University
of Basel

Managing and Distributing Software Updates Using Append-Only Logs

Maximilian Barth <m.barth@unibas.com>

Departement of Mathematics and Computer Science

Examiner: Prof. Dr. Christian Tschudin

Supervisor: MSc. Fabrizio Parrillo

12.07.2022

Outline

1. Goal
2. Tinyssb
3. Versioning System
4. Demo
5. Outlook

Scenario

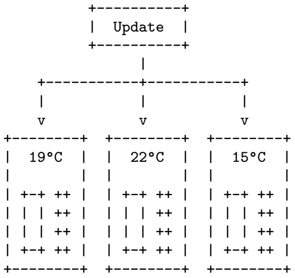
Sensor network

- Sensors are solar powered
- Irregular operating hours
- Nodes communicate via LoRa
- Limited processing power and memory

```
+---------+         +---------+
|  17°C   |         |  23°C   |
|         |         |         |
| +-+  ++ |         | +-+  ++ |
| | |  ++ |<------->| | |  ++ |
| | |  ++ |         | | |  ++ |
| +-+  ++ |         | +-+  ++ |
+---------+         +---------+
      ^                   ^
      |                   |
      |                   |
      v                   v
+---------+         +---------+
|  19°C   |         |  20°C   |
|         |         |         |
| +-+  ++ |         | +-+  ++ |
| | |  ++ |<------->| | |  ++ |
| | |  ++ |         | | |  ++ |
| +-+  ++ |         | +-+  ++ |
+---------+         +---------+
```

Goal

Distribute and **manage** updates across a solar-powered sensor network.
⇒ consider limitations of LoRa and hardware



```
         +----------+
         |  Update  |
         +----------+
               |
    +----------+----------+
    |          |          |
    v          v          v
+--------+ +--------+ +--------+
|  19°C  | |  22°C  | |  15°C  |
|        | |        | |        |
| +-+ ++ | | +-+ ++ | | +-+ ++ |
| | | ++ | | | | ++ | | | | ++ |
| | | ++ | | | | ++ | | | | ++ |
| +-+ ++ | | +-+ ++ | | +-+ ++ |
+--------+ +--------+ +--------+
```

Challenges

Network protocol must allow nodes to **catch up** with missed messages.

- Important for updates
- Still guarantee message authenticity and integrity

⇒ Append-only log protocol **Tinyssb**

github.com/tschudin/tinyssb

Tinyssb

- Append-only logs, so-called **feeds**
- Each feed entry (or packet) is **128B**
- Only the owner of a feed can append new **packets**
- Every node can **verify** a packet's authenticity and integrity
- Nodes request **missing** packets
- Feeds can have **child feeds**

```
bd27a9ff...
+--------+----------+----------+----------+
| Anchor | Packet 1 | Packet 2 | Packet 3 |
+--------+----------+----------+----------+


46ae8988...
+--------+----------+------------+----------+
| Anchor | Packet 1 | Make child | Packet 3 |
+--------+----------+------------+----------+
                           |
               +-----------+
ad6063dc...    v
+--------+----------+----------+
| Anchor | Is child | Packet 2 |
+--------+----------+----------+
```

Versioning system

Idea:

- Continuous code deployments
- Allow reversion of updates
- Enable users to create different update branches (similar to Git)
- Provide GUI for interaction with versioning system

```
+----------+
| main.py  |
+----------+
      |
      |
      v
+----------+
| update 1 |-----------+
+----------+           |
      |                |
      |                |
      v                v
+----------+     +----------+
| update 2 |     | update 4 |
+----------+     +----------+
      |
      |
      v
+----------+
| update 3 |
+----------+
```

Tinyssb and the versioning system

- The **update feed** contains all the information of the versioning system

```
Update feed
+-------+
|anchor|
+-------+
```

Tinyssb and the versioning system

- The **update feed** contains all the information of the versioning system
- Its first child feed is the **version control feed**, which contains the currently applied version number of each monitored file

```
Update feed
+------+----------+
|anchor|make_child|
+------+----------+
             |
             v

Version control feed
+------+----------+-------------------+
|anchor| is_child |apply v1 of main.py|
+------+----------+-------------------+
```

Tinyssb and the versioning system

- The **update feed** contains all the information of the versioning system
- Its first child feed is the **version control feed**, which contains the currently applied version number of each monitored file
- All remaining child feeds are **file update feeds**, each of which contains updates and their dependencies

```
Update feed
+------+----------+----------+
|anchor|make_child|make_child|
+------+----------+----------+
             |          |
             v          |
                        |
Version control feed
+------+----------+-------------------+
|anchor| is_child |apply v1 of main.py|-------+
+------+----------+-------------------+       |
                                              |
                        |                     |
             +----------+                     |
             v                                |
File update feed of main.py                   v
+------+----------+----------+----------+----------+
|anchor| is_child |make_child| main.py  |update: v1|
+------+----------+----------+----------+----------+
                        |
                        v
```

Bad updates

What happens in the case of bad updates?

- Already appended updates cannot be removed
- Large updates may congest the network

⇒ This must be handled

bad_update.c

```c
int main() {
  int x = 0;

  int* ptr = x;
  *ptr = 3;
}

```

Emergency feeds

Every file has an emergency feed, which can be activated through the following steps:

1. Create a new emergency feed
2. Append the file name to the old emergency feed
3. Append the emergency update

⇒ The emergency feed becomes the new file update feed.

```
File update feed
+------+----------+----------+----------+----------+
|anchor| is_child |make_child|main.py v0|update: v1|
+------+----------+----------+----------+----------+
                         |
                +--------+
Emergency feed  v
+------+----------+
|anchor| is_child |
+------+----------+
```

Emergency feeds cont'd

```
                        +--------+----------+------------+------------+------------+
File update feed        | anchor | is_child | make_child | main.py v0 | update: v1 |
                        +--------+----------+------------+------------+------------+
                                                   |
                                       +-----------+
                                       v
                        +--------+----------+
Emergency feed          | anchor | is_child |
                        +--------+----------+
```

Emergency feeds cont'd

```
                        +--------+----------+------------+------------+------------+----------------+
File update feed        | anchor | is_child | make_child | main.py v0 | update: v1 | bad update: v2 |
                        +--------+----------+------------+------------+------------+----------------+
                                                   |
                                       +-----------+
                                       v
                        +--------+----------+
Emergency feed          | anchor | is_child |
                        +--------+----------+
```

Emergency feeds cont'd

```
                        +--------+----------+------------+------------+------------+----------------+
Old file update feed    | anchor | is_child | make_child | main.py v0 | update: v1 | bad update: v2 |
                        +--------+----------+------------+------------+------------+----------------+
                                                   |
                                       +-----------+
                                       v
                        +--------+----------+------------+------------+
New file update feed    | anchor | is_child | make_child | main.py v2 |
                        +--------+----------+------------+------------+
                                                   |
                                       +-----------+
                                       v
                        +--------+----------+
New emergency feed      | anchor | is_child |
                        +--------+----------+
```

Emergency feeds cont'd

```
                        +--------+----------+------------+------------+------------+----------------+
Old file update feed    | anchor | is_child | make_child | main.py v0 | update: v1 | bad update: v2 |
                        +--------+----------+------------+------------+------------+----------------+
                                                   |
                                       +-----------+
                                       v
                        +--------+----------+------------+------------+----------------+
File update feed        | anchor | is_child | make_child | main.py v2 | fix update: v3 |
                        +--------+----------+------------+------------+----------------+
                                                   |
                                       +-----------+
                                       v
                        +--------+----------+
New emergency feed      | anchor | is_child |
                        +--------+----------+
```

$\Rightarrow$ ignore bad update

Representing updates

How should updates be encoded?
⇒ consider low memory and data rate

Three approaches:

1. Send the entire new file
2. Only send lines that have changed
3. Only send substrings that have changed

```
 original file           updated file
+----------------+     +----------------+
|1| x = 3        |     |1| x = 7        |
|2| y = x + 4    |     |2| y = x + 4    |
|3| print(x)     |---->|3| print(x + y) |
|4|              |     |4|              |
+----------------+     +----------------+
```

Representing updates cont'd

```
                      original file          updated file
                     +----------------+     +----------------+
                     |1| x = 3        |     |1| x = 7        |
                     |2| y = x + 4    |     |2| y = x + 4    |
                     |3| print(x)     |---->|3| print(x + y) |
                     |4|              |     |4|              |
                     +----------------+     +----------------+

+----+-------------------+    +----+-------------------+    +----+-------------------+
| 1. |                   |    | 2. |                   |    | 3. |                   |
+----+                   |    +----+                   |    +----+                   |
|  delete:               |    |  delete:               |    |  delete:               |
|                        |    |                        |    |                        |
|  1| x = 3              |    |  1| x = 3              |    |  line 1, pos 4: "3"     |
|  2| y = x + 4          |    |  3| print(x)           |    |                        |
|  3| print(x)           |    |                        |    |                        |
|  4|                    |    |                        |    |                        |
|                        |    |                        |    |                        |
|  insert:               |    |  insert:               |    |  insert:               |
|                        |    |                        |    |                        |
|  1| x = 7              |    |  1| x = 7              |    |  line 1, pos 3: "7"     |
|  2| y = x + 4          |    |  3| print(x + y)       |    |  line 3, pos 6: " + y" |
|  3| print(x + y)       |    |                        |    |                        |
|  4|                    |    |                        |    |                        |
+------------------------+    +------------------------+    +------------------------+
```

LCS

Idea of using the longest common subsequence **(LCS)** problem to determine the necessary insert and delete operations.

- Same approach as in UNIX's `diff` utility, developed by J. W. Hunt and M. D. McIllroy
- Compare LCS with original and updated file
- Results in compact updates

```
          original
      +---+---+---+---+
      | A | l | e | x |
  +---+---+---+---+---+
  | A | \ |   |   |   |
u +---+---+---+---+---+
p | l |   | \ |   |   |
d +---+---+---+---+---+
a | i |   | | |   |   |
t +---+---+---+---+---+
e | c |   | | |   |   |
  +---+---+---+---+---+     - deletions
  | e |   |   | \ | - |     | insertions
  +---+---+---+---+---+

  Resulting LCS: Ale
```

LCS example

```
 original:                          original
+-------------+                 +---+---+---+---+            deletions:
|    alex     |---------------->| a | l | e | x |            +-----------------+
+-------------+             +---+---+---+---+---+            |original:   alex |           update:
                     +----->| a | \ |   |   |   |      +---->|LCS:        ale  |-----+     +----------------+
 update:             |    u +---+---+---+---+---+      |     |deletion:      x |     |     |delete:         |
+-------------+      |    p | l |   | \ |   |   |      |     +-----------------+     |     |    index 3: x  |
|    alice    |------+    d +---+---+---+---+---+      |                             +---->|                |
+-------------+           a | i |   | | |   |   |      |     insertions:             |     |insert:         |
                          t +---+---+---+---+---+      |     +-----------------+     |     |    index 1: ic |
                          e | c |   | | |   |   |      |     |update:     alice|     |     +----------------+
                            +---+---+---+---+---+      +---->|LCS:        al  e|-----+
                            | e |   |   | \ | - |      |     |insertions:   ic |
                            +---+---+---+---+---+      |     +-----------------+
                                                       |
                             Resulting LCS: ale   -----+
```

Managing updates

- The original file is considered version 0
- Each update of a file depends on an already existing version
  ⇒ allows creation of different update branches
- Results in a dependency tree

```
            +-----+
            | v0  |
            +-----+
               |
               v
            +-----+
            | v1  |
            +-----+
               |
         +-----+-----+
         v           v
      +-----+     +-----+
      | v2  |     | v3  |
      +-----+     +-----+
         |
   +-----+-----+
   v           v
+-----+     +-----+
| v4  |     | v5  |
+-----+     +-----+
```

Version jumping

It is possible to jump in between **any** two versions of a file using its dependency tree:

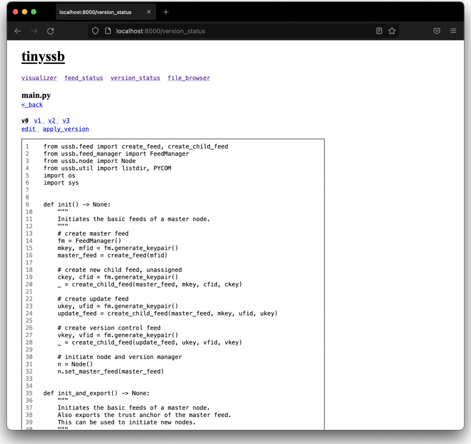
1. **Extract** a path using depth first search
2. **Revert** updates until the latest common predecessor version is reached
3. **Apply** the remaining updates

```
Dependency tree:

+----+    +----+    +----+    +----+
| v0 |--->| v1 |--->| v3 |--->| v4 |
+----+    +----+    +----+    +----+
            |
            v
          +----+
          | v2 |
          +----+

+---------------+------------------+
|jump: v4 to v2|                   |
+---------------+                  |
| 1. path = (v4, v3, v1, v2)       |
|                                  |
|    latest common predecessor: v1 |
|                                  |
| 2. Revert v4, v3                 |
|                                  |
| 3. Apply v2                      |
+----------------------------------+
```

Demo



tinyssb

visualizer feed_status version_status file_browser

main.py

< back

v0 v1 v2 v3

edit apply_version

```
from ussb.feed import create_feed, create_child_feed
from ussb.feed_manager import FeedManager
from ussb.node import Node
from ussb.util import listdir, PYCOM
import os
import sys


def init() -> None:
    """
    Initiates the basic feeds of a master node.
    """
    # create master feed
    fm = FeedManager()
    mkey, mfid = fm.generate_keypair()
    master_feed = create_feed(mfid)

    # create new child feed, unassigned
    ckey, cfid = fm.generate_keypair()
    _ = create_child_feed(master_feed, mkey, cfid, ckey)

    # create update feed
    ukey, ufid = fm.generate_keypair()
    update_feed = create_child_feed(master_feed, mkey, ufid, ukey)

    # create version control feed
    vkey, vfid = fm.generate_keypair()
    _ = create_child_feed(update_feed, ukey, vfid, vkey)

    # initiate node and version manager
    n = Node()
    n.set_master_feed(master_feed)


def init_and_export() -> None:
    """
    Initiates the basic feeds of a master node.
    Also exports the trust anchor of the master feed.
    This can be used to initiate new nodes.
    """
```

Outlook

- Introduce cross-file dependencies
- Integrate with Simon Laube's project
- Field-test in a larger network

```
            +---------+
            |  Update |
            +---------+
                 |
     +-----------+-----------+
     |           |           |
     v           v           v
+---------+ +---------+ +---------+
|  19°C   | |  22°C   | |  15°C   |
|         | |         | |         |
| +-+ ++  | | +-+ ++  | | +-+ ++  |
| | | ++  | | | | ++  | | | | ++  |
| | | ++  | | | | ++  | | | | ++  |
| +-+ ++  | | +-+ ++  | | +-+ ++  |
+---------+ +---------+ +---------+
```

Questions?

m.barth@unibas.ch