Visualizing APRS Messaging with Heml

Dr. Bruce Robertson
Mount Allison University, Canada
Dept. of Classics
brobertson@mta.ca

Kevin Green
Mount Allison University, Canada
Dept. of Computer Science
Dept. of Commerce
ksgrn@mta.ca

Abstract

This paper describes a web-accessible computer program which generates timelines and animated maps from APRS messaging data extracted from the APRSworld database.

Introduction

There can be no doubt that the Automatic Packet Reporting System (APRS), founded by Bob Bruninga (APRS Working Group, 2000), has enjoyed such wide popularity in part due to its integration with higher-level computer technologies, above all with websites and programs that provide maps and other visualization of APRS data in real time.[1] Programs such as Xastir[2], running on computers connected to the radio networks, provide compelling positional mapping, and mapping sites such as www.findu.com[3] and aprsworld.net[4] lower the barrier of access to this data, making APRS information understandable, and compelling, to the larger world. From search and rescue (Lehman 2003a, 2003b) to myriad other pursuits (Parry 2003, Horzepa 2004) APRS location visualization has arguably become the 'killer app.' for packet radio.

APRS message data evidently have not undergone a similar visualization. This is unfortunate because it can be useful to have a illustrating the timing of messages relative to each other and the location of the person sending the message. The work described in this paper uses an existing software package to generate animated maps and graphical timelines from APRS messages. These images are produced with the data stored at the APRSWorld database and using the historical visualization tools of the Historical Event Markup and Linking (Heml) Project, directed by Robertson at the Dept. of Classics of Mount Allison University in Canada.

The Heml Project

The Heml Project[5] was conceived out of experiences teaching and researching ancient history (Robertson 2002, 2004). It aims first to provide a XML markup language suited for recording and indexing events in human history, particularly as provided in web-based documents. The current markup language is defined in the W3C schema and is available at http://heml.mta.ca/Schemas/2003-09-17/. Secondly, the Heml Project explores the uses of such data: how they might be transformed into historical maps, timelines and animations. The webapp that provides these services is freely available and contains only open-source computer code.

Although the Heml Project might seem distant from the needs of APRS users, Heml has always intended to be useful to as wide a range of applications as possible. In particular, improvements in the markup language and visualization tools undertaken in the last two years provide a far finer chronological resolution, allowing it to encode and visualize events which take place over mere seconds. As an example of this capacity, we provide a sample document outlining the events in the last seconds of the Columbia shuttle accident. Example 1 shows the XML data representing two of these events.

Example 1. Sample Heml XML Data Representing Events

```
<Event uri="http://www.heml.org/docs/samples/heml/2002-05-
29/columbia.xmlFRAGe1">
  <EventLabelSet>
    <Label xml:lang="en">Shuttle Columbia reenters Earth's
atmosphere</Label>
  </EventLabelSet>
  <Chronology>
    <DateTime>2003-02-01T08:44:09</DateTime>
  </Chronology>
  <References>
    <Evidences>
      <Evidence>
        <NetworkedSourceSet>
          <SimpleLink
            xlink:href="http://spaceflightnow.com/shuttle/...">
            SpaceflightNow Columbia Timeline
          </SimpleLink>
        </NetworkedSourceSet>
      </Evidence>
    </Evidences>
  <References>
</Event>
<Eventuri="http://www.heml.org/docs/samples/heml/2002-05-
```

```
29/columbia.xmlFRAGe2">
  <EventLabelSet>
    <Label xml:lang="en">Last communication from shuttle
Columbia</Label>
  </EventLabelSet>
  <Chronology>
    <DateTime>2003-02-01T08:59:32</DateTime>
  </Chronology>
  <References>
    <Evidences>
      <Evidence>
        <NetworkedSourceSet>
 <SimpleLink xlink:href="http://anon.nasa-global.speedera.net/anon.nasa-
global/...">
            Columbia Accident Investigation Board Final Report Vol. 1
Ch. 2 p. 39.
          </SimpleLink>
        </NetworkedSourceSet>
      </Evidence>
    </Evidences>
  </References>
</Event>
```

This example illustrates the bare minimum required for Heml encoding: events comprise a label, a span of time and a reference to evidence. Finally, each Event element must be assigned a unique identifier in the uri attribute. Heml XML can encode other information pertinent to historical events, such as persons, their roles in events (such as 'victor', or 'runaway slave') and keywords. Additionally, in order to produce historical maps, Heml markup allows events to be associated with locations. Since more than one event might occur at the same location, the markup language requires that locations be defined in an earlier part of the document and referred to within Event elements. Example 2 illustrates how a location is defined.

Example 2. Sample Heml XML Data Representing Location

```
<Location
  uri="http://www.heml.org/docs/samples/heml/2002-05-
29/alexander.xmlFRAGthebes">
  <LocationLabelSet>
    <Label xml:lang="en">Thebes</Label>
  </LocationLabelSet>
  <geo:Point>
    <geo:lat>38.3333333333333</geo:lat>
```

```
    <geo:long>23.3333333333333</geo:long>
  </geo:Point>
</Location>
```

While exploring the uses of APRS, it became clear to Robertson that one might understand APRS messages as very recent historical events. The message and the callsigns of the communicating stations comprise the event's label. If it were possible to record a reasonably accurate representation of the time at which the message was received, there would be sufficient data to transform into Heml XML. From there, the existing Heml program would produce timelines such as the one represented in Figure 1. If location information could be determined, Heml's map-generating code could be employed as well.

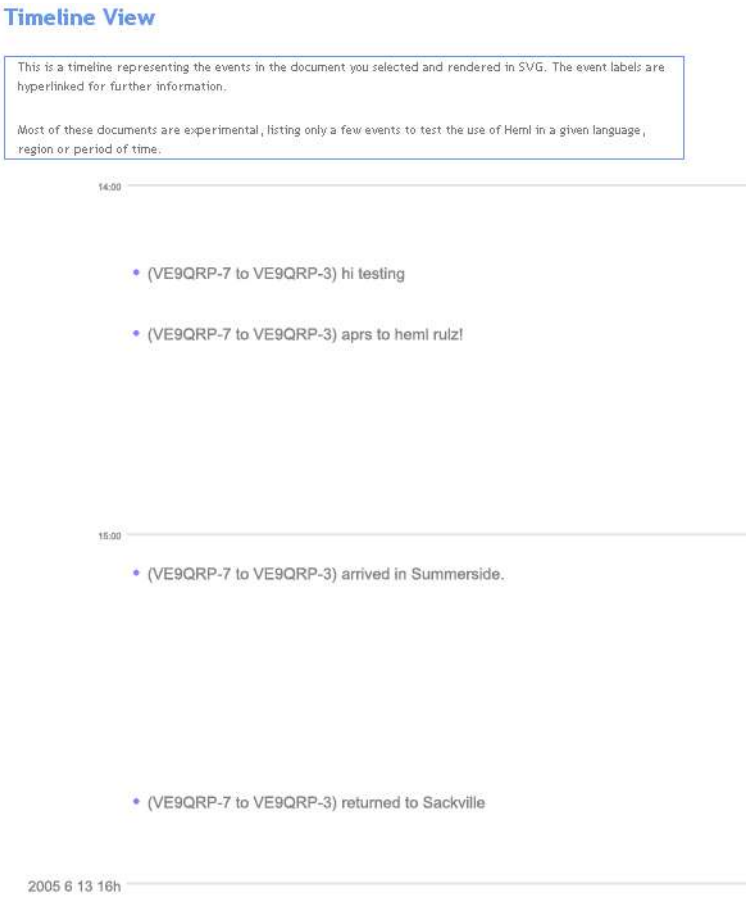

Figure 1. Timeline Example (Received Messages of VE9QRP-5)

Transforming APRS Packets Into Heml XML

We found that it is possible to generate XML from APRS data in many different ways. The aprsearch.net[6] website provides an XML interface to APRS data, but it does not allow the user to draw

on multiple messages, and we wanted to be able to visualize more than one message at a time. We considered taking the raw data off a live telnet stream and parsing it on the fly. However, this would require us to dedicate excessive computational resources. Our final, and preferred, course of action was to access an on-line database of processed APRS information. Though this approach adds a dependency -- the database must be up and connected --, it provides access to every igated APRS message and to related location packets, a feature we were able to exploit to considerable advantage.

APRSworld[7] provides easy-to-use access to APRS packets transmitted across the Internet. It stores these in a MySQL[8] database server to which authorized clients can directly connect across the Internet. Its director, James Jefferson Jarvis, kindly provided us with a username and password necessary for client access.

With this approach, APRS message data is transformed into Heml XML in two steps: first it is stored in the APRSworld database; then that database's fields are queried and transformed into XML. The first step is performed by the APRSworld site; the second takes place in computer code written by Green for this project. This information flow is outlined in Table 1 Example 3 and Example 4 shows the origin of the data in the APRS packets.

Table 1. Connection of APRS Data to Heml

Information	APRS Packet	APRSworld Database Fields	Heml XML Elements
Message Text	Example 3 - **(2)**	`message.text`	`heml:Label`
Recipient	Example 3 - **(1)**	`message.addressee`	`heml:Label`
Sender	Message Header	`message.source`	`heml:Label`
Date and Time[a]	n/a	`messages.packet_date`[a]	`heml:DateTime`
Unique Identifier	n/a	`messages.packet_id`	`heml:Event/@uri`
Latitude	Example 4 - **(1)**	`position.latitude`	`geo:lat`
Longitude	Example 4 - **(2)**	`position.longitude`	`geo:long`

Notes:
a. The date and time are given by APRSworld when it received the APRS message.

Example 3. APRS Message Packet

```
:(1)WU2ZVVVVV:(2)Testing{(3)003
```

(1) Callsign of the Recipient

(2) Message Text

(3) Unique Message Identifier

Example 4. APRS Location Packet

```
!(1)4903.50N/(2)07201.75W-(3)Test 001234
```

(1) Latitude

(2) Longitude

(3) Comment

A bare-bones Heml Event element requires a label, time and unique identifier. We generate labels using the message text and the callsigns of the two stations involved, both stored by APRSworld in appropriate database fields. Although APRS messaging currently does not include date-stamping, APRSworld conveniently assigns a date/time to the message packet when it is received. Because Igating is usually a rapid process, we believe this is a reasonable approximation of the time the message was sent. Finally, a unique identifier for the event is generated by appending the packet_id (assigned by the database) after the uri http://www.heml.org/haprs/. The resulting Heml XML can be seen in Example 5.

APRS messages do not themselves comprise location data, but we have used APRSworld to estimate a likely location, if desired. Since APRSworld stores both location and messaging packets, and all packets are assigned packet_id values which increment through time, we use the pertinent information from the location packet with the largest packet_id value smaller than that of the target message packet, assuming that this is the last location packet sent before that message. This is a relatively simple query and does not make excessive demands on the APRSworld database. The XML in Example 5 has a LocationRef element generated in this way. (It refers to a Location similar to the one in Example 2 but which is not visible.)

Example 5. Heml XML Generated from APRSworld APRS Data

```
<Event uri="http://www.html.org/harps/735305808">
  <LocationRef
   uriRef="http://www.html.org/harps/loc/lat/38.991667/long/-
76.872167"/>
    <EventLabelSet>
      <Label xml:lang="en">
      (K3CON to K3NOR) Hello there, Dave!!
      </Label>
```

```
    </EventLabelSet>
  <Chronology>
    <DateTime>2005-08-06T19:03:56</DateTime>
  </Chronology>
  <References>
    <Evidences>
      <Evidence>
        <NetworkedSourceSet>
          <SimpleLink xml:lang="en"
           xlink:href="http://db.aprsworld.net/datamart/switch.php?
           call=K3CON&amp;table=message&amp;showall=1">
             APRSWORLD Message Query for K3CON
          </SimpleLink>
        </NetworkedSourceSet>
      </Evidence>
    </Evidences>
  </References>
</Event>
```

Technical Details of Heml XML Generation

The technical details of generating the XML are as follows. The Heml web application is built upon Cocoon[9], a web application framework written in Java and hosted by the Apache group. Cocoon responds to web queries by causing XML data to flow through a set of defined steps. First XML data is generated, either from a stored file or some other means; then it is transformed; finally, it is serialized and transmitted to the client. These processes are aptly called 'pipelines' by Cocoon's designers because different XML generation pipes can be fed into common transformation and serialization processes. For this reason, our only task in this project was to generate the Heml XML. Once that is done, it is handed over to the existing webapp code, which draws the timelines and maps regardless of the origin of the XML.

The XML generation begins with a simple web form by which the user inputs the parameters for his or her search. Figure 2 is a screen-shot of a browser using this form, which appears in the top half of the figure. It provides the following options:

- The APRS callsign to search
- A choice of either sent or received messages
- The maximum number of results to be returned
- A choice to include or exclude locations
- A choice to include or exclude duplicates
- A choice to include or exclude acknowledgement messages

- A choice to include or exclude rejection messages

Since all of these values are passed through simple HTML GET parameters, a client need not necessarily use this form to query the visualization engine.

These parameters are used to create a query document that is passed to Cocoon's SQL transformer[10]. This process takes in an XML document with formatted SQL statements. It then extracts the SQL query from the XML file and replaces it with the results from the query. After the SQL transformation has taken place the data is in the form of row by row information formatted in XML; though not yet Heml XML, the data is now in the proper starting format for it to be transformed into conforming XML. The SQL query itself also takes care of an important task. It weeds out duplicate messages, determining uniqueness based on the messages' senders, addressees, texts, and id (this is an incrementing field to designate changing messages). The query also filters for acknowledgement and rejection messages.

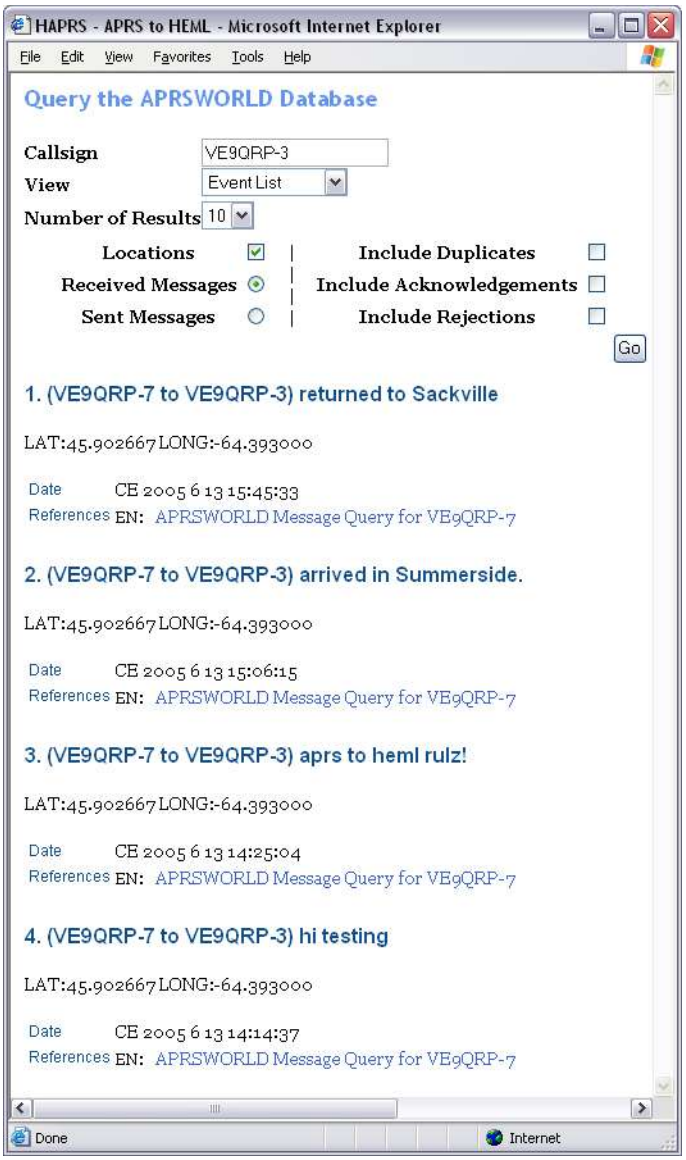

Figure 2. Event List Example (Received Messages of VE9QRP-3)

By processing this raw message data through an XSLT stylesheet, a valid Heml document like the one in Example 5 is generated. At this stage, though, the document does not include any location information. If this is desired another XSLT stylesheet is applied dynamically to generate the SQL queries needed to associate locations with the messages in the manner described above. To recreate a proper Heml document once again, the XML file goes through a few more transformation which format the locations, label them, and remove any duplicates. If location elements are included, other message visualizations are possible, including the animated map shown in Figure 3 and described below.

Visualizations

The Heml webapp provides several ways of viewing data that conforms to its XML Schema; not all of

these are appropriate for transformed APRS messages. Figure 2 shows one of the simplest views, an HTML table of events. Useful in other contexts, this is no great improvement over the text versions of messages provided at APRSworld. The more compelling views are those that transform the data into graphical representations. These maps and timelines are drawn using the Scalable Vector Graphics[11] standard, a means of drawing for the web that is akin to Flash but very easy to work with via XML data. (Browser plugins for most major operating systems and platforms are available from Adobe[12].)

Figure 1 shows part of a vertical timeline rendered in SVG. The left side of the figure shows the regular intervals of time generated by the webapp; the text of messages appears at the appropriate point in time on the graph. It should be emphasized that these timelines are rendered directly from the Heml XML data in real time: no additional scaling or time-slicing information is required.

When locations are assigned to the messages, they can be rendered on a map. One of the most intriguing Heml visualizations is a SVG map animation devised by Susan Barnett in 2002. Figure 3 is a picture taken of a playing animation representing recent messages sent to WB4APR. In it, a triangular slider moves across a bar from left to right, representing the passage of time from the first of the messages to the last. As the animation progresses, event labels -- here, the transmission of the message -- appear and disappear appropriately. The animation may be paused, stopped and restarted using the familiar controls at the top. Various base maps are available besides the one illustrated in the figure, and Heml code can automatically select the one appropriate for the data being viewed. (Currently the Adobe browser plugin is the only SVG viewer that can accurately render animations such as these.)

Figure 3. Animated Map Example (Received Messages of WB4APR)

Though still a work in progress, the query page and pertinent documents for this project are available at http://heml.mta.ca/haprs/. Since the underlying Heml and Cocoon code are provided in an open source license, all are free to use and modify it as they wish, but without any guarantee. We offer this work in the hope that, like scholars of ancient history, APRS users might find it useful to be able to visualize time in different ways.

The Future: Ideas for Improvement

A project such as this tests the limits of all its component technologies. As a result of our experiences, we propose the following ideas for improvements to APRS messaging transformation, the APRS protocol and Heml.

Determining Locations

Currently, our code assigns a location to a message through the last location packet sent out by the user before the message was sent. Although this can provide decent results, it is possible that more an accurate location was broadcast shortly *after* the message. Therefore it would be more accurate to use a database query which compares all location packets, before and after the message, and determines the one closest in time to the message; but we found this process to be far too taxing on the database server and unpredictable in the length of time it took to return data.

APRS Data

Our project would be much improved in reliability if APRS message packets included location and time information. Perhaps future versions of the protocol, or an alternative such as opentrac,[13] could adopt a message format that did so.

Heml

APRS messaging data is very challenging for the Heml timeline drawing routine and has exposed several bugs in this code that older historical material did not reveal. In particular, as it attempts to provide a legible timeline, this routine occasionally produces an extremely long timeline, occasionally crashing the server in the process. To improve upon this we will be working on making the timeline scrollable and putting a hard upper limit on timeline length.

Currently, the routine that selects a base map for a given set of locations is inaccessible to our code. This must be rectified so that users are presented with base maps of greater detail, which are especially likely to be useful in visualizing APRS messaging.

Bibliography

APRS Working Group. (2000) *APRS Protocol Specification Version 1.0.1*[14] I. Wade, ed.

Horzepa, Stan. (2004) *Aprs Moving Hams on Radio and the Internet* American Radio Relay League.

Lehman, Jeff. (2003a) “APRS and Search and Rescue, Part 1 of 2” *QST* Sept. 2003. p. 81.

Lehman, Jeff. (2003b) “APRS and Search and Rescue, Part 2” *QST* Oct. 2003. p. 75.

Parry, Richard. (2003) “Amateur Radio, Paragliding and an APRS Weather Station” *QST* Aug. 2003. p.

28.

Robertson, Bruce. (2002) “An Overview of the Historical Event Markup and Linking Project” F. Niccolucci, ed. *Dalla Fonte all Rete: Il linguaggio XML e la codifica dei documenti storici, archeologici e archivistici* University of Pisa. 2002.

Robertson, Bruce. (2004) “Improving Ancient History Online with Heml[15] ” C. Blackwell, R. Scaife, edd., *Classics*@ 2. C. Dué & M. Ebbott, executive editors, Center for Hellenic Studies of Harvard University. Edition of April 3, 2004.

Notes

1. This work has been funded by a summer student research grant from the New Brunswick Innovation Foundation.

2. http://www.xastir.org
3. http://www.findu.com
4. http://www.aprsworld.net
5. http://heml.mta.ca
6. http://aprsearch.net/xml/
7. http://aprsworld.net
8. http://www.mysql.com
9. http://cocoon.apache.org/
10. http://cocoon.apache.org/2.1/userdocs/transformers/sql-transformer.html
11. http://www.w3.org/Graphics/SVG/
12. http://www.adobe.com/svg/
13. http://www.opentrac.org
14. ftp://ftp.tapr.org/aprssig/aprsspec/spec/aprs101/APRS101.pdf
15. http://www.chs.harvard.edu/classicsat/issue_2/b-robertson_2004_all.html