Topology Discovery in Wireless Ad Hoc Networks*

Lei Luo, Adarshpal S. Sethi
Department of Computer and Information Science
University of Delaware
Newark, DE 19716
{*lluo, sethi*}*@cis.udel.edu*

Abstract

Accurate topology information is essential for fault localization and some other applications. In this paper we present an abstract topology discovery algorithm for wireless ad hoc networks. This algorithm constructs up-to-date topological map based on the local neighborhood information reported by each node in the network. We also present some preliminary simulation results, which demonstrate some nice properties of the algorithm.

1 Introduction

Recently wireless ad hoc networks have drawn much attention from military and industry. As the underlying technologies mature, more and more wireless ad hoc networks will be deployed in the future. For instance, it has been proposed to deploy wireless ad hoc networks in the future battlefields to provide better communications [8]. For those networks to work reliably and robustly, it is vital to quickly and accurately detect and diagnose their faults. This process is called fault localization (also referred to as root cause analysis). In this process, discovering the network topology plays an important role [25, 23, 22]. For example, in a battlefield ad hoc network, equipments may operate under some extreme conditions and may inevitably experience some failures. Given the various symptoms reported by different devices, it is critical to analyze the root causes of the failures (fault localization) and

*Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

recover from the failures (self-healing). The fault detection algorithm, in turn, relies on the network topology information to eliminate false alarms and localize the failures [12, 13, 14].

During the past decade, considerable research effort has been devoted to automatically discovering network topologies and many algorithms have been proposed [1, 2, 9, 10, 15, 19]. In general, these algorithms can be classified into two broad categories. One category focuses on wide-area networks like the Internet and discovers the logical topology on the IP router level, ignoring lower layer devices such as hubs and switches [9, 10, 19]. The other tries to discover the physical topology (ISO layer-2). Ethernet or local-area networks are usually their research targets [1, 2, 15]. Depending on the problem being addressed, some algorithms use SNMP queries to interrogate the network devices (routers, switches, etc) for their *ipRouteTable* and *ipAddrTable* contents to infer the topology [2, 15], whereas others employ traceroute, ping, DNS zone transfer, or active probing to achieve the same purpose [1, 9, 10, 19]. In this paper, we focus on the topology discovery schemes applicable to wireless ad hoc networks for fault management applications.

The goal of this paper is to present an abstract centralized topology discovery algorithm and, more importantly, to study through simulation the effects imposed by the underlying schemes (such as neighborhood detection) on the performance of the topology discovery algorithm. In this research we hope to accumulate knowledge and experience for future development of an efficient topology discovery algorithm (centralized or distributed) for wireless ad hoc networks. The algorithm studied here was designed to be centralized because most existing fault localization algorithms work in a centralized fashion [23, 25, 24].

The remainder of this paper is organized as follows. Section 2 discusses some related work in discovering network topologies. Section 3 describes the proposed topology discovery algorithm for wireless ad hoc networks. We present simulation results in Section 4 and conclude in Section 5.

2 Related Work

The research to discover the network topology starts from the effort to map the Internet and to discover both intra-domain and Internet backbone topologies on the IP level. Siamwalla *et al.* [19] proposed several heuristics and algorithms to infer the logical topology of a single administrative domain. They also proposed an algorithm to discover the Internet backbone topology using traceroute. However, rather than get a topological map, the traceroute approach only discovers a spanning tree rooted at the traceroute source. Cross-links — interconnections among other nodes — are ignored.

Similar to [19], the Mercator project [9] uses hop-limited probes to infer the router adjacency on the Internet from a single, arbitrary location. To handle the "cross link" issue, Mercator uses source-routing to get more adjacency information. The CAIDA project [10] uses a topology measurement tool, *skitter*, to capture the Internet's topological map. Each *skitter* monitor actively probes the routes to a list of destinations by using traceroute. The gathered topology data are fed to a central repository.

Besides mapping the network topology on the IP layer, researchers also proposed algorithms to discover the physical topology. Most such algorithms [2, 15] discover the physical topology based on SNMP information obtained from network devices, while one approach [1] uses active probing to achieve the goal.

The above schemes were mainly designed for wireline networks, and may not be easily applied to wireless networks. So let us focus on topology discovery schemes proposed for wireless networks.

Chandra *et al*. [4, 5] proposed a topology discovery algorithm for hybrid wireless networks. This algorithm aims at home and office networks. In such a network, all desktops are connected to the wireline network. Mobile devices use the wireless network to communicate with each other and access the Internet. The algorithm runs in two steps. In the "Diffusion" step, a node called "coordinator" floods the network with a topology request message using reliable broadcast. All the other nodes record the first k nodes from which they receive the request message as their parent nodes and then rebroadcast the message. During this phase, all nodes will learn their immediate neighbors by listening to the broadcast. Later, in the "Gathering" step, every node assembles its neighbor list, together with the topology information reported by its children, in a response message which it sends to its parents. The aggregate response message will eventually reach the coordinator, which then has complete knowledge about the network topology. The advantage of this scheme is that each node combines the topology information about its subtree so that only a limited number of response messages can finally reach the coordinator, thus mitigating the congestion in the vicinity of the coordinator. However, this scheme solely relies on the flooding to detect the neighborhood. If the topology changes after the flooding, the coordinator cannot keep track of the up-to-date topology unless it periodically initiates flooding. Due to the high overhead of flooding a network, this scheme is not suitable for tracking the topology change in a wireless ad hoc network.

RoyChoudhury *et al.* [17, 18] presented a topology discovery scheme for wireless ad hoc networks. This scheme employs multiple mobile agents. A mobile agent "hops from node to node, collects information from the nodes, meets other agents in its journey, and interacts with both to collect updates of parts of the network that they have not visited recently". The authors have chosen the

critical agent population to be half the number of nodes in the system. The idea here is novel but the overhead to move the agents around the network might be a problem. In addition, from the fault management's point of view, the time for the centralized fault manager (running on one node) to learn the topology change might be long and potentially unbounded.

Other than the specifically designed topology discovery algorithms, some routing algorithms can also provide topology information. For example, in a network running OSPF (Open Shortest Path First), each router periodically advertises its link state information. Using the collected link state advertisements (LSAs), each router can build its own topological map of the network. However, OSPF cannot be applied to wireless ad hoc network directly due to limited channel bandwidth.

To meet ad hoc networks' bandwidth constraint, researchers have proposed several routing algorithms optimized for wireless ad hoc networks. These algorithms can be classified as either reactive or proactive. Reactive algorithms such as AODV [16] and DSR [11] discover a path to a host only when a packet is to be sent to that host. A node only collects the local topology information when it is along an active path. On the contrary, proactive routing protocols like OLSR [7], GSR [6], and OSPF extension for MANET [3] maintain routes to all nodes, no matter if there is a packet to be sent or not.

GSR is an early attempt to migrate link state routing to ad hoc networks. Each node floods the link state information into the whole network once it detects a change on a link between its neighbors and itself. Therefore a node will know the whole topology when it obtains all link information. The scheme works well in static networks with reliable links. When the links change quickly, frequent global flooding will inevitably introduce huge overhead.

OLSR handles the overhead issue by selecting Multipoint Relays (MPRs) for each node. Each node periodically advertises its MPR selector list, and only MPRs for that node will forward this advertisement. However, since the advertisement only contains part of the node's neighbor list, the topology information collected at any node is not complete.

OSPF extension for MANET is another approach and is based on OSPF for IPv6 networks. Similar to OLSR, the extension selects a set of overlapping relays for each node. Each node advertises all of its locally connected neighbors but only neighbors in the active overlapping relay set will immediately relays this advertisement (just like an MPR in OLSR). A non-active overlapping relay will waits for a specified amount of time. During this time, if the neighbor determines that its flooding is redundant or it hears a re-flooding from another node, it suppresses its transmission. Otherwise, it transmits upon expiration of the time. This protocol differs from OLSR as it floods the complete neighbor list. Therefore the control overhead would be higher than OLSR, especially in a dense ad hoc network.

3 Topology Discovery for Wireless Networks

3.1 Why not Traceroute

The traceroute [26] program, written by Van Jacobson, is a network debugging tool that explores the route between a source and a destination. In short, a source keeps sending packets to a destination with progressively incrementing TTL, starting from 1. Every router seeing this packet decrements the TTL value and forwards the packet if the TTL is greater than 1; otherwise, the packet is discarded and the router sends back an ICMP "time expired" message. In this way, the sender can discover all the routers along the path. The problem with this scheme is the large number of packets generated. Suppose the source and destination are *k* hops away. Then a total of *2k* packets will be used to discover the path.

Another problem is that a node can only discover a tree spanned over the network. Cross-links are missed. To discover the full topology, all nodes have to report their trees to a central repository. Suppose the total number of nodes is *n* and the average distance between any two nodes is *d* hops. Then the total number of packets generated would be $n \times (n-1) \times 2d \approx n^2 \times 2d$. For a network with scarce channel bandwidth, the overhead and delay are prohibitive.

3.2 Topology Discovery Scheme for Static Ad Hoc Networks

In the basic topology discovery algorithm for static ad hoc networks, each node collects its neighbor list by sending HELLO messages to its neighbors (this step is skipped if the list is available from the routing protocol). Each node then periodically reports its local topology information to the central topology manager, who maintains the topological map. The topology manager can either poll every node for its neighbor list, or can wait for each node to report. We chose the latter approach since it incurs less overhead than the former one.

This scheme is based on two assumptions. First, all nodes within the network know the address of the topology manager. This could be accomplished either by some offline communications or by flooding/announcement during the initialization phase. Second, we assume all nodes within the network share the same network prefix, i.e., there is only one single subnet. Multi-subnet environment has not been tested.

In this abstract scheme we also assume that only two common MANET routing protocols, AODV and OLSR, will be used. The latter maintains full neighbor list, whereas the former usually does not. Therefore we have to distinguish these two scenarios. The pseudo code is shown in Figure 1.

The scheme employs two types of messages, HELLO and REPORT. The HELLO messages (sim-

```
TOPOLOGY_DISCOVERY(u)
{   /* Initialization Phase */
    if (OLSR is running)
        routing = ROUTING_OLSR
    else{
        routing = ROUTING_OTHER
        Start the HELLO timer }
    if (u is topology manager)
        Initialize adjacency matrix
    Start the REPORT timer
    /* Topology Discovery Phase */
    NL = ∅
    do {
        if (u receives HELLO message from v){
            NL = NL ⋃ v
            if (u is listed in HELLO message)
                mark v as SYM
            else mark v as ASYM
        }
        if (HELLO timer goes off & routing ≠ ROUTING_OLSR){
            Construct a HELLO message
            for each v ∈ NL
                insert v into HELLO
            Broadcast the HELLO message
            Restart the HELLO timer
        }
        if (REPORT timer goes off){
            Construct a REPORT message
            if (routing = ROUTING_OLSR)
                NL = OLSR neighbor table
            for each v ∈ NL
                if (v is of type SYM) insert v into REPORT
            Send REPORT to topology manager
            Restart the REPORT timer
        }
        if (u receives REPORT from v & u is topology manager)
            Update the adjacency matrix
    }while (TRUE)
}
```

Figure 1: Topology Discovery for Static Ad Hoc Networks

ilar to the HELLO messages in OLSR) are used for neighbor detection and link sensing, and the REPORT messages report the local topology information to the central topology manager. The scheme is divided into two phases: initialization and topology discovery. During the initialization phase, the system is queried for the routing protocol being used — if AODV is used, the HELLO timer is started. Since the neighbor list needs to be reported periodically, the REPORT timer is also launched.

When the HELLO timer expires, a HELLO message is broadcast. A node can learn 1-hop neighbors by listening to the HELLO messages. When the REPORT timer times out, each node reports its 1-hop neighbor list obtained either from the routing protocol or from its own neighbor table.

Due to congestion, some HELLO and REPORT messages may be lost, thus the detected connec-

tivity information may be incomplete. In this basic scheme, we assume congestion is the only cause of this incompleteness and no faults occur on devices. Therefore the topology manager can build up (or accumulate) its knowledge about the network topology steadily.

3.3 Topology Discovery Scheme for Mobile Ad Hoc Networks

In this section, we extend the basic scheme to mobile ad hoc networks, in which nodes can move randomly. This mobility brings the problem that a detected link may be stale after a while. So accumulating the topology information is not suitable any more. However, the historical connectivity information may still be valuable since, given the radio transmission range and the speed of mobility, a one-hop connection may stay valid for a while as long as the two nodes are still in each other's transmission range. Therefore the algorithm can still benefit from the historical data. The key points are how long we should keep the historical information as valid reference and how often each node should detect and report its neighbors. Obviously, the more frequently the nodes detect and report its neighbors, the higher the overhead, but also the higher the accuracy will be. So we have to make some trade-off among latency, accuracy, and overhead. In this scheme, we leave the frequency for detection and reporting as a parameter so that it can be appropriately adjusted by the manager.

The pseudo code is shown in Figure 2. Different from the basic scheme, this scheme maintains a time matrix at the topology manager, which records the last time the corresponding links are updated. Whenever a REPORT is received, the current time is recorded in the time matrix for the links reported. From time to time, this matrix is examined. If an entry has not been updated for a certain period of time, the corresponding entry of the adjacency matrix is flushed. This approach is based on the assumption that HELLO and REPORT messages can be lost, but the losses are random and inconsistent. Thus if a connection is transiently lost due to congestion, it may be recovered shortly thereafter. But if a node moves out of the range of another or a fault occurs, the loss will not be recovered.

4 Simulations and Analysis

We implemented the topology discovery algorithms with QualNet^{TM} [20], a discrete-event simulator for wireless and wired networks. In the simulations, 30 nodes are uniformly randomly distributed within a 1500x1500 meter area. All simulations are repeated for AODV and OLSR respectively. The periods to send HELLO and REPORT packets are 2 seconds and 15 seconds, as suggested in [7] for HELLO_INTERVAL and TOP_HOLD_TIME.

```
TOPOLOGY_DISCOVERY(u)
{   /* Initialization Phase */
    if (OLSR is running) {routing = ROUTING_OLSR}
    else{
        routing = ROUTING_OTHER
        Start the HELLO timer }
    if (u is topology manager){
        Initialize Adjacency matrix and Time matrix
        Start the FLUSH timer }
    Start the REPORT timer
    /* Topology Discovery Phase */
    NL = ∅
    do {
        if (u receives HELLO message from v){
            NL = NL ⋃ v
            if (u is listed in HELLO message) {Mark v as SYM}
            else Mark v as ASYM
        }
        if (HELLO timer exipres & routing ≠ ROUTING_OLSR){
            Construct a HELLO message
            for each v ∈ NL
                Insert v into HELLO
            Broadcast the HELLO message
            Restart the HELLO timer
        }
        if (REPORT timer goes off){
            Construct a REPORT message
            if (routing = ROUTING_OLSR) {NL = OLSR neighbor table}
            for each v ∈ NL
                if (v is of type SYM) {Insert v into REPORT}
            Send REPORT to topology manager
            Restart the REPORT timer
        }
        if (FLUSH timer goes off){
            for each entry t in Time matrix
                if (t is old enough)
                    Clear corresponding entry in Adjacency matrix
            Restart the FLUSH timer
        }
        if (u receives REPORT from v & u is topology manager)
            Update the Adjacency matrix and Timer matrix
    }while (TRUE)
}
```

Figure 2: Topology Discovery Algorithm for Mobile Ad Hoc Networks

Currently, we only considered static ad hoc networks since QualNet cannot provide actual topology during the simulation to compare with the topology discovered by the algorithm. Therefore we leave the simulations for mobile ad hoc networks for future work. On the application layer, we ran CBR or FTP applications with various number of data flows to see how the topology discovery scheme affects application traffic. We also evaluate the performance with and without the topology discovery scheme.

To minimize the effect of statistical fluctuation, for each traffic pattern (number of flows, transmission rate, etc.) , we repeated the simulation 100 times with randomly chosen seeds. For each seed, QualNet generates a unique topology. All the results shown in the graphs of this section are displayed with 90% confidence intervals for mean values computed over the 100 replications.

4.1 Evaluation Metrics

We evaluate the performance of the scheme in terms of four metrics: *overhead*, *throughput loss*, *completeness*, and *speed*. We measure the mean relative *throughput loss* by the throughput loss ratio, i.e., the fraction of absolute throughput loss (due to topology discovery process) over the overall throughput. We measure the *completeness* by the percentage of the topology discovered by the algorithm. To measure the *speed* of the algorithm, we repeatedly measure the completeness to see how soon it reaches a given threshold, say 95%.

The measurement of overhead is somewhat trickier. In a wireless network, sending one application packet will incur several accompanying packets, such as RTS, CTS, RREQ, and RREP, etc. So we cannot just count of number of topology discovering packets as overhead. Furthermore, these packets all contend for the channel. So when the topology discovery process is running, it is likely that fewer application packets be delivered to the destination, and thus fewer MAC layer packets be transmitted to convey the application data. Since we cannot distinguish the MAC layer packets transmitted for conveying application data from those for conveying control packets, it does not make much sense to count the amount of extra MAC layer packets as overhead. Therefore, we measure the overhead as:

$$overhead = \frac{MAC(w)}{Data(w)} - \frac{MAC(wo)}{Data(wo)} \quad (1)$$

- ***MAC(w)*** and ***MAC(wo)*** are the total MAC layer packets transmitted with and without the topology discovery process running, respectively.
- ***Data(w)*** and ***Data(wo)*** are the total data bytes delivered with and without the topology discovery process running, respectively.

Here we calculate the amount of MAC layer packets transmitted to convey a given amount of application data and measure the difference with and without the topology discovery process as overhead.

4.2 Simulation Results

4.2.1 Throughput Loss

Figures 3 and 4 show the throughput loss ratio with different number of CBR and FTP flows when AODV or OLSR is used, respectively. For a given network size, the throughput loss ratio decreases as the number of flows increases. The only exception is the FTP curve with OLSR, which has some negative values. The actual relative throughput loss in these cases is likely very close to zero, and the small negative values are within the confidence interval for these points.

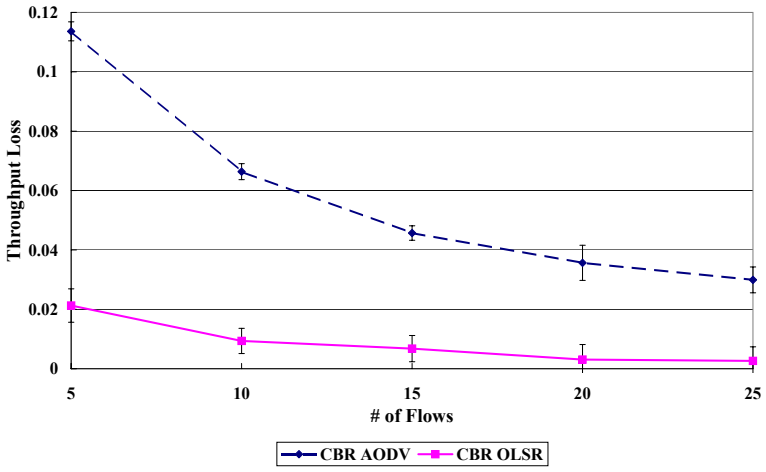

Figure 3: Throughput Loss Ratio for CBR Traffic

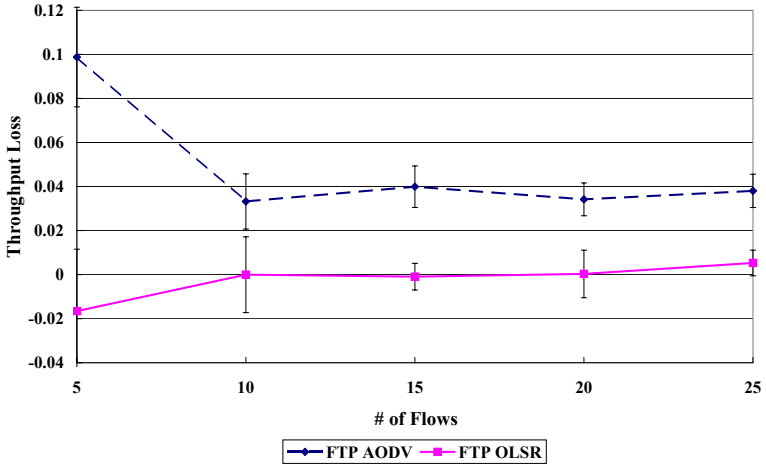

Figure 4: Throughput Loss Ratio for FTP Traffic

Our explanation for the decreasing throughput loss ratio is that, as the number of flows increases, the overall CBR and FTP throughputs increase accordingly, as shown in Figures 5 and 6. Meanwhile, the absolute throughput losses do not change much — the absolute throughput loss curves tend to be flat in Figure 7 and 8. These figures thus demonstrate a good property of the topology discovery scheme: the absolute throughput loss incurred by the topology discovery process is limited and is not significantly affected by the number of flows (at least up to certain number of flows). Note that the figures for absolute throughput losses are drawn under the same scale as those for throughputs, so we can see how little the absolute throughput losses are compared with the overall throughputs.

It should also be noticed that the relative throughput loss incurred by the topology discovery process is lower when OLSR is used than when AODV is used, which should be a direct consequence of

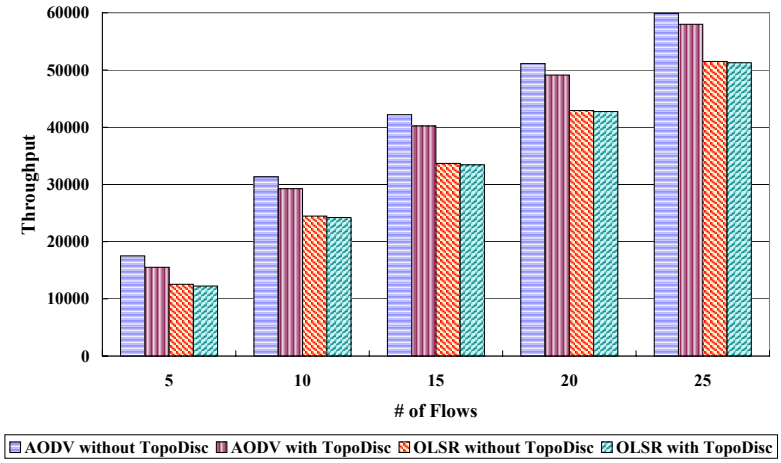

Figure 5: Throughput for CBR traffic

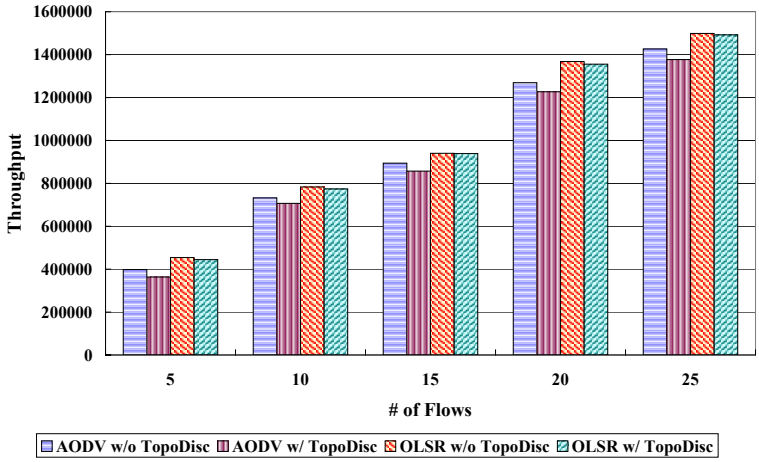

Figure 6: Throughput for FTP Traffic

the fact that less packets are transmitted for neighbor detection with OLSR. The relative throughput loss with OLSR is very close to zero.

4.2.2 Completeness and Speed

Figures 9 through 12 show the completeness of the collected topology information. We call the period with which each node reports its neighbor list a **flood**. Just as expected, the curves start from relatively low completeness levels since some REPORT packets might be lost due to congestion. As more and more neighborhood information is accumulated, the completeness approaches 100%. As a supplement, Figure 13 shows the time by when the topology manager has received REPORT messages from every node. It shows, for instance, when there are 5 CBR flows, it takes around 60 seconds to receive REPORTs from every node when AODV is used. Recall that the period of reporting is 15

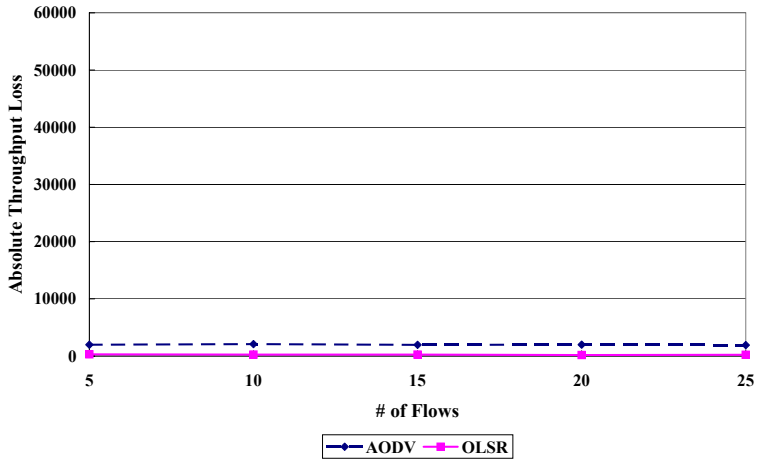

Figure 7: Absolute Throughput Loss for CBR traffic

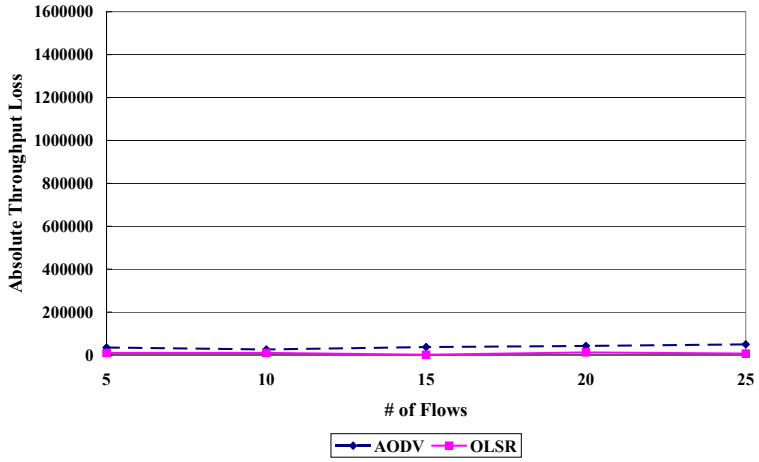

Figure 8: Absolute Throughput Loss for FTP Traffic

seconds in the simulation. That means the topology manager has to wait for at least four floods to hear from all nodes. However, at this time, the completeness is only around 95%. The reason is that some HELLO messages are dropped due to congestion and the corresponding neighborhood information is not complete in the REPORT.

One interesting fact is that, for each curve in the completeness figure, we can identify one critical point. Below this point, the curve ascends quickly, gaining 85% to 90% completeness within only a few periods, whereas above this point, the curve tends to be flat, slowly approaching 100%. The reason behind this is that, during the startup phase, each received REPORT packet will greatly enhance the manager's knowledge about the topology. But later, if most of the topological map has already been constructed, the received REPORT packet might just contain redundant information. Thus little improvement on the topological map will be made. This fact might be employed by certain applications

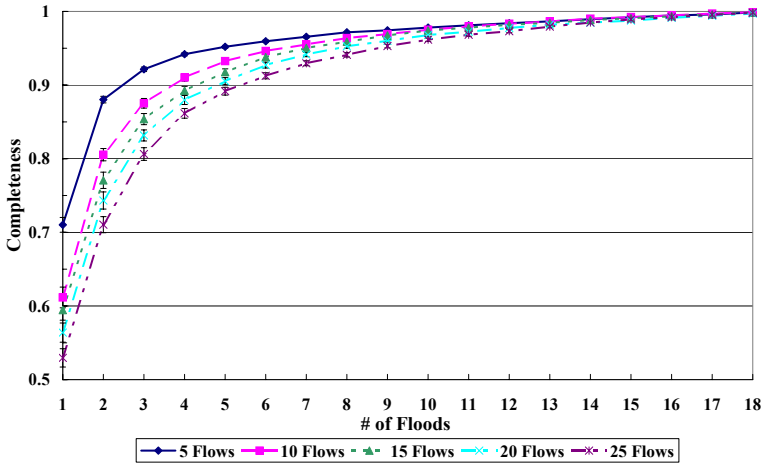

Figure 9: Completeness for CBR Traffic (AODV)

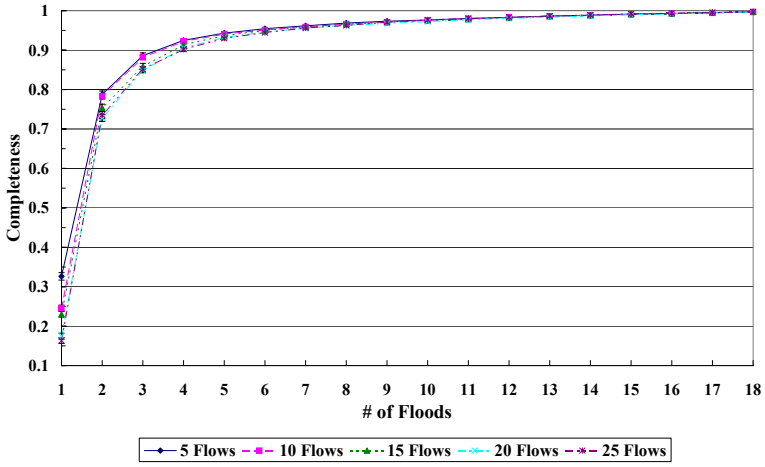

Figure 10: Completeness for CBR Traffic (OLSR)

that do not require 100% complete topology information to start functioning correctly, thus probably reducing their initialization time. One such application is fault detection and localization, where some localization algorithms have been shown to work reasonably well even when the underlying topology information is incomplete or inaccurate [21].

Figure 9 through 12 also show the speed with which the algorithm discovers the ad hoc network topology. Apparently, as the number of CBR flows increases, it takes longer to discover a large portion (say, 95%) of the topology since more HELLO and REPORT messages might be dropped due to congestion. But for FTP traffic, this may not always be true. Figure 12 shows that when OLSR is used, it may take longer to discover the topology when there are 10 flows than when there are 25 flows. In addition, the figures show that the algorithm constructs the topological map a little bit faster when OLSR is used than when AODV is used.

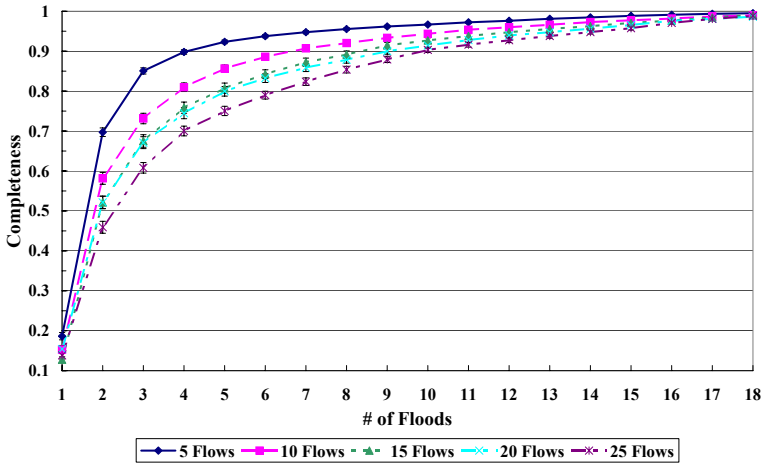

Figure 11: Completeness for FTP Traffic (AODV)

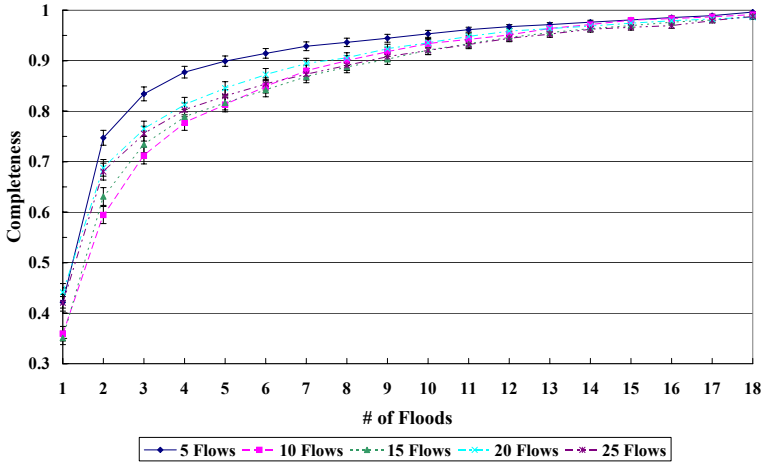

Figure 12: Completeness for FTP Traffic (OLSR)

4.2.3 Overhead

Figure 14 shows the overhead induced by the topology discovery process for CBR and FTP traffic. As seen in the figure, the mean overhead decreases monotonously as the number of flows increases. This may appear to be non-intuitive. So let us examine this issue further in the following analysis.

The overhead for a particular scenario is defined in equation 1 as:

$$overhead = \frac{MAC(w)}{Data(w)} - \frac{MAC(wo)}{Data(wo)}$$

Let us denote the decrease in data packets delivered with and without topology discovery algorithm by Δ, and the increase in MAC layer packets with and without topology discovery by $f(\Delta)$:

$$\Delta = Data(wo) - Data(w) \tag{2}$$

$$f(\Delta) = MAC(w) - MAC(wo) \tag{3}$$

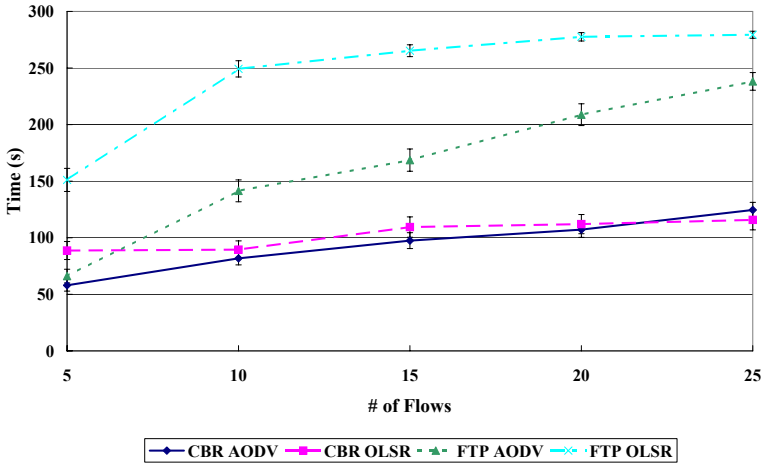

Figure 13: The Time to Receive Reports from Every Node

Then the expression for overhead can be written as:

$$
\begin{aligned}
overhead &= \frac{MAC(w)}{Data(w)} - \frac{MAC(wo)}{Data(wo)} \cdot \frac{Data(wo) - \Delta}{Data(w)} \\
&= \frac{MAC(w)}{Data(w)} - \frac{MAC(wo)}{Data(w)} + \frac{\Delta \cdot MAC(wo)}{Data(w) \cdot Data(wo)} \\
&= \frac{MAC(w) - MAC(wo)}{Data(w)} + \frac{MAC(wo)}{Data(wo)} \cdot \frac{\Delta}{Data(w)} \\
&= \frac{f(\Delta)}{Data(w)} + \frac{MAC(wo)}{Data(wo)} \cdot \frac{\Delta}{Data(w)} \qquad (4)
\end{aligned}
$$

Figure 15 shows $Data(wo)$ and $Data(w)$ for CBR traffic as well as their difference, $\Delta = Data(wo) - Data(w)$, the decrease in data packets due to the topology discovery process. The corresponding values for FTP traffic are shown in Figure 16. These two figures show that, as the number of flows increases (and so does the network traffic), both $Data(w)$ and $Data(wo)$ increase, but their difference does not change much. Compared with the amount of data delivered, the value of Δ is small and is monotonously decreasing. Thus, for a large number of flows the ratio $\frac{\Delta}{Data(w)}$ decreases.

The quantity $f(\Delta)$ represents the amount of extra MAC layer packets transmitted due to the topology discovery process. Figure 17 shows how the first and second terms in equation 4 vary as the number of flows increases. It is clear that, compared with the amount of data transmitted, the ratio of $\frac{f(\Delta)}{Data(w)}$, i.e., the first term in equation 4, decays rapidly . So we can infer that the increase of extra MAC layer packets transmitted due to the topology discovery process is limited compared with the total data traffic. Similarly, the second term in equation 4 also slowly decreases, thus making the total overhead decrease.

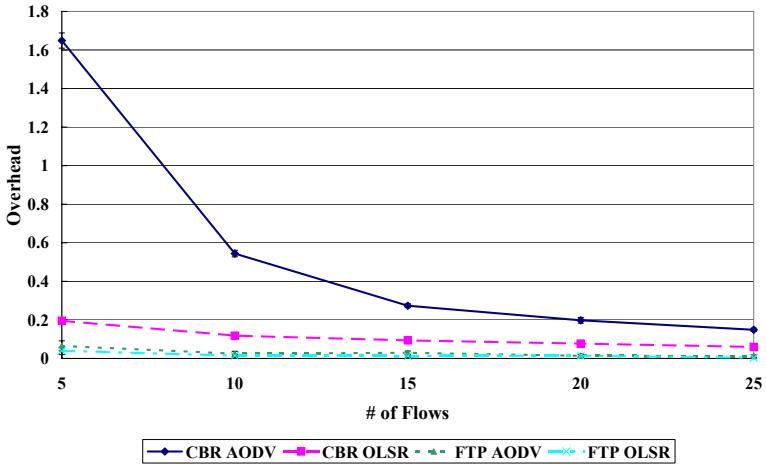

Figure 14: Overhead

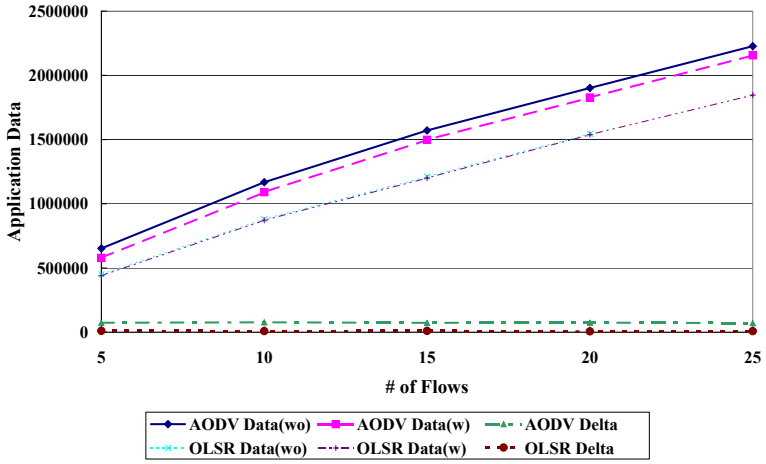

Figure 15: CBR Data

5 Conclusion

Topology information is critical for fault localization and some other applications such as network simulations, and topology-aware applications. In this paper, we presented an abstract topology discovery algorithm for wireless ad hoc networks and some preliminary simulation results. However, our focus here is not just on the algorithm itself, but rather on the knowledge and experience accumulated through the research. The simulation results demonstrate that, for a small-scale static ad hoc network with low-to-moderate traffic load, the algorithm can discover the network topology with acceptable completeness in a reasonably short time. They also demonstrate that the overhead introduced by this scheme is quite small. In addition, as the network traffic increases, the overhead decreases.

Furthermore, we extend the abstract topology discovery algorithm to mobile ad hoc networks, but limitations of the simulation environment have impeded our ability to study its performance in terms

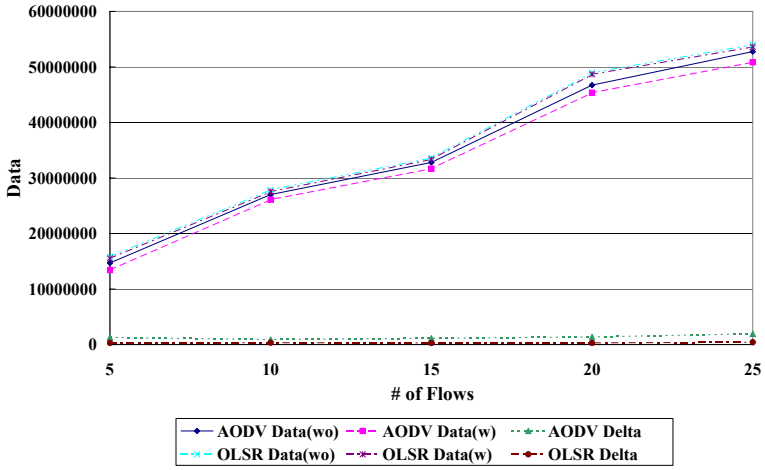

Figure 16: FTP Data

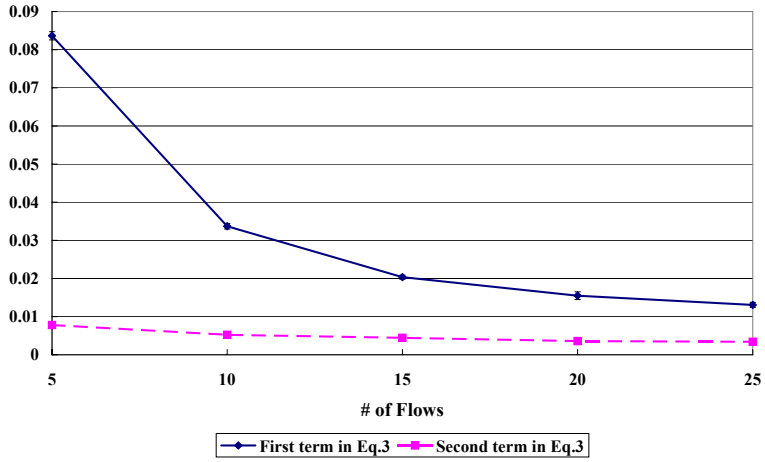

Figure 17: Analysis of Overhead Components for CBR Traffic

of its accuracy and speed. This is a difficult issue for which we are in the process of developing some solutions and which will be addressed in future work.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government.

References

[1] R. Black, A. Donnelly, and C. Fournet. Ethernet Topology Discovery without Network Assistance. In *Proc. of the 12th IEEE International Conference on Network Protocols(ICNP)*, Berlin, Germany, Oct 2004.

[2] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschats. Topology Discovery in Heterogeneous IP Networks. In *Proc. of IEEE INFOCOM 2000*, pages 265–274, Tel Aviv, Israel, 2000.

[3] M. Chandra. *Extensions to OSPF to Support Mobile Ad Hoc Networking, Internet-Draft*, October 2004.

[4] R. Chandra, C. Fetzer, and K. Hogstedt. Adaptive Topology Discovery in Hybrid Wireless Networks. In *Proc. of first International Conference on Ad-Hoc Networks and Wireless*, Toronto, Canada, Sept 2002.

[5] R. Chandra, C. Fetzer, and K. Hogstedt. A Mesh-based Robust Topology Discovery for Hybrid Wireless Networks. In *AT&T Technical Report*, 2002.

[6] T.-W. Chen and M. Gerla. Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks. pages 171–175, June 1998.

[7] T. Clausen and P. Jacquet. *Optimized Link State Routing Protocol (OLSR), RFC 3626*, October 2003.

[8] http://www.arl.army.mil/alliances/cnnoview.htm.

[9] R. Govindan and H. Tangmunarunkit. Heuristics for Internet Map Discovery. In *Proceedings of IEEE INFOCOM 2000*, pages 1371–1380, Tel Aviv, Israel, 2000.

[10] B. Huffak, D. Plummer, D. Moore, and K. Claffy. Topology Discovery by Active Probing. In *Proc. of the 2002 Symposium on Applications and the Internet (SAINT) Workshops*, page 90, Nara City, Japan, Feb 2002.

[11] D. B. Johnson, D. A. Maltz, and Y.-C. Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR) (Draft)*, 2004.

[12] L. Kant, W. Chen, C.-W. Lee, A. S. Sethi, M. Natu, L. Luo, and C.-C. Shen. D-FLASH: Dynamic Fault Localization and Self-Healing for Battlefield Networks. In *Proc. 24th Army Science Conference*, Orlando, FL, Dec. 2004. To appear.

[13] L. Kant, A. McAuley, R. Morera, A. S. Sethi, and M. Steinder. Fault Localization and Self-Healing with Dynamic Domain Configuration. In *Proc. Milcom-2003, IEEE Military Communications Conference*, Boston, MA, Oct. 2003.

[14] L. Kant, A. S. Sethi, and M. Steinder. Fault Localization and Self-Healing Mechanisms for FCS Networks. In *Proc. 23rd Army Science Conference*, Orlando, FL, Dec. 2002.

[15] B. Lowekamp, D. R. O'Hallaron, and T. R. Gross. Topology Discovery for Large Ethernet Networks. In *Proceedings of ACM SIGCOMM 2001*, pages 237–248, San Diego, CA, Aug 2001.

[16] C. E. Perkins, E. M. Belding-Royer, and S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing, RFC 3561*, July 2003.

[17] R. RoyChoudhury, S. Bandyopadhyay, and K. Paul. A Distributed Mechanism for Topology Discovery in Ad Hoc Wireless Networks Using Mobile Agents. In *MobiHoc 2000*, Boston, MA, August 2000.

[18] R. RoyChoudhury, S. Bandyopadhyay, and K. Paul. Topology Discovery in Ad Hoc Wireless Networks Using Mobile Agents. In *Proc. of the Second International Workshop on Mobile Agents for Telecommunication Applications*, Paris, France, September 2000.

[19] R.Siamwalla, R.Sharma, and S.Keshav. Discovering Internet Topology. http://www.cs.cornell.edu/skeshav/papers/discovery.pdf, July 1998.

[20] http://www.qualnet.com. Scalable Network Technologies (SNT).

[21] M. Steinder and A. S. Sethi. Increasing Robustness of Fault Localization Through Analysis of Lost, Spurious, and Positive Symptoms. In *Proc. of IEEE INFOCOM*, New York, NY, June 2002.

[22] M. Steinder and A. S. Sethi. Non-deterministic Fault Localization in Communication Systems Using Belief Networks. *IEEE/ACM Transactions on Networking*, 2004. To appear.

[23] M. Steinder and A. S. Sethi. Probabilistic Fault Diagnosis in Communication Systems Through Incremental Hypothesis Updating. *Computer Networks*, 45(4):537–562, July 2004.

[24] M. Steinder and A. S. Sethi. Probabilistic Fault Localization in Communication Systems Using Belief Networks. *IEEE/ACM Transactions on Networking*, 12(5):809–822, Oct 2004.

[25] M. Steinder and A. S. Sethi. A survey of fault localization techniques in computer networks. *Science of Computer Programming, Special Edition on Topics in System Administration*, 53:165–194, nov 2004.

[26] W. R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocol*. Addison Wesley, 1994.