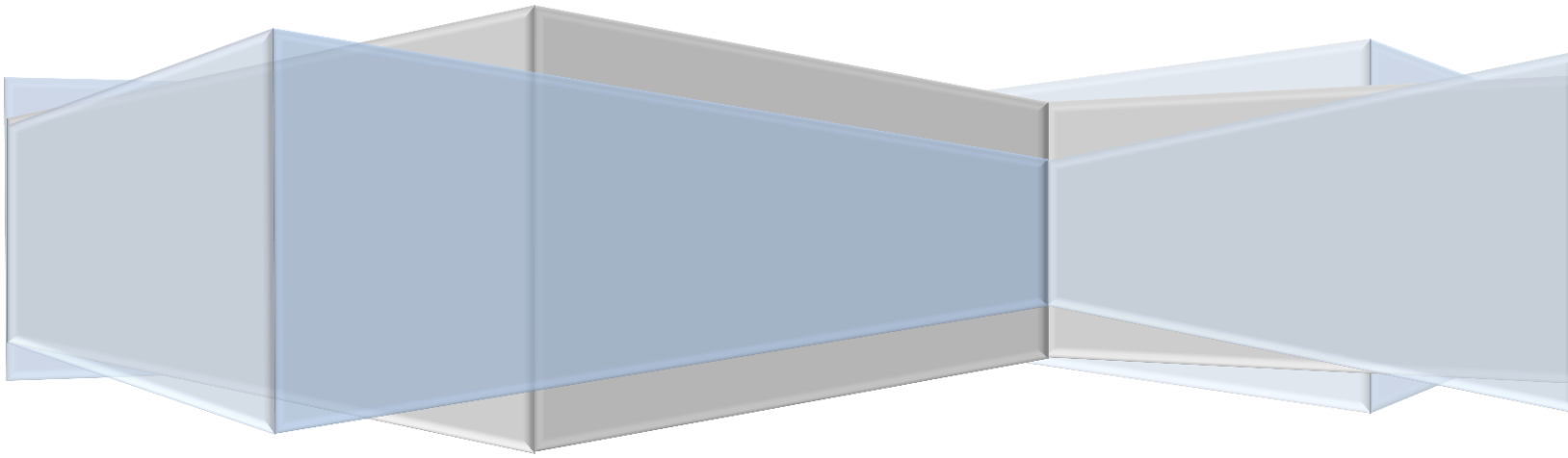
# Gossiping Ad-hoc Wireless networking for First Responders – GAWFR

## Design Project 2 Report

**Predrag Gruevski** (predrag@mit.edu, T11)
**Anton Anastasov** (aanastasov@mit.edu, R12)
**Logan Mercer** (lmercer@mit.edu, R1)
**(Strauss/Simosa/Melvold/Jackson)**

**May 12, 2013**



# Overview

The Gossiping Ad-hoc Wireless networking for First Responders (GAWFR, pronounced 'gopher') is a system that allows first responders to send pictures and location data back to a central HQ independently of existing communication infrastructure. Nodes in GAWFR use gossiping to disseminate network statistics and two new protocols to relay locations and pictures securely and efficiently back to the HQ. The location protocol ensures that the location updates of first responders arrive at the HQ within 5 minutes by using redundancy and replication, while a sophisticated batching system ensures that locations contribute relatively trivial congestion. The picture protocol ensures that throughput of pictures is maximized by utilizing a metric that minimizes the expected time it takes pictures to reach the HQ. Finally, GAWFR's security features prevent malicious third parties from interfering with the proper operation of the network in any way.

# Protocol Details

GAWFR is built on top of the 802.11n-2009 standard for wireless devices [1], and requires devices to possess an 802.11n radio with MPDU support [1] [2] in order to achieve optimal performance. Traffic on the GAWFR network is composed of gossip, location and picture messages, and each of the three types of traffic follows a separate communication protocol. GAWFR uses a queuing system to send and receive messages: messages received through the `receive()` API call are passed to the queue of the appropriate protocol based on the message type, while outgoing messages are placed into a single queue that reorders them by priority (see Figure 1).

| Message type | Type code | Priority |
|---|---|---|
| Location message | 2 | 5 |
| Location ACK | 4 | 4 |
| Gossip message | 1 | 3 |
| Picture message | 3 | 2 |
| Picture ACK | 5 | 1 |

Figure 1 – Message types, in order of decreasing priority from top to bottom.

GAWFR messages are no larger than one MPDU transmission unit (4095 bytes [2]), and are composed of a common message header, and contents that depend on the message types.

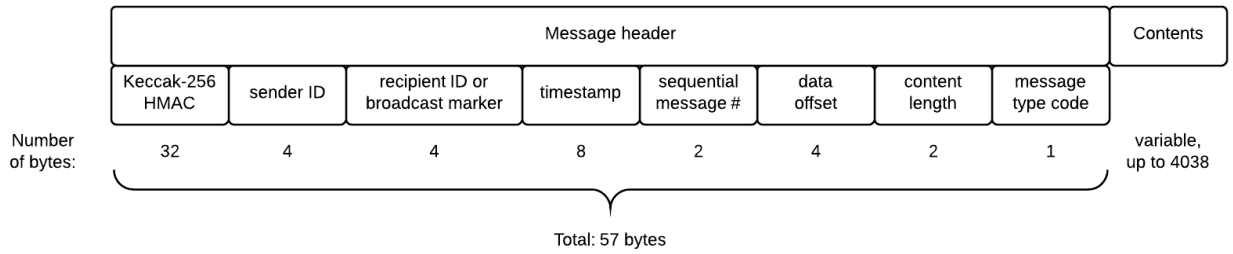



Figure 2 – Layout of the message header.

The header is split up into eight fields:

- Keccak-256 [3] HMAC [4]
  This is a hash-based message authentication code field (HMAC) [4] which verifies the integrity of the message. The way the HMAC is computed is discussed in detail in the Security section.
- Sender ID – contains the unique identifier of the node that sent the message.
- Recipient ID
  This field contains the unique identifier of the intended recipient node. If the message is a broadcast (i.e. not meant for any particular node) then this field is set to 0xFFFFFFFF and the `broadcast()` API call is used. Otherwise, the `send()` API call is used with the recipient specified in this field.
- Timestamp – contains UNIX time when the message was sent.
- Data offset
  Pictures do not fit inside one message, so they are split into multiple smaller pieces. The data offset field specifies the starting location of the subset of the picture byte stream that this message contains. If the high bit of the data offset field is set, then this message contains the end of the picture byte stream.
- Sequential message number
  Due to the high maximum bandwidth of 802.11n, multiple messages may be sent with the same timestamp. This field is a simple message counter (that resets to 0 after reaching its maximum value) so that any two messages from the same sender are guaranteed to have a different combination of timestamp and counter. Timestamp – contains UNIX time when the message was sent.
- Content length – contains the length (in bytes) of the variable-length content field.
- Message type code – represents the type of the message, as shown in Figure 1.

The layout of the header fields is shown in Figure 2.

## GOSSIP PROTOCOL

GAWFR uses a gossip protocol to disseminate statistics throughout the network and make routing decisions. Every second, each node in the network places into the GAWFR output queue a message marked for broadcasting that contains a set of network statistics (see Figure 3). The protocol uses the `scan()` API call every 30 seconds, as well as local information and other nodes' gossip broadcasts in order to compute the network statistics that compose the gossip.

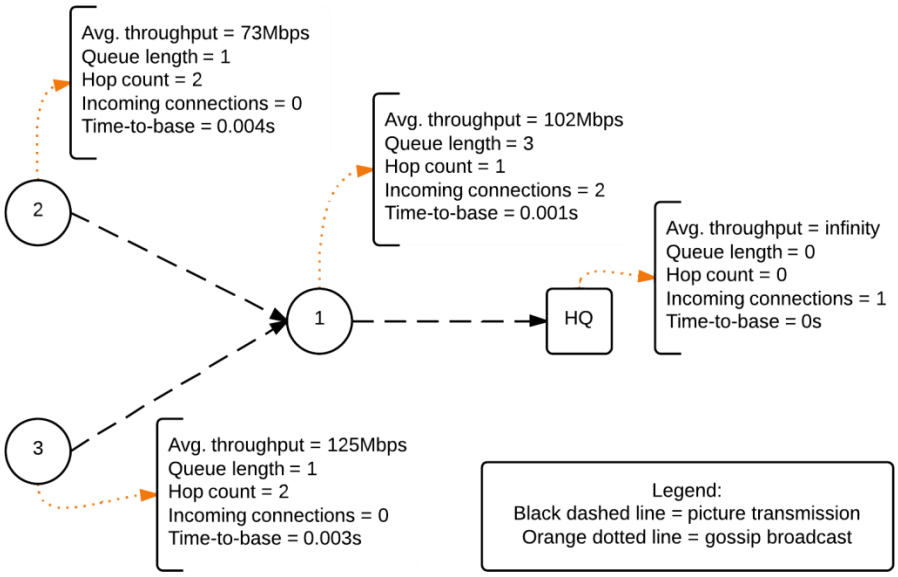



Figure 3 – Example of GAWFR gossip in action. Note that nodes 2 and 3 are outside of the HQ's radio range.

To describe the statistics and how they are computed, we will consider a fixed node $i$ and the set $V(i)$ composed of all nodes within radio range of $i$. Let $p_i(t)$ be the calculated packet loss probabilities from node $i$ to node $t$. The following statistics are calculated and broadcasted in gossip:

- Hop count (HC)

  HC is the number of hops needed to reach the HQ. It is computed based on gossip received during the last one second using the following formula:

  $$HC_i = 1 + \min_{t \in V(i)} HC_t$$

  The HQ has a HC of 0.

  After calculating the HC of node $i$, we create the set $\overline{V}(i)$ with all nodes $t \in V(i)$ for which $HC_t < HC_i$ holds.

- Average throughput (AT)
  AT is an estimate of how many pictures the node sends per second. It is computed as a 5-moving average [5] of the reciprocals of the time taken to send each of the five last pictures. If $k < 5$ pictures have been sent, a $k$-moving average is used instead. The HQ and nodes that have not sent any pictures yet have an AT of infinity.
- Queue length (QL)
  QL is the number of pictures currently in the picture queue of the node. The HQ has a QL of 0.
- Incoming picture connections (PC)
  PC is the number of GAWFR nodes currently sending pictures to this node (see the example on Figure 3).
- Time-to-base (TTB)
  TTB at node $i$ is an estimate of the time from when a picture leaves the output queue at node $i$ to when it is received by the HQ. The TTB is computed together with a related indicator we will call NH, which stands for "next hop". The NH represents the node that node $i$ has to send a picture to in order to achieve the TTB. The TTB and NH are computed as follows:

$$TTB_i = \min_{t \in \overline{V}(i)} \left( \frac{4}{15\, p_i(t)} + \frac{QL_t + PC_t}{AT_t} + TTB_t \right)$$

$$NH_i = \arg\min_{t \in \overline{V}(i)} \left( \frac{4}{15\, p_i(t)} + \frac{QL_t + PC_t}{AT_t} + TTB_t \right)$$

  Note that TTB and NH are computed over $\overline{V}(i)$, the set of all reachable nodes at lower HC. The intuition behind the terms in the formulae is as follows: $\frac{4}{15\, p_i(t)}$ is the expected time to transmit 5MB of data (approx. one high resolution picture) over a 150Mbps link with loss probability $p_i(t)$, while $\frac{QL_t + PC_t}{AT_t}$ is an upper bound on the expected time that a sent picture will spend waiting in the queue given the current conditions at node $t$. Then, the sum of the three terms for a node $t$ is an upper bound on the expected time for a picture sent from $i$ to $t$ to reach the HQ.
  Note that a node $t$ which was not found during the last `scan()` has $p_i(t) = 0$ and will never be chosen as the NH.

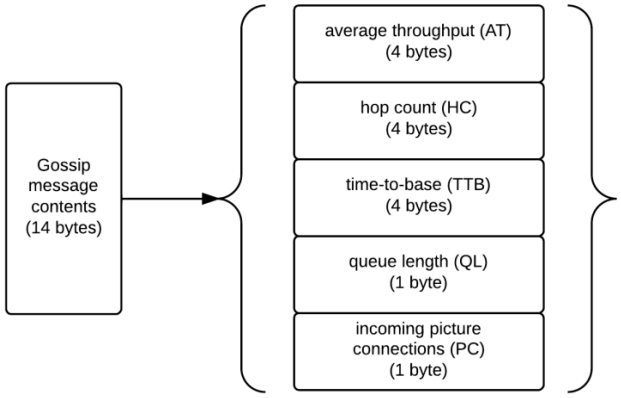



Figure 4 – Layout of gossip message contents.

Gossip follows the principle of best-effort delivery, and no ACKs are sent in response to gossip broadcasts. The layout of the gossip content can be seen on Figure 4.

## PICTURE PROTOCOL

The objective when sending pictures is twofold – we want to maximize the throughput at the HQ while adding the minimum possible congestion to the network. If we consider both queue congestion and channel congestion, then pictures start contributing to congestion as soon as they are taken and only stop when they are received by the HQ. Minimizing the expected time it takes pictures to reach the HQ is therefore a good routing metric, and this is what GAWFR uses. The metric is represented by the TTB and NH values computed by the gossip protocol and described in the previous section.

As soon as a picture is generated or received by a GAWFR node, it is added to a priority queue ordered by recency according to the picture's DCF DateTimeOriginal tag [6] – the time when the picture was taken as defined in the DCF picture standard [6] employed by all digital cameras. This step is taken to ensure that given a choice between multiple pictures, each node will choose to send the most recent picture first. While this decision implies that GAWFR does not provide any guarantees of fairness between nodes, it makes sure that the HQ receives the most up-to-date pictures available at all times.

Whenever the picture queue is not empty and no picture is being sent, the top picture in the queue is removed from the queue and split into parts no longer than 4038 bytes, so that they fit inside GAWFR messages. The data offset field in the header of each message is set to the offset within the picture from where the contents originated, and the messages are addressed to the node designated in the NH value computed by the gossip protocol. The messages are then put in the GAWFR output queue, and Picture ACKs are expected in response. In case no

ACK is received for a particular message, it will be resent up to ten times before aborting the transfer and placing the picture back in the picture queue.

When a transfer is aborted, the gossip protocol is forced to do an early `scan()` and update all statistics, including the TTB and NH values. The sending process then starts over.

The content field of Picture ACKs only contains the header of the picture message being acknowledged.

## LOCATION PROTOCOL

The location protocol also has two objectives – it must contribute the minimum possible congestion to the network while also ensuring that location updates arrive at the HQ within 5 minutes. The first objective is fulfilled by sending many location updates together in batches, while the second one is achieved using limited redundancy and replication.

The location of each node is packaged with the node's unique ID and a timestamp indicating when the measurement was taken. One or more of these 'location fragments' concatenated together represent the contents of a location message, as shown on Figure 5. If a node with HC of $k$ broadcasts a location message, the message will only be acted upon by nodes with HC of $k - 1$ or less in order to ensure that locations only propagate toward the HQ (note that nodes within radio range know each other's HC due to the constant gossiping). Nodes with HC of $k$ or higher will ignore the message.

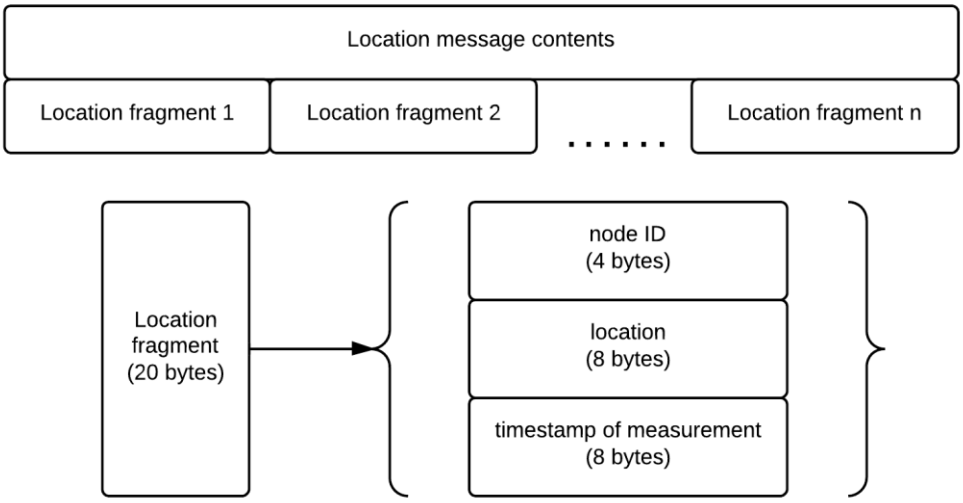



Figure 5 – Location fragment and message contents layout.

Since the expected size of a GAWFR network is several thousand nodes, it is feasible for each node to keep a hash table of all nodes whose locations it has received, their coordinates and a

timestamp indicating how old the coordinates are. The hash table is updated whenever a location fragment with a newer timestamp is received for any node, or if the node's ID did not have an entry in the hash table. For every added or modified entry in the hash table, values called 'wait time' (WT) and 'send time' (ST) are computed and put on a priority queue ordered by ST (the 'location queue') together with the corresponding location fragment. WT is the maximum amount of time (in milliseconds) the location fragment will be allowed to wait for a batch to be assembled, while ST is the latest time when the location fragment can be sent so that no more than WT time is spent waiting. For location fragment $x$ being processed at time $T$ at node $i$, ST and WT are computed as follows:

$$WT = \begin{cases} \dfrac{x.timestamp + 300000 - T - t_{max}\, HC_i}{HC_i + \varepsilon}, & HC_i \mod B \equiv 1 \\ 0, & \text{otherwise} \end{cases}$$

$$ST = T + WT$$

$t_{max}$, $B$ and $\varepsilon$ are parameters that can be modified to optimize the algorithm for different conditions. $t_{max}$ is the assumed worst-case time to broadcast a location message and ensure it was received by a node with lower HC. $B$ is a parameter that controls batching – every $B$ hops, a node will keep the location fragments in the queue for $WT > 0$ time in order to promote batching and decrease congestion. $\varepsilon$ controls how big a fraction of the remaining time each batching node consumes by waiting.

The idea behind the WT calculation is that we want that the location fragment has to be delivered to the HQ within $x.timestamp + 300000 - T$ milliseconds, and it may take us up to $t_{max}HC_i$ time to send the location message $HC_i$ times to get to the HQ. Then every $B$ hops, we can afford to batch locations by waiting for a fraction of the remaining time. Adjusting the $\varepsilon$ and $B$ parameters allows GAWFR to accommodate a range of maximum HC values (up to the theoretical maximum of $30000/t_{max}$). To increase the maximum HC for which GAWFR will deliver locations within 5 minutes, simply increase $B$ and $\varepsilon$. Figures 6a, 6b and 6c show how WT varies as a location fragment propagates toward the HQ, the total time spent waiting as a function of the HC of the node whose update is being propagated and the total time to reach the HQ assuming the worst-case $t_{max}$ time to broadcast at every step. In each of the three figures, $\varepsilon = 1$, $t_{max} = 50ms$ and $B = 3$.

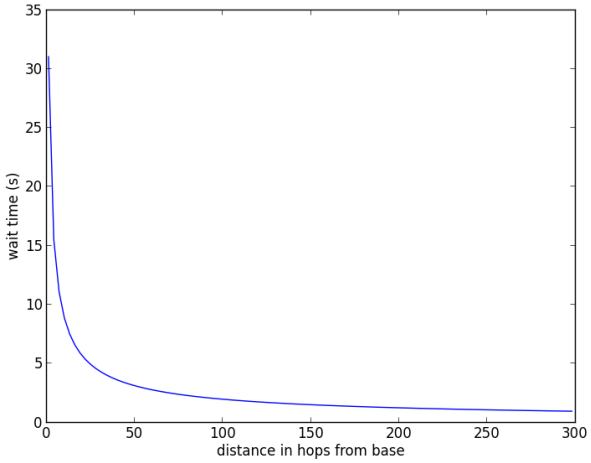



Figure 6a – WT at each hop count as a location fragment propagates toward the HQ

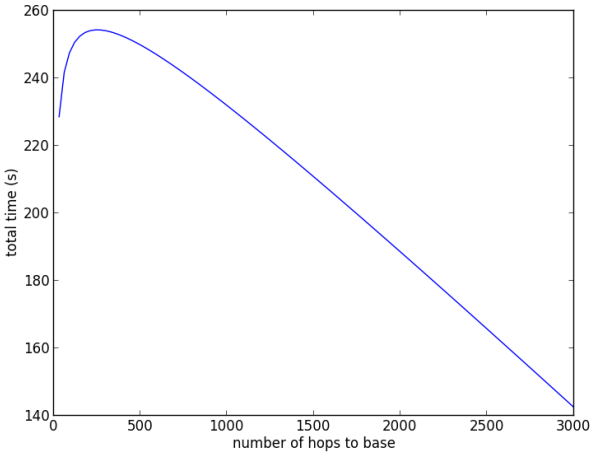



Figure 6b – Total time spent waiting as a function of HC

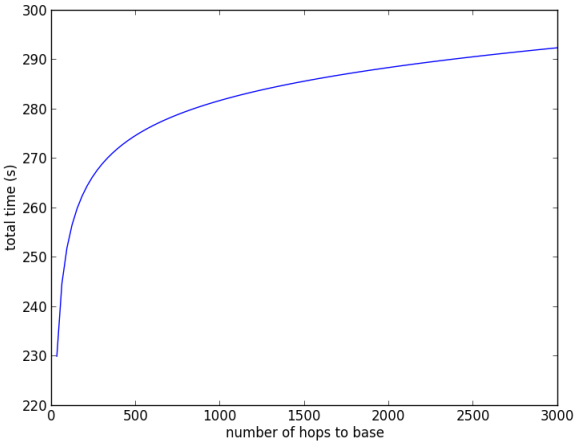



Figure 6c – Worst-case time taken for a fragment to reach the HQ.

Since the location queue is ordered by ST, all the location protocol needs to do to obey the batching contract is to form and send a location message no later than the earliest ST. The protocol will therefore put a message marked for broadcasting in the GAWFR output queue at the ST time of the top element in the location queue, or when the location queue grows to 201 elements (since a message can only contain up to 201 location fragments). The batch of fragments used for the message is composed of the fragments at the top of the queue (i.e. with minimal ST).

After the message is broadcasted, the sender waits for time $\frac{t_{max}}{5}$ to receive location ACK(s) for the message. Since the message is a broadcast, multiple nodes may decide to ACK as described

below. The message is rebroadcasted every $\frac{t_{max}}{5}$ until at least one ACK is received. The probability that it takes more than $t_{max}$ for the message to propagate one hop closer to the HQ is then smaller than $(1-p)^5 \ll 1$ where $p$ is the loss probability on the highest-quality link.

The location ACK is not a standard 802.11 [2] ACK since it is a confirmation of receipt of a *broadcast*, and not a point-to-point transmission. Location ACKs are also formatted as broadcasts whose message content is the header of the location message being acknowledged. GAWFR nodes listen to both location messages and location ACKs, and will only send a location ACK if the message they have received fewer than 5 ACKs from other nodes that acknowledge to the same message. This measure exists to limit congestion caused by excessive ACK messages if many nodes at the same HC receive a given location message, while at the same time providing a very high probability (greater than $1-(1-p)^5 \gg 0$ where $p$ is the loss probability of the highest-quality link) that the original sender received the ACK.

## SECURITY

The main security goal of GAWFR is to ensure that malicious third parties cannot interfere with the proper operation of the network without channel jamming techniques. GAWFR does not use link encryption in order to maximize performance, so any passive listener will be able to read data transmitted on the network, but will not be able to compromise its integrity. The security of GAWFR rests on two key assumptions: that prior to joining the GAWFR network, all nodes possess a randomly generated, securely distributed common secure password (CSP), and that the underlying API provides a `scan()` function that is secure against packet forging and replay attacks.

The assumption that `scan()` is secure is necessary because in its absence, an attacker controlling a large number of malicious nodes with powerful transmitters can make `send()` fail to reach its target with high probability. The attack is executed as follows: the malicious nodes are scattered across the network so that they can read a significant fraction of all network traffic. Each node is configured to replay all `scan()` packets and `scan()` responses it comes across using its powerful transmitter, so that effectively all GAWFR nodes think that any node in the network is within radio range. When a GAWFR node decides to pick a node to `send()` to, there is a probability $p$ to pick one that isn't within a radio range, resulting in a timeout. Since GAWFR can have thousands of nodes, but the radio range of each node is very small, this probability $p$ is much greater than 0. After the timeout, picking a new node to `send()` to results in a new timeout with the same probability $p \gg 0$, ultimately resulting in `send()` failing with high probability.

The assumption that all first responders know a common secret password (CSP) prior to joining the GAWFR network is also reasonable, since this is similar to the way companies often secure their on-site wireless networks. One of the many ways to fulfill this assumption is to distribute the CSP in encrypted form to all first responders, and then have them obtain the decryption code from a tamper-proof package in their possession. For best security, the CSP should be at least 256 bits long, should be generated using a truly random process and should be stored only on the cryptographic chip of the GAWFR-capable device.

The Keccak-256 [3] HMAC [4] field (Figure 2) within the header of every message contains a 32-byte hash value that allows a GAWFR node to securely verify the integrity of a message in the following manner. Let PT be the plaintext that results from concatenating CSP with the entire message, and modified so that the HMAC [4] field is populated with zeroes. The HMAC [4] value is obtained by transforming the PT using Keccak-256 [3], and this process can be replicated by any recipient of the message. If the computed HMAC [4] matches the HMAC [4] specified in the message header, the message is validated since attackers do not possess the CSP and cannot compute a new valid HMAC [4] as a result. If the two HMACs do not match, the message is dropped since its integrity cannot be confirmed. This approach was proven secure by the Keccak/SHA-3 [3] team, so if the CSP is not compromised, it is impossible for malicious third parties to impersonate GAWFR nodes, modify message contents or execute man-in-the-middle attacks.

Since all GAWFR nodes have synchronized clocks, nodes can easily discard messages whose timestamps are outdated. Should a malicious node replay a message while the timestamp is still valid, then the message will still be ignored by all recipients who received the original broadcast because each message specifies a sequential message number (Figure 2) which is guaranteed to be different for any two messages within the timestamp validity period. This combination of features ensures that a replay attack attempt can only increase the effective radio range of the sender node and the probability that the intended recipients received the transmission, both of which are desirable side effects.

## ANALYSIS

GAWFR is designed to perform as close to optimally as possible regardless of the number of nodes in the network or their configuration. We will analyze the performance of GAWFR under several different scenarios that are likely to be encountered in practical applications.

## Efficiency and Overhead Analysis

Since GAWFR seeks to provide strong delivery guarantees for locations, efficiency is not a priority for the location protocol. When a node broadcasts its location, every node with an HC less than or equal to the broadcasting node's HC will read the message and rebroadcast any location fragments it had not seen before, in order to combat the fact that connections are unreliable and packets may be dropped frequently. In order to minimize the congestion effects of this strategy, location fragments are designed to be tiny (only 20 bytes long) and will be batched up and broadcasted together whenever possible. In the absolute worst case, a node will have to transmit location fragments for all GAWFR nodes in the network. For a network size of 10000 nodes, this works out to 200000 bytes, or no more than one tenth of a single image. Furthermore, this broadcast will not happen more than once every 30 seconds (since that is the GPS unit's update frequency), meaning that in the worst case, this node's location broadcast uses up approximately 0.01% of the available bandwidth averaged over 30 seconds and using an effective maximum throughput of 50Mbps. Therefore, while the location protocol is not efficient, it has essentially negligible effect on the overall congestion level.

As the picture protocol was designed to maximize throughput, it is far more efficient than the location protocol. It takes advantage of the gossiping information to find the best possible target of their data. The only possible source of inefficiency is if a routing decision is made based on stale data; however, with gossip updating every second and packet loss probabilities updating every 30 seconds, the probability of such an event is extremely low.

While gossip traffic is pure network overhead and therefore a source of inefficiency, it is unreasonable to consider it in isolation. Gossiping allows the picture protocol to function at maximum throughput with high probability, so any inefficiency is more than made up for. In addition, gossip data is only 14 bytes in size, so broadcasting it at one-second intervals causes essentially no measurable effects on congestion.

## Failure to Deliver Locations to Base

While GAWFR will do its best to deliver location data on time, there are two situations where a node's location may fail to reach the HQ within five minutes. The first is when a node or group of nodes (not including the HQ) becomes isolated from the rest of the network for close to five minutes or more. The second situation is when a node has poor signal to all its neighbors -- if the probability of delivering a broadcast successfully is close to 0%, the node's location data may take longer than five minutes to reach base. However, both of these are local issues beyond our control that affect a relatively small number of nodes. Since the location protocol is robust and adjusts the waiting time (WT) of location fragments based on the worst-case time to

reach the base, we feel that such isolated instances of locations arriving more than five minutes late will be extremely rare and not a problem in practice.

It is important to note that the timely delivery of location data limits the scalability of our design. Each time location update passes from one HC to the next takes at most $t_{max}$ time. Thus our maximum HC is $\frac{300000}{t_{max}}$. At a HC higher than this value, GAWFR cannot guarantee that its nodes' locations will reach the base station within 5 minutes. Assuming that $t_{max}$ = 50ms, we have a maximum HC of 6000, or approximately 18 kilometers of distance from the HQ assuming 30m radio range, which we consider sufficient.

## INTERFERENCE FROM MALICIOUS NODES

As described in the Security section, GAWFR uses a combination of a timestamp, message number, explicit sender and recipient data, and a robust and provably secure HMAC [4] scheme to secure messages. The security of GAWFR rests on the assumptions that first responders do not turn malicious, that the common secret password is truly random and is not compromised, and that hash collision attacks on Keccak-256 [3] are extremely difficult. Violating any one of these assumptions may cause GAWFR to be vulnerable to malicious nodes and fail to deliver messages to the HQ; however, we believe that these assumptions are simple to fulfill with minimal training for first responders.

Another way that a malicious party may ensure that messages fail to reach the HQ is by jamming the wireless channel on which GAWFR operates. Such an attack requires specialized knowledge, careful preparation and non-trivial amounts of electricity, and cannot be sustained for a prolonged period from a mobile device. We believe that such a situation is unlikely in an emergency scenario, and GAWFR provides no guarantees about performance or proper operation if a jamming attack is in progress.

Replay attacks were a special concern when designing the GAWFR system, since a well-executed replay attack may increase congestion exponentially if the system is not well designed. Since GAWFR does not use encryption, it is trivial for a malicious third party to receive a GAWFR packet, read it and then decide to rebroadcast it a large number of times. However, this attempt as an attack will only increase the effective range of the original transmission since all nodes who received the original message will ignore any subsequent retransmissions since receiving a valid message is an idempotent operation by design. Namely, if node $i$ receives a message that is not a broadcast and was not addressed to it, or has an old timestamp, it will discard the message automatically. Assume that the timestamp is not outdated, and that the message is either a broadcast or addressed to node $i$. If the message was gossip, it carries no new information and the network statistics that node $i$ computes do

not change. If the message contained locations, the locations were already received previously, so an ACK will be sent in response but no further action will be taken (note that the timestamp has to be valid so this can only happen in a few-millisecond window once every 30 seconds). Finally, if the message contained picture data that was already received previously, the message will be dropped and no action will be taken.

### NODES RECEIVING UNEXPECTED MESSAGES

If node $i$ is sending node $j$ a picture message, it is possible that a third node, node $k$ will also receive the message. Even if node $k$ is at a lower HC (i.e. closer to the HQ) than node $j$, node $k$ will discard the message since it was not addressed to it. This design decision introduces some inefficiency but is crucial to avoid replay attacks and related exploits that may endanger the proper operation of GAWFR. Note that all other types of transmissions used in GAWFR are either ACKs (which are ignored if received multiple times) or are broadcasts and are not addressed to any particular node, so this scenario is only relevant for picture messages.

## PERFORMANCE

GAWFR achieves near-optimal performance under all load levels. Since gossip and location-related messages have the highest sending priority (as shown on Figure 1) and the size of those messages is miniscule as discussed in the Efficiency and Overhead Analysis section, the performance of the gossip and location protocols is always optimal independent of overall network load.

As we increase the size of the network (in terms of both number of nodes and maximum HC) while also increasing the number of pictures that need to be relayed back to HQ, we will eventually hit a performance bottleneck. This  when all nodes at a common (low) HC that are in radio range of the highest-traffic branch of the network reach maximal congestion and cannot sustain any more traffic. Thanks to the load-balancing characteristics of the routing metric and frequent updates generated by the gossip protocol, the equilibrium throughput reached under such heavy load will be very close to the theoretical maximum throughput for the given network configuration.

Let us consider a network with 10000 nodes uniformly distributed in a circle around the HQ with a 3km diameter, and that each HC level is defined by a 30-meter-wide circular ring centered on the HQ. That is, the HC=1 region is a 30m radius circle, the HC=2 region is a 30m wide ring around the HC=1 region etc. This network layout has a maximum HC of 50, and some simple arithmetic shows that there are four nodes one hop away from the HQ. These four nodes will have to relay all data generated by the entire network, so it is reasonable to assume

that they are all at the high-load equilibrium described above. We will assume that the link loss probability toward the HQ for all four nodes is 25% and that the HQ cannot receive data faster than 150Mbps in total (the maximum single-antenna speed of 802.11n [1]), and we will ignore the impact of locations and gossip since their total contribution is insignificant as described above. We can then calculate that the total effective throughput to the HQ is $.75 * 150 = 112.5$Mbps, or approximately seven 2MB-size pictures per second, and that the throughput achieved at each of the four non-HQ nodes is $.75 * 150/4 = 28.125$Mbps. These values are sufficient to fully saturate the bottleneck link at the HQ.

If traffic on the GAWFR network does not reach the high-load equilibrium for any node, we consider this a light-load scenario. We will use the same setup as above, but assume that only two of the four nodes are reachable of parts of the network at higher HC that produce pictures. Then, the total effective throughput to the HQ is again 112.5Mbps, but now each transmitting node at HC=1 uses approximately 56.25Mbps of bandwidth. These values are again sufficient to fully saturate the bottleneck arising from the finite capability of the HQ to receive data.

Finally, if no pictures are being transmitted via GAWFR, this is a no-load scenario and the network clearly functions optimally.

## Conclusion

GAWFR uses a combination of three custom protocols for gossip, location and picture traffic to meet the requirements of the project. The system ensures that locations of first responders reliably reach the HQ within 5 minutes, while also maximizing the throughput of pictures back to the HQ. Our analysis shows that the system is robust under different load scenarios and network configurations, and that it is resistant against malicious third parties seeking to interfere with its proper operation with very modest security assumptions.

# References

[1] IEEE Computer Society, "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements: Amendment IEEE Std 802.11n™-2009," The Institute of Electrical and Electronics Engineers, Inc., New York, 2009.

[2] IEEE Computer Society, "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements: IEEE Std 802.11™-2007," Institute of Electrical and Electronics Engineers, Inc., New York, 2007.

[3] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, "The Keccak sponge function family," 24 January 2013. [Online]. Available: http://keccak.noekeon.org/. [Accessed 3 May 2013].

[4] M. Bellare, R. Canetti and H. Krawczyk, "Keying Hash Functions for Message Authentication," in *Advances in Cryptology – Crypto 96 Proceedings*, 1996.

[5] E. W. Weisstein, "Moving Average," MathWorld -- A Wolfram Web Resource, [Online]. Available: http://mathworld.wolfram.com/MovingAverage.html. [Accessed 10 May 2013].

[6] Camera & Imaging Products Association, *Design rule for Camera File system: DCF Version 2.0 (Edition 2010),* Camera & Imaging Products Association, 2010.

Word count: 4990, not counting references