文章编号: 1001- 2486(2009) 01- 0064- 06

大规模无线传感器网络中面向 ANY 型查询的能量高效数据分发算法*

陶孜谨, 郦苏 丹, 徐金义, 龚正虎
(国防科技大学 计算机学院, 湖南 长沙 410073)

摘 要: 在事件数据的 push 和 pull 之间实现更好的平衡是无线传感器网络数据分发算法节能的关键。分析了两种典型的有结构和无结构的数据分发算法, 结合这两种算法使用的 push-pull 策略, 针对无线传感器网络的 ANY 型查询的特定需求, 提出了两种基于有结构和无结构存储模式相结合的混合型数据分发算法 SDC1&2。分析表明, 这两种算法在保证 push-pull 之间平衡的前提下解决了已有算法存在的热点问题、存储拷贝数多和查询性能低问题, 能更好地适应 ANY 型查询的特点, 是两种能量高效的数据分发算法。

关键词: 数据分发; push-pull 平衡; ANY 型查询; 通信代价; 负载均衡; 无线传感器网络
中图分类号:TP393 文献标识码: A

Energy-efficient Data Dissemination Algorithms for ANY-type Queries in Large-scale WSN

TAO Zi-jin, LI Su-dan, XU Jin-yi, GONG Zheng-hu
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: Striking a better balance between the push and pull of the event data is the key factor to the energy saving for the data dissemination algorithms in wireless sensor networks(WSNs). Two typical structured and unstructured data dissemination algorithms (DCS and CN) are analyzed first. By incorporating the push-pull strategies of the two algorithms, two new algorithms(SDC1 and SDC2) are proposed for the ANY-type queries in the different application situation in WSNs. They have resolved the problems of the high load of the hotspot, the large number of the event data replicas and the low energy efficiency of queries on the premise that it may ensure the balance between the push and pull. Results from the analysis indicate that they are much more appropriate for the WSNs which are large-scale and have large volume of data to be dealt with, and they are two energy-efficient data dissemination algorithms.

Key words: data dissemination; push-pull balance; ANY-type query; communication cost; load balance; WSN

目前传感器数据分发算法绝大多数都针对 ALL 型查询, 即 sink 节点需要所有匹配查询请求的事件数据。而 ANY 型查询是一类特殊的查询, sink 节点只需要一个想要的数据, 当查询请求搜索到第一个匹配数据后就取消后续的搜索。ANY 型查询有很大的现实意义。研究人员提出了采用网络存储和命名查询的以数据为中心的技术, 提出了许多相关算法: Directed Diffusion(DD)[1], TAG/TinyDB[2]、流言路由[3], 基于 Hash 的以数据为中心的存储[4], 混合 push-pull[5], 梳子—针(Comb-needles)[6], 两层数据分发模型(TTDD)[7] 等。文献[8] 比较了 3 种随机走查询策略的渐进性能, 显示了 push-pull 汇聚搜索有最佳的成功概率。

GHT- DCS[4] 算法存在的问题: (1) 存在热点。GHT 算法对于同一事件类型的数据都存储在同一节点上, 即家乡节点, 而且对此类型数据的查询也集中在家乡节点, 因此家乡节点成为通信和存储的双重热点。(2) 存储代价高。不管数据在哪儿产生, 侦测到的数据存储在网络中固定的一个点上。

CN(SCN) 算法存在的问题: (1) 查询搜索范围大。由于 CN 算法的存储策略类似本地存储, 与 DCS

* 收稿日期:2008- 04- 02
基金项目:国家部委基金资助项目
作者简介:陶孜谨(1973—), 男, 博士生。

算法相比, 这种分布式的存储必然导致在查询时的高代价, sink 节点必须搜索大量节点才能获取全部数据, 这将导致较长的查询响应延迟。(2) 广播低效问题。CN 算法的 Comb 过程是一个区域搜索过程, 而 Needle 过程也是一个区域覆盖过程, 所以 CN 算法引入了一个受限地理泛洪(Constrained Geographic Flooding, CGF) 把查询和存储信息发送到指定区域的每个节点, 其工作模式就类似于全局泛洪。

文献[9] 指出, 当某个时段内事件的平均数较多时, SCN 算法优于 GHT-DCS 算法; 当某个时段内查询概率较大时, DCS 算法优于 SCN。因此在这两个算法之间应该存在一个折中的方式来结合它们各自的优点。本文提出的两个算法 SDC1 和 SDC2(Sequential-DCS-CN) 采用了这种思想。

1 SDC1 算法

1.1 算法描述

SDC1 算法是 GHT-DCS 算法与 SCN 算法的结合, 节点的存储机制采用局部化的 DCS 机制进行, 而查询机制则涉及到 SCN 算法和 DCS 算法的共同采用。查询时, 处于网络左下角的 sink 节点发起查询, 查询沿着网格的边界遍历整个网络, 到达各网格的左下角节点(查询起始点), 然后在各网格内使用 Hash 机制把对应事件的查询发送到事件的存储点。查询响应首先返回到各网格的查询起始点, 然后返回 sink 节点。图 1、图 2 分别给出了 SDC1、SDC2 的工作示意图。

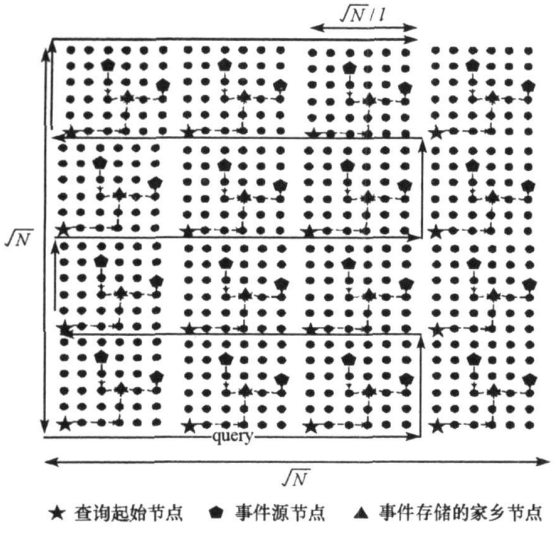

图 1 SDC1 算法工作示意图

Fig. 1 The sketch map of working process of SDC1

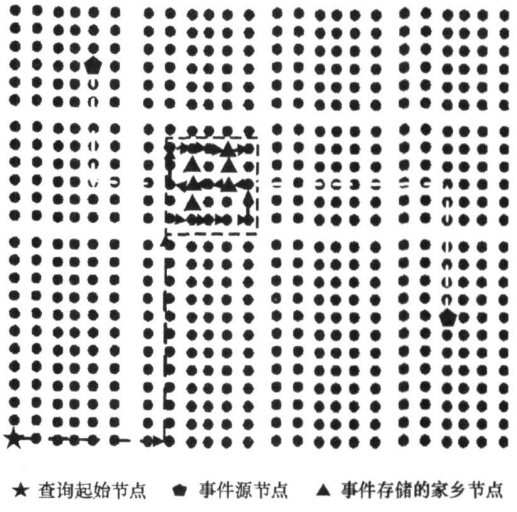

图 2 SDC2 的工作示意图

Fig. 2 The sketch map of working process of SDC2

事件缓存算法: 节点 n 使用地理路由算法把事件 m 消息打包发送到 S_m, 当接收到消息 m 的节点的坐标等于报文的目的地址或者消息处于 GPSR 周界模式而且重复访问当前边(通过右手规则计算下一跳节点, 如果当前节点等于进入周界模式的起点, 而且下一跳节点等于进入周界模式的第二个节点), 说明消息已发送到离子目标节点 $h_{i,j}^m = gHash(k_m, x, y, l)$ 最近的节点, 则存储事件 m。

查询转发算法: 查询算法根据网格划分信息计算网格的边界, 获得每个网格的子 sink 节点的坐标。查询从网络的左下角 sink 节点 $s_{0,0}$开始, 沿着 y 方向的网格边界转发, 顺序到达各网格的子 sink 节点 $s_{0,j}(0 \leqslant j < l)$, 子 sink 节点把查询发送到事件的存储位置 $S_m = h_{i,j}^m = gHash(k_m, x, y, l)$, 以及同层的下一个子 sink 节点 $s_{i+1,j}$。

当接收到消息 m 的节点的坐标等于报文的目的地址或者消息处于 GPSR 周界模式而且重复访问当前边, 说明查询已发送到离子目标节点 $h_{i,j}^m = gHash(k_m, x, y, l)$ 最近的节点, 则查询当前节点数据库, 如果有匹配查询属性的事件数据, 则把该数据返回 $s_{0,0}$。$gHash()$ 算法的工作过程如图 3 所示。

$gHash(k_m, x, y, l)$

Input: 给定事件 m 的关键字 k_m, x, y 分别为事件感知节点的横坐标、纵坐标, l 为网络边界区块划分数;

Output: 事件 m 的子 home 节点 $h_{i,j}$ 的坐标($h_{i,j}.x$, $h_{i,j}.y$);

1. $d_{lx}=(b_1-b_0)/l$, $d_{ly}=(b_3-b_2)/l$; // 计算区块大小
2. $i=\blacktriangleright(x-b_0)/d_{lx}\blacktriangleleft$, $j=\blacktriangleright(y-b_2)/d_{ly}\blacktriangleleft$; // 计算事件感知点所在区块
3. $xb=b_0+i*d_{lx}$, $yb=b_2+j*d_{jy}$; // 计算区块边界
4. $hash\_value=MD5(k_m)$; // 通过 MD5 进行Hash,得到16字节 MD5 值
5. $h_{i,j}.x=xb+hash\_value.upper \bmod d_{lx}$ // 用 MD5 Hash 值前8字节得到 x 坐标
6. $h_{i,j}.y=yb+hash\_value.lower \bmod d_{jy}$ // 用 MD5 Hash 值后8字节得到 y 坐标

图3 网络区块地理哈希函数的实现

Fig. 3 The implementation of the geographical Hash function in a block

1.2 算法性能分析与比较

考虑$\sqrt{N}\times\sqrt{N}$的网络, 共有 N 个节点。每个节点有4个邻居节点。节点之间的距离采用Manhattan距离。查询只在网络左下角的sink节点发生, 这是网络与外部的唯一接口。分析的目标是在每个时间段 T 内有最低的总能量代价。E 表示在时间T 内发生的事件的平均数。使用 Q 表示在时间 T 内期待的查询数目。由于时间信息在一个时段 T 内不改变, 所以 Q 总是在0和1之间, 表示在该时段内发出一个查询的概率。

假定SDC1算法把整个传感器网络区域的每边分为 l 等份, 则每个采用DCS算法的区域的边长为 $\sqrt{N}/l$。算法实际由两部分组成, 在各网格内的部分使用DCS算法, 根据文献[4], 由于在每个区块, 感知的事件数目相同, 因此如果把网络分割为 l^2 块, 那么每块边的长度为 $\sqrt{N}/l$, 感知到的事件数目为 E/l^2, 那么对于每一块, 是一个完整的DCS过程:

$$C_{Sl-DCS}=\begin{cases}\dfrac{\sqrt{N}}{l}\left(2Q'-\dfrac{2Q'^2}{\dfrac{E}{l^2}}+\dfrac{\dfrac{E}{l^2}}{2}\right), & Q'<\dfrac{\dfrac{E}{l^2}}{2}\\ \dfrac{\sqrt{N}}{l}\dfrac{E}{l^2}, & \text{其他}\end{cases}\tag{1}$$

SDC1算法的总代价为

$$C_{SDC1}=l^2C_{Sl-DCS}+C_{Sl-CN}\tag{2}$$

考虑(1)式的上半部分, 对于 l^2 个区块, 相当于由网格形成的一个链, 根据文献[4]中的分析, 该链的期望值为 $E[x]=\dfrac{l^2}{E+1}$, 因此SDC1的总代价为

$$C_{Sl-SCN}=C_{ql}+C_{qr}=E[x]\frac{\sqrt{N}}{l}Q'+E[x]\frac{\sqrt{N}}{l}Q'=2\frac{l^2}{E+1}\frac{\sqrt{N}}{l}=2\frac{\sqrt{N}l}{E+1}\tag{3}$$

为了获得(3)式的最小值, 对(3)式中的 l 求导, 并置为0, 得到 l 的最佳值

$$l=\sqrt{\frac{E(E+1)}{6Q}}\tag{4}$$

因为 l 必须为整数, 所以

$$l^*=\left\lceil\sqrt{\frac{E(E+1)}{6Q}}\right\rfloor\tag{5}$$

对于(1)式的下半部分, 有

$$C_{SDC1}=l^2C_{Sl-DCS}+C_{Sl-CN}=l^2\frac{\sqrt{N}}{l}\frac{E}{l^2}+\frac{2\sqrt{N}lQ}{E+1}=\sqrt{N}E\frac{1}{l}+\frac{2\sqrt{N}lQ}{E+1}\tag{6}$$

得到 l 的最佳值

$$l=\sqrt{\frac{E(E+1)}{2Q}} \tag{7}$$

因为 l 必须为整数, 所以

$$l^{*}=\lceil\sqrt{\frac{E(E+1)}{2Q}}\rceil \tag{8}$$

假定节点数 $N=10\,000$, 图 4 显示出 l 的变化, 图 5 显示 3 个算法的总代价, 图4、图5 只显示了上半部分的性能, 下半部分的性能与之类似, 限于篇幅略去。从图 5 可以看出, 3 种算法中 SDC1 的总能量代价最低, 并且与 SCN 算法接近, 因为 SDC1 把数据存储在网格内部的某个点, 然后通过与 SCN 相同的方式获取。当事件数目增加时, SDC1 的总代价缓慢降低, 与 SCN 类似(但略好于 SCN)。但 SDC1 在网络中只存储了一份数据拷贝, SCN 算法存储了多份, 从这一点看 SDC1 所占系统资源大大优于 SCN 算法。

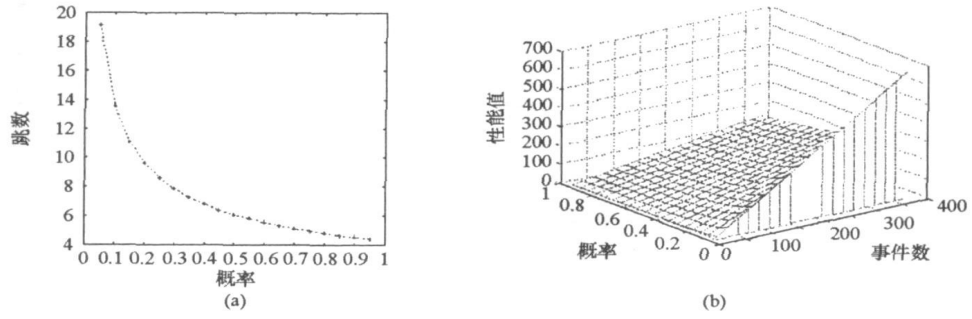

图 4 (a) l 随参数 Q($E=50$) 和(b) l 随参数 Q(从 0~0.95) 和 E(从 10~390) 变化关系图

Fig. 4 (a) l varies along the variation of the value of Q($E=50$) and (b) l varies along with Q (from 0 to 0.95) and E (from 10 to 390)

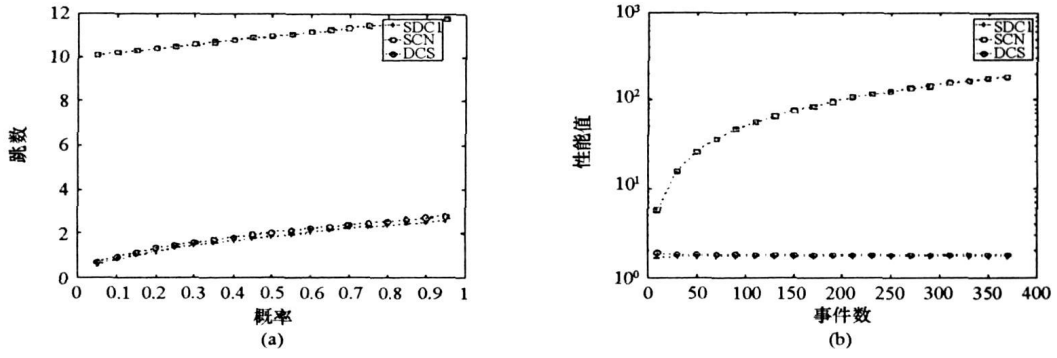

图 5 (a) 3 种算法在不同查询概率情况下的性能($E=20$) 以及(b) 3 种算法在不同事件数时的性能($Q=0.4$)(由于 DCS 有较大的能量代价值, 在图(b) 中 Y 轴以对数方式显示)

Fig. 5 (a) The performance of the three algorithms at different query probabilities ($E=20$) and (b) the performance of the three algorithms at different number of events ($Q=0.4$) (For DCS has a relatively large energy cost value, the Y-axis is shown in log-scale in (b))

3 种算法中, 对于 ANY 型查询, DCS 算法性能最低, 因为不管事件多少和网络规模, 都把所有的同一类型事件发送到同一个节点。如果考虑到以下度量值, SDC1 比 DCS 和 SCN 算法的优势就更突出。从表 1 中可以看出, SDC1 以最少的事件代价数目和更少的节点搜索范围取得了最佳的性能, 并且存储和通信流量为 DCS 的 $1/l^{2}$, 因此解决了 DCS 和 SCN 算法存在的主要问题。

表 1 四个算法的其他重要度量值

Tab. 1 Other important metrics of the four algorithms

度量值	事件存储拷贝数(S_d)	查询搜索节点数目(W_s)	热点存储量(S_h)	热点通信量(H_c)
DCS	E	1	E	$E+QE$
SCN	$S\times E=\sqrt{\frac{2NQE}{E+1}}$	$\mathrm{E}[x]+\mathrm{E}[y]=\frac{\sqrt{NE}}{\sqrt{2Q(E+1)}}+\frac{\sqrt{N}}{E+1}$	$S^*\times\frac{E}{N}=\sqrt{\frac{2NQ}{E(E+1)}}\frac{E}{N}=\sqrt{\frac{2QE}{(E+1)N}}$	$\sqrt{\frac{QE}{N}}+Q\sqrt{\frac{QE}{N}}$
SDC1	E	l^2	$\frac{E}{l^2}$	$\frac{E}{l^2}+Q\frac{E}{l^2}$
SDC2	E	l^2	$\sqrt{\frac{2EQ}{(E+1)l^2}}$	$\sqrt{\frac{2EQ}{(E+1)l^2}}+Q\sqrt{\frac{2EQ}{(E+1)l^2}}$

2 SDC2 算法

SDC1 算法对于事件分布均衡的网络取得了良好的综合性能。但是,如果事件产生的数据量分布不均衡, home 节点存储和查询的负载仍然很重。为了适应不均衡网络,本文提出另一种无结构和有结构混合型数据分发算法。该算法把数据存储在一片节点分布相对规则的区域中,通常是 home 节点周围的某个区域,称之为 home 区域。对于某个事件数据,具体存放的位置使用随机机制存放在 home 区域内的某些节点上,这些节点组成类似 CN 算法中 needle。当需要查询这些数据时,首先把查询发送到该区域左下角的节点,然后使用类似 SCN 算法中的 Comb 机制,对检索到的所有数据返回查询响应。

3 模拟实验结果

在 ns2 模拟器上对 4 种算法进行了对比实验,采用了 ns2 提供的 Monarch 无线通信模块。路由算法采用 GPSR 地理路由算法,节点数为 400,节点的通信距离为 50m。实验中假定查询报文和事件报文的大小都为 256 字节,所以对于各种数据分发算法的总代价值用路由总跳数来衡量。

对于网格型的节点分布图,分为两种情况,一种在保证节点联通性的前提下最大限度地扩展节点的分布,节点之间的距离略小于节点的最大通信距离,每个节点最多只有固定的 4 个邻居,记为 MaxDisGridG;另一种则考虑实际的节点覆盖,有文献指出节点的通信距离为节点距离的 2 倍时可以保证网络的可靠性和冗余性,因此节点间的距离为 23m,记为 DoubleDisGridG。对于随机图,设置网络区域为 400m× 400m 的矩形区域,标识为 RandomG。对于 SDC2,使用了 GPSR 和泛洪(Flooding)路由,GPSR 用在 sink 和 home 区域之间的通信,而泛洪用在 home 区域内的查询,由于 home 区域比较小,泛洪路由效率高且通信代价不会太高。查询响应使用 GPSR 直接发送给 sink。生存时间可以表明算法的能量均衡性和可用性,本文中假定发送一次报文消耗 5 个单位的能量,接收一个报文消耗 1 个单位能量,每个节点初始时能量为 500 个单位。生存时间采用查询次数表示,一次查询返回所有在这个时段内的事件。

由图 6~ 9 可知,如果节点密度较低时(对于 MaxDisGridG),在事件数较多时(查询概率低)不同算法的性能差异较大,DCS 和 SDC2 算法的开销略大于 SDC1, SCN 算法有最低的通信代价,而且随着事件数目的增加,SCN 算法的总代价基本保持稳定,其原因是 SCN 算法及时根据 push- pull 平衡的原理及时调整了存储方式,使得存储的代价增加不多,总代价保持在较好水平。在事件数增加(意味着查询概率降低)的情况下,SDC1 算法的总代价增加也不多,体现了事件局部化存储的优势。

对于节点密度较高的 DoubleDisG 图,与 MaxDisGridG 中的结果略有不同。SDC1 和 SCN 算法在大多数情况下都是最佳算法,DCS 和 SDC2 算法在查询概率很高时有优势。总的说来,当查询概率较低(< 1/16)时,SCN 和 SDC1 的性能较好;否则,DCS 和 SDC2 性能较好,这与前面的理论分析是一致的。对于随机图,其结果与理论值有较大的差异。4 种算法中 SCN 算法的性能最低,而且查询成功率也低于其他算法。其原因是 SCN 算法使用约束地理泛洪,使得许多不相关的节点参与数据传输。除此之外,随机图的节点分布不规则,CGF 把传输约束在一个固定的范围内,为了增加查询成功率,就不得不增加查询宽度,进一步增加了通信代价。

在随机图中,当查询概率较高时,DCS、SDC1 和 SDC2 获得了很好的性能,说明把网络分割为区块后,可以大大减少存储代价,并且对于不同的事件数和查询概率,有一个最合适的分割方式。在不牺牲

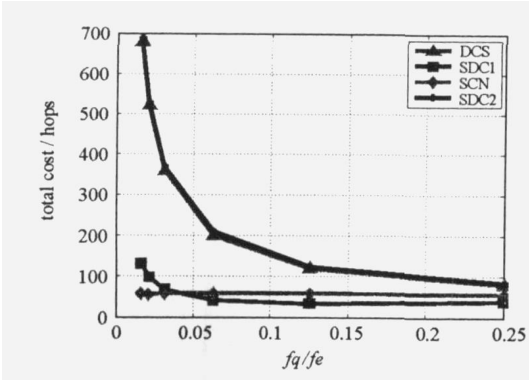

图 6 当查询数为 1 时, 4 种算法在 MaxDisGridG 图中的总通信代价

Fig. 6 When the number of query is 1, the average total communication cost of the four algorithms in MaxDisGridG

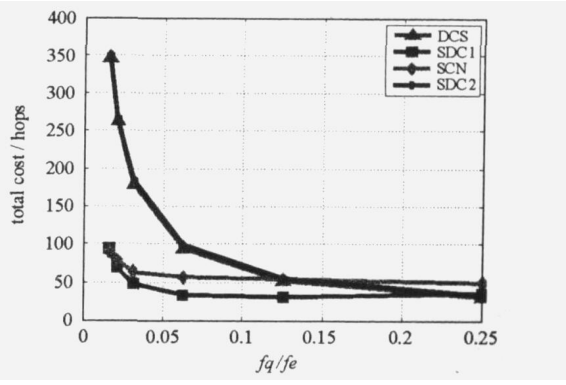

图 7 在时间段 T 内查询数目为 1 时, 4 种算法在 DoubleDisGridG 图中的总通信代价

Fig. 7 When the number of query is 1, the total communication cost of the four algorithms in DoubleDisGridG

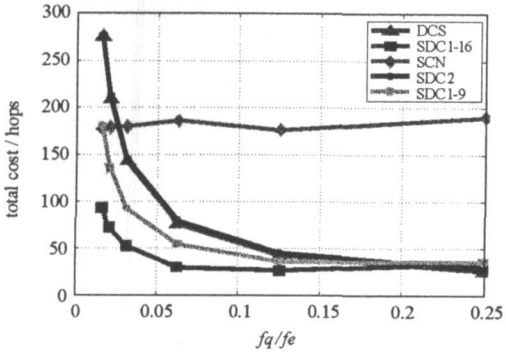

图 8 查询数目为 1 时, 4 种算法在 RandomG 图中的平均总通信代价

Fig. 8 When the number of query is 1, the average total communication cost of the four algorithms

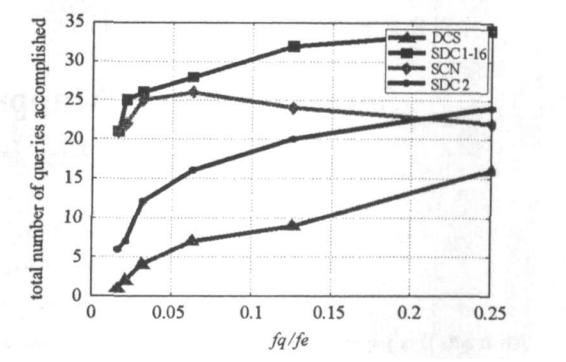

图 9 各算法在生存时间内完成的查询量

Fig. 9 The query number finished within the network lifetime by each algorithm

系统性能和增加数据拷贝数量的前提下, SDC1 和 SDC2 的热点负载大大减小, 这些优点正是无线传感器网络数据分发时最需要的。

4 总结及下一步工作

分析了有结构的 DCS 算法和无结构的 CN 算法各自的优缺点, 通过组合它们的工作模式提出了两种混合型数据分发算法 SDC1 和 SDC2。理论分析和模拟实验表明, 在不增加数据存储拷贝数的情况下, 两个算法的性能都优于已有算法, 解决了这两种算法存在的热点问题, 取得了更好的 push 和 pull 之间的平衡, 延长了系统的生存时间, 是两种能量高效的数据分发算法。下一步的工作是考察算法在更多节点、不同节点分布和事件分布情况下的性能, 并根据无线能量模型计算实际的能耗, 为 SDC2 算法选择一个合适的存储区域, 根据实际的节点分布划分网格, 选择合适的存储点。

参 考 文 献:

[1] Heinzelman W, Chandrakasan A, Balakrishnan H. Energy-efficient Communication Protocol for Wireless Microsensor Networks [C]// Proc. of the 33rd Annual Hawaii Int' l Conf. on System Sciences, Maui: IEEE Computer Society, 2000: 3005- 3014.

[2] Madden S, Franklin M J, Hellerstein J M, et al. TAG: A Tiny Aggregation Service for Ad-hoc Sensor Networks [C]// OSDI' 02: Proceedings of the 5th Symposium on Operating Systems Design and Implementation, ACM New York, NY, USA, 2002: 131- 146.

[3] Estrin D, Braginsky D. Rumor Routing Algorithm for Sensor Networks[C]// Raghavendra C S, ed. . Proc. of the 1st Workshop on Sensor Networks and Applications, New York, 2002: 1- 12.

[4] Ratnasamy S, Shenker B, et al. Data-centric Storage in Sensornets[J]. ACM SIGCOMM, Computer Communications Review, 2003, 33(1).

[5] Yao Y, Trigoni N, Demers A, et al. Hybrid Push-pull Query Processing for Sensor Networks [C]// Proceedings of the Workshop on Sensor Networks as Part of the GI-conference Informatik 2004, Berlin, Germany, 2004.

[6] Huang Q, Liu X, Zhang Y. Combs, Needles, Haystacks: Balancing Push and Pull for Discovery in Large-scale Sensor Networks [J]. ACM Sensys, November 2004.

[7] Luo H Y, Ye F, Cheng J, et al. TTDD: A Two-tier Data Dissemination Model for Large scale Wireless Sensor Networks [C]// Proceedings of the 8th Annual International Conference on Mobile Computing and Networking, New York: ACM Press, 2002: 148- 159.

[8] Shakkottai S. Asymptotics of Query Strategies Over a Sensor Network[J]. IEEE Trans. on Automatic Control, 2005, 50(5): 594- 606.