

An iPhone App for Transient Events

Bruce Truax

Diffraction Limited Design LLC

Abstract When the LSST comes online later this decade it should prove very efficient at discovering transient events in the heavens. Some of these transient events will be short lived or will move rapidly across the sky and will benefit from follow up observations by professional and amateur astronomers. Useful follow up can only occur if the interested observers can be made aware of the events in a timely manner. Under the auspices of the LSST program an iPhone application has been developed as a testbed for transient notification distribution. Using current surveys such as the Catalina Real Time Survey an iPhone app has been created which allows quick access to recent transient events and has the ability to send immediate notifications to users based on their interests.

The Goals of the Transient iPhone App

The Large Synoptic Survey Telescope (LSST) will provide a fire hose of transient events when it comes online in 2015: according to some estimates it will prove so effective at finding transient events that it may report thousands every night. Some of these transient events will be short lived or will move rapidly across the sky and will benefit from follow up observations by professional and amateur astronomers. There will also be many laypeople who will be interested in the transient events discovered each night. Part of the charter of the LSST is to disseminate the discoveries to interested parties in the astronomy community (both professional and amateur) as well as to the general public. The question of how best to distribute this information needs to be investigated well before opening the nozzle on the fire hose of discoveries.

Smart personal communicators such as the iPhone, Blackberry, Palm, and Windows Mobile devices offer an excellent method for dissemination of transient event information. Smartphones have the ability to provide a rich interface to information in an easy to use, always on and always handy device. These devices also have the ability to provide instant notifications to their owners. Creating client applications and a notification system for such devices allows interested individuals to be alerted to transient events almost immediately and the notifications can be linked to a summary display of information with enough detail to let an astronomer know whether they should leave dinner with their family and rush to the observatory.

One question which is often asked is “Why create an application dedicated to a device? Wouldn’t a web based solution which be more universal?”

Many content providers have sites specifically designed for mobile users. These sites reformat content in a way that better fits the smaller screens of mobile devices. A combination of email notifications and a mobile formatted site would be a potential alternative to building a transient event notification system and may actually be a worthwhile project on its own. The problem with this approach is that current web standards do not allow a browser-based approach to provide as rich and responsive an experience as a native application. Native applications have the ability to present the information formatted to take advantage of the strengths of a particular device. Additionally, there are internal device features that a native application can leverage. For example, almost all new smart phones have built in GPS (Global Positioning System). A native application can use the GPS to determine if an event is currently above the local horizon. One could even imagine using the Bluetooth capability of the phone to send coordinates to a computer controlled telescope. A native application can also take advantage of local storage and download and store information for events of interest. This information can be viewed at a later time when the device is out of contact with a network. In summary, mobile versions of the transient

repository web sites may be useful, but native applications can provide a much richer experience for the users.

Having decided to create a native application the next decision is the selection of the device. The device should be selected based on ease of development and a large, easy to reach user base that is a good match to the end user demographic being targeted. The platform must also have a good method for distribution of the software to the user.

The iPhone/iPod Touch platform does an excellent job addressing all these criteria. First, the user base is large and growing rapidly. If the application is written to work on both the iPhone and iPod Touch the current installed base is greater than 90 million units. The software development environment is very easy to use and the software API's provided as part of the development kit makes it possible to develop applications quickly. Apple also provides a very convenient method for application distribution in their very successful App store.

Another plus for the iPhone platform is the recent introduction of Push Notifications. This allows iPhone and iPod users to receive rapid notifications of events very soon after they are submitted to the notification system. The push notification system provides a way to send events (at no cost) to any subscribed iPhone or iPod touch user. The notification appears as a pop-up alert on the screen with an optional notification sound and short message. These events can be tied to the application allowing it to quickly download data related to the event as soon as the application is launched.

The Transient Events Application

The Transient Events application on the iPhone provides users with a single, easy to use application for notification and review of transient events from multiple sources. The major goals of the application are listed below:

- Aggregate transient events from multiple sources
- Allow specification of filter parameters so only those events of interest are presented to the user.
- Notify users of new events as they occur
- Provide a quick method to scan all recent events of interest
- Allow the user to easily view detailed information for specific events

The purpose of the application is not to replace the existing web sites which provide very detailed information on each event but rather to summarize the important information and present it to the user on a portable device which they have with them at all times.

The LSST project will not be built and operating for several years, so we have chosen to build a prototype with an existing transient alert survey, the Catalina Realtime Transient Survey (CRTS)[1], which operates on three telescopes, two in the Catalina mountains in Arizona and one in Siding Spring, Australia. The CRTS events are served through event-handling software called Skyalert [2], which is delivering CRTS events to the Transient App. In the future, we hope to deliver other event streams in addition to CRTS.

Application Design from the Users Perspective

The iPhone is a combination of a Unix based computer with full time web access, GPS, orientation sensing, a graphical touch interface and of course, a phone. The challenge when designing an application is to utilize these features to provide information in a compelling way so that the application is used on a regular basis.

Simple to use

A well-designed iPhone application is simple to use and requires zero, or minimal instruction. By using the Apple supplied user interface elements in a manner that is consistent with other iPhone programs and by following the Apple Human Interface Guidelines, programs are often self explanatory. If custom

designed interface elements are used their use should either be obvious or easily discoverable with experimentation. Simplicity also implies limiting features and avoiding too many complex options.

Responsive

The iPhone/iPod Touch platform uses a sub GHz processor for power conservation and it is limited to the available network speed that in the case of the telephone network in remote locations can be quite slow. Care must be taken when writing software for the platform not to allow the system to lock up because of long process steps or because the software is waiting on a network download.

It is not anticipated that the Transient Events App will perform any complex, time consuming processing steps but the data being displayed does need to be downloaded from the network which may, at times be slow. The recommendation for such situations is to download information in the background and update the views dynamically as the information becomes available. If possible, during the download process the user should be allowed to scroll through the completed data and even make selections. If this is not possible the user should be kept informed of the progress with a dynamically update progress indicator.

Responsiveness also requires that the data service be available 100% of the time. Since this application depends on a data repository for its content it is critically important that this information be available, otherwise the user will open the application and be presented with a blank screen.

Ability to use when no network is available

Having stated that the network data must always be available, there will be times when a user will want to use the software in a remote location when there is no network available. The two prime examples are iPod Touch users who only have network connections when in range of a WiFi hotspot and iPhone users who may be at a remote observing location that has neither WiFi nor cell phone service. To make the software useful when there is no network connection the user is allowed to bookmark interesting objects. When an object is bookmarked its overview and detail information are downloaded and stored on the device. If the user launches the application and the device does not detect an available network the user still has the ability to view bookmarked events.

Transient Events App Layout

The Transient Events application is a client application that displays event information from a source database. Upon launch the user is presented with a listing of recent transient events that meet a set of user defined criteria. Figure 1 shows a typical list of events. Each object will be displayed with a title, which is the event type (if characterized, if it has not been characterized then the type will simply state "Unknown"). Underneath the title will be the RA and DEC coordinates of the event and the time of detection in Universal Time. The title will be color coded to allow the user to quickly determine if the event is visible from the currently selected location. If the title is green the object is currently visible, if it is yellow then the object will be visible once it comes above the horizon (or above the minimum elevation angle if this parameter has been set) and if the title is red then the object never rises above the minimum elevation angle at this location. The location can be set using the built in GPS or by selecting a location from a built in list of observatories.



Figure 1. Event Listing

If the user is interested in a particular event they can touch the event taking the user to a detailed display of information. Figure 2 shows an example of the detailed data for a Supernova event. The detail display shows the event ID, coordinates, magnitude, event time and the Azimuth and Elevation coordinates. In addition, a series of images are shown which display the transient nature of the event. The transient area is highlighted with a box. The user can view the reference image and four of the current images by touching the forward and back triangles or simply by swiping the images left and right with a finger. There is also an option for viewing a detailed finder chart web page.

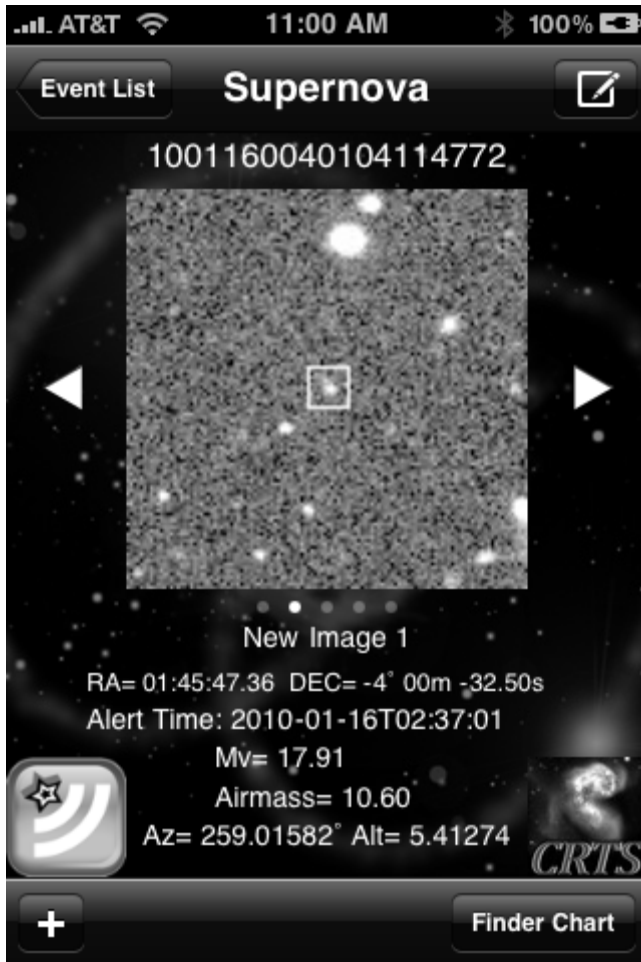


Figure 2. Event Detail

If the object is of interest the user can press the “+” symbol and add this event to a list of bookmarked events. When an event is bookmarked the data will be saved locally allowing the user to quickly retrieve the data in the future even if they are not connected to a network. Bookmarks are accessible by touching the Bookmarks icon shown in Figure 2.

The user can be notified of new events in three ways. If the application is running pressing the “Refresh” button shown in the upper right corner of Figure 1 downloads the latest list of events. Similarly, quitting and restarting the application reloads the latest list of events. The user also has the option of turning on notifications. If they turn on notifications a new event will result in a pop-up message on the screen of their iPhone or iPod even if the Transient Events application is not running. A sample notification is shown in Figure 3. This notification will be accompanied by an alert sound. If the user unlocks their phone the Transient Events application will launch and the detailed information for this event can be viewed immediately.



Figure 3. Transient Notification

Currently the number of transient events is relatively small, a few per night. As the surveys become more numerous and larger in scope the events could number in the 100's/night. If each of these events generates both a notification and an entry in the table, the amount of information will become overwhelming. To help the user manage the amount of information displayed and the number of notifications presented they have the option of adjusting filters to define which events are of interest. Figure 4 shows the filters which are currently available to the users. This page also allows the user to easily turn notifications on and off.

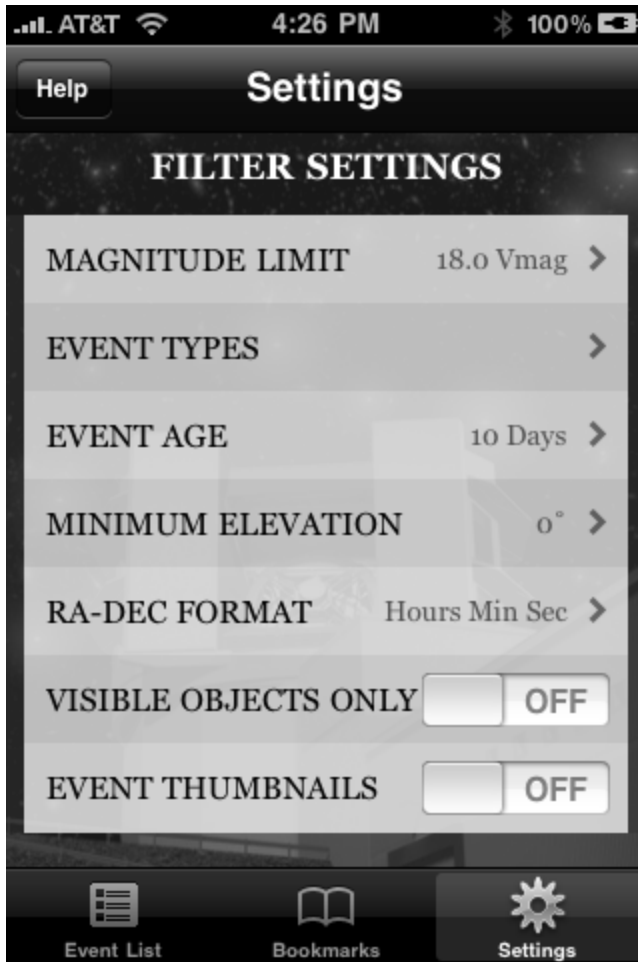
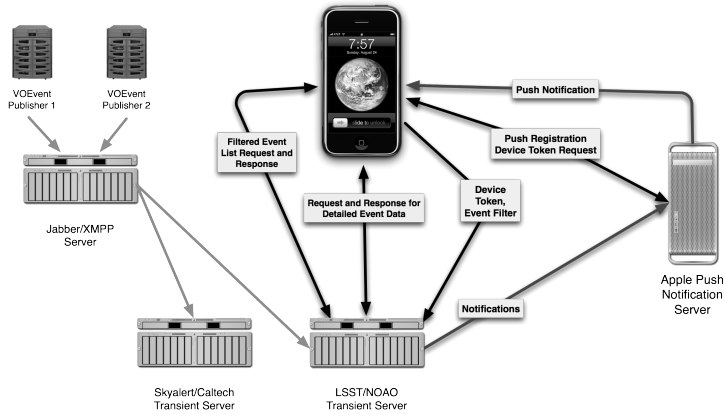


Figure 4. Event Filter Settings

Transient Events Infrastructure

Data is supplied to the Transient Events app through a server that is a duplicate of the Sky Alert system at Caltech. There are thus two redundant servers, each running the Skyalert software [1], and each getting real-time events (via Jabber/XMPP) from a server at Caltech, which is broadcasting events from various event streams. A schematic representation of the infrastructure is displayed in Figure 5. When a transient object is discovered a VOEvent stream is broadcast to all listeners. The Jabber/XMPP server subscribes to these events and then rebroadcasts them to the Skyalert/Caltech Transient Event server, and the LSST/NOAO transient event server which add them to their repository databases. When a user launches Transient Events the app builds a POST query and transmits it to the LSST server. The POST query contains the filter information set by the user consisting of the types of events, maximum magnitude and maximum event age. The server queries the database to retrieve events matching the filter settings and formats them into a JSON string that is returned to the app. The JSON string is parsed by the app into an array of separate events.



Transient Events App with Push Notifications

Figure 5. Transient App Infrastructure

Using http POST for the query allows the app to operate over port 80. This is important as it avoids problems with firewalls. The body of a typical POST query is shown in Figure 6. Each section of the post query is separated by an & and consists of a keyword and a value. Notice that separate entries are required for each “wantedClass”.

```
maxAge=11.000000&maxMagnitude=23.000000&streamNames=CRTS&resumptionToken=0&wantedClass=High+P
roper+Motion+Star&wantedClass=Mira+Variable&wantedClass=Comet
```

Figure 6. Typical POST query

The data from the server is returned in the form of a JSON (JavaScript Object Notation) string. JSON is similar to XML with keys and values but much less verbose. See Figure 7 for an example of the JSON data for a single event. In a JSON string the keys and values are separated by colons, arrays of objects are enclosed in square brackets and key/value pairs are separated by commas.

JSON strings are easy for both the server and the client because there are readily available libraries which can convert data to and from this format. On the client side the SBJSON library is used. This is a category which adds functionality to the NSString object. Using this category the JSON string is parsed with a single method call to JSONValue. This call parses the data into an array (NSArray) of dictionary objects. Each event is an entry in the array and each entry in the array is converted to an NSDictionary of key-value pairs.

```
[
  {
    "pastImgs": [
      "http://nesssi.cacr.caltech.edu/catalina/20100206/1002061210274147147x.html"
    ],
    "freshImageURL": [
      "http://lsst-tes.tuc.noao.edu/tesProxy/catalina/20100206/jpg/1002061210274147147-0001.arch.jpg",
      "http://lsst-tes.tuc.noao.edu/tesProxy/catalina/20100206/jpg/1002061210274147147-0002.arch.jpg",
      "http://lsst-tes.tuc.noao.edu/tesProxy/catalina/20100206/jpg/1002061210274147147-0003.arch.jpg",
      "http://lsst-tes.tuc.noao.edu/tesProxy/catalina/20100206/jpg/1002061210274147147-0004.arch.jpg"
    ],
    "firstIVORN": "ivo://nvo.caltech/voeventnet/catot#1002061210274147147",
    "objectName": [],
    "stream": "CRTS",
    "firstEventTime": "2010-02-06T04:54:44",
    "triggerIVORN": "1002061210274147147",
    "localIVORN": "1002061210274147147",
  }
]
```

```

    "positionalError": 0.0011999999999999999,
    "findingChartImage": [
      ""
    ],
    "furtherInfoURL": [
      "http://www.skyalert.org/events/8799/"
    ],
    "lightCurve": [
      "http://nesssi.cacr.caltech.edu/catalina/20100206/1002061210274147147p.html"
    ],
    "ingestTime": "2010-02-05T21:18:31",
    "magnitude": "17.912001",
    "referenceImageURL": [
      "http://lsst-
tes.tuc.noao.edu/tesProxy/catalina/20100206/jpg/1002061210274147147.master.jpg"
    ],
    "RA": 78.911569999999998,
    "otherImgs": [
      "http://nesssi.cacr.caltech.edu/catalina/20100206/1002061210274147147s.html"
    ],
    "triggerEventTime": "2010-02-06T04:54:44",
    "Dec": 21.059470000000001,
    "type": "Comet",
    "findingChart": [
      "http://voeventnet.caltech.edu/feeds/ATEL/CRTS/1002061210274147147.atel.html"
    ]
  },
  {
    "resumptionToken": "0"
  }
]

```

Figure 7. JSON response for a single event

Notice that the response from the server does not contain any image data. All images are represented by hyperlinks. This keeps the data downloads fast and relatively small. The images seen in the detailed event view are downloaded on demand in a background thread when the page loads. This is important because it keeps the network traffic and the use of client memory low. Using a background thread keeps the application responsive as the images are downloaded. As mentioned earlier these are important design considerations for portable devices.

Depending on the filter settings selected by the user the number of selected events can vary from none to hundreds. To avoid downloading large amounts of data over what could be a slow network connection the server sends event information in groups of 10. At the end of each group the server appends a resumption token. The resumption token is a pointer to the next database record to be searched if more events are requested. The token is set to 0 if there are no more events satisfying the filter parameters. When the application receives a non zero resumption token it displays “More...” in the last row of the table. If the user wishes to review more events they touch this row. When the app detects the touch of “More...” it sends a new POST request to the server and returns the latest resumption token. The server then uses the resumption token to know where to resume the event search and then it returns the next (up to) 10 events. Upon receiving the next batch of events the application appends them to the end of the table.

Producing and Handling Notifications

Producing a notification

Generation of a notification is divided into three separate tasks: registration for notifications, production of notifications and delivery to the device. The communications paths are shown schematically on the right side of Figure 5.

When the Transient Events app launches it contacts an Apple server and requests a device ID. This device ID remains the same for the life of the device. The ID is sent to the LSST Event server along with event filter information as a POST request and looks very similar to the post query shown in Figure 6. The event server adds this information to a database of devices (or updates the database in the event the device already exists). The filter information is parsed and implemented as one of the general trigger expressions that are available in the Skyalert-powered Event server. The device is now registered for push notifications.

Each time an event is received or an event classification is updated, the database of devices is queried to see which devices are interested in this event. For example, if an event is classified as a supernova the server will search through the list of those devices which are registered to receive supernova notifications. The server then checks that the brightness of the event satisfies the event magnitude setting for the device. If the event satisfies all filter criteria the server produces a notification packet containing the device ID, the text of the notification and some additional metadata and sends this packet to the Apple Push Notification Server.

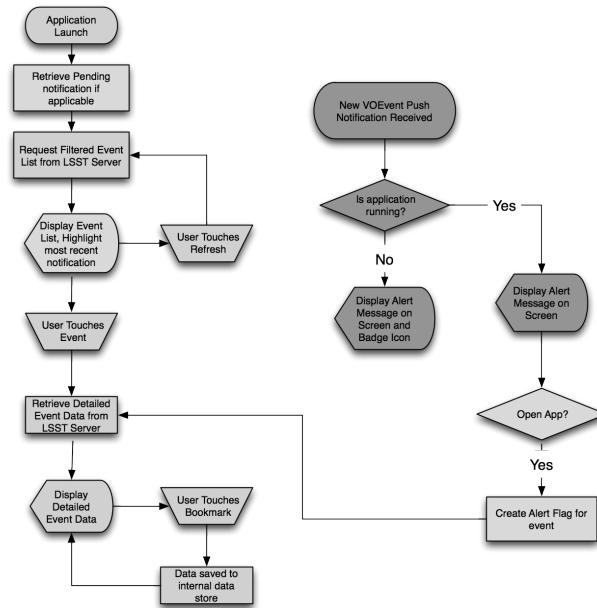
The Apple server handles delivery of the notification. Based on the information received when the device initially requested the device ID the push system knows how to contact the device (which cell system it is on or if it should be located via IP address on a WiFi network). The Apple server then takes responsibility for message delivery.

Handling a notification

When a notification arrives at the device (iPhone or iPod Touch) a notification message is displayed and an alert sound is played. The Transient Event app is also “badged” with the number 1 indicating that there is a notification waiting. This is all handled by the iPhone OS with no intervention by the application and no coding by the programmer.

If the device was in sleep mode when the notification arrived it will wake up and display the notification. If the device is unlocked before it re-enters sleep mode the iPhone OS will launch the application and pass the notification message packet to the application.

If the device is on when the notification arrives the notification is displayed and the notification box gives the user the option of dismissing the box by pressing “Close” or opening the app by pressing “View” (see Figure 3). If the user presses “View” the app is launched and the notification message packet is passed to the application.



Internal iPhone Data Flow

Figure 8 Flowchart of how data and notifications are handled by the application

When the application opens after a notification has been issued it receives an NSDictionary containing the notification message packet information. The application requests the value for the eventID key in the dictionary. This is the triggerIVORN which is displayed at the top of the event detail page (see Figure 2). After receiving and parsing this information the app requests the latest events from the server. As the events arrive their triggerIVORN is compared to the triggerIVORN which was passed in with the notification and the event with the matching triggerIVORN is highlighted with a blinking ALERT as shown in Figure 9.



Figure 9. Display of a Notification Event

It should be noted that an event can generate two notifications. In the above Figure if the user asks to be notified for both Unknown events and Comets a notification will be generated when the event is first entered into the system, at which point it will be classified as Unknown. When the event is later classified as a comet a second notification will be generated.

Server Protocol Support

To support the iPhone Transient App, a special query-response protocol has been built and implemented into the Skyalert server. It is a RESTful http query, with arguments representing the streams from which events are wanted, the brightness threshold, the maximum age, and the classification of the events, with 17 classes to choose from. The results come back in the JSON format, as noted above, including event time, position, and magnitude, as well as:

- **freshImageURL:** an array of links to images representing the new data. It is the difference between this and the reference that is presumably the reason this transient is being disseminated.
- **referenceImageURL:** an array of links to images representing the situation before the discovery of the event, for comparison with fresh images. In addition, there are the following parameters for each transient:
- **furtherInfoURL:** A link back to the Skyalert page for this event, including a portfolio of relevant information that may aid interpretation.
- **findingChart:** an array of HTML page where the finding chart is shown.

- **findingChartImage**: an array of image URLs where the finding chart is shown.
- **pastImgs**: an array of HTML page showing past images from this stream
- **otherImgs**: an array of HTML page showing archival data from other surveys
- **lightCurve**: an array of HTML pages or images showing magnitude history -- the light curve.

Extensibility to future surveys

Implementing the LSST Transient Event Server as the repository for the iPhone Transient Events App will make it relatively simple to extend the application to new surveys. Surveys that provide event data in the same form as the current CRTS stream will be simple to incorporate with only minor enhancements to the application. If the event streams are significantly different the application will check the eventStream value and format the display in a manner that is appropriate for the data.

The event query protocol was made purposely general to enable the Transient App is to be extended to more streams, including those with different telescopes and other wavelengths. However, terms such as “magnitude” and “light-curve” are not appropriate at X-ray or radio wavelengths, although closely related terms replace them. The emphasis on images as the primary means of discovery is perhaps more of a problem in extending this protocol.

Summary

An easy to use iPhone/iPod touch application has been created to provide users with a way to view recent transient events. The application uses a client/server model that minimizes network traffic only downloading data of interest to the user as it is requested. The application incorporates a notification system to inform the users of events of interest in a timely manner. The server/client software have been designed for easy future expansion as additional surveys come on line.

References

- [1] Catalina Realtime Transient Survey, <http://crts.caltech.edu>.
- [2] Skyalert, software for event dissemination and repository, Section 3.1 in this book and also at <http://skyalert.org>.