无线传感网中一种改进的分布式数据聚集调度算法*

刘文彬[1]，刘红冰[1]，付 沙[1]，文志强[2]
(1. 湖南财政经济学院 信息管理系，长沙 410205；2. 湖南工业大学 计算机与通信学院，湖南 株洲 412008)

摘 要：针对无线传感器网络中实时数据收集具有较高的延时问题，提出了一种改进的无通信冲突的分布式数据聚集调度近似算法。该算法首先在最大独立集的基础上建立一棵根在 sink 的数据聚集树，然后各个节点按数据聚集树分层进行数据调度。在数据聚集树的构造过程中，对于两个相距两跳的支配点，它们共同的、相距两跳的支配点，通过距 sink 最近的支配点加入数据聚集树；而在数据调度过程中，采用一种新的选择标准从竞争集中选择节点进行数据调度。通过这两方面的改进，有效地降低了数据的聚集延时。理论分析表明，该算法的延时上界为 $14R+\Delta$；仿真模拟的结果表明，该算法产生的数据聚集延时远低于现有算法。

关键词：数据聚集；最小延时；无线传感器网络；数据调度算法；通信冲突
中图分类号：TP393 **文献标志码**：A **文章编号**：1001-3695(2014)01-0243-05
doi:10.3969/j.issn.1001-3695.2014.01.057

Improved distributed data aggregation scheduling algorithm in WSNs

LIU Wen-bin[1], LIU Hong-bing[1], FU Sha[1], WEN Zhi-qiang[2]
(1. *Dept. of Information & Management, Hunan University of Finance & Economics, Changsha* 410205, *China*; 2. *School of Computer & Communication, Hunan University of Technology, Zhuzhou Hunan* 412008, *China*)

Abstract: This paper presented an improved distributed data aggregation scheduling algorithm without communication collision for minimum data aggregation latency due to existing algorithms have high time latency for data collection in wireless sensor networks. In this algorithm, it constructed a data aggregation tree rooted at the sink firstly. And then, the node could be scheduled layer by layer according to the data aggregation tree. In the process of the constructing the data aggregation tree, for the common neighboring dominators of two adjacent dominators, it selected a dominator which was close to the sink to join the data aggregation tree. It used new criteria for node selection amongst available competitors in the data aggregation scheduling scheme. Using these modifications, it reduced the latency for the sink collecting all sensors' data effectively. The theoretical analysis shows that the algorithm has a latency bound with $14R+\Delta$. Simulation results show that this algorithm has lower average latency than previous works.

Key words: data aggregation; minimum latency; wireless sensor networks; data scheduling algorithm; communication collision

0 引言

近年来，在无线传感器网络(wireless sensor networks, WSNs)中，最小延时数据聚集调度问题(minimum latency data aggregation scheduling, MLDAS)日益受到人们的普遍关注。这是因为对战场监控、森林火警监测、矿井瓦斯浓度监测等时间敏感的实时应用来说，要求传感器节点在采集到敏感数据后应尽快地将它们传送给 sink 节点进行处理，否则，过时的信息对系统监控毫无意义，甚至会导致灾难性的后果。而在数据收集过程中，如果发生通信冲突，即两个节点采用同一信道同时向其共同的邻居发送数据，那么，节点将不能成功接收数据，从而使 sink 要花费更多的时间来收集数据。此外，尽管数据聚集技术为降低传感器的能量损耗发挥了重要的作用，但是，它因等待接收数据、将数据进行聚合所产生的一些额外的网络延时也是一个不能忽视的问题。因此，针对 MLDAS 问题，如何设计一种有效的、无通信冲突的数据聚集调度方案，使 sink 成功收集所有传感器的数据所花的时间最少，对未来的无线传感器网络的实时应用具有非常重要的理论和应用价值。

目前，针对 MLDAS 问题，已设计了一些有效的集中式数据聚集调度算法[1-6]，但是，这类算法需要 sink 节点知道网络的全部信息、计算调度方案并分发给每个节点。只有当所有节点收到这种调度方案，它们才按照这种方案进行工作。一旦网络的拓补结构发生变化，如节点失效、节点在活动与休眠之间的状态切换等，sink 节点不得不频繁收集新的网络拓扑信息、重新计算调度方案并传送给每个节点，这将会消耗大量的能量，从而使这些集中式算法在实际应用过程中效率很低。

收稿日期：2013-04-03；**修回日期**：2013-05-26 **基金项目**：国家自然科学基金资助项目(61170102)；湖南省教育厅高等学校科学研究项目(12C0558,11C0215)；湖南省重点学科建设资助项目

作者简介：刘文彬(1975-)，男，湖南新化人，讲师，硕士，主要研究方向为无线网络、网络通信、算法优化(liumeitao@hnu.edu.cn)；刘红冰(1969-)，女，副教授，硕士，主要研究方向为网络通信、算法优化；付沙(1980-)，男，湖南长沙人，讲师，硕士，主要研究方向为网络通信、信息安全、数据挖掘；文志强(1973-)，男，湖南湘乡人，副教授，博士，主要研究方向为图像处理、目标检测与跟踪、网络通信.

针对集中式算法所固有的缺陷,人们开始设计分布式算法来解决数据聚集调度问题。文献[7]首次提出了一种无通信冲突的分布式数据聚集调度算法,即 DAS 算法,该算法的延时上界是 $24D+6\Delta+16$,其中 Δ 是 WSNs 中网络的最大度、D 是网络的直径。然而,文献[8]认为 DAS 算法的延时上界应该是 $61D+5\Delta-67$,并提出了一种改进的分布式数据聚集调度算法(简记为 DDAS 算法),其延时上界为 $61R+5\Delta-67$,其中 R 是网络半径。文献[9]则认为 DAS 算法的延时上界不低于 $48R+6\Delta+16$,同时,也设计了一种快速的分布式算法(FDAP 算法),其延时上界是 $17R+6\Delta+8$。这三种算法在最大独立集的基础上建立一棵数据聚集树,然后各个节点按数据聚集树分层进行数据调度。所不同的是,在数据调度过程中,以不同的选择标准在竞争集中选择节点进行数据调度。如 FDAP 算法根据节点层数、度和 ID 来选择被调度的节点,而其余两种算法则只根据节点 ID 来选择被调度的节点。

通过对现有分布式算法进行仔细分析后发现,一方面,在数据聚集树的构造过程中,通过减少支配点所需连接点的个数可有效降低数据聚集延时;另一方面,在数据调度过程中,从竞争集中采用一种新的选择标准选择节点进行数据调度,也可有效降低数据聚集延时。基于以上发现,本文在现有算法的基础上,提出一种改进的无通信冲突的分布式数据聚集调度近似算法,为描述方便,简记为 IDDAS 算法。

1 系统模型

1.1 网络模型

假设无线传感器网络用一个连通的无向图 $G=(V,E)$ 表示,其中 V 表示传感器节点的集合,用 $v_0\in V$ 表示 sink 节点。假设所有节点都有相同的传输半径 r,边 $(u,v)\in E$ 当且仅当这两个节点之间的平面几何距离小于或等于传输半径 r,即 $|uv|\leqslant r$。

假设网络采用单信道进行通信,即在同一时刻,每个节点不能同时接收和发送数据。传感器节点可以与它为中心、通信半径内的所有邻居节点通信。如果一个节点在同一时刻收到两个或两个以上的数据包,则会产生通信冲突。设每个节点的干扰半径为 r_I,在同一时刻,如果 u 发送数据给 v,x 发送数据给 y,并且 $|uy|\leqslant r_I$,则称 y 受到 u 的干扰。如果一个节点在接收数据时受到另一节点的干扰,本文也称之为通信冲突。

1.2 问题陈述

设 $A\subset V$,$B\subset V$ 且 $A\cap B=\varnothing$,如果集合 A 中的节点能在同一时间片(timeslot)内将其数据发给集合 B 中的某些节点而不发生通信冲突,则称集合 A 中所有节点的数据可以在一个时间片内聚集到集合 B。一个时间片为 L 的数据聚集调度是指一系列的传输调度集 $\{S_1,S_2,\cdots,S_L\}$,并且每个传输调度集 S_i $(i=1,2,\cdots,L)$ 满足以下条件:

a) $S_i\cap S_j=\varnothing$, $\forall i\neq j$;

b) $\cup_{i=1}^{L}S_i=V-\{\text{sink}\}$;

c) S_k 中的节点可以在第 k 个时间片内将数据聚集到 $V-\cup_{i=1}^{k}S_i$,其中 $k=1,2,\cdots,L$。网络中所有节点可以在 L 个时间片内将数据聚集到 sink。

给定一个由若干个传感器和 sink 组成的 WSNs,假设每个传感器都要给 sink 发送数据,MLDAS 问题是指设计一种有效的数据聚集调度方案,使 sink 成功收集所有传感器的数据所花的时间最少,并保证在任何一次数据传输过程中,都不会发生通信冲突。MDAL 问题可形式化描述为:给定一个图 $G=(V,E)$ 和 sink $\in V$,寻找一个数据聚集调度集 $\{S_1,S_2,\cdots,S_L\}$,使得 L 最小。

1.3 相关定义及术语

在图 $G=(V,E)$ 中,对于节点 u 和 v,如果 $(u,v)\in E$,则称节点 u 与 v 相邻;否则,节点 u 与 v 不相邻。如果集合 $U\subseteq V$ 中的任何两个节点不相邻,则称 U 为图 G 的独立集。如果 U 是图 G 的独立集,但是任意增加一个节点就破坏它的独立性,则称 U 为图 G 的最大独立集,简记为 MIS。设 $S\subseteq V$,对于 V 中的任意节点 u,要么 $u\in S$,要么与 S 中的某些节点相邻,则称 S 为图 G 的一个支配集,记为 DS。如果由支配集 S 生成的子图 $G[S]$ 是连通的,则称之为连通支配集,记为 CDS。显然,图 G 的每个 MIS 也是一个 DS。为了方便描述,本文称 DS 中的节点为支配点,或称支配节点;除支配点外,CDS 中其余的节点称为连接点,或称连接节点;网络中其余的节点称为普通节点。

在图 $G=(V,E)$,用 $d_G(u,v)$ 表示图 G 中从 u 到 v 最小跳数。$N_1(u)=\{v\mid(u,v)\in E\}$ 表示节点 u 的邻居,$N_2(u)=\{v\mid d_G(u,v)=2\}$ 表示节点 u 的 2-hop 邻居。$D_2(u)=N_2(u)\cap\text{MIS}$ 表示与 u 相距两跳、且相互独立的支配点集。此外,用 $\text{ann}(u,r_1,r_2,\alpha)$ 表示以节点 u 为圆心,角度为 α,半径在 r_1 和 r_2 之间的环形区域。

2 分布式数据聚集调度近似算法(IDDAS 算法)

这一部分主要描述本文提出的一种改进的无通信冲突的分布式数据聚集调度近似算法,为描述方便,简记为 IDDAS 算法。在 IDDAS 算法中,假设每个节点 u 具有唯一的 ID,并维护以下的数据信息:

a)节点唯一的 ID(u)、层次 level(u)(距 sink 的跳数)、度 degree(u)、父节点 $P(u)$、调度时间 $TS(u)$。

b)节点颜色 color(u)。其中 color(u) = black 表示节点 u 为支配节点,color(u) = blue 表示节点 u 为连接节点,color(u) = white 表示节点 u 为普通节点,color(u) = gray 表示节点 u 不能被设置为支配节点,节点 u 的初始状态为 color(u) = uncolor。

c)节点调度状态 sch(u)。其中 sch(u) = NY 表示节点不能进行数据调度,sch(u) = RY 表示节点准备进行数据调度,sch(u) = done 表示节点已完成数据调度。

d)邻居列表 NList(u),记录每个邻居节点的 ID、层次、度、颜色、调度时间、节点调度状态、孩子列表,等等。

e)接收点集 rev(u),表示接收 u 的数据节点集。

f)孩子列表 $Ch(u)$,孩子个数 $x=|Ch(u)|$。

g)竞争集 $CS(u)$,$CS(u)$ 中每个节点 v 维护两个信息:ID 及接收点集 rev(v)。$CS(u)$ 分为两个不相交的集合 $R_1(u)$ 和 $R_2(u)$,其中 $R_1(u)$ 表示准备调度的节点的集合,$R_2(u)$ 表示已完成调度的节点集合。

IDDAS 算法分为三个阶段:a)构造最大独立集(MIS);b)构造数据聚集树(DAT);c)分布式数据聚集调度。

对于 MIS 的构造,本文首先采用现有的分布式算法[8]构造一棵高度为 R、根在 sink 节点的 BFS 树,从而得到每个节点

的父节点及层次,然后采用现有的分布式算法[10]在整个网络中并行构造 MIS。由于篇幅所限,本文省略该阶段的描述。

2.1　分布式数据聚集树 DAT 的构造

构造最大独立集 MIS 之后,网络中的节点分为两类:一类颜色为 black,也称为支配节点;另一类是颜色为 white 的普通节点。为了描述方便,用 $D_1(u)$ 表示在节点 u 的传输范围内,颜色为 black 的节点,即 u 的 1 跳支配节点的集合;$D_2(u)$ 表示在节点 u 的 2 跳范围内,颜色为 black 的节点,即 u 的 2 跳支配节点的集合,$D_2(u) \cap D_2(v)$ 表示节点 u 和 v 公共邻居支配点集。

与现有算法相比,本文采用自顶向下的方式分层构造数据聚集树,其主要目的是以数量最少的连接节点将相距两跳的支配节点连接起来,形成一棵根在 sink 节点的数据聚集树 DAT。具体来说,在数据聚集树的构造过程中,假设 sink 节点以数量最少的连接节点将相距两跳的支配节点连接起来,形成一棵数据聚集树后,那么对于数据聚集树上某一支配点,只要将与之相距两跳、尚未加入数据聚集树的支配节点加入到数据聚集树。例如,假设有相距两跳的两个支配节点 u 和 v,如果支配节点 u 先将与之相距两跳、尚未加入数据聚集树的支配节点加入到数据聚集树,那么,对于支配节点 v,只要将 $D_2(v) - D_2(u) \cap D_2(v)$ 中的点加入到数据聚集树即可,这是因为 $D_2(u) \cap D_2(v)$ 中的支配点已通过支配节点 u 加入到数据聚集树。因此,数据聚集树的构造开始于 sink,每个节点维护一个布尔变量 flag,表示节点是否已加入聚集树。首先,将 sink 的 flag 设置为 T,其余节点的 flag 设置为 F。然后按以下的原则在整个网络中并行进行:

a)对于颜色为 black 的 u,若其 flag = T,则向它的直接邻居节点发送一个 request 消息,请求得到每个邻居节点的 1 跳支配节点的 ID、层次、是否已加入聚集树等信息。当节点 u 收集所有节点的 ack 消息后,首先,通过计算得到所有未加入聚集树的 2 跳支配节点,简记为 $DS_2(u)$;然后采用 IMC 算法[11]在 NList(u)中寻找一个节点数最小的连接节点集(记为 $C(u)$),将 u 与 $DS_2(u)$ 中的节点连接起来;最后,采用单播的方式向 $C(u)$ 中的各个节点发送一个 info 消息,告诉它及 $DS_2(u)$ 的父节点是谁。

b)对于颜色为 white 的 v,当收到黑色邻居节点 u 的request 消息后,向它发送一个 ack 消息,ack 消息中包含该节点的 1 跳支配节点的 ID、层次、flag 等信息。若它收到黑色邻居节点 u 的 info 消息,则将它的颜色设为 blue、flag 设置为 T、父节点设为 $P(v) = u$。若它有多个黑色邻居节点的 flag = T,则收集所有这些节点的 info 消息后,从中选择层次最低的黑色邻居节点作为它的父节点;如果层次相同,则选择 ID 最小的黑色邻居节点作为它的父节点。最后,根据父节点的 info 消息,向 flag = F 的黑色邻居节点发送一个 parent 消息,通知它们以父节点为 v 加入聚集树。

c)对于颜色为 black 的 u,若 flag = F,当收到邻居节点 v 的 parent 消息后,则将其 flag 设置为 T、父节点设为 $P(u) = v$。若同时收到多个 parent 消息,则从中选择层次最低的邻居节点作为它的父节点;如果层次相同,则选择 ID 最小的邻居节点作为它的父节点。

d)上述过程反复进行,直到所有黑色节点的 flag = T 为止。最后,对于所有颜色为 white、flag = F 的节点,选择层次最低的黑色邻居节点作为它的父节点;如果层次相同,则选择 ID 最小的黑色邻居节点作为它的父节点,并通过所选择的父节点加入聚集树。

2.2　分布式数据聚集调度

数据聚集调度的主要任务是为网络中每个节点分配一个时间片(timeslot),使该节点能在无通信冲突的情况下传输数据。类似于现有的分布式算法,本文也采用竞争集来描述一个节点在传送数据时,可能会有多少个其他节点影响其数据的发送。节点 u 的竞争集的定义[7]如下:

$$CS(u) = N(p(u)) \cup (\cup_{v \in N(u) \setminus Ch(u)} CH(v)) \setminus \{u, p(u)\}$$

从定义中可以看出,节点 u 的竞争集主要包括两类节点:a)节点 u 的父节点的邻居节点,但节点 u 和节点 u 的父节点的父节点除外;b)与节点 u 相邻的其他邻居节点的孩子节点。

与现有算法相比,本文采用一种新的选择标准从竞争集中选择被调度的节点,即根据节点的权值 $sw(u)$ 大小从竞争集中选择被调度的节点。节点的选择权值 $sw(u)$ 用一个三元组表示,即 $sw(u) = (|\text{rev}(u)|, \text{order}(\text{rev}(u)), \text{ID}(u))$。其中 $|\text{rev}(u)|$ 表示接收点集 rev(u)中节点的个数,order(rev(u))表示 rev(u)中节点按其 ID 的降序排列;当 $|\text{rev}(u)| = |\text{rev}(v)|$ 时,order(rev(u))与 order(rev(v))按节点 ID 逐一比较大小。因此,若 $sw(u) > sw(v)$,则必有下列条件之一成立:

a) $|\text{rev}(u)| > |\text{rev}(v)|$;

b) $|\text{rev}(u)| = |\text{rev}(v)|$ 且 order(rev(u)) > order(rev(v));

c) $|\text{rev}(u)| = |\text{rev}(v)|$, order(rev($u$)) = order(rev($v$)), $\text{ID}(u) > \text{ID}(v)$。

引理 1 可说明采用这种选择标准,可使更多的节点能并行传输数据,从而有效降低数据的聚集延时。

引理 1　若一个节点 u 有 $k(\leqslant \Delta)$ 个孩子节点,那么,节点 u 收集所有孩子的数据所需的时间片(timeslot)不超过 Δ,其中 Δ 是网络节点的最大度。

证明　设 $Ch(u)$ 表示节点 u 的孩子节点集,那么,对于任意 $v \in Ch(u)$,当 v 发送数据给 u 时,$Ch(u)$ 中其他节点不能发送数据,且与 v 相邻的其他接收节点都不能同时接收其他发送节点的数据。设 rev(v)表示与节点 v 相邻的接收节点的集合,因此,可用 $RN = \cup_{v \in Ch(u)} \text{rev}(v)$ 表示与节点 u 的孩子节点相邻的所有接收节点的集合。此外,用 $|S|$ 表示节点集 S 中节点的个数。下面用数学归纳法证明引理 1 成立。

a)若 $|RN| = 1$,即 $Ch(u)$ 中的每个节点都只与节点 u 相邻,那么,对于 $Ch(u)$ 中的每个节点分配一个时间片,故共需 $|Ch(u)|$ 个时间片。显然 $|Ch(u)| \leqslant \Delta$,故引理 1 成立。

b)若 $|RN| = 2$,不妨设 $RN = \{u, u_2\}$,即 $Ch(u)$ 中有部分节点同时与节点 u 和 u_2 相邻,那么,将 $Ch(u)$ 的节点分为两个互不相交的子集:$Ch(u) \cap Ch(u_2)$ 和 $Ch(u) - Ch(u) \cap Ch(u_2)$,同时,也将 $Ch(u_2)$ 的节点分为两个互不相交的子集:$Ch(u) \cap Ch(u_2)$ 和 $Ch(u_2) - Ch(u) \cap Ch(u_2)$。对于 $Ch(u) \cap Ch(u_2)$ 中的节点,它们发送数据时,节点 u 和 u_2 都会收到,因此,不管是节点 u,还是节点 u_2,都需要消耗 $|Ch(u) \cap Ch(u_2)|$ 个时间片接收来自 $Ch(u) \cap Ch(u_2)$ 的数据。另一方面,当 $Ch(u) - Ch(u) \cap Ch(u_2)$ 中的任意节点发送数据给节点 u 时,

$Ch(u_2)-Ch(u)\cap Ch(u_2)$中的任意节点也能发送数据给节点$u_2$而不会产生冲突，这是因为对于任意$w\in(Ch(u)-Ch(u)\cap Ch(u_2))$，$w\notin Ch(u_2)$。类似地，对于任意$w\in(Ch(u_2)-Ch(u)\cap Ch(u_2))$，$w\notin Ch(u)$。因此，无论是节点$u$，还是节点$u_2$，收集这些节点的数据所需的时间为：$\max\{|Ch(u)-Ch(u)\cap Ch(u_2)|,|Ch(u_2)-Ch(u)\cap Ch(u_2)|\}$。不妨设$u$收集$Ch(u)\cap Ch(u_2)$中的数据，那么，节点$u$和$u_2$收集其孩子节点的数据所需的时间为

$$\max\{|Ch(u)-Ch(u)\cap Ch(u_2)|+|Ch(u)\cap Ch(u_2)|,\ |Ch(u_2)-Ch(u)\cap Ch(u_2)|+|Ch(u)\cap Ch(u_2)|\}=\max\{|Ch(u)|,|Ch(u_2)|\}\leqslant\Delta$$

故引理1成立。

c)假设$|RN|=k$时，引理1成立。

d)假设$|RN|=k+1$时，不妨设$RN=\{u_1,u_2,u_3,\cdots,u_k,u_{k+1}\}$，其中$u=u_1$。类似于b)，将$Ch(u_{k+1})$的节点分为两个互不相交的子集：$Ch_{comm}=(\cup_{j=1}^{k}Ch(u_i))\cap Ch(u_{k+1})$和$Ch_{rest}(u_{k+1})=Ch(u_{k+1})-Ch_{comm}$，将$\cup_{j=1}^{k}Ch(u_i)$的节点也分为两个互不相交的子集：$Ch_{comm}=(\cup_{j=1}^{k}Ch(u_i))\cap Ch(u_{k+1})$和$\mathrm{Rest}Ch=\cup_{j=1}^{k}Ch(u_i)-Ch_{comm}$。对于$Ch_{comm}$中节点的数据，不管是发送给节点$u_{k+1}$，还是$RN-u_{k+1}$中的某一节点，都需要$|N_{comm}|$个时间片。而对于$x\in Ch_{rest}(u_{k+1})$和$y\in \mathrm{Rest}Ch$，都有$x\notin\cup_{w\in(RN-u_{k+1})}Ch(w)$和$y\notin Ch(u_{k+1})$，因此，$x$和$y$能同时向其相应的接收节点发送数据而不会产生冲突。设Δ_k为收集$\mathrm{Rest}Ch$中节点的数据到相应的接收节点所需的时间，那么，收集$Ch(u)$中节点的数据到相应的接收节点所需的时间为$\max\{\Delta_k+|Ch_{comm}|,|Ch_{comm}|+|Ch_{rest}(u_{k+1})|\}$。根据假设c)，当$|RN|=k$时，$Ch(u)$中的节点最多在时间片$\Delta$内将其数据聚集到相应的接收节点，即$\Delta_k+|N_{comm}|\leqslant\Delta$。因此，聚集$Ch(u)$中节点的数据到相应节点所需的时间为$\max\{\Delta,|N_{comm}|+|N_{rest}(u_{k+1})|\}\leqslant\Delta$，故引理1成立。

从引理1中可以得到，如果让权值$sw(u)$最大的节点先分配timeslots，那么不管节点$CS(u)$的个数有多少，那么，节点u总能在Δ时间片(timeslot)内完成数据调度。基于引理1，设每个节点的调度时间$TS(u)$初值为1，那么，分布式数据聚集调度算法描述如下：

a)采用分布式CRC算法[8]为每个节点构造竞争集。

b)如果节点u的孩子节点数为0或者孩子节点已完成数据调度，则将自己的调度状态设为$\mathrm{sch}(u)=RY$，同时，向它的竞争者发送quest消息。该消息中包括节点的ID、调度状态$\mathrm{sch}(u)$和父节点(即接收节点)等信息，同时，等待它的竞争者回送一个ack消息。

c)节点u收到节点v的quest消息，并将v的ID及$\mathrm{rev}(v)$增加至$R_1(u)$中，并将$R_1(u)$中的节点按选择权值sw进行排序，同时向v发送一个ack消息，包括节点的ID、调度状态$\mathrm{sch}(u)$和父节点(即接收节点)、可能最早的调度时间$TS(u)$等信息。

d)节点u收到节点v的ack消息，如果节点u处于RY状态，则在ack消息中检查节点v的状态，如果v处于RY状态，则将v插入有序集$R_1(u)$中。当节点u收到所有竞争者的ack消息后，若节点u的选择权值$sw(u)$是$R_1(u)$中最大的，则它为自己分配一个timeslots，并向它的竞争者及它的父节点发送一个包含$TS(u)$的Complete消息，然后将自己设为$\mathrm{sch}(u)=$done状态。

e)节点u收到节点v的complete消息后检查自己的调度状态。如果节点u处于RY状态，则将节点v增加到$R_2(u)$中，并从$R_1(u)$中将节点v删除。然后检查节点u的选择权值$sw(u)$是否是$R_1(u)$中最大的，若是，则它为自己分配一个timeslots，并向它的竞争者及它的父节点发送一个包含$TS(u)$的complete消息，然后将自己设为$\mathrm{sch}(u)=$done状态。如果节点u处于NY状态，则分几种情况进行处理。如果节点v在$CS(u)$中，则将节点v增加到$R_2(u)$中，并从$R_1(u)$中将节点v删除。如果节点u是节点v的父节点，则修改$TS(u)=\max(TS(v)+1,TS(u))$，并将$x$的值减1，如果$x=0$，则将它的状态设为$RY$状态，同时检查$R_1(u)$。若$R_1(u)$为空，则它为自己分配一个timeslots，并向它的竞争者及它的父节点发送一个包含$TS(u)$的complete消息，然后将自己设为$\mathrm{sch}(u)=$done状态；否则，向它的竞争者发送quest消息，并等待它的竞争者回送一个ack消息。

2.3 性能分析

为了方便性能分析，假设所有节点具有相同的传输范围，并标准化为一个单位，即传输半径$r=1$。这样，基本的通信图就可简化为单位圆盘图(unit disk graph，UDG)。下面以UDG为模型，分析算法在最坏情况下的数据延时。首先引入三个已经证明了的结果。

引理2[6,12] 假设支配点v和w是支配点u的2跳邻居，其相应的连接点分别为x和y，如果$\angle vuw\leqslant 2\arcsin(1/4)\approx 28.955°$，那么$|vy|\leqslant 1$或者$|wx|\leqslant 1$。

引理3[6] 一个支配点u最多需要12个连接点在两跳之内连接其相邻支配集$D_2(u)$中的节点。

引理4[12] 设S为支配点u的环形区域$\mathrm{ann}(u,1,2,180°)$内支配点的集合，那么支配点u最多需要7个连接点在两跳之内连接S中的节点。

根据引理2~4，可得到如下的推论，其证明过程可参考引理3和引理4，本文略。

推论1 设S为支配点u的环形区域$\mathrm{ann}(u,1,2,240°)$内支配点的集合，那么支配点u最多需要9个连接点在两跳之内连接S中的节点。

引理5 IDDAS算法的时延上界是$14R+\Delta$，其中R为BFS树的高度，Δ为网络的最大度。

证明 分两种情况讨论算法的时延上界。

设m是偶数，所有数据在t_m时间内汇聚到DAT树上的第m层节点，在t_{m-1}时间内所有数据聚集到DAT树上的第$m-1$层节点。注意到DAT树上偶数层节点是黑色节点，分两种情况讨论t_{m-1}。第一种是在t_m时间内，所有白色节点的数据已经发送给第$m-2$层的黑色节点；第二种情况是在t_m时间内，还有部分白色节点没有将数据发送给第$m-2$层的黑色节点。对于第一种情况，仅考虑第m层黑色节点的数据调度。对于第$m-1$层蓝色节点的最大度不超过5，这是因为半径为1的单位圆内，最多只有5个黑色节点。因此，最多在$t_{m-1}\leqslant t_m+5$的时间内，第m层黑色节点完成数据调度。对于第二种情况，根据引理1可得，最多在$t_{m-1}\leqslant t_m+\Delta$的时间内，第$m$层黑色节点完成数据调度。综合可得，最多在$t_{m-1}\leqslant 5+\max(t_m,\Delta)$的时间内，第$m$层黑色节点完成数据调度。

设m是奇数，所有数据在t_m时间内汇聚到DAT树上的第

m 层节点,在 t_{m-1} 时间内所有数据聚集到 DAT 树上的第 $m-1$ 层节点。注意到 DAT 树上奇数层节点是白色节点和蓝色节点,分两种情况讨论 t_{m-1}。第一种是在 t_m 时间内,所有白色节点的数据已经发送给第 $m-1$ 层的黑色节点;第二种情况是在 t_m 时间内,还有部分白色节点没有将数据发送给第 $m-1$ 层的黑色节点。对于第一种情况,仅考虑第 m 层蓝色节点的数据调度。当 $m=1$ 时,根据引理 3,第 0 层的黑色节点最多有 12 个第 m 层蓝色节点与之相连,因此,$t_0 \leqslant t_1 + 12$。而当 $m>1$ 时,根据推论 1,对于第 $m-1$ 层的黑色节点,最多有 10 个第 m 层蓝色节点与之相连,但其中有一个节点是它的父节点。因此,根据推论 1,最多在 $t_{m-1} \leqslant t_m + 9$ 的时间内,第 m 层蓝色节点完成数据调度。对于第二种情况,根据引理 1 可得,最多在 $t_{m-1} \leqslant t_m + \Delta$ 的时间内,第 m 层黑色节点完成数据调度。综合可得,当 $m>1$ 时,最多在 $t_{m-1} \leqslant 9 + \max(t_m, \Delta)$ 的时间内,第 m 层黑色节点完成数据调度。

从以上分析可得,IDDAS 算法产生的时延为

$$T = t_0 \leqslant t_1 + 12 \leqslant 12 + 5 + \max(t_2, \Delta) \leqslant \cdots \leqslant \underbrace{(12+5) + (9+5) + \cdots + (9+5)}_{k-1} + \max(t_{2k-1}, \Delta) \leqslant \underbrace{(12+5) + (9+5) + \cdots + (9+5)}_{k-1} + 9 + \max(t_{2k}, \Delta) \leqslant 14R + \Delta$$

2.4　仿真模拟

本文采用 MATLAB 2010 仿真软件,从网络节点数和节点的度两个方面对 IDDAS、DDAS[8] 和 FDAP 算法[9] 产生的聚集延时进行比较分析。网络仿真的主要参数如下:

a)仿真区域。所有传感器节点(包括 sink 节点)随机地分布在一个 500 m×500 m 的正方形区域内。

b)传感器节点。每个传感器节点在全网中具有唯一的 ID 号,所有传感器节点的传输半径和干扰半径均为 25 m。

c)网络节点数 N。指整个网络中节点的个数。

d)聚集延时。指 sink 节点收集所有节点的数据所需的时间,单位为 timeslot(时间片)。对于相同参数的网络,随机产生 50 个不同的网络拓扑图,聚集延时为 50 个拓扑图中各种算法产生的聚集延时的平均值。

首先考察当网络的度固定为 15,网络节点数 N 变化时对各种算法产生的聚集延时的影响。仿真结果如图 1 所示,其中横坐标表示网络节点数 N,纵坐标表示各种算法的平均聚集延时。当网络的度固定,网络节点数 N 增加,数据聚集树的高度也随之增加,因而算法产生的聚集延时也会相应增加。图 1 的仿真结果正好反映了这种变化,但随着网络节点数 N 增加,IDDAS 算法产生的聚集延时要低于 DDAS 和 FDAP 算法。

其次考察当网络节点数 N 固定为 500,网络的度发生变化时对各种算法所产生的数据聚集延时的影响。仿真结果如图 2 所示,其中横坐标表示网络的度,纵坐标表示各种算法的平均聚集延时。当网络节点数 N 固定,网络的度增加,数据聚集树的高度会降低,算法产生的聚集延时也会相应减少。但随着网络的度进一步增加,尽管数据聚集树的高度不会发生很大的变化,但算法产生的聚集延时却开始增加,这是因为它受网络的度的影响。图 2 的仿真结果正好反映了这种变化,但总的来说,IDDAS 算法产生的聚集延时要低于 DDAS 和 FDAP 算法。

3　结束语

针对 MLDAS 问题,本文提出了一种改进的分布式数据聚集调度算法。该算法首先在最大独立集的基础上建立一棵根在 sink 的数据聚集树,然后各个节点按数据聚集树分层进行数据调度。与现有算法相比,该算法的不同之处在于:a)在数据聚集树的构造过程中,对于数据聚集树上某一支配点,只要将与之相距两跳、尚未加入数据聚集树的支配节点加入到数据聚集树;b)在数据调度过程中,采用一种新的选择标准从竞争集中选择节点进行数据调度。理论分析表明,该算法的延时上界为 $14R+\Delta$;仿真模拟的结果表明,该算法产生的数据聚集延时远低于现有算法。今后将在此基础上,设计一种延时更小、更能应用于实际环境的分布式数据聚集调度算法。

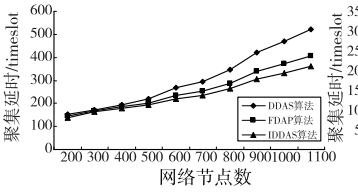

图1　节点的度固定,网络节点数变化时的延时曲线

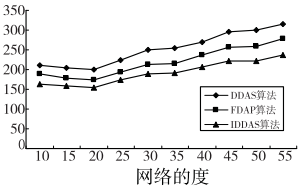

图2　节点数固定,网络的度变化时的延时曲线

参考文献:

[1] CHEN Xu-jin, HU Xiao-dong, ZHU Jian-ming. Minimum data aggregation time problem in wireless sensor networks [C]//Lecture Notes in Computer Sciences, vol 3974. 2005:133-142.

[2] HUANG S C H, WAN Peng-jun, VU C T, *et al*. Nearly constant approximation for data aggregation scheduling in wireless sensor networks [C]//Proc of IEEE INFOCOM. 2007:366-372.

[3] WAN Peng-jun, HUANG S C H, WANG Li-xin, *et al*. Minimum-latency aggregation scheduling in multihop wireless networks[C]//Proc of ACM MobiHoc. New York: ACM Press, 2009:185-193.

[4] REN Mei-rui, GUO Long-jiang, LI Jin-bao. A new scheduling algorithm for reducing data aggregation latency in wireless sensor networks [J]. International Journal of Communications, Network and System Sciences, 2010, 3(8):679-688.

[5] 郭龙江,任美睿,李金宝,等. 降低传感器网络数据聚集延迟的近似调度算法[J]. 黑龙江大学工程学报, 2011, 2(2):80-94.

[6] XU Xiao-hua, LI Xiang-yang, MAO Xu-fei, *et al*. A delay-efficient algorithm for data aggregation in multihop wireless sensor networks[J]. IEEE Trans on Parallel and Distributed Systems, 2011, 22(1): 163-175.

[7] YU Bo, LI Jian-zhong, LI Ying-shu. Distributed data aggregation scheduling in wireless sensor networks [C]//Proc of IEEE INFOCOM. 2009:2159-2167.

[8] LI De-ying, ZHU Qing-hua, DU Hong-wei, *et al*. An improved distributed data aggregation scheduling in wireless sensor networks [J]. Journal of Combinatorial Optimization, 2012(5):1-20.

[9] BOULKABOUL S, DJENOURI D, BADACHE N. FDAP: fast data aggregation protocol in wireless sensor networks[C]//Lecture Notes in Computer Science, vol 7469. 2012:413-423.

[10] WAN Peng-jun, ALZOUBI K M, FRIEDER O. Distributed construction of connected dominating set in wireless Ad hoc networks[J]. Mobile Networks and Applications, 2004, 9(2):141-149.

[11] HUANG S C H, WAN Peng-jun, JIA Xiao-hua, *et al*. Minimum-latency broadcast scheduling in wireless Ad hoc networks [C]//Proc of IEEE INFOCOM. 2007:733-739.

[12] WAN Peng-jun, YI Chih-wei, JIA Xiao-hua, *et al*. Approximation algorithms for conflict-free channel assignment in wireless Ad hoc networks[J]. Wireless Communications Mobile Computing, 2006, 6(2):201-211.