AllNet: using Social Connections to Inform Traffic Prioritization and Resource Allocation

Edoardo Biagioni
Department of Information and Computer Sciences
University of Hawaii at Mānoa
Honolulu, Hawaii, USA
esb@hawaii.edu

***Abstract*—AllNet is a new networking protocol designed to provide communication utilizing all available means, including Internet and cellular communications, but when these are not available, also ad-hoc networking and delay-tolerant networking.**

These latter mechanisms are best for low-bandwidth communications. Effective support of low-bandwidth networking needs message prioritization, which can benefit by knowing whether messages are being sent on behalf of someone to whom the owner of the mobile device is socially connected. By keeping track of the social network of each of the friends of the owner of the mobile device, the device can devote its resources to supporting better quality communication among people its owner cares about, and fewer resources to communication among people its owner doesn't know.

AllNet generalizes this notion by anonymously keeping track of friends, friends of friends, friends of friends of friends, and so on. Doing this while using only limited communication and storage is the challenge addressed by the AllNet social network connectivity algorithm described and evaluated in this paper.

***Keywords*-social network; ad-hoc network; peer-to-peer network; p2p; backup communications; emergency communications;**

I. Introduction

AllNet is a peer-to-peer communication technology designed to provide connectivity among mobile and fixed devices whenever possible, even in the absence of networking infrastructure.

AllNet was born of the observation that every mobile device has a two-way radio that is capable of communicating directly with other mobile devices. Amazingly, at present this hardware capability is only infrequently utilized by available software and protocols. AllNet is designed to use this capability to provide pervasive communication whenever possible.

Applications that are easy to envision include using mobile devices as two-way multimedia radios (video walkie-talkies), sharing internet connections (which is beginning to be supported by "personal hotspots"), and emergency communications.

The design of AllNet includes several components. The Internet and other infrastructure-based forms of communication are used when available. When not available, ad-hoc networking allows AllNet communications to reach beyond the immediate radio range of the mobile device. As the data is being forwarded over intermediate devices, ubiquitous encryption protects the data from eavesdropping. Built-in prioritization allows each device participating in a communication to establish its own level of support, creating a community where messages that can be seen to require the fewest resources are most likely to be forwarded. To maximize the chance of intermediate systems being willing to forward messages, applications can be designed to send the least possible number of messages that are as small as possible.

On the other hand, whenever two systems are within range of each other, or when the communication has been requested by the owners of the devices involved, AllNet supports high bandwidth communication as might be used for a multimedia walkie-talkie.

When the Internet or other infrastructure is available, AllNet traffic can be carried over the infrastructure network. Whenever such infrastructure is not available or not accessible to the particular mobile device (for example, due to lack of a valid SIM card), AllNet provides a backup low-bandwidth means of communication. Because of this, AllNet can be useful in emergency situations and whenever mobile devices might otherwise be offline. To appreciate the benefits of low-bandwidth communication, it is sufficient to realize the impact of services such as SMS and Twitter, famously limited to 140 characters per message, yet extremely useful in many different situations.

To provide interpersonal communication while using as few resources as possible, each device forwarding AllNet traffic prioritizes all messages. The goal of the design of AllNet is to support at least low-bandwidth communication sufficient for the exchange of short text messages among individuals. This paper, after presenting the overall design of AllNet in Section II, explains in Sections III-IV how this prioritization can be useful in favoring individuals to whom one is most closely connected while giving lower quality service to individuals that are not closely related. By anonymously keeping track of who is related to whom, it is easy to see that ultimately everyone is connected to everyone else, and that supporting communication among strangers is a good way to build a community where everyone supports communication among other individuals.

AllNet uses public key encryption. The secure exchange of public keys in AllNet is described in Section II-B. Once two devices have each other's public key, they can securely exchange additional information, and in particular, information about their own friends' or contacts' network. In this way, each mobile device can accumulate information about the entire social network. This information allows mutual strangers to infer the distance of their relationship without identifying the individuals through which they are connected.

Information about the social network allows individuals to directly evaluate the benefits of supporting AllNet both to themselves, and to their social group. One goal of AllNet is to have individuals selfishly devote most of their resources to supporting communication that directly benefits the individual, but reserve a small fraction of their resources to generously supporting communication whose benefits to the individual are less immediate. In this way, AllNet provides information to help build a community to support pervasive and ubiquitous communication.

This paper includes in Sections III and IV an evaluation of the effectiveness of the AllNet social network prioritization. Specifically, by storing about a million keys, any device can establish its owner's relationship distance to just about anyone on the planet by exchanging about 2 MB. Random populations as described in Section IV show that a million individuals, each with 100 contacts, of which 10 are random and the other 90 are randomly selected from friends of friends, have an average distance between any two random individuals of less than 3.5, and over 100 samples, a worst-case average distance of about 3.8 between any two random individuals. As amply made clear by other research (e.g. [1]), this confirms that in general, individuals in a social network are likely to have fairly close connections to each other.

II. AllNet Design

AllNet is a networking technology which, like the early Internet, includes a few driving application. For the early Internet, the driving applications were telnet, ftp, and email. For AllNet, they are secure chat, internet access, and local multimedia walkie-talkie.

Unlike the early Internet, the design of AllNet assumes that many applications communicate by exchanging individual messages more than by sending ordered streams of bytes. Because a substantial fraction of messages might be lost, AllNet applications should be designed to be stateless and to exchange idempotent messages[1] whenever possible. Ideally, each message would be useful in isolation. For example in a chat application user messages can be delivered to the user as soon as they are received. If some prior messages have not been received (so far), the application can give the user an indication of this, while displaying whatever messages have been received.

When exchanges are brief, the overhead of a TCP three-way handshake can be large if the delay is substantial, as it can be in ad-hoc networks. The establishment of quick TCP sessions is out of the question in Delay Tolerant Networking (DTN), where data is physically carried to the destination by the movement of the mobile device.[2] It is for these reasons that AllNet applications generally use connectionless message transfer rather than connection-oriented byte streams.

Per-connection congestion control is an important function of TCP connections. Instead of TCP-style congestion control, AllNet performs data throttling and prioritizes individual messages based on their size, the distance they may still travel, and the source and destination addresses.

The transfer of data can occur over a variety of media, again as in the Internet. AllNet can use addresses from other protocols, including specifically IPv4 and IPv6 addresses, and also its own addresses. When no specific route to a destination is known, AllNet can use broadcasts limited to a specific number of hops. All else being equal, nodes forwarding a message will prioritize broadcast messages with a smaller number of hops over messages specifying a larger number of hops.

More details of message addressing and forwarding are given in a prior paper [3].

A. *Security and Pseudonymous Authentication*

One of the concerns when sending sensitive data over a wireless ad-hoc network is the easy access to the data by other devices controlled by attackers who might want to eavesdrop on a conversation or inject malicious messages.

To prevent these attacks, AllNet signs and encrypts all personal messages using public key encryption.[3] In public-key encryption, the public key k_{pub} used to encrypt messages and to verify signatures is different from the private or secret key k_{sec} used to decrypt messages and to create signatures. The public key can be known to adversaries without compromising the security of the cypher. The only known way of attacking a secure Public Key Cryptosystem (PKS) is for the attacker to convince one of the parties to use a public key for which the attacker knows the corresponding secret key. It is for this reason that secure HTTP requires certificates signed by a trusted Certificate Authority. These certificates confirm that the public key provided by a secure web site indeed belongs to that web site.

In a decentralized, peer-to-peer network such as AllNet, every user should certify his or her personal friends and other contacts rather than rely on centralized certificate authorities. This is a social model of identity, which does not offer the same guarantees as provided by Certificate Authorities, but is sufficient for many purposes.

Since friends and contacts often meet in person or have other reasonably secure ways of exchanging private information, AllNet leverages this out-of-band information exchange to securely exchange public keys. The algorithm for secure key exchange is described in detail in Section II-B.

[1]A message is *idempotent* if it has the same meaning no matter where in the sequence of messages it is received. An application is *stateless* if its behavior does not depend on the sequence with which messages have been received. An early stateless protocol using idempotent messages is the Network File System, NFS [2].

[2]In this case the mobile device is known as a *data mule*.

[3]The current implementation of AllNet uses RSA public key encryption. Other public key encryption algorithms can be used without changing the AllNet protocol.

TABLE I
SYMBOLS USED IN SECTION II-B, SECURE KEY EXCHANGE

Symbol	Meaning
s	secret string, known only to Alice and Bob
k_A	Alice's public key
k_B	Bob's public key
N	nonce
H_A	hash of (s, k_A, N)

While in general trust may or may not be transitive depending on the context [4] [5], if two people wish to communicate with each other and do not have any other way of securely exchanging information, a trusted third party can help with the initial key exchange.

The keys used for communication between Alice and Bob need not be the same as the keys used for communication between Bob and Charlie, or between Alice and Charlie. The current design of AllNet uses a separate pair of public and private keys for every pair of users. Using 4,096-bit (512-byte) keys, if a user keeps keys for 10,000 contacts, the storage required for the keys is on the order of 10MB or less, which is very little on current mobile devices. And probably most devices will keep far fewer than 10,000 keys.

B. Secure Key Exchange in AllNet

Secure exchange of public keys over an insecure medium is relatively straightforward if the two parties share an unguessable secret. Techniques used for connecting Bluetooth devices to cellphones and mobile devices to secure wireless access points are not dramatically different from the algorithm described in this section.

Alice and Bob share the same unguessable string s. Alice wishes to send her public key k_A to Bob, who wishes to send his public key k_B to Alice. Since public keys can only be used to verify signatures and encrypt messages, knowledge of a public key does not allow an attacker to read any confidential information, and the exchange proceeds in the clear.

The algorithm used by AllNet is based on zero-knowledge cryptographic proofs used to exchange keys, but is considerably simpler than most schemes [6] [7]. This is because it only needs to securely exchange public keys, and the public keys themselves do not need to be kept secret.

Alice computes a hash $H_A = hash(s, k_A, N)$, where N is a nonce. She then sends to Bob (H_A, k_A, N). Any attacker that does not have access to s is unable to generate a message where H corresponds to k_A and N. Bob can then compute his own hash $H_B = hash(s, k_B, N)$, and sends to Alice (H_B, k_B, N).[4]

Without knowing the string s, Charlie cannot create a message (H_C, k_C, N) such that Alice will be fooled into believing that k_C is Bob's public key, because she can compute the hash $H'_C = hash(s, k_C, N)$ and note that $H'_C \neq H_C$.

[4]The nonces sent by Alice and by Bob may be the same or may be different. The main purpose of the nonce is to ensure that, in the event the same key is sent with the same secret at different times, it will produce a different hash.

To do a brute-force attack, Charlie must guess the string s and send Alice a message for each guess of s. If, on average, $\hat{g}$ guesses are needed before Charlie can guess s, Bob only needs for his message to arrive before the $\hat{g}$'s message sent from Charlie.

In practice, AllNet requires that a key exchange message created with the correct secret string be received within a small number of tries, currently 100.

The keys k_A and k_B exchanged in this manner are only used for communication between Alice and Bob. If Alice uses her private key to sign each message sent to Bob, Bob can have confidence that as long as Alice's keys have not been compromised, the message is indeed from Alice.

In this exchange, nowhere did Alice or Bob exchange their names or other identifying information,[5] other than the shared secret string s. That means Bob must use other factors (external to AllNet) to establish the identity of Alice. It is in this sense that authentication in AllNet is pseudonymous. Alice and Bob could exchange many different key pairs and use them interchangeably. In other words, and as long as keys are not compromised, there is at most one individual who has access to a given private key, but each individual typically uses many private keys.

This kind of authentication is powerful. For example, Alice may use her private keys to sign and send a message to some of her friends letting each of them know that from now on, she will use a different key.

C. Using AllNet Public Keys for Identification

Because Alice is likely to use a different key pair to encrypt messages to each of her friends, there is no way her friends can identify Alice to each other. However, this is sometimes desirable. In particular, having a token to identify each of a person's friends will be used in the next section to establish the relationship distance between individuals who otherwise would consider each other strangers.

The design of AllNet therefore includes one or more public/private key pairs used only for authentication. Multiple key pairs can be useful if a person wishes to have different identities, for example in different groups, but the remainder of this paper discusses the case of one key per person used for identification. In the design of AllNet, the public key k_{id} is securely distributed to each new contact, and updates of k_{id} are sent to all contacts whenever k_{id} changes.

III. ALLNET SOCIAL NETWORK CONNECTIVITY

AllNet is designed to be generally useful, but its usefulness depends on wide participation, that is, on having a large number of devices support AllNet and carry traffic for others. Motivations for supporting AllNet were explored in a prior paper [8], and range from selfish to altruistic.

This section explores the AllNet Social Network Connectivity Algorithm (ASNCA), a mechanism to provide each device and each user with additional information that might

[5]In practice, Alice and Bob might securely exchange profile information as one of the first steps after completing the key exchange.

TABLE II
SYMBOLS USED IN SECTION III, SOCIAL NETWORK

Symbol	Meaning
f, f^1	friends' keys
f^2	friends' friends' keys
f^i, $i > 1$	keys of the friends of the f^{i-1} group
$\|f^i\|$	number of keys in f^i
k_i	number of bits saved for keys in f_i
b	number of bits in the Bloom filter
n	number of keys (1 bits) in the Bloom filter
k	number of bits in each potentially matching hash
m_i	size of Alice's f_i

encourage users to consider perfect strangers as not-so-distant relations. In doing so, we hope that users might be even further incentivated to support AllNet, not only to build an abstract community, but also for the direct benefits that might come from supporting communication among people to whom the individual might have some measurable connection.

A. AllNet Social Network Connectivity Algorithm (ASNCA)

I store $|f|$ friends' public keys. Each key has k_1 bits (typically $k_1 \leq$ 4,096). Each of my friends also gives me the first k_2 bits of all of their friends' public keys (maybe $k_2 = 128$), k_3 bits of the keys of their friends' friends (maybe $k_3 = 64$).

If everyone has $|f| = 100$ $f$riends and contacts, in a random world $|f^2| \approx$ 10,000, $|f^3| \approx$ 1M, $|f^4| \approx$ 100M, and so on.

The world is not random, so a reasonable goal is to limit the number of keys stored, for example to about 1 million keys. At 64 bits for most of the keys, this requires 8MB of storage. If everyone has a million keys, and if the total number of keys is 10 billion,[6] then the probability of any one of my keys matching (at random) one of someone else's million keys is 1million/10billion = 1/10000. The probablility that at least one of my million keys matches one of the keys stored by someone else is then close to 1. That means it is very likely that for almost everyone on the planet, I will be able to determine the distance of their relationship to me, even though my device only stores a million keys.

ASNCA has everyone store the keys of their f^1, the first 128 bits of the keys of their f^2, and the first 64 bits of the keys of their f^3, up to one million keys or whatever number of keys is appropriate for the technology at hand. If $\Sigma_i f^i$ is small compared to the available space, devices can store f^4, f^5, and so on, but the remainder of this discussion assumes that each node stores only up to a million keys, and only in $f^1 \ldots f^3$.

Devices running AllNet try to provide at least a minimal level of service to all, but should give preferential service to people connected to the owner of the device. The AllNet Social Network Connectivity Algorithm (ASNCA) is designed to establish the relationship distance among individuals who have never directly communicated before. Specifically, when I wish to obtain better service from a device belonging to Alice, Alice would like to be able to reliably establish whether I am in her f^2, f^3, or more generally, the smallest i for which I am in her f^i.[7] Mathematically, this is only true if for some $j < i$, one of the people in my set f^j is in Alice's f^{i-j}.

Alice needs to establish a value of i. However, I do not wish to send in the clear either my k_{id} (defined Section II-C) or anyone else's k_{id}, since an eavesdropper (or even Alice, should she turn out to be malicious) could attempt to use these k_{id}'s to impersonate me or my friends.

At the beginning of the exchange, Alice sends me a nonce N. In response, I concatenate the nonce with each of the $|f|$ keys of of my friends and compute each hash $h = (N, k_{id})$. I use the $|f|$ hashes to build a Bloom filter with b bits, where $b > |f|$. I repeat the operation twice, once using k_2 bits of each friend's key, then k_3 bits ($k_1 > k_2 > k_3$). I send Alice the three Bloom filters.

Alice must perform the same hash $h = (N, k_{id})$ for each of the keys that she holds. This explains why I must hash my friends' keys using Alice's nonce – Alice can send me a nonce that is easy for her to use, for example, one of several nonces for which she precomputed the hashes at some earlier time when power was abundant.

For each of her friends' keys, Alice must do the hash computation three times, once using k_1 bits of the key (all the bits of the key), once using k_2 bits (128 bits), and once using k_3 bits (64 bits). For each key of the friends of Alice's friends, the computation must be done twice, and for each of the keys belonging to Alice's friends' friends' friends, the hash must only be computed once using k_3 bits.

If any of the bits in at least one the Bloom filters corresponds to any of Alice's hashes, Alice has tentatively identified one or more k_{id} that might match one or more of my friends. She then sends me the matching Bloom filter, the matching bits, and the k_i to use. To confirm that the keys indeed match, I must hash the nonce concatenated with k_i bits of each of the keys corresponding to the matching bits, and return the hashes to Alice.

If there is a match, Alice can confirm that I hold k_i bits of the key and thereby convince herself that I am in her f^{i+1}.

At this point, Alice and I share a secret (k_i bits of the matching key), and can use this secret to exchange a new key.

However, if there is no match, none of my friends is in Alice's $f^1 \ldots f^3$, so I am not in Alice's f^i for any $i \leq 4$.

I may then repeat the operation by sending her two Bloom filters for my f^2, and again if necessary, send her one Bloom Filter for my f^3. This should allow us to determine whether I am in Alice's f^i for all $i \leq 6$.

In the above example of a random world where everyone has 100 friends, $|f^6| \approx 100^6 = 10^{12}$, which is over 100 times more than the current population of the Earth.

[6]By the time, if ever, that the population of the planet exceeds 10 billion, it is likely that cheap mobile devices will have more bandwidth and much more storage space than at present.

[7]Since Alice and I have never yet had any contact through AllNet, I cannot be in Alice's f set.

B. Minimizing the Size of the Exchange

Frequently, when I wish to obtain a service from Alice, my mobile device will either be directly in range of Alice's mobile device, or both devices are likely to be connected to the Internet. In both of these cases, restricting bandwidth usage is not likely to be critical. However, it is interesting to consider how the ASNCA exchange can be optimized by minimizing the number of bits sent and received.

When I first connect to Alice to initiate the ASNCA exchange, I must send her Bloom filters corresponding to my friends' keys. For simplicity, this analysis only considers the set f^3, which in most cases will be much larger than the other sets, and has size $n = |f_3|$. The size of the Bloom filter for f^3 is $b > n$ bits. Assuming that keys are randomly distributed, each key that Alice holds will match a bit in my Bloom filter with probability n/b. If Alice has m_i keys at level i, the expected number of matches between my Bloom filter and Alice's keys at her level i is $n \times m_i/b$.

I want to choose b to minimize the total number of bits exchanged.

The number of bits I send Alice is the number of bits b in the Bloom filter.

The information that Alice sends to me is the list of bits in my Bloom filter that match her keys, at each level. In response, I will send Alice a hash for each match. If the total number of bits that Alice and I exchange for each match is k (the value for k is derived in the next two paragraphs), then the expected number of bits that Alice and I exchange for all the matches is $k \times n \times m_i/b$.

The information that Alice sends to me for each match is the list of bits in the Bloom filter that match her keys at level i. This can be encoded using $log_2 n$ bits for each match.

The hash that I send to Alice for each match must have at least as many bits as the smaller of the key that I have stored, or that Alice has stored, since additional bits do not give any additional information. If Alice stores k_i bits of the key and I store k_j bits of the key, then I need to send Alice $min(k_i, k_j)$ bits for each hash. The value of k is then $k = log_2 n + min(k_i, k_j)$.

Including the original Bloom filter and the hashes, the expected number of bits Alice and I exchange is:

$$\hat{bits} = b + k \times n \times m_i/b \qquad (1)$$

m_i is unknown until the exchange is complete (we don't know which of Alice's levels, if any, will contain the key of one of my friends), but if the number of stored keys is limited to one million, $m_i \leq$ 1,000,000.

I can differentiate equation 1 with respect to b and set the differential to 0 to find the optimal b to minimize the total number of bits sent.

$$\frac{d\hat{bits}}{db} = 1 - \frac{k \times n \times m_i}{b^2} = 0$$

$$b^2 = k \times n \times m_i$$

$$b = \sqrt{k \times n \times m_i} \qquad (2)$$

Using m_i = 1,000,000 I choose the nearest b_i that is a power of 2.

For example, if I have about 1,000,000 keys in my f^3 and $k_3 = 64$ bits, $b = \sqrt{64 \times 10^6 \times 10^6} = 8 \times 10^6$ or 8 million bits.[8] The nearest power of two is 2^{23} bits or 1,048,576 bytes.

The likelihood of any one of Alice's 1,000,000 keys matching a bit in my Bloom filter is about $1/8$, since $1/8$ of the bits in the Bloom filter are set. The expected number of matches is 125,000. For each of these matches I must send a 64-bit (8-byte) hash, amounting to 1,000,000 bytes. The total amount of data I send is then 2,048,576 bytes.

That this is optimal can be verified by looking at the next lower power of two, using 2^{22} bits in my Bloom filter, which leads to an expected 250,000 matches and my sending Alice 2,524,288 bytes. The next higher power of two, 2^{24}, only has an expected 62,500 matches whose hashes can be sent with 524,288 bytes, but the Bloom filter has 2,097,152 bytes, and the total data I send is 2,621,440 bytes. Both of these are higher than the 2,048,576 bytes I must send when the Bloom filter has 2^{23} bits, so from this example we see that $b = \sqrt{k \times n \times m_i}$ is indeed optimal.

IV. Analysis

A. Comparison of ASNCA to a Simpler Scheme

ASNCA is a little complex, and it is tempting to say that I should just send Alice 64 bits of the hash of the first 64 bits for each of the keys in my $f^1 \dots f^3$. Alice can then compare these hashes to hashes of her own keys.

Unfortunately, this simpler scheme is dramatically less efficient. If I send one million 64-bit (8-byte) hashes, this means sending 8MB, which is four times larger than the example using ASNCA.

B. Security Analysis of ASNCA

When trying to establish a connection with ASNCA, I will send to Alice some information about my social network. However, I know nothing about Alice, and in fact, she might be an attacker trying to obtain information from me rather than somebody truthfully offering me a service. Also, since the information is not encrypted, any eavesdropper or man-in-the-middle attacker might be able to get the same information. It is therefore appropriate to clearly identify what information is and is not sent.

No keys are sent in ASNCA, only key hashes.

Therefore, and as long as the hash functions are cryptographically secure, an attacker cannot identify the keys in my social network unless the attacker has somehow already obtained the keys.

The main concern is then that an attacker who already has one of the keys in my f^i will be able to tell that I have

[8]k is actually $k_3 + \log_2 1,000,000 = 84$. This is close enough to 64 (about 15% after taking the square root) that for simplicity for this example we use $k = 64$.

the same key. While this might pose some issues, it is also the main point of ASNCA that someone from whom I am legitimately seeking a service should be able to tell what our degree of relationship is, and therefore that we have received the same key from a person and device to which we are more or less closely connected. So it is necessary for Alice to be able to identify the keys I hold and that she also has.

There are some ways to lessen the amount of information disseminated.

In variant 1, I only send to Alice information about my immediate friends, which is a much smaller set than f^3, so an attacker is less likely to have accumulated any of those keys. If I do this, I have a correspondingly smaller chance of being found in Alice's $f^1 \dots f^4$ than in Alice's $f^1 \dots f^6$, but perhaps the extra privacy justifies giving up any benefits from being identified in Alice's f^5 or f^6.

In variant 2, I only send Alice the hash of my own public key, giving an attacker a chance to figure out who I might be, but no information about my social network. Then, Alice will only grant me higher priority if I am in her $f^1 \dots f^3$.

Variant 3 assumes that the information about keys in my f^3 holds little interest to an attacker. After all, there are probably few cases where an attacker would benefit from knowing who my friends' friends' friends are. Since I usually do not know myself who these people are, I may not mind the potential loss of privacy.

Variant 3 has the benefit that one of these keys is very likely to be found in Alice's $f^1 \dots f^3$, and Alice is therefore likely to grant me priority she would grant to anyone in her $f^4 \dots f^6$.

In variant 4, I send to Alice all my keys, in $f^1 \dots f^3$, but all together and only hashing k_3 bits, as if all these keys were in my f^3. Then, an attacker who finds a matching key does not know whether it is a key for one of my friends, one of their friends, or one of my friends' friends' friends.

Variant 5 combines variants 2 and 4. First I send Alice the hash of my own key, which Alice can use to figure out if I am in $f^1 \dots f^3$. If I am not, I can use variant 4 and send Alice all my other keys, allowing Alice to figure out if I am in her $f^4 \dots f^6$.

Because all of these approaches have advantages and disadvantages, the eventual decision of which to use can be left for the user. Most users will probably neither know nor care enough to make a decision, but those who do can choose whatever suits them best.

C. Connectivity

Results in the theory of random graphs and analysis of realistic social networks and on the web [1] indicate that both theoretical and real networks with at least some random connections have fairly low distance between any two nodes.

To confirm this, the author wrote a simple program to assign to each node in a graph 100 connections. Of these 100 connections, the first ten are selected at random. The remaining 90 connections are established to friends of friends: nodes selected at random from those connected to random nodes to which this node already has a connection.

Out of 1,000,000 nodes, 100 were randomly selected and distances to every other node in the network were computed using Dijkstra's shortest path algorithm. The average of the distances from each of these 100 nodes to every one of the million nodes in the network was 3.4. Over the 100 nodes analyzed, the largest average distance was 3.8. The largest distance from any node analyzed to any other node in the network was 4, and no nodes were disconnected from the network.

With 3,000,000 nodes, the results were similar: the average of the distances was 3.74, the largest average distance was 3.94, and largest distance overall was 5, again with no nodes disconnected from each node being analyzed.

These numbers not only confirm others' theoretical and experimental results, but indicate that keeping track of connections up to the sixth degree, as is done by the ASNCA algorithm, is an effective way of establishing connection distance to a potentially very large number of people. This is true even though in this program the number of random links is small, and most of each person's contacts are taken from their existing contacts.

V. AllNet Prioritization Algorithm

There is no requirement that a device use a specific AllNet prioritization algorithm, since each device may set its own priorities for message forwarding. However, it is useful to have an algorithm to use by default whenever no other particular priorities are defined.

The AllNet prioritization algorithm should be configurable by individual users, but have reasonable defaults. The algorithm is therefore parametrized by constants K_i that default to values D_i.

The prioritization algorithm assigns a value between 0 and 1 to each outgoing message, with 1 being highest priority. Messages are queued and sent in order of priority. The priority of outgoing messages is also used by AllNet to determine the amount of resources to devote to message transmission. In other words, more bandwidth and energy may be used to send higher priority messages than lower priority messages. However, even messages with a priority of zero may be forwarded eventually, if there is no competing traffic and resources are sufficient.

The inputs to the AllNet prioritization algorithm include the closeness of the relationship of the immediate forwarder of the data and of the final recipient of the data (distance d), the size of the message (s), the number of messages received from the same device in the recent past (weighted average rate $\bar{r}$), the expected number of further transmissions a message will require (remaining hops h), and the number of recipients that will need to attempt to decrypt a message (inversely proportional to the number of bits of the recipient's key in the destination address, bk).

The algorithm computes a social desirability factor sd that reflects the benefit, to the owner of the device, of forwarding the message. The algorithm also computes a message cost factor cf, proportional to the resources needed to forward the

TABLE III
SYMBOLS USED IN SECTION V, PRIORITIZATION

Symbol	Meaning
K_i	parameter of message prioritization
D_i	default value for K_i
d	social distance to the owner of another device, in hops
s	message size, in bytes
s_0	size of the smallest encrypted message, 1,024 bytes
h	number of hops remaining in the message header
bk	number of bits of destination ID in the destination address
sd	social benefit to the owner of forwarding a message
cf	overall resources needed to forward a message
$cost_{dev}$	resources this device will use to forward a message
$cost_{net}$	resources the network may use to forward a message

message both on this device, and also throughout the network. These two factors also have values between between 0 and 1, and the overall priority of the message is the product of the two: $priority = sd \times cf$.

The social desirability factor sd should be highest if either the original sender or ultimate receiver of the message is the owner of the device, still substantial for friends of the owner, and monotonically lower for people at greater social distance. For simplicity, only the lower distance of the sender or the receiver is used as the value d. To convert distance to a cost factor, AllNet raises a base K_1 to the exponent d, so that the owner of the device always has $sd = 1$, messages from the owner's friends have $sd = K_1$, their friends' messages have $sd = K_1^2$, and so on. With the default value $K_1 = D_1 = 0.7$, the owner's friends have $sd = 0.7$, their friends $sd = 0.49$, friends farther away $sd = 0.34$, $sd = 0.24$, $sd = 0.17$, and $sd = 0.12$, and finally total strangers have $sd = 0.08$.

The cost factor cf includes the cost to this device to forward the message, $cost_{dev}$, and the projected cost for other devices in the network to forward the message, $cost_{net}$. The two are mediated by another constant K_2, such that the overall cost factor is $cf = cost_{dev} \times K_2 + cost_{net} \times (1 - K_2)$. The default value for K_2 is $K_2 = D_2 = 0.5$, giving equal weight to both the individual cost, which can be measured fairly accurately, and the network cost, which, though potentially much larger, is only an estimate, and subject to other devices' determination of message priority.

Although forwarding a message is a complex operation, the AllNet prioritization algorithm assumes that the cost for this device to forward a message is simply inversely proportional to the size of the message. Since most AllNet traffic is encrypted, a cost factor of 1 is given to messages carrying a single encrypted payload of size $s_0 = 1,024$ and to any smaller messages.[9] Larger messages of size s have a cost factor $cost_{dev} = s_0/s$.

Estimating the cost to the network of forwarding a message takes into account information from the AllNet forwarding header, specifically the number of hops remaining and the number of bits of destination address specified by the sender. The latter is a mechanism to allow a sender and a receiver to communicate without intermediate hosts being able to perform traffic analysis. The sender specifies a variable number of bits, and any recipient whose keys match the specified bits is invited to decode the message. If bk bits are specified, approximately 2^{-bk} of all keys will match. So, the more bits are specified, the fewer recipients will need to decrypt a message.

[9]A message encrypted and signed with 4,096-bit RSA keys requires $1,024$ bytes.

Conversely, the more hops h a header specifies, the more resources the network will need to forward the message, and the more recipients the message is likely to reach. The number of recipients may grow exponentially with the number of hops, so the cost factor should grow in proportion.

Using these considerations, and acknowledging the imperfection of this formula in predicting the true cost for the network to forward the packet, we use $cost_{net} = \sqrt[bk]{K_3}/K_4^h$. Normally, $K_3 = D_3 = 0.5$, and $K_4 = D_4 = 2$, giving $cost_{net} = \sqrt[bk]{0.5} \times 2^{-h}$.

The overall formula for the priority of a message whose source or destination has a social distance d, with size s, showing bk bits of destination address and requesting to be forwarded h hops, is:

$$priority = K_1^d \times (K_2 \times s_0/s + (1 - K_2) \times \sqrt[bk]{K_3}/K_4^h)$$

Using default values, the formula is

$$priority = 0.7^d \times (0.5 \times 1,024 bytes/s + 0.5 \times \sqrt[bk]{0.5}/2^h)$$

While the computation of $\sqrt[bk]{K_3}$ might be somewhat CPU-intensive, the values can be pre-computed at the start of program execution for the first several values of bk. Should it be desired to avoid floating-point computations, all values can be scaled by a suitably large number N (e.g. 10^9 if 32-bit integers are to be used) and fixed-point computations performed instead of floating-point computation, yielding a priority between 0 and N.

VI. APPLICATIONS OF THE ALLNET SOCIAL NETWORK

AllNet was originally conceived as a network that could be used to convey information in times of emergency, when the infrastructure might be damaged but mobile devices might still be functional. In order to train users to be prepared to use AllNet should an emergency occur, the network should be useful on a daily basis.

From the existence of SMS, Twitter, and similar services, it seems likely that even exchanging short text messages will have broad usefulness. Unlike all these systems, a peer-to-peer system without a central point of failure or control might be useful in many cases, so the first application for AllNet is a chat system. Using 4,096-bit RSA keys, messages can be almost 500 bytes long, which is longer than either SMS or Twitter. Messages are persistent, and any two keys and a sequence number uniquely identify a message.

The chat system uses the AllNet network, but keeps track of the social network that can inform AllNet and help it prioritize messages.

The next obvious application of AllNet is web access. While this has yet to be be designed in detail, the basic idea is to provide at least bare-bones access to information on the web using the AllNet protocol. More advanced access would use the authentication capabilities of AllNet in place of the existing common use of user names and accounts. A new user registering on a server might send a new public key to that server over a connection authenticated using current technology.

The benefits of using AllNet for web access are the AllNet design and software for key management, and accessing the web even in situations where one would otherwise be disconnected. The challenges in this case consist mostly in adapting the message-oriented style of AllNet communication to the connection-oriented nature of web traffic, and in being able to establish secure connections with web sites without having to trust any intermediate host.

A third application of AllNet that might have substantial benefits is in distributing and finding out local information. Currently, this is done by searching centralized, global databases where location is one of the attributes to search on. Major search engines recognize the value of this information, and businesses may pay search engines to appear in the results of such searches. However, with AllNet it is straightforward for local businesses to do a local broadcast of information about themselves, or for searchers to locally broadcast requests for information. This exchange will normally be in the clear, though encryption may be appropriate when the information is only meant for a set of people that is known to the sender.

Since AllNet automatically identifies messages from persons of interest, any messages from others can be automatically ignored when the user is not searching among broadcast information.

The benefits of having local control over the broadcast of local information are many, and include the ability to update the information quickly according to local circumstances.

VII. Related Work

The design of AllNet builds on a number of independent areas of research in computer networks. Chief among these are ad-hoc networks and peer-to-peer (P2P) networks, especially P2P social networks and secure P2P networks. Emergency Communication Networks have been popular ever since the early days of Amateur radio, and several are in current use or have been recently proposed. This section provides a necessarily brief introduction to these fields of research.

While there has been much research on Quality of Service and more generally traffic prioritization, this section does not attempt to summarize the state of the art, and instead highlights the advantages of the approach taken in the design of AllNet.

A. Peer-to-Peer Social Networks

In recent years several proposals have been made for Peer-to-Peer (P2P) social networks. These include PeerSoN [9], Safebook [10], and PeerBook [11], though the last project no longer seems to be maintained. The very first secure P2P network was probably Phil Zimmerman's PGP [12], which provides security encryption for email messages. PGP introduced the idea of a web of trust based on social relationships. The major disadvantage of PGP is the practical difficulty of securely exchanging public keys, a difficulty that AllNet has addressed.

PeerSoN and Safebook appear to be progressing as good P2P social networks providing privacy and addressing many of the issues needed to provide a secure P2P social network. However, unlike AllNet, the focus of these project is on providing a distributed social network rather than a networking technology that will provide useful services whenever possible.

Freenet [13] is another network technology focused on secure data exchange among peers. Freenet is really designed for systems that are normally connected to the Internet rather than for systems whose connectivity changes frequently and is often missing.

Work in reputation-based systems (for example, [14]), to reward socially constructive behavior and discourage socially damaging behavior, is complementary to ASNCA.

B. Emergency Communication Networks

The cellular infrastructure as a whole is highly reliable, and the Internet only slightly less so. Both are often extremely useful in emergencies. On the other hand, disasters do affect existing communication infrastructure, and much research and development has gone into quickly providing communication services where necessary.

The Trilogy Emergency Response Application (TERA [15]) network developed for the International Federation of Red Cross and Red Crescent and deployed after the earthquake in Haiti. TERA uses cellular infrastructure to communicate with mobile devices, and emphasizes SMS (text message) communications as being the most efficient and effective way of exchanging and disseminating information.

Much research and development has gone into different ad-hoc networks for emergency communications, including the Byzantium Linux distribution [16], ECCA [17], and LifeNet [18]. In general, these systems share some of the goals of AllNet, but are much more focused on emergency communications. In contrast, one of the principles in the design of AllNet is that people should use in their daily lives systems such as AllNet that will also provide service during emergencies. If the system is in daily use, it is already deployed prior to the disaster or emergency, and people are already trained to use the system.

C. AllNet Secure Ad-Hoc Networking

There has been much research and a plethora of papers on securing ad-hoc wireless networks, too much to cover here. Instead, this section briefly reviews the technology used in securing communications in AllNet.

AllNet currently uses RSA public key encryption [19] [20] with 4,096-bit (512-byte) keys. However, AllNet does not require a standard cryptosystem. Any recipient of a message can attempt to decrypt and verify the message, and if either the decryption or the verification fail, discard the message.

Therefore, as long as a sender and one or more receivers agree on which cryptosystem to use, any encryption can be used with AllNet. This includes all existing and future public-key and symmetric-key algorithms where the key must be shared by the sender and the receiver.

Although the overall design of AllNet is independent of any specific hash function, standard AllNet operations described in this paper, including key exchange and ASNCA, do require the choice of a hash function. For AllNet, we have chosen SHA512 [21] as being the most future-proof hash function currently available.

D. Traffic Prioritization in Networks

AllNet is not designed to provide specific performance or QoS to a communication stream. Instead, the design goal for AllNet is to determine, based on message characteristics, what resources to devote to transmitting a message, and in what priority order it ought to be sent. AllNet uses social network connectivity to attempt to infer how many resources the owner of the device would wish to devote to the transmission of the message, and properties of the message to determine how many actual resources to use.

VIII. Conclusion

The idea of AllNet, free ubiquitous connectivity without the need for new hardware, has been an inspiration to a number of individuals. An implementation of AllNet is consequently in process. Work is continuing and up-to-date information is available on the project web site, `http://alnt.org`, but the current state of the implementation is described here.

Although plans call for AllNet to be implemented on all major mobile platforms, until now, AllNet has only been implemented for Linux. The current implementation includes two servers, `allnetd` and `wtxrxd`, and a chat program. The chat program provides a simple textual interface that supports message and key exchange, selection of who to chat with, and secure identification of incoming messages. The `allnetd` server implements the bulk of the AllNet protocol, and is designed to run with the least possible privilege on the system. On Linux, that means running as the `null` user in a chroot jail. ASNCA and the priority algorithm are not yet implemented, but the basics of peer-to-peer networking, including limited broadcasting of messages, keeping track of peers, bootstrapping, and persistent storage of messages are in place and have gone through at least preliminary testing and deployment.

In contrast, the `wtxrxd` server (the name stands for "Wireless Transmit Receive Daemon) is designed to run with enough privilege that it may send and receive broadcast messages on the networks to which the device can connect, to bring up wireless interfaces as needed for AllNet, and to bring them back down whenever possible to conserve energy. Because this server runs in privileged mode, it is as simple as possible.

The AllNet key exchange algorithm and data encryption, signing, decryption and verification are implemented in a user-space library used by the AllNet chat program.

This very early implementation has proven the basic principles of AllNet, that ad-hoc networking can be a complement to more established and better-performing infrastructure based networking, and can provide basic communication facilities for interpersonal exchanges even when nothing else can.

References

[1] F. Chung and L. Lu, "The average distance in a random graph with given expected degrees," *Internet Mathematics*, vol. 1, 2003.

[2] Sun Microsystems, Inc., "NFS: Network file system protocol specification," RFC 1094, March 1989. [Online]. Available: http://tools.ietf.org/rfc/rfc1094.txt

[3] E. Biagioni, "A ubiquitous, infrastructure-free network for interpersonal communication," in *The Fourth International Conference on Ubiquitous and Future Networks*, Phuket, Thailand, 2012.

[4] K. W. Wall, "Misunderstanding trust," June 20 2012. [Online]. Available: http: //www.infosecisland.com/blogview/21676-Misunderstanding-Trust.html

[5] B. Christianson and W. Harbison, "Why isn't trust transitive?" in *Security Protocols*, ser. Lecture Notes in Computer Science, M. Lomas, Ed. Springer Berlin / Heidelberg, 1997, vol. 1189, pp. 171–176, 10.1007/3-540-62494-5_16. [Online]. Available: http://dx.doi.org/10.1007/3-540-62494-5_16

[6] T. Wu, "The SRP authentication and key exchange system," RFC 2945, September 2000. [Online]. Available: http://tools.ietf.org/rfc/rfc2945.txt

[7] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, ser. Lecture Notes in Computer Science, vol. 1807. Springer, 2000, pp. 139–155.

[8] C. Desiato and E. Biagioni, "Sharing networking resources to create a pervasive infrastructure," in *Ninth International Conference on Technology, Knowledge, and Society*, Vancouver, Canada, 13-14 January 2013.

[9] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta, "PeerSoN: P2P social networking - early experiences and insights," in *Proceedings of the Second ACM Workshop on Social Network Systems Social Network Systems 2009, co-located with Eurosys 2009*, Nürnberg, Germany, March 31, 2009, pp. 46–52.

[10] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: a privacy preserving online social network leveraging on real-life trust," *IEEE Communications Magazine*, vol. 47, no. 12, December 2009.

[11] B. Birt, "Peerbook," June 29 2010. [Online]. Available: http://blogs.cs.st-andrews.ac.uk/peerbook/

[12] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, "OpenPGP message format," RFC 4880, November 2007. [Online]. Available: http://tools.ietf.org/rfc/rfc4880.txt

[13] I. Clarke, "A distributed decentralized information storage and retrieval system," 1999. [Online]. Available: http://freenetproject.org/papers/ddisrs.pdf

[14] S. Buchegger, "Coping with misbehavior in mobile ad-hoc networks," Ph.D. dissertation, Lausanne, Switzerland, 2004.

[15] "Trilogy emergency response application (TERA)." [Online]. Available: http://www.cdacnetwork.org/public/resource/ trilogy-emergency-response-application-tera

[16] "Byzantium live distro." [Online]. Available: http://wiki.hacdc.org/index.php/Byzantium_Live_Distro\#Goals

[17] T. Fujiwara and T. Watanabe, "An ad hoc networking scheme in hybrid networks for emergency communications," *Ad hoc Networks*, vol. 3, pp. 607–620, 2005.

[18] H. Mehendale, A. Paranjpe, and S. Vempala, "Lifenet: A flexible ad hoc networking solution for transient environments," Demo at SIGComm, 2011. [Online]. Available: http://www.cc.gatech.edu/~vempala/papers/lifenet.pdf

[19] R. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, February 1978.

[20] RSA Laboratories, "PCKS #1 v2.1: Rsa cryptography standard," 2002.

[21] Information Technology Laboratory, "Secure hash standard (SHS)," FIPS PUB 180-3, Gaithersburg, MD 20899-8900 USA, June 2007.