A Passive Multi-Channel Synchronization Solution for IoT

Keyi Zhang
kz005@bucknell.edu
Bucknell University
Lewisburg, PA

ABSTRACT

The Internet of Things (IoT) promises to allow everyday objects to connect to the Internet and interact with users and other machines ubiquitously. Regional networks such as LoRa and Sigfox have been deployed to provide connectivity to end devices, where multiple channels are used to increase network capacity. However, channel synchronization can induce significant overhead as devices join and leave the network. In this paper, we present a passive and fault-tolerant multi-channel synchronization solution, McSync, that requires no communication with the base station while achieving low join latency.

CCS Concepts

•**Networks** → **Network protocol design;**

Keywords

Internet of Things; Channel Synchronization; Low Power Network

1. INTRODUCTION

The Internet of Things (IoT) continues to grow as a new paradigm in which information and communication systems are embedded in our surroundings, enabling new services and applications such as including intelligent healthcare, environment monitoring, and smart cities. The IoT ecosystem spans monitoring, storage, communication, and analytical tools. However, creating a scalable and robust communication system for IoT is essential for success [3].

In typical wireless IoT application deployments where a star topology is used, devices need to synchronize time and channel hopping pattern to the base station. For a large scale IoT networks, the overhead for synchronization can be huge especially when a number of devices join the network then leave. For instance, in big cities, people commute across different network deployment regions for work. The IoT devices carried with them have to join and leave different networks on daily basis. In addition, to reduce energy consumption and increase signal reception, many IoT applications utilize low power networks with low data rates. Therefore, the synchronization overhead can be detrimental for the overall performance of the device. Furthermore, many manufacturers have to hard-code necessary information to join the network, such as LoRa [5], resulting in security risks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

S3'16, October 03-07, 2016, New York City, NY, USA

© 2016 ACM. ISBN 978-1-4503-4255-1/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2987354.2987366

In addition, many popular IoT networks employ an ultra narrow band PHY layer, which offers hundreds of available channels. For instance, a FCC compliant network operating at sub 1 Gz can use frequencies between 902-928 MHz. If the network uses a 12.5 KHz channel, it will have 2080 available channels. To avoid interference among adjacent regions, available channels are often partitioned and different subset of channels are used in different regions [6]. However, such practices impose another challenge for IoT devices, to discover the available channels in the current region, either through active communication or apriori information.

In this paper, we will present a novel solution for multi-channel synchronization called McSync. McSync borrows ideas from finite fields. There are several existing works using finite fields to support channel hopping, such as Disco [2], SSCH [1] and CHS [4]. These algorithms exhibit guaranteed network connectivity, yet they are either designed for ad-hoc networks or not suitable for networks suffering high packet loss. To our knowledge, there is no prior research on star-topology based multi-channel synchronization algorithm. McSync uses passive listening and does not have to transmit packets to discover and join the network.. Therefore it is suitable for a large scale IoT network where a multitude of devices can join the network dynamically.

2. ALGORITHMS AND ANALYSIS

McSync has two steps, namely, channel discovery and network joining. Channel discovery allows devices to discover which subset of the available channels the current region is using. Additionally, channel joining enables devices to synchronize the channel hopping patterns of base stations located within the region. McSync uses finite fields and the operations are defined as integer addition and multiplication with modulo. Without explicit notation, every arithmetic operation in this paper is a ring operation. For instance, we will drop the mod notation and $a = b^{-1}$ implies $ab \equiv 1$.

McSync can be applied to any PHY that supports multiple channels and is suitable for any TDMA-type MAC layer protocol. It also assumes the network is fully occupied by traffic; if not, the base station can broadcast random packets into the channel to fill in any unused timeslots. During each step, the devices needs to detect the preamble, which will be marked as received a packet. If a corrupted preamble not no preamble is received, it will be marked as packet loss.

2.1 Channel Discovery

Let p be a prime number, such that the total number of available channels is slightly greater than p^2. Let the set $\{0, 1, \ldots, p^2 - 1\}$ be our channels in consideration, denoted as F. It is easy to show that there are $p^2!$ bijections between actual frequencies and elements in F, hence we can use the bijection as a shared secret within the network known to all devices. For simplicity, we use elements in F to refer to a actual channel frequency.

As each region only uses a subset of F, we need to partition F in a way easy for devices to compute the available local channels. Let the number of base stations in the region be n, and packet loss rate be pl. We use a quotient field to partition the set. Recall that F is a finite field and $|F| = p^2$, by Lagrange's Theorem, the order of any non-empty proper subfield is p. Without loss of generality, we will use the simplest subfield $H = \{0, p, \ldots, p^2 - p\}$. Although a quotient field can be obtained multiplicatively, we will use addition for simplicity in this paper. Hence the quotient field $F|H$ is $F|H = \{\{a + pi | i \in F\} | a \in F\}$. Each adjacent region will be assigned to a different subfield to avoid interference as partition guarantees the disjointness as long as $n < p$. Once we know how the channels are partitioned, we can proceed to channel discovery, as shown in Theorem 1 and Algorithm 1. Although Theorem 1 only provides a upper bound, since there are n base stations using the same subset, the expected time to discover local channels can be determined as $\frac{p^2}{2n(1-pl)}$.

THEOREM 1. *With high probability it takes no more than p^2 units of time for a device to determine the quotient field.*

PROOF. It is easy to show that for all quotient field $H' = \{a + pi | i \in F\}$, $H' \cap H = \{a\}$, where $a < p$. Hence listening to set H suffices. Since channel a can be any time slot within the cycle, the device needs to listen to each possible cycles by shifting each cycle by 1. Therefore at most p^2 unit time the device will receive a signal from channel a. Taking packet loss into account, successfully receiving a packet forms a geometric distribution. Hence the probability to determine the local quotient field within p^2 time is thus $1 - pl^n$. □

Algorithm 1 Algorithm for channel discovery

```
procedure ChannelDiscovery
    i ← 0
    for j ← 0, p^2 do
        if transmission detected at channel i then
            a ← i
            break
        else
            if thenj ≠ 0 and j mod p = 0
                i ← i + 2
            else
                i ← i + 1
            end if
        end if
    end for
    return {a, a + p, ..., a + (p − 1)p}
end procedure
```

2.2 Network Joining

Once the device determines the channel subset, it can then begin the joining process. For each region, all the base stations share a secret number b and each of them will choose distinct channel offsets $a_1, \ldots, a_n$. The channel hopping pattern is defined as $f(i) = a + bi$.

Without loss of generality, we label each channel from the set $\{0, 1, \ldots, p-1\}$. The device will pick two distinct random channels, denoted as x_1, x_2 and listen at each channel for a full cycle (p unit times). If there is no packet loss, we can show that Theorem 2 will guarantee the algorithm's efficiency. Furthermore, even if there exists packet loss, the algorithm is still efficient when $pl \ll 1$ and n is small.

THEOREM 2. *If $x_2 \neq x_1$, it takes no more than $2p$ to determine the hopping pattern, that is $a_1, a_2, \ldots a_n$ and b, provided no packet loss.*

PROOF. We have two sets of equations $a_x + bi_x = x_1$ and $a_x + bj_x = x_2$ for each $x \in \{1, \ldots, n\}$, where i, j are the time receiving packet from x_1 and x_2 respectively. Notice that j_x does not correspond to i_x (e.g. $i_1 \neq j_1$). However, notice that $\sum_x (a_x + bjx) - \sum_x (a_x + bi_x) = \sum_x b(j_x - i_x) = n(x_2 - x_1)$. Therefore $b = n(x_2 - x_1)\big(\sum_k (j_x - i_x)\big)^{-1}$. After obtaining b we can compute the $a_1, \ldots, a_n$ values easily. For instance, $a_1 = x_1 - bi_1$ and so on. □

However, in reality, packet loss happens due to problems such as interference. Theorem 2 only applies to low packet loss environment. Using a probabilistic approach, we can compute b and a values chosen by each base stations, as shown in Algorithm 2.

Algorithm 2 Algorithm for network joining

```
procedure JoinRegionalNetwork
    x_1, x_2 = random(p) and x_1 ≠ x_2
    list_1, list_2 ← list()
    b_list ← [0, 0, ..., 0] // size p
    for i ← 0, p − 1 do
        if transmission detected at channel x_1 then
            list_1.insert(i)
        end if
    end for
    for i ← 0, p − 1 do
        if transmission detected at channel x_2 then
            list_2.insert(i)
        end if
    end for
    for i ← 0, p − 1 do
        list_copy ← list_2 − i // shifting list_2 by i
        count ← |list_1 ∩ list_copy|
        b ← (x_2 − x_1)i^{-1}
        b_list[b] ← count
    end for
    b ← ReceiveTwice(b_list) // testing b values
    for k ← 0, length(list_1) − 1 do
        i_k ← list_1[k]
        a_k ←= x_1 − b i_k
    end for
    return {b, a_1, a_2, ...}
end procedure
```

The essential idea behind Algorithm 2 is to recognize that all the base stations share the same b value yet different a values. Therefore each hopping frequency is shifted by some offset in the temporal domain. Hence, if we can enumerate all the possible offset shifts, we can find offsets that has the most matches and use that to calculate possible b values. Although false positive reception may interfere with the results, as the matching depends on the number of base stations n, the bigger n is, the lower latency and more accurate McSync will be.

A graphic illustration of listening and shifting is shown in Figure 1. In Figure 1, there are $n = 3$ base stations and $p = 7$. Each base station chooses a different channel offset, namely $1, 2$ and 4. The device does not receive a packet at time $i = 4$ in the first cycle. To apply the algorithm, we have to shift the second cycle and find the maximum number of matches. Since $shift = 4$ has the most matches, we have $b = (x_2 - x_1)i^{-1} = (1 - 6)4^{-1} = 2 \times 2 = 4$, as desired. Therefore the channel offset value a for a base station is $a = x_1 - bi = 6 - 1 \times 4 = 2$, which is base station 2. Other channel offsets can be obtained in a similar way.

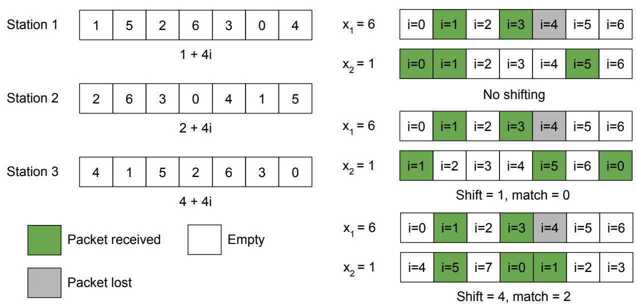

Figure 1: Example of channel joining process with available channels $p = 7$.

One caveat of Algorithm 2 is that the device may still suffer from packet loss when testing b values, as indicated in Line 21. One approach to solve the problem is to try to receive a packet from the base station with the strongest signal strength, twice before moving to the next test. Even for a bad link quality ($pl = 0.3$), receiving twice will increase the success rate rate to $1 - pl^2 = 0.91$. If receiving twice still fails, the device will try the next b value. Since we miss the correct parameter value, the device needs to test another $2p$ cycles before receiving packet from the correct channel again, although it has very low probability. The procedure of using receiving twice strategy to compute b is shown in Algorithm 3. Notice that this algorithm can be extended to receiving multiple times easily given undesirable link quality.

Algorithm 3 Algorithm for receiving twice

```
procedure ReceiveTwice(b_list)
    i ← 0
    while not resolved do
        b ← max_index(b_list, i)
        for j ← 0, 1 do
            test_channel = x_1 − b list_1[0] + bi
            if receive from channel x_1 then
                return b
            end if
            i ← i + 1
        end for
    end while
end procedure
```

2.3 Evaluations

Since McSync allows ideally infinite number of device simultaneously to join the network, we can focus on the performance of a single node. Figure 2 shows the simulated time for a node to join a regional network with various base stations. We assume there are $p = 37$ channels available regionally, thus yielding $p^2 = 1369$ total available channels globally. We measure the time from device starting up to ultimately joining the regional network, including channel discovery and joining time. As we can see from Figure 2, deploying more base stations in the region can significantly reduce the latency to join the network, allowing better performance in heavily deployed region such as populated cities.

To make the joining protocol independent, time is measured in number of time slots. Therefore, for the 900 MHz USA ISM band, each unit time is at most 200 ms (FCC regulation), giving us approximately 140 seconds for devices with packet loss rate $pl = 0.5$ to join the network where 4 base stations are located.

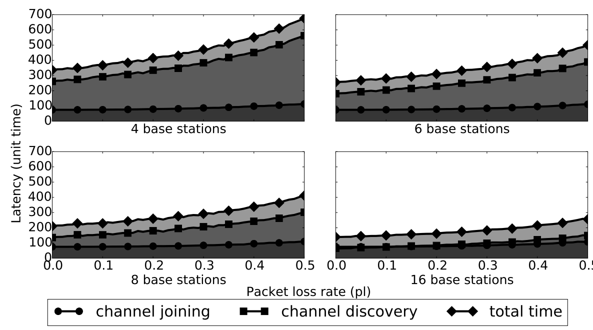

Figure 2: Average Joining time with various loss rate pl and base stations using receiving twice strategy (channel size $p = 37$, $n = 4$).

3. CONCLUSION AND FUTURE WORK

In this paper, we present McSync for multi-channel synchronization with high performance channel discovery and channel joining. McSync requires zero packet transmissions to join networks by passively overhearing the traffic. Hence it is suitable for a large scale regional IoT networks with thousands of connected devices. Future work includes real world deployment and performance benchmarking.

4. REFERENCES

[1] P. Bahl, R. Chandra, and J. Dunagan. Ssch: slotted seeded channel hopping for capacity improvement in ieee 802.11 ad-hoc wireless networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, 2004.

[2] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 2013.

[4] T.-Y. Lin, K.-R. Wu, and G.-C. Yin. Channel-hopping scheme and channel-diverse routing in static multi-radio multi-hop wireless networks. *IEEE Transactions on Computers*, 2015.

[5] N. S. (Semtech), M. L. (Semtech), T. E. (IBM), T. K. (IBM), and O. H. (Actility). LoRaWAN specification. Technical report, LoRa Alliance, January 2015.

[6] A. P. Subramanian, H. Gupta, S. R. Das, and J. Cao. Minimum interference channel assignment in multiradio wireless mesh networks. *IEEE Transactions on Mobile Computing*, Dec 2008.