Sputnik
RFID Ueberwachung zum Selberbasteln

Hannes Mehnert
hannes@mehnert.org

GPN7 – 1.6.07 - Karlsruhe

Projektuebersicht

- Hardware
- Middleware
- Visualisierung

Hardware

- OpenPCD
- 2.4GHz
- > 20 Access Points
- binary UDP
- alle n Sekunden pollen der umliegenden Tags, verschiedene Feldstaerken -> Abstand zwischen Tag und AP
- http://www.openbeacon.org/

Visualisierung

- 3D renderer
- 3D model
- Plugin fuer Rendering Engine, um Avatare darzustellen
- Touchscreen-Interface zum Auswaehlen der virtuellen Kameras und Avatare
- Wheel zum Verstellen der Perspektive der Kamera

undefined
undefined
254201
k: undefined
me: undefined
es visited

visitor
male
ID: 101
C
B
A

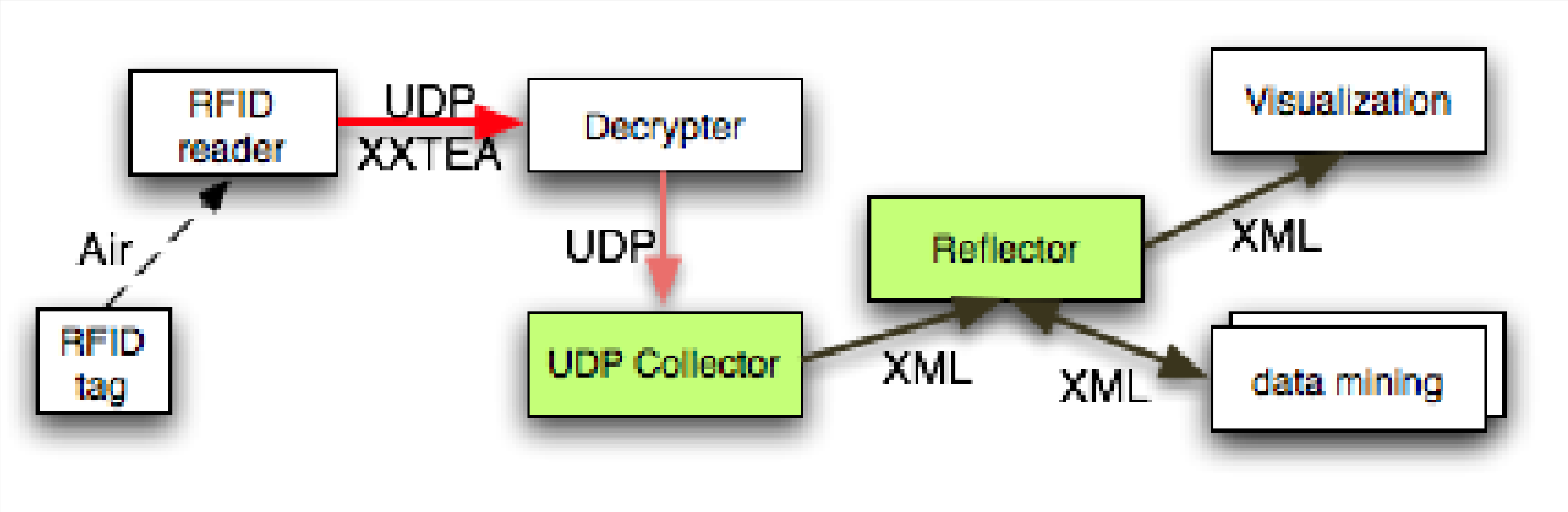
Statistiken

- lief ab 28.12. - am 27. noch debuggen
-
- 1000 Tags verkauft
- xtausend Observations der Access Points
- xhundert Assoziierungen von Tags zu Usern

Kommunikation

- binary UDP von Access Points
  - source IP, Tag ID, Signal Strength
- XML-push zur Visualisierung
  - relative Koordinaten (relativ zu Mitte der Cafete)
- XML-pull von der Visualisierung
  - Details ueber ein Tag
- HTTP

Software Data Flow



Middleware

- UDP Collector – Packetbeschreibungssprache [GPN6] – konvertiert nach XML
- Reflector – HTTP requests, XML priority parsen, multiplexer, native Interface
- Data mining
  - wer haengt mit wem zusammen rum?
  - welche Vortraege wurden besucht?
- User database: HTTP & XML interface

Conclusion

- Datensparsamkeit
  - “wer, wenn nicht wir?”
- Projekte mit verschiedenen Teams mit verschiedenen Hintergruenden durchaus interessant

Design Details

- Modular
  - Reflector zentraler Multiplexer
  - UDP-collector initiale raw Datenquelle
  - HTTP Interface abgegrenzt davon, einzige Referenz durch URL in Observation

UDP binary protocol

```
define protocol sputnik-udp-frame (container-frame)
  summary "from %s %s tx %d ID %= S %=",
    flags-summary, transmit-strength, unique-tag-id,
    sequence-number;
  over <udp-frame> 2342;
  field originator :: <ipv4-address>;
  field data-size :: <unsigned-byte>,
    fixup: byte-offset(frame-size(frame));
  field protocol-version :: <unsigned-byte> = 23;
  field reserved :: <6bit-unsigned-integer> = 0;
  field flag-sensor :: <1bit-unsigned-integer> = 0;
  field flag-ack :: <1bit-unsigned-integer> = 0;
  field transmit-strength :: <unsigned-byte>;
  field sequence-number :: <big-endian-unsigned-integer-4byte>;
  field unique-tag-id :: <big-endian-unsigned-integer-4byte>;
end;
```

XML Schema definition

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name='observation'
    <xs:attribute name='observer' type='xs:anyURI' use='required'/>
    <xs:attribute name='observed-object' type='xs:anyURI'
use='required'/>
    <xs:attribute name='time' type='xs:double' use='required'/>
    <xs:attribute name='priority' type='xs:integer' use='required'/>
    <xs:attribute name='min-distance' type='xs:float' default='0'/>
    <xs:attribute name='max-distance' type='xs:float' default='255'/>
    <xs:attribute name='direction' type='xs:string' default='[0,0,0]'/>
    <xs:attribute name='message' type='xs:string'/>
    <xs:attribute name='tags' type='xs:string'/>
    <xs:attribute name='position' type='xs:string'/>
  </xs:element>
</xs:schema>
```

XML observation

```
<observation
  observer=”http://sputnik/observer/10.2.3.42”
  observed-object=”http://sputnik/tag/2342”
  time=”23422324.232323e0”
  priority=”0”  />

<observation
  observer=”http://sputnik/observer/history-bot”
  observed-object=”http://sputnik/tag/2342”
  time=”23422324.232324e0”
  position=”[0.0e0,1.0e0,2.0e0]”
  priority=”23”  />
```

Reflector

- Master Thread
  - Listener TCP 8000
  - starts client thread, parses request line
    - GET /23 #everything with priority 23 or higher
  - reading observations
  - multiplexing observations to clients with specific priority, requires parsing
  - native interface for Dylan clients [no need to parse every observation multiple times]

location tracking

- Parses a file containing locations of access points at startup
- waits 5 seconds for observations
- emits an observation for each tag seen, singularizes location

<history-bot>

```
define class <history-bot> (<virtual-source>)
  slot messages :: <table> = make(<table>);
  slot message-lock :: <lock> = make(<lock>);
end;

//gather data
define method receive-data
 (h :: <history-bot>, o :: <observation>)
  with-lock(h.message-lock)
    let msgs = element(h.messages,
                                   o.observation-observed-object,
                              default: make(<stretchy-vector>));
    h.messages[o.observation-observed-object]
        := add!(msgs, o);
  end;
end;
```

process-incoming-data

```
define function process-incoming-data
 (source :: <history-bot>)
  let data = #f;
  with-lock (source.message-lock)
    data := source.messages;
    source.messages := make(<table>)
  end;
  for (key in key-sequence(data))
    format-out("Processing data for %=\n", key);
    let observation
       = process-data(source, key, data[key]);
    source.push-closure(observation);
  end;
end;
```

process-data

```
define method process-data
 (source :: <history-bot>, tid, data :: <collection>)
 => (res :: <observation>)
  let normalization-factor :: <float> = 0.0d0;
  let positions
    = map(method(x)
            let max-dist = x.observation-max-distance;
            let factor :: <float> = 1.0d0;
            if (max-dist > 0)
              factor := (1.0d0 / max-dist) * $max-distance;
            end;
            normalization-factor := normalization-factor + factor;
            let pos = get-position(x.observation-observer);
            if (pos)
              factor * pos
            else
              format-out("unknown accespoint %=\n",  x.observation-observer);
              as(<position>, "[0,0,0]");
            end;
          end, data);
  let npos = make(<position>,
     x: reduce1(\+, map(x, positions)),  y: reduce1(\+, map(y, positions)),
     z: reduce1(\+, map(z, positions))) / normalization-factor;
  let t = element(storage(<sputnik-tag>), tid, default: #f);
  t | t := make(<sputnik-tag>, id: tid);
  make(<history-observation>, object: tid, position: npos);
end;
```

Pentabarf matching

- Description of extensions of rooms
  - xml file, room is described by cuboids
- Description of lectures
  - xml output of pentabarf
- Look every 5 minutes whether tag X is in the same room, if so, add lecture of that room to tag X
- no extensions of rooms were available
  - no pentabarf matching done yet

HTTP Interface

- simple, web-0.8
- user, tag, room, observer
- add/edit/remove
- xml and html interface
- HTTP authentication
- Data Store – persistent object store
  - (“Why should I use databases?”)

Future Development

- USB Access Points
- P2P Firmware to get tracked
- Improve data mining
- integrate social network
- bidirectional data between tag and AP