Surveillance Ad hoc Network: Reliable Broadcast

Rawad Ghostin,

***Abstract*—In certain environments, infrastructure-based networks become less adapted to particular requirements, such as high mobility for devices and network portability. Ad hoc networks address the problem by making the devices become the sole infrastructure required for communications. However, some protocols designed with infrastructure-based networks in mind, such as the UDP/IP broadcast, are not well adapted to ad hoc transmissions. These unfitting protocols lose their efficiency and robustness, making them unreliable channels. In this report, we document a broadcast protocol we designed, the reliable broadcast, particularly adapted to a flying ad hoc network. The reliable broadcast is built on top of an unreliable channel, in our case the UDP/IP broadcast. The protocol introduces redundancies that strategically take advantage of the geographically dispersed nature of the network, in order to substantially minimize the probability of loss during transmissions.**

I. Introduction

THE objective of the surveillance ad hoc network is to assert the watch over a geographical region in a robust manner. The network consists of a set of drones on which is mounted a raspberry PI device and a camera. The region is divided into equal sub-areas, and each drone is responsible of maintaining surveillance over one assigned sub-area. The surveillance is done by flying in a cyclic pattern, periodically taking images of the fields within the scope of the camera. By design, each drone of the surveillance network is required to constantly have an up-to-date image of the whole photographed area. This allows operators of the network to connect to any drone in range, and observe, in real time, the complete surveyed region. In practice, each drone continuously broadcasts the photographed images of its area to the whole network, and mutually receives, via broadcast from other drones, images of other sub-areas. Flying ad hoc networks are highly peculiar, the constant movement of nodes and the potential physical obstructions between them cause prevalent broadcast methods, such as the UDP/IP broadcast, to be misplaced, as they introduce inconsistency and unreliability. The erratic character of such transmissions affects drastically the robustness of the surveillance network. In this report, we present a statistically reliable broadcast protocol that we designed, particularly adapted to flying ad hoc networks. The approach taken is to, first, investigate the flaws of standard solutions, then, attempt to construct a more adapted protocol that improves the reliability and robustness in such networks.

Supervisors: Matthieu Defrance, Nassim Versbraegen

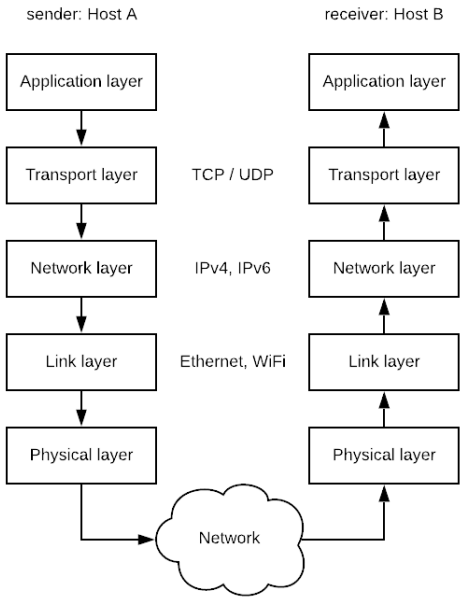
II. Material

A. Layered model of networks

In order to provide structural clarity for network communications, the very vast majority of networks are organized in layers. Each layer is responsible of providing a set of services - the so-called **service model** - to the layer above. One prevalent network model is the **TCP/IP** model, currently employed in the Internet infrastructure. The model is composed of 5 layers:

- **Application layer :** the application layer is where network applications reside. Applications are software programs that communicate with each other by transmitting and receiving data packets, called **messages**. In practice, network applications make usage of the lower layers via a software abstraction called a socket.
- **Transport layer :** the transport layer is responsible of transporting messages between applications. An encapsulated message is called a **segment**. The current Internet architecture mainly employs two transport protocols, **TCP** and **UDP**. The TCP protocol offers a well-supplied service-model, providing reliable data transfer, congestion control and flow control. These features require for each transmission sent, a feedback from the receiver for the sender to evaluate the status of the transmission and adjust accordingly. As a consequence, a TCP connection is *connection-oriented* and can only be established between exactly 2 endpoints. Conversely, UDP is a light-weight protocol with a minimal service-model. UDP operates by merely encapsulating application messages and transferring them to the lower layer protocol with a *best-effort delivery* policy (no delivery is guaranteed). Unlike TCP, no elaborate network features are provided in the service-model. UDP is *connection-less* and can be used for communications with one or multiple receivers.
- **Network layer:** the network layer is responsible for transferring segments between *hosts*. An encapsulated segment is called a **datagram**. Classical networks are based on an infrastructure of **nodes** (routers and switches), in which datagrams transit according to a routing algorithm, before arriving to destination. The service-model of the network layer is minimal and datagrams delivery adhere to the best-effort policy.
- **Link layer:** The link layer is responsible of carrying encapsulated datagrams, called frames, from one node to its physical neighbor. Neighbors are 2 nodes connected physically via a link (*e.g* 2 routers connected physically via an Ethernet cable). Links exist in many forms, the most common ones being

Figure 1. A transmission in the TCP/IP model



Ethernet(protocol 802.3) for wired links and WiFi (protocol 802.11) for wireless ones.

- **Physical layer:** The physical layer is responsible of transmitting frames on a link using physical signals. Signals for wired links may be electrical (802.3) or optical (802.8 - fiber optic) pulsations; as well as radio modulations for most wireless links (802.11, 802.15.1 - Bluetooth).

B. Ad hoc networks

An Ad hoc network is a type of **decentralized** network. Ad hoc networks do not depend on a pre-existent physical infrastructure, such as routers, links or access points in standard networks. Instead, each participant of the network, called **node**, participates in routing by forwarding packets to other nodes. The routing algorithm in an ad hoc network is extremely dynamic, as the determination of which node receives next the forwarded packets is solely based on network connectivity and range. Such dynamism enables **wireless ad hoc network (WANET)** nodes to move freely, and create or join self-configured networks "*on the fly*", anywhere and anytime. The decentralized nature of ad hoc networks makes them extremely robust and highly available. Availability is a key concept in information security (*cf.* [1]), in which highly available systems aim to remain available at all times, preventing service disruptions due to accidental or intentional hazards. An ad hoc networks is inherently safe from disruptions as its disruption would require the rupture of each one of its nodes.

III. State of the Art

A. Reliability of transfers

Commonly, network transmissions are far from being flawless. The process of delivering a packet from one device to another involve material transmissions and hardware on which physical limits apply, making the transmission channel unreliable. We take the opportunity to introduce the Shannon-Hartley theorem, that calculates the maximum rate at which data can be transferred over a link of a specified bandwidth in the presence of signal noise (*cf.* [2] and [3]):

$$C = B \cdot log_2(1 + \frac{S}{N}) \quad (1)$$

where:

- C is the capacity of the channel (bit/s)
- B is the bandwidth of the channel (Hz)
- S/N signal-to-noise ratio ($\%$) with S the signal power and N the noise power (dB).

Reliability in transfers is one of the most fundamentally important problems in networking. There are many forms of unreliability, namely data loss, data corruption and packet reordering. Achieving reliability can be tackled from different angles, such as engineering transmission channels to improve physical performances , or mathematically, with Shannon's information theory (*cf.* [3]), statistics or reliability algorithms (Finite State Machines based, (*cf.* section 3.4 in [4]). Since unreliability can occur at all layers of the TCP/IP model, reliability mechanisms are usually implemented at the transport layer (the highest layer of the model, excluding the application layer). TCP employs reliability protocols (Go-Back-N or Selective Repeat) (RFC793 [5]) based on *ARQ (Automatic Repeat reQuest* queries (*RFC3366* [6]) which is compatible for communications between exactly 2 endpoints. By contrast, UDP implements no reliability mechanisms whatsoever.

B. Broadcasts

The most common way of spreading information in a network is using broadcasts. Broadcasting is a method in which a sender simultaneously transmits information to all the devices in the network. In practice, network layer protocols include broadcasting in their service-model. The UDP/IP broadcast respects the following scheme:

1) The application message is encapsulated in the transport layer with a broadcast flag set to true, thus forming a segment.
2) The network layers receives the segment. The network layer protocol, in our case IPv4, detects the broadcast flag while inspecting it and computes the **IPv4 broadcast address** of the network. The algorithm for computing the broadcast address is deemed out of scope for this report.
3) The segment is encapsulated with IPv4 and has the broadcast address set as its destination, thus forming a datagram.
4) The datagram is dispatched down to lower layers and sent over the network.

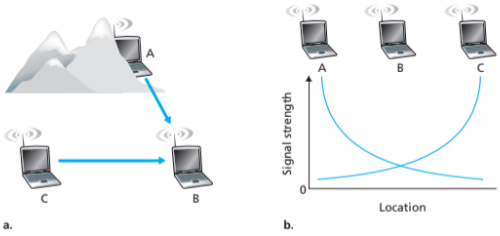
Crucially, we note that broadcasting involves a one-to-all communication, hence is not compatible with TCP, a protocol that operates between (exactly) 2 endpoints. As a result, broadcasts are only supported by connection-less and unreliable protocols such as UDP, in which packet delivery is not guaranteed.

Although theoretically fallible, UDP's *theoretical* unreliability for broadcasts is increasingly becoming acceptable given that network hardware has been considerably improving over the last decades. In fact, the cost of RAM today is cheaper than it has ever been before, and modern routers are gradually being equipped with larger buffers able to accept a tremendous amount of packets in their input queue before becoming saturated and dropping packets. Moreover, the switching fabric of routers has been heavily optimized with Banyan networks [7], a hardware component for switching packets from an input bus to an output bus, instead of using a standard Von Neumann architecture with a CPU and RAM. As a result, the input queue is cleared more quickly and a full buffer is less likely to occur [4]. As for other network components, wired links like Ethernet have been steadily upgraded (*cf.* appendix B, Ethernet evolution) over the time to uphold higher bandwidths (*cf.* [8]), thus decreasing the frequency of packets loss (in accordance with the *Shannon-Hartley theorem*), or collisions (*cf.* CSMA/CD Efficiency [4] and [9]- out of scope for this report).

Critically, these improvements are inapplicable in our case. The network is a flying ad hoc network in which network nodes are to be mounted on drones. Therefore, we employ raspberry PIs, light and compact computer-chips, to form the network. Compared to routers, the raspberry PI are unspecialized devices and do not possess a specially designed routing fabric such as a Banyan Network. Furthermore, the link layer protocol is required to be wireless since the nodes, flying drones, are highly mobile. Wireless links have a narrower transmission bandwidth due to the nature of their physical layer. Since wireless transmissions are a flood of radio modulations propagated through matter (*e.g* air), they are subject to signal interference from other sources as well as substantial signal strength attenuation, thus increasing the signal-to-noise ratio as the distance between sender and receiver increases (*cf.* appendix B signal attenuation and Shannon-Hartley eq.1). Finally, the nature of a flying ad hoc network introduces new transmission complications particular to its kind. For instance, rough geographic topology such as mountainous areas cause transmissions to incur the **Hidden terminal problem** (*cf.* [9]).

The figure III-B illustrates an example of the Hidden terminal problem occurring. We suppose that both station A and station C are transmitting to station B. In standard environments, A and C would detect each other's transmission and arrange a transmission turn to avoid collision at B (*cf.* Multiple Access protocols, specifically CDMA-CA [9]). However, the physical obstruction in the environment

Figure 2. The hidden terminal problem (*cf.* [9] and [4])



III-B may prevent A and C from hearing each other's transmissions, thus interfering at B.

To sum up, compared to standard network infrastructures, the likelihood of packet loss is increased in a mobile and wireless ad hoc network. The unreliability of UDP/IP transmissions is more pronounced and thus, broadcasted messages are more likely to be lost.

In this report, we will be presenting a broadcasting protocol that we designed, particularly adapted to resolve transmission complications in a flying wireless ad hoc network. The protocol aims to provide a **statistically reliable** broadcasting channel that operates on top of an error-prone and lossy channel that is the UDP/IP broadcast.

IV. Methods

A. Fundamentals

Although implemented as a software interface in the application layer, conceptually, the reliable broadcast protocol resides as an intermediate layer between UDP at the transport layer, and the imaging and tracking servers applications (examined in my colleague's report) at the application layer. The aim of the reliable broadcast protocol is to provide a statistically lossless broadcast channel built on top of a lossy channel. Fundamentally, we aim at minimizing the probability of losing a broadcasted packet by introducing transmission redundancies that take advantage of the geographically dispersed character of the network.

In an ad hoc network, it is frequent that 2 distant nodes A and B are not physical neighbors. In other terms, there doesn't exist a network link between A and B because each is out of scope of the other's antenna's range. We define the function inRange such that:

$$\text{inRange}(u,v) = \begin{cases} \bot & else \\ \top & \text{devices } u \text{ and } v \text{ are affiliated} \\ & \text{to the network and are in scope of} \\ & \text{each other's antenna range} \end{cases} \tag{2}$$

Let $G(N,E)$ be a graph representation of the ad hoc network, where N is the set of nodes, in our case raspberry PI devices, and E the set of edges connecting nodes such that:

$$\forall (u,v \in N) u \text{ — } v \iff \text{inRange}(u,v)$$

Let us consider a scenario where $\exists A,B \in N$ and A where to communicate with B. In an ideal ad hoc routing

protocol, the transmission from A to B would flow through an *internal relay (IR)* (*cf.* [10]) that relays the message of A to B such that:

$$IR(A,B) \coloneqq \underset{G}{\text{shortestPath}}(A,B) \setminus \{A,B\} \tag{3}$$

We note that, evidently, the communication is direct for 2 physical neighbors:

$$\forall (u,v \in N) IR(u,v) = \emptyset \iff \text{inRange}(u,v) \tag{4}$$

Naturally, the size of $IR(A,B)$ directly affects the probability of packet loss in a transmission from A to B. In fact, in a multi-hop route, the packet transits in multiple nodes and multiple links, each individually capable of losing the packet. **Proof**: Let us define $n = \#IR(A,B)$ and p the probability of losing a packet in a hop. For the sake of simplification, we consider a hop the union of a node and its entry link and that all hops are identical (same length of link, same CPU power, and so forth.). The probability of losing a packet transiting $IR(A,B)$ follows a binomial distribution $X \sim B(n,p)$ and is calculated as follows:

$$P[X \geq 1] = 1 - P[X = 0] = 1 - (1-p)^n \tag{5}$$

We note that:

$$\forall (n_1, n_2 \in \mathbb{Z}^+) \wedge X \sim B(n_1,p) \wedge Y \sim B(n_2,p) \wedge n_1 < n_2 \implies P[Y \geq 1] >= P[X \geq 1] \tag{6}$$

Inherently, if the geographical distance between A and B is important, $\#IR(A,B)$ is high and losses are more likely to happen. Similarly, the probability of having losses between A and B is minimal with an IR of minimal length, which is the case of physical neighbors with an $\#IR(A,B) = 0 \iff \text{inRange}(A,B)$. On the basis of the previous observations, the reliable broadcast relies on the *connectivity* of the graph G. Since B is connected to the network, there must exist a non-empty set of nodes Q to which B is directly connected, and that is able to reach B with minimal probability of loss:

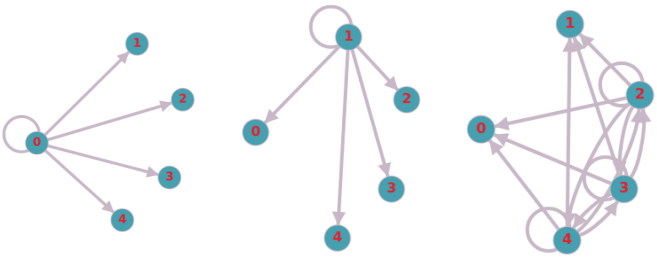
$$\exists Q \subset N \setminus \{B\} \wedge Q \neq \emptyset \wedge (\forall y \in Q \quad \text{inRange}(y,B)) \tag{7}$$

In other terms, the nodes of Q are geographically optimally placed to reach B, and the communication between both parties is intrinsically not subject to the hidden terminal problem. The approach taken to minimize losses seeking to maximize the probability of A successfully delivering the message to *each* node of Q. In that way, the probability of loss is heavily decreased. **Proof:** Assuming the packet arrived to all nodes of Q, a loss would require each of these optimally placed nodes to lose the packet. The probability of loss from the set Q to B would be $p^{\#Q}$.

Reaching any node $y \in Q$ is a recursive problem to which the same approach applies. Therefore, we can deduce the following simple rule on which the protocol will be based: Each node receiving a broadcasted message for the first time shall broadcast it back to the whole network, otherwise discard it.

We will call a repeated broadcast an *echo*, and the repeating node an *echoing node*.

Figure 3. Example flow of the echoing system



B. Elementary packet structure

The rule presented in the previous section is primitive and does not take into consideration practical use cases of networking. Firstly, a receiver is divested from his capability of deducing the origin of a message received, since the transmission might have been transferred by a echoing node and not the original sender directly. To counter this issue, each node is assigned a unique ID `nodeid`, to be included in its broadcasted packets to the network.

Secondly, due to the echoing system, a broadcasted message can remain in the network for a long time, bouncing off relaying nodes until each node that had received the message broadcasted it back. The long lifespan of messages ensues that receivers who initially didn't receive the message can potentially still receive it after a relatively long delay, without being aware of the said delay.

Finally, nodes could receive duplicates of a same broadcasted message as a result of neighboring echoes. In an attempt to uniquely identify messages, each node maintains a counter, `seq`, that is incremented and assigned as a *sequence number* on each new message to be sent. In this manner, other nodes are able to distinguish the sent message from other (new or duplicate) messages by looking at the pair (nodeID, sequence number). Accordingly, to prevent duplicates, each node maintains a hashmap `last_seq_map` storing, for each node in the network, the sequence number of the last message received. Then, messages are ruled in or out on the basis of the following algorithm:

```
bool isToBeIgnored(received_packet):
    alias rcv = received_packet
    alias last_seq_map = map
    if rcv.nodeid == self_nodeid:
        return true # ignore self tranmissions
    if not map[rcv.nodeid]:
        # first time we hear from node <nodeid>
        map.insert(rcv.nodeid, rcv.seqnum)
        return false
    elseif rcv.seqnum > map[rcv.nodeid].seqnum:
        # node <nodeid> broadcasted a new message
        map.update(rcv.nodeid, rcv.seqnum)
        return false
    else:
       # duplicate from echoes - ignore
       return true
```

Structure of the last_seq_map hashmap

nodeid (key)	last_seqnum (value)

Table I
Minimum required packet structure for reliable broadcast

NodeID	ID uniquely identifying the sender node
Sequence number	incremental number uniquely identifying the message from the node
Timestamp	time of submission using POSIX timestamp
Payload	the data to be communicated

Hence, the algorithm that runs in the receiver thread is the following:

```
while true:
    received_packet = socket_receive()
    if isTobeIgnored(packet):
        continue
    broadcast(received_packet)  # the echo
    processPacket(received_packet)
```

C. Optimizing node recovery schemes

While the algorithm for the `isToBeIgnored` function rightfully ignores irrelevant packets during the normal control flow, it does not prevent germane packets from getting ignored in a few edge cases. The first case occurs when the sequence number overflows. As a reminder, the sequence number of a message uniquely identifies it between other messages sent from a the same node. Each sender maintains a counter, `seq`, that is incremented and assigned as a sequence number on each new message to be sent. In practice, on 32-bit raspberry PI devices, `seq` is an `unsigned int` ranging $[0, 2^{32} - 1] \cap \mathbb{N}$. After a node sends its $2^{32}th$ message `seq` overflows and is reset to 0. This causes the node to be ignored *in perpetuum* by all other nodes of the network, since according to the other nodes' `last_seq_map`, the conditions `rcv.seqnum > map[rcv.nodeid].seqnum` or `not map[rcv.nodeid]` is false (*cf.* algorithm `isToBeIgnored`). We will call this phenomenon *network social exclusion.*

The other case occurs when a node needs to be rebooted in the event of a software crash. We consider a scenario where a node x crashed and its last `seq` was $seq = i$ with $0 < i < 2^{32}$. After rebooting x, the counter `seq` is reset to 0 causing x to be subject to a *temporary* network social exclusion, where the messages of x get ignored until `seq>i`.

Network social exclusions are resolved by introducing a constant maximum age, `MAX_AGE`, for each entry in the `last_seq_map`. When an entry is inserted or updated, its age is (re)set to 0. The age is then periodically incremented by 1 every 1 second until it is reset again by the next entry update, which happens when the node having the entry's `nodeid` sent a new message. On the event of an entry's age exceeding `MAX_AGE`, the entry is simply deleted from the map, thus avoiding any potential network social exclusion.

I summary, the algorithm is implemented in a parallel thread and operates on `last_seq_map` for which the new structure is:

nodeid (key)	<last_seqnum, age > (value)

The pseudo-code of the algorithm is the following:

```
while true:
    for entry in last_seq_map:
        if entry.age == MAX_AGE:
            last_seq_map.delete(entry)
        else:
            entry.age += 1
    sleep 1sec
```

An exclusion of a node can last in worst cases `MAX_AGE`. The choice of `MAX_AGE` needs to be made adequately, given that a high value slows down social inclusion, whereas a low value might rule delayed duplicate messages to be processed instead of being ignored. Studies on the choice of `MAX_AGE` is out of scope for this report. A value of 30 seconds has been set in our implementation.

D. Implementation

A UML diagram representing our (`C++`) implementation is included as an appendix (*cf.* appendix A UML diagram). The protocol is implemented in a generic way. Applications (tracking server and imaging server covered by my colleague) required to benefit from the reliable broadcast need to inherit from the relevant class. We note that in the previously presented algorithms, we have hidden concurrency related instructions (such as mutexes locks) for simplicity's sake.

V. Results and discussion

A. Statistical Analysis of robustness and reliability

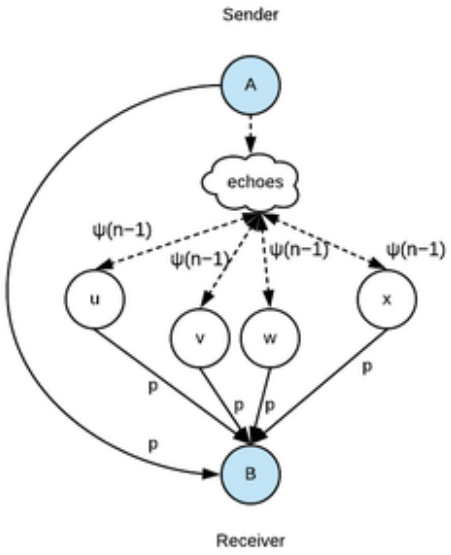
Due to the distributed character of the algorithm, the reliability of delivery is directly affected by the number of nodes affiliated to the network.

We consider a subset of the network G represented by the sub-graph $G'(N', E')$, such that:

$$\begin{gathered} N' = N \setminus \{B\} \\ \forall u', v' \in N' \wedge \forall u, v \in N \wedge u' \equiv u \wedge v' \equiv v \wedge \\ u' \text{ — } v' \implies u \text{ — } v \end{gathered} \tag{8}$$

Let p be the probability of losing a packet in the ad hoc network during a one-to-one transmission. Let $\psi(n-1)$ be the probability of a node not receiving a packet in an ad hoc network of $n-1$ nodes (G') while using reliable broadcast ($(n-1)^2$ transmissions). In a scenario where the node A broadcasts a message, we will examine, from the point of view of B, the recursive probability $\psi(n)$ of not receiving the message.

Looking at the figure V-A, there are 3 cases in which a packet expected to arrive at B is lost. It is required for all of the cases to occur for B not to receive the message.



- The direct transmission from A to B is lost. This is the original broadcast of A to the network. We note that in a standard broadcast, this is the only means of communication between A and B. The loss event happens with a probability of p (by definition).

  For each of the $n-2$ echoing nodes (namely u, v, w, x):
  - The echoing node didn't receive the packet, hence, cannot echo it to B. This event happens with a probability of $\psi(n-1)$ (by definition).
  - The echoing node did receive the package, yet, the packet is lost during the direct transmission from the echoing node to B. This event happens with a probability of $(1-\psi(n-1)) \cdot p$.

Therefore, the probability of losing a packet from the point of view of B is written as $\psi(n) : [2, \infty[\cap\mathbb{N} \mapsto [0, 1]$:

$$\psi(n) = p \cdot [\psi(n-1) + (1-\psi(n-1)) \cdot p]^{n-2} \qquad (9)$$

The formulation of ψ respects the 3 axioms of Kolmogorov, confirming its fidelity to the theory of probability (*cf.* [11]). The proof is deemed out of scope for this report.

In contrast, for a similar scenario using the standard broadcast, the probability of losing a packet from the angle of B is p (by definition of p).

Therefore, the reliable broadcast is equally or more reliable than the standard broadcast:

$$\forall(n \in [2, \infty[\cap\mathbb{N}) \quad \psi(n) <= p \qquad (10)$$

Proof: Calculating $\frac{\psi(n)}{p} = [\psi(n-1) + (1-\psi(n-1)) \cdot p]^{n-2}$. $\psi(n-1) <= 1$ and $(1-\psi(n-1)) \cdot p <= 1$ because they are a probability (by definition). Proving that $\psi(n-1) + (1-\psi(n-1)) \cdot p$ is lesser or equal to 1 is done by posing the *extremum* $p = 1$, yielding $\psi(n-1) + (1-\psi(n-1)) = 1$. Therefore, $\frac{\psi(n)}{p} <= 1$ and $\psi(n) <= p$.

1) Observations: The plot in appendix A fig.5 compares the reliability of the reliable broadcast against the standard UDP/IP broadcast. The computations are done for various environments, with a varying number of nodes and base probability of loss. The figure show that for a higher number of nodes, the reliable broadcast performs significantly better. The observations let us expect that $\lim_{n\to\infty} \psi(n) = 0$. This can indeed be demonstrated by proving that:

$$\forall(n \in [2, \infty[\cap\mathbb{N}) \quad (\psi(n) >= 0 \wedge \frac{\partial\psi(n)}{\partial n} <= 0)$$

(The detailed proof is deemed out of scope for this report.) However, these computations don't take network congestion in consideration which is presumed very high for $n \to \infty$.

B. Weaknesses and potential solutions

Compared to the UDP/IP broadcast, messages broadcasted with the reliable broadcast generate high activity throughout the network. Let us consider a network composed of a relatively high number of nodes, all of which are geographically closely positioned, we will call such a network a **dense** network. In a dense ad hoc network, the latency of communications is decreased. Consequently, the responsiveness of nodes is increased and the delay between echos is decreased. A broadcasted message on such a network generates causes echoes to be transmitted (practically) simultaneously on links, generating high traffic which can result in network congestion. Given that our implemented network consists of $n = 4$ raspberry PI devices, these circumstances could not have been tested. Although our network showed no signs of congestion, a potential solution to prevent it is to **artificially** force a delay between echoes. This is done by a having each node wait a random amount of milliseconds before echoing back to the message to the network.

Furthermore, the current reliable broadcast protocol only corrects data loss but not data corruption or packet reordering. The latter is inconsequential for our requirements since the system is stateless: the data to communicate is self-contained in one packet and messages do not depend on each others. As for data corruption, there is room for improvements potentially by implementing error correcting codes (*cf.* [2], more specifically [12]). ARQ queries (*cf.* [6]) might not be adapted in our case as it encourages congestion.

VI. Conclusion

IN CONCLUSION, in this report we present a broadcast protocol to improve the reliability and robustness of broadcast transmissions in geographically dispersed wireless ad hoc network. The reliable broadcast takes advantage of the geographically dispersed character of the network to decrease the likelihood of losses and circumvent unusual problems such as the hidden terminal problem, which are unique to wireless networks. Although the

approach taken is statistical, it approximates optimality for a reasonable amount of nodes, and is as optimized as the standard UDP/IP broadcast when the number of nodes approaches infinity. The protocol implemented is a primary version, and there is naturally room for revisions and enhancements: Further studies can be made as to (for instance) the congestion these transmissions induce, the geographical position of the nodes to maximize entropy (*cf.* [2]), or the integration of error correcting codes.

References

[1] Gary Stoneburner, Clark Hayden, and Alexis Feringa. Engineering principles for information technology security (a baseline for achieving security). page 32, 06 2001.

[2] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[3] I.M. Jacobs J.M. Wozencraft. *Principles of Communication Engineering*. John Wiley Sons, 1965.

[4] Keith W Ross James F Kurose. *Computer networking : a top-down approach*. Pearson, 2017.

[5] TRANSMISSION CONTROL PROTOCOL. RFC 793, September 1981.

[6] L.Wood G.Fairhurst. Advice to link designers on link Automatic Repeat reQuest (ARQ). RFC 3366, August 2002.

[7] Jae-Hyun Park, Hyunsoo Yoon, and Heung-Kyu Lee. The deflection self-routing banyan network: A large-scale atmswitch using the fully adaptive self-routing and its performanceanalyses. *Networking, IEEE/ACM Transactions on*, 7:588–604, 09 1999.

[8] Charles E. Spurgeon. *Ethernet, The Definitive Guide*. O'Reilly Media, Inc, 2000.

[9] R. Michael Buehrer. *Code Division Multiple Access (CDMA)*. Morgan Claypool, 2006.

[10] David Johnson and David Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Comput.*, 353, 05 1999.

[11] A.N Kolmogorov. Foundations of the theory of probability. *Chelsea Publishing Co.*, 1950.

[12]

[13] Andrej Hrovat, Gorazd Kandus, and Tomaz Javornik. Path loss analyses in tunnels and underground corridors. *International journal of communications*, 6:136–144, 08 2012.

[14] Jozef Chovan and F. Uherek. Photonic integrated circuits for communication systems. *Radioengineering*, 27:357–363, 06 2018.

Appendix A
Appendices

Figure 4. UML diagram representing our implementation of reliable broadcast

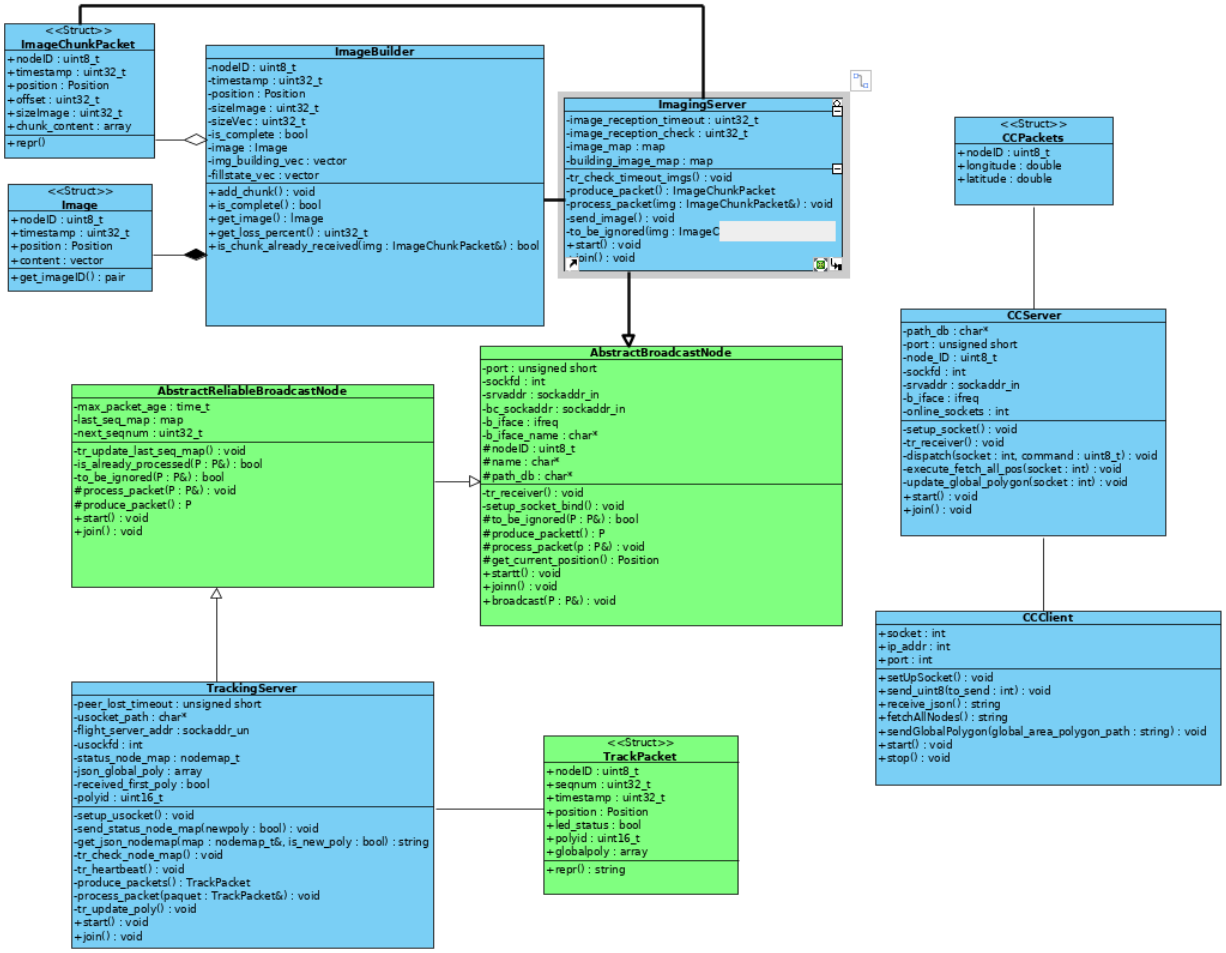
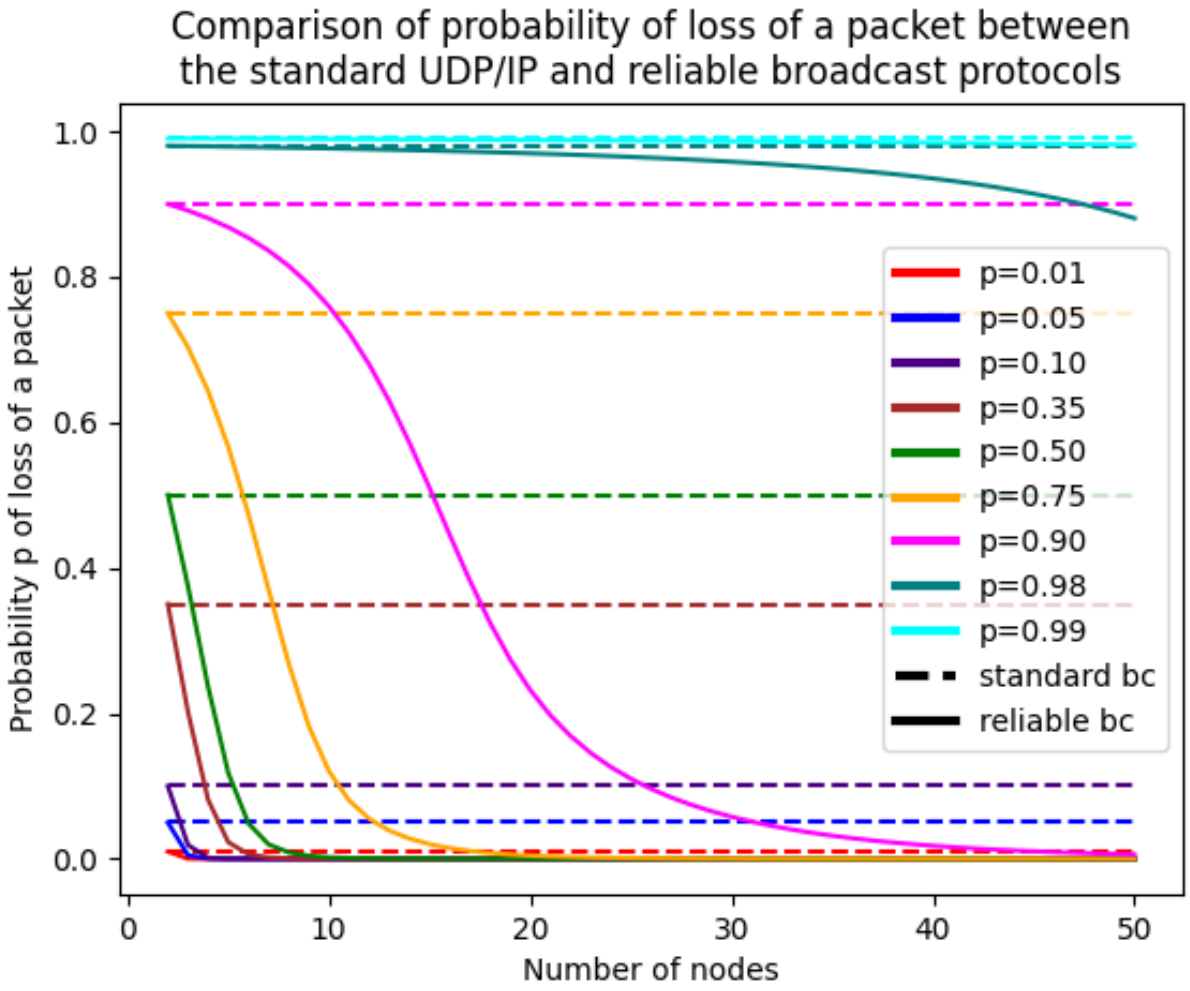


Figure 5.



Appendix B
Extra appendices

Figure 6. Wireless signal attenuation over distance (*cf.* [13])

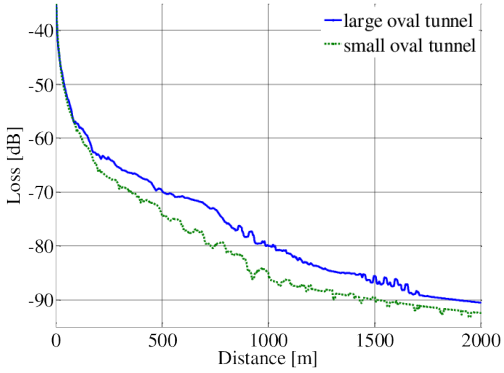
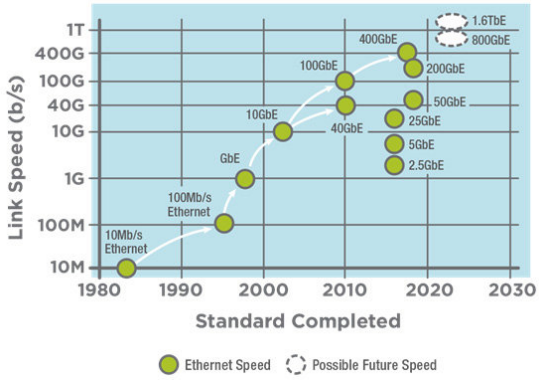


Figure 7. Evolution of Ethernet performance over the years [14]



Listing 1. Python script used for plotting fig. 5

```
from matplotlib import pyplot as plt
from matplotlib.lines import Line2D
import numpy as np


class RBCLossCalculator:
    def __init__(self, std_p_loss):
        self.p = p
        self.cache = dict()

    def __call__(self, n):
        assert n>=2  # function only defined on [2,+inf[

        if (n-2) == 0:
            return self.p

        if n in self.cache.keys():
            return self.cache[n]

        if (n-1) in self.cache.keys():
            fn_1 = self.cache[n-1]
        else:
            fn_1 = self(n-1)
            self.cache[n-1] = fn_1
        return p * (fn_1 +  p - fn_1*p )**(n-2)  # psi(n)


N=50          # max number of nodes in the network
probas = (0.01,  0.05,  0.1, 0.35, 0.5, 0.75, 0.9, 0.98, 0.99 )

colors = ('red', 'blue', 'indigo', 'brown', 'green', 'orange', 'magenta', 'teal', 'cyan')

x = np.arange(2,N+1)

# plotting for reliable broadcast
for p, clr in zip(probas, colors):
    phi = RBCLossCalculator(std_p_loss=p).__call__
    vphi = np.vectorize(phi.__call__)
    plt.plot(x, vphi(x), color=clr, label="␣rel␣bc␣p=%.2f"%p)

    # plotting for standard broadcast
    stdbc = np.empty(N-1)
    stdbc.fill(p)
    plt.plot(x, stdbc, color=clr, linestyle="dashed", label="std␣bc␣p=%.2f"%p)


plt.xlabel('Number␣of␣nodes')
plt.ylabel('Probability␣p␣of␣loss␣of␣a␣packet')
plt.title("Comparison␣of␣probability␣of␣loss␣of␣a␣packet␣between\nthe␣standard␣UDP/IP␣and␣reliable␣broadcast␣protocols")

# constructing the legend
lines = [Line2D([0], [0], color=c, linewidth=3) for c in colors]
lines.append(Line2D([0], [0], color='black', linewidth=3, linestyle='--'))
lines.append(Line2D([0], [0], color='black', linewidth=3))
labels = ["p=%.2f"%p for p in probas]+["standard␣bc", "reliable␣bc"]
plt.legend(lines, labels)

#plt.savefig('plot.png')
#plt.show()
```