Title	ネットワークトポロジー発見とそのアプリケーション
Author(s)	田，慧
Citation	
Issue Date	2006-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/972
Rights	
Description	Supervisor:Hong Shen，情報科学研究科，博士

Network Topology Discovery and Its Applications

by

HUI TIAN

Submitted to

Japan Advanced Institute of Science and Technology
In partial fulfillment of the requirements
For the degree of
Doctor of Philosophy

Supervisor: Professor Hong Shen

School of Information Science

Japan Advanced Institute of Science and Technology

March, 2006

Abstract

The knowledge of network topology is vital for various network management tasks and applications in different types of networks such as routing, flow control, traffic shaping, resource scheduling, performance evaluation and optimization. Network topology discovery has thus become a critical research area of increasing significance in both theory and applications. This area attracts numerous researchers to study various methods and techniques for effective discovery of network topology. This thesis is devoted to this important research area, concentrated on the topic of network topology discovery and its applications in the following three aspects covering both wired and wireless networks:

- Topology discovery for multicast network and its applications in performance evaluation.
- Mobile agent-based topology discovery and performance analysis.
- Topology analysis in wireless sensor network and its applications in routing.

For multicast network topology discovery, we apply multicast-based network tomography to infer the network topology and internal loss/delay performance because of its lower traffic burden and higher efficiency than other methods. Different from previous work, we propose Binary Loss Tree Classification with Hop count (HBLT) and Binary Hamming distance Classification (BHC) algorithms which takes hop count and hamming distance of sequences on receipt/loss of probe packets maintained at each pair of nodes into account respectively. The use of level information and hamming distance classification approach brings different benefits to topology inference. The HBLT algorithm that takes level information into account improves greatly in efficiency of inference procedure by grouping receivers according to hop count as initial steps. Each node in the network records its hop count information which is proved greatly helpful for topology inference by both theoretically analysis and simulation results. The BHC algorithm applies the hamming distance approach for siblings classification and hop count information. The hamming distance based siblings classification approach gains more benefits to topology inference, and thus enables BHC to achieve a better performance for topology discovery than all previous approaches based on the well-known technique of traditional A-approach in siblings classifications.

Based on discovered topology, we further propose network-internal loss/delay performance inference schemes. We propose a novel method to infer internal links' loss rates which significantly improves the efficiency of loss performance inference than the previous methods. We also present a hamming distance matrix-based loss/delay performance inference approach by employing end-to-end loss/delay measurements. The accuracy and efficiency of these schemes are proved by detailed theoretical analysis and validated by simulation results.

As a development in new approaches for topology discovery, we apply mobile agents technology in topology discovery and build statistical models to study its performance. We propose several mechanisms for both Internet and multicast network topology discovery, including the report-at-newly-found-nodes (RN) algorithm and report-at-leaf-nodes (RL) algorithm. Through analysis on the behavior of mobile agents with different report fashion, we study the performance of different mechanisms for both Internet and multicast network topology discovery and verify their feasibility in simulated networks. Generally speaking, the RN algorithm reports more frequently than the RL algorithm and thus brings heavier burden to the management station while the RL algorithm results in less burden to the management station. Thus RL outperforms RN in efficiency, while RN outperforms RL in the system reliability. By analysis and simulation, it is shown that, in mobile agent systems, topology discovery can be performed correctly and efficiently due to inherent advantages of mobile agents.

We further extend our research to wireless sensor networks (WSNs) and address coverage, connectivity, reliability and energy-efficiency, which are the most important issues in WSNs, from the topology point of view. Observing that topology control is more meaningful than topology discovery in WSNs because neighboring information is usually enough to support various applications of WSNs in practice, we study energy-efficient topologies in which deployed sensor nodes can cover the required area and guarantee their connection as well. We show that triangle-based, square-based, hexagon-based and strip-based topologies built with the same number of sensor nodes can meet different requirements in reliability and coverage. We further propose two routing schemes to achieve desired energy-efficiency. Our first protocol is developed by considering different requirements for energy consumption and transmission delay, and incorporating different route selection functions by combining the length of route and the number of streams at individual nodes. This strategy requires only neighboring information and has clear advantages for achieving different performance goals. Our second protocol employs the random walk technique for routing in WSNs with patterned topologies and shows performance improvements for small-size data transmission. This protocol achieves high successful transmission rate within a limited number of steps which is quantitatively analyzed for the first time to our knowledge, thus improving in energy-efficiency over other protocols. The performance of the random walk routing in energy-efficiency is comparative to that of the shortest path routing, while load balancing in the former cannot be achieved by the latter.

keywords: Algorithm, mobile agent, multicast, performance inference, routing protocol, topology discovery, wireless sensor network.

Acknowledgements

I take immense pleasure in acknowledging various people and organizations that have supported me in different ways. To all of them, I would like to convey my heartfelt gratitude.

First, I am deeply indebted to my supervisor, Professor Hong Shen, without whom this thesis would not be possible. His effective guidance, constructive comments and detailed feedback have helped me achieve my goal.

When I started my research, I often committed at least two mistakes. Sometimes I developed confused ideas without deep thinking. Prof. Shen's ability to see clearly through my confusion and find the potentially interesting elements hidden there has well enlightened me in logical thinking. Also, I sometimes had problems in presenting ideas clearly. Prof. Shen has given me invaluable suggestions that helped me improve my skills in writing and comprehension. During my three years PhD study I learnt from Prof. Shen the most significant mentality of concentration, dedication and self-criticism in doing scientific research. This mentality has guided me throughout my research and led to completion of this thesis.

I am very grateful to my sub-theme supervisor, Professor Teruo Matsuzawa, for his valuable suggestions on my thesis work. His broad knowledge and penetrating comments are so impressive and helpful to my research. I also wish to thank Professor Susumu Horiguchi, Professor Maneo Kaneko and Associate Professor Yasuo Tan for their time spent in examining my thesis.

Many thanks are expressed to all members in Shen laboratory for various discussions and suggestions on my research, and to all my friends and staff members in Japan Advanced Institute of Science and Technology (JAIST) for their help, support and friendship. I am grateful for the experiences that I gained over the time I spent with them at JAIST.

I wish to express my special thanks to the 21st Century COE Program "Verifiable and Evolvable e-Society" of Ministry of Education, Culture, Sports, Science and Technology, and Grant-in-Aid for Scientific Research General Research Scheme (**B**) Grant No. 14380139 of Japan Society for the Promotion of Science (JSPS), for provision of full support to my thesis work; to C&C Foundation, Inoue Foundation, Telecommunications Advancement Foundation and JAIST Foundation for supporting me during this research.

I am highly obligated to my family for their endless love and encouragements that gave me enormous moral support and helped me complete my work in a distant land. My heartiest thanks goes to my parents for teaching me to be a honest and diligent person, and to my sister for her love and friendship. I wish to dedicate this thesis to them.

Contents

List of Figures

Chapter 1

Overview of Network Topology Discovery

The knowledge of network topology is critical for many applications. As the network size increases, network topology discovery (NTD) has received more and more attention recently. We provide an overview of NTD in this chapter, which includes the methods for NTD, its significance and applications, and the related work.

1.1 Definition of Network Topology Discovery

Physical topology represents the topology model for layer 1 of the OSI stack - the physical layer (RFC2922). Physical topology consists of the devices in the network and how they are physically interconnected. These devices include communication infrastructure devices, such as hubs, switches, and routers, as well as 'leaf' devices such as workstations, printers, and servers. Generally, user data passes through infrastructure devices while leaf devices are sources and sinks of data. Knowledge of the physical topology of a network is extremely important for the successful execution of many network management tasks such as fault monitoring and isolation, server placement and resource sharing. The physical topology can be depicted as a graph, with internal nodes representing switching elements and edge nodes representing hosts.

Logical topology is related to the physical topology, and can also be represented as a graph. The logical topology graph can represent the topology model for data link layer and network layer of the OSI stack, for which we call layer-2 topology and layer-3 topology in this thesis respectively. Over a specific period of routing stability, the logical topology graph is determined by the paths traversed by packets sent from the sources to the receivers. It is clear that the logical topology is only stable during a period when no routing changes occur. In one sense, the logical topology displays less information than the physical topology, because it does not indicate all connectivity or switching elements. However, the logical topology provides some important information that cannot be generated from the physical topology: it identifies the paths traversed by packets sent from each source to the receivers and indicates where these paths diverge and merge. This information is extremely useful for the evaluation of the resource sharing capability of the network under the current configuration, the recovery of the loss and congestion, and also can guide the decisions of source-based routing algorithms, the determination of the routing bottleneck and so on.

Topology discovery can be defined as a process that discovers all required network entities and connections among them. According to different *objectives*, topology discovery is classified into layer-2 topology discovery and layer-3 topology discovery. In layer-2 topology discovery, network entities required to be discovered include switches, hubs, bridge, and so on. While for layer-3 topology discovery, it only requires to discover the router-to-router interconnections and router interface-to-subnet relationships. According to *operation* of topology discovery, it is classified into proactive and on-demand topology discovery. Proactive topology discovery is based on the transmission of probe packets to collect topology information and exchange of this information between nodes. On-demand topology discovery requires to set probers in the network. The probers monitor and analyze network traffic and report topology information to the network administrator. Proactive topology discovery requires less time than on-demand topology discovery, while brings heavier burden than on-demand topology discovery. Depending on *applications*, topology discovery can be classified into domain topology discovery and backbone topology discovery which aim to discover the topology within a domain and the backbone in a network respectively. Furthermore, depending on different *network environments*, topology discovery can be divided into Internet topology discovery, multicast network topology discovery and wireless network topology discovery.

The objective of NTD is to develop algorithms/tools that have following characteristics:

1. Fast: discover network topology timely and keep internal consistency of all the data;

2. Complete: identify as many entities as possible with the least probability of error;

3. Accurate: try to keep the results as correct as possible;

4. Efficient: consume the least possible network resources, that is, introduce the lowest possible overheads to the network.

1.2 Significance of Network Topology Discovery

Network topology constantly changes as new members join a network, links breakdown and other various network behavior. Keeping track of network topology manually is a difficult, if not impossible, job. So automatic topology discovery algorithms are needed. Moreover, discovered topology information is useful for many applications:

1. Simulation: In order to simulate a real network, the topology of the network must be first obtained.

2. Network Management: Network topology information is useful in deciding whether current hardware is configured correctly and new routers should be added. It also allows network managers to find bottlenecks and failures in the network.

3. Siting: A network map helps users determine where they are in the network so that they can decide where to site servers, and which ISP to join to minimize latency and maximize available bandwidth.

4. Topology-aware algorithms: Topology information enables a new class of protocols and algorithms that exploit knowledge of topology to improve performance. Examples include topology-sensitive policy and QoS routing, and group communication algorithms with topology-aware process group selection.

5. Location of mirror server: Appropriate location of mirror server determined by topology information can greatly decrease latency and solve the bottleneck problem.

6. Service Management: Quality of service can be improved significantly if topology information is transparent to users and administrators. For example, mail, ftp, web, SNMP(Simple Network Management Protocol), DNS(Domain Name System).

7. Network diagnostics: Topology information enables troubleshooting in a network and help build schemes for loss recovery and congestion control.

8. Network optimization: Topology information can help reconfigure a network to improve performance.

9. Designated data collection and analysis: Analysis on given paths or subnets is enabled if topology information is obtained.

Therefore, network topology not only has a great impact on network performance but also is very useful to many tasks of various applications. It hence becomes a challenging task to discover network topology automatically.

1.3 Methodologies for Network Topology Discovery

We discuss the methodologies of topology discovery (TD) according to the classification of Internet TD, multicast network TD and wireless network TD.

1.3.1 Methodologies for Internet Topology Discovery

Available tools/protocols which can provide information sources for Internet TD are listed as follows:

1. Ping:

   The “Ping” program contains a client interface to ICMP (Internet Control Message Protocol). It can be used by a user to verify whether an end-to-end Internet path is operational, and thus know whether a host is alive or not. The ping program also collects performance statistics, i.e. the measured round trip time and the number of times the remote server fails to reply. Usually, when a user sends a “Ping” probe packet (ICMP echo request) to a host, the host will return an ICMP echo reply message. The message shows the received sequence number (starting at 0 and incremented after each transmission), and the measured round trip time (in milliseconds). If the host is power off or does not exist, there will be no ICMP echo reply message to the user. To verify multi-path operation more efficiently, “Ping” can be executed asynchronously.

   Similar to “Ping”, there is another program called “Directed broadcast ping” which can verify all hosts’ availability within a subnet simultaneously. Though it can accelerate the discovery process. However, many routers and hosts prevent them from receiving these probe requests. Therefore, discovered topology may be very incomplete.

2. Traceroute

   The “traceroute” program also contains a client interface to ICMP. Like the “ping” program, it may be used by a user to verify whether an end-to-end Internet Path is operational, but also provides information on each of the Intermediate Systems (i.e. IP routers) to be found along the IP Path from the sender to the receiver. Traceroute uses ICMP echo messages. These are addressed to the target IP address. The sender manipulates the TTL (hop count) value at the IP layer to force each hop in turn to return an error message.

   The program starts by sending an ICMP Echo request message with an IP destination address of the system to be tested and with a Time To Live (TTL) value set to 1. The first system that receives this packet decrements the TTL and discards the message, since this now has a value of zero. Before it deletes the message, the system constructs an ICMP error message (with an ICMP message type of “TTL exceeded”) and returns this back to the sender. Receipt of this message allows the sender to identify which system is one link away along the path to the specified destination.

   Then the sender sends an ICMP Echo request message with TTL being 2. The first system decrements the TTL and forwards to the second system on the path to the destination.

The second system decrements TTL and discards the message, meanwhile constructs an ICMP error message with its own IP address and returns to the sender. The sender goes on increasing the TTL of ICMP Echo request message by 1 if the system that responds is not the intended destination. The process repeats as above until the sender receives a receives a response from the intended destination (or the maximum TTL value is reached). In this way, the sender learns the identities of all systems along the IP path to the destination.

Obviously, traceroute is very helpful to construct the topology of concerned network. However, some routers are configured to discard ICMP messages, while others process them but do not return ICMP Error Messages. Such routers hide the “topology” of the network, and thus impact correct operation of protocols. Some routers will process the ICMP Messages, providing that they do not impose a significant load on the routers, such routers do not always respond to ICMP messages. In all these cases, the discovered topology becomes incomplete.

3. SNMP (Simple Network Management Protocol)

   SNMP is the standard network management protocol for TCP/IP networks. The three major components of the SNMP that form an integral part of its foundation are the network device, the agent and the manager. A network device is also called the Managed Object, which requires to be monitored and managed by the manager. An agent is a mediator between the manager and the device. The agent resides inside the network device. It collects the management information from the device and makes it available to the manager. The agent is a program that resides in the device and is not a separate entity. The manager is a separate entity that manages the agents from a remote place. It queries the agent to know the functioning of the device.

   Management Information Base (MIB) is a collection of definitions which define the properties of the managed objects. It enables the SNMP manager to operate intelligently on the data available on the managed devices. The data about the names and types of the objects in the device that the manager needs to know are all included in MIB. Topology discovery based on these information can be operated efficiently and effectively. However, these information are not always available due to restriction of access to MIB. Thus, topology discovery based on SNMP is limited.

4. DNS

   The Domain Name System (abbreviated DNS) is an Internet directory service. It is a distributed hierarchical database, and its structure is akin to a computer’s filing system of folders and subfolders. DNS is developed because it is difficult to remember the IP addresses of the machines. A domain’s DNS name server keeps a binding from every name in the domain to its IP address. Most DNS servers respond to an “zone transfer” command by returning a list of every name in the domain. Thus, all hosts and routers within the domain can be identified through DNS zone transfer. This provides very useful information for topology discovery. It may not be complete, however, since hosts obtaining IP addresses using DHCP are not served by DNS. Moreover, some network managers disable DNS zone transfer due to security concerns.

One the advantages of DNS is that we can easily and quickly discover many devices. These can serve as a starting point for other algorithms. There are also a lot of disadvantages. First of all, the information stored in the DNS database may not be consistent. This means that some devices may not be active any more. Also, it is very possible that there are some devices which are active, but they are not registered in the database. Another disadvantage is that we cannot rely on the other information which is stored in the database. This is true, because there is no standard and widely-accepted format to encode this information (especially the Host Information field). Also, the information which we can take with DNS cannot help us to discover the structure of the actual network. By this we mean that we cannot extract information about the subnets or even the role of each device in the network. Of course we cannot extract any information about the performance of the network or the quality of service it gives to its users. Another drawback is that DNS zone transfer is not always possible. Some DNS servers don't allow zone transfer. Thus whether DNS is chosen to involve in topology discovery depends on various applications. If DNS is chosed, it must incorporate with other information sources.

5. Other techniques and proprietary protocols

   Depending on the exact type of network there are some other techniques which provide more about the topology of the network. For example a single node can easily find about its neighbors by simple looking its ARP (Address Resolution Protocol) table.

   Another information source comes from using the information of routing processes, such as OSPF (Open Shortest Path First), RIP (Routing Information Protocol) and BGP (Border Gateway Protocol). In the case of OSPF and RIP, the subnets, the number and position of routers in networks can be found. In the case of BGP, the path, which is a collection of Autonomous Systems, can be found.

   It is also possible to use specialized techniques to learn more about non-IP protocols. For example in the case of IPX (Internetwork Packet Exchange), we can use the SAP (Service Advertisement Protocol) to learn more about the servers of the network and their location in the network. We can also use vendor-specific techniques to learn more about the behavior of the devices and the network.

The above tools/protocols provide information source for topology discovery. However, each individual information source is neither complete nor reliable to discover the network topology. Therefore, many methods based on various sources have been developed to discover the more accurate and complete possible topology. All of these methods apply the same basic discovery process as follows.

Step 1: Come up with a temporary set of hosts in the network that may or may not exist;
Step 2: Go through each host and determine whether they really do exist. If they exist:
Step 3: Add to permanent set.
Step 4: Use some heuristics on those hosts to find more hosts which are added to the temporary set. Go back to step 2.

By use of different tools/protocols to abstract topology information, the existing methods can be classified as follows:

1. Network topology discovery based on SNMP and ping [60].

   This method is the simplest because it assumes that SNMP is available everywhere in the domain. MIB provides neighboring nodes information of each node, and Ping helps to find whether these nodes are alive. Thus all discovered nodes and connections reconstruct a layer-3 logical topology of the network.

   By use of SNMP and Ping, topology discovery can be efficient, fast, complete, and accurate. However, it can only be used on networks where SNMP is enabled on all routers. Thus, it fails to discover a real complete topology if there are some routers which do not support SNMP. Moreover, MIB is only accessible for authorized users, this limits the application of this method.

2. Network topology discovery based on DNS and broadcast ping [60].

   This algorithm assumes the domain allows DNS zone transfer and pings to broadcast addresses. DNS zone transfer is used to add more IP addresses to temporary set for identification, and ping will verify their existence and connections. In order to discover nodes not found in DNS zone transfer, broadcast ping is used to identify more nodes.

   This method can correctly determine network topology in the absence of SNMP. However, DNS zone transfer and broadcast ping may both be unavailable for security reason in practice. Therefore, a method which have little limitation in reality is desirable.

3. Network topology discovery based on DNS and traceroute [60].

   In this method, DNS zone transfer gets lists of all routers and hosts in the domain, ping verifies their existence and traceroute identifies their connections. It is faster than the previous one and has fewer overheads. It makes fewer assumptions than the previous algorithm, and thus is more widely applicable. However, if DNS lookup cannot return all the IP addresses of a router, or DNS zone transfer is disabled, this method cannot result in a complete topology. It cannot guarantee accuracy, either.

4. Network topology discovery based on Ping and traceroute [62].

   Since DNS zone transfer is disabled in many domains for reasons of security, the method only using ping, traceroute and intelligent guess about the IP addresses in a domain is proposed. It requires the least support from the network, thus can be applied most widely. However, the completeness cannot be guaranteed as above methods because ping and traceroute sometimes are also limited. The accuracy of this method depends on the configuration of entities in the network because intelligent guess may not work always correctly.

5. Topology inference from end-to-end measurements by unicast/multicast [26, 30].

   This method can be performed actively or passively. It is also called network tomography which has attracted many researchers in last five years. It obtains internal characteristics such as topology, loss and delay by inference from end-to-end measurements as will be introduced in detail in Chapter 2. To infer the topology, the probe packets are sent by unicast/multicast from a source to receivers. After the source collected the end-to-end measurements, the topology between the source and the receivers can be inferred by this method.

   Topology inference from unicast end-to-end measurements brings heavy burden to the network. Thus sending probe packets by multicast, which can lighten the burden than that by unicast, is preferred to be used in topology inference if the routers within the network of interest support multicast. Though this assumption may not be feasible in some networks, topology inference by multicast measurements is prevalent and works very well in multicast network.

There are also many other methods for different objectives, such as address forwarding tables (AFT)-based link layer topology discovery, routing protocol RIP and OSPF-based network layer topology discovery. Incorporating several methods to discover the topology of a network is sometimes feasible and needed. Generally speaking, methods for Internet topology discovery should be chosen depending on the expected results, which information source is available, and the network status.

1.3.2 Methodologies for Multicast Network Topology Discovery

This section introduces methodologies for multicast network topology discovery. Discovering multicast distribution tree is the main objective in multicast network topology discovery. Possible approaches are usually classified into three types [46]: multicast topology inference based on end-to-end measurements which does not need support from internal network nodes; mechanisms requiring the help of internal network nodes; and schemes that lie in their midst.

The key idea underlying the first approach is that receivers sharing common paths on the multicast tree associated with a given source will see correlations in their packet losses/delays. Thus based on the shared loss/delay statistics for transmitted probe packets, one can attempt to infer the multicast tree. This elegant approach to the problem is particularly advantageous in that it requires no support from internal nodes. It does however, potentially suffer from significant communication overheads required to periodically gather large amounts of loss data so as to adapt to changing memberships or topology, and processing overheads to assemble and perform the inference step. This is currently conceived as a centralized approach whose accuracy is unlikely to scale nicely. The approach assumes network links have steady state loss characteristics, which may or may not be realistic on the time-scales during which loss data are collected. A final point is that the approach permits identification of the "logical" multicast topology rather than the actual physical topology. As will be discussed in Chapter 2, this means that those single-child parent nodes where no branch exists in a path are inferred as a single node and their links are collapsed to a single "logical" link. In practice, this may not be an appropriate abstraction of the actual topology. The key advantage of this approach lies in its applicability to inferring multicast trees without requiring the help from internal nodes. Note that for multicast networks, logical topology only denotes layer-3 logical topology and physical topology means a graph of real connections of multicast members.

The second approach to multicast topology discovery is based on using the MTRACE feature currently implemented in the IGMP protocol. MTRACE enables tracing the path from a source to a destination on a given multicast distribution tree. A query packet is sent from the requester to the last multicast router (on the distribution tree) prior to a given destination. This query is then forwarded hop-by-hop along the reverse path from the "last-hop" router to "first-hop" router, i.e., that to which the source is attached. While the query packet traverses the tree, each router adds a response data block containing its interface addresses and packet statistics. When the query packet reaches the first-hop router it is sent back to the requester via unicasting or multicasting. The main advantage of this approach is that it provides full information on the multicast topology based on currently available IGMP features. The physical topology including interface addresses of routers can therefore be obtained. This however, shall rely heavily on special services at routers.

The third approach combines the advantages of the above two approaches. The work along this line can be found in [46]. Our work in [65, 66] also belong to this category. This method usually requires lighter support from internal nodes than the second method and can infer multicast topologies which are closer to the physical topology than the first approach.

1.3.3 Methodologies for Wireless Network Topology Discovery

Topology information in wireless networks is very helpful to control transmission power, avoid congestion, discover resources and gather data. However, it's much more complicated to discover topology for wireless networks than for wired networks. This is because of complex environment in wireless networks such as more frequently changing topology, complex channel and very limited resources. There is no IP subnet hierarchy or popular network management protocol, such as SNMP for wireless networks. Therefore, the methodology for topology discovery in wireless networks is significantly different from that in wired networks.

Existing work related to topology discovery for wireless networks can be listed as follows. A clustering scheme to discover ad hoc network topology is proposed in [22]. Ad hoc Network Management Protocol (ANMP) in [22] applies hierarchical structure and similar MIB as in SNMP. The cluster heads are dynamically chosen based on geographic location or network connectivity, and the MIBs at cluster heads are used to gather topology information. However, this scheme has the overhead of constantly maintaining cluster heads in the network. Additionally, the information in the MIBs might be stale due to mobility and could fail to provide a complete link information of the network. Another topology discovery algorithm is presented in [23]. It applies mobile agent to wireless network topology discovery. Mobile agents in the nodes periodically gather topology information and disseminate it to all the other nodes in the network. However, this scheme does not provide an instantaneous topology of the network. This algorithm is also extremely intensive in time and messages to discover a complete topology of the network. TBRPF [8] and OLSR [40] are link state routing protocols. They require each node to constantly maintain a partial topology of the network. This is an overhead when the link information is required temporarily at a few nodes. Link state routing protocols provide network layer topology information. Further, [29] proposes a hierarchical tree-based clustering scheme for wireless sensor networks. The proposed TopDisc algorithm gives an efficient way to get approximate yet structured information about the topology. The constructed network topology is a Tree of Clusters (TreC) rooted at the monitoring node. This topology is used for data dissemination and aggregation, duty cycle assignments and nework state retrieval. A mesh-based topology discovery protocol for hybrid wireless networks is proposed in [21]. It can, more powerfully than above methods, discover the entire topology information adaptively and maintain the topology at a few prespecified nodes. This protocol is a more reliable protocol for topology discovery in wireless networks.

Summarizing the above, methodologies for wireless network topology discovery can mainly be classified into: topology discovery based on cluster in hierarchical tree; topology discovery based on mobile agents; and mesh-based topology discovery.

1.4 Related Work and Our Contributions

Since topology discovery provides invaluable information for various applications in Internet, multicast networks and wireless networks, a large amount of work has been devoted to this area. Especially for Internet topology discovery, many projects were developed in late 90s.

1.4.1 Related Work

Topology information collected by Atlas [1] and CAIDA (Cooperative Association for Internet Data Analysis) [2] are earliest and most notable collections of Internet topologies. These topologies are neither automatically discovered, nor updated. Later on, automatic topology discovery was developed in Hewlett-Packard's OpenView (www.openview.hp.com), IBM/Tivoli's Netview (www.tivoli.com) and Ciscoworks Network Connectivity Monitor (www.cisco.com) which are commercialized products based on SNMP. Other commercially available tools for inferring layer-3 network topology include Actualit's Optimal Surveyor (www.actualit.com) and the Dartmouth Intermapper (intermapper.dartmouth.edu). These tools offer an IP-network management functionality for automatically discovering routers and IP subnets and generating a layer-3 topology. However, SNMP is not universally deployed and only administrators are authorized to perform topology discovery. Thus SNMP-based topology discovery has limited application. Projects Octopus and Argus in Cornell university developed a set of tools which have implemented three algorithms for automatic topology discovery using Perl language. One of them uses SNMP to lookup the ARP tables of routers. The other uses DNS zone transfer and traceroute. The third tries to guess some addresses and then use traceroute to find the topology. This implementation uses modified version of ping and traceroute, which speeds up the running time of the algorithm. Another popular tool is Skitter [3], which was developed by CAIDA. This tool uses traceroute to find the paths connecting two nodes and also to collect performance information from them in a distributed form. Similar tool using traceroute and ping is developed in Mercator project [36]. CAIDA developed another tool, Otter [4], to aggregate topology views from multiple routers which can further improve the accuracy of topology. This is also called aggregation view of MBGP topology. The Rocketfuel project [70] uses results from 294 public traceroute servers to build router-level ISP topologies; it also employs techniques that can reduce the amount of probing and resolve address aliasing.

Topology discovery for layer-2 is more challenging than layer-3 TD because it tries to discover more detailed connections among network elements. Layer-2 TD has been developed in Cisco's Discovery Protocol and Bay Networks' Optivity Enterprise (www.baynetworks.com). The proprietary tools and protocols are typically based on vendor-specific extensions to SNMP MIBs and are not useful on a heterogeneous network comprising elements from multiple vendors. Peregrine's Infratools software (www.peregrine.com), Riversoft's NMOS product (www.riversoft.com), and Micromuse's Netcool/Precision application (www.micromuse.com) claim to support layer-

2 topology discovery, but these tools are based on proprietary technology to which we do not have access. In an effort to some standards for physical-topology discovery, IETF has published an RFC proposing a standard for a physical network topology MIB (originally proposed by Cisco). Unfortunately, very few vendors implement the proposed physical MIB design. Loran Network Systems released a software for discovering layer-2 topology in heterogeneous networks. Briefly, their approach is based on trying to statistically map (i.e., correlate) the traffic patterns observed at the ports of different elements in the underlying network, and probabilistically inferring connections for ports with similar traffic characteristics [28, 55]. Their approach relies on statistical correlation, so it can only infer element connections with some (high) probability; furthermore, it is not at all clear if or how their proposed method would work in the presence of interconnections between network elements belonging to different IP subnets. Shao et al. [59] also propose correlating benchmark measurements to determine the functional differences between a central server and a collection of machines across a LAN; however, for network-management purposes, such a application-level view of the underlying network structure is often insufficient. Y. Breibart et al. [15] and [14] proposed algorithms that relies on standard SNMP information to discover the layer-2 topology of a heterogeneous, multi-subnet network. They utilizes information either from the address forwarding tables (AFTs) of elements capturing the set of medium access control (MAC, i.e., layer-2) addresses that are reachable from each element interface, or (in the absence of AFT data) from the elements' NetToMedia tables.

Multicast network topology discovery aims to discover layer-3 multicast trees. Proposed solutions to this problem rely on either correlating end-to-end multicast measurements [30, 33], or using some specialized mechanism that requires router cooperation [46, 65, 67] as presented in Section 1.3.2. The project MINC (Multicast-based Inference of Network-internal Characteristics) is developed for multicast network topology discovery and other internal characteristics such as loss and delay. They proposed a novel methodology for identifying internal network performance characteristics based on end-to-end multicast measurements. The methodology is based on statistical estimation theory, and can be used to characterize the internal loss and delay behavior of a multicast network. Our work is mainly based on the research in MINC.

For wireless networks, same as above, topology information is very important. The project WTD (Wireless Topology Discovery) of UC San Diego [71] is a project which focuses on wireless topology discovery and analysis. It collects data on the dynamic characteristics of a real world wireless network. Through analyzing the collected data, the long term goal of this project is to test and develop reliable and efficient routing protocols for large, and perhaps geographically constrained, wireless networks. Since the topologies of wireless networks change very frequently, maintaining a global view of topology for wireless networks is too energy consuming which is not worth in most applications. Local neighboring information, instead of the global topology, is usually sufficient to support the function of wireless networks. The final goal of topology discovery, either global or local, is to develop routing protocols as mentioned in [71] above which can satisfy various applications. Thus, topology discovery is always embedded in development of routing algorithms.

Wireless sensor networks (WSNs) represent an important type of wireless networks which have a wide range of applications in practice. For wireless sensor networks (WSNs), how to keep coverage and connectivity is the most fundamental issue. Both coverage and connectivity are strongly influenced by topologies of WSNs. Coverage problems which address how a point or a path is covered have been studied widely in [18, 38, 47, 49]. How to place node in WSNs to provide connected coverage has been studied in [10, 39, 41, 54]. All these work showed the significance of topology for connectivity and coverage in wireless sensor networks. Another important issue in WSNs is energy-constraint which is also related to network topology. There have been a lot of work in studying various methods to achieve energy efficiency. The existing approaches to save energy can be classified into two categories. One is turning off or changing some nodes to sleep mode as proposed in [20, 39, 57, 61]. The other is minimizing sensing range and transmission range while keeping connected coverage as proposed in [44, 51]. In both approaches network topology is critical for achieving energy efficiency.

1.4.2 Our Contributions

Our research work contributes mainly to the following three aspects: topology discovery for multicast network and its applications in performance evaluation; mobile agent-based topology discovery and performance analysis; and topology analysis in wireless sensor networks and its applications in routing.

Conceptually, we may classify all problems of network topology discovery (NTD) into three domains in the dimensions of application, methodology and model respectively: communication pattern, discovery methodology and network type. The three components contained in this thesis are drawn from these three different domains of NTD and represent new developments of NTD in these domains which we have made during our research. To study NTD comprehensively, we choose one representative in each domain: multicast in the communication pattern domain, mobile agent-based method in the discovery methodology domain and wireless sensor networks in the network type domain. We start with NTD for multicast networks because multicast is the most widely used communication pattern existing in various applications. We then move on to the proposal of a novel methodology of deploying mobile agents for the NTD task because this method is different from the traditional algorithmic approaches and it possesses some interesting properties. Finally we show how to carry out the NTD task in the form of topology analysis in wireless sensor networks which has different structural properties from wired networks and show increasing importance in applications. Our achievements made in these three domains provide a complete picture of the contributions of this thesis to network topology discovery this challenging research area.

For multicast network topology discovery, tomography from end-to-end measurements has received considerable attention recently because of its lower traffic burden and higher efficiency than other methods. We apply network tomography in our research to identify topology and internal loss/delay performance. We propose Binary Loss Tree Classification with Hop count (HBLT) and Binary Hamming distance Classification (BHC) algorithms which consider hop count and hamming distance of sequences on receipt/loss of probe packets maintained at each pair of nodes respectively. Both algorithms improve the accuracy and efficiency of tomography over previous algorithms from different prospectives. HBLT takes level information into account which benefits greatly to inference procedure. BHC algorithm incorporates hamming distance into hop count-based HBLT algorithm, and achieves the best performance for topology discovery among available methods. Based on the discovered topology, we further propose network-internal loss/delay performance inference schemes. They employ end-to-end loss/delay measurements to infer the internal loss/delay performance. We perform detailed analysis to show the accuracy and efficiency of these schemes and validate them by simulation results.

To develop new approaches for topology discovery, we apply mobile agents technology in topology discovery. We propose several mechanisms for both Internet and multicast network topology discovery, including the report-at-newly-found-nodes (RN) algorithm and the report-at-leaf-nodes (RL) algorithm. Through analysis on the behavior of mobile agents with different report fashions, we study the performance of different mechanisms for topology discovery of both Internet and multicast networks and verify their feasibility in simulated networks. In mobile agent systems, it is shown that topology discovery can be performed correctly and efficiently due to the inherent advantages of mobile agents. However, the assumption that all entities must support the execution of mobile agent codes may limit the application of mobile agent-based topology discovery algorithms in practice.

We extend our research to wireless sensor networks. Since coverage, connectivity, reliability and energy-constraint are the most important issues in WSNs, we address them from the topology point of view. Instead of topology discovery, topology control is more meaningful in WSNs because neighboring information is usually enough to support various applications of WSNs in practice. We thus focus on studying energy-efficient topologies in which deployed sensor nodes can cover the required area and guarantee their connection as well. We show that the four patterned topologies, triangle-based, square-based, hexagon-based and strip-based topologies, can meet different requirements for reliability and coverage. Each topology pattern provides a different performance metric for WSNs built with the same number of sensor nodes. For example, the highest reliability is supported by triangle-based WSNs and the maximal coverage area is provided by strip-based WSNs. We further propose two routing schemes to achieve desired energy-efficiency. Our first protocol considers different requirements for energy consumption and transmission delay, and incorporates different route selection functions by combining the length of route and the number of streams at individual nodes. We demonstrate clearly the advantages of applying this strategy to achieve different performance goals. Our second protocol employs random walk for routing in WSNs with patterned topologies. We show that this protocol has a high successful transmission rate by quantitative analysis for the first time to our knowledge. The comparison on energy-efficiency performance between our protocol and the shortest path routing shows the advantages of our protocol. These advantages become significant for small-size data transmission in WSNs.

Summarizing all the contents of our research, we list the following main contributions of this thesis:

- We propose new algorithms for multicast network topology discovery that are more efficient and accurate than previous methods;
- We develop new methods for network internal loss/delay performance inference based on discovered topology information;
- We propose new methods for mobile agent-based network topology discovery, and develop statistical models to analyze the behavior of mobile agents and system performance.
- We study different topology deployment schemes in wireless sensor networks and their impacts on reliability and coverage and develop energy-efficient routing protocols.

Performances of all our developed algorithms and protocols are both mathematically shown by theoretical analysis and experimentally verified through simulations.

1.4.3 Thesis Outline

Chapter 2 starts with an introduction of network tomography for topology inference. It then presents our algorithms for multicast network topology discovery using hop count and hamming distance classification and shows that these strategies can bring in different benefits to topology inference. It further presents our methods for multicast network internal loss/delay performance inference based on the discovered topology information, in both binary trees and general trees as their extension. Finally, it shows how to utilize the inferred loss rate in topology discovery process to improve the quality of the discovered topology, and proposes the method of using hamming distance matrix for network loss/delay analysis from end-to-end loss/delay measurements.

Chapter 3 first gives an introduction to mobile agents and its applications, then presents two algorithms, report-at-newly-found-nodes algorithm and report-at-leaf-nodes algorithm, that deploy mobile agents for both Internet and Multicast network topology discovery. It further presents a detailed analysis on statistical behaviors of the mobile agents deployed in our algorithms, represented by dwell time, life span, report time and interreport time these key statistical parameters, to show the performance of the proposed algorithms.

Chapter 4 first introduces necessary background knowledge of wireless sensor networks (WSNs) and some fundamental performance metrics of WSNs. Then it discusses four desirable topology patterns for WSNs which can provide both coverage for required area and connectivity between all nodes deployed in the network. Based on the analytical results on these patterns, it further presents two routing protocols, route selection function-based routing and random walk routing, to achieve desired energy-efficiency. It also gives quantitative analysis on the performance of these protocols.

Chapter 5 concludes the thesis with a summary of its main contributions and some future research tasks.

Chapter 2

Multicast Network Topology and Loss/delay Performance Inference

We begin this chapter with introducing network tomography which provides the theory for topology inference in our work. Then we present our new algorithms for topology inference which are more efficient and accurate than previous schemes, and new scheme for network-internal loss/delay performance inference. We give theoretical analysis on them and validate them by simulation results.

2.1 Introduction to Network Tomography

Network tomography, first proposed by Y. Vardi [72], stems from the similarity between network inference and medical tomography. It can be looked as a sort of method which mainly relies on various inferential theory to uncover the network characteristics.

There are two forms of network tomography according to summary of [25] on recent literatures. One is link-level parameter estimation based on end-to-end, path-level traffic measurements. The other is sender-receiver path-level traffic intensity estimation based on link-level traffic measurements. The latter one is essentially the antithesis of the former one. Its goal is to estimate path-level network parameters from measurements made on individual links. For example, the estimation of origin-destination (OD) traffic from measurable traffic at router interfaces has important applications in routing optimization.

Our work belongs to the first type of network tomography, which is link-level characteristics inference, such as topology, link loss rates, and link delay inference, from end-to-end, path-level traffic measurements. The end-to-end, path-level traffic measurements can be obtained by active probing or passive monitoring in multicast or unicast. Network tomography from unicast measurements mainly use the method of sending packet pairs or stripes (triples, quadruples, etc.) to measure packet delay/loss and then infer network internal characteristics. It applies the same inferential theory as multicast-based network tomography. However, unicast and multicast measurements have different advantages in tomography. Tomography based on unicast measurements is more practical than that based on multicast measurements because multicast is not supported by all the networks. However, tomography based on unicast measurements is more complex and add much heavier burden to the network than tomography based on multicast. By multicast measurements, each probe packet is transmitted only once per link while resulted end-to-end characteristics at different end-points are highly correlated. Another advantage of using multicast is scalability [31]. Assume N probe packets are sent to collect end-to-end measurements, the load by multicast measurements grows proportionally to N in all links. However, the load by unicast measurements grows proportionally to N^2 in some link. Therefore, we mainly focus on multicast-based network tomography in this thesis. Tomography based on unicast measurements is out of our interest, which can be referred to [9,24,26,27,32].

2.1.1 Multicast-based Network Tomography

Network tomography from multicast measurements extracts data by multicast probing or monitoring multicast traffic. In multicast network, when a packet encounters an internal router where branching occurs, it is duplicated and sent to each branching node. Thus each multicast receiver should get a copy of the same packet if they are transmitted successfully. In a simple multicast network of Figure 2.1, assume the sender 0 sends probe packets to receiver 2 and 3 by active multicast probing. Any probe packets sent by node 0, if transmitted successfully, should reach both node 2 and 3. If node 2 loses this probe packet, it can be determined that loss occurred on link 2 because successful reception on node 3 proves the packet has been successfully received by node 1. In order to get internal link delay information in the network, observation on the delay difference between receiver 2 and 3 can be used. If a probe packet reaches node 2 without delay while node 3 gets the delayed packet, it can be determined delay occurs on link 3 because this packet has been received by node 1 without delay. By repeatedly sending the probe packets and collecting statistically measured results, network internal loss rates and link average delay can be inferred. The correlation of measurements at any pair of nodes such as that at node 2 and 3 exists throughout the whole multicast network because the packets received by the pair of nodes traverse along the same path from the root to their nearest ancestor. The more common links two nodes share, the more correlated their received packets are. Thus based on correlation from end-to-end path measurements, network internal characteristics can be obtained.

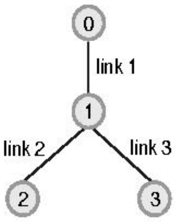


Figure 2.1: A simple multicast tree

2.1.2 Mathematical Models of Multicast Network

We use multicast tree model and loss/delay model which have been widely used in [17, 30, 31, 33, 34] to explain the correlation and inferential theory. A number of key assumption should be mentioned before going deep into tomography because they underpin current link-level network tomography techniques, determining measurement frameworks and mathematical models. The routing and topology are assumed to be constant during the measurement period. Thus dynamic routing and topology may restrict the amount of data collected for inference. Most current methodologies usually assume that performance characteristics on each link are statistically independent of all other links. However, this assumption is easily violated due to common cross-traffic flowing through the links. Temporal stationarity is also assumed in many cases. In link-level delay tomography, it is generally assumed that synchronized clocks are available at all senders and receivers. Although these simplifying assumptions do not strictly hold, such "first-order" approximations have been shown to be reasonable enough for the large-scale inference problems as presented in [25]. Thus, by above assumption, the multicast tree and its loss/delay measurements can be modelled as follows.

- Tree model

  The physical multicast tree is represented by a tree comprising actual network elements (the nodes) and communication links connecting them. Let $T = (V, L)$ denote a multicast tree with node set V and link set L. The root node 0 is the source of probe packets, and $R \subset V$ denotes the set of leaf nodes representing the receivers. A link is said to be internal if neither of its endpoints is the root or a leaf node. Let W denote $V \backslash (\{0, 1\} \cup R)$, where 1 is the child node of 0. Each non-leaf node k has a set of children node $d(k) = \{d_i(k) \mid 1 \leq i \leq n_k\}$, and each non-root node k has a parent $p(k)$. The link $\big(p(k), k\big) \in L$ is denoted by link k. Write $j \prec k$ if j is descended from k, $k = p^r(j)$ if j is r-level descended from k, where r is a positive integer. Let $a(U)$ denote the nearest common ancestor of a node set $U \subset V$. Nodes in U are said to be siblings if they have the same parent, i.e., if $p(k) = a(U), \forall k \in U$. The subtree of T rooted at k is denoted by $T(k) = \big(V(k), L(k)\big)$ where $V(k)$ and $L(k)$ are node set and link set rooted at k. The receiver set $R(k)$ is defined as the set of receivers descended from k, i.e., $R(k) = R \cap V(k)$. Figure 2.2 shows an example of multicast tree model.

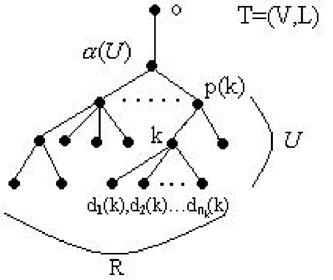


Figure 2.2: An example of a multicast tree

- Delay and Loss model

Probe packets are dispatched down the tree from the root node 0. Each packet arriving at a node k gives rise to copy sent to each child node of k. On each link, the packet is either lost, or transmitted with some delay. The delay can be represented as the sum of a fixed propagation delay and a variable queueing delay. Suppose Z_k to be the random variable that specifies the queueing delay of a packet traversing link k, $Z_k \in [0, \infty]$. $Z_k = \infty$ denotes packet loss. By convention $Z_0 = 0$. The queueing delay for the path from the root to a node k is $Y(k) = \sum_{j \succeq k} Z_k$. If a packet is lost on some link between node 0 and k, $Y(k) = \infty$. Likewise, if a packet does not encounter any queueing delay on each link between node 0 and k, $Y(k) = 0$.

Assume $\alpha_l(k)$ to be the probability of successful transmission on link k, and $\alpha_u(k)$ to be the probability of transmission without queueing delay on link k, i.e., $\alpha_l(k) = P[Z_k < \infty]$ $\alpha_u(k) = P[Z_k = 0]$. Thus, $1 - \alpha_l(k)$ denotes the probability of packet lost on link k. $1 - \alpha_u(k)$ denotes the the probability of link k being utilized because $Z_k > 0$ iff the link is utilized, which is also called link utilization. If $0 < \alpha_{l/u}(k) \leq 1, \forall k \in V \backslash \{0\}$, the loss tree is said to be a canonical tree. Any tree (T, α) in non-canonical form can be reduced to a canonical tree [30] by simply removing all subtrees rooted at the broken links whose successful transmission rates are 0. Henceforth only canonical loss trees are considered in this thesis. A loss tree can thus be modelled as (T, α_l). Similarly, and a delay tree can be modelled as (T, α_u).

For each link an independent Bernoulli loss (delay) model is assumed with each probe packet *being successfully transmitted* (or *transmitted without delay*) across link k with probability $\alpha_l(k)$ $(\alpha_u(k))$. Thus Z_k are independent random variables, and the progress of each probe packet down the tree can be described by Markov stochastic processes

$X_{l/u} = (X_{l/u}(k))_{k\in V}$ [33]. Here X_l denotes loss process and X_u denotes utilization process. For loss process, $X_l(k)$=1, if the probe packet reaches k (i.e., $Y(k) < \infty$) and 0 otherwise (i.e., $Y(k) = \infty$), $k \in V$. For utilization process, $X_u(k)$=1, if the probe packet reaches k without queueing delay (i.e., $Y(k) = 0$) and 0 otherwise (i.e., $Y(k) > 0$), $k \in V$. Their Markov properties come from the following facts:

$$X_{l/u}(0) = 1; X_{l/u}(p(k)) = 0 \Longrightarrow X_{l/u}(k) = 0;$$

$$P[X_{l/u}(k) = 1|X_{l/u}(p(k)) = 1] = \alpha_{l/u}(k).$$

Obviously, the loss and utilization processes are formally identical except that these processes represent the event of "no loss" and "no delay" respectively.

By above modelling, it is clear that observed end-to-end measurements are $(X_{l/u}(k))_{k\in R}$. Given $(X_{l/u}(k))_{k\in R}$, the objective is to infer the network topology, the network internal link-level loss and delay performance which have been modelled as $T = (V, L)$, and $(T, \alpha_{l/u})$.

2.2 Multicast Topology Inference

For multicast topology inference, end-to-end measurements on loss and delay, which result in performance measurements of loss rate, delay variance, average delay and link utilization can all be used. N. G. Duffield et al. discussed all measurements for topology inference in [34]. Detailed loss-based topology inference was studied in [16,30]. A method that adaptively choose loss and delay measurements according to the network situation was proposed in [33]. R. Castro et al. [19] concisely gave a description on maximum likelihood identification of topology which was also the key idea of above work. All of them are based on the measurements that can compose a metric holding the monotonicity property such as loss and delay. This generalized method will be introduced here because it's important for our work.

2.2.1 Binary Loss Tree Classification (BLT) Algorithm

As we have known, the more number of shared links or queues in the paths from the root to a pair of receivers, the more correlated the observed measurements on the pair of receivers are. Suppose a metric $\gamma = \{\gamma_{i,j}\}$ denote the correlation matrix where $\gamma_{i,j}$ is the correlation parameter of any node pair (i, j). This metric satisfies the following property:

Monotonicity Property: Let i, j and k be any three receivers and let p_i,p_j and p_k denote the paths from the sender to each. If p_i shares more links with p_j than with p_k, then $\gamma_{i,j} > \gamma_{i,k}$.

The correlation parameter of each pair in the matrix can be estimated from a number of different end-to-end measurements including counts of losses, counts of zero delay events (utilization), i.e., $(X_{l/u}(k))_{k\in R}$. The estimated matrix is denoted by $x = \{x_{i,j}\}$ that can be interpreted as observations of the true metric values γ contaminated by some randomness or noise [19]. The estimated metrics are randomly distributed according to a density (whose precise form depends on the contamination model) that is parameterized by the underlying topology T and the set of true metric values, written as $p(x|\gamma, T)$. The x are observed and fixed quantities. When $p(x|\gamma, T)$ is viewed as a function of T and γ it is called the likelihood of T and γ. The maximum likelihood tree is given by

$$T^* = argmax_{T\in\mathcal{F}} max_{\gamma\in\mathcal{G}} p(x|\gamma, T), \tag{2.1}$$

where $\mathcal{F}$ denotes the forest of all possible tree topologies connecting the sender to the receivers and $\mathcal{G}$ denotes the set of all metrics satisfying the monotonicity property.

Since it's extremely difficult to scan all possible tree topologies to compute the maximum likelihood tree in Equation (2.1), the deterministic algorithm that can obtain suboptimal results is considered. A representative example for deterministic algorithm is Deterministic Binary Tree (DBT) classification algorithm proposed in [34].

In DBT, $\gamma_{i,j}$ is the probability that a probe packet *reaches the nearest common ancestor of node pair* (i, j) *successfully* (if use loss measurements) or *reaches it without queueing delay* (if use utilization measurements). Denote it by $A(i, j)$ in function form. Let $X(k)$ generalize the measurements for loss and utilization $X_{l/u}(k)$. Followed $X(k)$ is written as X_k for simplicity. Thus, for any node pair (i, j), $i \neq j \neq a(i, j)$,

$$A(i,j) = \frac{P[\vee_{k\in R(i)} X_k = 1] P[\vee_{k\in R(j)} X_k = 1]}{P[\vee_{k\in R(i)} X_k = \vee_{k\in R(j)} X_k = 1]}. \tag{2.2}$$

Because each probability in Equation (2.3) is impossible to obtain in practice, $A(i, j)$ is estimated by $A^{(n)}(i, j)$ (which was denoted by $x_{i,j}$ in above estimated matrix) as used in all previous work, where n is the number of probe packets.

$$A^{(n)}(i,j) = \frac{\sum_{m=1}^{n} X_i^{(m)} \cdot \sum_{m=1}^{n} X_j^{(m)}}{n \cdot \sum_{m=1}^{n} X_i^{(m)} \cdot X_j^{(m)}}, \tag{2.3}$$

where $X_i^{(m)} = \vee_{k\in R(i)} X_k^{(m)}$. $X_k^{(m)}$, $m = 1, \cdots, n$ denotes the measured outcomes observed at receiver k by n probe packets. Each probability in Equation (2.2) is estimated by the observation from n probe packets as in Equation (2.3), e.g., $P[\vee_{k\in R(i)} X_k = 1]$ is estimated by $\sum_{m=1}^{n} X_i^{(m)}/n$. As n goes to infinity, $A^{(n)}(i, j)$ is consistent with $A(i, j), i, j \in V$.

Then the key clue of multicast network topology inference is minimizing $A^{(n)}(i, j)$ in each iteration and identifying the node pair (i, j) as siblings. For simplicity in describing siblings identification by this estimation approach, we call it A-approach in later sections. The main idea of DBT algorithm by use of A-approach is as follows.

Step1: Input the receiver set R, and the observed measurements on R, $X_k^{(n)}$, $k \in R$; denote the set of nodes to be inferred by R', the set of inferred nodes by V', and the set of inferred links by L'; let $R' = R$ and $V' = \emptyset$, $L' = \emptyset$;

Step2: Find a pair of nodes $U = \{u_1, u_2\} \subseteq R'$ which minimizes $A(U)$;

Step3: Mark the pair of nodes as siblings and remove them from R'; add them to the inferred node set V'; add a new node u' which denotes the siblings' virtual parent node to the node set R'; add the links from u' to each node in U to L';

Step4: Get the loss measurement of the virtual parent node u' for each probe packet by computing $X_{u'}^{(i)} := X_{u_1}^{(i)} \vee X_{u_2}^{(i)}$;

Step5: Iterate Steps 2–4 until R' contains only one element.

Step6: Output the inferred multicast tree (V', L').

By the above procedure the logical multicast tree is reconstructed in a bottom-up fashion. If loss measurements are used, DBT algorithm becomes Binary Loss Tree Classification (BLT) Algorithm. Since measurements on loss or delay do not make difference for inference procedure, we concentrate on BLT algorithm that based on loss measurements and study our new algorithm from end-to-end loss measurements.

BLT can identify the siblings in a multicast network without any single-child nodes. However, if there are single-child nodes in the multicast network, the algorithm cannot identify these nodes and their connecting links. Clearly, an inferred logical tree by BLT will differ from the actual physical tree if single-child node exists in the physical tree, and this difference increases proportionally to the increase on the number of single-child nodes. In practice there may be many internal routers connected with only one downstream multicast router because routers are configured to support multicast in a network only when necessary. Particularly, when there aren't many group members who require the multicast applications, the intervening routers are accordingly fewer, which results in a high probability on existence of single-child routers in the multicast network. Therefore, it's necessary to have a new scheme that can find those internal single-child nodes to reconstruct a topology closer to the physical topology than the previous schemes. We thus propose the usage of an important measurement — hop count, to develop a new algorithm called Binary Loss Tree Classification Algorithm with Hop Count (HBLT).

2.2.2 Binary Loss Tree Classification Algorithm with Hop Count (HBLT)

As mentioned above, in order to identify both the single-child nodes and the links connecting to single-child nodes, we take the hop count information into consideration at each node, because hop count brings significant benefits to sibling-relationship inference of individual nodes. By incorporating the level information in hop count into multicast topology inference, we propose the HBLT algorithm which can obtain a multicast topology closer to the physical tree than BLT.

Moreover, it should be noted that though one more measurement is taken into account, HBLT does not add any extra traffic burden to the multicast network because hop count can be computed by simply reading the TTL values of the probe packets. For internal nodes, the values of hop count can be easily computed by degression. Therefore application of hop count to topology inference can be done without deteriorating efficiency.

HBLT infers the multicast topology from the node with the maximum hop count and proceeds in a bottom up fashion. Since real siblings must have the same value of hop count, the set of all receivers is classified to different sets according to their values of hop count at the beginning. In the procedure of identifying siblings, only comparison between the sets with the same value of hop count is necessary, which in turn, avoids redundant comparison between nodes with different values of hop count. Below is the description of the HBLT algorithm:

```
1. Input: The set of receivers $R$, number of probe packets $n$, observed sequences at
   receivers $(X_k^{(i)})_{k\in R}^{i=1,\cdots,n}$;
2. $R' := R$, $V' := \emptyset$, $L' := \emptyset$, $h = \max_{k\in R}(k.hop)$, $W_e = \emptyset$, $(e = 1, \cdots, h)$; //$V'$ is the set
   of discovered nodes; $L'$ is the set of discovered links, $W_e$ is a set of nodes with hop count
   value $e$, $e$ is initialized as the maximum value of hop count for all nodes in $R$.//
3. for $k \in R$, do
4.     $W_{k.hop} := W_{k.hop} \bigcup \{k\}$ //Classify the receivers into different groups according to
       their hop count values.//
5. while $e > 1$ do
6.      while $W_e \neq \emptyset$ do
7.      Let $u$ be the first element in $W_e$; search for $v \in W_e$ to minimize $\widehat{A}(u, v)$, $(u \neq v)$;
8.      if $\widehat{A}(u, v) > \varepsilon_e$ then $S = \{u\}$, Set $r$ to be $u$'s virtual parent node; //Initially,
            sibling nodes set $S := \emptyset$, $\varepsilon_e$ is a given classification threshold of level $e$. If
            minimal value of $\widehat{A}(u, v)$ is still greater than $\varepsilon_e$, $u$ does not have siblings.//
9.          else $S = \{u, v\}$, Set $r$ to be $u$ and $v$'s virtual parent node; //$u$ and $v$ are
            siblings, $r$ is denoted as their virtual parent node.//
10.         for $i = 1, \cdots, n$ do $X_r^{(i)} := \vee_{l\in S} X_l^{(i)}$;
11.         $r.hop := e - 1$;
12.         $V' := V' \bigcup S$; $W_e := W_e \setminus S$; $W_{e-1} := W_{e-1} \bigcup \{r\}$; //The discovered two siblings
            are added to the discovered node set $V'$ and excluded from original set $W_e$, their
            parent node is added to the node set $W_{e-1}$.//
13.         for each $l \in S$, $L' := L' \bigcup \{(r, l)\}$;
14.         $S = \emptyset$;
15.     $e := e - 1$;
16. $V' := V' \bigcup \{0\}$; $L' := L' \bigcup \{0, r\}$; //Include root node and the link from root node to
    his child node to the discovered node set and link set.//
```

17. *Output*: Inferred topology (V', L').

The algorithm HBLT works as follows: Firstly all the receivers are classified into different node sets W_e $(1 \leq e \leq h)$ according to their values of hop count. Inference begins with identifying siblings in the node set with maximum value of hop count. The $\widehat{A}(u,v)$ $(u \neq v)$ of each node pair in W_e is calculated. Node v is identified to be u's siblings if $\widehat{A}(u,v)$ is minimum and less than the classification threshold of level e. Remove the siblings pair u and v from the node set with the hop count being e and add their parent node into the node set with hop count reduced by 1. The loss measurement $X_r^{(i)}$ of the parent node is obtained by "OR" operation of those of the siblings. When all nodes in W_e are grouped, decrease hop count value by 1. Repeat the same procedure among the nodes in the node set W_{e-1}. The algorithm ends when the hop count becomes 1.

2.2.3 Analysis on HBLT and BLT

We analyze the performance of HBLT and compare it with that of BLT from the following aspects: time complexity, misclassification probability and inference accuracy.

1. Analysis on Time complexity

As we discussed in HBLT, the application of the hop count information avoids redundant comparison between node pairs that are impossible to be siblings, because a pair of siblings must have the same value of hop count. However, BLT searches for the minimum $A(U)$ by performing comparisons among all possible combinations of node pairs. Therefore, HBLT is apparently superior to BLT in terms of the efficiency which is defined as the time needed to perform required comparisons in the algorithm concerned. Time complexity is thus used to measure algorithm performance. We compare the time complexities of both algorithms on different types of topologies in this section.

Before comparison, we need to define several parameters in order to describe the tree topologies and analyze time complexity.

Assume that there are n nodes in the binary tree distributed over h levels, and the number of nodes at level i is a_i, $1 \leq i \leq h$. Let the total number of leaf nodes in the tree be l. Clearly, $\log_2 l + 1 \leq h \leq l$. As only trees whose internal nodes all have two children are considered, we have $n = 2l - 1$ and $2 \leq a_i \leq 2a_{i-1}$ for $2 \leq i \leq h$.

For BLT algorithm, comparison on $A(U)$ is performed among all pairs of leaf nodes, so its time complexity $T_{BLT}(l)$ is determined by the total number of leaf nodes.

$$T_{BLT}(l) = \sum_{k=2}^{l} k * (k-1)/2 = \frac{1}{6}l^3 - \frac{1}{6}l \tag{2.4}$$

For HBLT algorithm, as we can see from the detailed algorithm in section 2.2.2, its time complexity is determined by the number of nodes at each level and computed by the following equation.

$$T_{HBLT}(a_1, a_2, \cdots, a_h) = \sum_{k=2}^{h} a_k^2 \tag{2.5}$$

a. Worst-case Time Complexity Comparison

Firstly, we compare the time complexities of both algorithms on the worst-case tree topologies where least level information can be obtained. A balanced full binary tree belongs to this case because all the leaf nodes are at the same level, their values of hop count are the same.

In this case, $h = \log_2 l + 1$ and $a_i = 2^{i-1}$ for $1 \leq i \leq h$. Thus, the time complexity of HBLT can be obtained from Equation (2.5).

$$T_{HBLT}(l) = \frac{4}{3}l^2 - \frac{4}{3} = O(l^2) \tag{2.6}$$

As for the time complexity of BLT on worst-case tree topology, we find that it is determined only by the number of leaf nodes and irrelevant to the tree structure. So its time complexity is always $O(l^3)$ as Equation (2.4) shows. Comparing Equation (2.6) with (2.4), we conclude that the time complexity of HBLT improves significantly by an order of $O(l)$ on BLT for the worst-case tree topologies.

Now let's have a glance at the best-case tree topology. The best-case tree topology for HBLT is a tallest tree with one leaf node on each level. Then it's obvious the time complexity is linear to the number of leaf nodes $O(l)$, while the time complexity of BLT is still $O(l^3)$. Though such great improvement resulted by level information in HBLT in this case cannot be taken as a general measurement of performance gain of HBLT over BLT, we can still see from this figure the huge benefits hop count can bring to the topology inference when many group members are not in the same level. To see the performance gain of HBLT over BLT in other cases, we compare the time complexities of both algorithms in a more general case in the following section.

b. Expected-case Time Complexity Comparison

In this section we take a sequence of different types of topologies between the best-case and worst-case tree topologies in order to compare both algorithms in a more general case, which we call expected-case time complexity. Note that the expected-case here is not the average-case in the strict sense because it is obtained by sampling the problem space to one of a manageable size. The strict average-case time complexity appears to be extremely difficult, if not impossible, to obtain due to the super-exponential number of trees one could construct considering different sizes and node distributions, and the difficulties in formulating all different trees in a manageable way. As will show below, we sample all trees according to their heights and widths of levels at different heights. Because this sample space covers all different heights and different widths at each level, the time complexity in our expected-case is a fair estimate to that of the average-case in the strict sense.

For a binary tree of n nodes, it is well known that the total number of all different binary trees is given by Catalan number. The number of structurally different binary trees can also be estimated. However, the time complexity analysis in our algorithm only relates to the height of the tree and number of nodes at each level, regardless of the tree's concrete structure. Even if trees have different structures, the time complexity of the algorithm for them remains the same if their heights and combination of nodes at all levels are the same.

Lemma 1 *The time complexity of HBLT depends only on the values of $a_i, 1 \leq i \leq h$, not on their order of appearance in the tree.*

Proof

$T_{HBLT}(a_1, a_2, \cdots, a_h) = a_1^2 + a_2^2 + \cdots + a_i^2 + \cdots + a_h^2$, therefore the value of T_{HBLT} is unchanged when a_i and a_j are swapped. □

Assume that the number of nodes at each level is an integer power of 2, $a_i = 2^k$. Let $l = 2^r$. Under this condition, there are still too many different binary trees to enumerate. But we can sample all trees based on lemma 1 and consider only leftist trees. In this way we can get tree topologies with different levels.

We begin with the best-case tree for HBLT, in which the height of the tree equals the total number of leaves, i.e., $h = l$. Then we fold the tree at the bottom level to reduce its height in bottom-up way, and make the width of the leftist tree as twice as before. Let i denote the number of foldings. Folding stops when the number of leaves at the bottom level under operation is maximized.

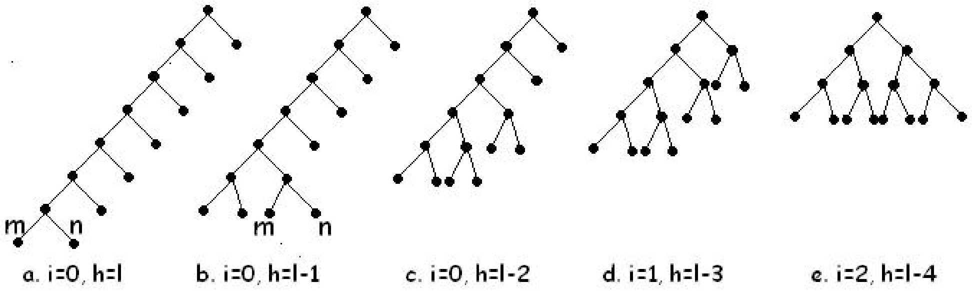


Figure 2.3: Multicast tree folding procedure

Figure 2.3 gives the folding procedure for 8 leaves. After twice of folding, the tree has been changed from the best case to the worst case. Folding is performed with the level number reduced by 1 each time. There are many possibilities for constructing a binary tree every time when the height is decreased by 1. However, as shown in Lemma 1, the time complexities remain the same for all such trees. From Figure 2.3(b), it is obvious no matter whose children nodes m and n are, the tree has the same time complexity. So it's unnecessary to compute time complexity for every such tree. When the tree is folded to the structure in Figure 2.3(d), it is called first (phase) folding and i equals to 1. The second folding ($i = 2$) transforms the tree in Figure 2.3(d) to that in Figure 2.3(e). Iterating the folding process to reduce the height, a complete binary tree is constructed as shown in Figure 2.3(e). For every case in the procedure, time complexities are calculated. Then the total time complexity divided by the total number of trees will give the expected-case time complexity.

Denote the total number of trees sampled in this way by TN, the total execution time by TT, and the expected-case time complexity of HBLT by $\widetilde{T}_{HBLT}$.

$$TN = \sum_{i=0}^{r-2}(2^{r-i-1} - 1) = 2^r - r - 1 = l - \log_2 l - 1 \tag{2.7}$$

$$\begin{aligned} TT &= \sum_{i=0}^{r-2} \sum_{h=2^{r-(i+1)}+i+1}^{2^{r-i}+i-1} [(2^{r-i} + i - h) \cdot 2^{2(1+(i+1))} + \sum_{k=1}^{h-(2^{r-i}+i-h)-(i+1)} 2^{2(i+1)} + \sum_{k=2}^{i} 2^{2k}] \\ &= 3l^2 \log_2 l - \frac{13}{2}l^2 + 6l + \frac{4}{3}r + \frac{4}{9} \end{aligned} \tag{2.8}$$

$$\widetilde{T}_{HBLT} = TT/TN = O(l \cdot \log_2 l) \tag{2.9}$$

Since the time complexity of BLT is only relevant to the number of leaf nodes, all these sampled trees have the same number of leaves, the expected-case time complexity of BLT $O(l^3)$, which is far worse than HBLT.

Although the above does not consider all binary trees, but binary trees included in calculation represent all cases influencing much to the performance of HBLT from the best case to the worst case. Therefore, the expected-case time complexity calculated on such sampled tree topologies represents a more general case. The analysis on this more general case can demonstrate a major performance advantage of HBLT over BLT in algorithm efficiency.

Hence, we have

Theorem 1 *The algorithm of binary loss tree classification with hop count has a worst-case time complexity of $O(l^2)$ and expected-case time complexity of $O(l \log l)$.*

2. Analysis on Misclassification Probability

In this section, we describe a model of failures of HBLT, and analyze the probability of misclassification to compare the performance of both algorithms. Since HBLT proceeds by recursively classifying receivers, topology misclassification can be analyzed by looking at how the receivers are misclassified in the estimated topology $\widehat{T}$.

Definition 1 *Let (T, α) be a loss tree with $T = (V, L)$, and $(\widehat{T}, \hat{\alpha})$ be an inferred loss tree with $\widehat{T} = (\widehat{V}, \widehat{L})$. Denote the receiver set of i by $R_T(i)$ and let $R_T = \cup_i R_T(i)$. If $\forall i \in W = V \backslash (\{0, 1\} \cup R)$, $R_T(i) = R_{\widehat{T}}(\hat{i})$, the node i is said to be classified correctly.*

For a binary tree, the topology is correctly classified if and only if $\forall i \in W$ is correctly classified. So we can study topology misclassification by looking at the misclassification of receivers for each i. Both BLT and HBLT use $A(U)$, which is defined as the probability of successful probing to $a(U)$ (the nearest common ancestor of a set of nodes U), to classify siblings. It can also be written as $A(i, j)$ as Equation (2.2) of Section 2.2.1, where $U = (i, j)$. While for HBLT, one more measurement on hop count in addition to $A(U)$ is used to classify siblings, making the classification process more efficient than BLT. To analyze misclassification probability of HBLT, we firstly have a look at how $A(U)$ affects classification in the procedure of topology inference, which is along the same line as proposed in [30].

A general form for $A(U)$ — $A(S_1, S_2)$ is used to analyze misclassification because it reflects how each receiver descended from i influences the classification on i, where S_1 and S_2 are two non-empty disjoint subsets of R_T. Similar to previous definition on $A(i, j)$, $A(S_1, S_2)$ denotes the probability of successful transmission on the path from the root to the the nearest common ancestor of nodes in S_1 and S_2. When the nearest common ancestor of nodes in S_1 and the nearest common ancestor of nodes in S_2 are siblings, $A(S_1, S_2)$ is minimized.

$$\begin{aligned} A(S_1, S_2) &= \frac{P[\vee_{j \in S_1} X_j = 1] P[\vee_{j \in S_2} X_j = 1]}{P[\vee_{j \in S_1} X_j \cdot \vee_{j \in S_2} X_j = 1]} \\ &= \frac{\gamma(S_1)\gamma(S_2)}{\gamma(S_1) + \gamma(S_2) - \gamma(S_1 \cup S_2)}, \end{aligned} \tag{2.10}$$

where $\gamma(S) = P[\vee_{k \in S} \vee_{j \in R(k)} X_j = 1]$, which is approximated by $\hat{\gamma}(S)$ in practice.

$A(S_1, S_2)$ can be estimated by $A^{(n)}(S_1, S_2)$. For simplicity, we denote $A^{(n)}(S_1, S_2)$ by $\widehat{A}(S_1, S_2)$. Then we have

$$\begin{aligned} \widehat{A}(S_1, S_2) &= \frac{\sum_{i=1}^{n} \vee_{j \in S_1} Y_j^{(i)} \sum_{i=1}^{n} \vee_{j \in S_2} Y_j^{(i)}}{n \sum_{i=1}^{n} (\vee_{j \in S_1} Y_j^{(i)}) \cdot (\vee_{j \in S_2} Y_j^{(i)})} \\ &= \frac{\hat{\gamma}(S_1)\hat{\gamma}(S_2)}{\hat{\gamma}(S_1) + \hat{\gamma}(S_2) - \hat{\gamma}(S_1 \cup S_2)}, \end{aligned} \tag{2.11}$$

where $Y_k^{(i)} = X_k^{(i)} = \vee_{j \in d(k)} Y_j^{(i)} = \vee_{j \in R_T(k)} Y_j^{(i)}$, $Y_k^{(i)}$ is the loss measurement of node k for the ith probe packet, which is the result of "OR" operation on the loss measurements of all the leaves descended from node k. For $U \subset V$, $\hat{\gamma}(S) = n^{-1} \sum_{i=1}^{n} \vee_{j \in S} Y_j^{(i)}$ is the ratio of number of probe packets that reach any receiver descended from nodes in U to the total number of probe packets. As the number of probe packets n increases, $\widehat{A}(S_1, S_2)$ asymptotically approaches to $A(S_1, S_2)$, i.e., $\lim_{n\to\infty} \widehat{A}(S_1, S_2) = A(S_1, S_2)$.

Though HBLT classifies the siblings based on above estimation on $A(S_1, S_2)$ in Equation (2.10), it works in a different way from BLT due to the incorporation of level information. We now study the condition of correctly classifying a node i by HBLT according to Definition 1.

We assume that the height of the multicast tree is h, so the maximum value of hop count is h. Each node set with the same hop count value k is denoted by $H_k, 1 \le k \le h$. The coherence coefficient of H_k, denoted by ε_k, is defined to be the correlation among the nodes within H_k, $1 \le k \le h$. Coherence coefficients $\varepsilon_1, \varepsilon_2, ...\varepsilon_h$ are used to decide whether two nodes are siblings. Let G_i be the event that HBLT correctly classifies nodes in $R_T(i)$ for some $i \in W$. G_i happens if every descendent of i finds its sibling in HBLT inference correctly.

Let $D(S_1, S_2, S_3)$ be the difference between $A(S_1, S_2)$ and $A(S_1, S_3)$, and $\widehat{D}(S_1, S_2, S_3)$ the difference between $\widehat{A}(S_1, S_2)$ and $\widehat{A}(S_1, S_3)$. Define a set $S(i)$ to be

$$\begin{aligned} S(i) = \{(S_1, S_2, S_3) : a(S_1), a(S_2), a(S_3) \in H_k, 1 \le k \le h, S_1, S_2 \subset R_T(i), \\ S_3 \subseteq R_T \backslash R_T(i), S_p \neq \emptyset, p = 1, 2, 3, S_p \neq S_q, p \neq q\}. \end{aligned}$$

$S(i)$ contains all (S_1, S_2, S_3) used in HBLT for classification of all nodes in the subtree rooted at i. Thus we can obtain the following Lemma for HBLT to decide if node i is correctly classified. Since HBLT works in a different way from BLT, the sufficient condition is also different from the condition given in [30].

Lemma 2 *A sufficient condition for correct classification of i is that*

$$\widehat{D}(S_1, S_2, S_3) = \widehat{A}(S_1, S_3) - \widehat{A}(S_1, S_2) > 0 \tag{2.12}$$

and

$$\widehat{A}(S_1, S_2) < \varepsilon_k \tag{2.13}$$

for all $(S_1, S_2, S_3) \in S(i)$.

Lemma 2 shows that, to classify node i, HBLT needs to access all elements in $S(i)$, i.e. to compare those descendant node pairs of i whose ancestors are in the same level. In contrast, the BLT algorithm [30] requires to access all elements in set $S'(i)$ defined as follows:

$$S'(i) = \{(S_1, S_2, S_3) : S_1, S_2 \subset R_T(i),$$
$$S_3 \subseteq R_T \backslash R_T(i), S_p \neq \emptyset, p = 1, 2, 3, S_p \neq S_q, p \neq q\}.$$

That is, BLT needs to compare all possible pairs of nodes descended from i without any level-based constraint.

Clearly, $S(i) \subseteq S'(i)$. Therefore, to classify any node (and its subtree), the number of comparisons required by HBLT is significantly smaller than that required by BLT, especially in the case of large tree height (h).

Let $Q(S_1, S_2, S_3)$ be the event that Equations (2.12) and (2.13) hold, Q_i be the event that $Q(S_1, S_2, S_3)$ holds for all $S_1, S_2, S_3 \in S(i)$, i.e., $Q_i = \bigcap_{(S_1,S_2,S_3) \in S(i)} Q(S_1, S_2, S_3)$. Let $\bar{G}_i$ be the event that i is classified incorrectly, P_i^f be the misclassification probability on node i. Then similar to the deduction steps in [30], we can obtain the misclassification probability of HBLT by the following steps.

$$G_i \supseteq Q_i, \tag{2.14}$$

So, an upper bound for probability of classified i incorrectly can be expressed as

$$P_i^f = P[\bar{G}_i] \leq \sum_{(S_1,S_2,S_3) \in S(i)} P[\bar{Q}(S_1, S_2, S_3)]. \tag{2.15}$$

We denote G to be the event that the multicast tree is correctly classified, then

$$G \supseteq \bigcap_{i \in W} Q_i = \bigcap_{i \in W} \bigcap_{(S_1,S_2,S_3) \in S(i)} Q(S_1, S_2, S_3). \tag{2.16}$$

Consequently the misclassification probability of the inferred tree by HBLT is,

$$P_{HBLT}^f = P[\bar{G}] \leq \sum_{i \in W} \sum_{(S_1,S_2,S_3) \in S(i)} P[\bar{Q}(S_1, S_2, S_3)]. \tag{2.17}$$

Thus we have the following theorem to describe the distribution of $\sqrt{n} \cdot (\widehat{D}(S_1, S_2, S_3) - D(S_1, S_2, S_3))$. Because HBLT compares fewer pairs of nodes than BLT in each iteration, we use a parameter ξ to distinguish the comparison times of both algorithms. The exact meaning of ξ is given in the proof of Theorem 2.

Theorem 2 *For each $i \in W$ and all $((S_1, S_2, S_3) \in S(i))$, $\sqrt{n} \cdot (\widehat{D}(S_1, S_2, S_3) - D(S_1, S_2, S_3))$ converges in distribution to a Gaussian random variable with mean 0 and variance $\frac{\sigma^2(S_1,S_2,S_3)}{\xi}$ as the number of probe packets $n \to \infty$, where $D(S_1, S_3, S_3) = A(S_1, S_3) - A(S_1, S_2)$, $\xi \geq 2$.*

Proof From Theorem 10 of [30], we know that by use of BLT, for each $i \in W$ and all $((S_1, S_2, S_3) \in S'(i))$, $\sqrt{n} \cdot (\widehat{D}(S_1, S_2, S_3) - D(S_1, S_2, S_3))$ can be expressed as a standard normal distribution $N(0,\, \sigma^2(S_1, S_2, S_3))$ as $n \to \infty$. For every classification of siblings descended from i, S_1, S_2 are selected from subsets of all nodes which are descendent of i, and S_3 is selected from the subsets that do not belong to the subtree rooted at i. In HBLT, only the subsets $S(i)$ where $a(S_1)$, $a(S_2)$, and $a(S_3)$ are in the same level are sampled from the set $S'(i)$. ε_k acts as a virtual node $a(S_3)$, which makes Equation (2.12) hold after replacing S_3 with ε_k. That is, Equation (2.13) is consistent with Equation (2.12) by assuming ε_k to be a virtual node $a(S_3)$. Assume the number of elements in the set $S(i)$ is ξ, then $((S_1, S_2, S_3) \in S(i))$, $\sqrt{n} \cdot (\widehat{D}(S_1, S_2, S_3) - D(S_1, S_2, S_3))$ can be considered as the sampled values from the original variable $((S_1, S_2, S_3) \in S'(i))$, $\sqrt{n} \cdot (\widehat{D}(S_1, S_2, S_3) - D(S_1, S_2, S_3))$. The mean of the sampled set remains the same as that of the original normal distribution, which is 0. The variance of the sampled set is the original variance divided by the number of elements in sampled set, i.e., ξ.

□

By Theorem 2, we can approximate $P[\bar{Q}(S_1, S_2, S_3)]$ by $\Psi\big(-\sqrt{n} \cdot \sqrt{\xi} \cdot \frac{D(S_1,S_2,S_3)}{\sigma(S_1,S_2,S_3)}\big)$, Ψ is the CDF (cumulative density function) of the standard normal distribution. Therefore, for a great number of probe packets n, $P[\bar{Q}(S_1, S_2, S_3)]$ can be approximated as follows.

$$P[\bar{Q}(S_1, S_2, S_3)] \approx e^{-(n/2) \cdot \xi \cdot \frac{D^2(S_1,S_2,S_3)}{\sigma^2(S_1,S_2,S_3)}} \tag{2.18}$$

The logarithmic asymptotic of $P[\bar{Q}(S_1, S_2, S_3)]$ is

$$\log P[\bar{Q}(S_1, S_2, S_3)] \sim -\frac{n \cdot \xi \cdot D^2(S_1, S_2, S_3)}{2\sigma^2(S_1, S_2, S_3)} \tag{2.19}$$

From Equation (2.15) we know that P_i^f is dominated by $P_{\max}(\bar{Q}(S_1, S_2, S_3))$. That is, when $\frac{D^2(S_1,S_2,S_3)}{\sigma^2(S_1,S_2,S_3)} \mid_{(S_1,S_2,S_3) \in S(i)}$ is minimized, $P(\bar{Q}(S_1, S_2, S_3))$ is maximized. At this time P_i^f can be approximated by $P_{\max}(\bar{Q}(S_1, S_2, S_3))$, i.e., $P_i^f \approx P_{\max}(\bar{Q}(S_1, S_2, S_3))$. Because from Theorem 10 in [30], we can deduce that for HBLT, when the nearest common ancestor node of nodes in S_1 and S_3 is just node i's parent node $p(i)$, $\frac{D^2(S_1,S_2,S_3)}{\sigma^2(S_1,S_2,S_3)} \mid_{(S_1,S_2,S_3) \in S(i)}$ is minimized, and

$$\min_{(S_1,S_2,S_3) \in S(i)} \frac{D^2(S_1, S_2, S_3)}{\sigma^2(S_1, S_2, S_3)} = \bar{\alpha}_i + O(\parallel \bar{\alpha} \parallel^2), \tag{2.20}$$

where $\bar{\alpha}_i$ is the loss rate of link $(p(i), i)$, i.e., $\bar{\alpha}_i = 1 - \alpha_i$, $\parallel \bar{\alpha} \parallel = \max_{k \in V} \bar{\alpha}_k$. Thus, for great n and small $\parallel \bar{\alpha} \parallel$, P_i^f can be approximated as:

$$P_i^f \approx e^{-\bar{\alpha}_i \frac{n}{2} \xi} \tag{2.21}$$

From Equation (2.17) we know that P^f_{HBLT} is dominated by $\max_{i \in W} P^f_i$, that is, when

$$\bar{\alpha}^f = \min_{i \in W} \bar{\alpha}_i \tag{2.22}$$

holds. ξ at this time is the number of elements in the sampled set from the entire set of all nodes descended from node i whose link loss rate $\bar{\alpha}_i$ is the minimal loss rate in the network. Under this condition, the logarithm on P^f_{HBLT} is expected to be asymptotically linear on n with negative slope $\frac{\xi}{2} \cdot \bar{\alpha}^f$.

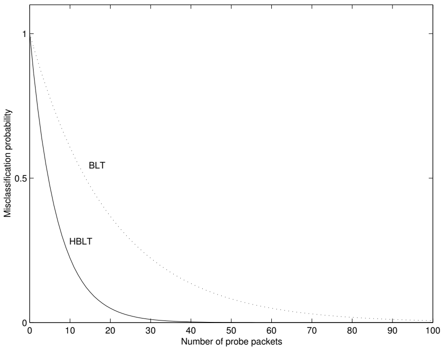


Figure 2.4: Misclassification probability comparison between BLT and HBLT when $\bar{\alpha}^f = 0.1$, $\xi = 3$.

Compared with the asymptotical slope of BLT, $\bar{\alpha}^f/2$, it is clear that HBLT has better performance because the misclassification probability of HBLT decreases more quickly than that of BLT as the number of probe packets increases, as depicted in Figure 2.4. According to different types of topologies and the link status in the multicast network, ξ may vary so it is a critical parameter that affects the misclassification probability of HBLT. If ξ is great, the misclassification probability of HBLT will decrease to 0 more quickly as the number of probe packets increases. If ξ is small, the misclassification probability will decrease to 0 slowly. As we mentioned above, ξ is greater than 2, henceforth HBLT can always infer a tree topology with a lower misclassification probability than BLT.

3. Analysis on Inference Accuracy

In this section, we compare the inferred trees by BLT and HBLT to analyze the inference accuracy. It is clear that the inferred tree by BLT contains only those nodes with two children in the physical tree, whereas that by HBLT contains nodes with both two children and one child in the physical tree. We say that the multicast tree is classified correctly by the algorithm if misclassification probability of an algorithm is 0. Considering the case that both algorithms classify correctly, the object trees inferred by BLT and HBLT may be very different from each other. From this view inference accuracy is discussed.

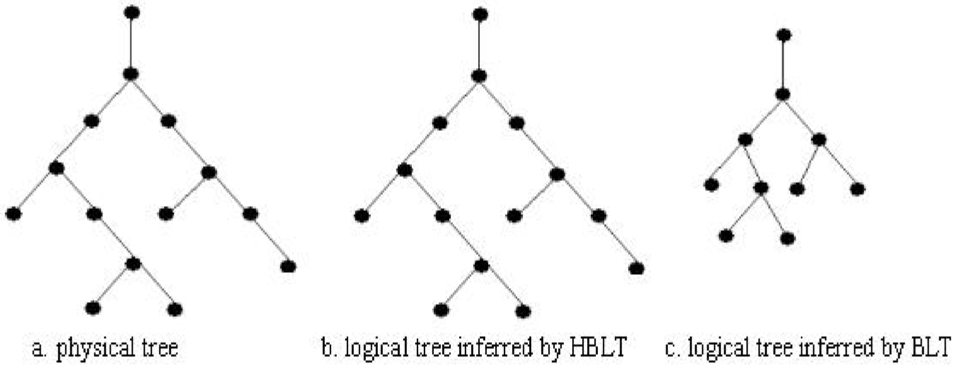

a. physical tree
b. logical tree inferred by HBLT
c. logical tree inferred by BLT

Figure 2.5: Comparison of logical trees inferred by BLT and HBLT in case of correct classification

Assume that both algorithms work on a binary tree network whose topology is depicted as the Figure 2.5(a). In the case of correct classification, the inferred tree by HBLT is the same as the physical tree as shown in Figure 2.5(b). The tree shown in Figure 2.5(c) is the inferred tree by BLT. It is obvious that the correctly inferred tree by BLT changes a lot in the structure from the original physical tree, whereas the correctly inferred tree by HBLT is the same as the original one. We now analyze their difference in inference accuracy.

Assume that the actual physical tree has n internal nodes. Every internal node has 1 child at probability δ, and 2 children at probability $1-\delta$. Let n_1 be the number of nodes that have 1 child, and n_2 be the number of nodes that have 2 children. Then we have $n = n_1 + n_2$. Denote by c_i the number of children of internal node i.

$$P\{c_i = x\} = \begin{cases} \delta & x = 1 \\ 1-\delta & x = 2 \end{cases} \tag{2.23}$$

$$\begin{aligned} E(n_1) &= \sum_{x=1}^{2} P\{c_i = x\} \cdot E[n_1 \mid c_i = x] \\ &= P\{c_i = 1\} \cdot E[n_1 \mid c_i = 1] + P\{c_i = 2\} \cdot E[n_1 \mid c_i = 1] \\ &= n \cdot \delta \end{aligned} \tag{2.24}$$

When all internal nodes have 1 child, $E[n_1 \mid c_i = 1]$ is n; if all internal nodes have 2 children, $E[n_1 \mid c_i = 2]$ equals 0.

Definition 2 *For a multicast tree with n internal nodes, the inference accuracy of an algorithm is defined as the number of inferred internal nodes divided by the number of actual internal nodes in the case of correct classification.*

Clearly, the expected number of internal nodes that have single child is $n\delta$. In BLT all these single-child nodes are deleted. It only infers all those internal nodes that have two children. Henceforth, the BLT's inference accuracy is

$$\frac{n - E(n_1)}{n} = 1 - \delta \tag{2.25}$$

Suppose that $\delta = 1/2$, a half of internal nodes will then fail to be identified. The inference accuracy of BLT in this case is 1/2. As δ increases, i.e., the probability of each internal node with single child increases, the inference accuracy of BLT will decrease. If at a large probability each internal node has single child, the inferred tree by BLT will be very different from the original physical tree. The inference accuracy of BLT is thus very low in such case. However, if $\delta = 0$, that is, a physical multicast topology does not contain any single-child nodes, BLT can also infer an accurate topology as HBLT does.

For HBLT, the inference accuracy is irrelevant to δ, so it can identify all the internal nodes regardless of the number of children they have. The inference accuracy is therefore always equal to 1 if only the number of nodes is taken into account according to Definition 2.

2.2.4 Experimental Results

In this section we verify the benefits from hop count measurement by simulating both HBLT and BLT. We show simulation results in several different types of networks with different internal link status, including a random tree without single-child node, a random tree with single-child nodes and a balanced full binary tree. To compare the performance of HBLT and BLT, we also define a similarity degree function to describe how the inferred tree is close to the real physical tree.

Firstly, we simulate both algorithms in the network topology shown in Figure 2.6. Node 0 is the sender, nodes $1 - 10$ are receivers. All internal links are configured with different capacities ranging from $0.1Mbit/s$ to $5Mbit/s$. The link $0 \rightarrow 1$ is set at $10Mbit/s$. The root node 0 generates probe packets in a 20K-1Mbit/s stream. Every probe packet comprises one UDP packet with 1000bytes.

By sending different numbers of probe packets, we can obtain the inferred topologies using HBLT and BLT. To compare how the inferred topologies are close to the original physical topology by HBLT and BLT, we define the similarity degree function as follows:

Definition 3 *Define $SimilarityDegree = \alpha \cdot s + \beta \cdot h$, where s denotes the ratio of the number of nodes whose siblings are identified correctly to the total number of nodes, h denotes the ratio of the number of nodes whose hop levels are inferred accurately to the total number of nodes. α, β are the weight of these two factors. When at least one of the two subtrees of a node's sibling is the same as that of the physical tree, we say that the node is inferred correctly.*

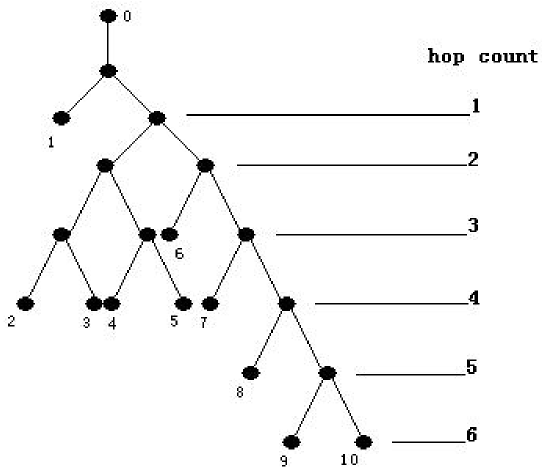
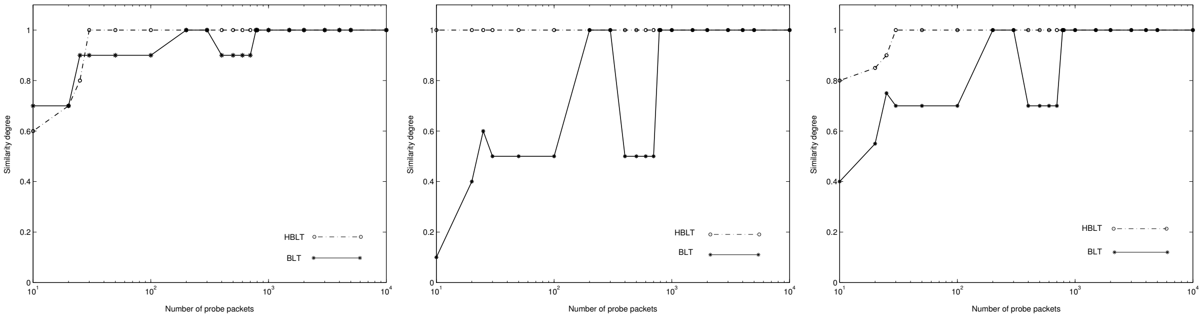


Figure 2.6: A multicast network without single-child nodes

Then the results on similarity degree comparison between HBLT and BLT are obtained as follows.



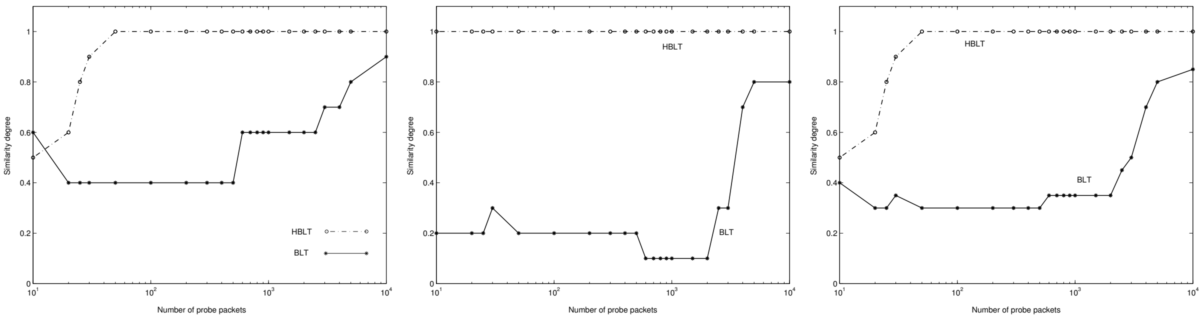
(a) $\alpha = 1, \beta = 0$ (b) $\alpha = 0, \beta = 1$ (c) $\alpha = 0.5, \beta = 0.5$

Figure 2.7: Similarity degree comparison of HBLT and BLT in a network with internal loss rates ranging from 0.197 to 0.683

C1)

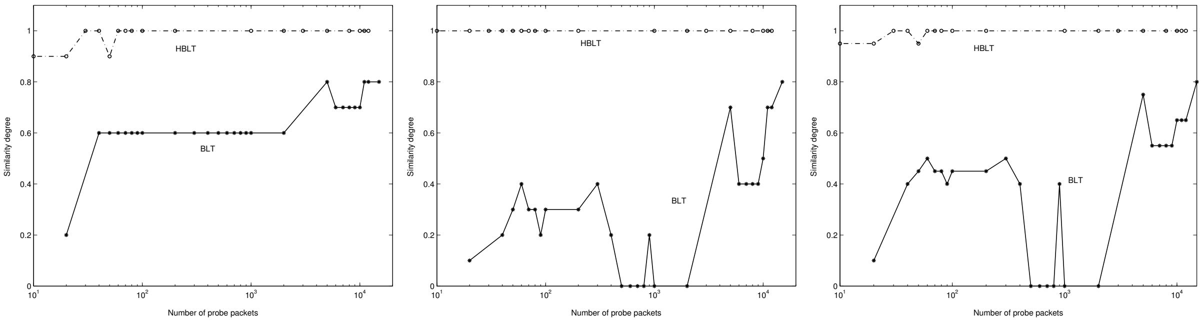
From Figures 2.7(a)-2.9(c), we can see HBLT can infer a correct topology with around 100 probe packets. As the internal link status varies, the number of probe packets required by HBLT to obtain a correct topology is just a bit different. However, BLT needs thousands of probe packets to infer a similar topology to the original one. Moreover, the performance of inferred topology by BLT is severely influenced by the internal link status.

We now compare the performance of HBLT and BLT in a large-scale network as shown in Figure 2.10(a). From Figure 2.10(b) we can find that HBLT can infer the correct topology with



(a) $\alpha = 1, \beta = 0$ (b) $\alpha = 0, \beta = 1$ (c) $\alpha = 0.5, \beta = 0.5$

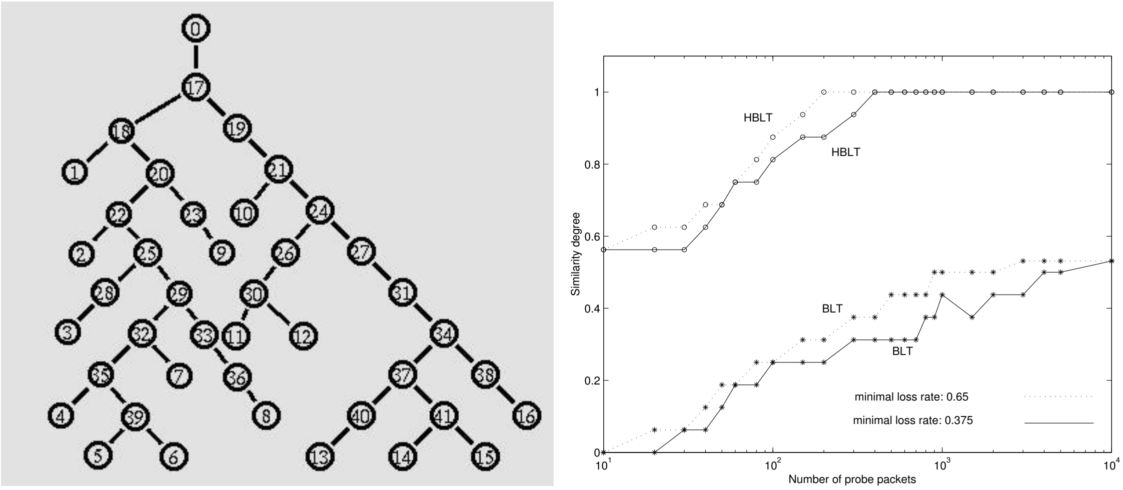
Figure 2.8: Similarity degree comparison of HBLT and BLT in a network with internal link loss rates ranging from 0.495 to 0.792



(a) $\alpha = 1, \beta = 0$ (b) $\alpha = 0, \beta = 1$ (c) $\alpha = 0.5, \beta = 0.5$

Figure 2.9: Similarity degree comparison of HBLT and BLT in a network with internal link loss rates ranging from 0.132 to 0.457

a much smaller number of probe packets than BLT. Due to the existence of single-child nodes in this network, we can see BLT cannot infer a correct topology with even 10000 probe packets because it cannot infer those single-child nodes and links connecting to them. From Figure 2.10(b), we can also find that in a network with more loss, HBLT and BLT can work with higher performance. This can be explained by lower misclassification probability if minimal loss rate is larger in Section 5.



(a) The tree topology with 42 nodes

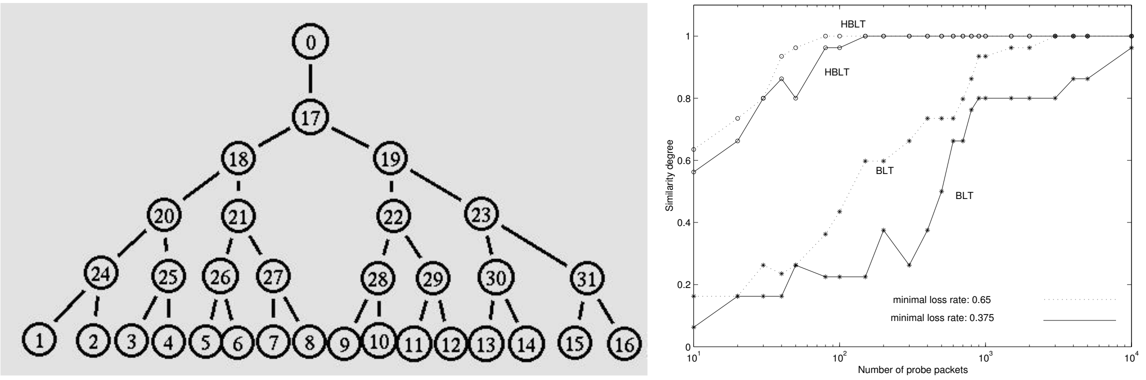
(b) $\alpha = 0.5, \beta = 0.5$

Figure 2.10: Similarity degree comparison between HBLT and BLT on a random binary tree

In a network with a balanced full binary tree topology as Figure 2.11(a) where hop count information may not be a great help as in other types of tree topologies, we compare the performance of both algorithms. From the result in Figure 2.11(b), we find HBLT can also work out a correct topology with a smaller number of probe packets.

All the above simulation results show that incorporating hop count into inference can yield exciting improvement. It avoids misclassification between nodes with different values of hop count which may have very similar loss measurements. The performance of BLT is significantly worse than that of HBLT because such case exists prevalently. Thus, the misclassification probability of HBLT is much lower than that of BLT, which is consistent with the analytical result in Section 2.2.3. In a large-scale network, more nodes with different values of hop count may have similar loss measurements. In this case, BLT works with a high misclassification probability and low efficiency. Henceforth, we can conclude that HBLT will work out more accurate and efficient multicast network topologies than BLT for large-scale networks in practice which validates the analyzes in above section.

We should note that hop count information obtained by TTL may not be always correct. Wrong information in hop count will affect the performance of HBLT. However, through simu-



(a) The tree topology with 32 nodes

(b) $\alpha = 0.5, \beta = 0.5$

Figure 2.11: Similarity degree comparison between HBLT and BLT on a balanced full binary tree

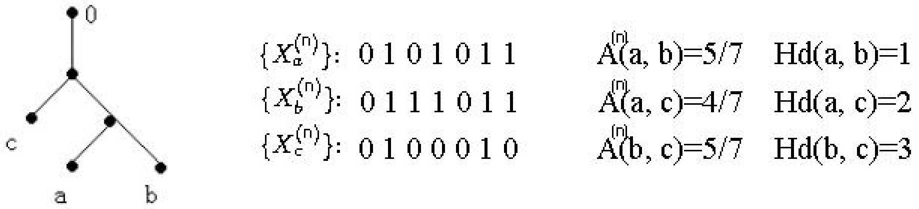
lation, we find that hop count information is always right in the small scale network in Figure 2.6. In the large scale networks as Figures 2.10(a) and 2.11(a), error information in hop count occurs in very few cases. Because we send many probe packets, we can get hop count information every time a probe packet is received at a receiver. It may happen to be wrong, but it will be corrected when next probe packet is received. Therefore, we calculate the hop count information by reading from several received probe packets, and we can therefore always obtain correct information.

2.3 Hamming Distance-based Multicast Topology Inference

Though both analysis and experimental results show the improvement on the performance of HBLT, it fluctuates as the number of probe packets increases when the scale of multicast network is large. Especially when the number of probe packets is small the performance of HBLT shows obvious instability. The performance of BLT also has this disadvantage. This is because both of them estimate the loss rate by limited probe packets as shown in Equations (2.3), which we called A-approach as mentioned in Section 2.2.1. The violation is inevitable and might be great when the number of probe packets is small. Thus, we would first give an example to explain this disadvantage clearly, then propose a new sibling classification approach for siblings classification, which is called hamming distance approach. Thereafter, comparison of inference accuracy between hamming distance and A-approach will be given. Incorporating hamming distance approach with hop count measurements which have been proven to be effective and efficient in Section 2.2.3, we get a new algorithm called Binary Hamming Distance Classification (BHC) algorithm. Finally we will justify the advantage of the new algorithm by experimental results.

2.3.1 Existing Approach for Siblings classification

In the previous work of [30,33,34,65], identification of siblings by A-approach plays the key role in network topology inference. In A-approach, $A(i,j)$ represents the product of the probabilities of successful transmission on each link between the root 0 to the nearest common ancestor node of receivers i and j. Each probability is estimated with a limited number of probe packets. With a finite number of probe packets, the estimation may cause mistakes. Figure 2.12 gives a simple example of determining whether two nodes are siblings with 7 probe packets.



$\{X_a^{(n)}\}$:	0 1 0 1 0 1 1	$A^{(n)}$(a, b)=5/7	Hd(a, b)=1
$\{X_b^{(n)}\}$:	0 1 1 1 0 1 1	$A^{(n)}$(a, c)=4/7	Hd(a, c)=2
$\{X_c^{(n)}\}$:	0 1 0 0 0 1 0	$A^{(n)}$(b, c)=5/7	Hd(b, c)=3

Figure 2.12: Comparison of Hamming distance($H_d(\cdot,\cdot)$) and the previous A-approach ($A^{(n)}(\cdot,\cdot)$) for each pair node

According to the estimated value $A^{(n)}(i,j)$ of Equation (2.13), if receiver j loses many probe packets, the value of $A^{(n)}(i,j)$ will be very small, even smaller than the value between i and its actual sibling, e.g., $A^{(n)}(a,c) < A^{(n)}(a,b)$, $n = 7$ in Figure 2.12. Thus, A-approach will misclassify nodes a and c as siblings. Even when the number of probe packet increases, such bias cannot be eliminated completely. Because the estimated probability in Equation (2.13) does not equal to the true probability unless the number of probe packets is infinite. In order to reduce the bias with a finite number of probe packets, we propose our hamming distance classification approach.

2.3.2 Hamming Distance Classification Approach

The hamming distance between nodes u and v is defined as the number of different bits between their sequences, which is given in the following equation, where "$\oplus$" is the exclusive-OR operator.

$$H_d(u,v) = \sum_{m=1}^{n} X_u^{(m)} \oplus X_v^{(m)}. \tag{2.26}$$

For instance, the hamming distances of each pair of sequences in Figure 2.12 are computed. Different hamming distances between different node pairs can be used to determine which pair of nodes are siblings. As illustrated in Figure 2.12, $H_d(a,c) > H_d(a,b)$ is congruent with the relationship between the nodes because nodes a and b are siblings. However, as we have discussed in Section 2.3.1, A-approach failed to identify the relationship of nodes a, b and c with 7 probe packets in Figure 2.12. The hamming distance approach distinguishes the siblings and non-siblings with different values successfully. However, $A^{(n)}(i,j)$ does not give the differences between the pair of sequences with 7 probe packets in Figure 2.12. We will discuss the reason that our hamming distance approach is superior to A-approach in Section 2.3.3.

In the multicast network, the nearer two nodes are located, the more similar two “0-1” sequences they maintained are. The reason for such phenomenon is that the probe packets from the root to the receivers may pass many common links. If the receivers are siblings, the paths the probe packets pass from the root to their parent nodes are the same. Therefore, the correlation of two “0-1” sequences between a node and its siblings is greater than that of all other pairs of sequences between the node and its non-sibling node.

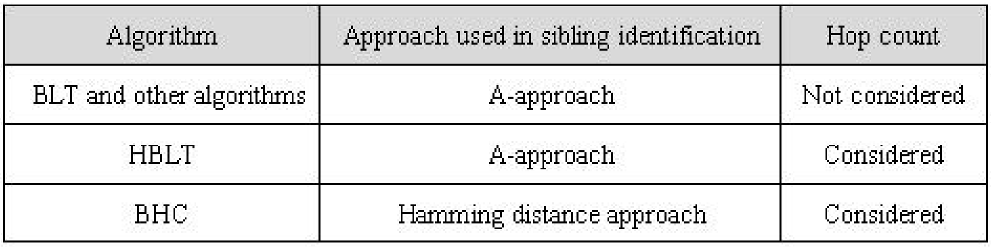
In order to infer the topology of the multicast network, all the siblings need to be identified correctly. That is, we should find out the similarity of each pair of sequences. The problem of identifying siblings in the multicast network can be stated as marking out the similarity and dissimilarity of all pairs of bit sequences. For a bit sequence, hamming distance is the simplest and most efficient method to identify the similarity and dissimilarity among different sequences. Therefore, we use hamming distance to identify siblings for multicast network topology inference.

It should be noted that our proposed approach generally works well regardless of temporal correlation between the losses in actual networks. No matter when we send the probe packets and how many probe packets we use to infer the network topology, the hamming distance approach can classify siblings correctly. In contrast, the previous A-approach is affected strongly by the temporal correlation of collected data. Different data collected in different time periods may cause different siblings classifications, especially when a small number of probe packets is used.

2.3.3 Binary Hamming Distance Classification (BHC) algorithm

We present the BHC algorithm based on the hamming distance classification approach in this subsection. BHC infers the multicast topology from the node with the maximum hop count and proceeds in a bottom up fashion. Since true siblings must have the same value of hop count, the set of all receivers is classified to different sets according to their values of hop count at the beginning. In the procedure of identifying siblings, only comparison between the sets with the same value of hop count is necessary, which in turn, avoids redundant comparisons between nodes with different values of hop count. For the nodes with the same value of hop count, a node pair is identified as siblings if the hamming distance is not only minimal among all hamming distances of all node pairs in the node set, but also less than a given threshold. Thus, by repeatedly identifying each sibling pair from the node set with the maximal hop count to the node set with minimal hop count, the multicast tree can be reconstructed.

Figure 2.13 shows the characteristics of the BHC algorithm and its main differences from other algorithms.



Algorithm	Approach used in sibling identification	Hop count
BLT and other algorithms	A-approach	Not considered
HBLT	A-approach	Considered
BHC	Hamming distance approach	Considered

Figure 2.13: Differences between BHC and HBLT, BLT

The detailed BHC algorithm is similar to HBLT in Section 2.2.2. Same as HBLT in [65], we associate each node v with a hop count $v.hop$. The v.hop values of leaf nodes can be obtained by simply reading the TTL values of the probe packets. For internal nodes, their v.hop values can be computed by degression from the v.hop values of leaf nodes during the topology inference procedure. For the nodes with the same value of hop count, a node pair is identified as siblings if the hamming distance is not only minimal among all hamming distances of all node pairs in the node set, but also less than a given threshold. The algorithm is run in bottom up fashion as HBLT. Thus, the BHC algorithm runs similar procedure of HBLT in Section 2.2.2 by replacing the following steps.

```
7.    Let $u$ be the first element in $W_e$; search for $v \in W_e$ to minimize $H_d(u, v)$,
      $(u \neq v)$;
8.    if $H_d(u, v) > \varepsilon_e$ then $S = \{u\}$, Set $r$ to be $u$'s virtual parent node;
```

Firstly all the receivers are classified into different node sets W_m $(1 \leq m \leq h)$ according to their values of hop count. Inference begins with identifying siblings in the node set with the maximum value of hop count. The hamming distances of each node pair in W_m are calculated. The node pair is identified to be siblings if its hamming distance is minimal and less than the threshold. Remove the siblings from the node set with the hop count being m and add the parent node into the node set with hop count reduced by 1. The “0-1” sequence of the parent node is obtained by “OR” operation of those of the siblings. When all nodes in W_m are grouped decrease hop count value by 1. Repeat the same procedure among the nodes in the node set W_{m-1}. The algorithm ends when the hop count becomes 1.

It should be noted that Levenshtein distance [6] may also be used for determining the similarity and dissimilarity between two strings. While Hamming distance is used for strings of the same length, Levenshtein distance is defined for strings of arbitrary lengths and hence more sophisticated. Levenshtein distance can be regarded as a generalization of the hamming distance. In our BHC algorithm, the bit sequences required to be compared are of the same length because n probe packets are sent and each node in the network has a record about whether each probe packet has been received or not — if the i-th probe packet is received by a node, the corresponding digit in the bit sequence of this node is 1, otherwise it is 0. Clearly, hamming distance suffices for our requirement. We therefore choose to use hamming distance for simplicity and efficiency.

2.3.4 Analysis on Inference Accuracy of Hamming Distance Classification Approach

BHC, HBLT and BLT can all get the consistent result with the real multicast network as the number of probe packets increases to infinity. However, with a finite number of probe packets, the BHC algorithm can obtain accurate results with a higher probability than HBLT and BLT, because the hamming distance is found to be superior to $A^{(n)}(i,j)$ used in [30, 33, 34, 65] in many cases.

Definition 4 *Let s_1 and s_2 be two nodes that have a common parent node i, s_3 be a node for which node i is not its parent, W_k is the node set with the hop count k. Define $s(i)$ as follows:*

$$s(i) = \{(s_1, s_2, s_3) : \forall s_1, s_2, s_3 \in W_k, 1 \le k \le h\}$$

Definition 5 *For $(s_1, s_2, s_3) \in s(i)$, let $D_H(s_1, s_2, s_3)$ be the difference of hamming distance between non-siblings and siblings, and $D_A(s_1, s_2, s_3)$ be the difference of $A^{(n)}(\cdot,\cdot)$ between non-siblings and siblings. That is,*

$$D_H(s_1, s_2, s_3) = H_d(s_1, s_3) - H_d(s_1, s_2),$$

$$D_A(s_1, s_2, s_3) = A^{(n)}(s_1, s_3) - A^{(n)}(s_1, s_2).$$

Lemma 3 *The sufficient conditions for correctly identifying nodes s_1 and s_2 as siblings are $0 < min_{(s_1,s_2,s_3)\in s(i)} D_H(s_1, s_2, s_3)$ and $H_(s_1, s_2) < \varepsilon_e$. For A-approach, the same condition must be satisfied by replacing $D_H(s_1, s_2, s_3)$ with $D_A(s_1, s_2, s_3)$, and $H_(s_1, s_2)$ with $A^{(n)}(s_1, s_2)$.*

Lemma 3 holds because the hamming distance or $A^{(n)}(\cdot,\cdot)$ of a node and its non-sibling nodes should be greater than that of it and its siblings.

We denote by $n^1_{s_i}$ the number of probe packets transmitted from the root to node s_i successfully, and by $n^1_{s_i s_j}$ the number of probe packets transmitted successfully from the root to both nodes s_i and s_j at the same time, $i = 1, 2, 3$.

Lemma 4 *For $(s_1, s_2, s_3) \in s(i)$, if inequality (2.27) holds, the hamming distance approach can identify the siblings while $A^{(n)}(\cdot,\cdot)$ cannot; if inequality (2.28) holds, $A^{(n)}(\cdot,\cdot)$ can identify the siblings while the hamming distance approach cannot; in all other cases, both approaches can identify siblings correctly.*

$$\frac{1}{2}(n^1_{s_2} - n^1_{s_3}) < n^1_{s_1s_2} - n^1_{s_1s_3} \leq \frac{n^1_{s_1s_3}}{n^1_{s_3}}(n^1_{s_2} - n^1_{s_3}), \tag{2.27}$$

$$\frac{n^1_{s_1s_3}}{n^1_{s_3}}(n^1_{s_2} - n^1_{s_3}) < n^1_{s_1s_2} - n^1_{s_1s_3} \leq \frac{1}{2}(n^1_{s_2} - n^1_{s_3}), \tag{2.28}$$

Proof

Since $H_d(s_1, s_2) = n^1_{s_1} + n^1_{s_2} - 2n^1_{s_1s_2}$ and $A^{(n)}(s_1, s_2) = \frac{n^1_{s_1} \cdot n^1_{s_2}}{n \cdot n^1_{s_1s_2}}$, we have,

$$\begin{aligned} D_H(s_1, s_2, s_3) &= n^1_{s_1} + n^1_{s_3} - 2n^1_{s_1s_3} - (n^1_{s_1} + n^1_{s_2} - 2n^1_{s_1s_2}) \\ &= (n^1_{s_3} + 2(n^1_{s_1s_2} - n^1_{s_1s_3})) - n^1_{s_2}, \end{aligned} \tag{2.29}$$

and

$$\begin{aligned} D_A(s_1, s_2, s_3) &= \frac{n^1_{s_1} \cdot n^1_{s_3}}{n \cdot n^1_{s_1s_3}} - \frac{n^1_{s_1} \cdot n^1_{s_2}}{n \cdot n^1_{s_1s_2}} \\ &= \frac{n^1_{s_1}}{n \cdot n^1_{s_1s_2}} \cdot ((n^1_{s_3} + \frac{n^1_{s_3}}{n^1_{s_1s_3}} \cdot (n^1_{s_1s_2} - n^1_{s_1s_3})) - n^1_{s_2}). \end{aligned} \tag{2.30}$$

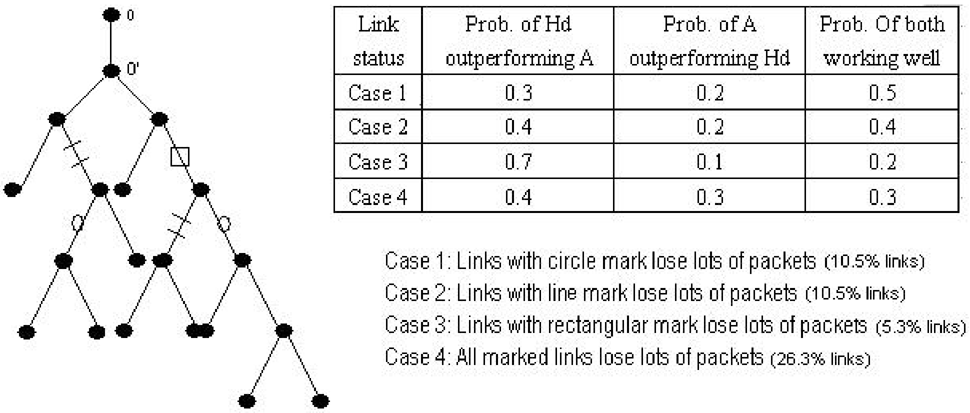
From Lemma 3, we know that nodes s_1 and s_2 will be identified as siblings if $D_H(s_1, s_2, s_3) > 0$ using the hamming distance approach for any $(s_1, s_2, s_3) \in s(i)$. If (s_1, s_2, s_3) results in $D_H(s_1, s_2, s_3) < 0$, the hamming distance approach will fail to identify s_1 and s_2 as siblings correctly. Similar conditions holds for $A^{(n)}(\cdot,\cdot)$ approach. Therefore, we can conclude that if any $(s_1, s_2, s_3) \in s(i)$ results in $D_A(s_1, s_2, s_3) < 0$ while $D_H(s_1, s_2, s_3) > 0$, the hamming distance approach is superior to the $A^{(n)}(\cdot,\cdot)$ approach in siblings identification, and vice versa. Lemma 4 describes the complete conditions based on equations (2.29) and (2.30). □

Lemma 4 shows when the hamming distance approach or A-approach can identify nodes s_1 and s_2 as siblings correctly. If the probability that inequality (2.27) holds is greater than the probability that inequality (2.28) holds, the hamming distance approach works better than previous A-approach. However, obtaining the exact probabilities for inequality (2.27) and (2.28) to hold is very difficult because both probabilities vary with different network connections and conditions. Thus, we only analyze the cases that inequality (2.27) holds and inequality (2.28) holds respectively and give an example through which we can have a clear view on which approach has a better performance in inference accuracy.

Firstly we should note that in most cases, both the hamming distance approach and A-approach can identify siblings correctly according to our analysis on the different receiving cases of nodes s_1, s_2 and s_3. When links s_1 and s_2 are in similar conditions, the hamming distance approach can identify siblings correctly if A-approach can do so. However we also find that sometimes A-approach cannot identify siblings correctly while the hamming distance approach can. When both nodes s_1 and s_2 receive almost all probe packets while node s_3 loses many probe packets, and $n^1_{s_1 s_3}$ happens to be equal to $n^1_{s_3}$, the hamming distance approach can identify siblings correctly while A-approach cannot. The above is also true when both nodes s_1 and s_2 lose many probe packets while s_3 receives almost all probe packets, and $\frac{n^1_{s_1 s_3}}{n^1_{s_3}} < \frac{1}{2}$. In very few cases, the hamming distance approach may not identify siblings correctly while A-approach can. This might happen only when links s_1 and s_2 are in different conditions which result in dissimilar sequences on nodes s_1 and s_2. In a multicast network, only a few siblings links may exhibit completely different performances among all siblings-link pairs. Even if under this condition, it can be noted that in most cases the hamming distance approach can identify siblings correctly as A-approach does. Thus from all cases discussed, we can see that the occurrence of the hamming distance approach outperforming A-approach is more frequent than that of the opposite situation.

As to the exact probabilities that inequalities (2.27) and (2.28) hold, we suppose a multicast network with 5%-27% links losing packets severely where the ratios can represent most cases in the real multicast networks. Less than 5% links suffering severe packet losses only occurs in some applications. Multicast networks with more than 10% links losing packets often result in unacceptable performance because most group receivers can be affected not to receive packets accurately. Therefore, the multicast network with 5%-27% links losing packets severely as shown in Figure 2.14 choose reasonable ratios of ill-performing links to represent general cases in practice. Assume other links experiences some loss and congestion to different degree. Different cases are discussed in this network and the probabilities of inequality (2.27) and (2.28) holding in each case are calculated respectively. We find that the probabilities vary with different link status. As we have discussed before, the probabilities also vary with different network topologies. Figure 2.14 illustrates clearly that the probability that the hamming distance approach succeeds but A-approach fails is greater than the probability in the opposite situation for this network. This conclusion may be extended to any network in general case. The simulation in the network given in Section 2.3.4 also supports this conclusion.

Though the probabilities that inequalities (2.27) and (2.28) hold are different as the network condition varies, we have seen from the above analysis that with a finite number probe packets, the hamming distance approach is more likely to work out the accurate topology than A-approach. In another word, due to the greater probability of the hamming distance approach outperforming A-approach in siblings identification, we may conclude that the use of the hamming distance approach in the BHC algorithm can result in a better performance than use of



Link status	Prob. of Hd outperforming A	Prob. of A outperforming Hd	Prob. Of both working well
Case 1	0.3	0.2	0.5
Case 2	0.4	0.2	0.4
Case 3	0.7	0.1	0.2
Case 4	0.4	0.3	0.3

Case 1: Links with circle mark lose lots of packets (10.5% links)
Case 2: Links with line mark lose lots of packets (10.5% links)
Case 3: Links with rectangular mark lose lots of packets (5.3% links)
Case 4: All marked links lose lots of packets (26.3% links)

Figure 2.14: Comparison on probabilities that inequality (2.27) or (2.28) holds in a certain multicast network

A-approach in inference accuracy in our tested networks.

2.3.5 Experimental Results on the BHC Algorithm

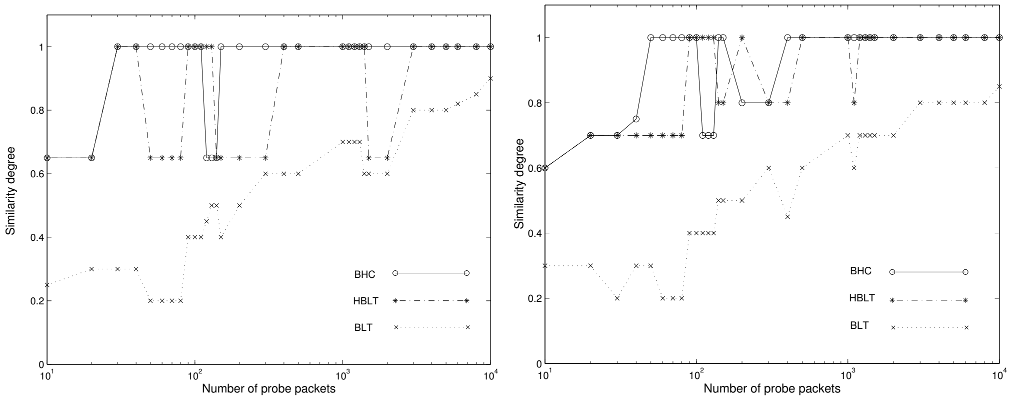
In this section, we validate the BHC algorithm by comparing it with HBLT and BLT. As given above, BLT provides a good combination of inference accuracy and computational efficiency comparing among many other topology inference algorithms in [30, 33, 34]. HBLT in Section 2 applies A-approach as BLT, however, it incorporates hop count into topology inference. We show the comparison result among these three algorithms in this subsection, which can be used to justify the superiority of the BHC algorithm sufficiently.

We implement all the algorithms in the network topology shown in Figure 2.6 of Section 2.2.4 by network simulator ns. Same as the configuration in Section 2.2.4, node 0 is the sender, nodes $1-10$ are receivers. All internal links are configured with different capacities ranging from $0.1Mbit/s$ to $5Mbit/s$. The link $0 \rightarrow 0'$ is set at $5Mbit/s$. The root node 0 generates probe packets in a 20K—2Mbit/s stream. Every probe packet comprises one UDP packet with 1000bytes.

Here we still use similarity degree of Definition 3 of Section 2.2.4 to describe how close the inferred topology is to the original physical network.

Let $\alpha = 0.5$ and $\beta = 0.5$, we get the compared results between three algorithms as Figure 2.15(b).

Figures 2.15(a) and 2.15(b) are obtained by changing different links' capacities such that the statistical loss rates of all links ranges from 0.053 to 0.572 which is different from configuration for simulation in Section 2.2.4. Both simulation results show that BHC requires fewer probe packets than HBLT and BLT to infer the accurate topology constantly. BLT shows the lowest



(a)

(b)

Figure 2.15: Similarity degree comparison among BLT, HBLT and BHC when $\alpha = 0.5, \beta = 0.5$

efficiency in topology inference. Since both HBLT and BLT applies A-approach, we can conclude that hamming distance classification approach used in BHC helps to infer the topology more efficiently and accurately.

Figures 2.15(a) and 2.15(b) also validates our analysis on siblings identification by the hamming distance approach and A-approach. It shows that the occurrence of that the hamming distance approach outperforming A-approach is more frequent than that of the opposite situation. This supports the conclusion we have drawn in Section 2.3.4, that is, the probability that hamming distance approach outperforming A-approach should be greater than the probability of the opposite situation. This also indicates why BHC works out better performance than HBLT and BLT in our simulated network.

In the followed simulation, we compare the performance of the BHC algorithm with previous algorithms in a larger network as show in Figure 2.10(a). From Figure 2.16, it is clear that BHC can infer the correct topology with the least number of probe packets.

2.4 Multicast Network Internal Loss Performance Inference

Above we introduce multicast-based network tomography for topology. Followed we will go into multicast-based network tomography for link-level loss performance.

When we infer the multicast network topology by the either HBLT or BHC, a “0-1” sequence is obtained for each node in the network including those internal nodes. We propose an approach to infer the loss rate for each link simultaneously when inferring multicast topology, taking advantage of these “0-1” sequences.

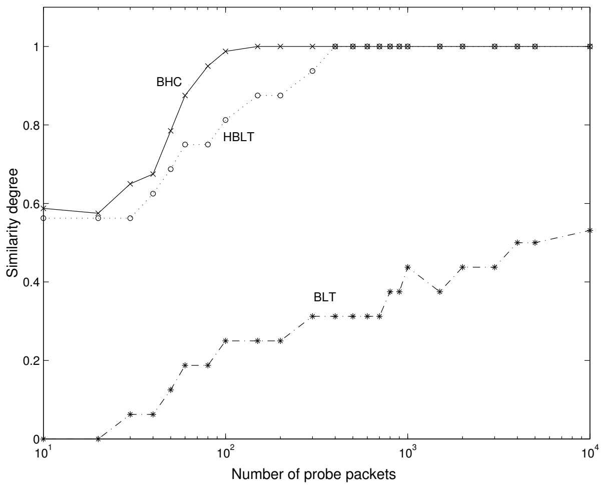


Figure 2.16: Similarity degree comparison among BLT, HBLT and BHC when $\alpha = 0.5, \beta = 0.5$ in the network of Figure 2.10(a) (minimal loss rate is 0.375)

2.4.1 Approach to loss performance inference

Multicast networks in the binary tree form are firstly considered. According to the BHC algorithm, we know the sequence of internal nodes can be deduced by the following equation.

$$X_i^{(n)} = \vee_{l \in R(i)} X_l^{(n)}. \tag{2.31}$$

In the sequence of node i, a bit of 0 means the correspondent probe packet is lost on the path from the root to node i. Comparing the sequence of node i with that of its siblings j, if node j receives a probe packet, and i hasn't received it, we consider the probe packet is lost on link i. Thus, it can be determined that those bits of 0's in the sequence of node i are caused by loss in link i if the correspondent bits in the sequence of node j is 1. Among those bits whose correspondent values of both node i and j are 0 (denote the number of these bits by n_{ij}^0), we know that some bits are resulted by packets lost in the path from the root to their parent node (denote this number by n_{ij-p}^0), other bits are resulted by packets lost in both the links directly connecting the siblings i and j at the same time. We use γ to denote the ratio of n_{ij-p}^0 to n_{ij}^0: $\gamma = \frac{n_{ij-p}^0}{n_{ij}^0}$.

The probability of a probe packet transmitted through link i successfully can be estimated as the ratio of the number of accepted probe packets at node i to the number of accepted probe packets at its parent node. As mentioned previously, the number of probe packets reaching the parent node of siblings can be expressed as $n_i^1 + n_j^1 - n_{ij}^1 + \gamma \cdot n_{ij}^0$, where $n_i^1 + n_j^1 - n_{ij}^1$ denote the number of 1's in "0-1" sequence of the parent node. Then $\hat{p}_i$ can be expressed as the following

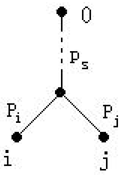


Figure 2.17: A Transmission Model

equations.

$$\begin{aligned}\hat{p}_i &= \frac{n_i^1}{n_i^1 + n_j^1 - n_{ij}^1 + \gamma \cdot n_{ij}^0} \qquad (2.32)\\ &= \sum_{k=1}^{n} X_i^{(k)} / [\sum_{k=1}^{n} X_i^{(k)} + \sum_{k=1}^{n} X_j^{(k)} - \sum_{k=1}^{n} X_i^{(k)} \wedge X_j^{(k)} \\ &+ \gamma \cdot (n - \sum_{k=1}^{n} X_i^{(k)} \vee X_j^{(k)})], \qquad (2.33)\end{aligned}$$

where n_i^1 and n_{ij}^1 denote the numbers of probe packets transmitted successfully trough link i and through both links i and j respectively; n_i^0 and n_{ij}^0 denote the numbers of lost probe packets on link i and on both links i and j respectively.

Let p_s be the probability of transmitting probe packets successfully from the root to the parent node of node pair i and j as shown in Figure 2.17.

Firstly, we give the following lemma to estimate p_s on which calculation of the loss rate on link i is based.

Lemma 5 *p_s can be estimated by the following formula*

$$\hat{p}_s = \frac{n_i^1 \cdot n_j^1}{n \cdot n_{ij}^1}. \qquad (2.34)$$

Proof

We knew that $A(i,j)$ can be estimated by $A^{(n)}(i,j)$:

$$A^{(n)}(i,j) = \frac{\sum_{m=1}^{n} X^{(m)}(i) \cdot \sum_{m=1}^{n} X^{(m)}(j)}{n \cdot \sum_{m=1}^{n} X^{(m)}(i) \cdot X^{(m)}(j)} = \frac{n_i^1 \cdot n_j^1}{n \cdot n_{ij}^1}. \qquad (2.35)$$

Because,

$$\lim_{n\to\infty} n_i^1/n = p_s \cdot p_i$$

$$\lim_{n\to\infty} n_j^1/n = p_s \cdot p_j$$

$$\lim_{n\to\infty} n_{ij}^1/n = p_s \cdot p_i \cdot p_j,$$

we have,

$$\lim_{n\to\infty} A^{(n)}(i,j) = \lim_{n\to\infty} \frac{n_i^1 \cdot n_j^1}{n \cdot n_{ij}^1} = p_s.$$

Therefore, with a finite number of probe packets we can use $\frac{n_i^1 \cdot n_j^1}{n \cdot n_{ij}^1}$ to estimate p_s. ♯

Lemma 6 *If node i and j are siblings in the real multicast network, the loss rate on link i and j denoted by $\hat{\alpha}_i$ and $\hat{\alpha}_j$ respectively can be estimated on both of their "0-1" sequences as follows:*

$$\hat{\alpha}_i = 1 - \hat{p}_i = 1 - \frac{\sum_{k=1}^n X_i^{(k)} \wedge X_j^{(k)}}{\sum_{k=1}^n X_j^{(k)}}, \tag{2.36}$$

$$\hat{\alpha}_j = 1 - \hat{p}_j = 1 - \frac{\sum_{k=1}^n X_i^{(k)} \wedge X_j^{(k)}}{\sum_{k=1}^n X_i^{(k)}}. \tag{2.37}$$

Proof

Since nodes i and j are siblings in the real multicast network, the number of lost probe packets observed at i and j at the same time composes of those lost on the link from the root to i and j's parent node and those lost on both links i and j.

$$n_{ij}^0 = n(1 - p_s) + np_s(1 - p_i)(1 - p_j).$$

We denote $\gamma \cdot n_{ij}^0$ in Equation (2.32) by Γ and equate it to the second factor of the above expression:

$$\Gamma = np_s(1 - p_i)(1 - p_j) = n_{ij}^0 - n(1 - p_s).$$

Since n_{ij}^0 by its physical meaning can also be obtained by the following equation.

$$n_{ij}^0 = n - (n_i^1 + n_j^1 - n_{ij}^1),$$

and p_s can be estimated by $(n_i^1 \cdot n_j^1)/(n \cdot n_{ij}^1)$ according to Lemma 1, we can easily derive the following formula for Γ.

$$\Gamma = \frac{n_i^1 \cdot n_j^1}{n_{ij}^1} - (n_i^1 + n_j^1 - n_{ij}^1). \tag{2.38}$$

Thus, replacing Γ in Equation (2.32) with Equation (2.38), we get the following equation to estimate p_i.

$$\hat{p}_i = \frac{n_i^1}{n_i^1 + n_j^1 - n_{ij}^1 + \Gamma} = \frac{n_{ij}^1}{n_j^1} = \frac{\sum_{k=1}^n X_i^{(k)} \wedge X_j^{(k)}}{\sum_{k=1}^n X_j^{(k)}}.$$

Likewise, p_j can be estimated as follows.

$$\hat{p}_j = \frac{n_{ij}^1}{n_i^1} = \frac{\sum_{k=1}^n X_i^{(k)} \wedge X_j^{(k)}}{\sum_{k=1}^n X_i^{(k)}}.$$

Thus, loss rates of the link i and j can be determined by Equations (2.36) and (2.37). $\sharp$

When n increases to infinity,

$$\lim_{n \to \infty} n_{ij}^1 / n = p_s \cdot p_i \cdot p_j,$$

$$\lim_{n \to \infty} n_j^1 / n = p_s \cdot p_j.$$

Clearly, $\lim_{n \to \infty} \hat{p}_i = p_i$.

Therefore, it can be concluded that the loss rate estimated by Lemma 6 is consistent with the real loss rate for each link as the number of probe packets goes to infinity.

2.4.2 Algorithm on loss inference

Since we have deduced the formula on the loss rate of a link with prior knowledge of the sequence maintained by the node, we can infer loss performance of all links in a multicast network. In the procedure of topology inference given in BHC, the "0-1" sequence for each internal node is obtained. So it becomes possible to infer the link loss rate with the help of our deduced result. We therefore extend the BHC algorithm to infer topology and all links' loss lates in the multicast network simultaneously.

The extension can be done by modifying Step 8 and 9 in the BHC algorithm as follows.

```
8. if $H_d(u,v) > \varepsilon_e$ then $S = \{u\}$, set $r$ to be the virtual parent node of $u$,$\alpha_u = 0$;//$\alpha_u$ is
   the loss rate of link $u$.//
9.     else $\{S = \{u,v\}$, set $r$ to be the virtual parent node of $u$ and $v$;
9(a).     for $i = 1$ to $n$
              Calculate $n_u^1 = \sum_{i=1}^n X_u^{(i)}$, $n_v^1 = \sum_{i=1}^n X_v^{(i)}$, $n_{uv}^1 = \sum_{i=1}^n X_u^{(i)} \cdot X_v^{(i)}$;
9(b).         Calculate loss rates of link $u$ and $v$,
              $\hat{\alpha}_u = 1 - \frac{n_{uv}^1}{n_v^1}$, $\hat{\alpha}_v = 1 - \frac{n_{uv}^1}{n_u^1}$;$\}$
```

Change Step 17 in BHC to

17. *Output*: Inferred topology and loss rate $(V', L', \hat{\alpha})$

In order to evaluate the performance of the algorithm for link loss rate inference, we compare the inferred loss rates with the real statistical loss rates as follows. The simulation has been done in the same network as depicted in Figure 2.6. We set link 7 to lose packets with a probability of 0.2, links 9 and 10 with probability of 0.5. From the simulation results shown in Figure 2.18, it can be easily seen that the inferred loss rates approach to the real loss rates as the number of probe packets increases.

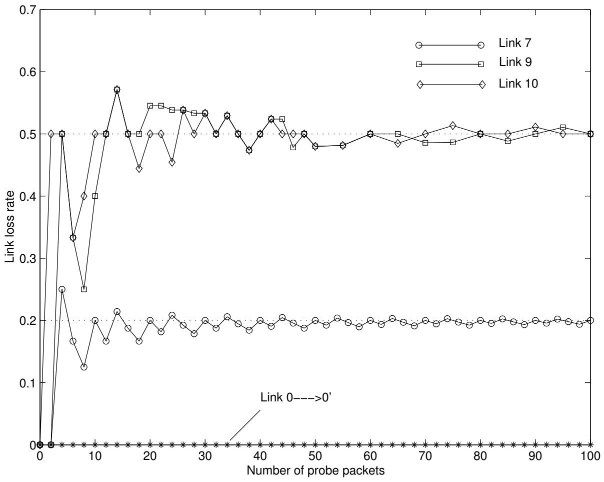


Figure 2.18: Evaluation on the inferred loss rates

2.5 Topology and Loss Performance Inference for General Trees

We extend the BHC algorithm for topology inference and the approach for link loss rate inference to general trees in this section.

2.5.1 Topology Inference for General Trees

It is more complicated to infer the topology for general trees than the binary case. We introduce a threshold η into grouping the siblings set S. The set S is grouped if the hamming distance between any pair of nodes in S is sufficiently close to being minimal.

The grouping step starts by finding a pair of nodes $\{u, v\}$ that has the minimum hamming distance in S, then adjoining further elements to it provided the following inequality is satisfied:

$$H_d(u, v')(1 - \eta) < H_d(u, v). \tag{2.39}$$

Thus we replace line 9 of the BHC algorithm by the following steps so that topology inference for general trees can be performed.

```
9a. else {$S = \{u, v\}$;
9b.     while there exists $v' \in W_e \backslash S$ such that $H_d(u, v')(1 - \eta) < H_d(u, v)$ do
9c.         $S := S \cup \{v'\}$; }
9d. Set $r$ to be the virtual parent node of all identified siblings in $S$.
```

We also use classification threshold ε_e for identification of siblings in general trees in a similar way as used in the BHC algorithm of Section 2.3.3. We can set ε_e to a given experience value for a network with binary tree topology due to the structure simplicity. However, for general trees, we set ε_e to be $\frac{n}{k} \cdot \lg e$ in order to reduce the ratio of misclassification of siblings. Here e is the level of nodes computed from the root, n is the total number of nodes in the multicast network, and k is the estimated expected number of branches of the multicast network. We set ε_e to be $\frac{n}{k} \cdot \lg e$ because we want ε_e to be linearly proportional to the number of nodes and to the logarithm of the level, and inversely proportional to the branches of each level in the multicast tree. The value may also need to be adjusted according to the real condition of the network.

As presented in [30], the violation of the condition described in inequality (2.39) has the interpretation that the ancestor $a(U)$ is separated from $a(\{u, v\})$ by a link with loss rate at least η. The convergence of the inferred topology to the true topology is mainly influenced by η. If only η is less than the internal link loss rates, the inferred topology will be convergent to the true topology. However, the internal link loss rates are unknown in advance. A small value of η is more likely to satisfy the above condition but at the cost of slow convergence. A large value of η, on the other hand, is more likely to result in systematically removing links with small loss rates. Thus it is convergent to a wrong topology. In order to choose an appropriate η to obtain more accurate and complete topology practically, we will propose the scheme of incorporating the link loss rate into topology inference in Section 2.5.3 based on the loss performance inference for general trees.

2.5.2 Loss Performance Inference for General Trees

Based on the priori knowledge of the inferred topology and "0-1" sequences for each node discussed above, we extend the approach in Section 2.4.1 to infer loss performance for general trees. Similar to Lemma 6, the link loss rate of general trees can be estimated by the following Lemma.

Lemma 7 *The loss rate of link s_1 can be estimated by $\hat{\alpha}_{s_1}$.*

$$\hat{\alpha}_{s_1} = 1 - \hat{p}_{s_1} = 1 - \frac{n^1_{s_1 s_2 s_3 \ldots s_m}}{n^1_{s_2 s_3 \ldots s_m}}, \tag{2.40}$$

where $s_1, s_2, \ldots, s_m$ are siblings.

Proof

As n increases to infinity,

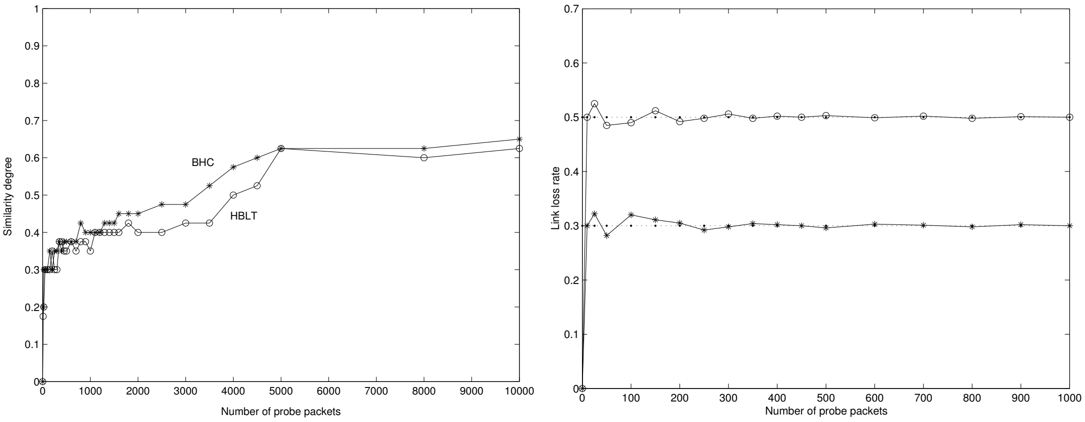
$$\lim_{n\to\infty} n^1_{s_1 s_2 s_3}/n = p_s \cdot p_{s_1} \cdot p_{s_2} \cdot p_{s_3},$$

$$\lim_{n\to\infty} n^1_{s_2 s_3}/n = p_s \cdot p_{s_2} \cdot p_{s_3}.$$

Thus, $\lim_{n\to\infty} \frac{n^1_{s_1 s_2 s_3}}{n^1_{s_2 s_3}} = \lim_{n\to\infty} \hat{p}_i = p_i$. ♯

As the number of probe packets goes to infinity, the loss rate estimated by $\hat{p}_i$ is consistent with the real loss rate for each link in general trees.

We have done the simulation for topology inference and loss rates inference in general trees with sizes ranging from 20 nodes to 200 nodes. Figures 2.19(a) and 2.19(b) gives the average results on these general trees.



(a) Topology inference by HBLT and BHC when $\alpha = 0.5, \beta = 0.5$

(b) Comparison between inferred loss rates and real link loss rates of are 0.3 and 0.5 respectively.

Figure 2.19: Simulation on a general tree network

2.5.3 Loss Rate-Combined Topology Inference

As discussed in Section 2.5.1, parameter η is used for grouping siblings in general trees, which requires η to be less than all internal link loss rates such that the inferred topology can be correctly convergent to the true topology. However, the internal link loss rates are unknown in advance, which may result in wrongly inferred topologies containing only high loss rate links as discussed in [30] about failure inference.

Thus, with the help of the approach to loss rate inference proposed in section 2.5.2, a scheme is built to adjust η timely based on the inferred loss rates.

In the procedure of grouping siblings, the loss rates of links are computed. Errors in grouping may exist due to an inappropriate choice of η. So when the minimum loss rate among all links is less than η, we change the value of η to be the minimum loss rate, and group the siblings again according to the modified η. The procedure is described as follows.

```
1. Initiate $\eta$, $\hat{\alpha}_{min}$.
2. While $\hat{\alpha}_{min} < \eta$ do
3.     $\eta = \hat{\alpha}_{min}$
4.     Group siblings according to inequality (2.39).
```

5. Compute the loss rates based on inferred topology,
denote the minimized loss rate as $\hat{\alpha}_{min}$.

In the above procedure, we initially set η to be smaller than $\hat{\alpha}_{min}$ that has a value between 0.5 and 1. So some pseudo siblings are grouped firstly, which will result in large loss rates, whose minimum value is less than η. Then all the nodes will be grouped again with a smaller η, some pseudo siblings are removed in this case, which consequently diminishes the minimum loss rate inferred in the newly grouped tree. The nodes are grouped again due to $\hat{\alpha}_{min} < \eta$. The procedure is repeated until η is adjusted to an appropriate value that satisfies the condition that η is smaller than the loss rates of all links in the network. Thus, the inferred topology can quickly converge to the true topology with the appropriate η. The following Lemma gives the relationship between η and the link loss rate.

Lemma 8 *A small η can remove some pseudo siblings, and result in decreased loss rates of the links.*

Proof

Assume that s_1 is identified to have m siblings including itself and some possible pseudo siblings. Then, let $\check{p}_{s_1}(m)$ denote the probability of successfully transmission on link s_1 with the hypothetical m siblings.

$$\check{p}_{s_1}(m) = \frac{n^1_{s_1 s_2 s_3 \dots s_m}}{n^1_{s_2 s_3 \dots s_m}}.$$

If a smaller η causes a node to be removed from its siblings set according to inequality (2.39),

$$\check{p}_{s_1}(m-1) = \frac{n^1_{s_1 s_2 s_3 \dots s_{m-1}} + x}{n^1_{s_2 s_3 \dots s_{m-1}} + y},$$

where x and y are integers, and $x < y$ due to the removal of the node whose "0-1" sequence is much different from those of other nodes. The minimum hamming distance among all pairs of sequences maintained by the removed node and other nodes is far from those pairs among other siblings, which leads x to be less than y, and both are greater than 0.

Thus,

$$\check{p}_{s_1}(m-1) < \check{p}_{s_1}(m).$$

So a smaller η leads the adjusting procedure to remove some pseudo siblings and thus results in decreased loss rates of the links. $\sharp$

By adjusting η the loss rate-based topology inference can thus be made more accurate than those inferred in [30, 67].

2.6 Hamming Distance Matrix for Loss/Delay Performance Analysis

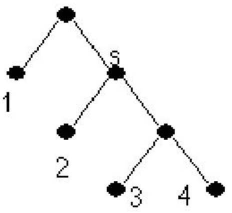
This section generalizes the application of hamming distance-based network internal loss/delay performance inference, which can use either end-to-end loss measurements for network internal loss performance diagnosis or end-to-end delay measurements for delay performance diagnosis. From above sections, it is seen that every receiver maintains a bit sequence from end-to-end loss measurements. When end-to-end delay measurements is counted, similarly, if a probe packet has some delay when received by a receiver, the correspondent digit of bit sequence maintained by this receiver is 0, otherwise it's 1 as modelled in Section 2.1.2. Each internal node can also get a bit sequence by use of the same procedure as in BLT, HBLT, or BHC based on end-to-end loss measurements. From all loss/delay measurements, we can use a hamming distance matrix-based scheme for analyzing the network internal loss/delay performance.

A hamming distance matrix is first defined in this section. The bit sequence maintained by each receiver is denoted by $\{X_r^{(m)}\}$, $1 \leq m \leq n$, $r \in R$. We assume the number of receivers in a multicast network to be l. For simplicity, let d_{ij} denote the hamming distance between receiver i and j, i.e., $d_{ij} = H_d(i, j)$. Denote D to be the hamming distance matrix. Then D is defined as follows:

$$\mathbf{D} = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1l} \\ d_{21} & d_{22} & \dots & d_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ d_{l1} & d_{l2} & \dots & d_{ll} \end{pmatrix}.$$

Obviously, the matrix is a symmetric matrix, that is, $d_{ij} = d_{ji}$ for $i \neq j$. And it is also easy to know $d_{ij} = 0$ if $i = j$. Apart from these properties, the matrix implies more information. For instance, if receiver i and j are siblings, d_{ij} is supposed to be smaller than d_{ik}, $k \neq j$ and $k \neq i$. More generally, the nearer two nodes are located, the smaller their hamming distance is supposed to be because their sequences are more similar due to more shared common link condition which we have discussed in Section 2.3. Another property of this matrix is that hamming distance among those receiver-pairs with small distance values (except receiver-pairs whose small distance is because both receive almost all packets without loss/delay) is transitive because they share common links that are experiencing heavy link loss/delay, and hence the matrix can be transformed into a blocked matrix. Therefore, such a hamming distance matrix shows a lot of information on internal topology in the condition that those internal links have different congestion and loss situations.

Now let's see what this matrix can do for internal loss/delay performance analysis and identification. Since we have inferred the multicast network topology in Section 2.3, how to discover the internal loss/delay performance based on the topology is now considered. For clearly clarifying the applications of the hamming distance matrix in loss/delay performance analysis, we firstly consider a network in Figure 2.20 where only one receiver observes severe loss. We assume here the loss measurements are counted. Internal link delay performance can be obtained if the end-to-end delay measurements observed on receivers are counted. Suppose link s in Figure 2.20 is in very poor condition and all other links work in good condition with few losses.



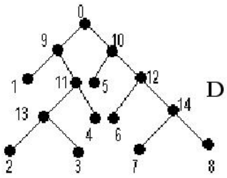
$$\begin{array}{cc} & \begin{array}{cccc} \text{Receiver:} & 1 & 2 & 3 & 4 \end{array} \\ \mathbf{D} = & \begin{pmatrix} 0 & 88 & 85 & 92 \\ 88 & 0 & 0 & 3 \\ 85 & 0 & 0 & 2 \\ 92 & 3 & 2 & 0 \end{pmatrix} \end{array}$$

Figure 2.20: A simple hamming distance matrix.

According to the hamming distance matrix in Figure 2.20, we can easily find the hamming distances between 1 and any of the rest receivers are very great, while the hamming distances among receiver 2, 3 and 4 are quite similar and small. Then we can infer that the nearest common link of receiver 2, 3 and 4 work in poor condition.

As for a network where internal links condition may be very complex, we need to transform the matrix into several blocks according to different values of components in the matrix. Usually we classify the components by an experienced difference which depends on how many probe packets are sent in total. With the experienced difference, we can do elemental transformation on the hamming distance matrix and obtain a matrix in several regular blocks. The receivers are accordingly classified into several groups. Thus we can infer all the possible bad links in the topology according to those classified receivers by blocks easily. For example the network topology, the hamming distance matrix and the transformed matrix are given in Figure 2.21, we aim to analyze the internal link loss/delay performance and locate the ill-performing link easily.

There is a group including receiver 1, 4 and 5 which receive almost all the probe packets. This group is called base group which can be easily identified by the 0-1 sequence maintained by any receiver in this group. The links in the path from the source to the receiver in this group all works in good condition. We find all other groups have the common property, that is, the hamming distance between any receiver in these group and any receiver in base group is very great while the hamming distance between the receivers in each group is very small. These groups are called loss group (or delay group in the case of internal delay inference). We can then decide the link connecting to the nearest common ancestor node of each loss/delay group is one of the links who are *causing severe loss* (experiencing severe delay). In the topology of



$$\mathbf{D} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \hline \left(\begin{array}{cccccccc} 0 & 72 & 73 & 0 & 0 & 91 & 102 & 102 \\ 72 & 0 & 3 & 73 & 75 & 43 & 52 & 51 \\ 73 & 3 & 0 & 61 & 62 & 53 & 66 & 63 \\ 0 & 73 & 61 & 0 & 0 & 92 & 105 & 101 \\ 0 & 75 & 62 & 0 & 0 & 96 & 101 & 102 \\ 91 & 43 & 53 & 92 & 96 & 0 & 27 & 30 \\ 102 & 52 & 66 & 105 & 101 & 27 & 0 & 0 \\ 102 & 51 & 63 & 101 & 102 & 30 & 0 & 0 \end{array}\right) \end{array}$$

Hamming distance matrix in blocks:

$$\mathbf{D} = \begin{array}{c} \begin{array}{ccc|cc|c|cc} 1 & 4 & 5 & 2 & 3 & 6 & 7 & 8 \end{array} \\ \hline \left(\begin{array}{ccc|cc|c|cc} 0 & 0 & 0 & 72 & 73 & 91 & 102 & 102 \\ 0 & 0 & 0 & 73 & 61 & 92 & 105 & 101 \\ 0 & 0 & 0 & 75 & 62 & 96 & 101 & 102 \\ \hline 72 & 73 & 75 & 0 & 3 & 43 & 52 & 51 \\ 73 & 61 & 62 & 3 & 0 & 53 & 66 & 63 \\ \hline 91 & 92 & 96 & 43 & 53 & 0 & 27 & 30 \\ \hline 102 & 105 & 101 & 52 & 66 & 27 & 0 & 0 \\ 102 & 101 & 102 & 51 & 63 & 30 & 0 & 0 \end{array}\right) \end{array}$$

Figure 2.21: Hamming distance matrix and matrix in blocks

Figure 2.21, link 13, 6 and 14 are links we aim to identify. All other links work well with few loss or delay in this network.

Therefore, if we can transform a hamming distance matrix into blocks according to the values of components, the receivers are classified into several groups. Then we can determine a base group which include those receivers who *receive almost all probe packets* (or *receive most probe packets without delay*), i.e., there is not any links causing big loss (or severe delay) in the path from the source to these receivers. And apart from this group, there are many loss/delay groups. The link connecting to the nearest common ancestor node of each loss/delay group are identified as those links where severe loss/delay is caused. By this means, a hamming distance matrix D of a network can help to analyze and identify the network internal loss/delay performance. Moreover, a phenomenon which values our attention is that there are usually only a few links who are in poor condition in a large scale network in practice. Therefore, the components in hamming distance matrix usually appear to be different obviously and can be classified very easily.

The above method can be accomplished by the following algorithm:

```
INPUT: Hamming distance Matrix $D$ and the set of $l$ receivers $R$; //D is a symmetric
matrix whose row/column indices are kept in $R$ denoting different receivers.//

Step 1: $i = 1$, $b = 1$, $R_b = \emptyset$, $R = \{1, 2, 3, \ldots, l\}$;

Step 2: While $i \leq l$ do

Step 2.1:    Compare all elements in row $i$ and record in $R_b$ the original column indices
```

of all the elements whose values are not greater than an experienced threshold $n \cdot \alpha$. Assume $R_b = \{r_1, r_2, \ldots, r_{p_b}\}$; //$n$ is the number of probe packets, α is an experienced ratio.//

Step 2.2: Do row swaps and column swaps to move rows $r_1, r_2, \ldots, r_{p_b}$ to top and columns $r_1, r_2, \ldots, r_{p_b}$ to left in consecutive positions from $\sum_{j=1}^{b-1} p_j + 1$ till $\sum_{j=1}^{b} p_j$. Record in R the original row/column indices of the permutated rows/columns according to the original hamming distance matrix;

Step 3: $i = i + p_i$; $b = b + 1$;

OUTPUT: Hamming distance matrix D in blocks with grouped receivers and index array R, where $R(i)$ is the ith row/column's (in the output D) original row/column index in the input D.

The algorithm's time complexity is decided by Step 2. Since Step 2.1 takes $O(l)$ and Step 2.2 requires $O(p_i \cdot l)$ in the worst case, the dominating factor in Step 2 is Step 2.2, that is, the number of element-exchanges for row swaps and column swaps. Assume that the output D contains b groups of receivers, where group i is of size p_i, represented in the form of blocks along the main diagonal in D. Clearly, $\sum_{i=1}^{b} p_i = l$. The worst-case is that all these groups are produced by row swaps and column swaps. In this case, the total number of element-exchanges involved in row swaps and column swaps is:

$$2 \cdot \sum_{i=1}^{b} l \cdot p_i = 2l \cdot \sum_{i=1}^{b} p_i = 2l^2.$$

Hence the worst-case time complexity of the above algorithm is $O(l^2)$.

2.7 Concluding Remarks

In this chapter, we have introduced network tomography for topology inference. We have proposed new approaches of using hop count and hamming distance classification, which bring different benefits to topology inference. We have developed new methods for network internal loss/delay performance inference based on discovered topology information.

Based on hop count, we have proposed an improved algorithm for multicast network topology discovery, Binary Loss Tree Classification with Hop Count (HBLT). Through analysis on time complexity, misclassification probability, and inference accuracy, and comparisons with the previous binary loss tree classification algorithm (BLT), the advantage of the measurement on hop count is justified. Experimental results also support the usage of hop count in topology inference and have been shown that HBLT outperforms BLT greatly with a significantly lower misclassification probability and higher inference accuracy and efficiency.

We have further proposed the application of hamming distance classification approach, namely Binary Hamming distance Classification (BHC), which improves the traditional A-approach used in HBLT and previous algorithms. With a finite number of probe packets, the topology inferred by the BHC algorithm has been shown to be more accurate than those inferred by previous algorithms based on A-approach. It has also been shown that BHC significantly outperforms previous algorithms in efficiency according to the implementation results of BHC, BLT and HBLT algorithms in our simulated networks. Since all algorithms discussed in this chapter are based on end-to-end loss measurement, they may not work very efficiently when inferring topology for a multicast network with few losses. In this case, same as other work in the literature [30,34], only when the number of probe packets is very large, the inferred topology may approach to the accurate topology. To avoid use of an excessive number of probe packets for topology inference, we may simply replace loss measurement by delay measurements. In multicast networks with few losses, delay-based topology inference would provide a more efficient inference procedure. We can also choose the measurement according to the network status adaptively as [33]. Thus, the use of hop count measurement and hamming distance classification approach can be easily applied to inference algorithms based on other types of measurements. This also indicates why we can infer internal loss/delay performance by hamming distance matrix approach from end-to-end loss/delay measurements as discussed in Section 2.6.

Extension of the hamming distance approach to general trees has also been discussed in this chapter. It has been shown that our proposed method of incorporating hop count and hamming distance-based topology inference is also quite effective for topology inference of general trees.

Apart from topology inference, we have also proposed a novel approach to inferring the network internal loss performance, making it possible for the first time to infer multicast topology and link loss rate simultaneously. It is based on the data collected in the procedure of topology inference. Experimental results have shown that the loss performance inferred by our approach is consistent with the real loss performance in the multicast network.

Our loss performance inference approach has also been extended to the general trees. Since topology inference for general trees is very complicated and cannot achieve as good result as for binary trees. By incorporating approaches for loss performance inference into topology inference, we have given a new idea on a technique that can improve the inferred topology to one that is correctly convergent to the true topology.

Chapter 3

Network Topology Discovery by Mobile Agents

Since mobile agents have been successfully deployed for collecting and processing network management information in the Internet, telecommunication networks and other types of networks, it is possible to extract topological information by mobile agents. In this chapter, we propose mobile agent-based network topology discovery methods. We first give an introduction on mobile agent and its applications, then develop algorithms of deploying mobile agents for both Internet and Multicast network topology discovery. We further establish statistical models to show the behavior of mobile agents and give a complete analysis on the performance of our proposed algorithms. Finally we give the experimental results. It is shown that due to the inherent advantages of mobile agents, the proposed algorithms provide effective techniques for employing mobile agents in topology discovery which has great promises in the future.

3.1 Introduction

A mobile agent is a software that can move within the network and act on behalf of a user or another entity. Mobile agents can function independently or cooperatively to solve problems. Deployment of mobile agent is one of the most promising technologies for managing complex and large scale networks. It has potential applications in various functional areas as discussed in [7, 11, 12, 43, 50, 52, 53, 56, 73, 75], for instance, network management, information retrieval, network services delivery, and topology discovery [48].

The introduction of mobile agent brings a lot of benefits to various applications. It reduces traffic congestion because agents are smaller in size. It gains enhanced security over protected data especially in a broadcast mode. It avoids unnecessary data transfer. The transfer of user intention enables selection of required data and intelligently computed abstraction. It is a modular approach to distributed applications. In addition, it enables interoperability through a new agent layer.

Due to all these advantages brought by mobile agents inherently, topology discovery based on mobile agents can be performed more efficiently with lighter burden to the network than before. However, mobile agent-based schemes must have support from the network hardware. Usually the special protocol is developed for mobile agent-based system. Therefore, our study on mobile agent-based topology discovery is developed in these infrastructures.

This chapter mainly focuses on the problem of automatic topology discovery using mobile agents. We build a network model for the mobile agents running for topology discovery where automatic discovery is achieved in two different ways, report-at-newly-found-nodes (RN) and report-at-leaves(RL), along the same line as proposed in [48]. In this model, several important statistics including dwell time distribution, life span distribution, the report time distribution of a mobile agent and the interreport time distribution at the management station are analyzed. These analytical results give significant insight to reveal the behaviors of mobile agents performing the topology discovery task. We finally give a clear view on the performance of different schemes through comparison among them.

3.2 Network Model

A network comprises a set of nodes and communication links connecting them. Each node is assigned an unique ID (e.g., IP address). Each node has the knowledge of the IDs of itself and its neighboring nodes. Each link is bidirectional and symmetric. Transmission speeds are identical in both directions of the same link, but may be different over different links. A management station is located at one node in the network. Receivers are a set of nodes whose IP addresses are known by the management station. But the management cannot know how they connect to each other. Thus the management station initiates the task of network topology discovery by generating a discovery agent. The goal is to discover all the receivers and how they connect to the management station. This model is similar to the model built in [48, 68].

The generated discovery agent can mark the nodes and the links it passed. It can also spawn new discovery agents if searching along different paths is needed and another type of mobile agents, report agent, when it decides to report the topology information to the management station. The report agent can extract the information of the nodes and their neighborhood, provide storage to store the required topology information, and report the information to the management station.

At the beginning of the topology discovery task, the discovery agent visits the node where the network management station resides. Based on the neighborhood information maintained at the node, the discovery agent will spawn more discovery agents and dispatch them to each of the neighboring nodes. The discovery agents interact with the nodes they visit and may collect information from these nodes. When necessary, a discovery agent will spawn a report agent to send topology information collected in the discovery agent back to the management station. After receiving the topology information from the report agent, the management station stores these data into its topology database, and sends an acknowledgement (ACK) to the discovery agent. After it gets the ACK from the management station and makes sure that its information has been sent back successfully, the mobile agent will either terminate itself or continue on discovering other topology information, depending on if all neighborhood nodes have been visited or not. If there is no ACK from the management station after a certain period, the mobile agent will send back the information once again and decide whether to go on with the searching or not. Decision on whether the report agent should be spawned depends on different report ways proposed in the algorithms.

Two ways of report fashion are proposed for mobile agents, persistent report and intermittent report. Persistent report means that a mobile agent reports from every node it visits. In the model of RN for multicast topology discovery, mobile agents are persistent report agents. When a mobile agent reports from some of the nodes it has visited, it is called intermittent report. RL and RN algorithms for Internet topology discovery and RL algorithm for multicast topology discovery apply this type of report.

3.3 Mobile Agent-based Topology Discovery

Four mobile agent based topology discovery algorithms are proposed in this section. Two of them are for Internet topology discovery and the other two are for multicast topology discovery. All of them use controlled flooding to discover the network topology.

As assumed in the model, the network management station generates a discovery agent at the node where it resides at the beginning. If the discovery agent arrives at a node which hasn't been visited by any other discovery agent, the discovery agent marks the node "visited" and the link from which it enters the node as "upstream" link. It then checks the node's IP address to see if it is a receiver. If the node is a receiver, the discovery agent terminates itself after reporting the topology information successfully. Otherwise, it regards all other links at this node as "downstream" links and spawns more discovery agents so that each of the downstream links gets one discovery agent. Each discovery agent moves to a neighboring node along a downstream link. When a discovery agent arrives at a node which has been marked "visited", the discovery agent will stop searching because there is no downstream links at a node already visited by another mobile agent. The link from which the discovery agent enters a visited node is marked "N" indicating that this link is neither an upstream link nor a downstream link. Clearly, an "N" link causes a cycle in the network. Thus the flooding is controlled by checking statuses of nodes. In this way, the discovery agents spawn themselves and flood the network as fast as possible. The management station can construct the topology of the network according to the information sent back by report agents.

We call the set of nodes containing all "visited" nodes and un-visited receivers *leaf nodes.* Clearly, a discovery agent will terminate itself when arriving at a leaf node.

3.3.1 Mobile agent-based algorithms for Internet topology discovery

We propose two algorithms for Internet topology discovery that differ in the way of carrying back topology information to the management station.

1. The report-at-newly-found-nodes (RN) algorithm

In this algorithm, when a discovery agent arrives at a node that has never been visited by any other discovery agent, it immediately spawns a report agent to carry the information of the newly discovered node and its neighboring nodes back to the management station. The report agent moves along the "upstream" links back to the management station. At the same time, it generates new discovery agents, each being dispatched to a link connecting a neighboring node.

When the management station receives the newly discovered topology information, it stores the information in its topology database. It doesn't send acknowledgement to discovery agents in the RN algorithm because the report agents report frequently. Even if a report agent carrying the topology information of a node is lost, the management station can still extract the information about the node if any of its neighboring nodes reports correctly.

After the mobile agents finish the topology discovery task, the management station constructs the topology according to its topology database. It may get incomplete topology information when the network runs in heavy load. In this case, the management station can initiate the topology discovery task again with the lost nodes as destination only. In order to determine whether all required topology information of the network have been received, the management station checks the status of each node in its database to see if there is any node that has not been visited by a discovery agent.

2. The report-at-leaf-nodes (RL) algorithm

In this algorithm, a discovery agent does not always spawn a report agent every time it finds new topology information. When a discovery agent arrives at a node, it checks to see if this node has been marked "visited". If yes, there can't be downstream links for the mobile agent to continue on searching because the previously arrived mobile agent must have dispatched mobile agents to all links connecting to this node. If the node hasn't been visited by another mobile agent, the mobile agent checks if there are downstream links. If so, the discovery agent spawns several discovery agents, each being dispatched to one downstream link. Otherwise, it spawns a report agent. In another word, a discovery agent spawns a report agent when it determines that a node is a leaf node of the network.

The report agent moves along the "upstream" links back to the network management station. It collects the topology information along the path. When the network management station received the topology information, an "ACK" is sent to the discovery agent. After receiving "ACK", the discovery agent is terminated. If "ACK" hasn't been received, the discovery agent will spawn a report agent to report again until it gets "ACK" finally. After a certain period of time, the management station checks its topology database to see if all required information is obtained. If yes, the topology can be constructed according to the information. In case if too much information is lost, the topology can also be constructed as an incomplete topology. Unless a lot of topology information is lost, the management station will discover the topology again.

In the report-at-leaves algorithm, the report agents are spawned at leaf nodes where no downstream links exist. As we have said, if a node has been marked "visited", there can't be downstream links. If a node is a receiver, there can't be downstream links. Thus, a mobile agent reports at leaf nodes, and terminates itself after it receives "ACK" on report success notification.

3.3.2 Mobile agent based algorithm for multicast network topology discovery

The network management station administrates one or more multicast groups. According to multicast protocol [74], each router supporting multicast is assigned an extra multicast address specifically. All the group members including routers are assigned the identical multicast address. The task of mobile agents is to discover all the group members and how they connect with each other in the multicast network.

When the management station requests the information on the topology of the multicast network, one mobile agent is generated. The first node it visits is the node where the management station is located. The discovery agent spawns several new discovery agents to all connected routers supporting multicast except the router where the discovery agent comes from. The task of multicast network topology discovery will finish when all the group members are discovered. The detailed algorithms in different report fashions are given in succeeding paragraphs.

1. The report-at-newly-found-nodes (RN) algorithm

Similar to RN algorithm for Internet topology discovery, the discovery agent spawns a report agent each time it visits a new group membership. The report agent reports the latest topology information as soon as possible. Because there isn't any cycle in the multicast network, the discovery agent spawns a report agent at each step, which leads to persistent report.

The management station doesn't send an "ACK" to discovery agents when it receives the topology information reported by report agents. After spawning the report agent, the discovery agent will continue to discover more group memberships until it finds there is no downstream node. Then it reports and terminates itself at leaf nodes(also group receivers in multicast network).

There are two kinds of multicast distribution trees: source trees and shared trees, also called RP trees or CBT (core-based trees). For source trees, the RN algorithm marks the link where the discovery agent enters as “upstream” link. For core-based trees the mark “USR” is used to denote the link from the source to the shared root where unicast is used. When the discovery agents are multicasted from the shared root to the receivers, the links they pass are marked as “upstream”. Thus, each report agent spawned before its discovery agent arrives at the shared root returns the collected information to the management station along “USR” path. The report agents spawned after its discovery agent arrives at the shared root return the information along “upstream” firstly to the shared root. Then the report agent returns from the shared root to the management station by “USR” path. When a mobile agent finds there isn’t any “downstream” link connecting another group member, it terminates itself after reporting the topology information. After all report agents have sent back the collected information, the management station can reconstruct the multicast tree based on the reported information.

2. The report-at-leaf-nodes (RL) algorithm

In RL algorithm for multicast topology discovery, the report agent is only spawned at leaf nodes. At the intervening routers the discovery agents mark the path “upstream” or “USR” without generating the report agent. Only when the discovery agent finds it arrives at leaf nodes (where there is no downstream link), it generates a report agent. The report agent is sent back to the management station through the marked link by which it collects the topology information along the path. The management station sends an “ACK” to the discovery agent after it receives the report agent. When the discovery agent gets the “ACK” from the management station it terminates itself. If no “ACK” is sent to discovery agent after some period, it will spawn a report agent to report the topology information again until it gets the “ACK” from the management station, which shows that the topology information has been received successfully.

3.4 Analysis on Mobile Agents for Topology Discovery

In this section the characteristics of mobile agents for topology discovery are analyzed. These include dwell time distribution at a host, life span distribution of a mobile agent, report time distribution of a mobile agent and interreport time distribution at the management station. The technique of Laplace transform is applied as presented in [42].

3.4.1 Dwell time distribution

Dwell time is defined as a random variable representing the time of a mobile agent staying in a host. The mobile agent concerned here is the discovery agent only. When the management station needs to discover the topology of a network, many mobile agents are generated in the procedure to collect information. These mobile agents travel from one host to another host, mark the nodes and links, collect relevant information, and report the collected information to the management station. Analysis on how long a mobile agent needs to stay in one node is beneficial to determine how long the topology discovery task will take.

Assume that a mobile agent is served immediately when it arrives at a host. Then the dwell time of the mobile agent at a host denoted by T_D doesn't include the queuing delay. T_D is the summation of mark time T_M and report time T_O. If the mobile agent doesn't spawn the report agent at a host, $T_O = 0$. For instance, the mobile agent will not report if it hasn't arrived at the leaf nodes in the RL algorithms described previously.

$$T_D = T_M + T_O. \tag{3.1}$$

The cycle time T_C which describes the time period of a mobile agent at a host is the dwell time at the host T_D plus its travel time to the next host T_V as shown in Figure 3.1.

$$T_C = T_D + T_V. \tag{3.2}$$

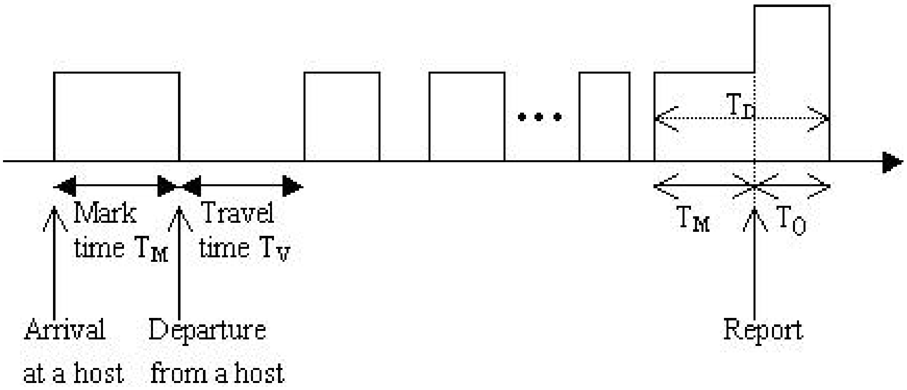


Figure 3.1: Mobile agent state description

Dwell time depends on the network host status such as processor speed of a host. Cycle times of different mobile agents are assumed to be independent of each other. Reporting time contains such latency as spawning time, report propagation delay, acknowledgment delay from the source, and twice propagation time or more.

Suppose the mark time probability density function (pdf) to be $m(t)$, the reporting time pdf to be $o(t)$, the travel time pdf to be $v(t)$, dwell time pdf to be $d(t)$ and the cycle time pdf to be $c(t)$. Because the sum of two independent random variables results in a convolution $(*)$ of the their probability density functions, we have,

$$d(t) = m(t) * o(t), \tag{3.3}$$

$$c(t) = d(t) * v(t) = m(t) * o(t) * v(t). \tag{3.4}$$

Proof If we assume

$$T_C = T_D + T_V$$

We can obtain,

$$E[T_C] = E[T_D] + E[T_V]$$

It can also be described by the following equation,

$$\int_0^\infty te(t)dt \cdot \int_0^\infty r(t)dt + \int_0^\infty tr(t)dt \cdot \int_0^\infty e(t)dt = \int_0^\infty td(t)dt$$

Where $\int_0^\infty r(t)dt = 1$ because $r(t)$ is the probability density function. Similar case holds for $e(t)$.

The above equation is the result of the following when we assume $s = 0$.

$$\int_0^\infty te(t)e^{-st}dt \cdot \int_0^\infty r(t)e^{-st}dt + \int_0^\infty tr(t)e^{-st}dt \cdot \int_0^\infty e(t)e^{-st}dt = \int_0^\infty td(t)e^{-st}dt$$

$$\frac{\partial \int_0^\infty e(t)e^{-st}dt \cdot \int_0^\infty r(t)e^{-st}dt}{\partial s} = \frac{\partial \int_0^\infty d(t)e^{-st}dt}{\partial s}$$

Then,

$$\int_0^\infty e(t)e^{-st}dt \cdot \int_0^\infty r(t)e^{-st}dt = \int_0^\infty d(t)e^{-st}dt$$

It's,

$$E^*(s) \cdot R^*(s) = D^*(s)$$

Thus,

$$d(t) = e(t) * r(t)$$

□

Take the Laplace transform of the dwell time pdf, then the above relationships can be expressed in the following equations.

$$D(s) = M(s) \cdot O(s), \tag{3.5}$$

$$C(s) = D(s) \cdot V(s) = M(s) \cdot O(s) \cdot V(s). \tag{3.6}$$

Here, $M(s)$, $O(s)$ and $V(s)$ are accordant to the definition of Laplace transform: $X(s) \equiv \int_0^\infty x(t)e^{-st}dt$, for $X = M, O, V$ and $x = m, o, v$.

Then the statistical property of the dwell time at a host is given by the following Lemma.

Lemma 9 *The mean $E[T_D]$ and variance $Var[T_D]$ of T_D can be calculated by the following equations.*

$$E[T_D] = -\frac{dD(s)}{ds}|_{s=0}, \tag{3.7}$$

$$Var[T_D] = \frac{\partial^2 D(s)}{\partial s^2}|_{s=0} - [\frac{dD(s)}{ds}|_{s=0}]^2. \tag{3.8}$$

Proof For Equation (3.7),

$$D(s) = \int_0^\infty d(t)e^{-st}dt,$$

$$\begin{aligned} -\frac{dD(s)}{ds}|_{s=0} &= \int_0^\infty d(t)te^{-st}dt|_{s=0} \\ &= \int_0^\infty d(t)tdt \\ &= E[T_D]. \end{aligned}$$

For Equation (3.8),

$$Var[T_D] = E[T_D^2] - E^2[T_D],$$

$$\begin{aligned} \frac{\partial^2 D(s)}{\partial s^2}|_{s=0} &= \frac{\partial^2 \int_0^\infty d(t)e^{-st}dt}{\partial s^2}|_{s=0} \\ &= \int_0^\infty d(t)t^2dt \\ &= E[T_D^2]. \end{aligned}$$

Thus,

$$\frac{\partial^2 D(s)}{\partial s^2}|_{s=0} - [\frac{dD(s)}{ds}|_{s=0}]^2 = Var[T_D].$$

□

3.4.2 Life span of a mobile agent

Life span of a mobile agent is the time between its generation and its termination. A mobile agent can be generated at the management station or any internal router. Termination of a mobile agent happens in the following cases. When a discovery agent receives an "ACK" from the management station showing that the report agent has returned topology information successfully, it will be terminated according to the proposed algorithms. This kind of termination happens at leaf nodes. The source can discard returned report agents. But termination of a report agent isn't taken into account when analyzing the life span of a mobile agent. Because once the condition of links is given, the life span of a report agent is the transmission time. Therefore, we only consider the life span of a discovery agent.

1. Analysis on the mean of life span

Before a mobile agent is terminated, the total life span of the mobile agent includes several cycle times in the intervening nodes and one dwell time, as shown in Figure 3.2. Denote the life span by T_L. It can be expressed as Equation (3.9).

$$T_L = t_1 + t_2 + t_3 + \ldots + t_N = \sum_{k=1}^{N} t_k, N \geq 1. \tag{3.9}$$

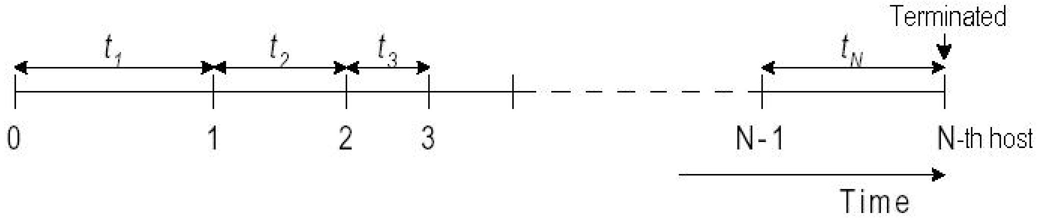


Figure 3.2: Total life span of a mobile agent

Here, N is the total number of hosts visited by a mobile agent, t_k, $1 \leq k \leq N-1$ is cycle time in host k which includes the dwell time in host k plus the travel time, and t_N includes only the dwell time in (the last) host N.

Because t_k, $1 \leq k \leq N$, is independent of the total number of hosts visited by the mobile agent, the average life span of a mobile agent can be inferred as follows.

$$\begin{aligned}
E[T_L] &= E[\sum_{k=1}^{N} t_k] \\
&= E[E[\sum_{k=1}^{N} t_k|N]] \\
&= \sum_{n=1}^{\infty} Pr(N=n)E[\sum_{k=1}^{N} t_k|N=n] \\
&= \sum_{n=1}^{\infty} Pr(N=n)E[\sum_{k=1}^{n} t_k] \\
&= \sum_{n=1}^{\infty} nPr(N=n)E[t_k] \\
&= E[N]E[t_k]. \qquad (3.10)
\end{aligned}$$

2. Analysis on a given network

For a given network that has r' receiver nodes, l' links, and n' nodes in total including the management node, the average number of mobile agents generated for topology discovery can be estimated based on the above result. If there isn't any cycle in the network, the number of links l' is equal to $n'-1$. Otherwise, l' cannot be decided simply by the total number of nodes.

From the network in Figure 3.3 we know that the first mobile agent spawned in the management station generates three mobile agents at the start. They are dispatched into each direction of the network. When the mobile agents pass through branches, new mobile agents are generated so that each downstream link has one agent to continue searching. All the links and nodes will be marked correctly after they have been visited as Figure 3.3 shows. In this network, mobile agents terminate themselves in two cases. One is when they arrive at the receivers, the other is when they visit a link that has already been marked "N", indicating that another agent has already visited the link.

The average life span of all mobile agents spawned for the task of discovering topology of a network can be calculated according to Equation (3.10). Because automatic topology discovery by mobile agents uses flooding algorithm, each link is visited only by one mobile agent.

Therefore, the average number of hops a mobile agent made to finish the topology discovery task is the total number of hops H made by all discovery agents divided by the total number of generated discovery agents M in the network. The number of links in the network is the total number of hops made by all discovery agents, i.e., $l' = H$. The total number of discovery agents generated in the network is determined by the number of receivers and the number of cycles in the network. To each receiver, there must be one discovery agent dispatched from

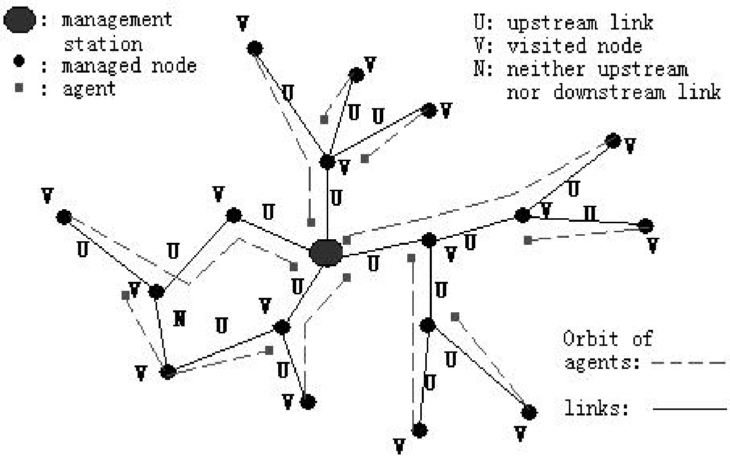


Figure 3.3: Network topology discovered by flooding algorithm using mobile agents

the root. Each cycle connecting the root to either one “end-node” (when the cycle contains an even number of links) or two “end-nodes” (when the cycle contains an odd number of nodes), where an “end-node” has an “upstream” link and an “N” link on the cycle which are marked by two discovery agents dispatched to the node. While one agent will continue search on a downstream link, the other agent will be terminated because the end-node is already marked “visited” by the first agent. Therefore, each cycle will add two additional discovery agents if the cycle contains an odd number of links, and one discovery agent if it contains an even number of links, all terminated at “end-nodes”, into the network. Assuming that both cases occur at an equal probability, we have that the total number of discovery agents is:

$$M = r' + \frac{3}{2}\mathcal{C}, \tag{3.11}$$

where $\mathcal{C}$ is the number of cycles in the networks.

Thus, the average number of hops a mobile agent made to finish the topology discovery task is:

$$E(N) = \frac{H}{M} = \frac{l'}{r' + \frac{3}{2}\mathcal{C}}. \tag{3.12}$$

Then, according to Equation (3.10) and (3.12), the average life span of a mobile agent in the network shown in Figure 3.3 is,

$$E(T_L) = E(N) \cdot E(t_k) \quad \doteq \quad \frac{l' \cdot E(T_C)}{r' + \frac{3}{2}\mathcal{C}}, \tag{3.13}$$

where $E(t_k)$ can be approximated to $E(T_C)$ because $t_k, 1 \leq k \leq N-1$, is the cycle time of the mobile agent except t_N doesn't include the travel time.

Thus, we can estimate the total time T_{TD} required to finish topology discovery. An upper bound and a lower bound are given by the following inequality. Consequently, while the management station initiates the topology discovery task, it can estimate when the task will be finished so that it can start to construct the topology based on the collected information.

$$E[T_L] \leq T_{TD} \leq E[M] \cdot E[T_L]. \tag{3.14}$$

The lower bound is given because the time of topology discovery must be greater than the expected life span of a mobile agent. However, the time required by the topology discovery task is less than the total life span of all the mobile agents.

3. Analysis on the life span distribution function

In order to describe the variable life span, we assume that the life span pdf is $l(t)$, and denote the Laplace transform of $l(t)$ by $L(s)$.

$$\begin{aligned} L(s) &= E[e^{-sT_L}] \\ &= \sum_{n=1}^{\infty} E[e^{-sT_L}|N = n]Pr(N = n) \\ &= \sum_{n=1}^{\infty} E[\prod_{k=1}^{N} e^{-st_k}|N = n]Pr(N = n) \\ &= \sum_{n=1}^{\infty} (C(s))^{n-1}D(s)Pr(N = n), \end{aligned} \tag{3.15}$$

where, $C(s) = E[e^{-st_k}]$, $1 \leq k \leq N - 1$ and $D(s) = E[e^{-st_N}]$ because t_N is the dwell time of the mobile agent when it visits the N-th host.

In the task of Internet topology discovery(ITD), a mobile agent encounters either a leaf node or a non-leaf node by the algorithms proposed in Section 3.3.1. Suppose a mobile agent encounters a leaf node with probability p_I, and a non-leaf node with probability $1 - p_I$, in a hop. As we have said, the mobile agent is terminated at leaf nodes. Therefore, the probability that a mobile agent is terminated after the n-th hop is,

$$Pr_I[N = n] = (1 - p_I)^{n-1}p_I. \tag{3.16}$$

In the task of multicast topology discovery(MTD), the nodes a mobile agent visits are either internal un-visited routers or receivers because there isn't cycle in multicast network. Assume that p_M is the probability of a mobile agent encountering a leaf node in each hop. So the probability of a mobile agent that is terminated after the n-th host-visit in the MTD case is,

$$Pr_M[N = n] = (1 - p_M)^{n-1}p_M. \tag{3.17}$$

Then $L(s)$ can be obtained by replacing Pr in Equation (3.15) with Pr_I in Equation (3.16) for ITD case and with Pr_M in Equation (3.17) for the MTD case respectively.

We can determine the pdf of life span distribution by inverse Laplace transform of $L(s)$. The mean and variance of the life span of a mobile agent are given as follows.

Lemma 10 *The mean and variance of T_L can be calculated by the following equations.*

$$E[T_L] = -\frac{dL(s)}{s}|_{s=0}, \tag{3.18}$$

$$D[T_L] = \frac{\partial^2 L(s)}{\partial s^2}|_{s=0} - [\frac{dL(s)}{ds}|_{s=0}]^2. \tag{3.19}$$

Proof of Lemma 10 is identical to the proof of Lemma 9.

3.4.3 Report time distribution at the management station

Report time is defined as the time from the beginning of the topology discovery task to a report generated by a mobile agent. The report time distribution can be applied to analyze the interreport time distribution which will be discussed in Section 3.4.4.

The report time in persistent report case can be easily obtained which is a multiple of cycle times. We mainly study the report time in the intermittent report case. The RN algorithm for Internet topology discovery and the RL algorithm for both Internet and multicast network topology discovery belong to this case. In the intermittent reporting case, the cycle time of those mobile agents that don't spawn report agents includes only mark time and travel time.

We assume that the report time of a mobile agent denoted by T_R is the time from the initiation of the topology task to the spawn of report agent observed at the management station. Let T_I be the time interval between two reports: $T_I = T_{R_1} - T_{R_2}$, $T_{R_1} \geq T_{R_2}$. Figure 3.4 shows the relationship between T_R and T_I.

According to the model, no matter a mobile agent is generated early or late, T_R is composed of several cycle times and one dwell time. Then,

$$\begin{aligned} E[T_R] &= E[\text{Report time from the beginning}] \\ &= E[t_1 + t_2 + \ldots + t_{H-1} + t_H], \end{aligned} \tag{3.20}$$

where H is the total hops from the management station to the host that the report agent is spawned. t_H doesn't include travel time.

We assume $r(t)$ to be the pdf of the report time T_R. Thus, the Laplace transform of $r(t)$ denoted by $R(s)$ can be expressed by the following equation.

$$R(s) = \sum_{n-1}^{\infty} E[e^{-st_k}|H = n] \cdot Pr[H = n]. \tag{3.21}$$

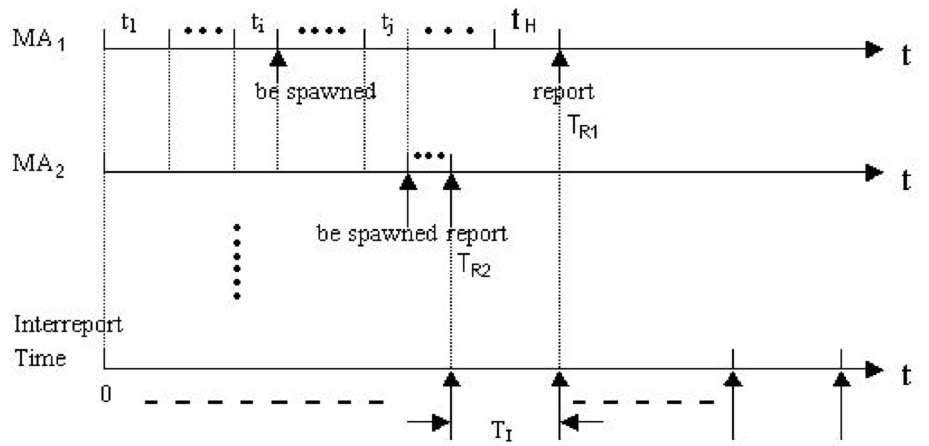


Figure 3.4: Interreport time in intermittent reporting case

For RL and RN algorithms in the intermittent report case, the probability a mobile agent reports at the H-th hop host from the root is given as follows. In the past $H-1$ hops itself and its parent mobile agent haven't generated any report agent. In the ITD case, we have assumed that a mobile agent encounters a leaf node in a hop with probability p_I. Suppose the mobile agent encounters a "visited" node (caused by a cycle) with probability q_I. Thus,

$$Pr[H=n] = \begin{cases} (1-p_M)^{H-1}\cdot p_M & \text{RL algorithm for MTD,} \\ (1-p_I)^{H-1}\cdot p_I & \text{RL algorithm for ITD,} \\ \begin{cases} 1 & H=1 \\ q_I^{H-1}\cdot(1-q_I) & H>1 \end{cases} & \text{RN algorithm for ITD.} \end{cases}$$

Then the pdf of intermittent interreport time distribution $r(t)$ can be obtained by inverse Laplace transform of $R(s)$. The mean and variance of interreport time in the intermittent case can be determined by the following Lemma.

Lemma 11 *The mean and variance of the report time of any mobile agent T_R can be calculated by the following equations.*

$$E[T_R] = -\frac{dR(s)}{s}|_{s=0}, \tag{3.22}$$

$$D[T_R] = \frac{\partial^2 R(s)}{\partial s^2}|_{s=0} - [\frac{dR(s)}{ds}|_{s=0}]^2. \tag{3.23}$$

The proof of Lemma 11 is similar to that of Lemma 9.

3.4.4 Interreport time distribution at the management station

Interreport time is defined as a random variable that describes the time interval between the arrival time of the previous report and that of the next report at the management station as shown in Figure 3.4. Because the report time of each mobile agent follows the report time distribution discussed above, we can analyze the distribution of interreport time at the management station. Accordingly, we can measure the burden that each algorithm brings to the management station based on the analysis.

As shown in Figure 3.4, we have that $T_I = T_{R_1} - T_{R_2}$. Assume the pdf of interreport time is $i(t)$, it can be deduced by the pdf of the report time of any mobile agent $r(t)$.

$$i(t) = r(t) * r(-t). \tag{3.24}$$

Then according to Equation (3.24), we can get the Laplace transform of $i(t)$ denoted by $I(s)$:

$$I(s) = R(s) \cdot R(-s). \tag{3.25}$$

Similarly, we can determine the mean and variance of interreport time by the following Lemma.

Lemma 12 *The mean and variance of the interreport time at the management station can be calculated by the following equations.*

$$E[T_I] = -\frac{dI(s)}{s}|_{s=0}, \tag{3.26}$$

$$D[T_I] = \frac{\partial^2 I(s)}{\partial s^2}|_{s=0} - [\frac{dI(s)}{ds}|_{s=0}]^2. \tag{3.27}$$

Lemma 12 can be derived similarly to Lemma 9.

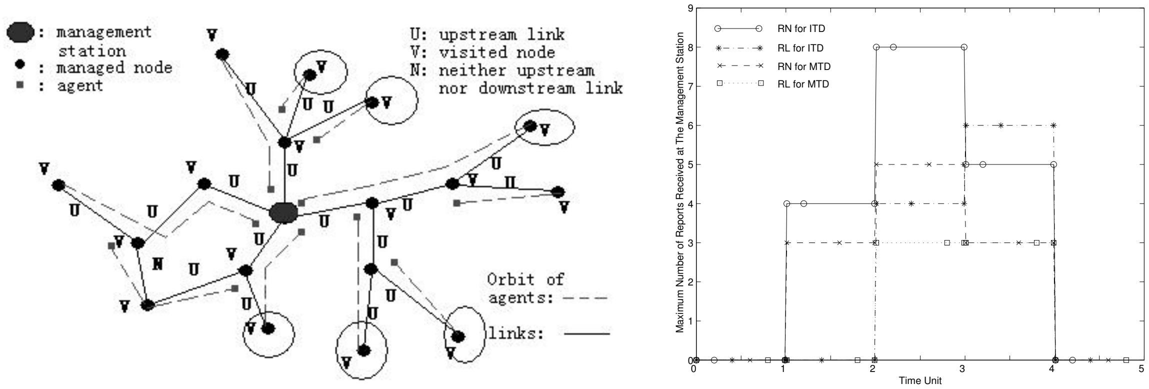
The RN algorithm for multicast topology discovery follows persistent report. In this case, the time interval of two reports generated by the same mobile agent is one cycle time. The interreport time observed at the management station also approximates to one cycle time if the statuses of all links are similar. Denoting interreporting time in persistent report case by T_P, we have

$$E[T_P] = E[T_C]. \tag{3.28}$$

Obviously, the mean of the persistent interreport time is less than that of the intermittent interreport time. From the interreport time point of view, we compare the burdens of these two schemes to the management station in a given network in the next section.

3.4.5 Comparison between the RN and RL Algorithms

According to the above analysis, we can conclude that the RN algorithm reports more frequently than the RL algorithm and thus brings heavier burden to the management station. The management station has to spend more time in handling all the topology information. However, the RL algorithm generates much fewer report agents, which results in less burden to the management station. Our experiments show that overall the RL algorithm brings less burden to the management station and therefore is superior to the RN algorithm in this sense. If considering system reliability, RN is better than RL because the network topology can still be reconstructed even if some reports are lost. Figure 3.5 shows the superiority of the RL algorithm to the RN algorithm. The given network topology is shown in Figure 3.5(a), where the multicast membership is the receivers with big circles. The X-axis in Figure 3.5(b) describes the unit time as one cycle time of a mobile agent.



(a) A given network.

(b) Report number of different algorithms.

Figure 3.5: Comparison of burdens to the management station by different algorithms

However, the RN algorithm is more reliable than the RL algorithm in the event of heavy congestion in the network. Because any lost information of one node can be extracted from the information of its neighborhood in the RN algorithm. As to the RL algorithm, the lost information of a node will cause unrecoverable loss for the topology reconstruction unless the task of topology discovery is initialized again to search the lost nodes.

3.5 Concluding Remarks

Mobile agents based topology discovery algorithms for Internet and multicast networks have been proposed in this chapter, which can construct the required network topology effectively. The network model has been built for analyzing the statistical characteristics of the mobile agents for topology discovery. In order to avoid the endless spawning of a mobile agent, the management station may enforce the mobile agent to jump within a limited number of steps by simply setting a timer inside the agent. Laplace transform and inverse laplace transform are applied to analyze the behavior of mobile agents. According to the analytical results, we can determine the mean, variance and distribution function of dwell time at a host, life span, report time of a mobile agent and interreport time at the management station.

Applying the proposed mobile agents based topology discovery algorithms to a given network, we can obtain more practical performances of the mobile agents based on the analytical results. For example, it has been estimated how long mobile agents need to collect all the topology information to finish topology discovery. We have also compared the burdens on the management station from the proposed algorithms in the network. It has been shown that the report-at-newly-found-nodes algorithm is more costly from the point of view of the network load and burden to the management station than the report-at-leaf-nodes algorithm, and is, however, more reliable from the system model's point of view. For the efficiency of the proposed algorithms and the inherent advantages of mobile agents, it is promising to develop more significant applications based on our methods and analytical results.

Chapter 4

Topology Analysis in Wireless Sensor Networks and Its Applications

We study topology analysis in wireless sensor networks in this chapter. Because the topology of wireless sensor networks changes less compared with other types of wireless networks such as cellular networks and mobile ad hoc networks, it gives more significance to discover, maintain and employ the topology information of wireless sensor networks. Furthermore, because the topology of wireless sensor networks (WSNs) can provide great help to network routing and resource management, it motivates our study on the topology analysis for wireless sensor networks and the routing algorithms development based on the topology information. In the following sections, we first give some necessary background knowledge of WSNs, followed by a description on some fundamental performance metrics of WSNs. Then we study some desirable topology patterns which can provide both coverage for required area and connectivity between all nodes deployed in WSNs. Based on the analytical results on these patterns, we develop the route selection function-based routing protocols and the random walk routing protocol.

4.1 Preliminaries of Wireless Sensor Networks

The advancement in wireless communication and sensor technology is expediting the development of WSNs, which have a wide range of environmental sensing applications such as danger alarm, vehicle track, battle field surveillance, habitat monitor, etc. [5, 35]. A WSN consists of hundreds to thousands of sensors and a base station. To gather information from the environment and deliver the processed messages to the base station, each sensor is capable of collecting, storing, processing signal, and communicating with neighbors. The base station decides if an unusual or concerned event occurs in the sensing area after aggregation and analysis of the messages from the sensors.

Similar to mobile ad-hoc networks (MANET), WSNs apply multi-hop communications where the packets sent by the source node are relayed by several intermediate nodes before reaching the destination node. However, they are significantly different in several aspects. First, the communication mode of a WSN is mainly many-to-one, that is, multiple sensor nodes send data to a base station or aggregation point in the network, whereas MANET support communication between any pair of nodes. Second, unlike MANET, data collected by different sensor nodes in a WSN might be the same and needs to be processed in the intermediate nodes. Third, in most envisioned scenarios the sensor nodes are immobile and keep on sensing and monitoring the area assigned beforehand until the system energy is exhausted. Finally, the energy constraint is much more stringent in WSN than that in MANET because the communication devices handled by human users can be replaced or recharged relatively often in MANET, whereas battery recharging or replacement is usually infeasible for a WSN because it's often deployed in hostile or inhospitable places. Thus, maintenance of unattended sensor nodes in a WSN to lengthen the system lifetime becomes extremely important when deploying the WSN. All of these properties of sensor networks, in turn, highlight one fundamental and important issue, that is how to keep connected coverage in a WSN with as less power consumption in routing as possible.

The connected-coverage problem is to achieve two goals when deploying sensor nodes: coverage and connectivity [41]. Coverage is to ensure that the entire physical space of interest is within the sensing range of at least one of the active sensors. Connectivity is to ensure that all the sensor nodes can communicate with the base station by either single-hop or multi-hop path. The connected-coverage problem in WSNs can easily be solved if the number of sensor nodes and energy-constraint needn't to be concerned. However, it is not possible to construct a connected-coverage WSN without energy and economy concerns in practice. Energy-constraint, instead, is extremely stringent in WSNs. Therefore, it is significant to study how to construct a connected-coverage WSN while consuming as least energy as possible so as to maximize the networking lifetime which is defined as the duration that the WSN provides satisfactory performance on sensing and transmission.

Networking lifetime is directly affected by power consumption in the procedure of sensing, communication and data processing. Since all the sensor nodes are battery-powered, it is paramount to develop efficient methods to save energy. The existing work to save energy can be classified into two categories. One is turning off or changing some nodes to sleep mode as proposed in [20, 39, 61]. The other is minimizing sensing range and transmission range while keeping connected coverage as proposed in [44, 51]. As will be discussed in later sections, we address the issue of keeping connected coverage with power efficiency by studying the topology patterns which is different from above methods. It minimizes the redundant overlapping among neighboring sensor nodes while keeps required connectivity with least sensor nodes alive to save energy.

Our study is different from existing studies in either connectivity and coverage problems, or in energy-efficiency routing protocol. It addresses all of these problems by performing topology patterns analysis. WSNs in patterned topologies are assured to provide longer networking lifetime than randomly deployed WSNs if the same number of sensor nodes is used in both types of WSNs. We call these patterned topologies energy efficient topologies. They have many significant applications in practice. For example, node placement in topology patterns can efficiently save energy and achieve long networking lifetime in some scenarios where priori node deployment for WSNs are possible, such as danger alarm and vehicle tracking. In such cases, study on topology patterns can guide to construct WSNs with potentially more energy saving and longer lifetime. It is also worth noting that patterned topologies can also instruct to choose duty-on nodes to keep connected coverage in a WSN and put all other sensor nodes in sleeping mode so as to avoid redundant overlapping area, save energy, and thus prolong the networking lifetime [39]. Therefore, we begin with studying different patterned topologies for WSNs and comparing their performance on different measures in the succeeding section. Later on, we propose several routing protocols for WSNs with patterned topologies based on different selection functions and compare their performance by simulation. This work provides a supplement to [39] which only address how to save energy by choosing duty-on sensor nodes based on patterned topologies. The proposed routing protocols require only local information, which are different from DSAP in [54]. Our routing protocols can achieves energy efficiency and perform in a simple and effective way. Followed by this piece of work, we propose the random walk routing in WSNs with patterned topologies which is applicable in WSNs where sensor nodes do not have capability of computation and comparison. By this routing protocol, neighboring information exchanging does not required and less computation burden is given to each sensor node, thus energy is further saved.

4.1.1 Modelling Connected-Coverage WSNs

A connected-coverage WSN is defined as a wireless sensor network that can guarantee coverage of all the required region and connectivity among all sensor nodes in the WSN. We assume the region of interest to be 2-dimensional. Assume the area of the region to be A. There are N sensor nodes and one base station placed in the region. Each sensor node deployed in the region can sense any event within the disk with radius r_s centered at the sensor node. Each sensor node can communicate with other sensor nodes whose Euclidean distance between them is no more than r_t, that is, nodes s_1 and s_2 can communicate with each other if their Euclidean distance $Ed(s_1, s_2) \leq r_t$. Otherwise, they cannot. The sensing radius of a sensor node can be either equal or unequal to its communication radius in the WSN.

The sensing ability of each sensor node diminishes as distance increases. In [47], the sensing ability at point y of sensor node s_i is assumed to be inversely proportional to $Ed(s_i, y)^k$ where k is a sensor technology-dependent parameter. This characteristic of sensor nodes introduces an important parameter, we call it sensing strength factor d_{mm}, stating how well region A is covered and sensed. If we define $min_i Ed(s_i, y)$ as the distance of point y to its closest sensor node, $y \in A$, then all points in A have a distance at most $max_{y \in A} min_i Ed(s_i, y)$. We use d_{mm} to denote this distance:

$$d_{mm} = max_{y \in A} min_i Ed(s_i, y).$$

Thus d_{mm} is the maximum distance from any point to its closest sensor node. Usually a WSN is required to be deployed with a particular sensing strength factor equal to d_{mm} so that distance from any point to its closest sensor node is no more than d_{mm} to ensure coverage and sensing strength. The less d_{mm} is, the better each point is sensed in the WSN. In [49] and [10], similar parameters can be found, but they were proposed for other applications.

The power consumption is another important parameter to measure how much energy different topology patterns can save for WSNs. Since each sensor node usually includes a sensing unit, a processing unit, a transceiver unit and a power unit as modelled in [5], power consumption can be divided into three domains: sensing, communication, and data processing. Of the three domains, we are only concerned with the maximum energy spent by a sensor node in data communication. This involves both power consumed in data transmission, denoted by P_t, and in data reception, denoted by P_r. That is, the power consumed by a sensor node is $P_s = P_t + P_r$.

4.2 Patterned Topologies for Connected-coverage WSNs

As we have discussed in in [64], sensor nodes can be placed in hexagon, square, and triangle-based topologies. In [39] strip-based topology has also been proposed to place nodes to construct a connected-coverage topology for WSNs. We will discuss all these topology patterns in this section and compare the performance of WSNs in different patterns. The case that the sensing range of a sensor node equals to its transmission range is discussed in the comparison part. We begin with additional sensing area analysis, followed by topology analysis for different coverage schemes.

4.2.1 Analysis on additional sensing area

We define the new sensing area provided by adding a sensor node as the additional sensing area of this node. The shadow area in Figure 4.1 is the additional sensing area of node j. We denote it by S_{A_j}. Let d be the distance between nodes i and j. Assume the sensing area of sensor node i to be S_i. Thus we can derive $S_{A_j} = S_j - S_i \cap S_j = \pi r^2 - INTS(d)$, where $INTS(d)$ is the intersection area of two circles covered by two nodes whose distance is d.

$$INTS(d) = 4\int_{d/2}^{r} \sqrt{r^2 - x^2}dx \tag{4.1}$$

When $d > r$, i and j cannot communicate with each other. If either i or j does not have any else neighbor nodes, the sensor node will be isolated so that it cannot report any sensed event to the base station. Thus, each sensor node in the network must keep at least one neighbor whose distance from it is no more than r as Lemma 1 will give. Even if the additional sensing area provided by j if $d > r$ ($d \leq 2r$) would be great, i and j need the third sensor node to cover both of them, which in turn, results in the actual additional area of j being the additional area under the condition $d \leq r$. Therefore, the additional area of j is studied under $d \leq r$. When $d = r$, the additional area S_{A_j} is the largest, which equals $\pi r^2 - INTS(d) = r^2(\frac{\pi}{3} + \frac{\sqrt{3}}{2}) \approx 0.61\pi r^2$.

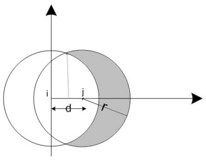


Figure 4.1: Additional area provided by sensor node j

Lemma 13 *In order for a WSN to sense any abnormal event in its covered area, each sensor node must have at least one neighbor node whose distance from it is no more than its communication range, i.e. $d \leq r$.*

We then work on how to deploy sensor nodes under the above constraint and cover as large area as possible which may be in any shape. The simplest way is deploying sensor nodes in a linear array, but it is only limited to a strip area to be covered. We have to find a general approach to cover the area in any shape. Thus, the strips can be deployed into parallel array.

As we have derived, the additional sensing area by deploying a new sensor node is maximized when $d = r$ under the condition of their communication availability. In a linear array-deployed network, assuming in horizontal direction, every two sensor nodes share a sensor node with distance r to maintain communication. That is, each sensor node has two neighbors. To enable the communication along vertical direction while maximize the coverage area, we set a sensor node connecting every two strips. The most efficient way to keep vertical communication is let this sensor node having three neighbors because the arc covered by a neighbor with largest additional area is $2\pi/3$. Thus, to maximize the coverage while minimize the number of sensor nodes, deploying parallel strips and keeping a node connecting every two strips is the optimal scheme. This scheme is clearly specified in Figure 4.2 of the next subsection.

4.2.2 WSNs with Strip-based Topology

To keep the connectivity of two sensor nodes, their distance should be no more than r_t. To maximize the coverage area sensed by use of a fixed number of sensor nodes, [39] proposes a strip-based topology as in Figure 4.2. To compare it with other WSNs with different topologies, we place the same number of sensor nodes (25) for all WSNs.

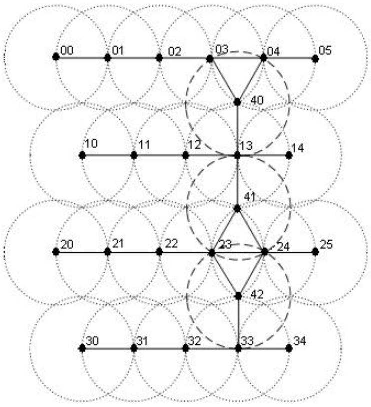


Figure 4.2: A WSN with strip-based topology

The strip-based WSN in Figure 4.2 clearly shows that sensor nodes 40, 41 and 42 connect 4 self-connected strips $00 - 05$, $10 - 14$, $20 - 25$ and $30 - 34$. By this way of node placement, these sensor nodes construct a connected WSN with strip-based topology. The total number of sensor nodes is 25. We assume node 05 to be the aggregation node.

4.2.3 WSNs with Hexagon-based Topology

In hexagon-based WSNs, each sensor node has three neighbor nodes located uniquely around the node. Connecting all sensor nodes to their neighbor nodes obtains the minimum unit in the shape of hexagon. Thus the WSN in this topology pattern is called the hexagon-based WSN. The distances of the node to its neighbor nodes are all set to r_t so that direct communication is available between the node and its neighbor nodes, and each neighbor provides maximal additional sensing area [64]. Figure 4.3 specifies a WSN with hexagon-based topology. The number of deployed sensor nodes is 25 as above. We assume node 06 to be the aggregation node which plays the role of aggregating the sensed information in the WSN and reporting to the base station.

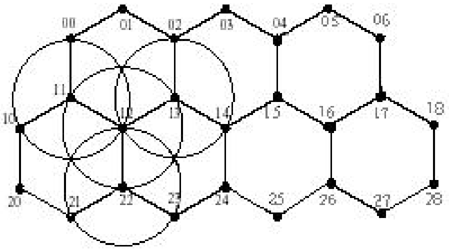


Figure 4.3: A WSN with hexagon-based topology

4.2.4 WSNs with Square-based Topology

In square-based WSNs, each sensor node has four neighbor nodes located uniquely around the node. Connecting all sensor nodes to their neighbor nodes obtains the minimum unit in the shape of square. Thus the WSN in this topology pattern is called the square-based WSN. The distances of the node to its neighbor nodes are set to r_t. A WSN composed of 25 sensor nodes is given in Figure 4.4. Node 04 is assumed to be the aggregation node.

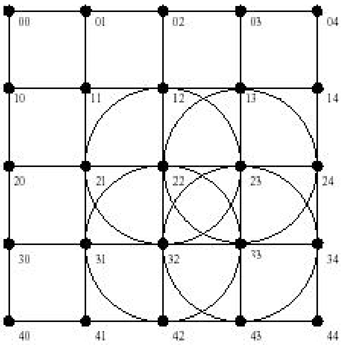


Figure 4.4: A WSN with square-based topology

4.2.5 WSNs with Triangle-based Topology

In triangle-based WSNs, each sensor node has six neighbor nodes located uniquely around the node. Connecting all sensor nodes to their neighbor nodes obtains the minimum unit in shape of triangle. Thus the WSN in this topology pattern is called the triangle-based WSN. Same as above, the distances of the node to its neighbor nodes are set to r_t. Figure 4.5 is a triangle-based WSN with 25 sensor nodes deployed. Node 04 is assumed to be the aggregation node.

In WSNs with triangle-based topology, we find that every point within the area is covered by at least two sensor nodes. We call the reliability provided by such kind of node placement

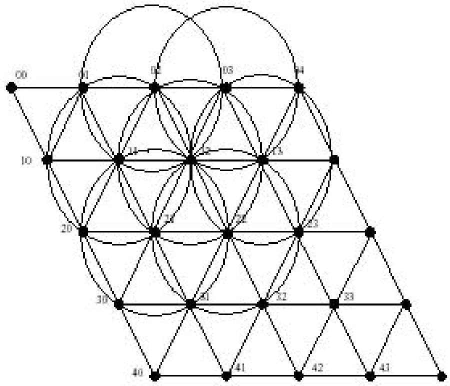


Figure 4.5: A WSN with triangle-based topology

2-reliability. A 2-reliability WSN can maintain its connected-coverage for any single sensor node failure. When every point is covered by at least k sensor nodes, the sensor network is called k-reliability. The WSNs with other patterned topologies are 1-reliability as we have discussed in [64].

4.2.6 Performance Comparison

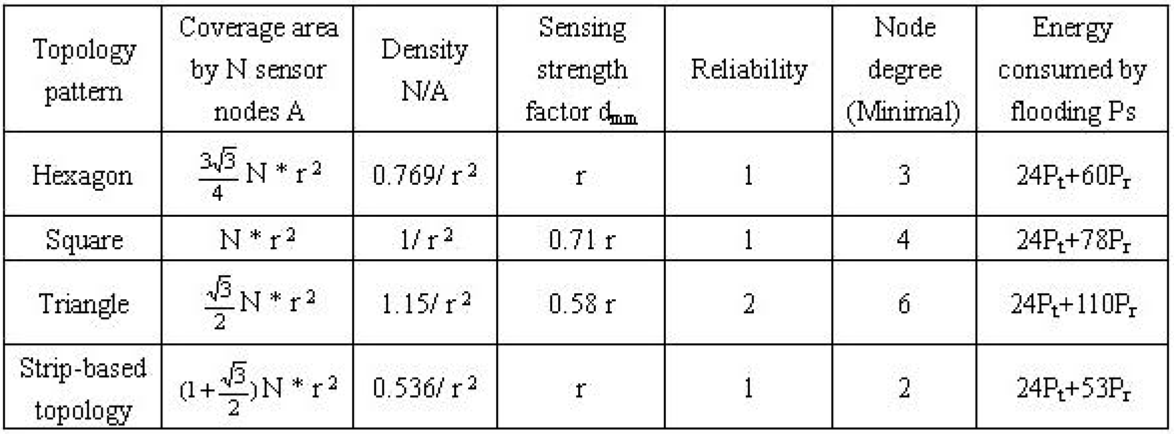
Given the same number of sensor nodes, we compare the above four types of patterned topologies on coverage area, coverage density, sensing strength factor, reliability, node degree and energy consumption. The coverage area is denoted by A as mentioned above. The coverage density is the number of sensor nodes per area unit which can be obtained by $D = \frac{N}{A}$. The sensing strength factor d_{mm} is the maximal distance of a point to its closest sensor node as described in Section 2. The value of k in k-reliability denotes how many sensor nodes each point in the WSN is at least covered by. Node degree can be used to measure the resilience of a WSN because it denotes how many routing neighbor options each sensor node can have. Energy consumption is the most important measurement for routing protocol.

We assume $r_t = r_s = r$ in all WSNs with four different topologies. For energy consumption comparison, we fix the destination to be the aggregation node as designated above. The source is fixed to be a node with distance $4r$ from the destination. In this case, the less the energy is consumed, the better the topology pattern is.

We first calculate the coverage area by all WSNs with four different topologies. For triangle-based WSNs, it's easy to get the coverage area of each triangle is $s_{\triangle} = \frac{\sqrt{3}}{4}r^2$. Each node is a corner of six triangles. Each triangle has three corners. Thus N nodes approximately composes $2N$ triangles. The coverage area of triangle-based WSNs by N nodes can be approximate to $S_{\triangle} = \frac{\sqrt{3}}{2}Nr^2$. For square-based WSNs, the coverage area of each square is $s_{\square} = r^2$. Each node is a corner of four squares. Each square has four corners. Thus N nodes approximately composes N squares. The coverage area of square-based WSNs by N nodes is $S_\square = Nr^2$. Similarly, N nodes in hexagon-based WSNs composes approximately $\frac{N}{2}$ hexagons. The area of each hexagon is $s_\odot = \frac{3\sqrt{3}}{2}r^2$, thus the coverage area of the hexagon-based WSN by N nodes is approximate to $S_\odot = \frac{3\sqrt{3}}{4}Nr^2$. For strip-based WSNs, we assume each strip comprises M nodes, thus the WSN has $\frac{N}{M}$ strips in total. The distance between two neighboring strips is $(1+\frac{\sqrt{3}}{2})r$ which can be easily calculated according to Figure 4.2. Thus the coverage area of strip-based WSNs by N nodes can be approximate to $S_\sim = Mr \cdot \frac{N}{M}(1+\frac{\sqrt{3}}{2})r = (1+\frac{\sqrt{3}}{2})Nr^2$.

The coverage density can accordingly be obtained by $D = \frac{N}{A}$. Therefore, the densities of triangle (square, hexagon, strip) -based WSNs are $D_\triangle = \frac{1.15}{r^2}$, $D_\square = \frac{1}{r^2}$, $D_\odot = \frac{0.769}{r^2}$, $D_\sim = \frac{0.536}{r^2}$. In [39], it has been proved that the density of nodes required by the optimal topology (OPT) providing connected-coverage was $d_{OPT} \geq \frac{0.522}{r^2}$. We can see the densities of nodes in all topology patterns discussed in this report are greater than $\frac{0.522}{r^2}$. We also note that for a WSN with nodes being randomly deployed, [39] empirically obtained by simulation that 90% area will be connected and covered if the density of nodes is around $\frac{1.45}{r^2}$. If the density of nodes is less than $\frac{1.45}{r^2}$, the fraction of connected-coverage area decreases abruptly for randomly deployed WSNs. However, for WSNs with patterned topology, the densities can be less than $\frac{1.45}{r^2}$ (even half or $\frac{1}{3}$ of $\frac{1.45}{r^2}$) and keep connected coverage as well. It is also worth noting that WSNs in triangle-based topology provide 2-reliability with the density $\frac{1.15}{r^2}$ which is much less than $\frac{1.45}{r^2}$. Therefore, WSNs in patterned topologies can efficiently save the number of sensor nodes and thus economy costs. It will be shown that WSNs in patterned topologies can also have more energy-efficient routing protocol than randomly deployed WSNs.

The table in Figure 4.6 gives the results of these performances comparison.



Topology pattern	Coverage area by N sensor nodes A	Density N/A	Sensing strength factor d_{mm}	Reliability	Node degree (Minimal)	Energy consumed by flooding Ps
Hexagon	$\frac{3\sqrt{3}}{4}N * r^2$	$0.769/r^2$	r	1	3	$24P_t+60P_r$
Square	$N * r^2$	$1/r^2$	0.71 r	1	4	$24P_t+78P_r$
Triangle	$\frac{\sqrt{3}}{2}N * r^2$	$1.15/r^2$	0.58 r	2	6	$24P_t+110P_r$
Strip-based topology	$(1+\frac{\sqrt{3}}{2})N * r^2$	$0.536/r^2$	r	1	2	$24P_t+53P_r$

Figure 4.6: Performance comparison among WSNs with different patterned topologies

From the table in Figure 4.6, we can see that strip-based topology provides maximal connected-coverage with the same number of sensor nodes and consumes least energy by the routing protocol of flooding. WSNs in triangle-based topology provide the best reliability and the best sensing strength while trading off total coverage area and energy consumption.

4.3 Route Selection Function-based Routing Protocols in Patterned WSNs

We propose several routing protocols in this section. Different from Directional Source-Aware Protocol (DSAP) [54] where each node must have the knowledge of global information of topology, our routing protocols only require local information.

We define a route selection function $f(h, s)$ for a sensor node to choose neighbor nodes when routing the message back to the aggregation node. The function is determined by the hop count value h of neighbor nodes and stream units s which has been sent by neighbor nodes. Here we assume the stream sent by a sensor node can be measured by stream unit, thus s means how many units have been sent by the sensor node. We denote the battery life of sensor node i by b_i.

We propose three approaches to route back the message for different aims. All of them are based on the route selection function $f(h, s) = \alpha h + \beta s$.

> **Approach 1:** Maximize the total energy saving for WSNs, i.e., $B = \sum_i b_i$: This can be obtained by minimizing first the hop count value h when choosing next-hop neighbor and then minimize s. In this case, $\alpha = 1$, and $\beta = \sigma$, where σ is a small number which approximates to 0.
>
> **Approach 2:** Maximize the minimal battery life at all sensor nodes, i.e., $min_i b_i$: This can be obtained by minimizing first the stream units s of next-hop neighbor and then h. In this case, $\alpha = \sigma$, and $\beta = 1$.
>
> **Approach 3:** Maximize networking lifetime by considering both total energy saving and minimal battery life: This can be obtained by minimizing the sum of h and s. In this case, $\alpha = 1$, and $\beta = 1$.

We name the protocols as route selection function-based protocols. It works as follows:

1. Distance identification: The aggregation node floods the discovery message in the WSN with a determined TTL value. Each sensor node records its distance from the aggregation node by hop count. If a sensor node receives several broadcast messages, it records the least value of hop count.

2. Data collection: When a sensor node senses any abnormal event and needs to report the event, it chooses a neighbor with minimized $f(h, s)$ to route back the message.

Thus, our protocols with different route selection functions can achieve maximal total energy saving, maximal individual node's battery life, and maximal networking lifetime respectively. Approach 1 achieves less transmission delay by trading off the networking lifetime. Approach 2, instead, tries to prolong the networking lifetime by trading off transmission delay. By combining the both measurements hop count and stream units together, the networking lifetime can be maximized. The route selection function-based routing algorithm initiated by any node i is given as follows: (Assume each node in the WSN has n neighbors.)

```
Step 1: For $n$ neighbors
            node $i$ periodically inquires $h$ and $s$ from its neighbors and
            update the information in its routing table;
Step 3: For $j = 1$ to $n$
            Calculate $f_j(h, s)$ to get $min_j f_j(h, s)$ and mark $j = p$ as its
            next-hop neighbor;
Step 4: Send message to node $p$;
Step 5: $s_i = s_i + k$ //$k$ is stream units.//
```

To compare the performance of our protocols, we simulate the square-based WSN with the routing protocol for simplicity. We assume the networking lifetime is from the start to the time that any node exhausts its power in the WSN since one node failure results in an unconnected coverage for WSNs with square-based topology.

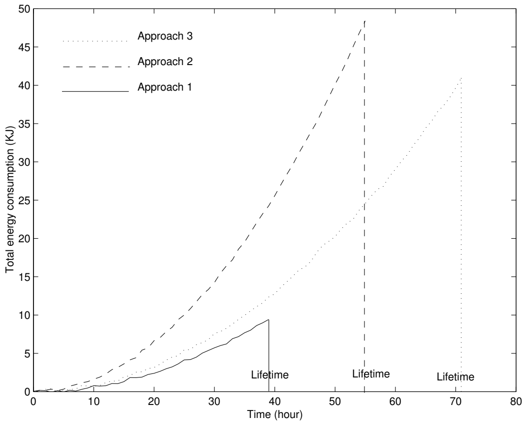
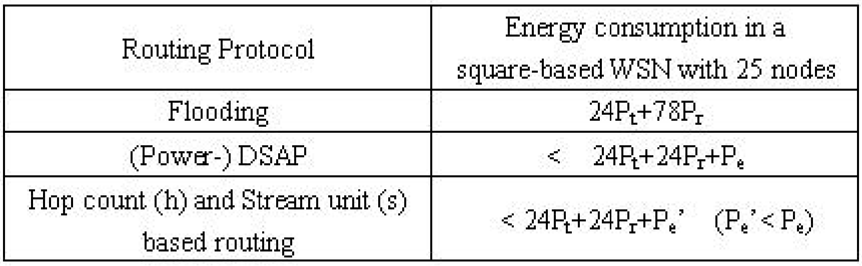


Figure 4.7: Energy consumption and lifetime comparison among three approaches

From Figure 4.7, we can see that Approach 1 provides least network lifetime. Approach 2 gets a longer lifetime than approach 1 and, however, trades off much more energy consumption by choosing a longer path to the aggregation node in the WSN. Approach 3 can provide the longest lifetime, which is almost as twice as that provided by Approach 1 because it tries to find a shorter path and a next-hop neighbor with more energy in every step.

The routing protocol proposed above is compared with flooding protocol, DSAP and Power-DSAP [54]. Figure 4.8 gives the comparison result.



Routing Protocol	Energy consumption in a square-based WSN with 25 nodes
Flooding	$24P_t+78P_r$
(Power-) DSAP	< $24P_t+24P_r+P_e$
Hop count (h) and Stream unit (s) based routing	< $24P_t+24P_r+P_e'$ (P_e' < P_e)

Figure 4.8: Energy consumption and lifetime comparison among three approaches

In Figure 4.8, P_e and P_e' are the extra energy consumed in the network initiation stage, neighbor information maintenance and so on. They are much less than energy consumed in sensing data aggregation. $P_e' < P_e$ because our routing protocol does not require global information maintenance. Therefore, Figure 4.8 shows our routing protocol consumes less energy than flooding, DSAP, and Power-DSAP in each message aggregation. Thus it's potentially energy efficient in routing for WSNs due to consideration of both hop count and stream unit.

4.4 Random walk routing protocol

In this section, we will discuss a more energy-efficient routing protocol, namely, random walk routing protocol. It is especially developed for small-size data collection in WSNs.

Generally speaking, the sensed data are classified into three types: large-size data, mid-size data, small-size data. Small-size data is defined to be in comparative size with the inquiry data among neighboring nodes for choosing the next-hop neighbor. It is a beep-like message and contains only several bits besides the header, destination and source ID information. Large-size data are those complex data such as image. Mid-size data lies in the midst of small-size and large-size. Most routing protocols are developed for mid-size data transmission. While the family of Sensor Protocols for Information via Negotiation (SPIN) [37, 45] is especially appropriate for large-size data transmission because it applies data negotiation before sending the real large-size data. By avoiding redundant large-size data transmission, SPIN family can save a lot of energy. When transmitting small-size data, flooding or other routing protocols which do not have special constraints can be chosen as alternative protocols.

The random walk routing protocol in this section is mainly proposed for small-size data transmission in WSNs with patterned topologies. As mentioned above [64,69], deploying WSNs in patterned topologies can help to efficiently save energy and achieve long networking lifetime and thus have many applications in environment monitor, danger alarm and so on [64, 69]. In these applications, sensor nodes need report their status to the Base Station from time to time by sending a short message. How to route these messages from sensor nodes to the BS efficiently in these WSNs is a very important issue that we aim to solve. When the data size is comparatively small with the query message between neighboring nodes, we find that routing packets based on random walk can obtain significant performance improvement. We thus propose the routing based on random walk, for which we just simply name as the random walk routing. The main advantages of the random walk routing are listed as follows. First, it does not require any location information which is needed in DSAP [54] developed also for WSNs with patterned topologies. Second, the random walk routing achieves load balancing property inherently for WSNs which is difficult for other routing protocols. Third, it is proved that the random walk routing consumes the same amount of energy as the shortest path routing in the scenarios where the message required to be sent to the BS is in comparatively small size with the inquiry message between neighboring nodes. As we know, the shortest path routing, if the shortest paths can be easily found in WSNs with global information, consumes the least energy among all other protocols. Our random walk protocol can route the message successfully by consuming the same amount of energy as the shortest path routing.

The idea of using random walk for routing has been proposed previously in [13] and [58]. Rumor routing in [13] applied random walk for its long-lived search agents in a randomly-deployed architecture. With agents forwarded by random walk, the route passed by the agents are recorded. Once there is a query interested in some event informed by some agents, the query will go along the previously recorded path to withdraw the interested message. We can see random walk in rumor routing is used in different architectures and goals from ours. In [58], constrained random walk in grid was studied. It set the packet to choose the next-hop neighbor only in the shortest path direction. And in the two directions of the shortest path, the probability of forwarding in each direction is recalculated in every step so that the load balancing is reached for multi-path routing. Our work in this paper is also different from [58] because we do not constrain the direction and probability of random walk in each step. The packet in our scheme does not need to calculate the probability of forwarding to different neighbors. The behavior of the packets forwarding in our random walk is truly random in this sense. To the best of our knowledge, our work is the first work that focuses on the random walk routing on patterned topologies and analyzes the successful transmission probabilities.

In this section, we will further present a density-aware deployment scheme [63] for the scenarios where the topology can be pre-deployed and the BS is fixed. The motivation comes from the consideration that though the random walk routing provides load balancing in WSN with patterned topology, the nodes near the BS are inevitably under heavier burden than the nodes far from the BS. Our density-aware deployment scheme aims to guarantee that the heavy-load nodes do not affect the networking lifetime even if they are exhausted. In this way, the networking lifetime is prolonged.

4.4.1 Description on The Random Walk Routing for WSNs

Reviewing three types of patterned topologies, hexagon-based topology provides maximal connected-coverage with the same number of sensor nodes, WSNs in triangle-based topology provide the best reliability and the best sensing strength while trading off total coverage area and energy consumption and square-based topology provides the performance which lies in their midst and the simplest architecture. Since routing packets based on random walk can achieve load balancing in a network, we now present how to apply random walk to route packets in WSNs with patterned topologies. For simplicity, we choose square topology to be the basic architecture on which we devise the routing protocol with random walk as the first stage of this work. We choose square-based WSNs is also because square-based architecture provides a moderate performance in every aspect among WSNs with all the topologies as mentioned above.

In a WSN, the way for a sensor node to report an event to the base station (or the sink) is usually forwarding a packet to a selected neighbor after communicating with all neighbors. In the scenarios where the data required to be sent back is in small size such as danger alarm system and abnormal determination system, the communication cost between each neighbor pair for choosing the next-hop neighbor is comparative with that of transmitting the real data. In square-based WSNs, each node needs to communicate with three neighbors before forwarding the data. We assume that the communication cost between any neighbor pair for inquiry is the same as the transmission cost of real data, and denote it to be one energy unit e. Therefore, the total energy cost of routing the data from a node to its next-hop neighbor is four energy units. In the square-based WSN as Figure 4.9 shows, the shortest path from the node 40 to the sink 04 is eight hops. Then the energy cost of routing on the shortest path is 32 energy units.

Now let's have a look at the energy cost if we apply routing with random walk on the square-based WSNs. Routing with random walk is that when a node receives data, it just randomly selects a neighbor and forwards the data. Since random walk routing does not require communication before forwarding the data, the energy cost of transmission to a next-hop neighbor is one energy unit. In order to analyze how the routing with random walk performs in square-based WSNs and compare it with the shortest path routing, we analyze the probability of the data being sent successfully within 8 hops (length of the shortest path), 32 hops and 40 hops respectively from node 40 to 04 in the WSN of Figure 4.9.

We assume two kinds of routing schemes with random walk. One is that a node selects a neighbor node randomly among all its neighbors including the neighbor where the data came

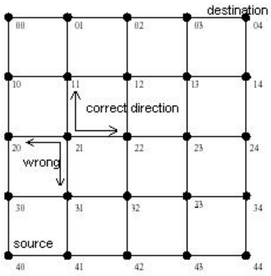


Figure 4.9: A WSN with square-based topology

from (Case 1). The other one is that a node selects a neighbor node randomly among all its neighbors except the neighbor where the data came from (Case 2).

Case 1: Firstly we calculate the probability that a packet can be sent successfully from the node 40 to 04 at 8 hops. It is obvious that the packet must be forwarded to the neighbor in either right or up direction (we call correct direction) in order to reach the destination with 8 hops. If the packet is forwarded downward or leftward, we call it is in wrong direction. Since the packet is forwarded to four neighbors randomly in Case 1, the probability of forwarding packet in either right or wrong direction is $\frac{1}{2}$ (we ignore the difference between the probabilities of non-boundary nodes and boundary nodes in the grid for simplicity). At 8 hops in total, the packet is forwarded 8 hops in correct direction and 0 hops in wrong direction, thus the probability of successful transmission from node 40 to 04 is as follows:

$$P\{d = 8\} = \binom{8}{0}\frac{1}{2^8} = 0.003906,$$

where d means the number of hops the packet is forwarded.

If the packet is forwarded one hop in wrong direction, the least number of hops the packet requires to be sent back is 10 hops. Thus the number of hops of each packets from node 40 to 04 is $d = 8 + 2 \cdot i$, $i = 0, 1, 2, \cdot$. Similar as above, with 10 hops in total, the packet is forwarded 9 hops in correct direction and 1 hops in wrong direction; with 12 hops in total, the packet is forwarded 10 hops in correct direction and 2 hops in wrong direction; and so on. Thus the probabilities of successful transmission at 10, 12, 14, …, 32, …, 40 hops are calculated as follows:

$$P_1\{d = 10\} = \binom{10}{1}\frac{1}{2^9} \cdot \frac{1}{2} \doteq 0.009766,$$

$$P_1\{d = 12\} = \binom{12}{2}\frac{1}{2^{10}} \cdot \frac{1}{2^2} \doteq 0.016113,$$

$$P_1\{d = 14\} = \binom{14}{3}\frac{1}{2^{11}} \cdot \frac{1}{2^3} \doteq 0.022217,$$

$$\vdots$$

$$P_1\{d = 32\} = \binom{32}{12}\frac{1}{2^{20}} \cdot \frac{1}{2^{12}} \doteq 0.052571.$$

$$\vdots$$

$$P_1\{d = 40\} = \binom{40}{16}\frac{1}{2^{24}} \cdot \frac{1}{2^{16}} \doteq 0.057164.$$

Thus, the probability of reaching destination at any given number of hops d is

$$P_1\{d = k + 2i\} = \binom{k+2i}{i}\frac{1}{2^{k+2i}}, \;\; i = 0, 1, 2, \ldots \tag{4.2}$$

where k is the number of hops in the shortest path routing. $k = 8$ for routing packets from node 40 to 04 in our exampled WSN.

Summing up all above probabilities, we can obtain that the probability of successful transmission within H hops is

$$P_1\{d \leq H\} = \sum_{i=0}^{\frac{H-k}{2}} \binom{k+2i}{i}\frac{1}{2^{k+2i}}. \tag{4.3}$$

Therefore, within 32, 40 and 50 hops, the probabilities are calculated as follows:

$$P_1\{d \leq 32\} \doteq 0.43279925, \tag{4.4}$$

$$P_1\{d \leq 40\} \doteq 0.65550625, \tag{4.5}$$

$$P_1\{d \leq 50\} \doteq 0.9502225. \tag{4.6}$$

From above calculation, we can see that the probability of successful transmission increases as the number of hops increases. Random walk routing of Case 1 can guarantee near half packets to reach the destination with the same energy cost ($4k$) as the shortest path routing as Equation (4.4) shows. Within 50 hops which is less than twice energy cost of the shortest path routing, the random walk routing in Case 1 can provide more than 95% successful transmission.

Case 2: In Case 2, since the packet will not return to the neighbor where it comes from, the node randomly select one neighbor in the other three neighbors. Without taking the boundary of grid into account, we consider the probabilities of forwarding in correct direction and wrong direction as follows. If a packet comes from left or down direction (correct direction in last hop), the probability of forwarding the packet in correct (wrong) direction to next-hop neighbor is $\frac{2}{3}$ ($\frac{1}{3}$). If a packet comes from right or up direction (wrong direction in last hop), the probability of forwarding the packet in correct (wrong) to next-hop neighbor is $\frac{1}{3}$ ($\frac{2}{3}$). Then the average probability of forwarding a packet in correct (wrong) direction depends on the probability of occurrence of forwarding packets in correct (wrong) direction. For example, in a 30-hop path where the packet is forwarded 19 hops in correct direction and 11 hops in wrong direction, the average probability of forwarding a packet in correct direction can be estimated by $\frac{19}{30} \cdot \frac{2}{3} + \frac{11}{30} \cdot \frac{1}{3}$. Similarly, the average probability of forwarding a packet in wrong direction can be estimated by $\frac{19}{30} \cdot \frac{1}{3} + \frac{11}{30} \cdot \frac{2}{3}$. Thus, the probability of reaching destination with 30 hops for the WSN in Figure 4.9 can be estimated by

$$P_2\{d = 30\} = \binom{30}{11} \cdot (\frac{19}{30} \cdot \frac{2}{3} + \frac{11}{30} \cdot \frac{1}{3})^{19} \cdot (\frac{19}{30} \cdot \frac{1}{3} + \frac{11}{30} \cdot \frac{2}{3})^{11}.$$

Thus we estimate the probability of forwarding a packet with $k + 2i$ hops for Case 2 as follows:

$$P_2\{d = k + 2i\} = \binom{k+2i}{i} \cdot (\frac{k+i}{k+2i} \cdot \frac{2}{3} + \frac{i}{k+2i} \cdot \frac{1}{3})^{k+i} \cdot (\frac{k+i}{k+2i} \cdot \frac{1}{3} + \frac{i}{k+2i} \cdot \frac{2}{3})^{i}, \quad (4.7)$$

where k is the number of hops in the shortest path routing and $i \geq 0$ is an integer.

$$P_2\{d \leq H\} = \sum_{i=0}^{\frac{H-k}{2}} \binom{k+2i}{i} \cdot (\frac{k+i}{k+2i} \cdot \frac{2}{3} + \frac{i}{k+2i} \cdot \frac{1}{3})^{k+i} \cdot (\frac{k+i}{k+2i} \cdot \frac{1}{3} + \frac{i}{k+2i} \cdot \frac{2}{3})^{i}. \quad (4.8)$$

Comparing equations (6) and (1), we can easily prove $(\frac{k+i}{k+2i} \cdot \frac{2}{3} + \frac{i}{k+2i} \cdot \frac{1}{3})^{k+i} \cdot (\frac{k+i}{k+2i} \cdot \frac{1}{3} + \frac{i}{k+2i} \cdot \frac{2}{3})^{i} > \frac{1}{2^{k+2i}}$ by inductive method. Thus, it is clear that the probability that the packet reaches destination within H hops in Case 2 is greater than that in Case 1. For the exampled path in Figure 4.9, the probabilities of reaching destination within 8, 24 and 32 hops are calculated as follows.

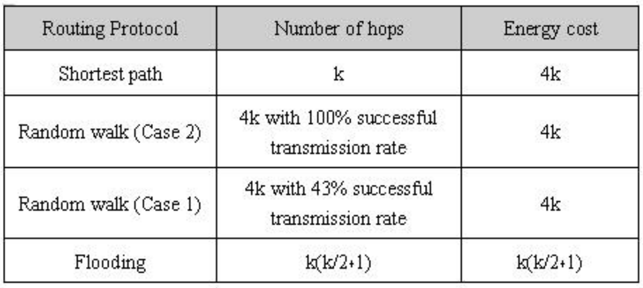
$$P_1\{d \leq 8\} \doteq 0.039018, \quad (4.9)$$

$$P_1\{d \leq 24\} \doteq 0.699459, \quad (4.10)$$

$$P_1\{d \leq 32\} \doteq 1.068195. \quad (4.11)$$

It is found that to guarantee around 65% successful transmission probability, the random walk routing in Case 2 set each packet to jump three times hops of the shortest path ($3k = 24$, Equation (4.10)). However, to guarantee the similar successful transmission probability, random walk routing in Case 1 requires each packet to jump five times hops of the shortest path ($5k = 40$, Equation (4.6)). For random walk routing of Case 2, 100% successful transmission rate can be guaranteed when setting the packet to jump four times hops of the shortest path as Equation (4.11) shows. Here the estimated successful transmission probability is greater than 100% because the difference of the probabilities of forwarding a packet in correct (wrong) direction in boundary and interior area is ignored. For a large-scale square-based WSN, the bias brought by the difference of boundary and interior area can be ignored.

As we have mentioned before, the shortest path routing requires $4k$ energy cost where k is the number of hops of the shortest path because each step it requires to communicate with all neighboring nodes to choose the next-hop node. If we use the routing with random walk of Case 2, the energy cost is also $4k$ because 100% successful transmission probability is provided within $4k$ hops. For small-size data WSNs applications, flooding is also a popular routing protocol in order to guarantee successful transmission. After calculation, we obtain that the flooding protocol requires $2g(g-1)$ energy cost in a WSN with grid size being $g \times g$, g is the number of sensor nodes in each edge. It is obvious that the flooding protocol consumes a huge amount of energy when some sensor node requires to report the event, especially when the WSN is in a large scale. However, the routing with random walk in Case 2 can guarantee successful transmission with the same energy cost as the shortest path routing. Figure 4.10 compares the energy cost of these routing protocols by routing messages on the longest path in the grid ($k = 2(g-1)$). Therefore, we can conclude that the routing with random walk achieves significant performance improvement in not only energy cost but also in load balancing which is not available for the shortest path routing.



Routing Protocol	Number of hops	Energy cost
Shortest path	k	4k
Random walk (Case 2)	4k with 100% successful transmission rate	4k
Random walk (Case 1)	4k with 43% successful transmission rate	4k
Flooding	k(k/2+1)	k(k/2+1)

Figure 4.10: Performance comparison on routing protocols

4.4.2 Density-Aware Topology Deployment

We have proved that the random walk routing can provide energy efficiency and load balancing in square-based WSNs. However, if the sink (or aggregation node) is fixed, the simple topology of grid cannot guarantee long lifetime for WSNs though the random walk routing has outperformed other protocols for load balancing. It is found the closer a sensor node is to the sink, the faster its energy is consumed. This is because more packets are routed by these nodes to reach the sink than by those nodes who are far from the sink. For any WSNs with fixed sinks, this is the inevitable problem of designing the topology of WSNs. Therefore, we propose a density-aware topology deployment scheme in this section which can solve this problem.

Density means the number of sensor nodes per unit area. The main idea of density-aware topology deployment is to place more sensor nodes at the area which is closer to the sink and fewer sensor nodes at the area which is farther to the sink. Assume the size of grid is $N = g \times g$. The distance of each neighboring nodes pair is 1 unit. There are $N \cdot \alpha$ messages sensed by the sensor nodes during the period from the initiation to the time the WSN ends its task. Here α is an estimation factor for the number of messages the WSN will generated in total. We suppose the abnormal events exist uniformly in the whole area of the WSN. Suppose the energy consumption of each message required to be sent is e_m and the battery life of each sensor node is E_n. The distance from a sensor node to the sink is denoted by the number of hops d. Assume the sink is located at any edge corner of the grid, then the density, denoted by σ, at any point in grid should be set as follows:

$$\sigma_{grid} = \begin{cases} \lceil \frac{N\alpha \cdot e_m}{(1+d) \cdot E_n} \rceil & d \leq g \\ \lceil \frac{N\alpha \cdot e_m}{(2g-d-1) \cdot E_n} \rceil & d > g \end{cases},$$

where

$$\alpha \propto \begin{cases} \frac{(g-1)^2 - \frac{1}{2}d^2}{(g-1)^2} & d \leq g \\ \frac{\frac{1}{2}(2(g-1)-d)^2}{(g-1)^2} & d > g \end{cases}.$$

Here the number of sensor nodes which should be placed at each cross point in the grid is given by σ_{Grid}. It is estimated according to the energy requirement at each place. Taking ceiling is for satisfying the estimated maximal energy requirement by least number of sensor nodes. Since the number of messages generated is assumed to be proportional to the area as above, we set α to be proportional to the ratio of message-generated area to the whole area. Here the message-generated area is defined as the area covered by the sensor node whose distances are greater than d. The message generated in the message-generated area must be routed to the sink via the nodes with the distance d to the sink.

Figure 4.11 gives an example for density-aware deployment in grid where the sink is node 04 and $e_m = 1J$, $E_n = 10J$.

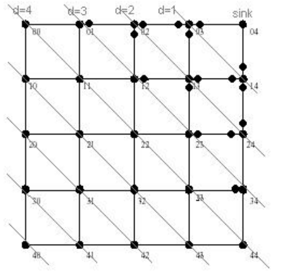


Figure 4.11: Density-aware deployment for a WSN with square-based topology

We can see that, in order to guarantee that nodes with heavier load distribution task do not affect the networking lifetime after exhaustion, some supplementary sensor nodes at the same location must be waken up to maintain the active WSN. In this way, the WSN can provide longer networking lifetime. In Figure 4.11, the nearer a node is to the sink, the more supplementary sensor nodes are provided.

If the sink is located at the center of the grid, the density σ is a little bit different from the above formula.

$$\sigma_{grid} = \lceil \frac{N\alpha \cdot e_m}{4d \cdot E_n} \rceil. \tag{4.12}$$

Similarly, the density formula can be extended to triangle-based WSNs, hexagon-based WSNs and WSNs with general topologies. Assume the sink is located at the center of the WSN.

$$\sigma_{tri} = \lceil \frac{N\alpha \cdot e_m}{6d \cdot E_n} \rceil \tag{4.13}$$

$$\sigma_{hex} = \lceil \frac{N\alpha \cdot e_m}{3d \cdot E_n} \rceil \tag{4.14}$$

$$\sigma_{gen} \propto \lceil \frac{N\alpha \cdot e_m}{d \cdot E_n} \rceil \tag{4.15}$$

The above formulas show how to deploy the sensor nodes effectively and efficiently to provide longer lifetime for WSNs. The deployment density depends on the distance from the sink. For triangle-based WSNs, there are $6d$ nodes who take the task of routing messages generated in the area covered by sensor nodes whose distances are greater than d. For hexagon-based WSNs, there are $3d$ nodes who share to route messages generated in the area covered by sensor nodes whose distances are greater than d. Similar to patterned topologies, for WSNs with general topologies, a general formula can be deduced as Equation (4.15). By employing these formulas, we can see that if the deployment of a sensor node can be made density-aware, the performance of the WSN will be improved greatly.

Though this kind of pre-deployment may not be applicable in many scenarios, the formulas can still be used to predict and evaluate the network performance. Also they can guide to develop density-aware deployment protocols (or topology control scheme) in dynamic environment.

4.5 Concluding Remarks

This chapter has addressed how to deploy WSNs in patterned topologies which can provide the required coverage area and the connectivity of all sensor nodes. We have discussed different patterned topologies for WSNs by theoretically analyzing their coverage area and comparing the characteristics of WSNs in these topologies. It has been found that strip-based topology provides the maximal connected coverage and consumes the least energy by flooding protocol, whereas triangle-based topology reaches the best performance in reliability and sensing strength. Square-based topology provides the simplest architecture and a moderate performance which lies in their midst.

In WSNs with patterned topology, existing routing methods mainly adopt shortest path routing with the knowledge of global location information. We have proposed several routing protocols that require only local information and work in a simple and effective way to achieve to achieve different performance goals. Our simulation results have shown that the networking lifetime is maximized by selecting route based on both the length of route (hop count) and the number of streams (stream unit) of the next-hop neighbor. Therefore, patterned WSNs equipped with these routing protocols provide great promises and guarantee their potential applications to meet different needs. We have observed through analysis and simulation that our routing protocols can achieve desired performance goals by suitably taking into account of local parameters (hop count and stream unit) in route selection. This interesting property will also hold if our routing protocols are used in WSNs with random-deployed topology.

To achieve a better energy efficiency, particularly for small-size data transmissions, we have proposed another protocol of applying random walk routing. This routing protocol does not require any location information, neither the exchange information between neighboring nodes. Moreover, it has an inherent property of load balancing which is difficult to achieve by other routing protocols in WSNs. It has also been proved that random walk routing consumes the same amount of energy as the shortest path routing when the message to be sent to the base station is in comparatively small size to the inquiry message between neighboring nodes. This is a usual case for sensor networks adopted in emergency treatment scenarios where sensor nodes send beep-like short messages to the base station to report their status. Based on the regular architectures and the random walk routing, we further proposed a density-aware deployment scheme to guarantee a longer networking lifetime than other routing schemes for WSNs. The density-aware deployment scheme avoids the impact of early energy exhaustion at heavy-load nodes on the whole network's lifetime. This kind of deployment can also be extended to general WSNs and thus provides an alternative method for prolongation of networking lifetime.

Chapter 5

Conclusion

The information of network topology is vital for routing, resource scheduling, flow control and network management. This thesis presents new methods and techniques for network topology discovery and its applications which were developed during my PhD research in JAIST. Our main contributions can be summarized as follows:

Firstly, we studied network tomography for multicast network topology discovery. We proposed two new topology inference algorithms which took the level information and hamming distance of sequences on receipt/loss of probe packets maintained at each pair of nodes into account. Both algorithms show impressively better performance than previous inference algorithms based on the well-known A-approach in inference accuracy and efficiency. Based on the discovered topology information, we further developed network internal loss performance inference schemes which simplified the inference procedure and improved efficiency of the previous methods. The hamming distance matrix-based loss/delay performance inference approach was also proposed as a generalized method for network internal performance inference.

Secondly, we developed a novel method of deploying mobile agents for network topology discovery as a new application of mobile agents technology. Due to the inherent advantages of mobile agents, mobile agent-based topology discovery algorithms have shown potential efficiency and effectiveness. Two proposed algorithms, report-at-newly-found-nodes (RN) and report-at-leaf-nodes (RL), exhibit own advantages and disadvantages for different types of networks including Internet and multicast networks. This shows that they can be effectively applied according to practical system requirements. However, as a general restriction to mobile agents technology, the assumption that all entities must support the execution of mobile agent codes may limit the application of mobile agent-based topology discovery algorithms in practice.

Thirdly, we studied topology deployment for wireless sensor networks (WSNs) and its application in routing. We analyzed four types of patterned topologies in details, and compared the performance of WSNs in different topology patterns. It has been shown that energy-efficiency, reliability, and maximal connected coverage can be provided by different strategies of topology deployment for WSNs. Based on patterned topologies, we proposed two types of routing protocols for various scenarios. Both of them require only local information, and work in a simple and effective way to achieve different performance goals. The first protocol provides a generalized method for selecting the next-hop neighbor by different functions on local parameters to achieve desired energy efficiency in total energy saving, battery life at individual nodes, and networking lifetime respectively. Choice of route-selection functions is determined by the requirements for particular applications. The second protocol uses the random walk technique and applies mainly to WSNs in which the size of data in transmission is small. We showed through quantitative analysis that random walk is a quite effective technique to achieve energy-efficiency and load balancing.

Network topology discovery is a fast growing research area. Many new challenging problems in this area will catch our attention along with the emergence of new types of networks and applications. Below are some tasks for our future research, as extensions of the work reported in this thesis:

- Hamming distance of sequences on receipt/loss of probe packets maintained at each pair of nodes was proposed for the first time in this thesis for multicast-based tomography. It has been observed to have more potential applications in detailed link delay performance inference which will be studied further in the future.

- As the prevalent applications of mobile agents technology, we intend to validate our proposed algorithms in a real mobile agent-based system and improve their performance. We will also develop new mobile agent-based network topology discovery models and methods to incorporate new network properties and application requirements.

- As wireless sensor networks are mainly application-driven, different applications may have different requirements for energy-efficiency and reliability. To extend our work on this line, we will concentrate on particular types of applications and develop new methods and techniques to meet different requirements for energy-efficiency and reliability.

- Applying network topology discovery techniques to achieve better network security has recently attracted an increasing attention. We plan to extend our developed network topology inference methods and techniques for detection of potential intrusions and security holes by comparing a network's topology and regional traffic status in the secured (normal) state with the discovered ones. By analyzing a network's topology, we can discover these weak components/areas in the network and thus take necessary means to protect them from being attacked.

- Semantic web is an emerging research topic which has shown great promises in various applications of Internet and web technology. We plan to explore the possibility of applying network topology discovery methods and techniques for semantic web topology discovery. Such topologies will reveal the semantic relations among web access patterns and thus provide valuable information for different kinds of web information analysis to improve the quality of service.

Bibliography

[1] An atlas of cyberspace. http://www.geog.ucl.ac.jk/casa/martin/atlas/atlas.html, 1998.

[2] Caida mapnet viewer. http://www.caida.org/Tools/Mapnet/, 1998.

[3] Skitter. http://www.caida.org/tools/measurement/skitter, 1998.

[4] Otter. www.caida.org/tools/visualization/otter/, 1999.

[5] I.F. Akyildiz, Y. Sankarasubramaniam W. Su, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, 2002.

[6] Mikhail J. Atallah, editor. *Algorithms and Theory of Computation Handbook*. CRC Press LLC, 1999.

[7] M. Baldi, S. Gai, and G.P. Picco. Exploiting code mobility in decentralized and flexible network management. In *Proc. of the 1997 1st International Workshop on Mobile Agents*, 1997.

[8] B. Bellur and R. G. Ogier. A reliable, efficient topology broadcast protocol for dynamic networks. In *Proc. of IEEE INFOCOM*, 1999.

[9] A. Bestavros, K. Harfoush, and J. Byers. Robust identification of shared losses using end-to-end unicast probes. In *Proc. of IEEE Internation Conference on Netowrk Protocols*, Osaka, Japan, Nov. 2000.

[10] Edoardo S. Biagioni and Galen Sasaki. Wireless sensor placement for reliable and efficient data collection. In *Proc. of the 36th Hawaii International Conference on System Sciences (HICSS)*, 2003.

[11] A. Bieszczad, B. Pagurek, and T. White. Java-based intelligent mobile agents for open system management. In *IEEE the 9th IEEE International Conference on Tools with Artificial Intelligentce*, pages 492–501, 1997.

[12] A. Bieszczad, B. Pagurek, and T. White. Mobile agents for network management. *IEEE Communications Surveys*, 1:2–9, 1998.

[13] D. Braginsky and D. Estrin. Roumor routing algorithm for sensor networks. In *Proc. of the First Workshop on Sensor Networks and Applications (WSNA)*, 2002.

[14] Y. Breitbart, M. Garofalakis, Ben Jai, C. Martin, R. Rastogi, and A. Silberschatz. Topology discovery in heterogeneous ip networks: The netinventory system. *IEEE/ACM Transactions on Networking*, 12:401–414, 2004.

[15] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz. Topology discovery in heterogeneous ip networks. In *Proc. of IEEE INFOCOM*, pages 265–274, 2000.

[16] R. Caceres, N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley. Loss based inference of multicast network topology. In *Proc. of IEEE Conference on Decision and Control*, Phoenix, AZ, 1999.

[17] R. Caceres, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Trans. on Information Theory*, 45(7):2462–2480, 1999.

[18] M. Cardei and J. Wu. Coverage in wireless sensor networks. Handbook of Sensor Networks, 2004.

[19] Rui Castro, Mark Coates, and Robert Nowak. Maximum likelihood identification from end-to-end measurements. In *Proc. of DIMACS Workshop on Internet Measurement, Mapping and Modeling*, DIMACS Center, Rutgers University, Piscataway, New Jersey, February 2002.

[20] Alberto Cerpa and Deborah Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *Proc. of INFOCOM*, 2002.

[21] Ranveer Chandra, Christof Fetzer, and Karin Hogstedt. Adaptive topology discovery in hybrid wireless networks. *Informatics*, 2002.

[22] W. Chen, N. Jain, and S. Singh. Anmp: Ad hoc network management protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1506–1531, 1999.

[23] Romit Roy Choudhury, Somprakash Bandyopadhyay, and Krishna Paul. Topology discovery in ad hoc wireless networks using mobile agents. In *Proceedings of the Second International Workshop on Mobile Agents for Telecommunication Applications*, pages 1–16, 2000.

[24] M. Coates and R. Nowak. Network loss inference using unicast end-to-end measurement. In *ITC Seminar on IP Traffic: Measurement and Modelling*, Monterey, CA, 2000.

[25] M. J. Coates, A. O. Hero, R. Nowak, and B. Yu. Internet tomography. *IEEE Signal Processing Magazing*, 19(3):47–65, May 2002.

[26] Mark Coates, Rui Castro, and Robert Nowak. Maximum likelihood network topology identification from edge-based unicast measurements. In *ACM sigmetrics*, 2002.

[27] Mark Coates, Rui Castro, Robert Nowak, Manik Gadhiok, Ryan King, and Yolanda Tsang. Maximum likelihood network topology identification from edge-based unicast measurements. In *Proc. of SIGMETRICS*, Marina Del Rey, California, 2002.

[28] N. Dawes, D. Schenkel, and M. Slavitch. Method of determining the topology of a network objects. U.S. Patent 6411997, 25 June 2002.

[29] B. Deb, S. Bhatangar, and B. Nath. A topology discovery algorithm for sensor networks with applications to network management. In *Proc. of IEEE CAS Workshop on Wireless Communications and Networking*, 2002.

[30] N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley. Multicast topology inference from measured end-to-end loss. *IEEE Trans. on Information Theory*, 48(1):26–45, 2002.

[31] N. G. Duffield and F. Fo Presti. Multicast inference of packet delay variance at interior network links. In *Proc. of IEEE INFOCOM*, pages 1351–1360, Tel Aviv, Israel, 2000.

[32] N. G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. In *Proc. of IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[33] N.G. Duffield, J. Horowitz, and F. Lo Presti. Adaptive multicast topology inference. In *Proc. of IEEE INFOCOM*, pages 1636–1645, Anchorage, Alaska USA, April 2001.

[34] N.G. Duffield, J. Horowitz, F.Lo Presti, and D. Towsley. Multicast topology inference from end-to-end measurements. ITC Seminar on IP Traffic Measurement, Modeling and Management, Monterey, CA, Sept. 2000.

[35] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proc. of ACM MobiCom*, 1999.

[36] R. Govindan and H. Tangmunarunkit. Heuristics for internet map discovery. In *Proc. of IEEE INFOCOM*, pages 1371–1380, 2000.

[37] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proc. of 5th ACM/IEEE MobiCom*, pages 174–185, 1999.

[38] Chi-Fu Huang and Yu-Chee Tseng. The coverage problem in a wireless sensor network. In *Proc. of ACM Int'l Workshop on Wireless Sensor Networks and Applications(WSNA)*, 2003.

[39] Rajagopal Iyengar, Koushik Kar, and Suman Banerjee. Low-coordination topologies for redundancy in sensor networks. In *Proc. of Mobihoc*, 2005.

[40] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot, and T. Clausen. Optimized link state routing protocol. Technical report, www.ietf.org/rfc/rfc3626.txt, 2001.

[41] Koushik Kar and Suman Banerjee. Node placement for connected coverage in sensor networks. In *Proc. of WiOpt*, 2003.

[42] S-H. Kim and T.G. Robertazzi. Mobile agent modeling. Technical report, University at Stony Brook, College of Engineering and Applied Science, 2000.

[43] H. Ku, G. W. R. Luderer, and B. Subbiah. An intelligent mobile agent framework for distributed network management. In *Proc. IEEE GLOBECOM'97*, pages 160–164, 1997.

[44] Martin Kubisch, Holger Karl, Adam Wolisz, and Lizhi Charlie Zhong andJan Rabaey. Distributed algorithms for transmission power control in wireless sensor networks. In *Proc. of IEEE Wireless Communicationsand Networking Conference (WCNC)*, 2003.

[45] J. Kulik, W. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks*, 8:169–185, 2002.

[46] Jangwon Lee and Gustavo de Veciana. Resource and topology discovery for ip multicast using a fan-out decrement mechanism. In *Proc. of IEEE INFOCOM*, pages 1627–1635, Anchorage, Alaska USA, 2001.

[47] Xiang-Yang Li, Peng-Jun Wan, and Ophir Frieder. Coverage in wireless ad-hoc sensor networks. In *Proc. of IEEE ICC*, 2002.

[48] Hwa-Chun Lin and Chien-Hsing Wang. Automatic topology discovery using mobile agents. In *The International Workshop on Agent Technologies over Internet Applications*, 2001.

[49] Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, and Mani B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proc. of IEEE INFOCOM*, 2001.

[50] N. Minar, K. H. Kramer, and P. Maes. Cooperating mobile agents for mapping networks. In *Proc. of the 1st Hungarian National Conference on Agent Based Computing*, 1998.

[51] Jianping Pan, Y. Thomas Hou, Lin Cai, Yi Shi, and Sherman X. Shen. Topology control for wireless sensor networks. In *Proc. of MobiCom*, 2003.

[52] Vu Anh Pham and Ahmed Karmouch. Mobile software agents: An overview. *IEEE Communications Magazine*, pages 26–37, 1998.

[53] A. Sahai. Toward distributed and dynamic network management. In *Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS'98)*, pages 455–464, 1998.

[54] Ayad Salhieh, Jennifer Weinmann, Manish Kochhal, and Loren Schwiebert. Power efficient topologies for wireless sensor networks. In *Proc. of Int'l Conf. on Parrallel Processing*, 2001.

[55] D. Schenkel, M. Slavitch, and N. Dawes. Method of determining topology of a network of objects which compares the similarity of the traffic sequences/volumes of a pair of devices. U.S. Patent 2926462, 20 July 1999.

[56] C. Schramm, A. Bieszczad, and B. Pagurek. Application-oriented network modeling with mobile agents. In *Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS'98)*, 1998.

[57] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani B. Srivastava. Topology management for sensor networks: Exploiting latency and density. In *Proc. of MobiCOM*, 2001.

[58] S. Servetto and G. Barrenechea. Constrained random walks on random graphs; routing algorithms for large scale wireless sensor networks. In *Proc. of 1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[59] G. Shao, F. Berman, and R. Wolski. Using effective network views to promote distributed application perfoormance. In *Proc. of Int. conf. Parallel and Distributed Processing Techniques and Applications*, pages 2649–2656, 1999.

[60] R. Siamwalla, R. Sharma, and S. Keshav. Discovering internet topology. Technical report, Cornell University, 1999.

[61] Di Tian and Nicolas D. Georgannas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *Proc. of ACM Int'l Workshop on Wireless Sensor Networks and Applications(WSNA)*, 2002.

[62] Hui Tian and Changxing Pei. An improved algorithm of network topology discovery. *Radio Engineering of China*, 2002.

[63] Hui Tian and Hong Shen. Random walk routing for wireless sensor network. In *The Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'04)*, Dalian, China.

[64] Hui Tian and Hong Shen. An optimal coverage scheme for wireless sensor networks. In *Proc. of International Conference on Networking*, pages 722–730, 2005.

[65] Hui Tian and Hong Shen. An improved algorithm of multicast topology inference from end-to-end measurements. *International Journal of Communication Systems*, 2005 (accepted). Preliminary version partially in Proc. of the 5th Int'l Symp. on High Performance Computing ISHPC-V, Tokyo, Oct. 2003, 376-384.

[66] Hui Tian and Hong Shen. Analysis on binary loss tree classification with hop count for multicast topology discovery. In *Proc. of IEEE CCNC'04*, Las Vegas, USA, Jan. 2004.

[67] Hui Tian and Hong Shen. Hamming distance and hop count based multicast network topology inference. In *Proc. of IEEE AINA'2005*, pages 267–272, Taiwan, China, March, 2005.

[68] Hui Tian and Hong Shen. Mobile agents based topology discovery algorithms and modelling. In *Proceedings of ISPAN'04*, pages 502–507, Hongkong, China, May. 2004.

[69] Hui Tian, Hong Shen, and Teruo Matsuzawa. Developing energy-efficient topologies and routing for wireless sensor networks. In *Proc. of IFIP International Conference on Network and Parallel Computing*, Dec. 2005.

[70] K. Tutschku and H. Baier. Characterizing network performance for enterprise networks. In *Proc. of Passive and Active Measurements (PAM)*, 2001.

[71] San Diego University of Califoria. Wireless topology discovery. http://ramp.ucsd.edu/wtd/wtd.html.

[72] Y. Vardi. Network tomography: estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association*, 91:365–377, 1996.

[73] T. White, B. Pagurek, A. Bieszczad, G. Sugar, and X. Tran. Intelligent network modeling using mobile agents. In *Proc. of the IEEE GLOBECOM'98)*, 1998.

[74] Beau Williamson. *Developing IP Multicast Networks.* Cisco Press, 2001.

[75] M. Zapf, K. Herrmann, and K. Geihs. Decentralized snmp management with mobile agents. In *Proc. of the IEEE /IFIP International Symposium on Integrated Network Management (IM'99)*, 1999.

Publications

1. Hui Tian and Hong Shen, “Multicast Based Inference for Topology and Network-Internal Loss Performance from End-to-end Measurements”, accepted by *Computer Communications*, Elsevier, Dec. 2005.
2. Hui Tian and Hong Shen, “An improved algorithm of multicast topology inference from end-to-end measurements”, accepted by *International Journal of Communication Systems*, John Wiley & Sons, Oct. 2005.
3. Hui Tian, Hong Shen and Teruo Matsuzawa, “Random Walk Routing for Wireless Sensor Networks”, *International Journal of Computer Science and Network Security*, accepted, 2005.
4. Hui Tian, Hong Shen and Teruo Matsuzawa, “Energy-Efficient Topologies and Routing for Wireless Sensor Networks”, *GESTS International Transaction on Computer Science and Engineering*, No.1, Vol.8, pp. 79-89, May 2005.
5. Hui Tian and Hong Shen, “Multicast Topology Inference and Its Applications”, to appear in *Handbook of Approximation Algorithms and Metaheuristics*, Taylor & Francis Books (CRC Press), 2006.
6. Hui Tian, Hong Shen and Teruo Matsuzawa, “Developing Energy-Efficient Topology and Routing in Wireless Sensor Network”, *Lecture Notes of Computer Science* (Proc. of 2005 IFIP Int. Conf. on Network and Parallel Computing, Beijing, Dec. 2005), Springer-Verlag.
7. Hui Tian, Hong Shen and Teruo Matsuzawa, “Random Walk Routing in Wireless Sensor Networks”, *Proc. Of 2005 International Conference on Parallel and Distributed Computing, Applications and Technologies* (PDCAT’05), Dalian, China, Dec. 2005, IEEE press.
8. Hui Tian and Hong Shen, “Discover multicast network internal characteristics based on hamming distance, *Proc. of 2005 IEEE International Conference on Communications* (ICC’05), Seoul, Korea, May 2005, CD-ROM, IEEE press.

9. Hui Tian and Hong Shen, “An optimal coverage scheme for wireless sensor network”, *Lecture Notes of Computer Science*, Vol. 3420 (Proc. of 2005 International Conference on Networks (ICN’05), Reunion Island, France, April 2005), pp. 722-730, Springer-Verlag.

10. Hui Tian and Hong Shen, “Hamming distance and hop count based classification for multicast network topology inference”, Proc. of *Proc. of IEEE 19th International Conference on Advanced Information Networking and Applications* (AINA’05), Taiwan, China, pp. 267-272, March 2005, IEEE press.

11. Hui Tian and Hong Shen, “Lossy link identification for multicast network”, *Lecture Notes of Computer Science*, Vol. 3320 (Proc. of The Fifth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT’04), Singapore, Dec. 2004), pp.416-419, Springer-Verlag.

12. Hui Tian and Hong Shen, “Multicast-based inference of network-internal loss performance”, *Proc. of 2004 International Symposium on Parallel Architectures, Algorithms and Networks* (I-SPAN’04), Hongkong, China, pp.288-293, May 2004, IEEE press.

13. Hui Tian and Hong Shen, “Mobile agents based topology discovery algorithms and modelling”, *Proc. of 2004 International Symposium on Parallel Architectures, Algorithms and Networks* (I-SPAN’04), Hong Kong, China, pp.502-507, May 2004, IEEE press.

14. Hui Tian and Hong Shen, “Analysis on binary loss tree classification with hop count for multicast topology discovery”, *Proc. of 2004 IEEE Consumer Communications and Networking Conference* (CCNC’04), Las Vegas, USA, Jan. 2004, CD-ROM, IEEE press.

15. Hui Tian and Hong Shen, “An improved algorithm of multicast topology inference from end-to-end measurements”, *Lecture Notes of Computer Science*, Vol. 2858 (Proc. of The Fifth International Symposium on High Performance Computing (ISHPC-V), Tokyo, Japan, Oct. 2003), pp.376-384, Springer-Verlag.