Computer Communications 32 (2009) 368–375

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Distributed topology control algorithm based on one- and two-hop neighbors' information for ad hoc networks

Mehdi Kadivar [a], M.E. Shiri [a,*], Mehdi Dehghan [b]

[a] Department of Mathematics and Computer Science, Amirkabir University of Technology, No. 424, Hafez Ave., Tehran, Iran
[b] Department of Computer Engineering and Information Technology, Amirkabir University of Technology, No. 424, Hafez Ave., Tehran, Iran

ARTICLE INFO

Article history:
Received 16 March 2008
Received in revised form 3 November 2008
Accepted 7 November 2008
Available online 24 November 2008

Keywords:
Distributed algorithms
Topology control
Ad hoc networks
Energy efficient

ABSTRACT

In this paper, we present a topology control algorithm for ad hoc networks. By considering the weight of the links, each node orders its one-hop neighbors in an ordered list and then the ordered lists are exchanged between the neighbors. This information enables the nodes to compute their transmission radius on the basis of its one- and two-hop neighbors' information. We demonstrate that compared to the best known algorithms, the degree and transmission radius of the nodes in the topology produced by the proposed algorithm are smaller. In addition to Euclidean graphs, the algorithm works correctly on general weighted graphs. Also an extension of our algorithm is proposed which adapts its topology to network changes. Finally, we use the four metrics, node degree, transmission radius, the power stretch factor and, packet loss ratio to measure the performance improvements of the presented algorithms through simulations.

1. Introduction

An ad hoc network is a group of wireless nodes that communicate over radio. These mobile nodes do not need any infrastructure. These kinds of networks are very flexible and suitable to use in applications such as natural disaster recovery or fast deployment of military units in unpopulated areas. Since the nodes in ad hoc networks are usually battery equipped, they have a limited amount of energy which must be used as efficiently as possible. A complete solution to energy efficiency involves many areas of research, such as hardware, operating systems, networking, and applications [7]. Our work focuses on a mechanism named topology control. The goal of topology control is reducing energy consumption by discarding inefficient links, while still satisfying connectivity. The power required to transmit messages between nodes increases as the exponent $\alpha \geqslant 2$ [14] of the distance between them. Therefore, compared to direct transmission, a smaller amount of energy can be consumed by a node u if it passes on its messages via a series of intermediate nodes to a node v. The topology control problem can be formalized as follows.

Let the graph $G = (V, E)$ denote the wireless ad hoc networks, where V is the set of ad hoc nodes and E is the set of communication links. Our aim is presenting an algorithm that supplies a connected subgraph $G_{tc} = (V, E_{tc})$ of G, where $E_{tc} \subseteq E$, with the following properties:

A.1 – *Connectivity*: The main property of subgraph G_{tc} is connectivity. Two nodes are connected if there is a path between them. If two nodes are connected in G, then they should be connected in G_{tc}. A.2 – *Symmetry*: G_{tc} should contain only bidirectional links, because most routing protocols for ad hoc networks implicitly assume that wireless links are bidirectional. In addition, because of the high overhead needed to handle unidirectional links [11], bidirectional links are preferred. A.3 – *Spanner*: Graph G_{tc} is a spanner if the ratio between the minimum power-cost path in G_{tc} and in G is of a constant order. To study this property, we use the power stretch factor of G_{tc} (which is defined precisely in Section 4). A.4 – *Low degree*: Since a node u may cause interference at nodes in its transmission range, a smaller node degree usually reduces the amount of interference. Hence, it is desirable if each node has a small constant bounded degree. If the degree of the nodes has a constant bound, then the number of G_{tc}'s edges is of the order of the number of its nodes, namely $|E_{tc}| = O(|V|)$. This kind of graph is called a *sparse* graph. The algorithm itself should have the following attributes: B.1 – *Local and fully distributed*: Since ad hoc networks have no centralized authority, any node u independently collects the information of its neighbors (at most two-hop neighbors) in G and then determines the edges of G_{tc} incident in it on the basis of the local information. B.2 – *Low-quality information*: Since obtaining very accurate information such as node locations is usually expensive, the topology control algorithm should rely on 'low-quality' information (e.g. number and identity of neighbor nodes or the distance between nodes).

* Corresponding author. Tel.: +98 21 64542548; fax: +98 21 66497930.
E-mail addresses: m_kadivar@aut.ac.ir (M. Kadivar), shiri@aut.ac.ir (M.E. Shiri), dehghan@aut.ac.ir (M. Dehghan).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.11.014

1.1. Related work

Many topology control algorithms have been designed for ad hoc networks. The topology control algorithms may be classified as centralized and distributed. In the centralized algorithms (such as [1,10]), a particular node is responsible for evaluating an optimum network topology based on global information. These algorithms cannot be used on distributed nodes and, therefore, we do not deal with them in this paper. The other class contains distributed algorithms. *CBTC*(α) [7] is a distributed algorithm in which each node finds the minimum power that ensures it can reach some nodes in every cone of degree α. It has been shown in [7] that if $\alpha \leqslant 5\pi/6$ then connectivity is preserved. The basic idea of the COMPOW [12] and CLUSTERPOW [6] approaches is that each node uses the smallest common power required to maintain connectivity. The major disadvantage of COMPOW is its message overhead, since each node has to exchange link state information with other nodes. Several geometry structures such as Gabriel graph [3], relative neighbor graph [16], Yao graph [21], local minimum spanning tree (LMST) [8], and ESPT [18] have been proposed for topology control in wireless ad hoc networks which are produced in a distributed manner. In some structures such as Gabriel graph, relative neighbor graph, Yao graph, and ESPT, the stretch factor is bounded but the node degree is not bounded. On the other hand, in topologies produced by XTC [20] and LMST, the node degree is bounded but the power stretch factors are not bounded. Still in some others, such as OrdYaoGG and SYaoGG graphs [15], the node degree and the power stretch factor are bounded.

Now we introduce the topology control algorithms XTC and Gabriel graph (GG), and we will compare our contribution with them in the next sections.

1.1.1. RNG- and GG-based algorithms

Let G be a geometrical graph (i.e. the set of G's nodes are placed on the plane). RNG-based algorithms construct a subgraph of G known as relative neighborhood graph (RNG) [16] by removing all edges $(u,v) \in G$ if there exists a node w such that $d(u,v) > d(u,w)$ and $d(u,v) > d(v,w)$, where $d(u,v)$ is the Euclidean distance between u and v. The GG [3] is a special case of RNG, where node w is restricted to the disk with diameter $d(u,v)$.

1.1.2. XTC algorithm

In [20], the XTC topology control algorithm was introduced exhibiting properties A.1 – A.4 and B.1 and B.2. This algorithm operates on a graph $G = (V,E)$ and produces the topology control graph $G_{XTC} = (V,E_{XTC})$. The algorithm consists of three steps: neighbor ordering, neighbor order exchange, and link selection. In the first step, each node u establishes a link meter, denoted by $\|uv\|$, which indicates the quality of the communication link (u,v) between u and its neighbor v. Then u computes an order $\prec_u$ over all its one-hop neighbors in G. All u's neighbors are sorted decreasingly in $\prec_u$ with respect to the link meter values, namely a node w appears before node v in $\prec_u$ if $\|uw\| < \|uv\|$ (this is denoted by $w \prec_u v$). In the second step, node u exchanges its order with all its one-hop neighbors. In the third step, node u selects a node v as its neighbor in graph G_{XTC} if there exists no node w with $w \prec_u v$ and $w \prec_v u$.

2. Proposed algorithm for synchronized nodes

We propose our algorithm, denoted by OTTC algorithm to stand for one- and two-hop topology control algorithm, in this section. Based on this algorithm, as the majority of the existing algorithms, each node receives Hello messages from its one-hop neighbors and then updates its local view when all Hello messages are received. This scheme works correctly if all nodes start at the same time, so we assume that all nodes have synchronized clocks. The OTTC algorithm, similar to the XTC algorithm [20], consists of three main steps: (1) Neighbor detection under a (maximum) normal transmission range and neighbor ordering. (2) Information exchange. (3) Neighbor selection. In the first step, each node u collects its neighbor information through periodic Hello messages and establishes an increasing ordered list $\prec_u$ on its neighbor set based on the quality of the links to the neighbors. If the quality of the links is equal, the lexical order of the node IDs of the link endpoints is taken into account. Node u can use criteria, such as Euclidean distances, packet arrival rate, or received signal strength, to establish its ordered list. In this paper, the quality of each link (u,v) is considered as its weight, which is denoted by $\|uv\|$. We assume that in addition to the order of the nodes, the ordered list $\prec_u$ contains the weight of the edges incident in node u. In the second step, each node broadcasts its ordered list and receives the ordered lists of its neighbors. After exchanging the ordered lists between the nodes, each node can know its two-hop neighbors, in addition to the one-hop neighbors. Therefore, after receiving the ordered list $\prec_v$ from a neighbor v, node u computes its relative two-hop neighbor set, denoted by THN[1](u,v), as

$$THN(u,v) = \{w \mid w \in \prec_v, w \neq u.\}.$$

In the final, node u considers an unprocessed neighbor v of the smallest order in $\prec_u$. Then it checks whether there exists a node w with $w \prec_v u$ and $w \prec_u v$, or whether there exists a node w in $\prec_u$ such that $\|uv\| > \max\{\|uw\|, \|vz\|, \|wz\|\}$ for some z in $THN(u,v) \cap THN(u,w)$. If one of the above conditions is satisfied, then edge (u,v) is discarded and node v is included in the discarded neighbors set $\widetilde{N}(u)$ Otherwise, node v is included in the neighbor set of u, $N(u)$, in the final topology of the network. When all the nodes in $\prec_u$ have been checked, this step terminates. After the execution of the algorithm, the topology control graph $G_{OTTC} = (V,E_{OTTC})$ is produced where E_{OTTC} is the edge set of the graph indicated by $E_{OTTC} = \{(u,v) \mid v \in N(u)\}$. The pseudo-code of OTTC is given in Algorithm 2.1.

In summary, there are two differences between XTC and OTTC. (1) In OTTC we assume that in addition to the order of the nodes, the ordered list $\prec_u$ contains the weights of the edges incident in node u. (2) In XTC, the neighbor selection process indicates that the shortest cycle in G_{XTC} is of length 4, namely it does not contain any cycle such as $u - v_1 - v_2 - u$ through nodes u, v_1, v_2 [16]. Lines 9–13 of the OTTC algorithm indicate that G_{OTTC} does not have cycles such as $u - v_1 - v_2 - u$ and $u - v_1 - v_2 - v_3 - u$. Therefore, the shortest cycle in G_{OTTC} is of length 5.

Algorithm 2.1. OTTC algorithm

```
1: Establish the ordered list $\prec_u$ over $u$'s neighbors in $G$.
2: Broadcast $\prec_u$ to each neighbor in $G$; Receive ordered lists from
   all neighbors.
3: Establish $THN(u,v)$ as the set of two-hop neighbors of $u$ for all
   neighbor
4: nodes $v$
5: $N(u) := \{\}, \widetilde{N}(u) := \{\}$
6: while $\prec_u$ contains unprocessed neighbors do
7: $v :=$ unprocessed neighbors of the smallest order in $\prec_u$;
8: if $(\exists w \in N(u) \cup \widetilde{N}(u) : w \prec_v u)$ or $(\exists w \in N(u) \cup \widetilde{N}(u) : \|uv\| >$
   $\max\{\|uw\|, \|vz\|, \|wz\|\})$ for some $z$ in
9: $THN(u,v) \cap THN(u,w)$ then
```

[1] Two-hop neighbor.