A DHT-Based Heuristic Service Discovery Approach in MANETs

# Clustering

在MANET的无线网络环境中，由于数据的传送是以broadcast的方式实现的，因此若不对数据包的flooding加以限制，将会对网络带来较大的负载压力。Clustering，通过将网络中的节点分配不同的角色，将网络以cluster为单位进行划分，是一种限制数据包flooding的有效方法。

对网络节点进行分簇的方法大体可以分成proactive和reactive两大类。目前大多数的cluster方法是proactive的。在一般的 proactive cluster方法中，MANET中的节点持续地建立和维护cluster的结构，这导致了在网络中始终存在着大量的控制报文交换，使得MANET中宝贵的网络带宽资源被占用。

而reactive cluster方法不同，cluster的建立和维护是被动进行的。在Efficient Flooding with Passive Clustering—An Overhead-Free Selective Forward Mechanism for Ad Hoc/Sensor Networks中，作者提出了一种称为Passive Clustering的节点分簇策略。这种方法采用overhear网络中的数据包的方式建立和维护cluster，几乎不引入新的控制数据包。每个节点根据自身的状态以及overhear到的数据包中状态位的情况来动态维护cluster状态。

本论文提出的服务发现方法采用了类似Passive Clustering的reactive分簇策略，以最小的代价实现MANET中节点的clustering。

在网络中的每一个节点有以下几种状态：Initial、Cluster-Head、Gateway、Ordinary。Initial状态是节点的初始状态，在这个状态下的节点不属于任何cluster。Cluster-head和Gateway状态分别是一个cluster的leader和同时属于多个cluster的节点，这两种类型的节点被称为critical node。Ordinary表示节点是一个cluster中一个普通的节点。这四种状态需要piggyback在发送的数据包中已告知其他节点自身的状态，可以使用两个标志位表示。这些状态将会被overhear到这些数据包的节点获取并根据状态的不同

采取不同的处理。另外，节点还会有一种内部状态cluster-head-ready，这个状态信息不会随着数据包被其他节点获知。该状态表示节点有条件并将要声明自己为一个cluster head。

当一个处于初始状态Initial的节点接收到不是来自于cluster-head的数据包，则该节点有条件成为一个cluster-head节点，于是将状态修改为cluster-head-ready，如果在该节点转发此数据包之前仍然没有收到来自来自于cluster的数据包，则节点奖自身状态改为cluster-head并将该状态piggyback在转发的数据包中。此时该节点成为一个cluster-head。

如果此initial状态的节点接收到来自于cluster-head的数据包，则该节点成为该cluster中的一个Ordinary节点，并将cluster-head的信息存储在【cluster表中】。

如果一个ordinary节点收到超过一个的cluster-head发送的数据包，则说明该节点处于多个cluster的交叉之中，是两个cluster之间数据包transmission的critical节点。【根据情况？】修改本节点状态为Gateway。

而如果一个cluster-head节点收到来自另一个cluster-head节点的数据包，由于一个cluster只能有一个cluster-head存在，此时发生了冲突。必须有一方放弃cluster-head的角色。在此方法中，将有nodeid较小的节点主动放弃。

在此方法中，每个节点保留的【cluster表】中的项处于“软状态”，在项创建时会为其分配一个timer，如果在timer到期之前没有再收到来自该节点的数据包，则将其移除。【这里是不是应该使用neighbor detection的方法？】

在本论文提出的服务发现方法中采用分簇策略，一方面抑制了网络中广播的数据包数量，另一方面【使用在DHT Routing Table中存放cluster head id减小routing stretch影响】。

# DHT

P2P网络所提供的一个重要的功能是在分布式网络环境中实现资源的定位和索引。早期的P2P网络，如Napster、Gnutella等提出了一些比较原始的解决方案。Napster【？】采用了中心服务器对资源信息进行索引，并为P2P网络中的客户节点提供查询的服务，然而这种使用中心服务器的架构中节点并不symmetric，这和P2P网络分布式的特性并不符合，而且会有single point of failure问题的存在。Gnutella【】则保持了P2P网络的分布特性，当节点需要查询资源时，则向邻居P2P节点发出查询信息，采用在overlay network中进行flooding的方式查询资源。显然，这也并不是一个有效率的方案。

DHT是在分布式存储中进行资源索引的有效办法。在大多数的DHT实现中，一般采用一致性哈希函数（Consistent Hashing）为网络中的资源映射一个键值，并将一定区域的键值对应到网络中的每一个节点，以此在物理网络上构建一个覆盖网络（Overlay Network），使得网络中参与的节点数为的情况下，能够在覆盖网络跳数内获得需要的资源信息。一些常见的DHT实现，如CAN、Chord、Pastry、Tapestry都采用了类似的思路。

P2P网络与移动自组织网络有很多的相似性：两者都是自组织以及无中心架构的。因此，可以考虑将P2P网络中采用的DHT资源索引方法应用在自组织网络中以提高资源发现的效率和可靠性。然而，MANET相对于P2P网络又有很多的差异。其中最显著的在于节点的高移动性和网络的低可靠性。DHT维护中往往需要节点之间持续的数据交换，这在基于Internet的P2P网络中是可以接受的，但如果直接在MANET中应用将会带来过高的代价。

【Tapestry是一种典型的DHT算法。采用了类似【Plaxton中】Incremental Suffix-based Routing的方法，并考虑了节点的物理位置。】Pastry是另一种类似的算法，相较Tapestry，Pastry中的节点

除了存储一个路由表，还有一个Leaf Set和一个Neigborhood Set用来存储Node Id接近和物理上接近的节点信息。两者在locality的处理上也有一些小的差别。

【在【MMOG】中，作者提出了一种Hierarchical的DHT设计以支持在MANET中进行P2P多人网络游戏的设计，MANET中的节点clustering，cluster和cluster之间形成层次结构，在底层的cluster中采用了VRR进行索引，而cluster与cluster之间采用了Pastry进行索引。。。。】

【Etka也是一个将DHT应用在MANET中尝试】

【在。。。中，作者将Pastry应用在MANET中，提出了MADPastry方法。Pastry的特点在于Proximity-based，在节点的DHT路由表初始化以及维护的过程中采用Heuristic的策略使得路由表中存放的entry尽可能地在Physically接近本节点，使得能够尽量减少overlay hop所对应的physically hop。与Pastry一样，MADPastry同样采用了前缀匹配的路由选择方式。另外，MADPastry方法中采用了clustering的策略，在路由表中存放cluster head而不是普通节点的信息以增强entry的locality，以及减小了路由表的size。另外为了减少维护代价，MADPastry不再确保所有信息的up to date。】

在诸多DHT中，Pastry和Tapestry由于Proximity-based，比较适合对于路由跳数敏感的MANET。但是这两者尤其是Pastry节点保存的信息比较复杂，维护的开销对MANET而言会很大。MADPastry将Pastry加以简化，但其调整限于加入了clustering，不再确保neighboring Set的完全正确性，以及对网络中的数据包overhear并cache。从实验数据来看，这样的优化起到了效果，但这样的方法仍然给网络带来了过大的负载，并且在网路拓扑结构变化较快的情况下效果并不突出。

本论文提出的方法采用了类似于Tapestry的Incremental Suffix-based Routing 的DHT设计，并将其和与Passive Clustering相似的clustering方法相结合，在路由层采用AODV路由协议，整体采用了Reactive的设计，减少了维护的工作量和开销。