Distributed Computer Systems Lab
http://disco.informatik.uni-kl.de

OpenSky's Data, Algorithms, and Tools

Matthias Schäfer

I'd like to use OpenSky's data! How does it look like??

Message Format

- Avro files contain serialized sensor data with the following schema:

```
{
  "name": "ModeSEncodedMessage",
  "type": "record",
  "namespace": "org.opensky.avro.v2",
  "fields": [
      {"name": "sensorType",            "type": "string"},
      {"name": "sensorLatitude",        "type": ["double", "null"]},
      {"name": "sensorLongitude",       "type": ["double", "null"]},
      {"name": "sensorAltitude",        "type": ["double", "null"]},
      {"name": "timeAtServer",          "type": "double"},
      {"name": "timeAtSensor",          "type": ["double", "null"]},
      {"name": "timestamp",             "type": ["double", "null"]},
      {"name": "rawMessage",            "type": "string"},
      {"name": "sensorSerialNumber",    "type": "int"},
      {"name": "RSSIPacket",            "type": ["double", "null"]},
      {"name": "RSSIPreamble",          "type": ["double", "null"]},
      {"name": "SNR",                   "type": ["double", "null"]},
      {"name": "confidence",            "type": ["double", "null"]}
  ]
}
```

A Typical Instance

```
{
  'sensorType':           'OpenSky',
  'sensorSerialNumber':   699700118,
  'sensorLatitude':       50.04854,
  'sensorLongitude':      8.4879,
  'sensorAltitude':       121.0,
  'timeAtServer':         1441929601.158738,
  'timestamp':            86844078.0,
  'rawMessage':           '8d4054a760c380936e7a746ffa90'
}
```

Example Message: Sensor Information

```
{
  'sensorType':           'OpenSky',
  'sensorSerialNumber':   699700118,
  'sensorLatitude':       50.04854,
  'sensorLongitude':      8.4879,
  'sensorAltitude':       121.0,
  'timeAtServer':         1441929601.158738,
  'timestamp':            86844078.0,
  'rawMessage':           '8d4054a760c380936e7a746ffa90'
}
```

- `sensorType` determines availability of metadata and how to interpret it
  - E.g. accuracy of timestamp, availability of RSSI, …
- `sensorSerial` is the unique identifier of each sensor device
- `sensorLatitude` and `sensorLongitude` in decimal degrees
- `sensorAltitude` in feet

Example Message: Timestamps

```
{
  'sensorType':           'OpenSky',
  'sensorSerialNumber':   699700118,
  'sensorLatitude':       50.04854,
  'sensorLongitude':      8.4879,
  'sensorAltitude':       121.0,
  'timeAtServer':         1441929601.158738,
  'timestamp':            86844078.0,
  'rawMessage':           '8d4054a760c380936e7a746ffa90'
}
```

- `timeAtServer` is unix timestamp for the arrival at the OpenSky server
  - Contains Internet jitter
- `timestamp` is a hardware timestamp for the message detection
  - Unit and accuracy depend on hardware
  - Can be used e.g. for multilateration

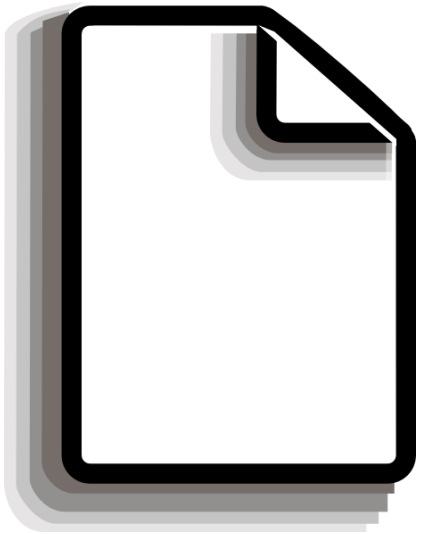
Example Message: Timestamps

```
{
  'sensorType':           'OpenSky',
  'sensorSerialNumber':   699700118,
  'sensorLatitude':       50.04854,
  'sensorLongitude':      8.4879,
  'sensorAltitude':       121.0,
  'timeAtServer':         1441929601.158738,
  'timestamp':            86844078.0,
  'rawMessage':           '8d4054a760c380936e7a746ffa90'
}
```

- `rawMessage` is the Mode message in hex representation
  - CRC is set to 000000 from some sensors
- A complete open-source ADS-B decoder is avlbl on our github account
  - https://github.com/openskynetwork/java-adsb

Mhmm… okay, and what to do when I get a pile of data?

Example file:



raw2015091100.avro
(Size: 8.5GB)

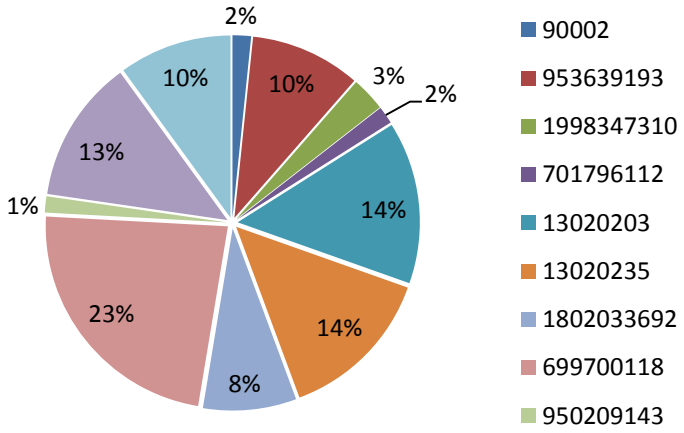
Tools are available in decoder.jar at
github/openskynetwork/osky-sample

Tool #1: AvroInfo

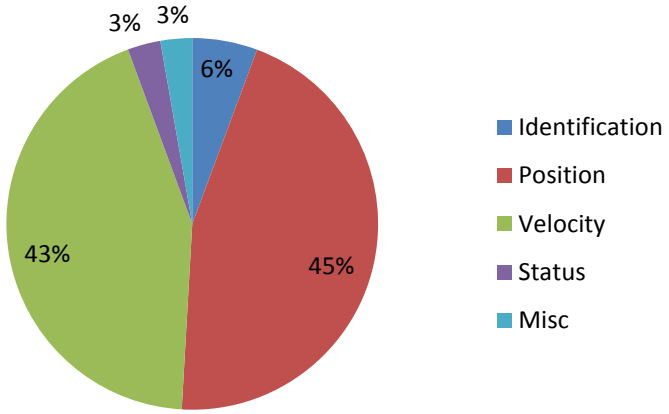
- Provides highlevel information about OpenSky's avro files

```
$ java -cp decoder.jar org.opensky.tools.AvroInfo raw2015091100.avro

...
Counting entries: 94918705
Earliest entry: Fri Sep 11 02:00:00 CEST 2015
Latest entry: Sat Sep 12 01:59:59 CEST 2015
...
```

Messages/Sensor:



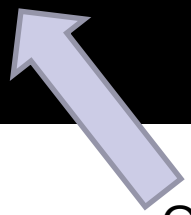
Message Type Distribution:



Tool #2: Avro2KML

- Want to get a visual idea of the data? Let‘s generate a KML file containing the first 1000 flights of our sample.
- Keyhole Markup Language (KML) used to visualize data in Google Earth

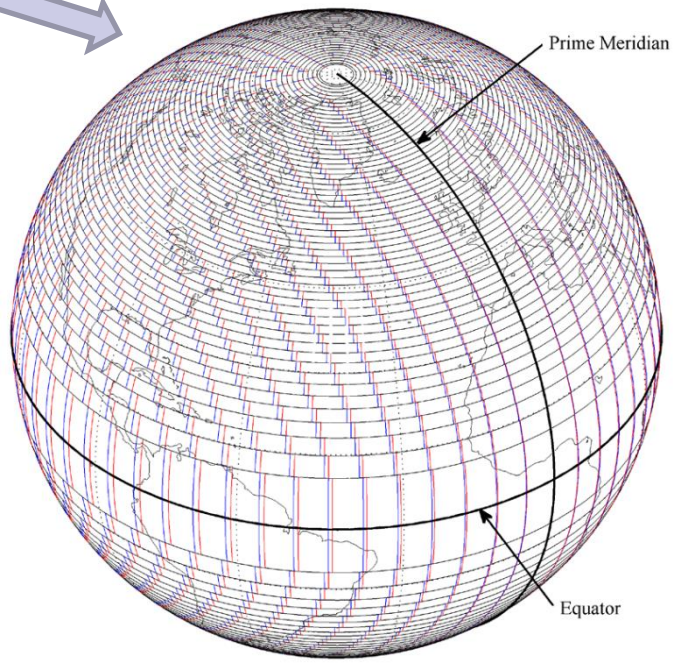
```
$ java -cp decoder.jar org.opensky.tools.Avro2Kml -0 -n 1000
  raw2015091100.avro 1000.kml

...
2015-11-24 12:53:09 WARN  PositionDecoder - Position messages
should be ordered!
2015-11-24 12:53:09 WARN  PositionDecoder - Position messages
should be ordered!
2015-11-24 12:53:09 WARN  PositionDecoder - Position messages
should be ordered!
...
```



Oops… what happened?

Primer on Position Decoding in ADS-B

- ADS-B uses Compact Position Reporting (CPR)
  - Encoding saves 10 bits per airborne position (14 bits per surface position)
- CPR uses special coordinate system for encoding/decoding
  - There are even and odd positions
  - Each of them alone cannot be decoded uniquely



red lines: "even zones"
blue lines: "odd zones"

- Each ADS-B position report contains either even or odd position, not both ☹

- ADS-B position decoding:
  To get unique position we need an even and an odd message (global CPR) or a nearby reference position (local CPR)
  - For global CPR being applicable, both reports (even and odd) must encode positions close to each other


Ah, I see! That's why position messages have to be decoded in order! But how can I sort the avro file?

Tool #3: AvroSort

- Sorts all entries of an avro file by their `timeAtServer`

```
$ java -cp decoder.jar org.opensky.tools.AvroSort
  raw2015091100.avro raw2015091100_sorted.avro

Warning: make sure you have enough main memory. Otherwise, use
AvroSplit first.
```

- But be careful: Sorting the file requires keeping the whole data into RAM
  - The file has 8.5 Gbyte! Loading it completely to RAM consumes even more space.

- Idea: Split file before sorting

Tool #4: AvroSplit

- Splits avro files into *n* smaller pieces with the following properties:
  - The pieces have roughly the same size
  - All messages of messages of an aircraft are in the same file (important for position decoding!)

```
$ java -cp decoder.jar org.opensky.tools.AvroSplit -n 10
  raw2015091100.avro -o raw2015091100_part
...
$ ls -alh raw2015091100_part*
-rw-r--r-- 1 matze users 788M Nov 25 16:26 raw2015091100_part1.avro
-rw-r--r-- 1 matze users 899M Nov 25 16:26 raw2015091100_part10.avro
-rw-r--r-- 1 matze users 842M Nov 25 16:26 raw2015091100_part2.avro
-rw-r--r-- 1 matze users 859M Nov 25 16:26 raw2015091100_part3.avro
-rw-r--r-- 1 matze users 988M Nov 25 16:26 raw2015091100_part4.avro
-rw-r--r-- 1 matze users 791M Nov 25 16:26 raw2015091100_part5.avro
-rw-r--r-- 1 matze users 898M Nov 25 16:26 raw2015091100_part6.avro
-rw-r--r-- 1 matze users 854M Nov 25 16:26 raw2015091100_part7.avro
-rw-r--r-- 1 matze users 834M Nov 25 16:26 raw2015091100_part8.avro
-rw-r--r-- 1 matze users 872M Nov 25 16:26 raw2015091100_part9.avro
```

Sorting continued…

- Now we can sort the smaller files

```
$ for f in raw2015091100_part*; do java -cp decoder.jar
  org.opensky.tools.AvroSort ${f} sorted_${f}; done
...
$ ls sorted_raw2015091100_part*
sorted_raw2015091100_part1.avro  sorted_raw2015091100_part5.avro
sorted_raw2015091100_part10.avro sorted_raw2015091100_part6.avro
sorted_raw2015091100_part2.avro  sorted_raw2015091100_part7.avro
sorted_raw2015091100_part3.avro  sorted_raw2015091100_part8.avro
sorted_raw2015091100_part4.avro  sorted_raw2015091100_part9.avro
```


But I would like to decode the whole dataset! Not only a 10th at a time…

Joining + Generating KML Files

- AvroSplit also accepts multiple input files
  - It can be used to concatenate files by setting n=1

```
$ java -cp decoder.jar org.opensky.tools.AvroSplit -n 1
  sorted_* -o raw2015091100_sorted
```

  - Note: The resulting file (raw2015091100_sorted1.avro) is not completely ordered. Only the messages of a certain aircraft are ordered by `timeAtServer`!
- And now try generating the KML again:

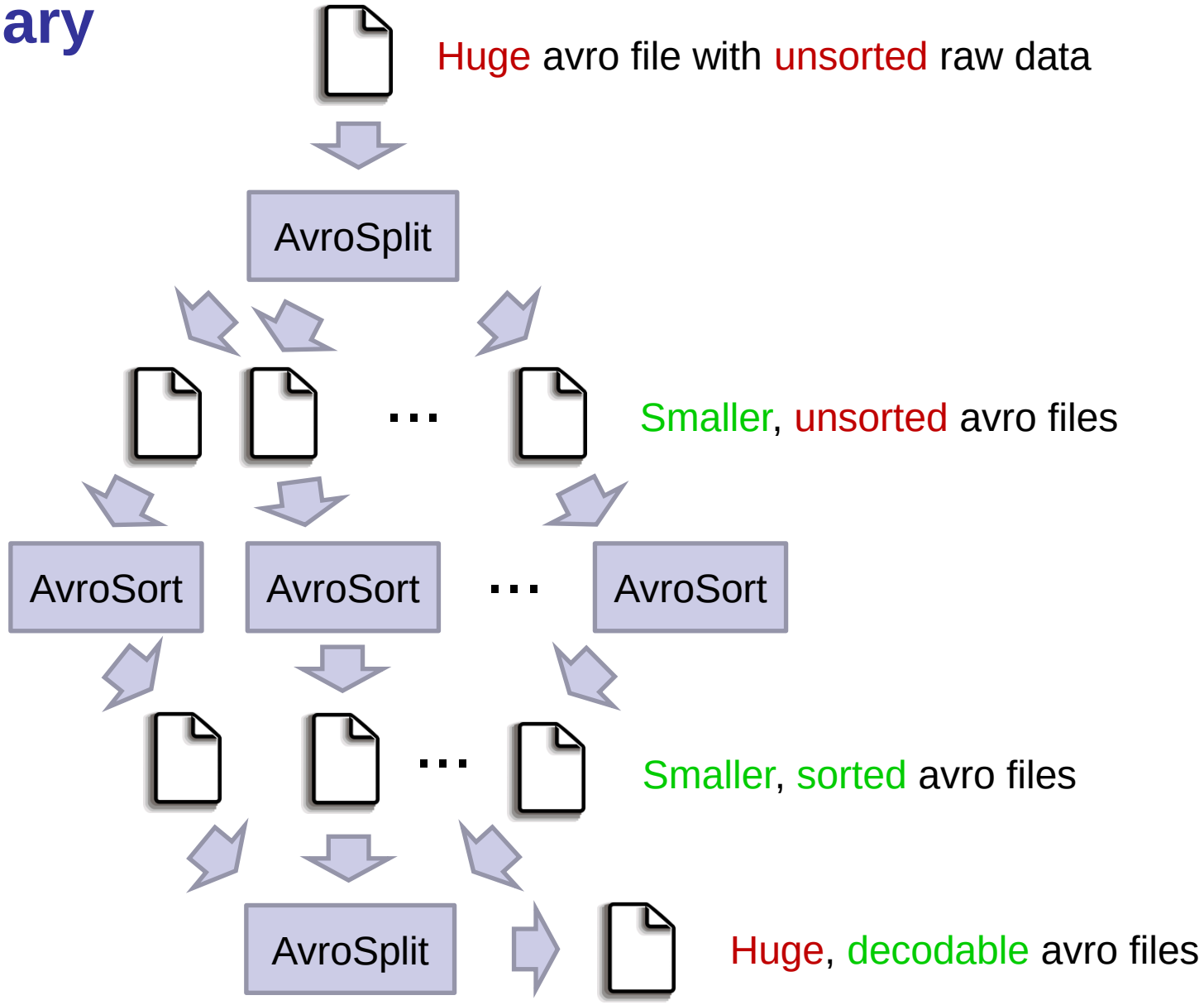
```
$ java -cp decoder.jar org.opensky.tools.Avro2Kml -0 -n 1000
  raw2015091100_sorted1.avro 1000.kml
```

- Have a look at 1000.kml with Google Earth ☺


Copenhagen
Lithuania
Vilnius
Belarus
Ireland
Dublin
United Kingdom
London
Netherlands
Berlin
Poland
Warsaw
Muir of Ord
Belgium
Germany
English Channel
Czech Republic
Kyiv
Ukraine
Celtic Sea
Paris
Slovakia
Vienna
Bratislava
Austria
Budapest
Hungary
Moldova
Chisinau
France
Switzerland
Alps
Slovenia
Croatia
Romania
Bay of Biscay
Belgrade
Bucharest
Sarajevo
Serbia
Bulgaria
Italy
Montenegro
Prishtina
Sofia
Rodgorica
Skopje
Rome
Adriatic Sea
Tirana
Albania
Spain
Madrid
Balearic Sea
Tyrrhenian Sea
Greece
Athens
US Dept of State Geographer
© 2015 Google
Image Landsat
© 2009 GeoBasis-DE/BKG
Ionian Sea
Google
Algiers
Tunis
Imagery Date: 4/10/2013 lat 46.291113° lon 8.990651° elev 405 m eye alt 2696


Darmstadt
Frankfurt
Wiesbaden
Rüsselsheim
Darmstadt
Bad Kreuznach
© 2015 Google
Image Landsat
© 2009 GeoBasis-DE/BKG
Imagery Date: 4/10/2013 lat 49.966147° lon 8.321413° elev 139 m eye

Summary


Huge avro file with unsorted raw data
AvroSplit
Smaller, unsorted avro files
AvroSort
AvroSort
AvroSort
Smaller, sorted avro files
AvroSplit
Huge, decodable avro files

I want to have some statistics and analyse flight paths!!

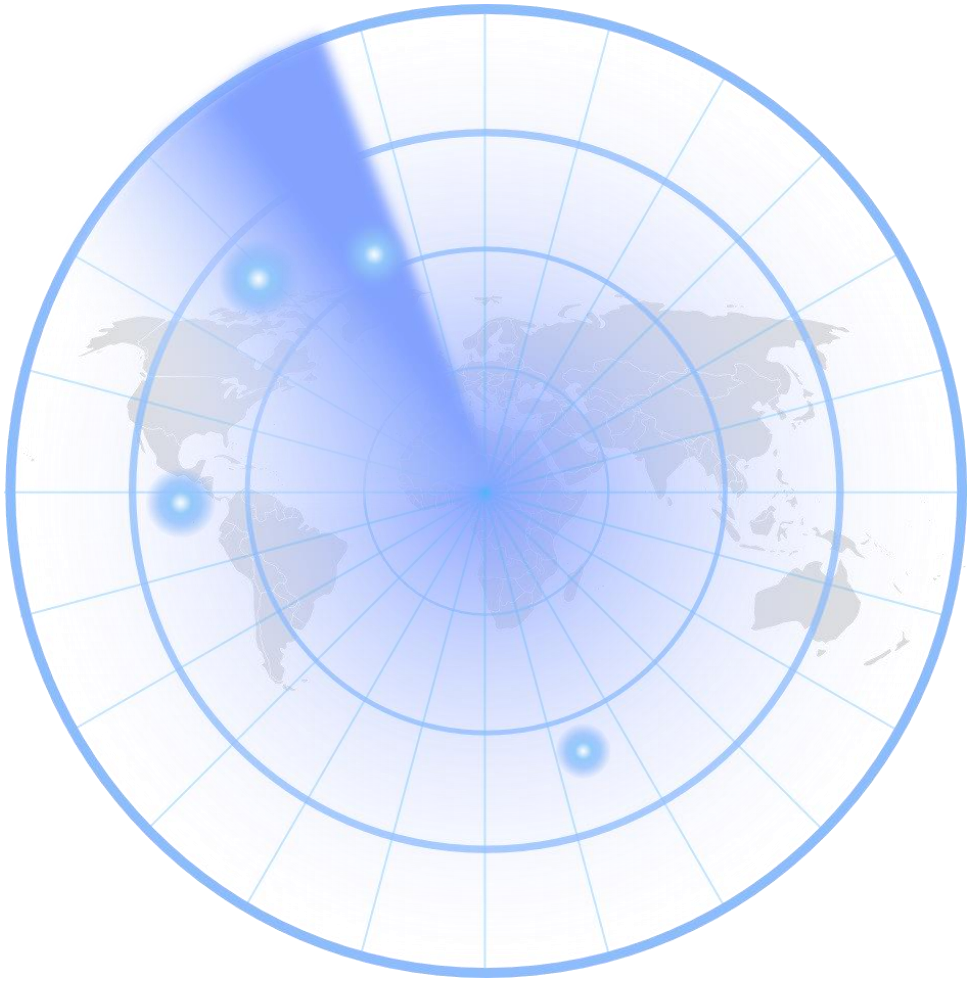
Tool #5: Avro2SQLite

- Generates SQLite database with decoded flights including their positions and velocities
  - Can be queried and loaded for your analyses
  - But be careful: it does not scale ☹

```
$ java -cp decoder.jar org.opensky.tools.Avro2SQLite
  raw2015091100_sorted1.avro raw2015091100.sqlite
...
$ sqlite3 raw2015091100.sqlite
sqlite> SELECT * FROM flights ORDER BY RANDOM() LIMIT 1;
54951|1441986565.60435|1441989692.76459|a2b728|UPS213
sqlite> SELECT * FROM positions WHERE flight=54951;
54951|1441986828.41693|-3.001072474888836|53.4179077148437|10668.0|370.4
54951|1441986829.43119|-2.99722726004461|53.417541503906 2|10668.0|370.4
...
sqlite> SELECT * FROM velocities WHERE flight=54951;
54951|1441986606.30583|253.06671140109|97.240590129645|0.0
54951|1441986611.21757|253.387563099794|96.87973246540 71|0.65024
...
```

Feel free to extend our tools or create new ones! All source codes and examples can be found on github:

https://github.com/openskynetwork/java-adsb



Thanks ☺