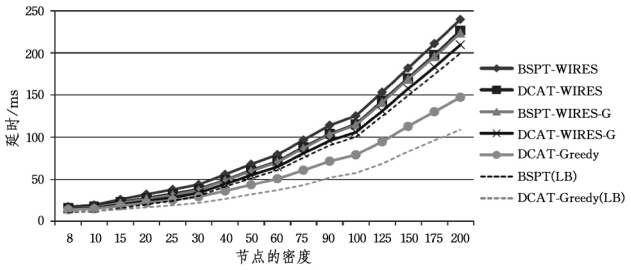

图 4　网络规模为 5*5 时的平均汇集延时

图 5 给出了调度算法对大型传感器网络的性能。结果特点与上文类似。然而，几乎在所有密度设置下，DCAT-WIRES 的性能均优于 BSPT-WIRES-G。实际上，在中等密度水平，DCATWIRES-G 的性能比 BSPT-WIRES-G 高出 15%。DCAT-Greedy 和 DCAT-WIRES-G 的性能相近，但当密度在 15 到 75 之间（含）时，DCAT-WIRES-G 略优于 DCAT-Greedy。与 BSPT-WIRES 相比，DCAT-WIRES-G 的性能提升了 34.66%，而 DCAT-Greedy 相对于 BSPT-WIRES

提升了 32.25%。请注意，DCAT-Greedy 结果的标准差低于 DCAT-WIRES-G(4.48 vs 5.35)。

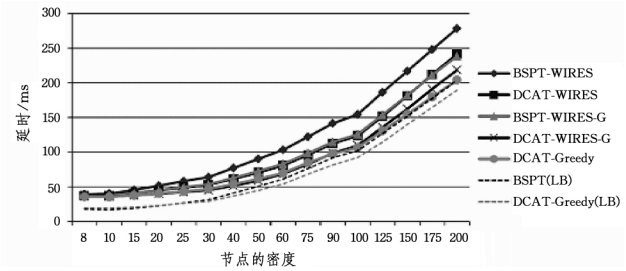

图 5 网络规模为 20×20 时的平均汇集延时

为了更好地理解性能特点，我们考察了图中节点度与汇集树中子节点平均数量之间的关系。图 6 给出了密度为 30 时中等随机网络下的关系。可以看出，各种基于 DCAT 的算法为度数较高的节点分配的子节点数量很少，这与我们的预期相一致。DCAT-Greedy 算法为度数较低的节点分配的子节点数量最多，为高度节点分配的子节点数量最少。这也是该算法在实验中表现优异的原因。对高密度的设置具有类似特点。

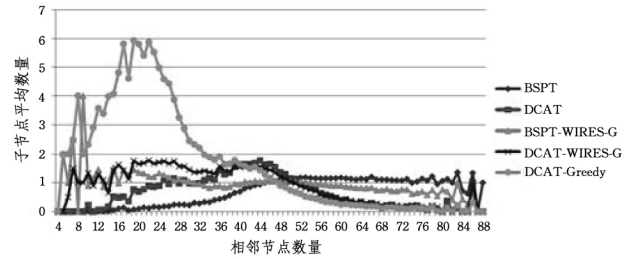

图 6 图的度数与汇集树子节点平均数量之间的关系

另外，我们还考察了汇集树中子节点数量最多的节点所处的位置。图 7 给出了中等规模网络在密度为 30 时的运行结果。可以看出，汇点(距离为 0 的节点)在除 DCAT-Greedy 的各种算法下，子节点数量均较高。这是因为我们开始时采用最短路径树，所以所有与汇点相距一跳距离的节点将是汇点的子节点。DCAT-Greedy 不存在这个问题，因为它为了同时调度尽可能多的节点而修改了初始树。

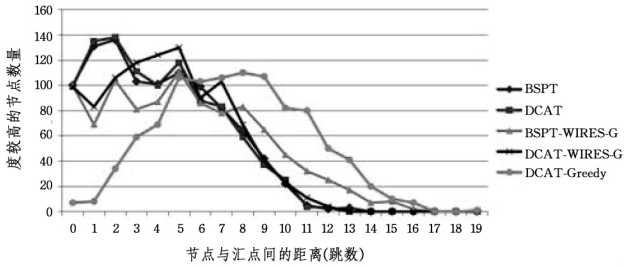

图 7 汇集树中度数较高节点的位置(密度为 30)

此外，我们发现，如果度较高的节点与树顶较近，往往会导致延时上升。这是因为若与汇点的距离近，可供选择的节点减少，并行化的概率则会降低。这使度较高的节点位于树的底层，也不利于性能提升，因为只有当所有节点传输完毕才能使数据向上传输，这解释了对于密度中等的大型网络为何 DCATWIRES-G 的性能要优于 DCAT-Greedy。为此，一种可能的思路是设计一种将上述两种算法综合起来的混合算法来提升面对大型传感器网络时的性能。例如，对树的底层采用 DCAT-WIRES-G 算法，对树的高层采用 DCAT-Greedy 算法。

结束语 本文研究了 WSN 中进行 TDMA 调度时的数据汇集问题，提出一种新的汇集树构建算法及两种新的调度算法。与当前最优算法相比，如果将本文提出的汇集树构建算法与调试算法结合起来，可显著降低延时。在下一步工作中，我们研究的重点主要包含两个方面：1)在多种应用场景下分析数据汇集可靠性与汇集树构建以及调度算法之间的关系，以进一步提高数据汇集的质量；2)基于压缩感知理论，分析数据汇集树的构建过程对延长网络生命周期的影响，进而提出一种可提高网络生命周期的基于压缩感知的数据汇集算法。

参 考 文 献

[1] BGAAA M, YOUNIS M, DJENOUI D, et al. Distributed low-latency data aggregation scheduling in wireless sensor networks [J]. ACM Transactions on Sensor Networks (TOSN), 2015, 11(3): 49-60.

[2] 石为人，唐云建，王燕霞. 基于拥塞控制的无线传感器网络数据汇集树生成算法[J]. 自动化学报，2010，36(6)：823-828.

[3] YOUSEFI H, MALEKIMAJD M, ASHOURI M, et al. Fast aggregation scheduling in wireless sensor networks [J]. IEEE Transactions on Wireless Communications, 2015, 14(6): 3402-3414.

[4] LIU X Y, ZHU Y, KONG L, et al. CDC: Compressive data collection for wireless sensor networks [J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(8): 2188-2197.

[5] 杨庚，李森，陈正宇，等. 传感器网络中面向隐私保护的高精确度数据融合算法 [J]. 计算机学报，2013，36(1)：189-200.

[6] 邱立达，刘天键，傅平. 基于稀疏滤波的无线传感器网络数据融合[J]. 电子测量与仪器学报，2015，29(3)：352-357.

[7] GUO L, LI Y, CAI Z. Minimum-latency aggregation scheduling in wireless sensor network [J]. Journal of Combinatorial Optimization, 2016, 31(1): 279-310.

[8] HUANG S C H, WAN P J, VU C T, et al. Nearly constant approximation for data aggregation scheduling in wireless sensor networks[C]// 26th IEEE International Conference on Computer Communications (INFOCOM). Anchorage, Alaska, USA: IEEE Press, 2007: 366-372.

[9] MALHOTRA B, NIKOLAIDIS I, NASCIMENTO M A. Aggregation convergecast scheduling in wireless sensor networks [J]. Wireless Networks, 2011, 17(2): 319-335.

[10] HARVEY N J A, LADNER R E, LOVÁSZ L, et al. Semi-matchings for bipartite graphs and load balancing[J]. Journal of Algorithms, 2006, 59(1): 53-78.

[11] INCEL Ö D, GHOSH A, KRISHNAMACHARI B, et al. Fast data collection in tree-based wireless sensor networks [J]. IEEE Transactions on Mobile Computing, 2012, 11(1): 86-99.

[12] YAO Y, CAO Q, VASILAKOS A V. EDAL: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks [J]. IEEE/ACM Transactions on Networking, 2015, 23(3): 810-823.