BlackHoleCam WP1.1 task d deliverable 1

Analyzing trigger software for reuse in fringe testing software for mmVLBI observations

B. Eldering, H. Verkouter
Revision 1.0

May 6, 2015

1 Introduction

As described in the document "eVLBI interface for testing of mmVLBI observations" (see `http://www.jive.nl/jivewiki/lib/exe/fetch.php?media=bhc:documents:blackholecam-autofringetest-0.4.pdf`), finding Very Long Baseline Interferometry (VLBI) fringes at the (sub)mm wavelengths the BlackHoleCam project (BHC) will observe is challenging at best, even on the brightest sources.

The short atmospheric coherence time and strong dependence on current weather conditions necessitates frequent recalibration of the receivers. Small pointing errors cause immediate decorrelation and are solved by very frequent (compared to cm wavelength VLBI) pointing calibration observations. In the BHC consortium, where many oversubscribed mm-observatories participate, it is important to provide quick feedback on the current performance of the array, in order to waste as little of the scarce observing time granted.

This feedback will be provided by a fringe verification system. The outline of this system is described in the document mentioned above ("eVLBI interface for testing of mmVLBI observations"). One part of the fringe verification system is distributing data sample extraction scripts to participation stations. Distributing schedules to stations is a problem that has been solved within the EVN as part of a system to support triggered observations. Therefore the triggered

observation software was identified as a likely useful basis for scheduling fringe verification.

This document describes the trigger software in more detail (Section 3) and investigates how it can be used for scheduling fringe verification (Section 4). An alternative to using the trigger software as a basis is discussed in Section 5. In the last section (Section 6) a recommendation on how to proceed and a corresponding time estimation is made. But first, the next section will give a summary of the requirements for the fringe test scheduling software.

2 Fringe test scheduling software requirements

The minimal requirements for a functional system are:

- A system at a central location which provides the latest fringe test data sample extraction scripts on request.
- A system at the stations which polls the central location for data sample extraction scripts and starts the data sample extraction script parser when a data sample extraction script has been received.

However these are the bare minimum requirements. To be able to properly operate the system, the following features would certainly be helpful:

- One major problem with the model of stations pulling the data sample extraction scripts from the central location is that it is near impossible to monitor the fringe test system of the stations from the central location. To reduce the impact of this problem, the system at the stations should push their logging information to the central location. Even if there is no logging information to send, the system at the station should send a heartbeat signal to the central location. This should allow the central location to see well in advance of the actual fringe test whether the system at the stations is operational.
- The data sample extraction scripts should include a versioning system. This has two major advantages:
  - It is conceivable that a last minute change of the fringe test schedule is only picked up by a subset of the stations. A versioning system would allow us to keep track of which stations have been sending which data.
  - When a station already has the latest script, it would save bandwidth to not resend the complete script. A versioning system can help reduce the amount of data transfers this way.

3 Trigger Software

The trigger software package is based on the client-server model and uses HTTP as a communication protocol.

3.1 Server

The web server is written in python and based on the webpy framework, see `http://webpy.org/`. The web server of the EVN trigger software runs on a machine at JIVE and exposes four methods as URLs through GET and POST requests:

trigger A POST request to the /trigger URL schedules a triggered observation. A parameter for the POST request is a VEX file describing the triggered observation. Only authorized users are allowed to perform this action.

A GET request has a station name as a parameter and returns the VEX file of the currently scheduled triggered observation for that station, if such an observation exists. Otherwise it returns an HTTP 404 error code.

join A GET request to the /join URL is used by stations to indicated whether they accept the currently scheduled triggered observations.

status A GET request to the /status URL returns a JSON dictionary, with the station names as keys and their status regarding the currently scheduled triggered observation as values. If no triggered observation is scheduled, an HTTP 404 error code is returned.

clear A GET request to the /clear URL will remove any scheduled triggered observation.

3.2 Clients

The trigger software package contains three clients.

1. The first client is used to submit a triggered observation in the form of a VEX file using a POST request to the /trigger URL.

2. The second client runs at the stations and polls the server for triggered observations using a GET to the /trigger URL. When a triggered observation is found, the client will present the operator on duty with a choice to accept or reject the triggered observation. The decision is passed on to the server through a GET request of the /join URL. When the triggered observation is accepted, the client will configure the station's Field System according to the supplied VEX file.

3. The third client runs at the correlator. Like the client running at the station, the client at the correlator polls the server for a triggered observation. When a triggered observation is found, the client will show the correlator operator in a GUI the parameters of the triggered observation and the current status of all stations involved. The client keeps track of the status of the stations by polling the server with GET requests on the /status URL. If the correlator operator decides to reject the triggered observation (for example, because not enough stations have accepted the observation), the client will send a GET request to the /clear URL.

4 Preliminary design using HTTP

We can apply the design of the trigger software to the fringe test scheduling software requirements, which results in the following methods to be provided as URLs:

schedule A GET request to the /schedule URL will provide the latest data sample extraction script. The parameters of the request are the station name and the current version of the data sample extraction at the station. If the current version of the data sample extraction script for the requesting station at the central location is more recent, the most recent version will be returned along with its version identifier. If no more recent version of the data sample extraction script is available, the HTTP code 204 (No Content) is returned.

log A POST request to the /log URL will take two parameters. The station name and the log message to append to the logging information repository of the station.

heartbeat A POST request to the /heartbeat URL will take only one parameter: the station name. The time stamp of the POST request will be recorded.

5 Alternative

To help decide whether the trigger software should be reused for the fringe verification software, it is helpful to compare the HTTP communication protocol as used by the trigger software package to another communication protocol. An alternative is to use SSH to transfer the data sample extraction scripts. SSH has a few advantages over HTTP:

- It is arguably easier to set up security for SSH transfers as you can use SSH keys and the UNIX user permission system (as opposed to creating an SSL certificate and implementing an user/password system as done by the trigger software).
- Most Linux versions come with a set of programs based on the SSH protocol installed. For example rsync might be useful to update the data sample extraction scripts to the latest version.

The advantages of HTTP as used by the trigger software package are:

- HTTP is better suited if the data sample extraction scripts have to be dynamically generated, possibly based on parameters passed by the station's client program.
- The idea of the webpy framework is that the programmer has to provide the code to generate the content of the HTTP reply. This makes it easy to insert side effects into the handling of a HTTP request. For example, requests for the latest data sample extraction scripts could be logged, such that it is possible to determine which stations are running which version of the scripts.

- Feedback from external programs like rsync is harder to evaluate than feedback from library modules. The feedback from external programs is limited to a return code and standard output/error, which makes it harder, for example, to extract the root cause of errors, compared to an exception thrown by a library function.

6 Conclusion

HTTP as used by the trigger software seems to be a good candidate for a protocol to be used to transfer data sample extraction scripts. The framework used by the trigger software called webpy makes it easy to dynamically generate the extraction scripts and makes it possible to add side effects to the request for these scripts. This flexibility will allow us to implement more of the logic of the transfer of the data extraction scripts at the server side, which will help during development as deploying a new version of the server software will be easier than deploying a new version of the client software at all stations.

If this flexibility is not required, using SSH as a transfer protocol is a simpler solution. If the flexibility is required, the triggered observation software can be used a as reference. However, using it as the starting point for the implementation of the fringe verification software is not a good idea as, by far, most of the code implements logic to handle triggers. It is easier to copy the few lines of code that deal with setting up the HTTP web server than it is to remove the rest of the code.

The minimal requirements can be implemented using SSH, but for the extra features HTTP will be the preferred communication protocol. For the time estimation required to implement the fringe test scheduling software, the selection of communication protocol is not significant. Both protocols have been used successfully before and most of the effort will probably go to handling any error condition that might appear, making sure both the server at the central location and the client at the station are robust. Although the error conditions are different for HTTP and SSH, handling them properly will require a comparable amount of effort.

The amount of effort required to implement the communications part of the fringe test scheduling system at both the server and client side is estimated at two months.