目录

1 绪论

随着物联网的不断发展，物联网节点数量开始以指数级的趋势增长，因此，对于物联网设备的通信网络协议设计与接入网络部署问题成了重中之重。在这里，尤为困难的是对传感器网络的通信协议与部署进行设计。当今要求传感器网络具有无线传输、自组织、低成本等特性，并且由于传感器节点普遍具有计算力低下、能量受限、不易更换充能等特性，导致无法使用现有的计算机网络路由协议。

自 LEACH 等协议出现，“聚簇”手段开始普遍应用于无线传感器网络（WSN）。聚簇协议将网络分为簇头节点（CHs）与簇内节点（Members），Members 和 CHs 组成一个个簇，簇头可以通过 TDMA（时分复用）等手段收集 Members 的信息，这一举动一是避免了大量碰撞导致的额外开销，二是十分适用于数据融合，并且对于网络的节点要求较低，故该思想很快被采纳。但是聚簇并不意味着一劳永逸，网络中的 CHs 要承担额外开销，故如何选出最合适的簇成为了热点问题。

本文将针对无线传感器网络，对网络能耗进行系统建模，最终设计一种基于 PSO 算法的网络节能聚簇协议，以在有限的开销下得出最优的聚簇方式，同时使用相同协议得出最优接入控制节点的部署方法，从而解决 WSN 的网络协议设计和接入网络部署问题。

1.1 研究背景

物联网是指通过各类信息传感器、射频识别基础（RFID）、全球定位系统（GPS）、红外感应器、激光扫描器等各种装置与技术，实时采集任何需要监控、连接、互动的物体或过程，采集其声、光、热、电、力学、化学、生物、位置等各种需要的信息，通过各类可能的网络接入，实现物与物、物与人的泛在连接，实现对物品和过程的智能化感知、识别和管理。相比较于
机器对机器(M2M)通信技术为设备提供了通过有线和无线系统相互通信的能力。采用数据聚合是降低 M2M 网络能耗的有效途径。

1.2 国内外研究现状

.

1.2.1 国外研究现状

近年来，大量的研究集中在数据收集上。数据聚合中有三个重要问题。第一个问题是选择聚合点；第二个问题是数据聚合中的路由算法；第三个问题是关于过期条件(例如，基于时间的聚合机制中的缓冲时间)。大多数关于聚集的相关工作只研究了前两个问题。例如，引用[1]和[2]建议使用聚合树来查找聚合点，以最大化生存期。在[3]中，Pandey 建议使用聚类算法来选择几个好的聚合点，以减少要发送到网关的消息的数量。在[4]中，Krishnamachari 提出了一种以数据为中心的路由算法，以减少传输次数并延长生存期。在[5]研究中提出了一种能量感知路由算法。此外，Mottola 通过实验表明，数据聚合可以提高整个网络 80%的生存期。然而，数据聚合的副作用是显著的，其中传输延迟随聚合增加约 4 倍。此外，[5]只比较有聚合的路由算法和没有聚合的路由算法的性能。上面的工作没有研究聚合的过期条件(例如，设备应该为数据聚合缓冲多长时间的消息)。

1.2.2 国内研究现状

1.3 本文结构安排

本文将根据上述阐述的问题，深入探讨异构物联网的优化目标并建模，最终提出一种基于 PSO 算法的节能聚簇协议。

第二章将介绍本文使用的网络系统模型，包括需求分析，网络节点定义，异构性定义，能耗模型，数据融合模型。

第三章将介绍网络的优化目标，并给出优化目标的数学建模。

第四章将介绍如何使用 PSO 算法聚簇，探讨适应函数设计与协议详细流程。

第五章将介绍本文使用的测试系统，并给出基于网络生存时间、网络能耗、网络剩余能量分布等维度的测试结果，再与其他传统协议和同类型协议进行比较，得出对本文提出协议的客观评价。

2 网络系统模型

近年，物联网设备种类愈发繁杂，从智慧家居到智慧城市，从传感器节点到自动化大型设备，每个 IoT 节点的功能不尽相同，着重点更是不尽相同。故目前的物联网网络模型难以涵盖全部的 IoT 种类，在异构方面需要有所取舍。由于当今 IoT 终端设备多为传感器设备，本文将着重于研究传感器节点的接入控制与能耗管理。此外，为适应其他种类 IoT 终端设备，如 Edge/Fog 节点、云计算节点、协同节点等，此网络模型将引入多种异构性，以保证后文的自组织网络协议应用场景更加全面。

本章节将从物联网功能切入，引出各类节点的通信需求，进而构建完整的网络系统模型。

2.1 物联网功能分析与通信需求

物联网网络需求可以划分为接入控制与路由管理两大类，接入控制即实现对网络内节点登记，授权，计费等功能，并且提供网络转换等服务；路由管理则是指对网络内节点报文进行转发，提供基本的 M2M 通信服务。而针对传感器网络而言，由于传感器能量有限并且不易补充，故网络最大的问题在于如何提高网络的生存时间。总结而言，本网络模型中将主要考虑两项指标：

1） 网络覆盖率

2） 网络能耗

由于网络中包含大量协同任务以及数据收集与融合任务，本网络模型将引入分层概念，引出三类逻辑节点：

- 普通节点：即功能实现实体，物联网终端设备，例如传感器节点、智能设备等。
- 汇聚节点：完成数据融合、数据存储、接入控制、网络转换等功能的节点，例如 Edge/Fog 节点、云计算节点、接入管理节点、网关等。
- 中央节点：实现区域能耗管理、区域总数据库、接入授权等任务的节点。

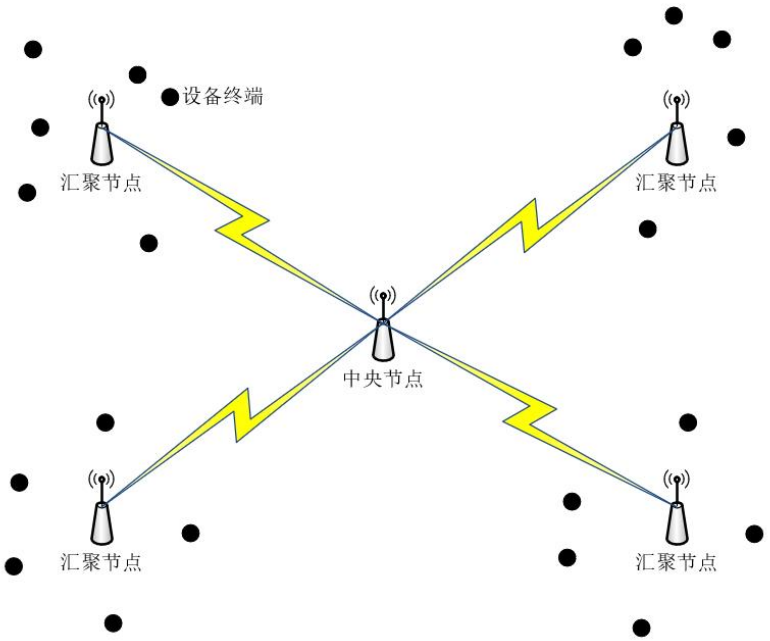


图 逻辑节点模型示意图

根据上述三类节点，网络内通信需求可分为点-点式、点-汇聚式以及点-汇聚-中央式：

- 点-点式：指普通节点间的直接通信，不需要汇聚节点协助转发。点-点式通信可以为多跳传输。由于此类通信需求仅发生于智慧节点进行协同任务时，并且认为协同任务中节点距离较近，故可使用 DSDV 等路由发现类协议。此类协议已经较为成熟，故本文不加以赘述。
- 点-汇聚式：认为此类通信需求发生于汇聚节点收集普通节点数据，节点接入控制，或者有异构网络转换请求等，后文将给出基于 PSO 算法的异构网络接入与能耗管理协议，用于实现上述任务。
- 点-汇聚-中央式：此类通信发生于中央节点对整个区域内进行控制调度、数据分析等过程中。为简化网络模型，认为汇聚节点与中央节点可以通过计算机通信网络或移动通信网络等现有基础设施通信。

根据上述分析，后文给出的模型与协议将主要应用于点-汇聚式通信。

2.2 物联网异构网络模型

本章节将探讨本文所使用的网络模型，用于后文算法协议描述。

网络包含一个汇聚节点BS，N个传感器节点与M个接入控制节点，将其分布在$X \times Y$的矩形场地中。传感器节点使用集合$S = \{s_1, s_2 \ldots, s_n\}$表示，接入控制节点使用集合$C = \{c_1, c_2, \ldots, c_n\}$表示。为了可以公式化聚簇问题，本网络模型将基于以下几点假设：

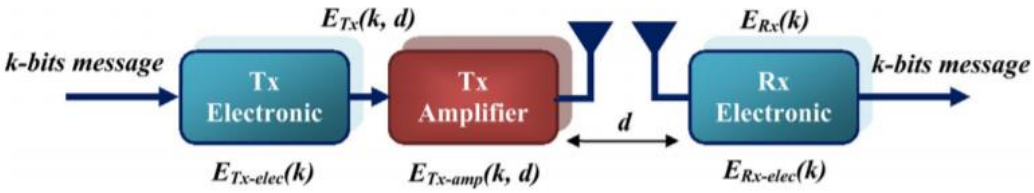
- BS 能量足够充足，并且有足够的计算能力。
- 所有的传感器节点具有唯一的 ID，并且都具有 GPS 等地理位置感知功能。[1][2]
- 所有节点包括 BS，在部署后都为静止的（后文将给出适应移动 BS 的方法）。
- 节点可以根据对发射功率的控制，来调整发射距离。
- 聚簇后，由于簇内信息高度相似，簇头可以对簇内节点的信息进行数据融合，而簇间传输时，信息不能认为是高度相似的，故信息不能被融合。

针对于网络异构性，给出以下几点假设：

- 网络内传感器节点发言概率近似，并且报文长度恒定。
- 节点地理位置分布不一定为均匀分布，存在分布异构。

2.3 能耗模型

为了与其余聚类路由协议相对比，本文将使用下图发射机与接收机模型：



对于发射机而言，发射$kbit$数据，发射距离为d时，所消耗的能量为：

$$E_{Tx}(k, d) = kE_{elec} + E_{Amp}kd^{p} \quad (1)$$

接收机所耗能量为：

$$E_{Rx}(k) = kE_{elec} \quad (2)$$

其中，E_{elec}和E_{Amp}分别代表电路中的耗散能量与放大器中的耗散能量，p代

[1] Y. Yao, N. Jiang, Distributed wireless sensor network localization based on weighted search, Comput. Netw. 86 (2015) 57–75.

[2] G. Wu, S. Wang, B. Wang, Y. Dong, S. Yan, A novel range-free localization based on regulated neighborhood distance for wireless ad hoc and sensor networks, Comput. Netw. 56 (2012) 3581–3593.

表信道损耗。根据距离阈值$d_0 = \sqrt{E_{fs}/E_{mp}}$，当$d < d_0$时，$p = 2$；当$d > d_0$时，$p = 4$。

除通信过程中的能量耗散，每一个节点在进行收集数据时也会消耗十分明显的能量，当收集$kbit$数据时，节点所耗能量为：

$$E_{Sen}(k) = kE_{sensing} \quad (3)$$

由于簇内传感器数据被认为是高度相似的，故簇头CHs可以进行数据融合技术将多个传感器数据合并为一个数据包，本文认为此过程也会消耗大量能量。对于CHs融合m个$kbit$数据包消耗的能量为：

$$E_{agg}(k) = mkE_{DA} \quad (4)$$

当各节点根据协议成簇后，网络被分成L个大小不等的簇，并且每个簇包含$m_i(i = 1,2,\dots,L)$个簇内节点，每个节点发出的报文为$kbit$，每个簇内节点需要收集数据并将数据传输至 CH，因此给出簇内节点l与簇头i之间，一次通信过程中簇内节点l所消耗的能量如下：

$$E_{mem-i}(l,k,d) = E_{Sen}(k) + E_{Tx}(k,d)$$
$$= \left(kE_{sensing}\right) + (kE_{elec} + E_{Amp}kd(l,i)^p) \quad (5)$$

其中$d(l,i)$代表节点l与 CH 节点i之间的距离。

另一方面，CH 所消耗的能量包括聚合簇内数据能量损耗，发射与转发数据包能量损耗。本文认为 CH 在一轮内将经历以下几点能量损耗过程：

- CH 自身收集数据的能量损耗。
- CH 收集簇内信息的能量损耗。
- CH 对簇内信息进行聚合的能量损耗。
- CH 接受其他 CH 的报文的能量损耗。
- CH 像 BS 发送报文的能量损耗。

故簇头i的能量损耗可以表示为：

$$E_{CH}(i,k,d) = kE_{sensing} + kE_{elec}m_i + kE_{DA}(m_i + 1) + kE_{elec}realys(i) +$$
$$(relays(i) + 1)\left(kE_{elec} + E_{Amp}kd(i,NH)^p\right) \quad (6)$$

其中，$d(i,NH)$是 CH 节点i到下一跳节点的距离，$relay(i)$代表从其他 CHs 接受到的转发包数量。

2.4 物联网数据聚合模型

物联网通信技术为设备之间提供了有线和无线通信的能力，其中机器对机器通信（M2M）占有了很大的比例。主要的 M2M 应用包括报警和实时数据监测。考虑图 1 的场景，其中数据在 M2M 设备(如传感器)上收集(参见图 1(a))。然后，M2M 设备通过 M2M 网关(见图 1(c))将数据发送到应用服务器或中间件(见图 1(b))。

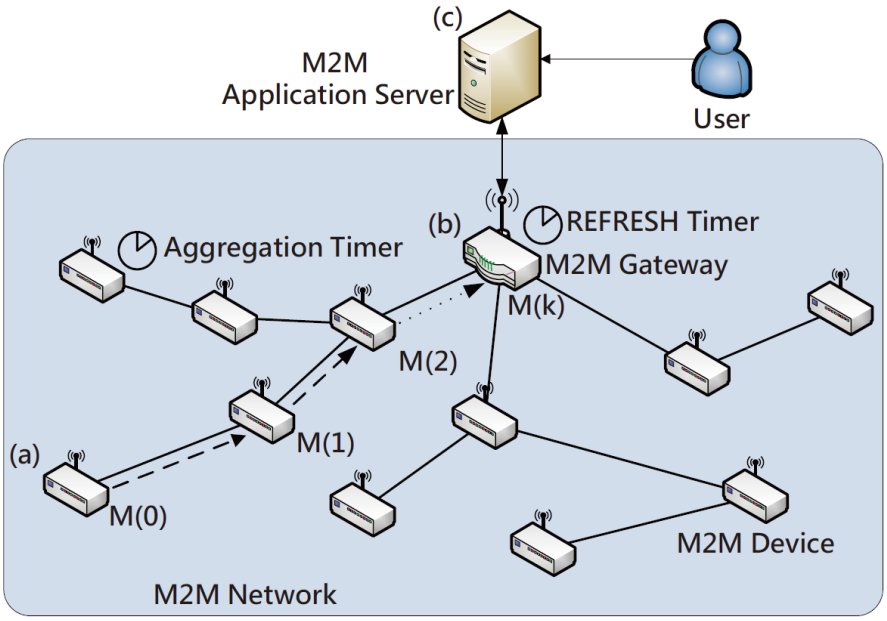


图 1　M2M 通信结构

当 M2M 网络的规模变得非常大时，可以在 M2M 设备(包括簇头和传递节点)上观察到大量的传输开销。由于 M2M 设备可以从同类设备中检索传感数据，因此通过数据聚合减少传输开销是延长 M2M 网络寿命的有效方法。

为了利用数据聚合节省能量开销，M2M 设备需要进行消息缓冲。可以方便地基于时间的机制利用缓冲计时器。在缓冲计时器到期时，M2M 设备将检索在缓冲期间收集的数据，并将它们聚合到一条新消息中。当设备延长缓冲时间时，消息中的总容量和能量效率都会提高;但是，从端节点到应用服务器(或 M2M 网关)的传输延迟会增加。为了解决这个问题，我们提出了一个分析模型来研究缓冲时间的配置，以平衡利弊。

3 优化问题模型化

本章将对网络连通性与网络能耗两个优化目标建模，并利用 MOIA 算法给出解决多目标问题办法。

3.1 网络连通性问题

针对传感器节点$sn_l(x_l, y_l)$，此节点被簇头节点$CH_i(x_i, y_i)$所涵盖的概率可以描述为：

$$P(sn_l, CH_i) = \begin{cases} 1, & \sqrt{(x_i - x_l)^2 + (y_i - y_l)^2} \le R_C(i) \\ 0, & otherwise \end{cases} \quad (7)$$

由于网络涵盖L个 CHs，因此节点sn_l被涵盖的概率可以表示为：

$$P(sn_l, CHs) = 1 - \prod_{i=1}^{L} \left(1 - P(sn_l, CH_i)\right) \quad (8)$$

因此，网络的覆盖率（R_{Cov}）可以由被涵盖的节点数量与全部存活的节点数量计算得：

$$R_{Cov} = \frac{\sum_{l=1}^{N_{live}} P(sn_l, CH_s)}{N_{live}} \quad (9)$$

3.2 多目标问题

网络最优条件分为最大化网络覆盖率与网络生存时间，本文认为网络生存时间指网络开始运行到网络中第一个节点死亡的时间，故该问题可以描述成求一个多目标问题（MOP）的最优解。而最优解可以被描述为：

- 最小化网络中传感器在一轮内的能量总消耗：

$$\min\left(E_{disp} = \sum_{i=1}^{L}\left[\left(E_{CH}(i,k,d)\right) + \sum_{l=1}^{m_i}\left(E_{mem_i}(l,k,d)\right)\right]\right) \quad (10)$$

- 通过最小化节点未被覆盖的概率，从而使网络覆盖率最大化如下：

$$\min(R_{Uncov} = 1 - R_{Cov}) \quad (11)$$

因此，此 MOP 问题可以使用 MOIA 算法，最小化下式解决：

$$F = w_1 E_{disp} + w_2 R_{Uncov} \quad (12)$$

其中w_1, w_2为$[0,1]$的权重参数，并且$w_1 + w_2 = 1$，参数值越高，代表该目标的着重程度越高，选簇结果也越偏向于该目标。下一章节将引入 PSO 算法，寻找上式的全局最优解，以达到网络高覆盖率与低能耗的目的。

4 基于 PSO 的异构网络接入及能耗管理协议

本章节要解决的问题可以分为两个：低功耗网络路由协议与网络接入控制节点的部署。其中前者为网络节点可以相互通信的基本前提，后者为网络接入控制的基本前提。对于路由协议设计，主要考虑到网络能耗，网络生存时间（第一个节点的死亡时间与最后一个节点的死亡时间），网络剩余能量分布与网络覆盖率四项指标；对于接入控制节点部署问题，主要考虑网络覆盖率与非控制节点的能耗两项指标。下面将阐述本协议的设计思路，易于对后文的理解。

网络形成可以分为三个部分，一为簇头选举。簇头是指为实现多跳传输与数据融合，需要在普通节点中选举出一些节点作为 CH 而承担上述任务。二为簇的形成，非簇头节点（ON）需要根据权重选出自身即将接入的簇头。三为数据传输，簇内节点将数据传入 CH，CH 进行数据融合后，再通过其他 CH 多跳传输，最终传输至 BS 节点。聚簇类型协议会带来很多好处，比如物理层易发生的碰撞问题可以使用 TDMA 的方式解决，簇头对簇节点进行时分复用，既减少了碰撞，又使得簇内节点可以进行定期休眠以节省能量，同时还满足了数据融合的需求。在使用自组织的聚簇协议后，网络的可扩展性和自愈能力大大提高，这对于物理网来说是十分关键的。故对于大型网络而言，聚簇是目前最有效的手段。

当然，引入聚簇并不意味着解决了所有问题，由上述网络形成流程可见，CH 虽然在物理结构上与其他节点相同，但是会承担更多任务，例如数据融合、信息转发等等。所以预设置 CH 的方法是不可取的，这会使 CH 节点加速死亡，导致网络拥塞、网络分割化等致命问题。LEACH 协议提出了 CH 轮换机制，使每个网络

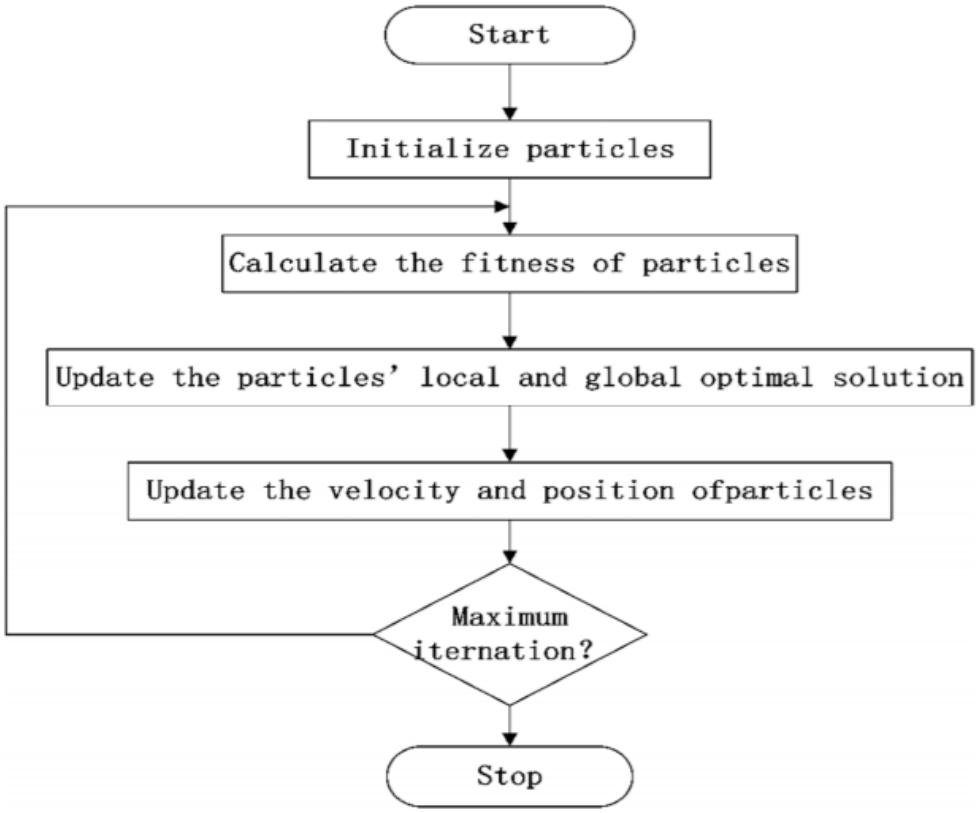
节点都有成为 CH 的机会，如此一来聚簇所带来的额外开销可以分布到整个网络上，以提高网络生存时间。然而 CH 的开销在不同节点上是不同的，根据第二、三章可以清楚的发现 CH 的选择直接影响网络开销，一组坏的 CH 甚至会加速网络死亡，如何在一个节点个数为 N 的网络中选出最优数目的 CH 是一个 NP 完全问题，并且每个 CH 下连接哪些节点同时是一个 NP 完全问题。所以可见，网络聚簇技术是一个十分难解决的问题，当然我们可以使用遍历的方法，模拟出每一种聚簇方式，通过计算式(12)来评估该聚簇方法的好坏。然而经过简单计算可以，当网络节点个数为 N，CH 数目为 L 时，遍历的复杂度高达$O(C_N^L \times L \times N)$，当网络节点个数过高，尤其在传感器网络种，这种方式显然是不可取的。所以，本文提出一种基于 PSO 算法的优化方式，来达到在有限遍历次数中选出较优秀的聚簇方式。至于控制节点部署问题，同样可以理解为另一种聚簇方式，故可以使用相同的算法来解决。

4.1 PSO 算法概述

PSO 算法全称粒子群优化算法（Particle swarm optimization），是一种进化计算技术（evolutionary computation）。源于对鸟群捕食的行为研究。粒子群优化算法的基本思想：是通过群体中个体之间的协作和信息共享来寻找最优解。

粒子群算法通过设计一种无质量的粒子来模拟鸟群中的鸟，粒子仅具有两个属性：速度和位置，速度代表移动的快慢，位置代表移动的方向。每个粒子在搜索空间中单独的搜寻最优解，并将其记为当前个体极值，并将个体极值与整个粒子群里的其他粒子共享，找到最优的那个个体极值作为整个粒子群的当前全局最优解，粒子群中的所有粒子根据自己找到的当前个体极值和整个粒子群共享的当前全局最优解来调整自己的速度和位置。

随着迭代次数增加，粒子群中的全局最优解将会逐渐逼近待解决优化模型的最优解，算法流程图如下：



4.2 使用 PSO 算法求优化模型最优解

PSO 算法目标有两个，一是寻找最佳 CHs，二是寻找最佳接入控制节点位置，二者本质上都可以理解为寻找一种聚簇方案。

先设想一个传感器网络，所有节点与 sink 节点通信都是单跳的并距离小于 d0，那么在不考虑碰撞带来的额外开销，整个网络的能耗一定是最低的，因为没有任何额外的开销。那假设我们需要引入聚簇手段来实现 M2M 通信，并且簇内节点与 CH 的通信是单跳的，则仅有 CHs 会承受额外的开销。但我们并不知到 CH 位于何处时网络开销最小，故本文引入一组虚拟 CHs（VCHs）用于描述假定的 CH 所在的地理位置，通过算法迭代得出最优 VCHs，再通过寻找最近的大于平均能量的普通节点，作为当前轮次的 CHs。

4.2.1 虚拟网络划分

在聚簇协议中，一个十分关键的问题是如何确定簇头数量，本文将引入虚拟

网络的概念，根据节点密度划分大小不同的虚拟区域，一个虚拟区域中包含一个VCH。实际上单纯的 PSO 算法在一个均匀分布的节点群落中效率很高，但是一旦出现分布上的密度不均匀，网络生存时间则会有显著下降，这是由于

（1）虚拟 CH 节点的分布在非均匀分布的情况下不合理

（2）在相邻的两次簇头迭代替换之间部分节点的能量损耗差值过大造成的 PSO 算法失衡造成的。

因此基于上述原因，本文提出根据节点分布密度，将节点分布区域分割为大小不同的虚拟区域，来满足非均匀密度下的虚拟 CH 的合理分布需要。以下首先就如何分割小区域进行讨论，其次就引入虚拟区域后的能量损耗模型修正进行讨论。

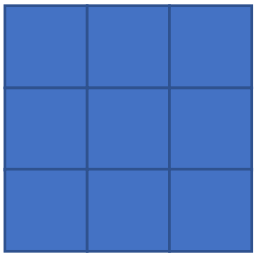
（A）区域密度分割算法

a）初始化时，先将区域划分成为 m×n 的小区域，如下图（1）所示

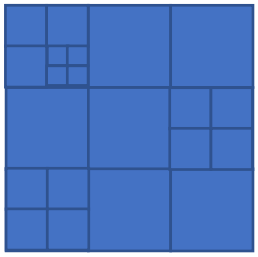
b）分别计算每个小区域内的密度，与阈值比较，若大于阈值则执行 c），若所有小区域的密度均小于阈值，则执行 d）

c）将超过阈值的小区域继续划分为 4×4 的小区域，返回步骤 b）

d）获取区域中所有的划分好的区域（不等大），放入区域列表中，供后续指派虚拟 CH 使用。分割完成后区域如下图（2）所示



(1)



(2)

（B）能量损耗模型修正

当引入虚拟划分时，在每个虚拟区域都有一个 CH，这样保证了网络覆盖率一定为 100%，故此时(12)式简化为：

$$F_{CHs} = E_{disp} \quad (13)$$

在寻找最佳接入控制节点位置时，不需要考虑接入控制节点的能量损耗，则此时(12)式简化为：

$$F_C = w_1 \sum_{j=1}^{M} \left(\sum_{l=1}^{m_j} E_{Tx}\left(k, d_{lj}\right) \right) + w_2 R_{Unocv} \quad (14)$$

式中d_{lj}代表节点l与控制节点j的距离。

下一节将着重讲解两个不同目标下，PSO 粒子的权重设置。

4.2.2 寻找最优 CHs 时 PSO 粒子权重设置

此时 PSO 粒子P_i含义可以定义为网络中所有 CH 的地理位置(X_{CHs}, Y_{CHs})，每个粒子下的 CH 的地理位置记为P_{il}，即第i个粒子中的第l个 CH。根据式(13)，给出每个 CH 的权重：

$$F_{il} = E_{CH}(i,k,d) + \sum_{l=1}^{m_i} \left(E_{mem_i}(l,k,d)\right) \quad (15)$$

需要注意的一点为，此时的P_{il}上不一定有一个真正的节点，故假定一个虚拟 CH 用于维持算法运行，在算法结束后会就近寻找一个节点作为真实 CH。

最终可得粒子P_i权重为：

$$F_i = \sum_{l=1}^{L} F_{il} \quad (16)$$

4.2.3 寻找最佳接入控制节点位置时 PSO 粒子权重设置

此时 PSO 粒子P_i定义为接入控制节点的地理位置(X_{Cs}, Y_{Cs})，每个粒子下的接入控制节点C_j的地理位置记为P_{ij}，根据式(14)第一项，对于每个节点而言，需要计算所有 member 的能耗，即：

$$F_{ij} = \sum_{l=1}^{m_j} E_{Tx}(k, d_{lj}) \quad (17)$$

式(14)第二项可以通过计算所有被覆盖到的节点个数得出，不再赘述。则粒子P_i的权重为：

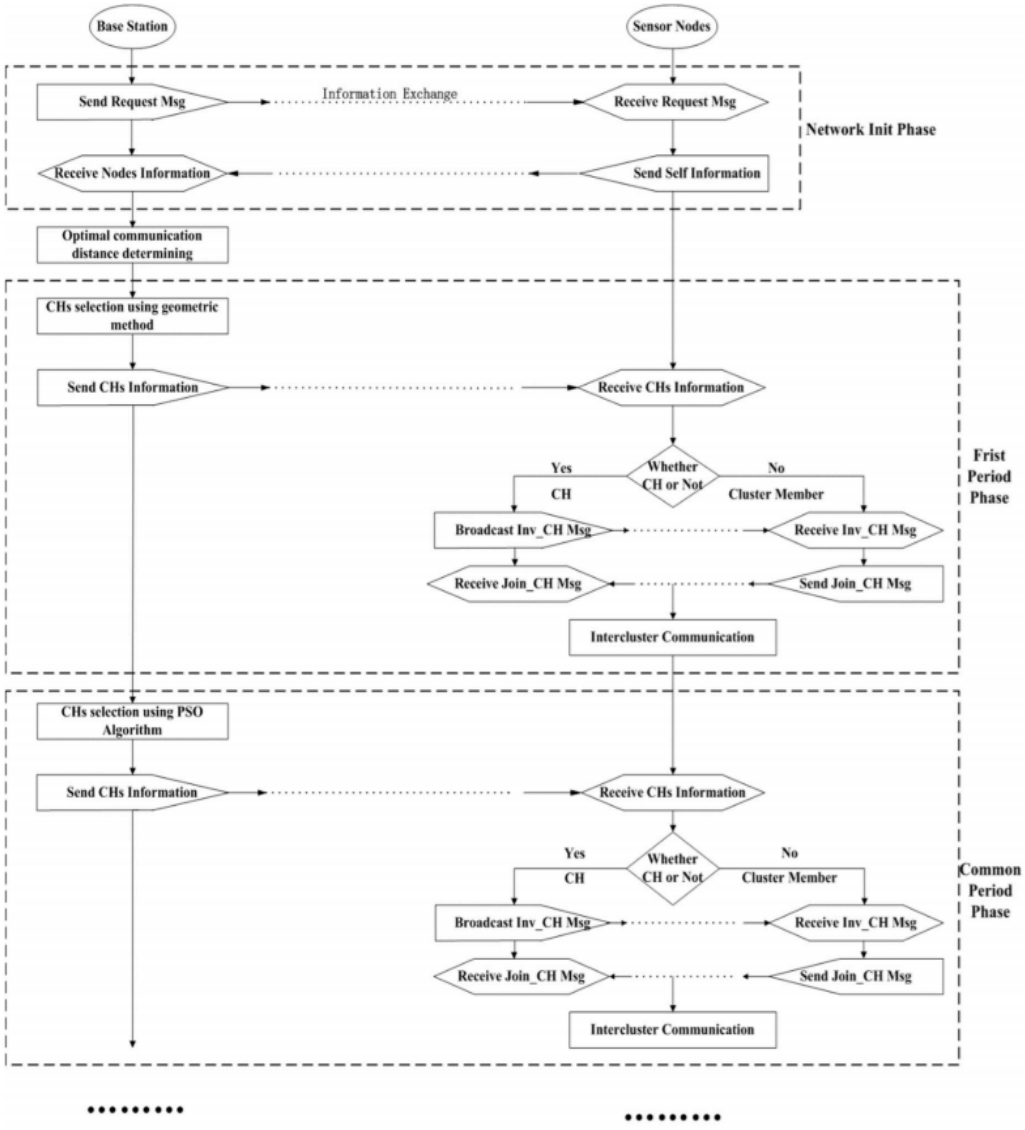
$$F_i = w_1 \sum_{j=1}^{M} F_{ij} + w_2 R_{Unocv} \quad (18)$$

4.3 网络初始化阶段

此阶段需要对网络进行虚拟划分，由于节点分布可能为非均匀的，下面介绍对于分均匀节点的虚拟划分机制。

在划分完虚拟区域后，对每个区域随机设立一个虚拟 CH，开始运行 PSO 算

法。整个协议运行流程图如下图所示：



4.4 簇头选举阶段

许多使用虚拟划分来确定簇头的算法中，都隐含着一个前提，即每个区域内的簇只能由每个区域内的所有节点组成。然而这本身就是一个求次优解的过程，因为并不能证明最优簇中仅含本虚拟区域中的节点。故该阶段实际上隐含两部分内容，一为最佳簇头位置（VCH）的确定，二是每个簇的 members 的确定。首先根据能量模型，节点通信受比特数与距离两个参量决定。在网络模型中，我们假设每个节点的发言概率相同，并且每次发言的比特数相同，故每个节点通信一次

所消耗的能量仅由距离决定。那么在聚簇时，显然节点要选择距自身最近的一个CH，那么如果我们按照密度划分虚拟区域，则每个 VCH 下的 member 个数的数学期望应为 N/L，这就保证了每个簇的节点个数是相同的，簇头所承担的额外开销会均匀分布在其他节点上，大大提高了网络的生存时间。下面使算法运行的具体过程：

设使用 M 个粒子，每个粒子中有 N 个 VCH 节点，则粒子群可以表示为矩阵形式如下：

$$S = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_M \end{bmatrix} = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,N} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,N} \\ \vdots & \vdots & \cdots & \vdots \\ P_{M,1} & P_{M,2} & \ldots & P_{M,N} \end{bmatrix} \quad (20)$$

其中$P_{i,j} = (x_{CH}, y_{CH})$，代表每个虚拟 CH 的位置。

下面是 PSO 算法的具体过程：

Step 1：初始化粒子位置与速度，在每个虚拟区域内随机生成 CH 节点位置，粒子速度同样使用随机的方式。在每一轮开始时，每个节点将播报它的基本参数，包括 ID，地理位置，剩余能量等。

Step 2：当 Sink 节点接受到报文后，使用公式(17)计算每个粒子的 F 权重，并选出粒子最优粒子 Pbest 与全局最优 Gbest 粒子。

Step 3：通过下式更新每个粒子速度与位置：

$$V_i(t+1) = wV_i(t) + c_1 \times rand \times \left(P_{ibest} - P_i(t)\right) + c_2 \times rand \times (G_{best} - P_i(t))$$

$$P_i(t+1) = P_i(t) + V_i(t+1)$$

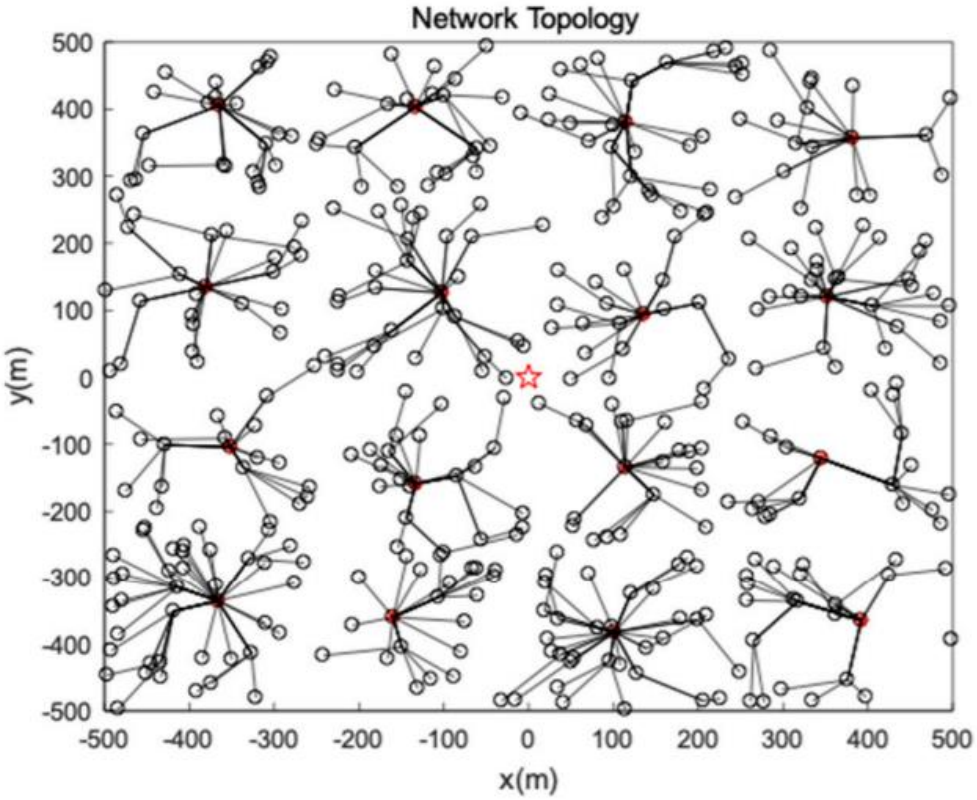
其中c_1，c_2代表学习因子，w代表惯性因子。

当迭代次数超过预先规定数值后，我们可以获取到当前 CH 的最优地理位置G_{best}，之后开始选择实际 CH。

对于每个虚拟 CH 中，计算簇的平均能量作为阈值：

$$E_{Th} = \frac{\sum_{l=1}^{m_i} E_{res}}{m_i}$$

在超过阈值的节点中，选择与虚拟 CH 位置最为接近的节点作为此轮的 CH，至此 CH 选举结束。



4.5 簇内通信阶段

我们先介绍基于时间的数据融合机制。由于 M2M 通信网络的异构性，我们知道不同种类的节点具有不一样的数据包大小，以及对延迟的敏感性。每一个数据包最后都将被送到应用服务器。就像在图 1 中描绘的那样，数据被第一次采集之后通过的节点被依次描述为 M(0)、M(1)一直到 M(k)，我们的基于时间的数据融合机制被简化描述为以下几个步骤：

Step1：当节点 M(i)收到来自 M(i-1)的含有感知数据 e0 的数据包时，M(i)将它放入融合缓存等待检索。

Step2：如果融合缓存是空的，M(i)会启动一个融合计时器，单个周期是 T。

Step3：如果融合时间超时，M(i)会取走所有的缓存数据并把它们合成为一条新的信息，然后把这条信息传递给 M(i+1)

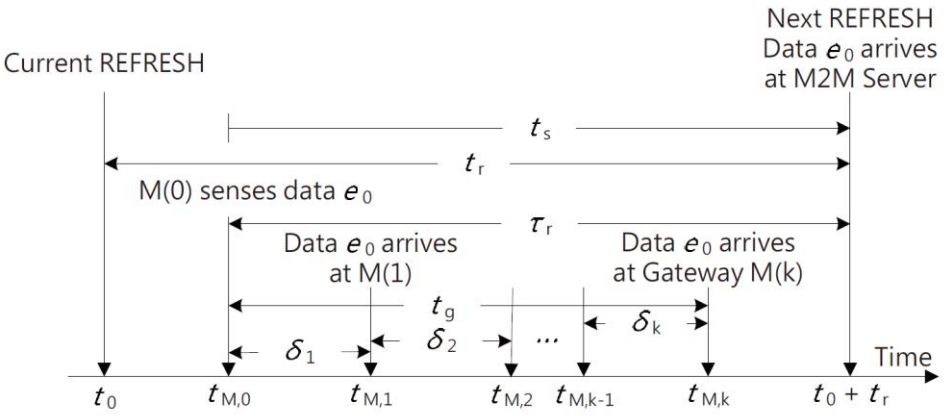
通过上面的描述，我们发现评价基于事件的数据融合机制的性能指标可以有

- 平均传递时延：数据 e0 从 M(0)处采集直到 M2M 应用服务器收到这个数据所

需要的平均时间。

- 平均融合大小：在一条消息中融合的数据包总数的平均值。

一个更直观的时序图展示如下：



假设从 M(i-1)到 M(i)的传输时延为 δ i,tg 表示为 M(0)到 M(k)的传输时延，

$$t_g = t_{M,k} - t_{M,0} = \sum_{i=1}^{k} \delta_i$$

但是，基于精确时间的数据融合机制会给仿真带来诸多不便，为了寻找一种更简单的仿真方法，我们下面的实验是基于轮数开展的。在程序预设中，我们假定有三种不同的节点，他们具有不同的数据包长度和时间（延迟）敏感度，并且都介于 0 到 5 之间。其中，0 意味着拒绝等待，需要立即发出。而 1 意味着可以容忍 1 轮的等待时间，以此类推。而每一个簇头（或称传递节点）具有有限的融合缓存，一旦融合缓存达到上限，需要立即将所有缓存数据踢出（检索），在这个实验中，我们的缓存上限预设定为 20。为了确保能保障数据包的时间延迟安全，我们在每一轮的开始会给所有的簇头减 1 个时间敏感度。那么，总结起来，当满足以下两种情况的任意一种时，簇头将会把数据融合后传递给父级：

1. 时间敏感度为 0
2. 缓存达到了上限

由于节点是概率发言，每一轮发言的节点不一样，那么，每一轮簇头的时间敏感度更新将会面临不同的情况。假如有从 1 到 k 的节点向这个簇头发来了数据包，同时这个簇头目前的敏感度为 s(ch0)，则更新后的敏感度 s(ch)表示如下

$$s(ch) = \min\{s(ch0), s(1), \cdots, s(k)\}$$

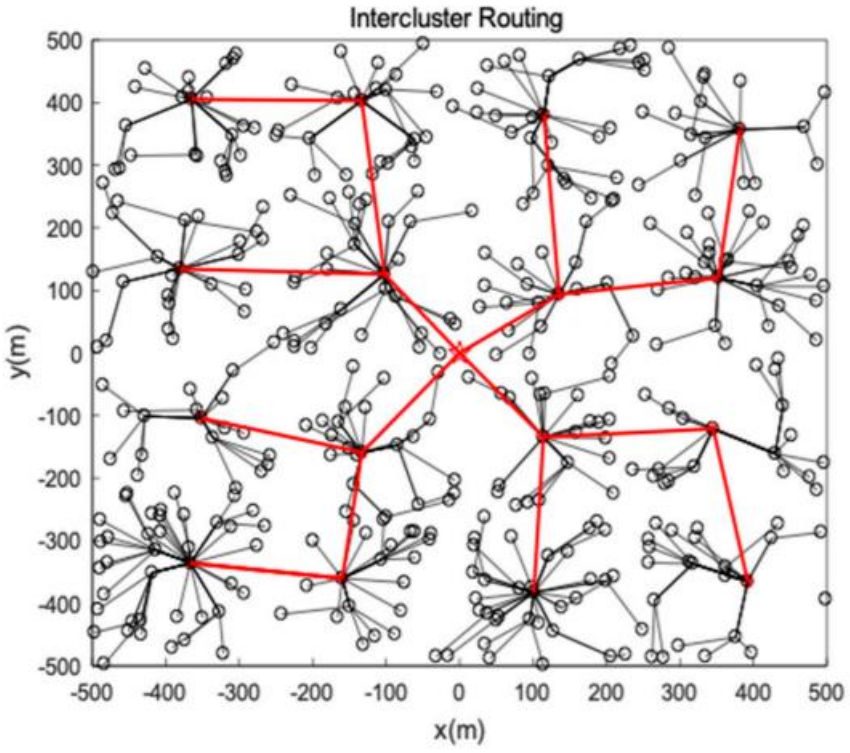
一旦 s(ch)更新到 0，则会判定为缓存时间超时，将缓存提出。

4.6 簇间通信阶段

传统的簇间通信方式为，在 BS 与 CH 组成的图结构中，按剩余能量形成一个最小生成树，或者使用一些其他的方式。但均不能解决“热点“问题，即越靠近 BS 的 CH，由于需要转发其他 CH 的报文，导致自身能量快速消耗，使节点加速死亡。故本阶段可以使用传统的簇间通信方式，但为了解决”热点“问题，本文引入移动 BS 的方法，后文将给出基于移动 BS 的簇间通信方法。

开始阶段时，移动收集器将位于图的中央，等待 BS 确定 CH 的地理位置。在取得 CH 的地理位置后，移动至簇平均能量最高的 CH 节点；移动收集器将播报 CH 的 ID 与收集器自身的位置，CH 接受到自身 ID 后，把簇信息发送至移动收集器；为使收集过程更加迅速，以该 CH 为源节点组成一个 H 跳的 BFS 树，于是周边 CH 信息可以一同被传输。

簇间传输时的网络拓扑结构如下图表示：



4.7 异构网络接入办法

此部分为计算接入控制节点的最佳部署位置，实际上与簇头选举方法一致，但粒子的权重函数 F 使用式(19)计算。显然在 4.4 章节中计算得来的最佳 CH 地理位置完全可以作为接入控制节点的地理位置，但为了使算法更具有适应性，下面给出使用 PSO 算法计算最佳控制节点部署方案的流程。

Step 1：初始化粒子位置与速度，将 M 个虚拟控制节点以均匀分布随机部署在待定区域中。BS 收集所有节点的基本信息，包括 ID、地理位置与剩余能量等。

Step 2：当 Sink 节点接受到报文后，使用公式(19)计算每个粒子的 F 权重，并选出粒子最优粒子 Pbest 与全局最优 Gbest 粒子。

Step 3：通过下式更新每个粒子速度与位置：

$$V_i(t+1) = wV_i(t) + c_1 \times rand \times \left(P_{ibest} - P_i(t)\right) + c_2 \times rand \times (G_{best} - P_i(t))$$

$$P_i(t+1) = P_i(t) + V_i(t+1)$$

其中c_1，c_2代表学习因子，w代表惯性因子。

最终可以求得最优部署方式G_{best}，将控制节点部署在相应位置即可。

5 接入及能耗管理系统仿真

本章节将介绍本文所使用的仿真模型，并给出仿真结果。本文使用 Python 进行算法仿真，并与其他同类经典或最新算法相比，从网络生存时间、网络能耗两方面进行比较，并引入节点发言概率异构与节点分布异构。首先将介绍在未使用移动 sink 技术时的指标检测，而后也会给出使用移动 sink 时的指标检测。下面将给出该仿真中的具体参数：

参数含义	值
网络规模(G)	$525 \times 525m^2$
节点数目(N)	1000
节点最大广播距离(R_{max})	$150m$
节点初始能量(E_0)	$1J$
电路能量损耗(E_{elec})	$150nJ/bit$

自由空间信道损耗参数(ε_{fs})	$150pJ/bit/m^2$
信息融合能量损耗(E_{Amp})	$15nJ/bit$
多径信道损耗参数(ε_{mp})	$0.015pJ/bit/m^4$
距离阈值(d_0)	$\sqrt{\varepsilon_{fs}/\varepsilon_{mp}}m$
数据包大小(l)	$1000bits$

PSO 算法具体参数如下表：

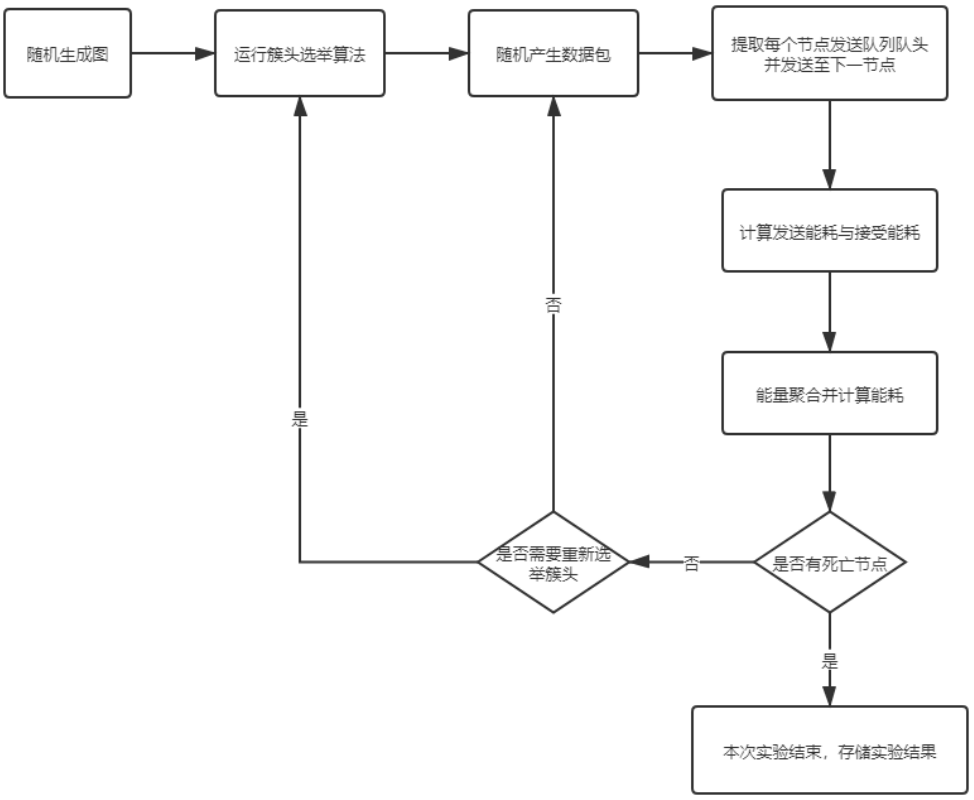
粒子个数	10
迭代次数(G_k)	20
惯性因子(w)	0.5
学习因子c_1	0.4
学习因子c_2	0.6

5.1 仿真系统设计

本仿真系统分为测试系统与子算法两部分，为保证在比较各个算法性能时的公平性，子算法与测试系统需互不干扰，并且参数完全相同，下面将分别介绍两部分的程序架构与测试内容。

5.1.1 测试系统程序架构

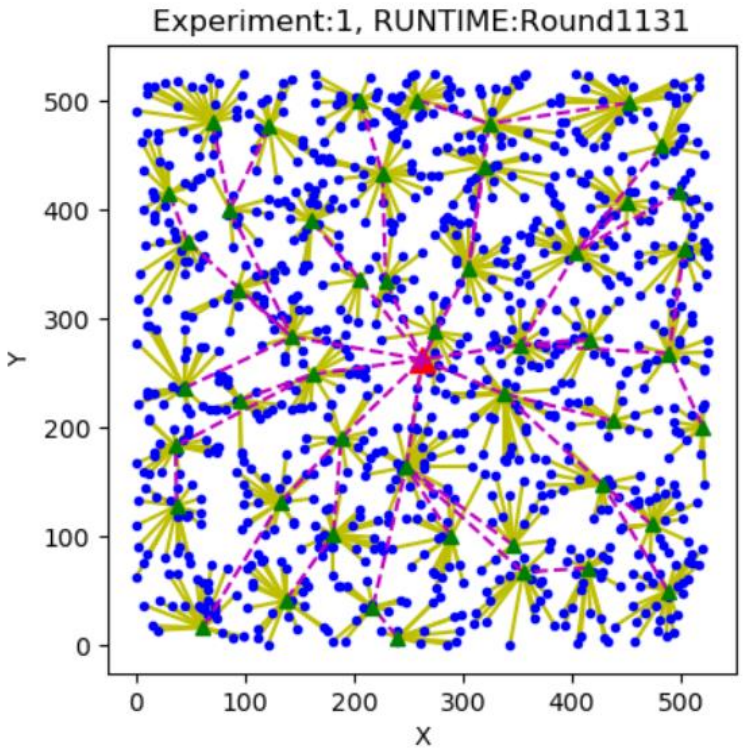
以测试第一个死亡节点为例，测试系统程序流程图如下：



5.2 网络能耗仿真结果

本章将给出 LEACH、ECPSO 与本文提出的 NewPSO 三种选簇算法的网络生存时间与网络能耗比较结果，并给出仿真结论。

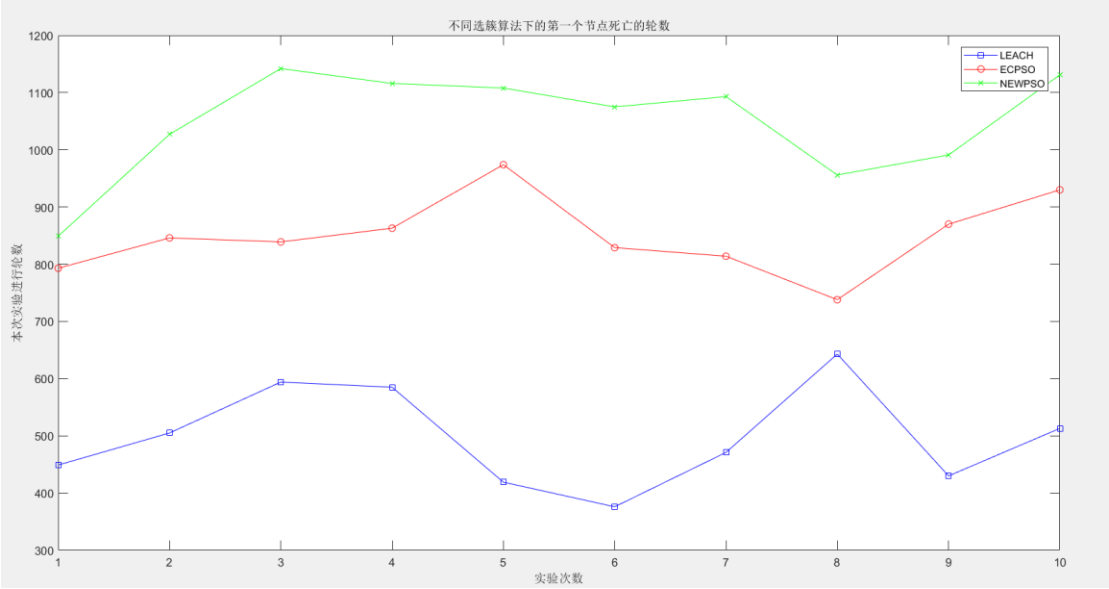
网络拓扑结构如下图所示：



其中，红色三角形为源节点，绿色三角形为当前轮选出的簇头节点，蓝色节点为普通传感器节点。数据将由蓝色节点产生，通过聚簇的方式将数据包传入绿色簇头节点，而后簇头进行数据融合。普通节点与簇头间用黄色线段表示。簇头间的传输路径由紫色虚线表示，最终将数据包传入源节点。

5.2.1 网络生存时间仿真结果

当前文献中评判网络生存时间的判断标准有两个，分别是第一个节点的死亡时间与最后一个节点的死亡时间。其中，第一个节点的死亡时间可以描述选簇算法所选出的簇头的额外消耗能量是否被其他节点所均摊，故大量文献以该指标为评判选簇方法好坏的标准。本仿真进行了 10 次实验（每次实验的网络拓扑图均为随机），每次实验的结果如下：

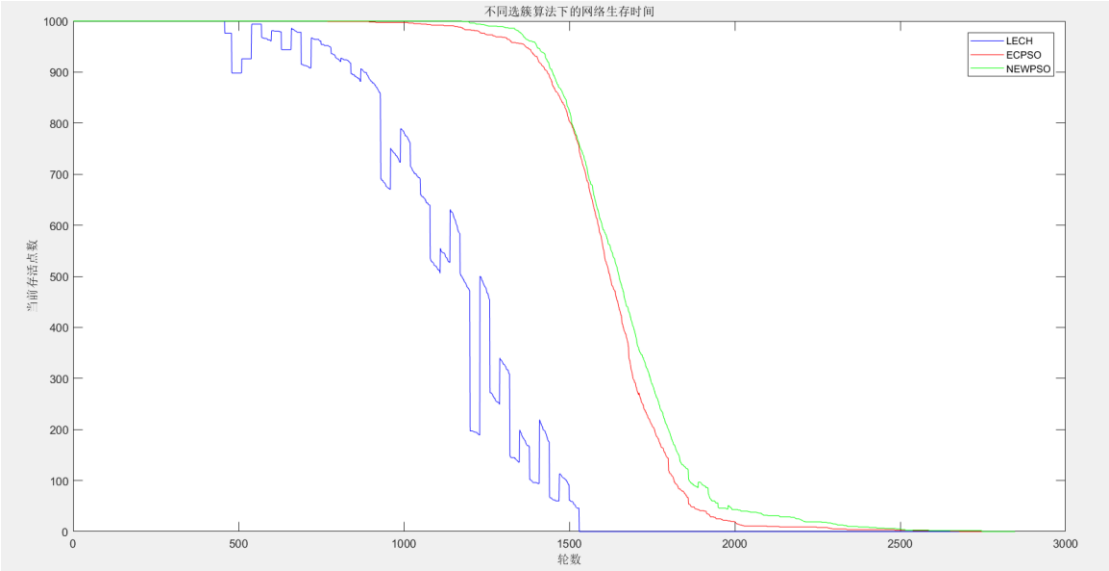


可见本文提出的 NewPSO 选簇算法延迟了网络中第一个节点死亡的时间，对十次实验取平均值可得：

- LEACH 平均生存轮数：498.5 轮
- ECPSO 平均生存轮数：849.6 轮
- NewPSO 平均生存轮数：1048.8 轮

相比于 LEACH 与 ECPSO 分别提升了 110.4%、23.4%，故得出结论：本文提出的 NewPSO 显著增加了网络生存时间。

若以网络中所有节点死亡为终止时间，三种算法所得到得网络生存时间图如下图：



此仿真中，认为如果节点无法连接到源节点，同样视为死亡，故有可能在选

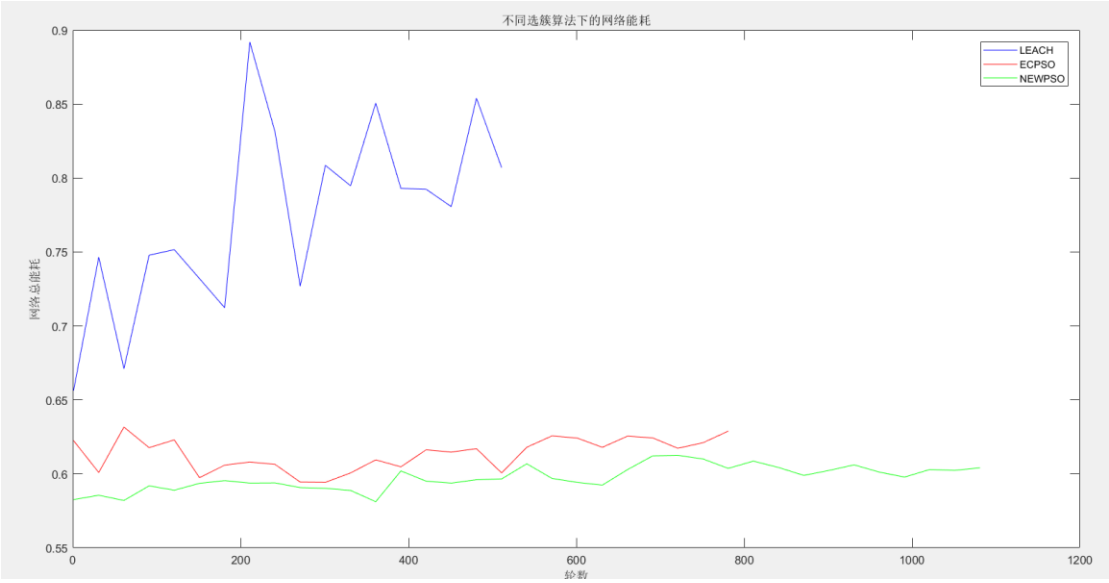
取不同的簇头时，生存节点个数不降反增，可以看到 LEACH 算法中生存节点数量波动很大，即反应了 LEACH 所选举的簇头并不能保证所有节点都被接入的缺陷。

同样的，我们可以看到 NewPSO 有效的延长了网络生存时间。

5.2.2 网络能耗仿真结果

由于节点大量死亡后，网络会因为拥塞、网络切割等问题大大降低网络传输效率，导致选簇结果不能反应真实的算法性能，故本指标将截取第一个节点死亡前的网络能耗数据，并最后给出剩余节点的能量状态与平均剩余能量。

网络能耗图如下：



由网络能耗的波动性我们可以发现，NewPSO 不仅降低了网络能耗，其还可以维持网络能耗在一个可控的范围内，进一步证明了算法的正确性。反观 LEACH 与 ECPSO 算法，网络波动较大，从理论上看 LEACH 为随机选择簇头，显然簇头选择位置会很大程度影响网络能耗；而 ECPSO 通过选取网络中的能量中心作为 CH，显然并不能作为网络能耗最小的最优解，仅能让剩余能量更高的节点担任 CH 来提高网络生存时间。

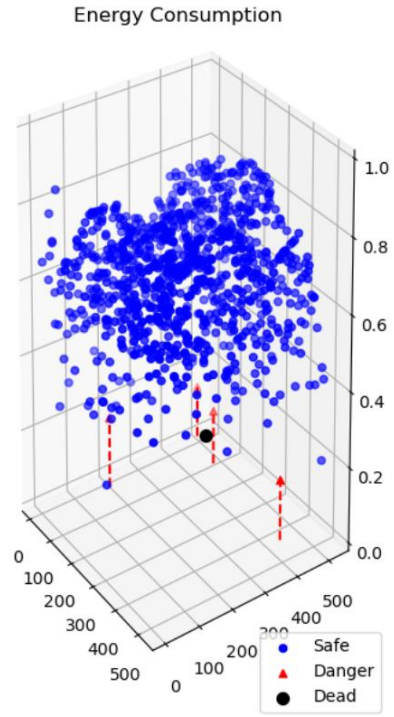
可以算出网络平均能耗分别为：

- LEACH: 0.7749J/轮
- ECPSO：0.6137J/轮
- NewPSO：0.5974J/轮

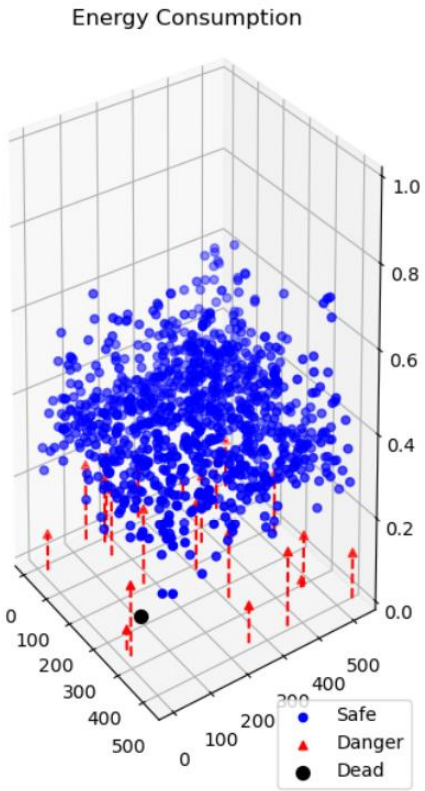
相比与 LEACH 与 ECPSO，NewPSO 所得的平均网络能耗分别降低了 22.9%、2.7%。

第一个节点死亡后，网络中存活节点的平均剩余能量分布为：

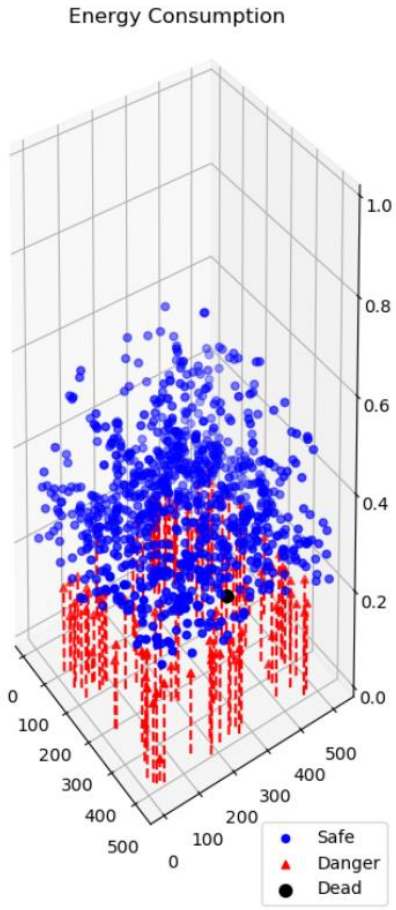
LEACH



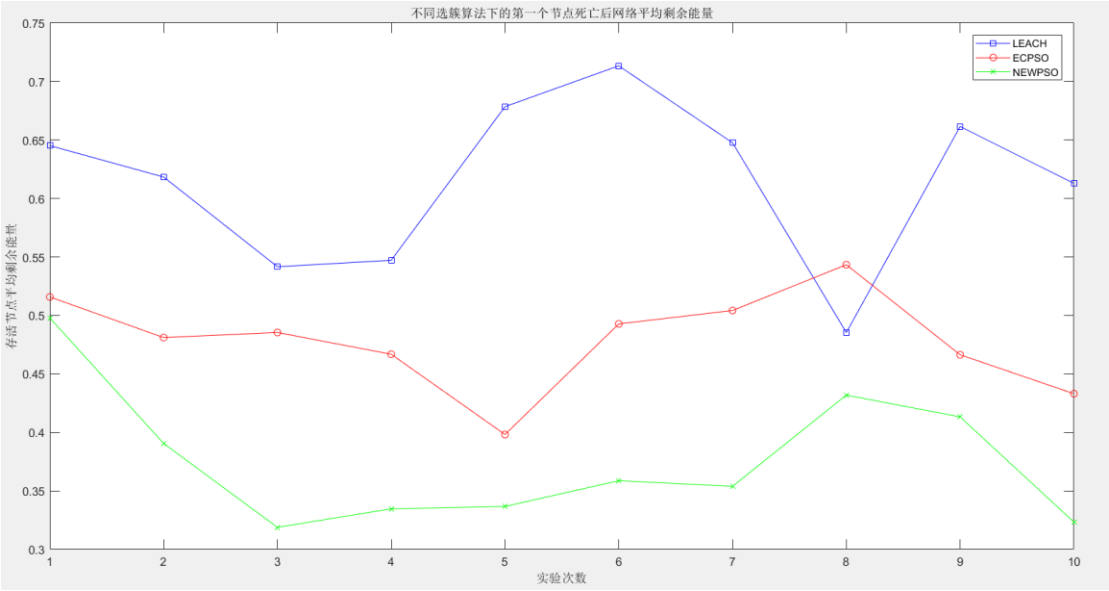
ECPSO



NewPSO



其存活节点的平均剩余能量为：



对十次实验取平均可得：

- LEACH：0.6152J

- ECPSO：0.4787J
- NewPSO: 0.3760J

可以发现在第一个节点死亡时，NewPSO 所得的网络节点分布更加均匀，意味着 CH 所承担的能量损耗更均匀的分布在网络上。

综上所述可以得出结论，NewPSO 算法显著降低了网络整体能耗，且网络能量分布更均匀。

5.3 网络接入仿真结果