Journal of Information Assurance and Security 1 (2008) 67-79

Network Simulation of Group Key Management Protocols

David Manz, Jim Alves-Foss and Shanyu Zheng
Center for Secure and Dependable Systems
University of Idaho
POBOX 441008
Moscow, ID 83844-1008
davidmanz@vandals.uidaho.edu, jimaf@uidaho.edu, zhengshanyu@hotmail.com

Abstract: : The need for rapidly configurable, secure communication among groups of participants has resulted in the study of group key agreement protocols. The study of these protocols has been primarily theoretical. In this paper, we present the results of simulation studies of the methods provided by four group-key agreement protocols, EGK, TGDH, STR and CCEGK. The results of the simulation clarify the theoretical metrics, but also provide insight into the actual relative impact of the metrics, specifically the impact of synchronization. Overall CCEGK performed better in all categories than the other three protocols.
Keywords: Group-key agreement, simulation, NS-2

I. Introduction

Currently, much of group key management analysis remains theoretical [9, 16]. We have created simulations to mimic the ways the four group key management protocols, Efficient Group Key (EGK) [1], Tree Group Diffie-Hellman (TGDH) [9, 12], Skinny TRee (STR) [10, 11] and Computation and Communication Efficient Group Key (CCEGK) [15], will perform in real-world settings. The goal of this research is to provide a more realistic evaluation of the behavior of group key management protocols, taking into account the performance cost of bandwidth utilization and timing. In order to accomplish this, we first establish the feasibility of implementing group key management protocols over network simulation software. Second, we corroborate previous theoretical estimates from our earlier work [15, 16]. Finally, we compare simulated results against our original theoretical results which allows for the determination of how accurately they correlate to real-world networks.

Group key management is a subcategory of cryptography. Cryptography concerns itself with securing information so that unauthorized individuals cannot understand the messages sent. Key management is responsible for the proper and secure distribution, creation, and revocation of the keys used to secure messages.

Group key management, sometimes called group key exchange, applies to groups of participants who want to communicate securely through insecure channels. In this paper, we focus on decentralized group key management, often called group key agreement. Where each participant contributes to the entropy of the shared key. This type of group key management is more suitable for large and dynamic groups, by avoiding centralized control and potential computational or communication bottlenecks.

The idea of group key agreement stems from the earlier work of two-party key management. In 1976, Diffie and Hellman introduced a two-party key exchange protocol, DH, that allowed two participants to create a private key through the use of publicly exchanged messages [6]. This protocol allowed two participants, without any prior shared secrets, to securely establish a shared session key. DH is now at the heart of many two-party secure communication protocols, including Secure Socket Layer [7] and Secure Shell [14].

With the prevalence of the Internet and networked technologies, there are applications that would benefit from a group key agreement protocol that provides the same protection as DH, but for groups of more than two participants. This list includes conference calls, distributed computation, whiteboards, distributed databases, Unmanned Aerial Vehicles, and battlefield communications, among many others. To ensure secure and reliable communication in these applications, there have been several attempts to create efficient group key agreement protocols for large and dynamic groups based on the DH algorithm [1, 3–5, 8, 13, 15].

Although the DH algorithm was first created in 1976, a substantial flurry of new work arose in the field during the late 1990s. Prior to 2000, different groups simultaneously developed their own versions of group key agreement protocols that utilized tree-based information structures. The algorithms utilized by these protocols scale logarithmically with the number of participants, increasing performance greatly over prior work that scales linearly with the number of par-

Received January 18, 2008

ticipants. In an attempt to extend the usefulness of two-party key management to an n party arena, these groups created the following protocols: Efficient Group Key (EGK) [1], Tree Group Diffie-Hellman (TGDH) [9, 12], and Skinny TRee(STR) [10, 11]. Later on, Computation and Communication Efficient Group Key (CCEGK) [15, 16] was added to the mix.
With these four protocols, (CCEGK, EGK, TGDH, STR), the designers attempted to address a handful of performance concerns. Chief among them were communication costs and computation costs. The designers of EGK and TGDH focused more on the cost of computation than on that of communication. STR improved the cost of communication over the cost of computation, in keeping with Becker and Willie's original work on efficient communication in group key protocols [3]. The newest protocol, CCEGK [15] emphasizes both the costs of communication and computation.
The analysis of these protocols has been primarily theoretical. The main goals of the work presented in this paper are to simulate these protocols to both validate and extend the theoretical comparisons. Specifically we focus on:

- Implementing additional core functionality of NS-2 and extending it to the four group key management protocols.
- Implementing two network topologies (Internet, routed) with various test configurations.
- Illustrating the current limitations to the group key management protocols as described in the literature.
- Corroborating our theoretical data with this experimental data.

The remainder of this paper is organized as follows. In Section II we define the performance metrics used to compare the four protocols, and provide a brief discussion of NS-2, the simulator used in this work. In Section III we discuss the two specific simulation scenarios evaluated. Sections IV & V we present the results of the two scenarios. Finally we conclude in Section VI with a discussion of the results and where we plan to go in the future.

II. Background

A. Costs and Performance Metrics

In this section we define how the protocols are compared. Most protocols implement five to eight standard operations (or methods). These operations define how the protocol manages and secures the shared group keys.
For example, in order for an individual to join a group, he or she must initiate a join operation. Then the entire group participates in this operation, and at the end, the entire group, including the new member, has a new shared key. The rest of the operations are quite similar.
It is commonly held that there are eight main operations for each group key management protocol. However, we have found that not every protocol implements, or at least documents, every operation (see Table 1.)
Due to the simultaneous development of these group key management protocols, standardization of categories and ambiguities in their classification led to the development of categories to be applied to all operations discussed in this paper.
We define the common eight operations as follows:

Initialization The initial creation of the group key and organization of the key management infrastructure.

Join (Add) This operation brings a new member into the existing group.

Mass join (Mass add) Allows many new members to be added to an existing group simultaneously when these new members have not already formed a group of their own.

Merge (Group fusion) As opposed to mass join, merge is used when another group is combined with the existing group to become a new group.

Leave (Remove) This operation is used to remove a member from the group.

Mass leave This operation is used when multiple members are simultaneously removed from the existing group.

Split (Partition or Group fission) Different from mass leave, split occurs when a single group is divided into two or more component groups.

Key refresh This operations is used to refresh the existing group key in order to prevent an adversary from having a sufficient time or resources to break the key.

Not every protocol documented in the literature includes the operations of mass add, split, initialization, and refresh. Therefore we limited our simulation to the operations that have been sufficiently and completely described to enable their implementation on a real network. Specifically we analyze the operations of merge, add, leave, and mass leave.
To compare these operations, we use a set of metrics related to the cost of their execution. Costs come in two categories, communication and computation; however, because NS-2 is a communication simulator, only the communication costs are taken into account in this paper. We assume that the theoretical computation costs are sufficient for comparison [16].

Number of rounds A generic time unit used to compare the number of steps taken in different operations. The protocols often require synchronization between rounds, so this number becomes important when real-world implementations take synchronization time into account.

Table 1: Group Key Agreement Protocols and their Methods

	TGDH	STR	EGK	CCEGK
initialization			X	X
join (add)	X	X	X	X
mass join (mass add)			X	X
merge (group fusion)	X	X	X	X
leave (remove)	X	X	X	X
mass leave	X	X	X	X
split (partition or group fusion)			X	X
refresh	X	X		X

Number of unicast messages The sum of all messages every member sends to other single members in the group in the operation. This number is useful for determining total communication and is important if many or all nodes are on the same network collision domain, thus forcing these messages to be sent sequentially and not in parallel.

Number of broadcast messages The sum of all messages sent by each member to all the other members in the group for the operation. Since the messages go to all members of the group, it greatly affects total communication costs depending upon the underlying network topology.

Number of messages The sum of all unicast messages and broadcast messages. We use this number to determine the total time of communication in an underlying broadcast network.

B. Introduction to NS-2

Currently there are no simulators like Network Simulator Version 2 (NS-2) for group key management protocols, and consequently, any comparative analysis is done at a theoretical level [15,16]. NS-2 allows a user to emulate network traffic as it actually occurs on physical networks. The software simulates switches, routers, connections and traffic sources of various kinds. Traditionally used to model new versions of communications protocols like Transmission Control Protocol (TCP) and Ethernet, NS-2 can also model new forms of traffic generation, like the group key management simulation. The field of group key management would benefit from a real implementation of these protocols or a realistic simulation on a reliable network simulator. The NS-2 implementation presented in this paper aims to do just that.
Kim et al have worked on simulating their protocols on very specific hardware topologies [9]. While limited to the specifics of the particular computational hardware, this contribution to realistic simulations of group key management must be explored in future work.
The goal of the simulation presented in this paper is to provide a more realistic and real-world evaluation to the field of group key management protocols. Our research will establish the feasibility of implementing group key management protocols over NS-2 network simulation software. Additionally, basing these experiments on our previous experience with theoretical analysis [16], we help shed light on how realistic and meaningful previous literature has been [3–5, 8, 13] by corroborating previous theoretical estimates.

III. Simulation Design

A. Network Topologies

Two distinct network topologies were chosen for this simulation. In an attempt to cover as wide a range as possible, and to better understand where group key protocols might be employed, we simulate a small portion of the Internet and a router subnetwork. The nodes 0-9 are the member nodes used in the operations and node 10 represented the switch/router.
A very small slice of the Internet is depicted in Figure 2. This was created to mimic a small subsection of the Internet with numerous interconnections and hops between nodes. The nodes depicted with circles (and numbered 0-9) are the members of the group operations. The remaining nodes depict intermediate nodes in the internet.
The routed subnetwork we used is shown in Figure 1. The subnetwork utilizes a star topology where every user is directly connected to the shared router.

B. Simulation Overview

The goals of the research reported in this paper are to verify the validity of using NS-2 to simulate communication performance for group key management protocols and to corroborate our previous theoretical estimates [16].
We ran a total of 56 different experiments. We tested four different operations (join, leave, merge, and mass leave), for four different protocols, CCEGK, EGK, TGDH, and STR, on two different topologies with several different starting group sizes.
The simulations are divided into two parts. Part 1 explores the bandwidth utilization and timing costs for the four protocols, CCEGK, EGK, TGDH, and STR, over two network topologies (subnetwork and Internet) for the four operations, join, leave, merge, and mass leave. The results of Part 1 enable us to corroborate our prior theoretical data. Part 2 extends this simulation by examining two specific cases (join

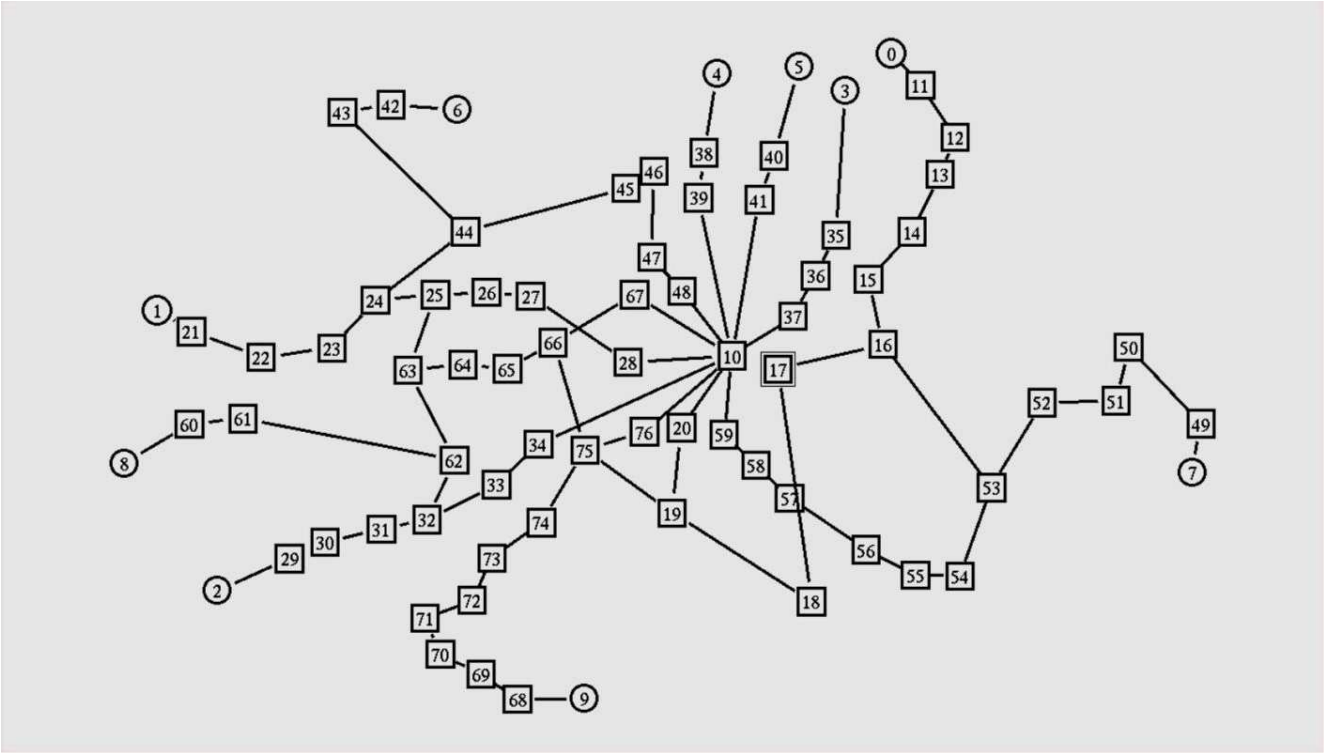

Figure. 2: Internet Topology

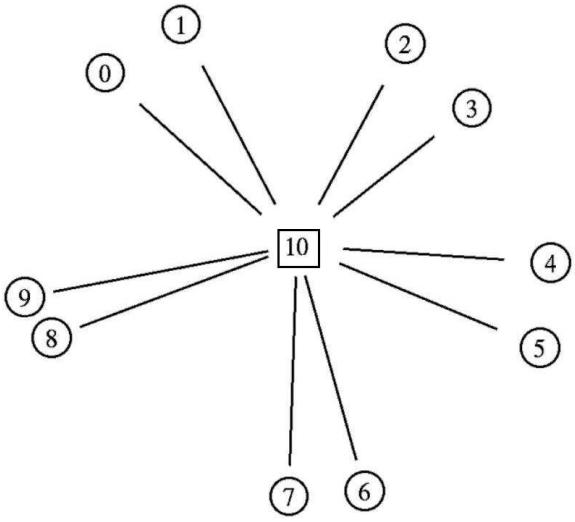

Figure. 1: Subnetwork Topology

and mass leave operations) with group sizes of 5, 10, and 30. In Part 1 the join operation test was a sequential add of exactly 10 members to an empty group. In Part 2, the join operation is tested three times, sequentially adding 5, 10, and 30 members to an empty group. This approach was explored to give a better insight into the trends in communication costs. Similarly, in Part 1, the mass leave operation involved having three members simultaneously leave a group of size 10. In Part 2 the mass leave operation was used to simultaneously remove three members from groups of size 5, 10, and 30.

In order to meet the goals described above, we created simulations that accurately reflected certain "real world" situations, based on the two topologies. For our simulations we needed to create experiments that highlighted each protocol's performance for each tested operation. We performed 14 different experiments for each of the four protocols (two groups of four experiments in Part 1, one group for each network topology, and one group of six experiments in Part 2), giving us a total of 56 different experiments. The parameters of the 14 different experiments are summarized in Table 1.

C. Protocol Comparison for Fixed Group Size (Part 1)

Our analysis of the simulations divides up the network traffic over time to evaluate how the performance varies between group key management protocols. One key metric we use is bandwidth utilization per operation. The term "bandwidth", used in our metrics, should more accurately be called

throughput. However, in this paper we use the word bandwidth to define the total throughput capacity of a link, given in bytes per second, following the definition often found in trade literature. Additionally, bandwidth utilized is a measurement meant to indicate how a given service is utilizing the available bandwidth. For example, if service A utilized 20% of a link with 100 Mbps bandwidth, we would say that service A had a bandwidth utilization of 20 Mbps. The term "bandwidth utilization" is meant to convey the traffic cost of certain services.

We graph the results of the simulations based on bandwidth utilization over a fixed time unit for routed or subnetwork topology, and then again for Internet topology. We also graph the aggregate total bandwidth utilized for each protocol for each operation. Note that the amount of traffic over the routed topology is exactly the same as the amount of traffic over the Internet topology. The only way they differ is through packet loss or corruption, which we do not simulate. Our simulator samples the traffic by summing up the total amount of traffic on the entire network during several intervals and reports that amount. Therefore, depending on how we slice our time, we could get different bandwidth utilization results per slice; however, the total amount of traffic on the network during a full run will never change. To be impartial, we treated all four protocols using the same sampling scheme. Since the Internet topology takes substantially longer, we increased the time slices to a tenth of a second, while in the subnetwork topologies, the time value is represented in the nearest hundredth of a second.

For the sake of this simulation, a typical packet is 1050 bytes. This allows around 1000 bytes for the security key and group key management overhead, leaving 50 bytes for network communication overhead. Packet size selection does not affect the trends and results that were discovered. Table 2 summarizes the parameters of the test cases. The first four (Join, Leave, Mass Leave, and Merge) are for Part 1. The latter six (5+3 Join, 10+3 Join, and 30+3 Join, and Mass Merge (5, 10, 30)) are for Part 2. Initial size is the number of individual members in the group at the start of the experiment. Ending size is the number of members in the group at the end of the experiment. The "# of Groups" is the number of groups used in the experiment and the "# of Ops" is the number of operations (e.g. sequential join operations) in the experiment. Lastly, the "# of Topologies" is the number of unique topologies tested in the experiment. Table 3 summarizes the overall NS-2 network parameters for all experiments.

D. Part 1: Bandwidth utilization and timing costs with two network topologies

For Part 1, the test of the join operation involves 10 sequential join operations, the leave operation involves three sequential leave operations, and the merge and mass leave are both one operation each. Furthermore, the merge operation consists of two groups of size five merging together, while the mass leave operation is a simultaneous mass leave of three members. All operations in Part 1 are simulated on both the Internet and subnetwork (routed) topology. The simulations are run on both the "internet" and "subnetwork" topologies. Each operation (except for join) starts with 10 members. The link speed (bandwidth) is 100 Mbps for the subnetwork topology and 1 Mbps for the Internet topology.

1) Incremental Join Method

For the first experiment, we start with 10 individual nodes and a group size of zero. The nodes are added sequentially to this group for both topologies.

2) Incremental Leave Method

For the leave experiment, we start with a group of size 10 and sequentially remove three members for both topologies (as with all experiments in Part 1). We selected nodes 9, 3, and 0 to be removed from our group. These nodes were randomly selected before the experiments were started. These were selected to give the simulator a more realistic approach than simply removing the first three nodes.

3) Mass Leave

The mass leave operation experiment starts with a group of size 10 and then three (3, 4, and 9) members perform a mass leave for both topologies. These members were randomly selected before the start of the simulation to provide a more realistic mass leave operation.

4) Merge

For the merge operation experiment, we start with two groups of size 5 and merge them together for both topologies.

E. Part 2: Bandwidth utilization and timing costs with differing group sizes

The goal of Part 2 of the simulation is to determine the realistic communication costs for two operations, join and mass leave, when the group size varies. Due to similarities between performance of these operations over the subnetwork and Internet topologies, we use only the subnetwork topology for this experiments. We simulate the total bandwidth utilization per protocol for groups of size 5, 10, and 30. The simulation illustrates trends that are affected by group size and allows meaningful comparison between protocols.

1) n+3 Join

For the n+3 join, we start with a group with n members and sequentially join three additional members to it where n is 5, 10, or 30.

Table 2: Simulation Design Parameters

Part 1	Initial size	Ending size	# of Groups	# of Ops	# of Topologies
Join	0	10	1	10	2
Leave	10	7	1	3	2
Mass Leave	10	7	1	1	2
Merge	10	10	2	1	2
Part 2					
5+3 Join	5	8	1	3	1
10+3 Join	10	13	1	3	1
30+3 Join	30	33	1	3	1
Mass Leave 5	5	2	1	1	1
Mass Leave 10	10	7	1	1	1
Mass Leave 30	30	27	1	1	1

Table 3: Summary of Common Parameters for all Experiments

Packet Size	1050 bytes* (for one sender to one receiver)
Simulator	NS-2
Underlying Network Protocol	TCP/IP
Network Speed (Internet)	1 Mbps
Network Speed (Subnetwork)	100 Mbps
Link Noise	none
Router queuing	DropTail

2) Mass Leave n

For the Mass Leave n, we start with a group with *n* members and simultaneously remove three members from the group where *n* is 5, 10, or 30.

IV. Simulation Results Part 1

A. Incremental Join Method

The join operation was analyzed for the four protocols. Examining the time axis of Figure 3 for the routed topology, it is clear that CCEGK and EGK are following similar paths, differing only in a bit of amplitude. This is because CCEGK and EGK both perform the join operation in one round (per addition). The only real variance is EGK's additional overhead (a bit of extra traffic), which accounts for the greater magnitude. Specifically, the CCEGK protocol calls for one broadcast message and one unicast message. Given that every broadcast message is sent to every member of the group, it makes sense that any given broadcast message will consume more bandwidth than a given unicast message. Consequently, the EGK protocol, which performs essentially the same join operation that CCEGK does (but uses two broadcast operations and no unicast), utilizes more bandwidth because of added overhead. Additionally, STR performs exactly the same as TGDH, while taking longer to finish than EGK and CCEGK (because STR and TGDH use multiple rounds per add). Again, because the STR protocol is based on the TGDH protocol, it makes sense that their join costs are the same.
For the Internet topology (Figure 4), CCEGK and EGK once again have similar contours; however, both STR and TGDH take over three times as long to complete. This added time is attributed to their use of multiple rounds per add. When a topology contains several links (as in the Internet topology with up to 20 hops in one TCP connection), the round-trip time becomes a significant factor. Additionally, because TGDH and STR are broadcasting, every message sent is forced to be sent to every member, which greatly adds to bandwidth consumption or utilization. Finally, since both TGDH and STR broadcast the full key tree for much of their traffic, their messages can be up to 10 times as large, compared to the standard 1050 bytes for CCEGK and EGK. This also contributes to their much higher bandwidth utilization.
When looking at total network traffic for join (Figure 7), TGDH and STR utilize around four times as much bandwidth as EGK, which is more expensive than CCEGK.

B. Incremental Leave Method

For the subnetwork topology (Figure 5), we note that CCEGK, STR and TGDH all perform identically. This corroborates our prior theoretical results [16]. Indeed, each of the three protocols implements the same operation for leave, so it should be no surprise that their performance on a routed network is identical. Unfortunately, because EGK recreates the entire tree each time a node leaves, the costs in time and bandwidth utilization are larger. Looking at EGK's cost, it is clear that it follows an inclining sawtooth-shaped cost curve. This is due to the fact that the EGK protocol has to recreate the key tree three times, one for each remove. The amplitude of all the values gets slightly smaller for each successive leave operation because there are fewer nodes in the key tree.
The Internet topology (Figure 6) results are quite similar to the subnetwork topology, as expected. However, the time

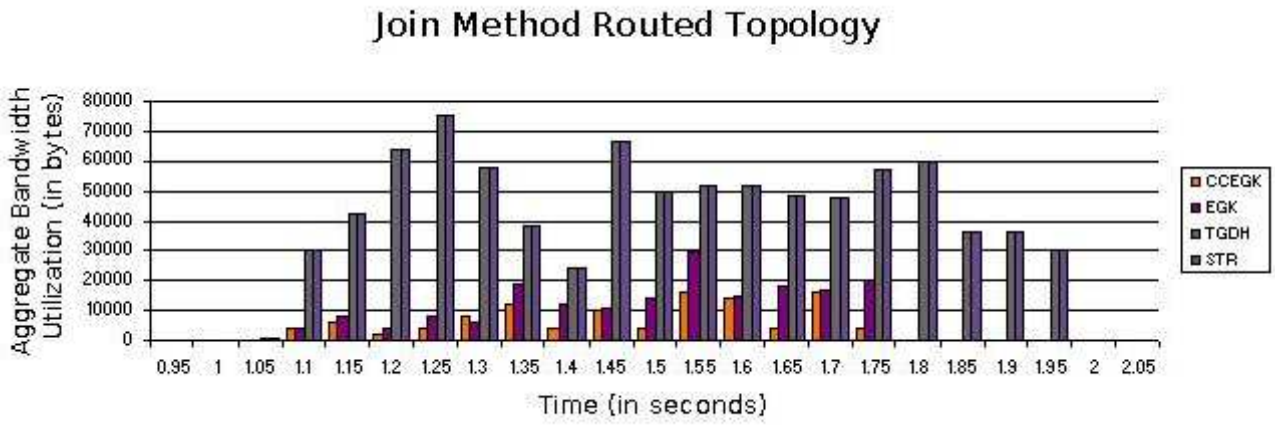

Figure. 3: Bandwidth Utilization for Join Method Routed

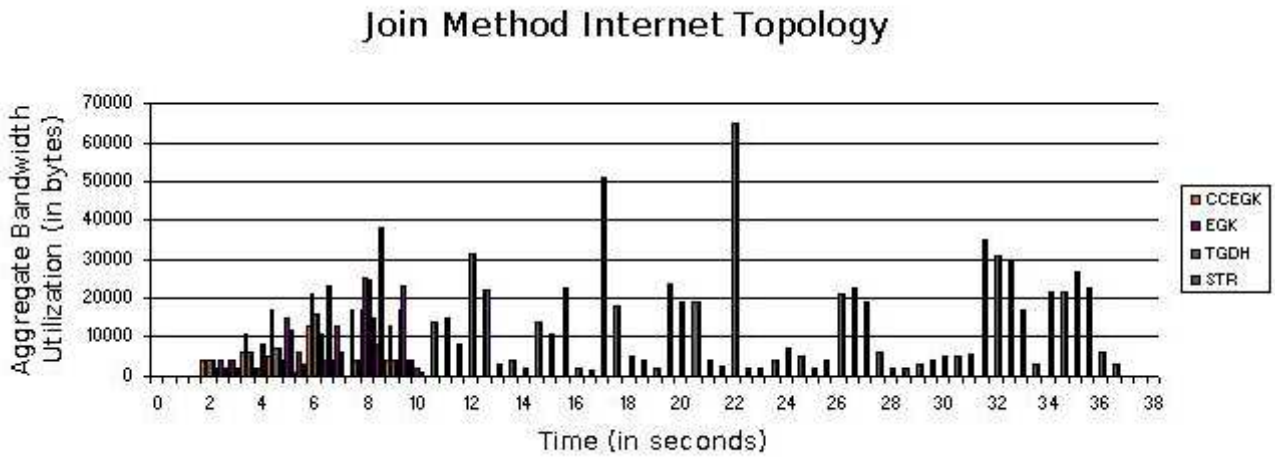

Figure. 4: Bandwidth Utilization for Join Method Internet

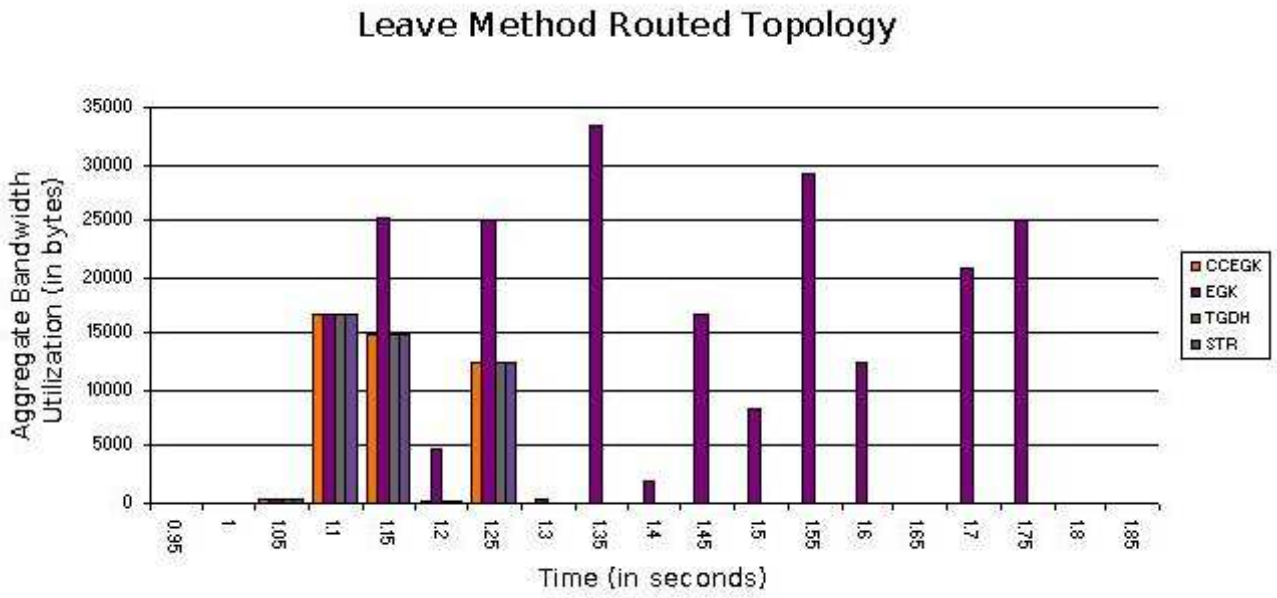

Figure. 5: Bandwidth Utilization for Leave Method (Routed)

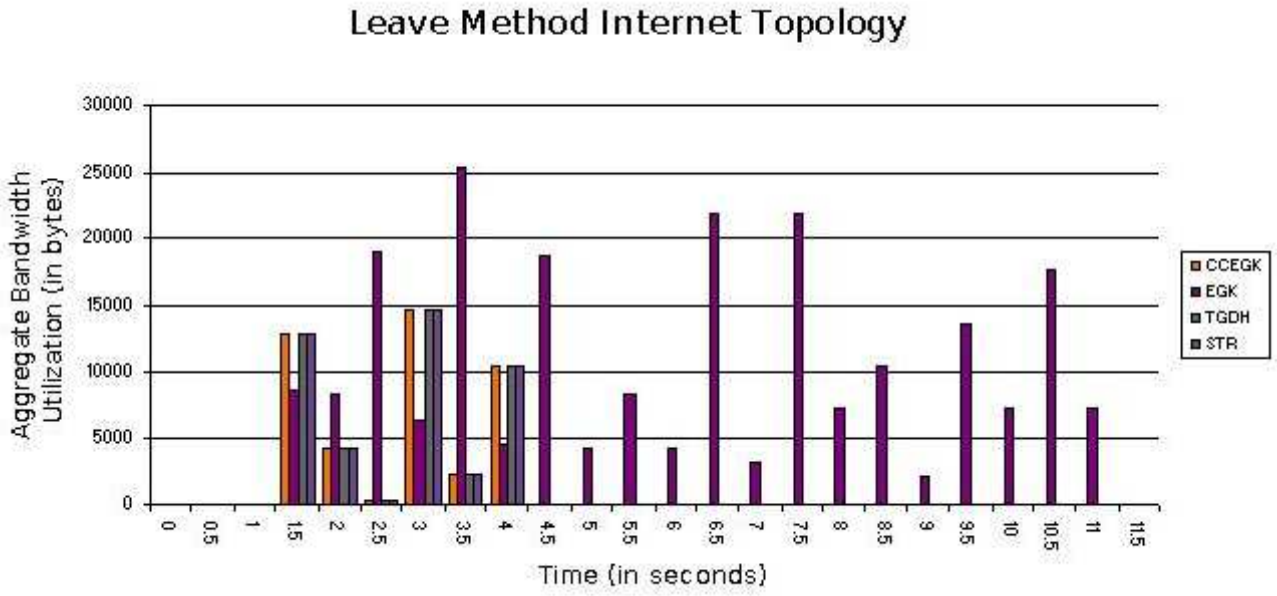

Figure. 6: Bandwidth Utilization for Leave Method (Internet)

table is much longer for the Internet topology. What took the subnetwork topology 0.3 seconds takes the Internet topology nearly four seconds. This again is attributed to the vastly longer path the packets must take.
When looking at total bandwidth utilization (Figures 7-8), it is clear that CCEGK, TGDH and STR consume the same amount of bandwidth EGK, however, consumes over four times more.

C. Mass Leave

TGDH and CCEGK protocols both can implement the same mass leave operation as the STR protocol, which is quite inexpensive. Additionally, however, they can implement a mass leave operation that re-balances the tree [16], something STR does not do.
The costs for CCEGK and EGK are identical (Figure 9). CCEGK and TGDH have no better means of performing the mass leave operation, so they implement a similar operation to the one that EGK uses for mass leave (and leave), a complete rebuild of the internal tree structure. However, it is important to note that while TGDH and CCEGK implement the same operation for mass leave, TGDH again broadcasts the full tree structure which greatly increases overall bandwidth consumption[1].
For the mass leave operation in the Internet topology (Figure 11) there are two things to note. First, the time axis is far longer than its companion subnetwork topology. This is due to the fact that the Internet topology takes much longer to traverse. Second, unlike the preceding graph, CCEGK and EGK are no longer identical, and even though they are performing the exact same operation with the same amount of traffic, their graphs appear different. After careful examination, we find that the two protocols are still essentially the same, but the sampling rate has affected how they are displayed. For example, if one takes the bandwidth utilization slice of CCEGK that occurs at 4 seconds and adds it to the slice at 3.5, it will equal the bandwidth utilization for EGK at 3.5 seconds, and so on. The data have become offset because one sampling slightly lagged behind the other, and since this is a discrete bandwidth utilization analyzer, our sampling rate managed to make it look like CCEGK was offset in time. STR still costs less than the other protocols.
CCEGK and EGK are identical in bandwidth consumption (Figures 9). Furthermore, STR is cheaper than either of these, while TGDH is more expensive.

D. Merge

The merge operation separates the protocols and is a great example of how similar the pairs of protocols are. CCEGK and EGK perform identically, as do STR and TGDH (Figures 13-14). This will come as little surprise to those who know the history of these particular group key management

[1] After these experiments were completed, we developed a more efficient mass leave for CCEGK [16].

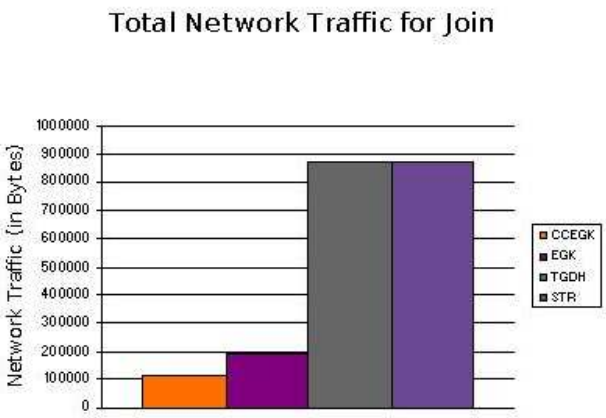

Figure. 7: Total Utilization Bandwidth for Join

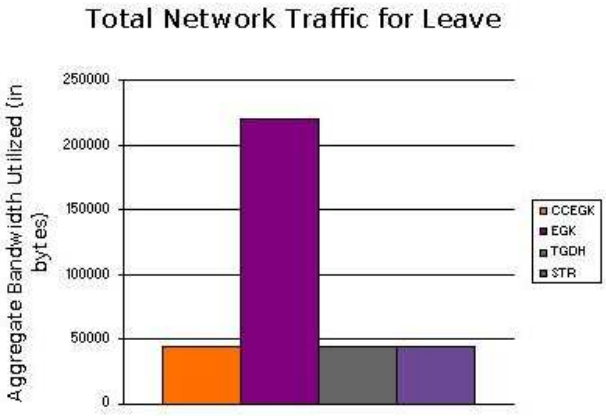

Figure. 8: Total Utilization Bandwidth for Leave

protocols. CCEGK is an evolution of EGK, as STR is an evolution of TGDH. Consequently, the lineage is quite apparent in the merge operation. Again, STR and TGDH take more time and consume more bandwidth (sending more information per packet than necessary; see Figures 13-14).
When looking at the total bandwidth utilized (Figure 12) we still see that CCEGK behaves identically to EGK, and TGDH behaves identically to STR in this experiment.
Additionally, the merge operation is a significant deviation from the theoretical results found in Zheng [16]. This is readily explained by the fact that the theoretical results were based upon a paper by Kim, Perrig, and Tsudik, published in 2000 [9]. However, this simulator uses data from a 2004 paper by Amir, Kim, Nita-Rotaru and Tsudik [2]. The published results for STR and TGDH differ greatly between these two papers which accounts for this discrepancy.

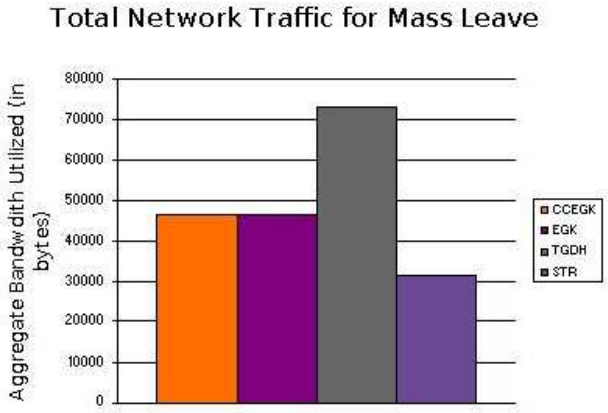

Figure. 9: Total Utilization Bandwidth for Mass Leave

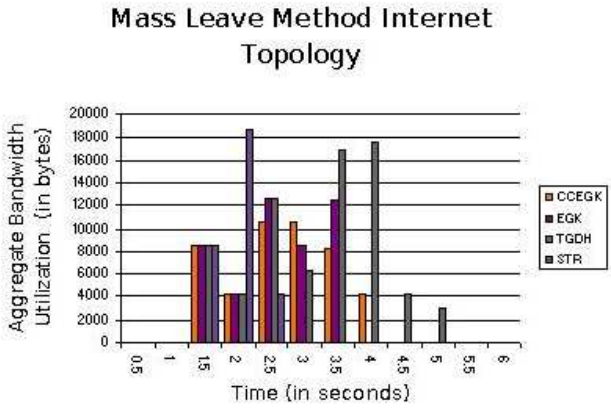

Figure. 11: Bandwidth Utilization for Mass Leave (Internet)

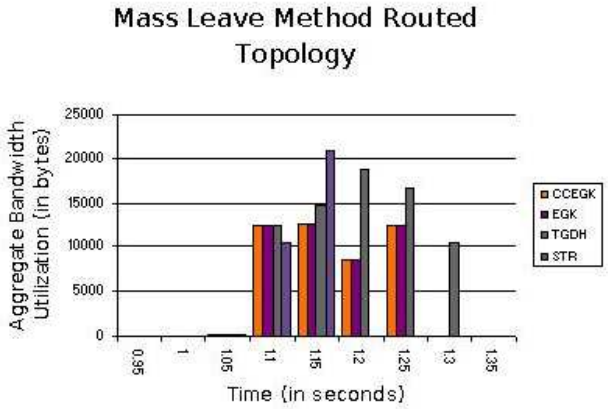

Figure. 10: Bandwidth Utilization for Mass Leave (Routed)

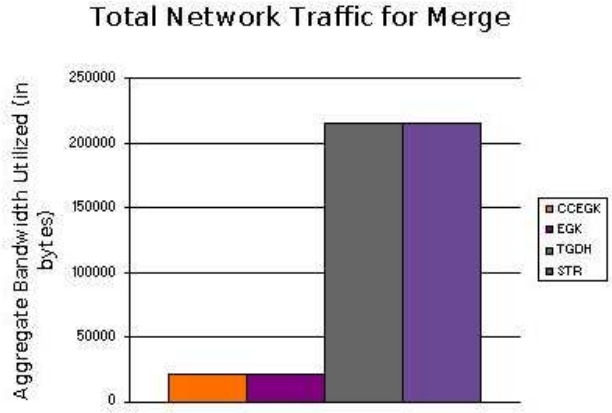

Figure. 12: Total Utilization Bandwidth for Merge

V. Simulation Results Part 2

Part 2 compares protocols for varying group sizes. The goal of Part 2 is to determine the realistic communication costs for two operations, join and mass leave, when the group size varies. Due to similarities between subnetwork and Internet topologies, we use only the subnetwork topology for these experiments. We simulate the total bandwidth utilized per protocol for groups of size 5, 10, and 30. Illustrating trends that are affected by group size and allows meaningful comparison between protocols.

A. Join

Figure 15 depicts the results of the three experiments. The three entries along the bottom are for a sequential add of three members to groups sized 5, 10, and 30, respectively. Starting with CCEGK and EGK, we notice a gradual increase in the bandwidth utilized as group size increases. Given that the same operation (adding one member) requires more traffic on larger groups, since the message must be sent to everyone in the group, the increase in cost is expected. The major change is in TGDH and STR, each exhibit a significant increase in cost for groups of size 30. This cost far exceeds a predicted linear triple increase in the cost for a group of size 10. Again this can be attributed directly to the increased number of recipients (and therefore more traffic on the network) but also an increased packet size, since these protocols send all keys to all members. This linear increase in packet size, in addition to the linear increase in the total number of traffic recipients, accounts for a considerable increase in overall bandwidth consumption. Therefore, while CCEGK and EGK have a linear increase in bandwidth utilization in relation to the increasing number of group members, STR and TGDH have a polynomial increase in costs due to the increasing group size and the increased packet payload.

We can create a formula for bandwidth utilized where we sum up all packets transmitted on the network. This is a function of the number of transmitters (senders) which broadcast to everyone in the group. Additionally, because each packet

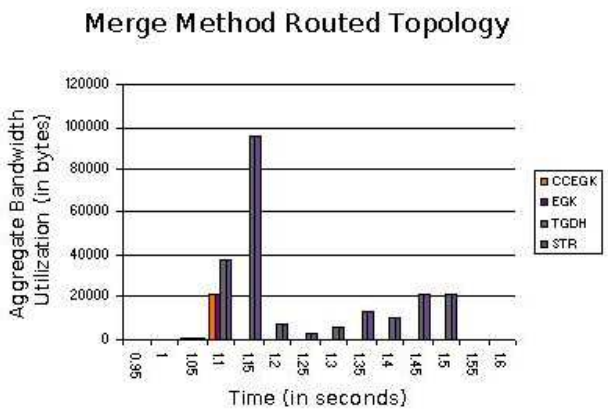

Figure. 13: Bandwidth Utilization for Merge (Routed)

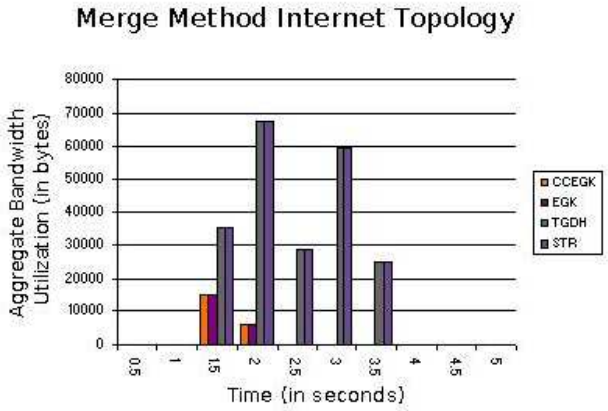

Figure. 14: Bandwidth Utilization for Merge Method

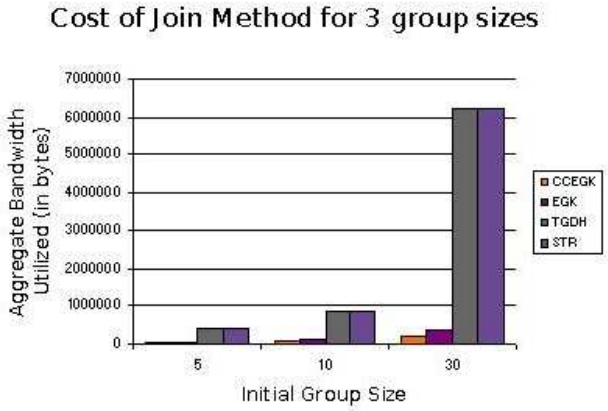

Figure. 15: Bandwidth Utilization for Join Method over three group sizes

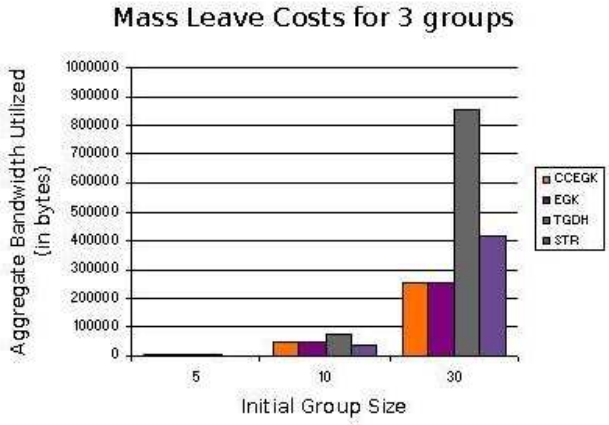

Figure. 16: Bandwidth Utilization for Mass Leave Method on three group sizes

has a payload of every key for every member, that too is a function of group size.
Bandwidth Utilized = total traffic =

$$\sum_{\#ofpackets}^{N} packetsize = N \cdot s$$

= (group size · # transmitters in group) · group size = $(groupsize)^3$

Note that $(groupsize)^3$ is the worst case where every person in the group is broadcasting to everyone else in the group.

B. Mass Leave

For the mass leave operation, we have three different initial group sizes 5, 10, and 30. We simultaneously remove three members from these initial groups to determine our costs. All of these experiments are run on the subnetwork topology.
Figure 16 depicts a graph that looks a bit like the Join Method graph for the three group sizes (Figure 15). While it is impossible to tell from this graph, STR is cheaper on the first group of size five. It maintains an advantage in the group of size 10, but becomes more expensive than CCEGK and EGK in the group of size 30. TGDH is more expensive than any other protocol. The reason for this considerable increase in cost is the same as that for the join operation. CCEGK, TGDH and EGK all perform the same operation for the mass leave operation; they simply rebuild the tree. While not cheap, this is an effective means for a mass leave. However, where CCEGK and EGK continue to have an edge is the constant packet payload; as group size increases, TGDH and STR are forced to not only send messages to more people (just as CCEGK and EGK do), but they also increase the packet size by including the keys for all group members in each message. This adds tremendous costs and is reflected in the graphs.

VI. Conclusions

We implemented group key management protocols in a realistic network topology, utilizing TCP/IP, and included previously unaccounted for packet size, which enabled us to firmly examine how CCEGK, EGK, STR, and TGDH perform in real-world simulations. This ground work provides insight into how group key management protocols perform in a practical networked environment, as opposed to the pen-and-paper theoretical frameworks previously published.

Having set out to determine if NS-2 was a feasible program to implement group key management protocols, we fully implemented four group key management protocols on two network topologies in our simulation. While NS-2 does have its drawbacks and idiosyncrasies, it is fully featured enough to provide the basics for network connectivity, and thus provides a capable platform upon which to simulate the protocols. Additionally, NS-2's extensibility into the wireless arena continues to make it an attractive choice.

We also discovered how great an effect the number of rounds in a given operation had. Because rounds are not a fixed size, but depend on the protocol and the network topology, the actual cost in time is not accurately reflected in the number of rounds. Timing is of great concern, and the number of rounds directly affects timing. When comparable protocols take three times longer to complete because of a reliance on previous rounds, it has a serious consequence on network performance.

We also discovered how time values are strongly linked to the underlying network topology and settings; the bandwidth utilization values are more network-independent but still rely on a few network variables. For example, if we had a faster network, the time values would drastically decrease, but the total amount of traffic would largely stay the same. This means that the time measurements are more fundamentally coupled with the network than the bandwidth utilization measurements, which are more directly related to the protocols.

We also see that for our test cases the protocols' bandwidth utilization converges somewhat. This is due to the fact that all of the measured protocols are partially based upon each other. It is unsurprising to find that STR, which is based upon TGDH, and CCEGK, which is based upon EGK, behave like their respective parents. This illustrates the point that protocols designed to perform the best in rare, worst-case scenarios might perform little better in more 'average' or usual situations. This is not to say that modifications for worst-case eventualities are unnecessary, rather that in a simulator attempting to model real network behavior, these extreme cases might not be evidenced.

Overall this research has revealed two main points:

- The number of rounds required greatly influences the total time for an operation to be completed.
- The number of full broadcasts (and messages with a large payload) greatly increases the total bandwidth utilization cost.

With the topology and the group key management simulator code from a companion technical report, any trained NS-2 user could implement new and further experiments to continue researching EGK, CCEGK, TGDH, and STR's behaviors on new network simulations. More work can be done to widen the scope of these simulations. Further experiments can give us a broader insight into how each protocol behaves. There are other variables that could be adjusted to see how they affect the simulator. For the sake of realism, we elected to use the usual defaults, but factors such as link noise (and link failure), multicasting, alternative routing, and droptail queuing could all be changed for additional experiments. However, the goal of our experiment simulator was to reflect the real-world conditions as closely as possible.

Furthermore, the entire field of group key management is entering into the exciting realm of wireless and ad-hoc communications. This simulator is uniquely poised to launch into this arena. The future focus will be on wireless networks, a field that NS-2 is well equipped to simulate and research. With the rise in popu larity of these new arenas, designers of group key management protocols can no longer ignore communication in favor of computation, or vice versa. In some environments, the power cost of communication may be sufficiently high to warrant low-cost communication protocols, whereas in other environments the computation cost may be the dominant feature. Coupled with a new ad-hoc CCEGK protocol, this extended simulator will provide an additional contribution to the field of group key management.

Acknowledgment

This material is based on research sponsored by AFRL and DARPA under agreement number F30602-02-1-0178 and the NSF SFS grants DUE-0114016 and DUE-0416757. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expresses or implied, of NSF, AFRL and DARPA or the U.S. Government.

References

[1] J. Alves-Foss. An efficient secure authenticated group key exchange algorithm for large and dynamic groups. In *Proc. 23^{rd} National Information Systems Security Conference*, pages 254–256, Oct. 2000.

[2] Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik. On the performance of group key agreement protocols. *ACM Transactions on Information and System Security*, 7(3):457–488, Aug. 2004.

[3] K. Becker and U Willie. Communication complexity of group key distribution. In *Proc. 5^{th} Conference on*

Computers and Communication Security, pages 1–6, 1998.

[4] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology - EUROCRYPT'94*, pages 275–286, May 1994.

[5] M. Burmester and Y. Desmedt. Efficient and secure conference key distribution. In *Security Protocols: International Workshop*, pages 119–129, Apr. 1996.

[6] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–652, Nov. 1976.

[7] A. Freier, P. Kartion, and P. Kocher. *The SSL Protocol: Version 3.0*. Netscape Communications, Inc., Mar. 1996.

[8] M. Just and S. Vaudenay. Authenticated multi-party key agreement. Technical Report SCS-TR-96-04, Carleton University, Computer Science Department, Ottowa, Canada, 1996.

[9] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerance key agreement for dynamic collaborative groups. In *Proc. of 7^{th} ACM Conference on Computer and Communications Security*, pages 235–244, Nov. 2000.

[10] Y. Kim, A. Perrig, and G. Tsudik. Communication-efficient group key agreement. In *Proc. of IFIP SEC 2001*, pages 229–244, Jun. 2001.

[11] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, 53(7):905–921, Jul. 2004.

[12] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, Feb. 2004.

[13] M. Steiner, G. Tsudik, and M. Waidner. Diffie- hellman key distribution extended to groups. In *Third ACM Conference on Computer and Communications Security*, pages 31–37. ACM Press, Mar. 1996.

[14] T.Ylonen. S s h– secure login connections over the internet. In *the Sixth USENIX Unix Security Symposium*, pages 37–42, Jul. 1996.

[15] S. Zheng, D. Manz, and J. Alves-Foss. A communication-computation efficient group key algorithm for large and dynamic groups. *Computer Networks*, 51(1):69–93, 2007.

[16] S. Zheng, D. Manz, J. Alves-Foss, and Y. Chen. Security and performance of group key agreement protocols. In *Proc. IASTED Networks and Communication Systems*, pages 321–327, Mar. 2006.

Author Biographies

David Manz is a Ph.D. student in the Department of Computer Science at the University of Idaho. During the summer of 2002 he participated in a Research Experience for Undergraduates at the University of Idaho. There he was exposed to security research, and decided that graduate school was for him. He finished up his B.S. degree from the Robert D. Clark Honors College at the University of Oregon in 2003. He completed his MS degree in 2005 at the University of Idaho, conducting research into simulations of group key agreement protocols.

Dr. Jim Alves-Foss earned a BS in physics, mathematics and computer science in 1987 from the University of California at Davis, California, USA. He earned his MS and PhD in computer science in 1989 and 1991 respectively, also from the University of California at Davis, California, USA. He is a professor of computer science at the University of Idaho, Moscow, Idaho USA and has been a faculty member there since 1991. He is currently the director of the University's Center for Secure and Dependable Systems. His work focuses on techniques for the design and analysis of secure systems.

Shanyu Zheng is currently a system analyst in a company in Virginia, U.S.A. He was a Ph.D student in the Department of computer science at the University of Idaho, USA, and was a research associate with the Center for Secure and Dependable Systems. His research interest includes cryptography, network security, and wireless systems, with an emphasis on group key management in wired and wireless networks. Before that, he received BS, MS in computer science from Sichuan University, Chengdu, Sichuan, China, in 1999, 2002, respectively.