Modeling Key Agreement in Multi-Hop Ad Hoc Networks *

Giovanni Di Crescenzo
Telcordia Technologies
One Telcordia Dr,
Piscataway, NJ, 08854
(001)7326992108
giovanni@research.telcordia.com

Maria Striki
The Institute for Systems & Research
University of Maryland, College Park
College Park, MD, 20742
(001)3014056558
mstriki@glue.umd.edu

John S. Baras
The Institute for Systems & Research
University of Maryland, College Park,
College Park, MD, 20742
(001)3014056606
baras@isr.umd.edu

ABSTRACT
Securing multicast communications in ad hoc networks has become one of the most challenging research directions in the areas of wireless networking and security. This is especially true as ad hoc networks are emerging as the desired environment for an increasing number of civilian, commercial and military applications, also addressing an increasingly large number of users. In this paper we study a very basic security question for Ad Hoc Networks: Key Agreement against passive adversaries. Despite being a widely studied area in wired networks, the problem becomes significantly more challenging for ad hoc networks, and even more for sensor networks, due to lack of trusted entities, infrastructures, full connectivity, routing structures, and due to severe limitations on the resources and capabilities of network nodes. In this paper we perform a comprehensive investigation of Key Agreement over resource constrained ad hoc networks. First, we formally model the key agreement problem over multi-hop ad hop networks, and we directly extend known key agreement protocols for wired networks, and evaluate the efficiency of such approaches. We then go beyond natural extensions of such protocols, by proposing non-trivial extensions based on efficient topology-driven simulations of logical networks over an arbitrary physical network, in order to optimize the most significant metrics of interest for such networks: i.e. bandwidth, latency, processing cost. Indeed, the resulting protocols are significantly more efficient in some or all of the above metrics, as our analytical results indicate.

Categories and Subject Descriptors
C.2 [**Computer-Communication Networks**]: C.2.1 Network Architecture and Design: *Network Communications, Network Topology*, C.2.2 Network Protocols: *Routing Protocols*, C.4 Performance of Systems: Modeling Techniques, Performance Attributes, G.2 [**Discrete Mathematics**]: G.2.2 Graph Theory: *Graph Algorithms, Network Problems, Trees*.

General Terms
Algorithms, Performance, Design, Security.

IWCMC'06, July 3-6, 2006, Vancouver, British Columbia, Canada.

Keywords
Group Key Agreement, Topology Driven Protocols, Performance Evaluation, Optimization, Approximation Algorithms, Efficiency

1. INTRODUCTION
A basic security area in building secure protocols for group communication is that of group key agreement (GKA). Indeed, this area has received a significant amount of attention in the wired network literature, and since the foundational Diffie-Hellman (DH) protocol [2], several other protocols have been proposed for the group case. The lack of infrastructure and, frequently, of trusted entities in ad hoc networks further increases the importance of KA over such networks. In fact, a secure KA protocol for ad hoc networks can be used as the crucial component to bootstrap security in most, if not all, applications over ad hoc networks that require any type of security property. Previous work on KA over ad hoc networks does *not* deal with what we believe to be basic features of any realistic protocol implementation over the discussed environment: partial connectivity due to limited radio range, and parties' self-processing of routing duties due to lack of trusted authorities. As a result, the analysis of previously proposed protocols only gives a partial account on their real performance. Also, existing performance evaluation of the protocols has been done only with respect to a logical abstraction of the underlying physical network, without consideration of the underlying routing. While routing is often assumed to be efficient and available between any two parties in wired networks, this is not the case for ad hoc networks. Therefore, in evaluating the efficiency of direct adaptations of protocols for wireless networks, some protocols lose even significant efficiency properties. On the other hand, it might be the case that KA solutions naturally take advantage of the unique ad hoc network features, such as topology-driven redundancy and localization of communications and computations, possibly resulting in more efficient protocols.

A thorough understanding of these basic features is essential towards the design of optimal or at least more suitable KA schemes for the environment of study, and in this paper, we propose methodologies towards that. We propose non-trivial extensions of known KA protocols over logical networks based on efficient simulation of

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

logical networks over arbitrary physical networks. This defines, for each known KA protocol, a **network simulation problem** similar to the simulation (or, embedding) of logical networks over physical networks of parallel computers, where the goal is to minimize some metric inherent to the performance of the specific protocol itself. Our contribution in this paper is to: *a*) formulate the corresponding network simulation problem, *b*) define a number of optimization problems related to the KA execution over the network, and *c*) present our own solutions to these problems.

2. MODEL

We model the connectivity among the parties with a connectivity graph $G(V,E)$, where $E \in V \times V$ and an edge between any two nodes exists if and only if the two associated parties are within each other's radio range (bidirectional connectivity considered). We make the simplifying assumptions that each message from a node in G is timely sent (i.e., there is no congestion) and reliably and timely received by all neighbors. Additionally, in this current analysis we do *not* consider link failure or party mobility. We stress that we only deal with KA over an ad hoc network in the passive adversary model. Extending our approaches to deal with party mobility or more involved adversaries (i.e., active or Byzantine), is the object of our future research. Similarly, the re-keying problem is out of the scope of this paper. Our schemes inherit the same security properties as those of their original ancestors. In addition to designing key agreement protocols over arbitrary ad hoc networks satisfying correctness and security requirements, we want to meet efficiency requirements of low *bandwidth* (defined as the maximum, over all possible graphs G and executions of protocol instances P over G, of the total size of messages exchanged by all parties in $V(G)$), *computation* (maximum, over all possible graphs G and executions of P over G, of the number of exponentiations performed by all parties in V(G)), and *latency* (maximum, over all possible graphs G and executions of P over G, of the number of parallel protocol atomic steps, such as communication rounds or 1-hop transfers).

3. TOPOLOGY BLIND GKA OVER AHNets

3.1 Overview of Logical GKA Schemes

The most immediate way to construct GKA protocols over AHNets is a topology-blind approach, denoted as "tb", where known GKA protocols over logical wired networks are composed with an underlying arbitrary routing protocol that implements on the physical AHNet G any single step done on the logical wired networks. In this paper we describe this approach for some among the most efficient, to our knowledge, GKA protocols over wired networks currently: GDH.1, GDH.2 [1], ING [4], and BD [3], referred all in [1]. We briefly recall them for the sake of self-containment.

3.1.1 GDH1: This protocol assumes that all parties are connected according to a logical Hamiltonian path and consists of two stages: up-flow (collecting members' contributions) and down-flow (allowing all members to compute the common key). In the up-flow, each member does 1 exponentiation and the message between M_i and M_{i+1} contains i intermediate values. After obtaining K_n, M_n initiates the down-flow stage. Each member M_i does i exponentiations: 1 to compute K_n and (i-1) to provide intermediate values to lower indexed members, by raising them to the power of its own exponent. The size of the down-flow message decreases on each link, as a message between M_{i+1} and M_i includes i intermediate values.

3.1.2 GDH2: In order to reduce the total number of rounds, GDH.1 is slightly varied, so that: *a*) in the up-flow stage each member has to compose i intermediate values (each with i-1 exponents) and one cardinal value with i exponents; M_n is the first member to compute the key K_n and the last batch of intermediate values, *b*) in the down-flow stage M_n broadcasts the (n-1) intermediate values to all group members. It is assumed that all parties are connected through a logical Hamiltonian path, and the last party on the path can reach all others using a broadcast channel.

3.1.3 BD: This scheme only requires 2 logical rounds and can be divided into three phases: (*1*) Member M_i generates random N_i and broadcasts $z_i = \alpha^{N_i}$; (*2*) Every member M computes and broadcasts $X_i = \left(z_{i+1} / z_{i-1}\right)^{N_i}$; (*3*) M_i can compute the group key $K_n = z_{i-1}^{nN_i} X_i^{n-1} X_{i+1}^{n-2} \ldots X_{i-2} \bmod p = \alpha^{N_1N_2+N_2N_3+\ldots+N_nN_1}$. The key defined by this scheme is different from the previous protocols. It is assumed that parties can simultaneously reach all others through broadcast channels.

3.1.4 ING: It requires that all parties are connected according to a logical ring, and completes in (n-1) rounds after a synchronous start-up. In any round, every party raises the previously-received intermediate key value to the power of its own random exponent and forwards the result to the next party. After (n-1) rounds everyone computes the same key.

3.2 Topology Blind Implementation over Arbitrary AHNets

The referred protocols describe the KA algorithms and the resulting overhead from key exchanges only. If we consider however executing the latter protocols in a resource constrained multi-hop network, then every message exchange among the intended group members may involve multiple relays that carry the intended key messages. The overhead invoked by these relays is not considered in the previous analyses. Also, in certain cases, it is assumed that a member can directly reach the rest of the group members and broadcast to them in one round, which is unrealistic in AHNets. In the case that the selection of the parties is not "topology-aware", (e.g. based on members' *ID*s), the placement of members and consequently the routes formed are random and the physical graph generated is actually not expected to resemble the logical graph and therefore is not optimal. This arbitrary factor that emerges when we merge the key generation algorithm "blindly" with the underlying routing is what we try to capture and model. We quantify it in the analytical results that follow as we merge and measure the overhead of both communication and routing exchanges for the aforementioned protocols. We denote as D the diameter $diam(G)$ of G; that is, the max number of hops between a pair of nodes in V. Moreover, we denote the number of hops in the path between two graph nodes N_i and N_{i+1}, as Route (N_i, N_{i+1}) = $R_{i,i+1}$. Throughout the analysis, we will use the following two facts: first, for each *i,* it holds that $R_{i,i+1} \leq D \leq |V(G)|=n$; second, a logical broadcast step requires at most $|E(G)|$ messages (assuming a simple controlled flooding strategy) and at most $D \leq |V(G)|=n$ hop transfers over the network graph G. We denote by K the length of an element in the algebraic group used (over which the decision DH problem is assumed to be hard). By *Bdw* we denote the total bandwidth, and by *Lt* the total latency encountered to the network by any KA protocol.

We discuss the computation of metric performances for all protocols and summarize them in a table.

3.2.1 GDH.1: We compute the efficiency metrics for this protocol by summing the metrics obtained for the up-flow and down-flow stages; since the two stages are symmetric, it is enough to describe the computation for one stage only, and then multiply both metrics by 2.

Up-Flow: On a logical Hamiltonian path, member M_i composes i values to send to member M_{i+1} through all relays in $R_{i,i+1}$.All relays in $R_{i,i+1}$ will carry the same message ($i \times K$ bit size). Hence:

$$Lt = \sum_{i=1}^{n-1} R_{i,i+1} \leq D \times (n-1), \qquad Bdw(M_i) = R_{i,i+1} \times i \times K,$$

$$Bdw = \sum_{i=1}^{n-1} R_{i,i+1} \times i \times K \leq \sum_{i=1}^{n-1} D \times i \times K = D \times K \times \tfrac{1}{2} n \times (n-1),$$

3.2.2 GDH.2: We compute the efficiency metrics by summing those obtained for the up-flow and down-flow stages.

Up-Flow: The analysis for the bandwidth and latency is identical to the up-flow stage of GDH.1.

Down-Flow: Member M_n composes (n-1) values to broadcast to all members M_i. We denote this **broadcast cost** as $B_n(n\text{-}1)$. Assuming that in a realistic implementation each node broadcasts the same message only once to its neighbors, we could upper bound the **broadcast cost** as follows: $B_n(n\text{-}1) \leq |E(G)| \times (n\text{-}1) \times K$. Under the assumption of omni-directional antennas, allowing each node to send messages to all of its neighbors, we can bound $B_n(n\text{-}1)$ by : $B_n(n\text{-}1) \leq D \times (n\text{-}1) \times K$. Another physical implementation of the logical broadcast can be obtained by (n-1) **simultaneous unicasts** from the sender to each member of a single message corresponding to each member i through the routing path $R_{n,i}$. The **latency** is determined by the last member to get the broadcast and requires time D in the worst case. Hence:

$$Lt \leq D, \qquad Bdw = \sum_{i=1}^{n-1} R_{n,i} \times K \leq D \times n(n\text{-}1) \times K.$$

3.2.3. BD: It can be abstracted as the execution of two simultaneous broadcasts from each member to all others. We can analyze each broadcast from a user in each phase according to the down-flow stage of GDH.2. If we use a controlled-flooding implementation of broadcast we obtain:

$$Bdw = |E(G)| \times K \times 2n, \qquad Lt = D \times 2.$$

Using simultaneous unicasts instead, we obtain the following analytical expressions for the previous costs:

$Lt = Lt(M_i) = 2R_{i,j} \leq 2D$ (max. distance for M_i), and

$$Bdw = \sum_{i=1}^{n} 2 \sum_{j=1, j \neq i}^{n-1} R_{i,j} \times K \leq 2n \times (n\text{-}1) \times D \times K.$$

3.2.4 ING: It uses a logical ring where each member M_i processes any value received from $M_{i\text{-}1}$ and communicates a new value to M_{i+1} for (n-1) times, through $R_{i,i+1}$. Then: $Bdw(M_i) = R_{i,i+1} \times (n\text{-}1) \times K$. Observe that in the worst case scenario, the size of the physical implementation of the ring is $D \times (n\text{-}1)$. This also means that it takes time (and routing) $D \times (n\text{-}1)$ for the contribution of member M_{i+1} to reach eventually member M_i, or, equally, for any member to complete the (n-1) execution rounds. The related metrics are now:

$$Bdw = \sum_{i=1}^{n} R_{i,i+1} \times (n-1) \times K \leq D \times n \times (n\text{-}1) \times K, \quad \text{and}$$

$Lt(M_i) = Lt = D \times (n\text{-}1)$.

Summary: Table 1 summarizes *Lt* and *Bdw* of the 4 protocols (*Bdw* is divided by a factor of K), w.r.t. first the logical and then the *tb* implementation over AHNETs. Observe that the efficiency of essentially all *tb* protocols decreases by at least a factor of D.

Table 1. Performance of: (*a*) KA protocols over logical AHNets, (*b*) *tb* KA protocols over AHNets

	Wired *Lt*	Wired *Bdw*	tb-*Lt*	tb-*Bdw*
ING	n-1	$n(n\text{-}1)$	*D(n-1)*	*Dn(n-1)*
BD	2	$2n$	*2D*	*2Dn(n-1)*
GDH1	$2(n\text{-}1)$	$n(n\text{-}1)$	*2D(n-1)*	*Dn(n-1)*
GDH2	n	$(n\text{-}1)(n/2+2)$	*Dn*	*Dn(n-1)/2*

3.3 Implementation of GKA over MANETs

We have just evaluated the above protocols by executing them blindly on a real network, where multi-path routing is required for group members to communicate. We ran them on top of this framework based merely on member *ID*s as designated by the key generation algorithms. This *tb* approach leads to excessive unnecessary routing and relay nodes, and high bandwidth requirements, as also seen from the table of our relevant results. In AHNets, bandwidth and power consumption are valuable resources, that nodes cannot afford to waste. Reducing the combined costs resulting from the routing and communication becomes essential if we want to apply the KA schemes on resource-constrained AHNet. In this paper, we attempt to improve the efficiency of each of these protocols. Towards this end, we are exploring the potential of optimizing the combined communication and routing costs for these protocols with the use of a **topology-driven** "*td*" simulation of the logical network over any arbitrary AHNet graph G. We formulate several network simulation problems, which can be seen as variations of the classical "network embedding problem" studied in the area of Parallel Algorithms (see, e.g. [5]), with a few technical differences: (a) we have to embed a logical simple graph such as a cycle or a (broadcast) star into a given, arbitrary graph having the same number of nodes; (b) we obtain various optimization functions, depending on the behavior and performance of the original protocol on the logical network. Specifically, **each protocol** poses **two different optimization problems as the routing structure** of each defines a specific optimization function for each of the two metrics of **latency** and **bandwidth**. In summary, we define and later focus on minimizing the following 6 quantities, or performance metrics:

Bandwidth:

$$Bdw_1 = (n\text{-}1) \times \sum_{i=1}^{n} R_{i,i+1} \qquad (1),$$

$$Bdw_2 = \sum_{i=1}^{n-1} 2 \times i \times R_{i,i+1} \quad (2),$$

$Bdw_3 = (n\text{-}1)B_n$ (3),

Latency:

$$Lt_1 = \sum_{i=1}^{n} R_{i,i+1} \text{ , or } = (n\text{-}1) \times max\{ R_{i,i+1} \} \quad (4),$$

$$Lt_2 = 2 \times \sum_{i=1}^{n} R_{i,i+1} \quad (5),$$

$Lt_3 = \max\_length\ (B_n)$ (6).

Given these 6 quantities, we can develop bounds for the bandwidth and latency of the four considered protocols (after scaling them down by *K*) as follows: ING: from (1), (4), (5), GDH.1: from (1), (2), (5), GDH.2: from (3), (5), (6), BD: from (3), (6).

Thus, finding approximations of optimal solutions to these 6 quantities, or even improving those provided via the *tb* approach, results in more efficient metrics for the four protocols. We introduce our *td* approach in the following section. In the remainder of this section, we present two crucial low-cost auxiliary protocols that generate the core framework for running our *td* extensions: ***a randomized distributed leader election algorithm***, and ***an arbitrary rooted spanning tree generation algorithm***.

3.3.1 A: Root Election Protocol

We wish to construct a rooted spanning tree, denoted as ST, with the goal to perform a traversal of all tree members. In order to traverse the tree, a starting point is required. However, each node, initially at least, is ignorant of the global network topology: it is directly aware only of its neighbors and their identities (IDs). Even if we impose that a node with a certain attribute only initiates the tree traversal or a broadcast, the issue is that it is not known among all nodes which one acquires this attribute. Multiple nodes may attempt to start the traversal thinking that they are the best or the only candidates. The problem of a distributed network of nodes discovering one another through their network connections is known as the resource discovery problem, analyzed in [6, 7] and elsewhere. Discovering a single node with a certain attribute constitutes a subset of the former problem (nodes need to be aware of one node only) and we propose a different approach from the existing, in order to lower the resulting overhead as much as possible. To prevent a potentially overwhelming flooding, we use a *probabilistic approach* that allows only a small subset of members to initiate the search to learn or propagate the *ID* of the root.

3.3.1.1 Overview

In a first trial each of the *n* network members attempts to become an "initiator" with probability $p = \frac{1}{n}$. In the end, the member with the highest (or lowest) *ID* among the initiators becomes the global root. Each initiator propagates its *ID* through link state information. When a node receives a member *ID* it propagates it if it is greater than the *ID*s received before, otherwise it drops it. Eventually, all members will learn of the highest *ID* initiator, who becomes the root. If a node does not hear from any other node for sometime, that could designate that there are no initiators, or that the associated messages are lost. To remedy this, the next trial is run and each node re-runs the probabilistic trial for itself, after remaining idle for a given "silence period", defined from the maximum number of rounds required for a message to be propagated to the network.

3.3.1.2 Properties and Analysis

A message originating from the initiator is propagated along the vertices of the network graph. The message carrying the highest *ID* will be allowed to propagate to all network nodes. Messages carrying lower *ID*s will be propagated up to a point since they will be dropped by nodes that have already received messages of higher or equal *ID*s. In the worst case, they reach all nodes before they are dropped. In order to compute an upper bound on the average overhead incurred by *B*, we first bound the overhead due to any initiator and then multiply it by the expected number of initiators in the network. We see that for any initiator *v*, it takes at most *D* steps for all nodes for all nodes to receive *v*'s *ID* (since at round *i* all nodes at distance *i* from *v* receive its *ID*) or to discover that the current execution has produced no initiators. Moreover, for any initiator *v*, at most *3n* messages are sent to disseminate *v*'s *ID*, since each node will receive *v*'s *ID* at most twice and send it at most once. Let *X* be the random variable denoting the number of initiators; the probability that no initiator is generated in a trial is $Pr\{X=0\} = q = (1-\frac{1}{n})^n \approx 1/e$. Then, $Pr\{X=k\}=q^{k-1}(1-q)$ (geometric distribution), and the expected number of trials before success is $E[X]= 1/p$. Then the average *Bdw* of protocol *A* is:

$E[Bdw]= 1/p \times 3n = \frac{e}{e-1} \times 3n < 4.8\ n,$

And its average latency is

$E[Lt] = 1/p \times 2D = \frac{e}{e-1} \times 2D < 3.2D.$

3.3.2 B: Generation of a rooted Spanning Tree (ST)

In order to derive a good approximation for the minimum number of hops necessary to visit all group members, we use the well-known approach of simulating Hamiltonian paths and cycles with a full walk over a rooted spanning tree – denoted as ST - of the connected network graph G (*V*, *E*). A ST of *G* is a connected acyclic sub-graph $T(V', E')$ of *G* , s.t. $V=V'$ and $E' \subseteq E$, in fact $|E'| = |V|\text{-}1$. In a rooted ST, the tree edges are consistently directed w.r.t. a particular node (root). The purpose of our framework provides the flexibility to generate STs with potentially various attributes (i.e. low weight, where weight is typically the latency, loss rate, inverse of bandwidth, or low node degree to promote load balancing, low diameter, low edge count, etc.) at the expense of extra overhead however. The impact of running our KA protocols on top of STs with specific characteristics is not within the scope of this current work, it is being explored in our on-going work. For the purposes of this study, we are interested in applying a very simple and lightweight algorithm for the generation of a ST. Hence, an arbitrary ST is sufficient for our current focus. On the other hand, we do not consider a weighted graph for our current GKA algorithms. Below, we give an overview of the following - among others - simple, low cost protocols to generate a ST.

3.3.2.1 Overview

It turns out that for the applications in this paper (i.e. to obtain upper bounds on latency and bandwidth) we can even use the most naive approach of generating a distributed ST after reduction to the standard sequential algorithm (that is, first each node propagates its neighborhood information to all other nodes, and then each node

computes the same ST by running the sequential algorithm). This algorithm has latency O(D) and bandwidth O(n^2) and yet it only affects the overall performance of our KA protocols by a low-order factor.

A more communication-efficient approach goes as follows. Each node x that becomes part of the ST sends a single "connect" message to each one of its neighbors (except for its ancestor in the tree). A neighbor node y accepts node x as its parent (and sends an ACK message to all 1-hop neighbors) if it receives the "connect" message for the first time from node x (i.e. it did not receive and accept a prior request from another node and it is not part of the ST already). It is obvious that this algorithm converges (as long as the graph remains connected) and does not generate cycles, since each node accepts only one "connect" message, that is, it obtains only one parent.

More generally, to deal with mobility issues in AHNets (which we do not target in this paper), one may use self-stabilizing spanning tree generation protocols (see, e.g. Dolev *at al* [11], or [13] for a survey). The algorithm based on [11] has latency O(D) and bandwidth approximately $Bdw = O(D \times n)$.

3.4 Topology Driven AHNet Simulation for known GKA Schemes

We now use the auxiliary protocols A, B, to generate efficient embeddings of logical networks over arbitrary ad hoc networks for the protocols we are considering. We analyze the bandwidth and latency of the resulting performance by computing upper bounds on the six quantities related to the optimization problems defined in section 3.3. All bounds reported in this section are scaled by K.

3.4.1 Solution to (1), (5): ST Full Walk

Using protocols A and B we can simulate Hamiltonian paths and cycles, by just performing a full walk of the rooted ST. Any of the well-known tree visit walks traverses every edge exactly twice, resulting in a cost twice the number of tree members. We then obtain that: $\sum_{i=1}^{n-1} R_{i,i+1} \leq 2n$ in the simulation of **Hamiltonian paths** and **rings**, and the quantities (1), (5) are upper bounded as

(1) $\leq 2n(n-1)$ and

(5) $\leq 4n$.

However, this is still not enough to obtain an efficient simulation for problem (2), which we now address.

3.4.2 Solution to (2), (4): Extended ING Ring with Dilation 2

Under this approach, the basic idea is that of "doubling" the size of the Hamiltonian ring or path by using the rooted ST generated obtained at the end of the distributed protocols described in (A, B), and then executing the same protocol previously run over a logical path or ring, which is now of at most twice the size. Specifically, a (say, preorder) visit of all ST nodes goes through at most (2n-1) nodes. We map every node visited consecutively during the full walk, to an "entity" placed on a new Hamiltonian ring/path. This ring/path consists of (2n-1) "entities" that represent the nodes visited in the order of the full walk. For example, a visit of a depth-2 complete binary tree results in the sequence of node visits: N_1, N_2, N_4, N_2, N_5, N_2, N_1, N_3, N_6, N_3, N_7, N_3, N_1. These entities are in fact the n = 7 network nodes, some of which are replicated in the extended ring/path, in the spaces designated by the full walk. All parties are one hop away from their predecessor and successor (if any) in the new path, as is the case in the original logical n-size ring/path.

3.4.3 Solution to (2): Closest Point Heuristic

The following inequality should hold for the minimization of (2) that corresponds to GDH.1 Bdw: $R_{n-1,n} \leq R_{n-2,n-1} \leq .... \leq R_{1,2}$.

Since the number of values a member communicates to its successor in the schedule is always incremental in GDH.1, the routing paths of successive members must be non-increasing. In GDH.1 the last visited member M_n uses the established schedule backwards to forward the intermediate messages to the rest of members. An approximation to (2) is given by the ***closest point heuristic***. It begins with a trivial cycle of an arbitrarily chosen vertex. At each step, a vertex u that is not on the cycle but whose distance to any vertex on the cycle (e.g. vertex v) is minimum is identified. We extend the cycle to include u by inserting u just after v. We repeat this procedure until all vertices are on the cycle. This heuristic returns a tour whose total cost is not more than twice the cost of an optimal tour. If we fix the **backward schedule first**, so that the first edge selected in the cycle (minimum) is assigned to relay the maximum number of messages (n-1), the second edge is assigned to relay (n-2) messages, etc., we immediately satisfy the "non-incremental" requirement. We can obtain a ST by deleting any edge from a tour, and we already acquire the MST in our setting. Thus, the closest point heuristic has direct application to our problem. We use the **appropriate traversal method** to visit all vertices of the **ST** and establish the **GDH.1 backward schedule first**. By examining the common traversal methods, we select the *pre-order* tree walk. An intuitive reason for this is that a *pre-order* tree walk prints (visits) the root before the values in either sub-tree. So, it uses a rather greedy approach by adding the "best nodes" first, in a forward manner, the earliest possible in the generated schedule. Under this method, the **root** is the **last** member **visited**, whereas the **node** that **initiates** GDH.1 is the last node listed by the *pre-order* walk. We want to upper limit (2) by using a *td* schedule that generates the backward GDH.1 path of the n nodes. The topology of the nodes and their configuration determines the nature of the formed ST, and consequently the value of (2). Towards this end, we studied a few examples of particular ST and computed (2) over them. A few indicative cases considered are: the *single chain tree*, the *star tree of depth X*, the *fully balanced Z-ary tree*, etc. It can be shown that the value of (2) computed via pre-order traversal of any *ST*, is upper bounded by $\frac{3}{2}n^2$.

3.4.4 Solution to (3), (6): Broadcast Tree

The controlled flooding strategy discussed before is based on a **broadcast tree (BT)** over the network nodes. An internal node that receives a message from its parent, forwards it to its immediate tree offspring only once. For each different source, the same ST is used, but rooted at the given source each time. It is assumed the parent-children associations are modified on the fly. Now, we only need to perform broadcast over a rooted tree, which is quite simple as the root can broadcast to its children who can recursively broadcast into their sub-trees. W.r.t. GDH.2, (3) is **further improved** if each parent **removes** this part from the received message that corresponds to itself before forwarding the **remaining message** to its **children**. Under the

worst case scenario that the ST is a chain, (3) $= \frac{(n-1)n}{2}$. Computing (3), (6) on a BT with height $h(BT)$ we obtain: (3)$< \frac{(n-1)n}{2}$, and (6) $\leq h(BT) \leq D$.

3.4.5 Solution to (3),(6): Simultaneous Unicasts

Another **physical implementation** of the logical broadcast is obtained by (n-1) **simultaneous unicasts** from the sender M_n to each member M_i of a single message through $R_{n,i}$. We then obtain: (3) $= \sum_{i=1}^{n-1} R_{n,i} \leq D \times (n\text{-}1)$, and (6) $\leq D$.

Summary: Table 2 summarizes the upper bounds on the bandwidth and latency of the 4 protocols, when the above *td* improvements are performed; all scaled by K. These values should be contrasted with those in Table 1. In almost all cases the efficiency of all protocol increases by a factor of D or n.

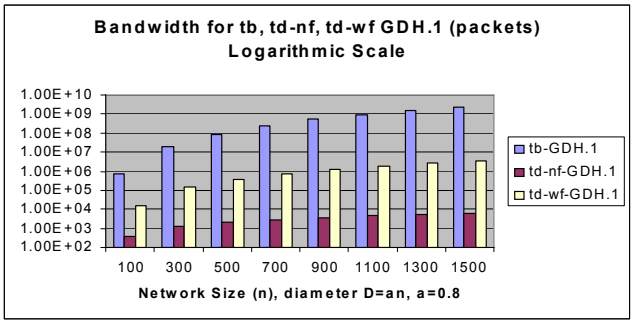
Table 2. Performance Comparison of 4 td protocols over AHNets.

	Latency	Bandwidth	Exp.
tdING	$7n+4.2D+3$	$2n^2 + n \times (2D+7.6)$	n^2
tdBD	$3D$	$2n^2 + (D\text{-}2)n$	n^2+n
tdGDH1	$4.2D+9n$	$\frac{3}{2}n^2 + (D+7.6)n$	$(n^2+3n)/2$
tdGDH2	$7n+6.2D$	$\frac{5}{2}n^2 + \frac{17}{2} + (D\text{-}0.4)n$	$(n^2+3n)/2$

3.5 Comparative Performance Evaluation

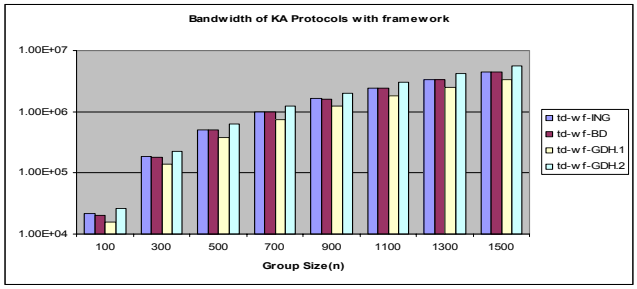
We compared the *td*-known KA schemes in terms of the following characteristics: *a*) latency, *b*) bandwidth resulting from the combined consideration of the relays for the message exchanges over multi-hop routing, which reflects the actual communication overhead in the network, and *c*) the total number of modular exponentiations .

Figure 1. Comparative evaluation of *tb*-GDH.1 and *td*-GDH.1 (framework (wf) vs. non-framework (nf) consideration).



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

Figure 2. A comparison of td-GKA protocols w.r.t. bandwidth, including the OH for framework generation, *D* is assumed spheric.



4. REFERENCES

[1] M. Steiner, G. Tsudik G, M. Waidner, Diffie-Hellman Key distribution extended to groups. *3d ACM Conf. on Computer & Communication Security*, ACM Press, 1996, 31-37.

[2] W.Diffie, M.Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, 22 (1976), 644-654.

[3] M. Burmester, Y. Desmedt. A Secure and Efficient Conference Key Distribution System. *Advances in Cryptology-EUROCRYPT '94, Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, 1994.

[4] I. Ingemarsson, D. Tang, C. Wong. A Conference Key Distribution System. *IEEE Transactions on Information Theory*, 28(5):714-720, Sept. 1982

[5] F.T. Leighton. Introduction to Parallel Algorithms and Architectures, Morgan Kauffman, 1992

[6] M. Harchol, T. Leighton, D. Lewin. Resource discovering in distributed networks. *Procs 15th, ACM Symp., on Principles of Distributed Computing*, May 1999, 229-237

[7] S. Kutten, D. Peleg. Deterministic distributed resource discovery. *In 19th annual ACM SIGOPS symp. on Principles of Distributed Computing*, Portland, OR, 16-19 July 2000.

[8] R. Gallager, P. Humblet, P. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems* (TOPLAS), 5(1): 66-77, January 1983

[9] C. Cheng, I. Cimet, S. Kumar. A protocol to maintain a minimum spanning tree in a dynamic topology. *In symp. Procs' on Communication architectures and protocols*, 330-337, ACM Press, 1988

[10] A. Mooij, W. Wesselink. *A formal analysis of a dynamic distributed spanning tree algorithm.* Computer Science TR 03-16, Technische Universiteit Eindhoven, Dec. 2003

[11] S. Dolev, A. Israeli, S. Moran. Self-stabilization of dynamic systems assuming only read/write atomicity. *Distributed Computing*, 7:3-16,1993.

[12] L. Blin, F. Butelle. The first approximated distributed algorithm for the minimum degree spanning tree problem on general graphs. *Int'l Parallel and Distributed Processing Symposium* (IPDPS'03), p. 161a.

[13] F. Gärtner, A Survey of Self-Stabilizing Spanning-Tree Construction Algorithms, EPFL Technical Report, 2003.